Blender User Manual

Release 2.78

Blender Community

Jan 23, 2017

Contents

Welcome to the Blender Manual!

This is the manual for the 3D animation software from Blender.org.

CHAPTER 1

Getting Started

1.1 Getting Started

1.1.1 About Blender

Introduction

Welcome to Blender, the free and open source 3D animation suite.

Fig. 1.1: Blender 2.5 with a Big Buck Bunny scene open.

Blender can be used to create 3D visualizations such as still images, video, and real-time interactive video games.

Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process.

It is cross-platform and runs on Linux, macOS, and MS-Windows systems with a small memory and disk footprint. Its interface uses OpenGL to provide a consistent experience across all supported hardware and platforms.

Key Features

- Blender is a fully integrated 3D content creation suite, offering a broad range of essential tools, including *Modeling*, *Rendering*, *Animation*, *Video Editing*, *VFX*, *Compositing*, *Texturing*, *Rigging*, many types of *Simulations*, and *Game Creation*.
- Cross platform, with an OpenGL GUI that is uniform on all major platforms (and customizable with Python scripts).
- High-quality 3D architecture enabling fast and efficient creation work-flow.
- Excellent community support from forums and *IRC*.
- Small executable size, optionally portable.

You can download the latest version of Blender here.

Fig. 1.2: A rendered image being post-processed.

Blender makes it possible to perform a wide range of tasks, and it may seem daunting when first trying to grasp the basics. However, with a bit of motivation and the right learning material, it is possible to familiarize yourself with Blender after a few hours of practice.

This manual is a good start though it serves more as a reference. There are also many online video tutorials from specialized websites, and several books and training DVDs available in the Blender Store and on the Blender Cloud.

Despite everything Blender can do, it remains a tool. Great artists do not create masterpieces by pressing buttons or manipulating brushes, but by learning and practicing subjects such as human anatomy, composition, lighting, animation principles, etc.

3D content creation software such as Blender have the added technical complexity and jargon associated with the underlying technologies. Terms like UV maps, materials, shaders, meshes, and "subsurf" are the mediums of the digital artist, and understanding them, even broadly, will help you to use Blender to its best.

So keep reading this manual, learn the great tool that Blender is, keep your mind open to other artistic and technological areas and you too can become a great artist.

Blender's History

In 1988, Ton Roosendaal co-founded the Dutch animation studio NeoGeo. NeoGeo quickly became the largest 3D animation studio in the Netherlands and one of the leading animation houses in Europe. NeoGeo created award-winning productions (European Corporate Video Awards 1993 and 1995) for large corporate clients such as multinational electronics company Philips. Within NeoGeo Ton was responsible for both art direction and internal software development. After careful deliberation Ton decided that the current in-house 3D toolset for NeoGeo was too old and cumbersome to maintain, and needed to be rewritten from scratch. In 1995 this rewrite began and was destined to become the 3D software creation we all know as Blender. As NeoGeo continued to refine and improve Blender it became apparent to Ton that Blender could be used as a tool for other artists outside of NeoGeo.

In 1998, Ton decided to found a new company called Not a Number (NaN) as a spin-off of NeoGeo to further market and develop Blender. At the core of NaN was a desire to create and distribute a compact, cross-platform 3D application for free. At the time, this was a revolutionary concept as most commercial 3D applications cost thousands of dollars. NaN hoped to bring professional level 3D modeling and animation tools within the reach of the general computing public. NaN's business model involved providing commercial products and services around Blender. In 1999 NaN attended its first SIGGRAPH conference in an effort to more widely promote Blender. Blender's first SIGGRAPH convention was a huge success and gathered a tremendous amount of interest from both the press and attendees. Blender was a hit and its huge potential confirmed!

Following the success of the SIGGRAPH conference in early 2000, NaN secured financing of €4.5M from venture capitalists. This large inflow of cash enabled NaN to rapidly expand its operations. Soon NaN boasted as many as fifty employees working around the world trying to improve and promote Blender. In the summer of 2000, Blender 2.0 was released. This version of Blender added the integration of a game engine to the 3D application. By the end of 2000, the number of users registered on the NaN website surpassed 250,000.

Unfortunately, NaN's ambitions and opportunities did not match the company's capabilities and the market realities of the time. This over-extension resulted in restarting NaN with new investor funding and a smaller company in April 2001. Six months later NaN's first commercial software product, Blender Publisher was launched. This product was targeted at the emerging market of interactive web-based 3D media. Due to disappointing sales and the ongoing difficult economic climate, the new investors decided to shut down all NaN operations. The shutdown also included discontinuing the development of Blender. Although there were clearly shortcomings in the then current version of Blender, such as a complex internal software architecture, unfinished features and a non-standard way of providing the GUI, the enthusiastic support from the user community and customers who had purchased Blender Publisher in the past meant that Ton could not justify leaving Blender to fade into insignificance. Since restarting a company with a sufficiently large team of developers was not feasible, Ton Roosendaal founded the non-profit organization Blender Foundation in March 2002.

The Blender Foundation's primary goal was to find a way to continue developing and promoting Blender as a community-based open source project. In July 2002, Ton managed to get the NaN investors to agree to a unique Blender Foundation plan to attempt to release Blender as open source. The "Free Blender" campaign sought to raise €100,000 so that the Foundation could buy the rights to the Blender source code and intellectual property rights from the NaN investors and subsequently release Blender to the open source community. With an enthusiastic group of volunteers, among them several ex-NaN employees, a fundraising campaign was launched to "Free Blender". To everyone's surprise and delight the campaign reached the €100,000 goal in only seven short weeks. On Sunday, October 13, 2002, Blender was released to the world under the terms of the GNU GPL. Blender development continues to this day driven by a team of dedicated volunteers from around the world led by Blender's original creator, Ton Roosendaal.

Version/Revision Milestones

The start!

- 1.00 – January 1994: Blender in development at animation studio NeoGeo.
- 1.23 – January 1998: SGI version published on the web, IrisGL.
- 1.30 – April 1998: Linux and FreeBSD version, port to OpenGL and X11.
- 1.3x – June 1998: NaN founded.
- 1.4x – September 1998: Sun and Linux Alpha version released.

- 1.50 – November 1998: First Manual published.
- 1.60 – April 1999: C-key (new features behind a lock, $95), MS-Windows version released.
- 1.6x – June 1999: BeOS and PPC version released.
- 1.80 – June 2000: End of C-key, Blender full freeware again.
- 2.00 – August 2000: Interactive 3D and real-time engine.
- 2.10 – December 2000: New engine, physics, and Python.
- 2.20 – August 2001: Character animation system.
- 2.21 – October 2001: Blender Publisher launch.
- 2.2x – December 2001: macOS version.

Blender goes Open Source

- **13 October 2002: Blender goes Open Source, 1st Blender Conference**.
- 2.25 – October 2002: Blender Publisher becomes freely available, and the experimental tree of Blender is created, a coder's playground.
- 2.26 – February 2003: The first truly open source Blender release.
- 2.27 – May 2003: The second open source Blender release.
- 2.28x – July 2003: First of the 2.28x series.
- 2.30 – October 2003: Preview release of the 2.3x UI makeover presented at the 2nd Blender Conference.
- 2.31 – December 2003: Upgrade to stable 2.3x UI project.
- 2.32 – January 2004: A major overhaul of internal rendering capabilities.
- 2.33 – April 2004: Game Engine returns, ambient occlusion, new procedural textures.
- 2.34 – August 2004: Particle interactions, LSCM UV mapping, functional YafRay integration, weighted creases in subdivision surfaces, ramp shaders, full OSA, and many many more.
- 2.35 – November 2004: Another version full of improvements: object hooks, curve deforms and curve tapers, particle duplicators and much more.
- 2.36 – December 2004: A stabilization version, much work behind the scene, normal and displacement mapping improvements.
- 2.37 – June 2005: Transformation tools and widgets, softbodies, force fields, deflections, incremental subdivision surfaces, transparent shadows, and multi-threaded rendering.
- 2.40 – December 2005: Full rework of armature system, shape keys, fur with particles, fluids, and rigid bodies.
- 2.41 – January 2006: Lots of fixes, and some Game Engine features.
- 2.42 – July 2006: The nodes release, array modifier, vector blur, new physics engine, rendering, lip sync, and many other features. This was the release following Project Orange.
- 2.43 – February 2007: Multi-resolution meshes, multi-layer UV textures, multi-layer images and multi-pass rendering and baking, sculpting, retopology, multiple additional mattes, distort and filter nodes, modeling and animation improvements, better painting with multiple brushes, fluid particles, proxy objects, sequencer rewrite, and post-production UV texturing.
- 2.44 – May 2007: The big news, in addition to two new modifiers and re-awakening the 64-bit OS support, was the addition of subsurface scattering, which simulates light scattering beneath the surface of organic and soft objects.
- 2.45 – September 2007: Serious bug fixes, with some performance issues addressed.

- 2.46 – May 2008: The Peach release was the result of a huge effort of over 70 developers providing enhancements to provide hair and fur, a new particle system, enhanced image browsing, cloth, a seamless and non-intrusive physics cache, rendering improvements in reflections, AO, and render baking, a mesh deform modifier for muscles and such, better animation support via armature tools and drawing, skinning, constraints and a colorful Action Editor, and much more. It was the release following Project Peach.
- 2.47 – August 2008: Bugfix release.
- 2.48 – October 2008: The Apricot release, cool GLSL shaders, lights and GE improvements, snap, sky simulator, shrinkwrap modifier, and Python editing improvements. This was the release following Project Apricot.
- 2.49 – June 2009: Node-based textures, armature sketching (called Etch-a-Ton), boolean mesh operation improvements, JPEG2000 support, projection painting for direct transfer of images to models, and a significant Python script catalog. GE enhancements included video textures, where you can play movies in-game, upgrades to the Bullet physics engine, dome (fish-eye) rendering, and more API GE calls made available.

Blender 2.5x – The Recode!

2.5x – From 2009 to August 2011: This series released four pre-version (from Alpha 0 in November 2009 to Beta in July 2010) and three stable versions (from 2.57 - April 2011 to 2.59 - August 2011). It is one of the most important development projects, with a total refactor of the software with new functions, redesign of the internal window manager and event/tool/data handling system, and new Python API. The final version of this project was Blender 2.59 in August 2011.

Video: From Blender 1.60 to 2.50

Blender 2.6x to 2.7x – Improvements & Stabilizing

- 2.60 – October 2011: Internationalization of the UI, improvements in animation system and the GE, vertex weight groups modifiers, 3D audio and video, and bug fixes.
- 2.61 – December 2011: The Cycles renderer was added in trunk, the camera tracker was added, dynamic paint for modifying textures with mesh contact/approximation, the Ocean Sim modifier to simulate ocean and foam, new add-ons, bug fixes, and more extensions added for the Python API.
- 2.62 – February 2012: The Carve library was added to improve boolean operations, support for object tracking was added, the Remesh modifier was added, many improvements in the GE, matrices and vectors in the Python API were improved, new add-ons, and many bug fixes.
- 2.63 – April 2012: Bmesh was merged to trunk with full support for n-sided polygons, sculpt hiding, a panoramic camera for Cycles, mirror ball environment textures and float precision textures, render layer mask layers, ambient occlusion and viewport display of background images and render layers, new import and export add-ons were added, and 150 bug fixes.
- 2.64 – October 2012: Mask editor, improved motion tracker, OpenColorIO, Cycles improvements, sequencer improvements, better mesh tools (Inset and Bevel were improved), new keying nodes, sculpt masking, Collada improvements, new skin modifier, new compositing nodes backend, and many bugs were fixed.
- 2.65 – December 2012: Fire and smoke improvements, anisotropic shader for Cycles, modifier improvements, bevel tool now includes rounding, new add-ons, and over 200 bug fixes.
- 2.66 – February 2013: Dynamic topology, rigid body simulation, improvements in UI and usability (including retina display support), Cycles now supports hair, the bevel tool now supports individual vertex beveling, new *Mesh Cache* modifier and the new *UV Warp* modifier, new SPH particle fluid solver. More than 250 bug fixes.
- 2.67 – May 2013: Freestyle was added, paint system improvements, subsurface scattering for Cycles, Ceres library in the motion tracker, new custom Python nodes, new mesh modeling tools, better support for UTF-8 text and improvements in text editors, new add-ons for 3D printing, over 260 bug fixes.

- 2.68 – July 2013: New and improved modeling tools, three new Cycles nodes, big improvements in the motion tracker, Python scripts and drivers are disabled by default when loading files for security reasons, and over 280 bug fixes.
- 2.69 – October 2013: Even more modeling tools, Cycles improved in many areas, plane tracking is added to the motion tracker, better support for FBX import/export, and over 270 bugs fixed.
- 2.70 – March 2014: Cycles gets basic volumetric support on the CPU, more improvements to the motion tracker, two new modeling modifiers, some UI consistency improvements, and more than 560 bug fixes.
- 2.71 – June 2014: Deformation motion blur and fire/smoke support is added to Cycles, UI pop-ups are now draggable, performance optimizations for sculpting mode, new interpolation types for animation, many improvements to the GE, and over 400 bug fixes.
- 2.72 – October 2014: Cycles gets volume and SSS support on the GPU, pie menus are added and tooltips greatly improved, the intersection modeling tool is added, new sun beam node for the compositor, Freestyle now works with Cycles, texture painting workflow is improved, and more than 220 bug fixes.
- 2.73 – January 2015: Cycles gets improved volumetric support, major upgrade to grease pencil, MS-Windows gets Input Method Editors (IMEs) and general improvements to painting, freestyle, sequencer and add-ons.
- 2.74 – March 2015: Support for custom-normals, viewport compositing and improvements to hair dynamics.
- 2.75 – July 2015: Integrated stereo/multi-view pipeline, corrective smooth modifier and new dependency graph *(enable as a command line option)*.
- 2.76 – November 2015: Pixar OpenSubdiv support, Viewport and File Browser performance boost, node auto-offset, and a text effect strip for the Sequencer.
- 2.77 – March 2016: OpenVDB support for caching for smoke/volumetric simulations, improved cycles Subsurface Scattering, Grease pencil stroke sculpting and improved workflow, and reworked library handling to manage missing and deleted data-blocks.
- **2.78 – September 2016:** Cycles support for spherical stereo images for VR, Grease Pencil works more similar to other 2D drawing softwares, Alembic import and export support, and improvements to Bendy Bones for easier and simpler rigging.

About Free Software and the GPL

When one hears about "free software", the first thing that comes to mind might be "no cost". While this is typically true, the term "free software" as used by the Free Software Foundation (originators of the GNU Project and creators of the GNU General Public License) is intended to mean "free as in freedom" rather than the "no cost" sense (which is usually referred to as "free as in free beer" or *gratis*). Free software in this sense is software which you are free to use, copy, modify, redistribute, with no limit. Contrast this with the licensing of most commercial software packages, where you are allowed to load the software on a single computer, are allowed to make no copies, and never see the source code. Free software allows incredible freedom to the end user. Since the source code is universally available, there are also many more chances for bugs to be caught and fixed.

When a program is licensed under the GNU General Public License (the GPL):

- You have the right to use the program for any purpose.
- You have the right to modify the program and have access to the source codes.
- You have the right to copy and distribute the program.
- You have the right to improve the program, and release your own versions.

In return for these rights, you have some responsibilities if you distribute a GPL'd program, responsibilities that are designed to protect your freedoms and the freedoms of others:

- You must provide a copy of the GPL with the program, so that recipients are aware of their rights under the license.
- You must include the source code or make the source code freely available.
- If you modify the code and distribute the modified version, you must license your modifications available under the GPL (or a compatible license).
- You may not restrict the licensing of the program beyond the terms of the GPL. (you may not turn a GPL'd program into a proprietary product.)

For more on the GPL, check its page on the GNU Project website.

Note: The GPL only applies to the Blender application and **not** the artwork you create with it; for more info see the Blender License.

The Blender Community

Being freely available from the start, even while closed source, helped considerably in Blender's adoption. A large, stable, and active community of users has gathered around Blender since 1998. The community showed its support for Blender in 2002 when they helped raise €100,000 in seven weeks to enable Blender to go Open Source under the GNU GPL.

Who uses Blender?

Blender has a wide variety of tools making it suitable for almost any sort of media production. People and studios around the world use it for hobby projects, commercials, feature films, games and other interactive applications like kiosks, games and scientific research.

Check out the User Stories page on the Blender website for more examples.

Independent Sites

There are several independent websites such as forums, blogs, news and tutorial sites dedicated to Blender.

One of the largest community forums is Blender Artists, where Blender users gather to show off their creations, get feedback, ask and offer help and, in general, discuss Blender.

Support

Blender's community is one of its greatest features, so apart from this user manual, there are many different ways to get support from other users, such as *IRC* and Stack Exchange.

There are also more official sources of support, such as Certified Trainers and the Blender Cloud. If you think you have found an issue with Blender, you can easily report a bug.

More details about support can be found on the support page.

Development

Being open source, some of Blender's development is done by volunteers. Communication between developers is done mostly through three platforms:

- The developer.blender.org system
- Various mailing lists
- The #blendercoders IRC channel (see below)

If you are interested in helping develop Blender, see the Get Involved page.

IRC Channels

For real-time discussion, there are some Blender IRC channels on the Freenode network. You can join these with your favorite IRC client:

- #blender Community support channel.
- #blenderchat For general discussion or off topic chat.
- #blendercoders For developers to discuss Blender development.
- #blenderpython For support for developers using the Python API.
- #gameblender For discussion on issues related to game creation with the GE.
- #blenderwiki For discussion related to Blender's documentation.

Note: If you do not have an IRC client, you can access IRC using webchat.

There also several more Blender-related channels not listed here (e.g. channels for speakers of a particular language). We recommend you search Freenode to see them all.

Other Useful Links

- Blender FAQ (Can I use Blender commercially? What is GPL/GNU? ...)
- Demo and benchmark files
- Developers Ask Us Anything!

1.1.2 Installing Blender

Introduction

Blender is available for download for Linux, macOS and MS-Windows.

Minimum Requirements

Check if your system meets the minimum or recommended requirements.

Always check that the graphics drivers are up to date and that OpenGL is well supported.

Support for other hardware such as graphic tablets and 3D mice are covered later in *Supported Hardware*.

Download Blender

The Blender Foundation distributes Blender in three different ways that you can choose from, to better suit your needs.

The options comprise binary packages for all the supported platforms and the source code. Within the binary packages, you can choose from a stable release or a daily build. The first has the benefit of being more reliable, the latter provides the newest features, as they are developed. Blender is released approximately every three months. You can keep up to date with the newest changes through the release notes.

Latest Stable Release This is a binary distribution of the latest version of Blender. It is considered stable and without regressions.

Daily Builds This is a binary distribution of Blender that is updated daily to include the newest changes in development. These versions are not as thoroughly tested as the stable release, and might break, although they are official and generally not highly experimental.

Build from Source Blender's source is available for reference and installation, with the following advantages:

- Blender is always up to date,
- it allows access to any version or branch where a feature is being developed,
- it can be freely customized.

Note: This is included for completeness, but it is **not** expected that regular users should have to compile their own Blender builds.

Install Blender

The procedure for installing a binary, either the last stable release or a daily build, is the same. Follow the steps for your operative system:

Installing on Linux

Check the *minimum requirements and where to get Blender*, if you have not done so yet.

Specific packages for distributions

Some Linux distributions may have on their repositories a specific package for Blender.

Installing Blender via the distribution's native mechanisms ensures consistency with other packages on the system and may provide other features (given by the package manager), such as listing of packages, update notifications and automatic menu configuration. Be aware, though, that the package may be outdated comparing to the latest official release, or not include some features of Blender. For example, some distributions do not build Blender with CUDA support for licensing reasons.

If there is a specific package for your distribution, you may choose what is preferable and most convenient, otherwise, there is nothing wrong with the official binary on blender.org.

Download from blender.org

Download the Linux version for your architecture and uncompress the file to the desired location (eg. `~/software` or `/usr/local`).

Blender can now be launched by double-clicking the executable.

For easy access, you can configure your system by adding a menu entry or shortcut for Blender and associate and open blend-files with Blender when opening from the file browser. These settings typically belong to the Window Manager (KDE, Gnome, Unity).

Running from the terminal

To run Blender from the terminal without needing to be in the executable directory, add the extracted folder to the environment `PATH`.

Add the following command to `~/.bashrc` or `~/.profile` pointing to the directory with Blender's binary:

```
export PATH=/path/to/blender/directory:$PATH
```

Tip: If you use daily builds and update Blender frequently, you can link or always rename your folder to 'blender' and use this name for the `PATH` environment variable and for keeping the window manager menu up to date.

Avoiding Alt+Mouse Conflict

Many Window Managers default to `Alt-LMB` for moving windows, which is a shortcut that Blender uses to simulate a three button mouse. You can either have this feature disabled *User Preferences* → *Input* → *Emulate 3 Button Mouse* or you can change the Window Manager settings to use the *Meta* key instead (also called *Super* or *Windows key*):

- **KDE:** System Settings > Window Behavior > Window Behavior > Window Actions , Switch 'Alt' for 'Meta' key
- **Unity/Gnome:** enter the following in a command line (effective at next login):

```
gsettings set org.gnome.desktop.wm.preferences mouse-button-modifier '<Super>'
```

Installing on macOS

Check the *Installing Blender* page to find the minimum requirements and where to get Blender (if you have not done so yet).

After downloading Blender for macOS, uncompress the file and drag `blender.app` into the Applications folder.

Tip: Blender does not use the standard macOS menu system so, you will likely have a redundant menu-bar at the top.

To remove it see this post on Macworld, but beware that it is somewhat complex. As an alternative: simply make Blender full screen by `Alt-F11` or by *Info* → *Window* → *Toggle Window Fullscreen*.

Installing on MS-Windows

Check the *minimum requirements and where to get Blender*, if you have not done so yet.

You will also need the Visual C++ 2013 Redistributable Package.

Download the `.zip` or `.msi` for your architecture (64-bit is preferable if your machine supports it).

The `.msi` will run an installer to choose where to place Blender and to configure MS-Windows to have an entry on the menu and to open .blend-files with Blender. Administrator rights are needed to install Blender on your system.

Note: With `.zip` you have to manually extract Blender to the desired folder, where you can double-click the executable to run Blender.

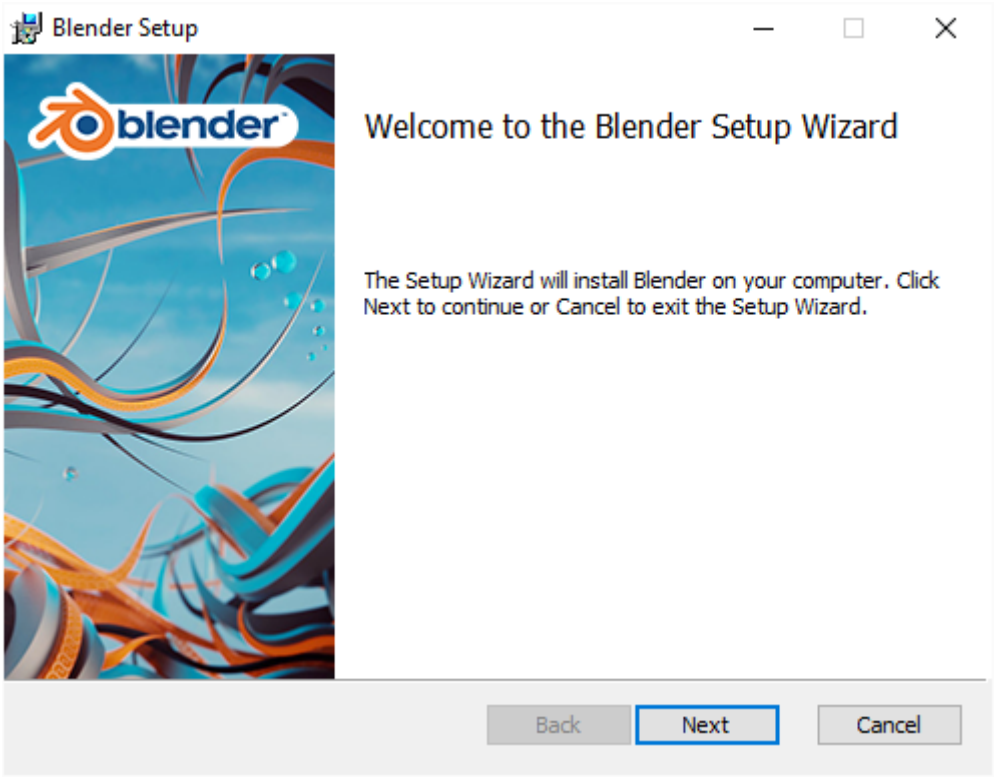


Fig. 1.3: MS-Windows installer.

There is no installer to place Blender on the menu, but there is also no need for administrator rights. With this option, it is possible to have multiple versions of Blender without conflicting, as they are not actually installed on the system.

However, if you want a particular version to be registered with your computer the simply run `blender -r` from the *Command Line*.

Configuring Blender

Introduction

Here are some preferences that you may wish to set initially. The full list and explanation of the user preferences are documented in the section *User Preferences*.

Language

At *File* → *User Preferences* → *System*, enable *International Fonts* to choose the *Language* and what to translate from *Interface*, *Tooltips* and *New Data*.

See *Internationalization* for details.

Input

If you have a compact keyboard without a separate number pad enable *File* → *User Preferences* → *Emulate Numpad*.

If you do not have a middle mouse button you can enable *File* → *User Preferences* → *Emulate 3 Button Mouse*.

See *Input Preferences* for details.

File and Paths

At *File* → *User Preferences* → *File* you can set options such as what external *Image Editor* to use, such as GIMP or Krita, and the Animation Player.

The *Temporary Directory* sets where to store files such as temporary renders and auto-saves.

Tip: The // at the start of each path in Blender means the directory of the currently opened blend-file, used to reference relative paths.

If you trust the source of your blend-files, you can enable *Auto Run Python Scripts*. This option is meant to protect you from malicious Python scripts that someone can include inside a blend-file. This would not happen by accident, and most users leave this option on to automatically run scripts often used in advanced rigs (such as "Rigify" that controls the skeleton of a human rig).

See *File Preferences* for details.

Configuring Peripherals

Displays

Todo.

Multi-Monitor Setup

Fig. 1.4: This is an example of Blender's multi-monitor support.

Input Devices

Blender supports various types of input devices:

- Keyboard (recommended: keyboard with numeric keypad, English layout works best)

- Mouse (recommended: 3 button mouse with scroll wheel)
- NDOF Devices (also known as *3D Mouse*)
- Graphic Tablets

Mice

Mouse Button Emulation

If you do not have a 3 button mouse, you will need to emulate it by checking the option in the *User Preferences*.

The following table shows the combinations used:

3-button Mouse	LMB	MMB	RMB
2-button Mouse	LMB	Alt-LMB	RMB

Keyboards

Numpad Emulation

If you do not have a numeric Numpad on the side of your keyboard, you may want to emulate one (uses the numbers at the top of the keyboard instead, however, removes quick access to layer visibility).

See also:

Read more about *Numpad Emulation* in the *User Preferences*

Non English Keyboards

If you use a keyboard with a non-english keyboard layout, you still may benefit from switching your computer to the UK or US layout as long as you work with Blender.

Note: You can also change the default keymap and default hotkeys from the *User Preferences*, however, this manual assumes you are using the default keymap.

Graphic Tablets

Graphics tablets can be used to provide a more traditional method of controlling the mouse cursor using a pen. This can help to provide a more familiar experience for artists who are used to painting and drawing with similar tools, as well as provide additional controls such as pressure sensitivity.

Note: If you are using a graphic tablet instead of a mouse and pressure sensitivity does not work properly, try to place the mouse pointer in the Blender window and then unplug/replug your graphic tablet. This might help.

3D Mice

3D Mice or NDOF (N-Degrees of Freedom) devices are hardware that you can use to navigate a scene in Blender. Currently only devices made by 3Dconnexion are supported. These devices allow you to explore a scene, as well as *Walk/Fly modes*.

See also:

See *Input Preference* for more information on configuring peripherals.

Configuring Directories

There are three different directories Blender may use, their exact locations are operating system dependent.

LOCAL Location of configuration and runtime data (for self-contained bundle)

USER Location of configuration files (normally in the user's home directory).

SYSTEM Location of runtime data for system wide installation (may be read-only).

For system installations both **SYSTEM** and **USER** directories are needed.

For locally extracted Blender distributions, the user configuration and data runtime data are kept in the same sub-directory, allowing multiple Blender versions to run without conflict, ignoring the **USER** and **SYSTEM** files.

Note: You may need to have the "show hidden files" option checked in your file browser settings.

Platform Dependant Paths

Here are the default locations for each system:

Linux

LOCAL

```
./2.78/
```

USER

```
$HOME/.config/blender/2.78/
```

SYSTEM

```
/usr/share/blender/2.78/
```

Note: The path ./2.78/ is relative to the Blender Executable & used for self-contained bundles distributed by official blender.org builds.

Note: The **USER** path will use `$XDG_CONFIG_HOME` if its set:

```
$XDG_CONFIG_HOME/blender/2.78/
```

---

macOS

LOCAL

```
./2.78/
```

USER

```
/Users/$USER/Library/Application Support/Blender/2.78/
```

SYSTEM

```
/Library/Application Support/Blender/2.78/
```

Note: macOS stores the Blender binary in `./blender.app/Contents/MacOS/blender`, so the local path to data & config is:

```
./blender.app/Contents/MacOS/2.78/
```

MS-Windows

LOCAL

```
.\2.78\.
```

USER

```
C:\Documents and Settings\$USERNAME\AppData\Roaming\Blender Foundation\Blender\2.78\
```

SYSTEM

```
C:\Documents and Settings\All Users\AppData\Roaming\Blender Foundation\Blender\2.78\
```

Path Layout

This is the path layout which is used within the directories described above.

Where `./config/startup.blend` could be ~/.blender/2.78/config/startup.blend for example.

`./autosave/ ...` Autosave blend-file location. (Windows only, temp directory used for other systems).

Search order: `LOCAL`, `USER`.

`./config/ ...` Defaults & session info.

Search order: `LOCAL`, `USER`.

`./config/startup.blend` Default file to load on startup.

`./config/userpref.blend` Default preferences to load on startup.

`./config/bookmarks.txt` File Browser bookmarks.

`./config/recent-files.txt` Recent file menu list.

`./datafiles/ ...` Runtime files.

Search order: `LOCAL`, `USER`, `SYSTEM`.

`./datafiles/locale/{language}/` Static precompiled language files for UI translation.

`./datafiles/icons/*.png` Icon themes for Blender's user interface. (Not currently selectable in the theme preferences).

`./datafiles/brushicons/*.png` Images for each brush.

`./scripts/ ...` Python scripts for the user interface and tools.

Search order: `LOCAL`, `USER`, `SYSTEM`.

`./scripts/addons/*.py` Python add-ons which may be enabled in the user preferences include import/export format support, render engine integration and many handy utilities.

`./scripts/addons/modules/*.py` Modules for add-ons to use (added to Python's sys.path).

`./scripts/addons_contrib/*.py` Another add-ons directory which is used for community maintained add-ons (must be manually created).

`./scripts/addons_contrib/modules/*.py` Modules for `addons_contrib` to use (added to Python's sys.path).

`./scripts/modules/*.py` Python modules containing our core API and utility functions for other scripts to import (added to Python's `sys.path`).

`./scripts/startup/*.py` Scripts which are automatically imported on startup.

./scripts/presets/{preset}/*.py Presets used for storing user defined settings for cloth, render formats etc.

./scripts/templates_py/*.py Example scripts which can be accessed from *Text Editor* → *Templates* → *Python*.

./scripts/templates_osl/*.py Example OSL shaders which can be accessed from *Text Editor* → *Templates* → *Open Shading Language*.

./python/ ... Bundled Python distribution, only necessary when the system Python installation is absent or incompatible.

Search order: `LOCAL`, `SYSTEM`.

Temporary Directory

The temporary directory is used to store various files at runtime (including render layers, physics cache, copy-paste buffer and crash logs).

The temporary directory is selected based on the following priority:

- User Preference (see *File Paths*).
- Environment variables (`TEMP` on MS-Windows, `TMP` & `TMP_DIR` on other platforms).
- The `/tmp/` directory.

1.1.3 Help System

Blender has a range of built-in and web-based help options.

Tooltips

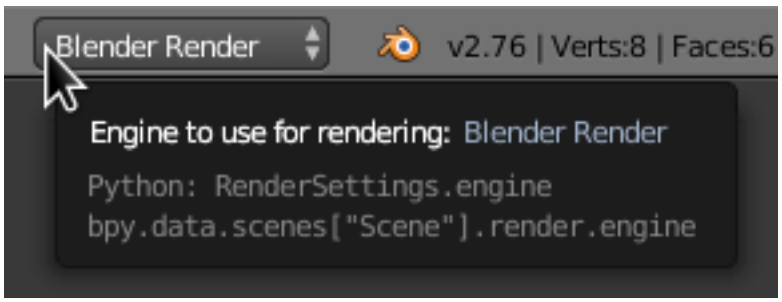


Fig. 1.5: Tooltip of the Render Engine selector in the Info Editor.

When hovering the mouse cursor over a button or setting, after a few instants a tooltip appears.

Elements

The context-sensitive Tooltip might contain some of these elements:

Short Description Related details depending on the control.

Shortcut A keyboard or mouse shortcut associated to the tool.

Value The value of the property.

Python For *scripting* – A Python command associated to the control (usually an operator or property).

Context Sensitive Manual Access

Reference

Mode: All modes
Menu: `RMB`, *Online Manual*
Hotkey: `Alt-F1`

You may want to access help for a tool or area from within Blender.

Use the key-shortcut, or context menu to visit pages from this reference manual within Blender. This opens a webpage relating to the button under the cursor, supporting both tool and value buttons.

Note: We do not currently have 100% coverage, you may see an alert in the info header if some tools do not have a link to the manual.

Other times buttons may link to more general sections of the documentation.

Help Menu

The *Help* menu in the Info Editor header.

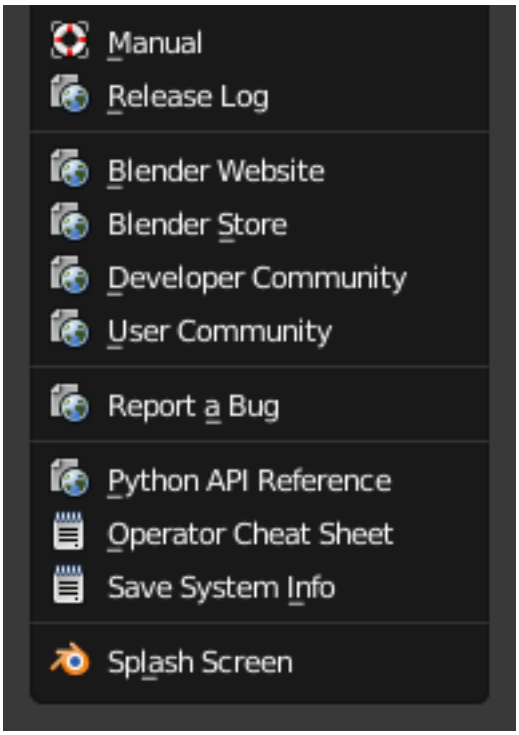


Fig. 1.6: Help Menu.

Web Links

The first options of this menu provide direct links to Blender related websites: The same links can also be found in the *Splash Screen*.

Manual This is a link to the *Official Blender Manual* which you are now reading.

Release Log The release notes on the Web for the changes made for the current Blender version.

Blender Website The blender.org home page.

Blender Store The Blender Store , where you can buy Training DVD's, books, t-shirts and other products.

Developer Community The *blender.org* Get Involved page. This is the launch page for Blender software development, bug tracking, patches and scripts, education and training, documentation development and functionality research.

User Community Lists of many different support venues.

Report a Bug The Blender Bug Tracker (registration needed).

Tip: Browser and Internet Connection

Some forms of Help start up your web browser and access the Blender Foundation's web servers. In order to do this, you must have configured a default web browser for your Operating System, and have a connection to the Internet.

Scripting Reference

Python API Reference Python application programming interface (API) Reference.

Operator Cheat Sheet Creates the `OperatorList.txt` text-block, which you can access in the *Text Editor*. You can also use Blender Search to generate the file. It lists the available Python operators.

Save System Info

Access *Help* → *Save System Info*.

This extracts system information which can be useful to include in bug reports, inspecting the configuration or diagnosing problems.

You will be prompted to save a text file `system-info.txt`.

The text file contains sections:

Blender This section shows you the Blender version, details about the build configuration, and the path in which Blender is running.

Python The Python version you are using, showing the paths of the Python programming language paths.

Directories Paths used for scripts, data-files, presets and temporary files.

Those directories are configured using the *User Preferences* Editor.

OpenGL This section shows the OpenGL version, the name of the manufacturer, and lists the capabilities of your hardware and driver.

Splash Screen

Shows the *Splash Screen*.

CHAPTER 2

Sections

2.1 User Interface

2.1.1 Splash Screen

When starting Blender, the splash screen appears in the center of the window. It contains help options under link and the recently open blend-files. A more detailed description can be found below.

To close the splash screen and start a new project, click anywhere outside the splash screen (but inside the Blender Window) or press `Esc`. The splash screen will disappear revealing the default screen.

To reopen the splash click on the Blender icon in the *Info Editor* header or select *Info Editor* → *Help* → *Splash Screen*.

Title Besides the Blender icon and text, it shows the Blender version. i.e. the current version is 2.78.

Image An image where you can identify package and version.

Date At the top-right corner, you can see the date on that Blender version was compiled.

Hash The Git Hash. This can be useful to give to support personnel, when diagnosing a problem.

Branch Optional branch id.

Interaction Key configuration the same as *User preferences* → *Input*.

Links Links official web pages, the same could be found in the *Help* Menu of the Info Editor. See *Help Menu*.

Recent Your most recently opened blend-files. This gives quick and easy access to your recent projects.

Recover Last Session Blender will try to recover the last session based on temporary files. See *Recovering Data*.

2.1.2 Window System

Introduction

After starting Blender and closing the *Splash Screen* your Blender window should look something similar to the image below. Blender's user interface is consistent across all platforms.

Interface Elements

```
Window → Screen → Areas → Editors → Regions → (Tabs) → Panels → Controls
```

The interface can be customized to match specific tasks using *Screen Layouts*, which can then be named and saved for later use. The default screen is described below.

A screen is organized into one or more *Areas* with each area containing an *Editor*.

Fig. 2.1: Blender Splash Screen.

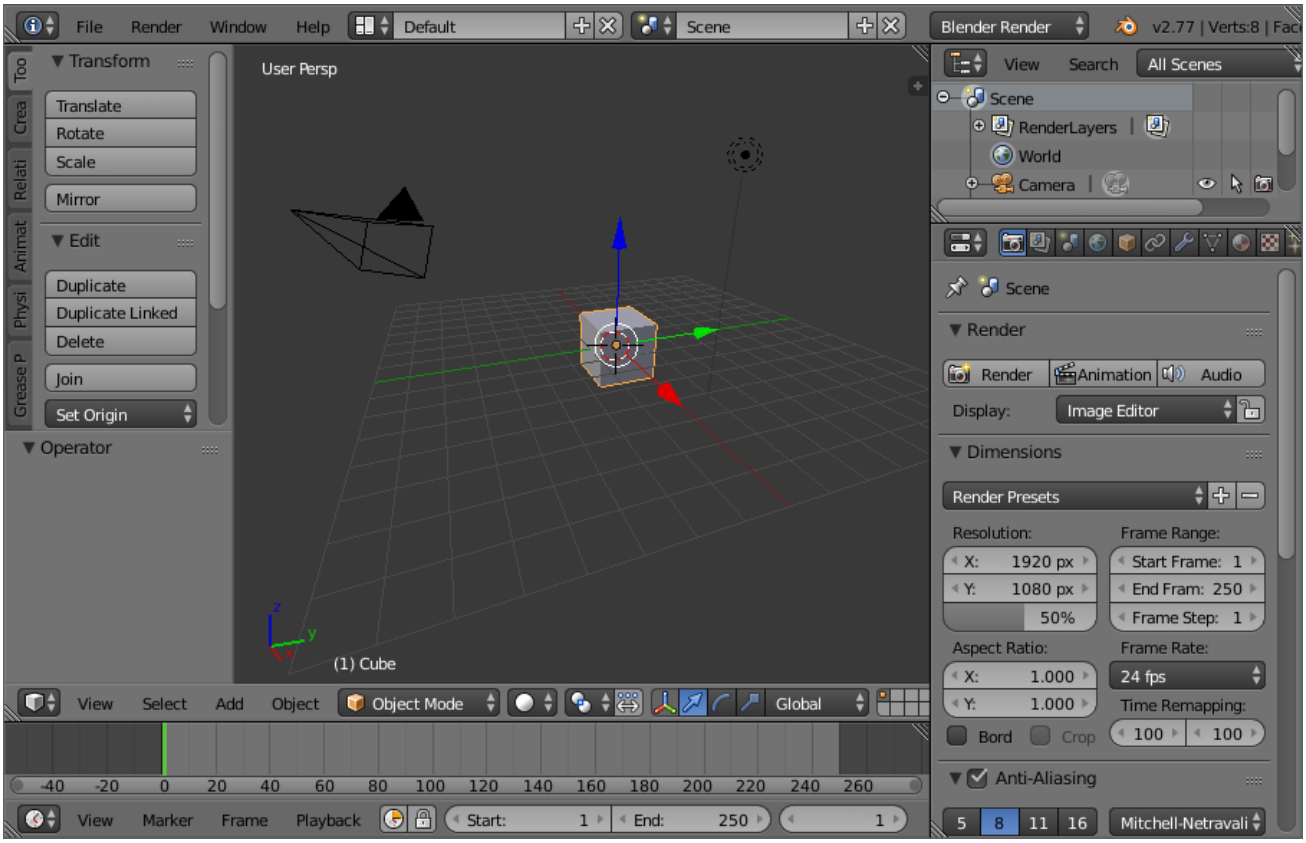


Fig. 2.2: The default startup Blender window.

The Default Screen

By default Blender starts up showing the default screen, which is separated into five areas containing the Editors listed below:

- The Info Editor at the top.
- A large 3D View.
- A Timeline at the bottom.
- An Outliner at the top right.
- A Properties Editor at the bottom right.

Components of an Editor

In general an editor provides a way to view and modify your work through a specific part of Blender. Editors are divided into *Regions*. Regions can have smaller structuring elements like *tabs and panels* with buttons, controls and widgets placed within them.

User Interface Principles

Non Overlapping The UI is designed to allow you to view all relevant options and tools at a glance without pushing or dragging editors around.

Non Blocking Tools and interface options do not block the user from any other parts of Blender. Blender typically does not use pop-up boxes (requiring users to fill in data before running an operation).

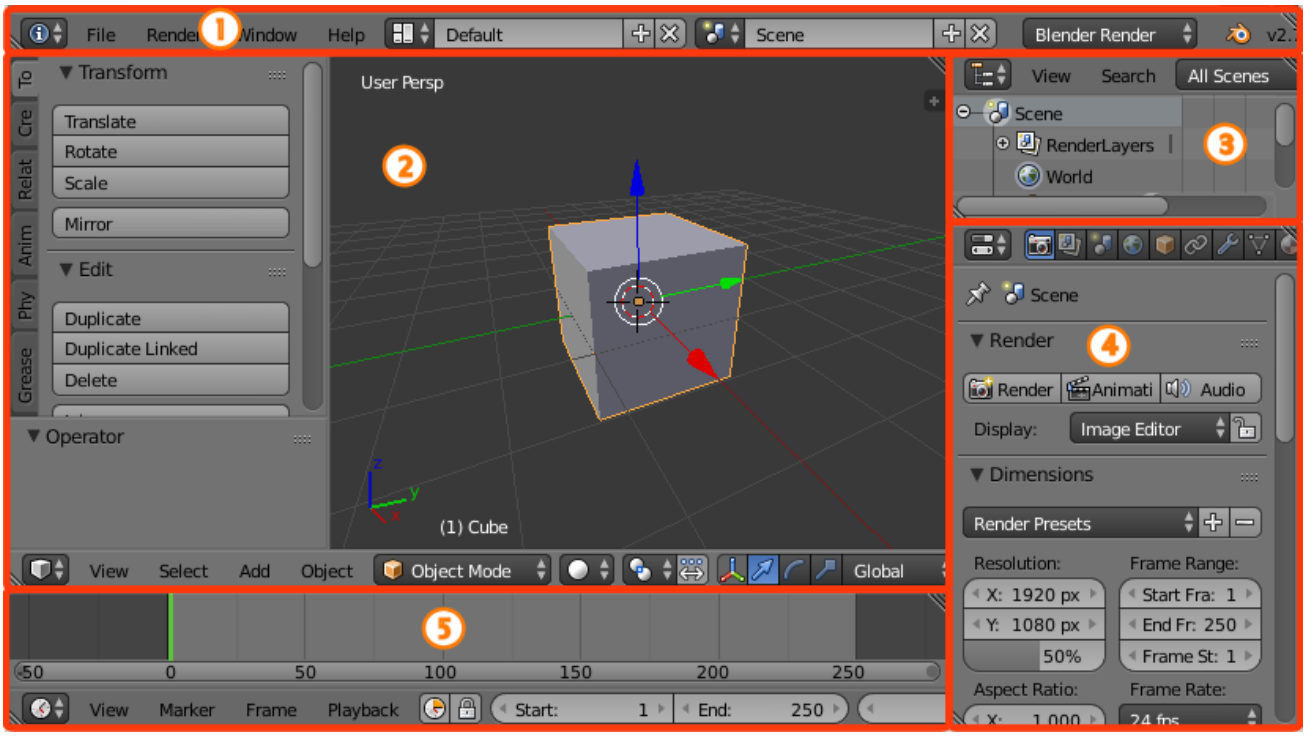


Fig. 2.3: Blender's default Screen Layout with five Editors.
Info (1), 3D View (2), Outliner (3), Properties (4) and Timeline (5).

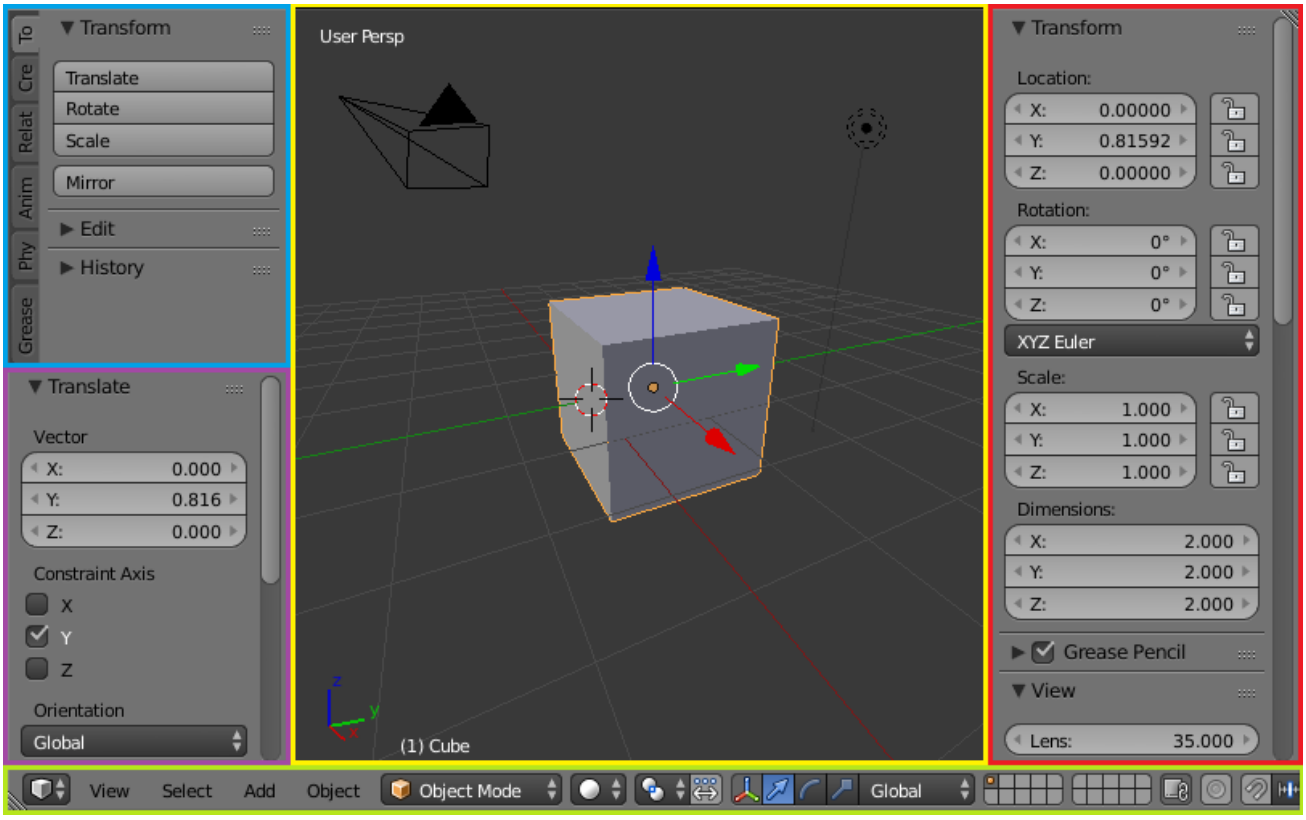


Fig. 2.4: The 3D View editor.
Yellow: Main Region, green: Header, blue: Tool Shelf, purple: Operator Panel, red: Properties Region.

Non Modal Tools Tools can be accessed efficiently without taking time to select between different tools. Many tools use consistent and predictable, mouse and keyboard actions for interaction.

Customization

Blender also makes heavy use of keyboard shortcuts to speed up work. These can also be customized in the *Keymap Editor*.

Theme colors

Blender allows for most of its interface color settings to be changed to suit the needs of the user. If you find that the colors you see on screen do not match those mentioned in the Manual then it could be that your default theme has been altered. Creating a new theme or selecting/altering a pre-existing one can be done by selecting the *User Preferences* editor and clicking on the *Themes* tab.

Screens

Screens are essentially pre-defined window layouts. Blender's flexibility with areas lets you create customized working environments for different tasks such as modeling, animating, and scripting. It is often useful to quickly switch between different environments within the same file. See *area controls* for how to move frame borders, split and consolidate areas.

The Screen data-block menu, that lets you select the layouts, is located in the *Info Editors* header.

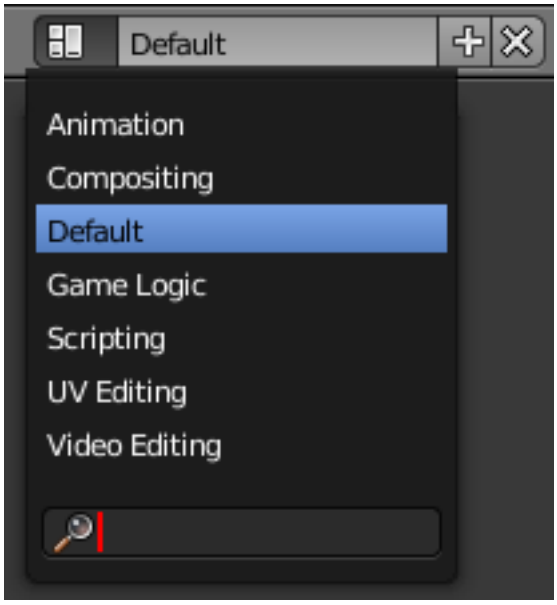


Fig. 2.5 The Screen data-block menu with pop-up.

Controls

Screen Layout A list of available Screen layouts. See *Default Screens*.

Add + Click on the *Add* button and a new frame layout will be created based on your current layout.

Delete X You can delete the selected screen by using the *Delete* button.

Hint: By default, each screen layout 'remembers' the last *scene* it was used on. Selecting a different screen layout will switch to the layout **and** jump to that scene.

Shortcuts

To cycle between screens use `Ctrl-Right` and `Ctrl-Left`.

Warning: On macOS you may need to disable the shortcuts for "Mission Control" in your computer's preferences. These can be found in *System Preferences* → *Keyboard* → *Shortcuts*.

Default Screens

3D View Full A full screen 3D View, used to preview your scene.

Animation Making actors and other objects move about, change shape or color, etc.

Compositing Combining different parts of a scene (e.g. background, actors, special effects) and filter them (e.g. color correction).

Default The default layout used by Blender for new files. Useful for modeling new objects.

Game Logic Planning and programming of games within Blender.

Motion Tracking Used for motion tracking with the movie clip editor.

Scripting Documenting your work and/or writing custom scripts to automate Blender.

UV Editing Flattening a projection of an object mesh in 2D to control how a texture maps to the surface.

Video Editing Cutting and editing of animation sequences.

Save and Override

The screen layouts are saved in the blend-file. When you open a file, enabling the *Load UI* in the file browser indicates that Blender should use the file's screen layouts and overriding the current layout. See *Load UI*.

A custom set of screen layouts can be saved as a part of the *Startup File*.

Additional Layouts

As you become more experienced with Blender, consider adding some other screen layouts to suit your workflow as this will help increase your productivity. Some examples could include:

Modeling Four 3D Views (top, front, side and perspective), Properties editor for Editing.

Lighting 3D Views for moving lights, UV/Image editor for displaying Render Result, Properties editor for rendering and lamp properties and controls.

Materials Properties editor for Material settings, 3D View for selecting objects, Outliner, Library script (if used), Node Editor (if using *Node based materials*).

Painting UV/Image Editor for texture painting image, 3D View for painting directly on object in UV Face Select mode, three mini-3D Views down the side that have background reference pictures set to full strength, Properties editor.

Areas

The application window is always a rectangle on your desktop. It is divided up into a number of re-sizable areas. An area contains the workspace for a particular type of editor, like a 3D View Editor, or an Outliner.

Arranging

Blender uses a novel screen-splitting approach to arrange areas. The idea is that you split up that big application window into any number of smaller (but still rectangular) non-overlapping areas. That way, each area is always fully visible, and it is very easy to work in one area and hop over to work in another.

Changing the Size

You can resize areas by dragging their borders with `LMB`. Simply move your mouse cursor over the border between two areas, until it changes to a double-headed arrow, and then click and drag.

Splitting and Joining

Area Split Widget

In the upper right and lower left corners of an area are the area split widgets, and they look like a little ridged thumb grip. It both splits and combines areas. When you hover over it, your cursor will change to a cross ().

LMB and drag it inward *split* the area. You define the direction of that border by either dragging horizontally or vertically.

In order to *join* two areas LMB click and drag the area splitter outward. They must be the same dimension (width or height) in the direction you wish to join. This is so that the combined area space results in a rectangle.

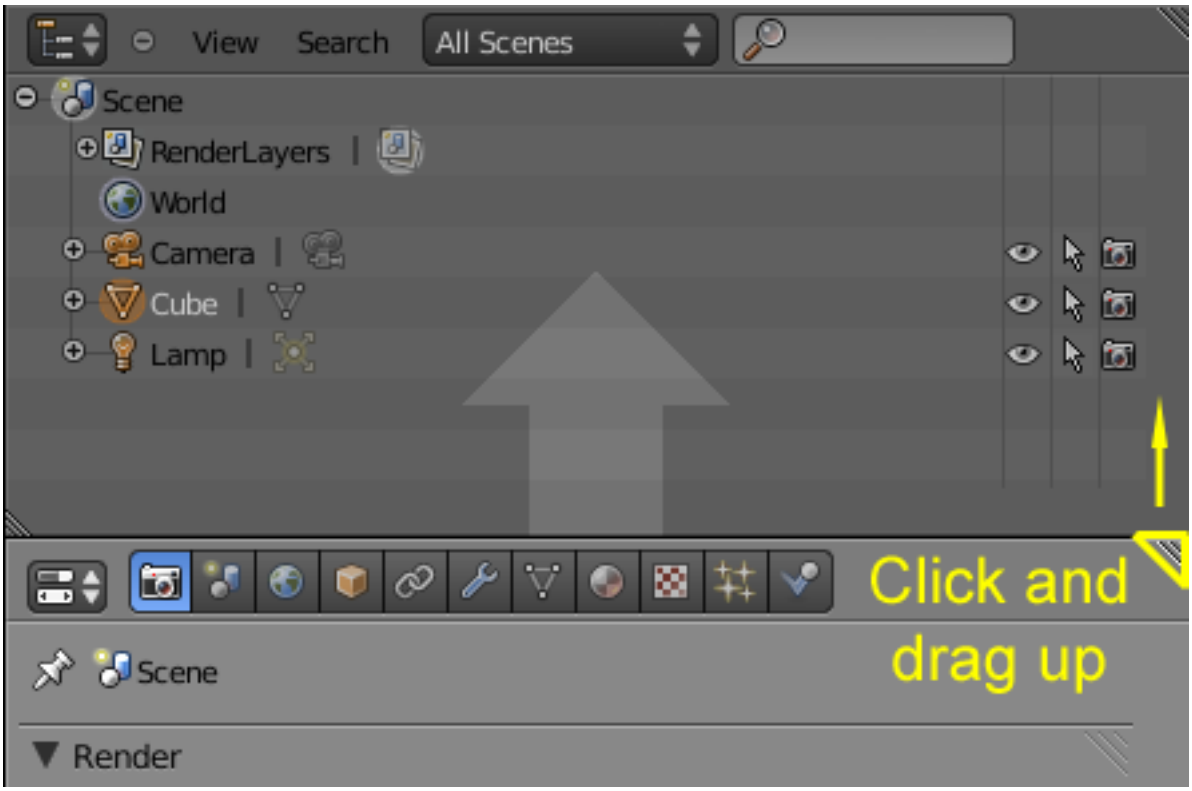


Fig. 2.6: The Properties Editor is being merged "over" the Outliner.

The area that will closed gets a dark overlaid with an arrow. Now you can select the area to be closed by moving the mouse over it.

Release the LMB to complete the join. If you press Esc or RMB before releasing the mouse, the operation will be aborted.

Area Options

RMB on the border opens the *Area Options*.

Split Area Shows a indicator line that lets you select the area and position where to split. Tab switches between vertical/horizontal.

Join Areas Shows the join direction overlay.

Confirm or cancel works as described above.

Swapping Contents

You can swap the contents between two areas with Ctrl-LMB on one of the splitters of the initial area, dragging towards the target area, and releasing the mouse there. The two areas do not need to be side by side, though they must be inside the same window.

Duplicate Area into new Window

Reference

Menu: *View* → *Duplicate Area into new Window*

The new window is a fully functional window, which is part of the same instance of Blender. This can be useful, i.e. if you have multiple monitors.

A new window can be created from *View* → *Duplicate Area into new Window*.

You can also create a new window from an existing area by Shift-LMB on the area splitter widget, then drag slightly.

The window can be closed with the OS *Close Window* button.

Toggle Maximize Area

Reference

Menu: *View* → *Toggle Maximize Area*
Hotkey: Ctrl-Up, Shift-Spacebar

The maximized area fill the whole application window. It contains the Info Editor and the select area.

You can maximize an area with the *View* → *Toggle Maximize Area* menu entry. To return to normal size use again menu entry, or RMB on the editors header and select *Maximize Area* and *Tiled Area* to return. In the Info Editor header the *Back to Previous* button on the right of the menus also returns to tiled areas.

A quicker way to achieve this is to use the shortcuts: Shift-Spacebar, Ctrl-Down or Ctrl-Up to toggle between maximized and normal areas.

Note: The area your mouse is currently hovering over is the one that will be maximized using the keyboard shortcuts.

Toggle Fullscreen Area

Reference

Menu: *View* → *Toggle Full Screen*
Hotkey: `Alt-F10`

The fullscreen area contains only the main region (without the header). To exit the fullscreen use the shortcut `Alt-F10`.

Regions

An Editor is subdivided into regions.

Main Region

At least one region is always visible. It is called the main region and is the most prominent part of the editor.

Each editor has a specific purpose, so the main region and the availability of additional regions are different between editors. See specific documentation about each editor in the *Editors* chapter.

Header

A header is a small horizontal strip with a lighter gray background, which sits either at the top or bottom of the area. All editors have a header acting as a container for menus and commonly used tools. *Menus* and buttons will change with the editor type and the selected object and mode.

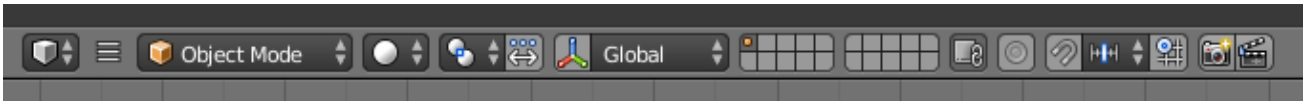


Fig. 2.7: The Header of the 3D View editor.

If you move the mouse over an area, the header of its editor changes to a slightly lighter shade of gray. This means that it is "focused". All hotkeys you press will now affect the contents of this editor. The header can be hidden with `Alt-F9`.

Tool Shelf

The *Tool Shelf* by default on the left side contains the tool settings. `T` toggles the visibility of Tool Shelf Region.

Operator Panel

The Operator panel is a region that is part of the Tool Shelf containing only one panel. In the 3D View it displays the properties of the *last operator* executed and in the File Browser the file import/export options.

Properties Region

The *Properties Region* is by default on the right side. It contains *Panels* with settings of objects within the editor and the editor itself. `N` toggles the visibility of Properties Region.

Arranging

Scrolling

A region can be scrolled vertically and/or horizontally by dragging it with the MMB. If the region has no zoom level, it can be scrolled by using the Wheel, while the mouse hovers over it.

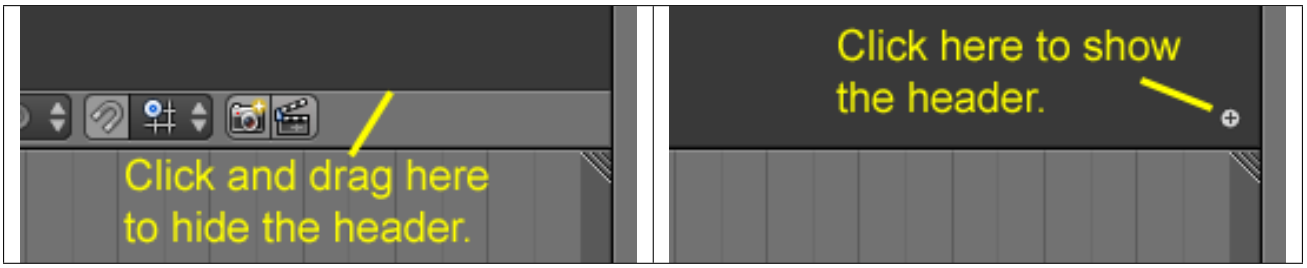
Changing the Size and Hiding

Resizing regions works the same way as *Areas* by dragging its border.

To hide a region scale it down to nothing. A hidden region leaves a little plus sign (see picture). By LMB on this, the region will reappear.

The Tool Shelf and Properties region have a shortcut assigned to toggle between hide and show.

Table 2.1: Hiding and showing the Header.



Position

To flip a region from one side to the opposite press F5, while the Region is under the mouse pointer.

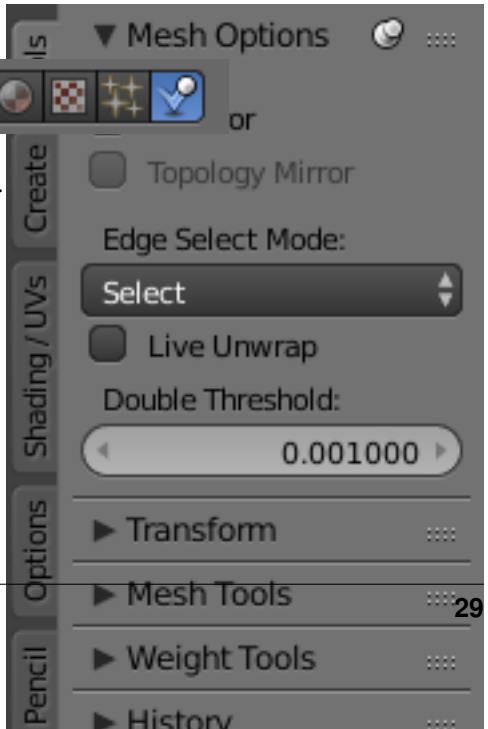
The header can also be flip by RMB on it and select the appropriate item from the pop-up menu. If the header is at the top, the item text will read “Flip to Bottom”, and if the header is at the bottom the item text will read “Flip to Top”.

Tabs & Panels

Tabs

Tabs are overlapping sections in the user-interface. The Tabs header can be vertical (Tool Shelf) or horizontal (Properties Editor, User Preferences).

Fig. 2.9: Horizontal tab header.



Pinning

Often it is desirable to view panels from different tabs at the same time. This has been solved by making panels pinnable.

A pinned panel remains visible regardless of which tab has been selected. You can pin a panel by Shift clicking its header, or by right clicking on the header and choosing *Pin*.

In the image shown to the right, is an example of the *Mesh Options* pinned in the tools tab.

Panels

The smallest organizational unit in the user interface is a panel. Panels can be collapsed to hide its contents. They are used in the *Properties Editor*, but also for example in the *Tool Shelf* and the *Properties region*.

In the image on the right there are three panels: *Transform*, *Edit* and *History*. The *Edit* panel is expanded and the other two panels are collapsed.

Collapsing and expanding

A triangle on the left of the title shows the expanded () and collapsed () state of the panel.

- A click with the `LMB` on the title area of a panel expands or collapses it.
- A `LMB` drag motion over the title area will expand or collapse many at once.
- A `Ctrl-LMB` click on the title area of a specific panel will collapse all other panels and make this the only expanded one.

Position

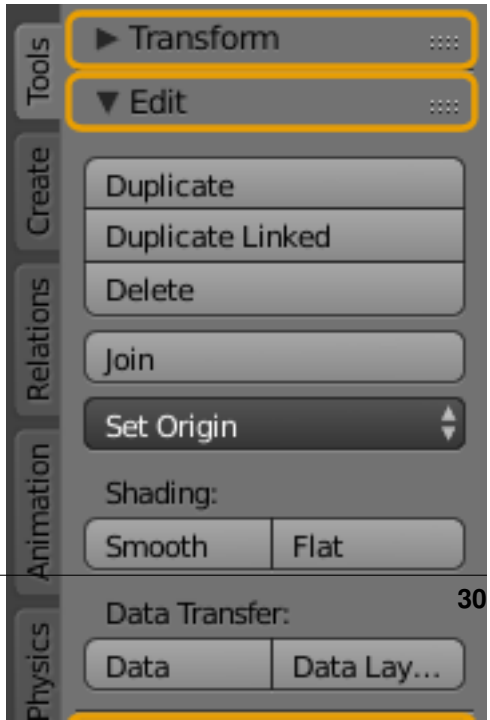
You can change the position of a panel within its region by clicking and dragging it with the `LMB` on the grip widget (::::) in the upper right corner.

Zoom

The zoom factor of a whole region with panels can be changed by `Ctrl-MMB` clicking and moving the mouse anywhere within that region or use the `NumpadPlus` and `NumpadMinus` to zoom in and out the contents. Pressing `Home` (Show All) will reset the zooming at the screen/panel focused by the mouse pointer.

Alignment

The alignment of the panels in the *Properties Editor* can be changed between vertical and horizontal. To do this click with `RMB` somewhere within the main region of the *Properties Editor* and choose either *Horizontal* or *Vertical* from the appearing menu. Keep in mind though that the panels are optimized for vertical alignment.



2.1.3 Interface Controls

Buttons and Controls

Buttons

Operation Buttons

These are buttons that perform an operation when clicked with LMB. They can be identified by their gray color in the default color scheme.

Fig. 2.11: Operation button.

Text Fields & Search Fields

Text fields have a light gray background and a darker outline. They hold text strings, and provide the means to edit it by *standard* text editing. Search fields show a magnifying glass icon on the left side. Start typing in the field to search. Only items which matching text will be shown.

For text fields with an icon and gray pop-up see *Data ID*.

Fig. 2.12: Text and Search.

Color Buttons

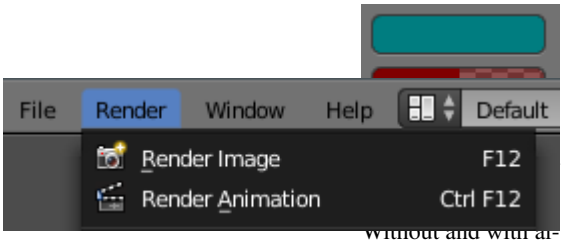
The color button stores a color value shown in its background. LMB color buttons opens the *Color Picker*. Color buttons with an alpha channel are divided in half: On the left the color is shown without an alpha channel and on the right the color with an alpha channel drawn over a checker pattern. Colors can be drag and dropped.

Menus

Blender uses a variety of different menus for accessing options and tools.

Header Menus

Most *headers* exhibit a set of menus, located immediately next to the first *Editor Type* selector. Header menus are used to configure the editor and access tools. All Menu entries show the relevant shortcut keys, if any.



Without and with alpha.

Fig. 2.14: The Info Editor menu buttons.

Collapsing Menus

Sometimes it's helpful to gain some extra horizontal space in the header by collapsing menus, this can be accessed from the header context menu, simply right click on the header and enable it to collapsed.

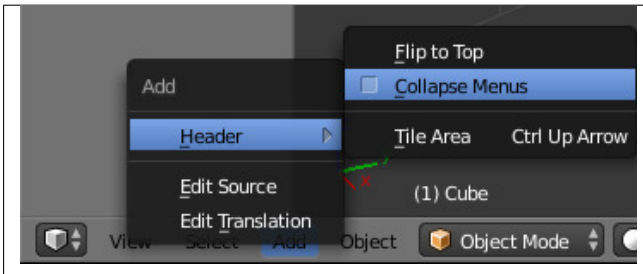


Fig. 2.15: Right-click on any of the header menus.

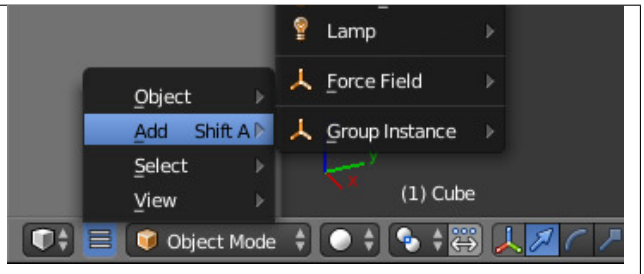


Fig. 2.16: Access the menu from the collapsed icon.

Select Menus

The Select menu or short selector lets you choose between a set of options. They can show a text and/or a icon. The options are shown in a pop-up. The selected option is then shows as active.

Fig. 2.17: The 3D View mode select menu.

Pop-Up Menus

Pop-up menus are overlays. They are spawned by menus showing up and down triangles on the right or after a key input at the mouse position.

If the content is too large to fit on the screen, small indicator triangles appear. When moving the mouse over them scrolls the pop-up.

For example, the *Viewport Shading* button will produce a pop-up menu with the available shading options.

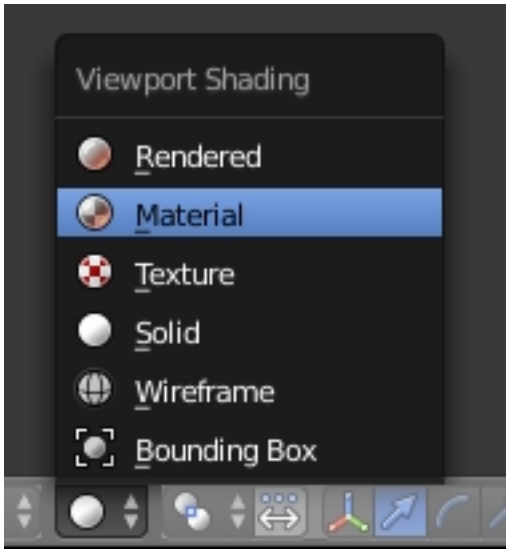


Fig. 2.18: The Viewport Shading pop-up menu.

Mouse selection `LMB` on the desired item.

Numerical selection You can use the number keys or `Numpad` to input an item in the list to select. For example, `Numpad-1` will select the first item and so on.

Pop-ups can be moved by dragging their title.

Shortcuts

- Use `Wheel` while hovering with the mouse.
- Arrow keys can be used to navigate.
- Each menu item has an underlined character which can be pressed to activate it.
- Number keys or numpad can be used to access menu items. (Where `1` is the first menu item, `2` the second... etc. For larger menus `Alt-1` the 11th... up to `Alt-0` the 20th)
- Press `Return` to activate the selected menu item.
- Press `Esc` to cancel the menu, or move the mouse cursor far from the pop-up, or by `LMB` clicking anywhere out of it.

Context Menu

Context menus are pop-ups opened with the `RMB`. Only the common options are listed below.

Single set or gets the value of the button under the mouse pointer. *All* on the other hand includes all combined buttons.

Reset All/Single to Default Value(s) Replaces the current value by the default `Backspace`.

Unset

Copy Data Path For scripting – Copies the Python path of the property, relative to the data-block.

Copy To Selected Copies the property value to the selected objects corresponding property. A use case is if the Properties editor context is pinned.

Add Shortcut Lets you define a keyword or mouse shortcut and associates it with the control. To define the shortcut you must first move the mouse cursor over the button that pops up, and when "Press a key" appears you must press and/or click the desired shortcut.

Change Shortcut Lets you redefine the shortcut.

Remove Shortcut Unlinks the existing shortcut.

Online Manual See *Context Sensitive Manual Access*.

Online Python Reference Context-sensitive access to the Python API Reference.

Edit Source For UI development – Creates a text data-block with the source code associated with the control, in case the control is based on a python script. In the Text Editor it points at the code line where the element is defined.

Edit Translation For UI development – Points at the translation code line.

See also:

Common Shortcuts.

Pie Menus

A pie menu is a menu, whose items are spread radially around the mouse. Pie menus have to be activated in the User Preferences through *Add-ons* → *UI* → *Pie Menus Official*.

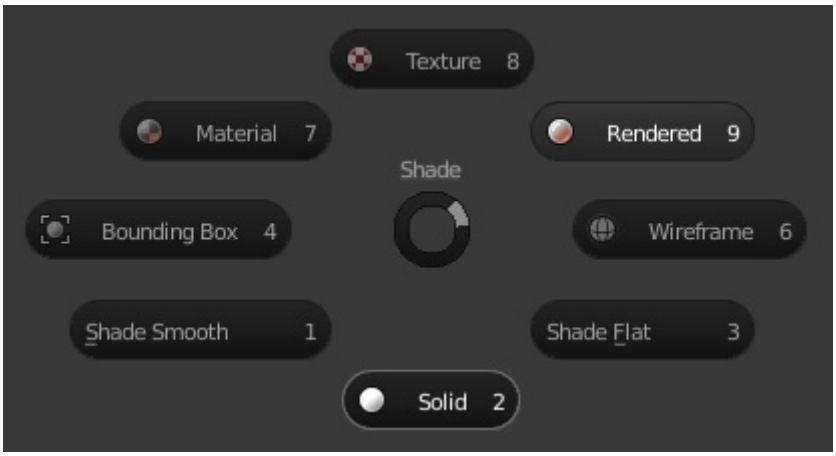


Fig. 2.19: The shade pie menu.

Interaction

The pie menu is spawned by a key press.

3D View

- `Tab` Interaction Mode
- `Z` Shade and solid or smooth shading
- `Q` View directions and perspective or ortho. and camera
- `Tab-Shift-Ctrl` Snapping
- `.` Pivot
- `Ctrl-Space` Manipulator

Movie Clip Editor

- `W` Clip Setup

- Q Marker Setup
- E Tracking
- Shift-S Solving
- Shift-W Scene Reconstruction
- OS-A Playback Operators

Grease Pencil

- D-Q Main tools menu (context sensitive)
- D-W Quick Settings

Releasing the key without moving the mouse will keep the menu open and the user can then move the mouse pointer towards the direction of a pie menu item and select it by clicking. Releasing the key after moving the mouse towards a pie menu item will cause the menu to close and the selected menu item to activate.

An open disc widget at the center of the pie menu shows the current direction of the pie menu. The selected item is also highlighted. A pie menu will only have a valid direction for item selection, if the mouse is touching or extending beyond the disc widget at the center of the menu.

Pie menu items support key accelerators, which are the letters underlined on each menu item. Also number keys can be used to select the items.

If there are sub-pies available, it is indicated by a plus icon.

See *Pie menu settings*.

Toggle & Radio Buttons

Checkboxes & Toggle Buttons

These buttons are used to activate or deactivate options. Use LMB to change its state.

On checkboxes a tick is shown when the option is activated. Toggle buttons are used to set an on/off status. When state is on, they appear like pressed (dark). Clicking this type of button will toggle a state but will not perform any operation. Some toggle buttons have an icon version for each state.

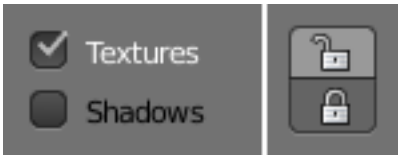


Fig. 2.20: Toggle Buttons.

Dragging

To change many value at once, you can LMB drag over multiple buttons, This works for checkboxes, toggles and to select a radio button value.

Tip: For layer buttons (a type of toggle button) it is often useful to hold Shift at the same time, to set or clear many layers at once.

Radio Buttons

Radio buttons are used to choose from a small selection of "mutually exclusive" options.

Fig. 2.21: Radio Buttons.

Cycling

Use Ctrl-Wheel, while hovering with the mouse over it, to cycle between the the options. Cycling works also for number button and select menus.

Number Buttons

Number buttons hold numeric values.

Number buttons can be identified by the triangles pointing left () and right () on the sides of the button. The second type number sliders have a bar in the background and are used for values in a range, i.e. percentage values. Both types have round corners. In most cases they contain a name and a colon followed by the number. The value can be edited in several ways:

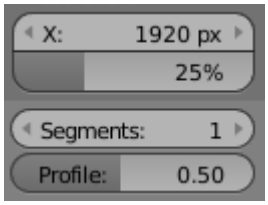


Fig. 2.22: Number buttons.
(grouped or single).

In/Decremental Steps To change the value in steps, click LMB on the small triangles (number button only).

Dragging To change the value in a wider range, hold down LMB and drag the mouse to the left or right. Hold Ctrl to snap to the discrete steps while dragging or Shift for precision input.

Text Input Press LMB or Return to edit the value as a text field.

When entering values by hand, this button works like any other text field:

- Press Return or LMB outside the field to apply the change.
- Press Esc or RMB will cancel the value.
- Press Tab to jump to the next number button.

Press Minus while hovering over the button to negate the value.

Multi-Value Editing

Number buttons can be edited multiple values at once (object scale or render resolution for example). This can be done by clicking on the button and dragging vertically to include buttons above/below. After the vertical motion you can drag from side to side, or release the LMB to type in a value.

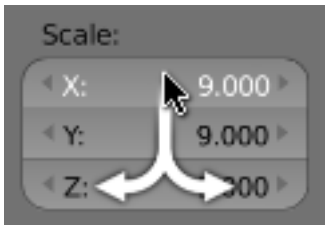


Fig. 2.23: Multi-value editing.

Limits

Most *Number Buttons* has two types of "limits" imposed on them. The first of these is a "soft limit", this means that the property cannot surpassed the value of the "soft limit" without having to LMB and input the value with the Numpad. The second is the "hard limit", this is the value that cannot be surpassed even by LMB and inputing a value.

Expressions

You can also enter expressions such as 3*2 instead of 6. or 5/10+3. Even constants like pi (3.142) or functions like sqrt(2) (square root of 2) may be used.

See also:

These expressions are evaluated by Python; for all available math expressions see: math module reference

Expressions as Drivers

You may want your expression to be re-evaluated after it is entered. Blender supports this using *Drivers* (a feature of the animation system).

Expression beginning with `#`, have a special use. Instead of evaluating the value and discarding the expression, a driver is added to the property with the expression entered.

The expression `#frame` is a quick way to access map a value to the current frame, but more complex expressions are also supported `#fmod(frame,24) / 24` for example.

This is simply a convenient shortcut to add drivers which can also be added via the `RMB` menu.

Units

As well as expressions, you can mix units with numbers; for this to work, units need to be set in the *scene settings*.

To use units simply write either the unit abbreviation or the full name after the value.

Examples of valid units include:

- `1cm`
- `1m 3mm`
- `1m,3mm`
- `2ft`
- `3ft/0.5km`
- `2.2mm + 5' / 3" -2yards`

Note: Some notes about using units:

- Commas are optional.
- You can mix between metric and imperial even though you can only show one at a time.
- Plurals of the names are recognized too, so `meter` and `meters` can both be used.

Eyedropper

The eyedropper (pipette icon) allows you to sample from anywhere in the Blender window. The eyedropper can be used to select different kinds of data:

Color This is the most common usage.

Objects/Object-Data This is used with object buttons such as parent, constraints or modifiers to select an object from the 3D View.

Camera Depth Number buttons effecting distance can also use the eye-dropper.

This is used to set the cameras depth of field so the depth chosen is in focus.

- `E` will activate the eye-dropper while hovering over a button.
- `LMB` dragging will mix the colors you drag over, which can help when sampling noisy imagery.
- `Spacebar` resets and starts mixing the colors again.

Extended Controls

Data-Block Menu

A set of menu buttons used to link *Data-Blocks* to each other. Data-blocks are items like meshes, objects, materials, textures, and so on. If data-blocks are linked the data will be updated across all of the users when edited.

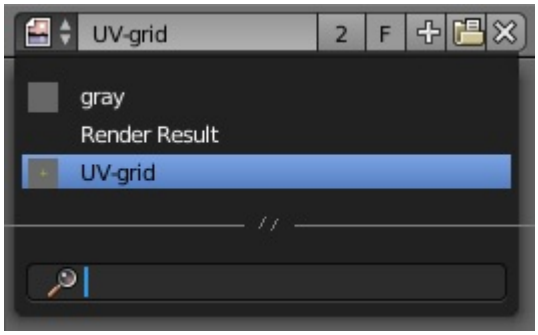


Fig. 2.24: The Data-Block menu with a search input.

Type Shows an icon. Opens up the following pop-up menu.

List A list of data-block available in the current blend-file or link in to select an item from. The menu may show a preview besides the items and a search box to search the items in the list by name.

Name Displays the internal name of the linked Data-Block, which can be edited as a regular text field. If a name already is in assigned Blender will add a digit to the name like ".001".

User count Displays the number of users of the data. Clicking on it to make it a single-user copy, with it linked only to the active object/object's data.

Fake User F Keeps the data-block saved in the blend-file, even if it has no real users.

New/Add + Creates a new data-block or duplicates the current data-block and applies it.

Open file Opens the *File Browser*.

Unpack file *Unpack* the file packed into the current blend-file to external ones.

Unlink data-block X Clears the link.

Sometimes there is a *list* of applied data-blocks (such as a list of materials used on the object).

Data-Block Types

Preview

In the Tool Shelf is a version of the data-block menu with a bigger preview.

Data ID

A Data ID is a text field with a data-block type icon on the left, which opens a gray pop-up. It is used to references data-blocks selected by there name. It only accepts data-blocks of the type specified by the icon. The text field functions as a search field by matching elements in the list. If you type an invalid name, the value will remain unchanged. Click the X button to remove the reference. In some Data IDs there is an *Eyedropper* available through the pipette icon on the right side.

List Views & Presets

List Views

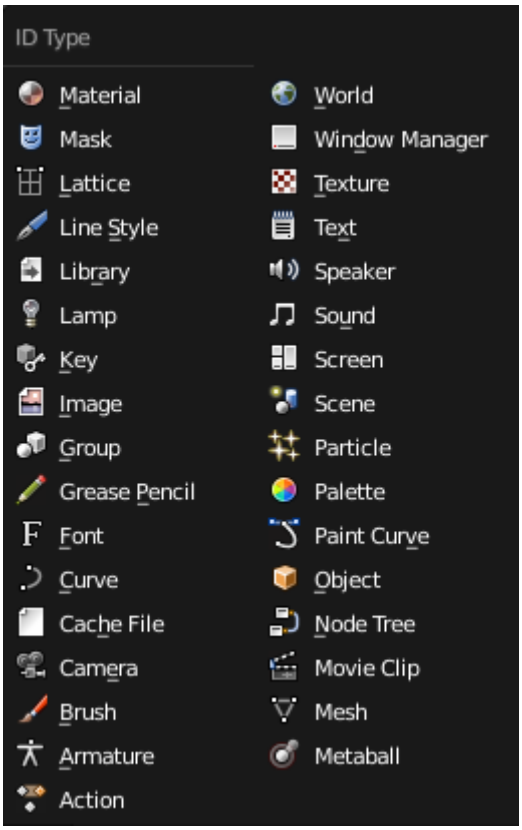


Fig. 2.25: Data-blocks types with their icon.

Fig. 2.26: The Data-Block menu with preview.

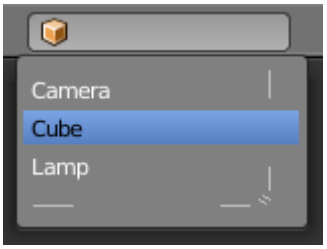


Fig. 2.27: The Data ID.

This control is useful to manage lists of items. They can be found in example in the object data properties.

Select To select an item, `LMB` on it.

Rename By double clicking on an item, you can edit its name via a text field. This can also be achieved by pressing `Ctrl-LMB` over it.

Resize The list view can be resized to show more or fewer items. Hover the mouse over the handle (==) then click and drag the handle to expand or shrink the list.

Filter Click the *Show filtering options* button (+) to toggle filter option buttons.

Search Type part of a list item's name in the filter text field to filter items by part of their name.

Filter Include When the magnifying glass icon has a + sign then only items that match the text will be displayed.

Filter Exclude When the magnifying glass icon has a − sign then only items that do not match text will be displayed.

Sort Sort list items.

Alphabetical This button switches between alphabetical and non-alphabetical ordering.

Inverse Sort objects in ascending or descending order. This also applies to alphabetical sorting, if selected.

One the right of the list view are additional buttons:

Add + Adds a new item.

Remove − To remove the selected item.

Specials The down arrow on dark background opens a pop-up menu with operators context-sensitive to the item type. i.e. copy paste, or operations on all items.

Move Up The button showing an up arrow moves the selected item up one position.

Move Down The down arrow moves the item down.

Presets

Selector A list of available presets. A selection will overrides the included properties.

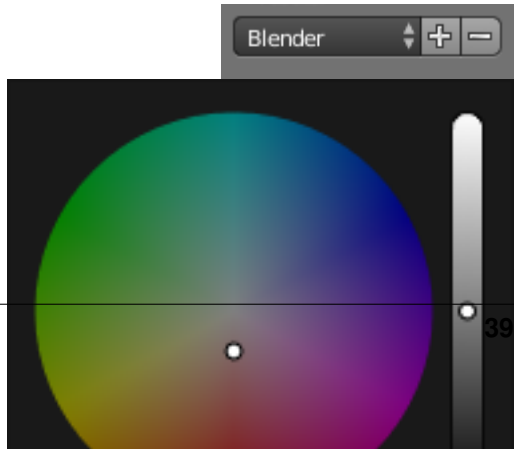
Add + New presets can be added based on the in the preset included properties, which will be saved for later re-use. A pop-up opens where you can set a name after which you can select it from the list and in some cases additional settings.

Remove − Deletes the selected preset.

Specials The down arrow on dark background opens a pop-up menu with operators context-sensitive to the preset type. i.e. copy paste.

Color Picker

The color picker is a pop-up that lets you define a color value.



Color field Lets you pick the the first and second color component. The shape can be selected by the *Types*.

Color slider The slider with a gradient in the background lets you define the third color component. It can also be controlled with the `Wheel`.

Color space Selects the *Color Space* for the number buttons below.

RGB, HSV/HSL, Hex

Color values Blender uses (0 to 1.0) values to express colors for RGB and HSV colors.

Hexadecimal (Hex) values are expressed as RRGGBB. Shorthand hex colors are also supported as RGB, i.e. dark-yellow FFCC00, can be written as FC0.

For operations that are capable of using Alpha, another slider "A" is added.

Eyedropper The *Eyedropper* (pipette icon) can be used to sample a color value from inside the Blender window.

Note: Blender corrects Gamma by default

For more information about how to disable Gamma correction in Blender, see: *Color Management and Exposure* page.

Types

The default color picker type can be selected in the user preferences, see: *System*.

Circle The color values ranging from center to the borders. The center is a mix of the colors.

Square The Borders of the square are the axis for the two color components, with the center on the bottom right.

Table 2.2: Square (HV + S).

Color Ramp Widget

Fig. 2.35: Color-Ramp.

Color Ramps enables the user to specify a range of colors based on color-stops. Color-stops are similar to a mark indicating where exactly the chosen color should be. The interval from each of the color-stops added to the ramp is a result of the color interpolation and chosen interpolation method. The available options for Color Ramps are:

Add + Clicking on this button will add a stop to your custom weight paint map. The stops are added from the last selected stop to the next one, from left to right and they will be placed in the middle of both stops.

Delete – Deletes the selected color-stop from the list.

Flip <–> Flips the color band, inverting the values of the custom weight paint range.

Color Mode Selection of the *color space* used for interpolation.

RGB Blends color by mixing each color channel and combining.

HSV/HSL Blends colors by first converting to HSV or HSL, mixing, then combining again. This has the advantage of maintaining saturation between different hues, where RGB would de-saturate, this allows for a richer gradient.

Interpolation Options Enables the user to choose the types of calculations for the color interpolation for each color stop.

B-Spline Uses a *B-Spline* Interpolation for the color stops.

Cardinal Uses a *Cardinal* Interpolation for the color stops.

Linear Uses a *Linear* Interpolation for the color stops.

Ease Uses a *Ease* Interpolation for the color stops.

Constant Uses a *Constant* Interpolation for the color stops.

Active Color Stop Index of the active color-stop (shown as a dashed line). Allows you to change the active color when colors may be too close to easily select with the cursor.

Position This slider controls the positioning of the selected color stop in the range.

Color Button Opens a color picker for the user to specify color and Alpha for the selected color stop. When a color is using Alpha, the Color button is then divided in two, with the left side showing the base color and the right side showing the color with the alpha value.

Shortcuts

- `LMB` (drag) moves colors.
- `Ctrl-LMB` (click) adds a new control point.

Curve Widget

The purpose of the *Curve Widget* is to allow the user to modify an input (such as an image) in an intuitive manner by smoothly adjusting the values up and down using the curve.

The input values are mapped to the X-axis of the graph, and the Y-axis is mapped to the output values.

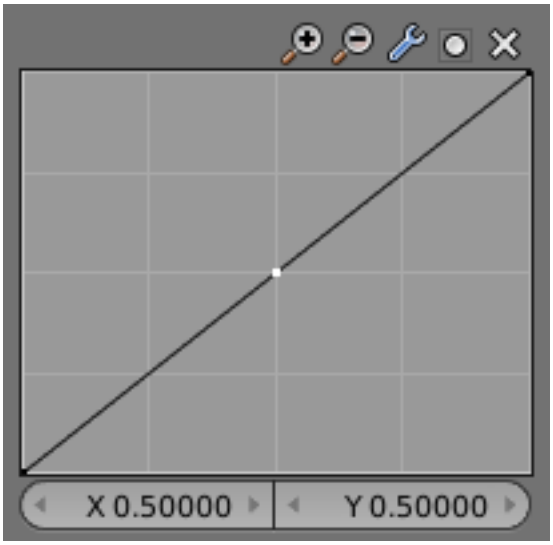


Fig. 2.36: Curve Widget.

Control Points

Like all curves in Blender, the curve of the *Curve Widget* is controlled using *control points*.

By default, there are two control points: one at (0.0, 0.0) and one at (1.0, 1.0), meaning the input is mapped directly to the output (unchanged).

To move a control point Simply click and drag it around.

To add a new control point Click anywhere on the curve where there is not already a control point.

To remove a control point Select it and click the X button at the top right.

Controls

Above the curve graph is a row of controls. These are:

Zoom In Zoom into the center of the graph to show more details and provide more accurate control. To navigate around the curve while zoomed in, click and drag in an empty part of the graph.

Zoom Out Zoom out of the graph to show fewer details and view the graph as a whole. You cannot zoom out further than the clipping borders (see *Clipping* below).

Tools

Reset View Resets the view of the curve.

Vector Handle Vector type of curve point's handle. Breaks the tangent at the curve handle, making it an angle.

Auto Handle Automatic type of curve point's handle.

Extend Horizontal Causes the curve to stay horizontal before the first point and after the last point.

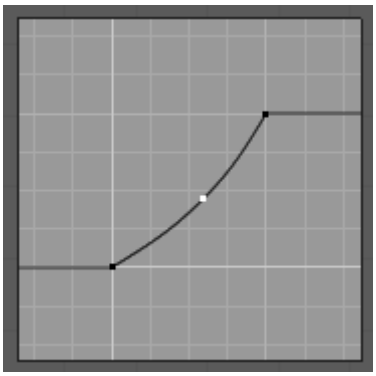


Fig. 2.37: Extend Horizontal.

Extend Extrapolated Causes the curve to extrapolate before the first point and after the last point, based on the shape of the curve.

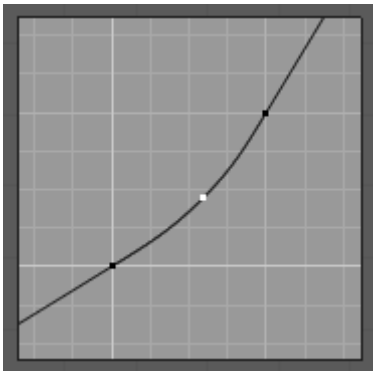


Fig. 2.38: Extend Extrapolate.

Reset Curve Resets the curve in default (removes all points added to the curve).

Clipping

Use Clipping Forces curve points to stay between specified values.

Min X/Y and Max X/Y Set the minimum and maximum bounds of the curve points.

Delete Remove the selected control point. The first and last points cannot be deleted.

X, Y The coordinates of the selected control point.

Operator Search

A menu with access to all Blender commands is available by pressing `Spacebar`. Simply start typing the name of the command you want to refine the list. When the list is sufficiently narrowed, `LMB` on the desired command or navigate with `Down` and `Up`, activate it by pressing `Return`.

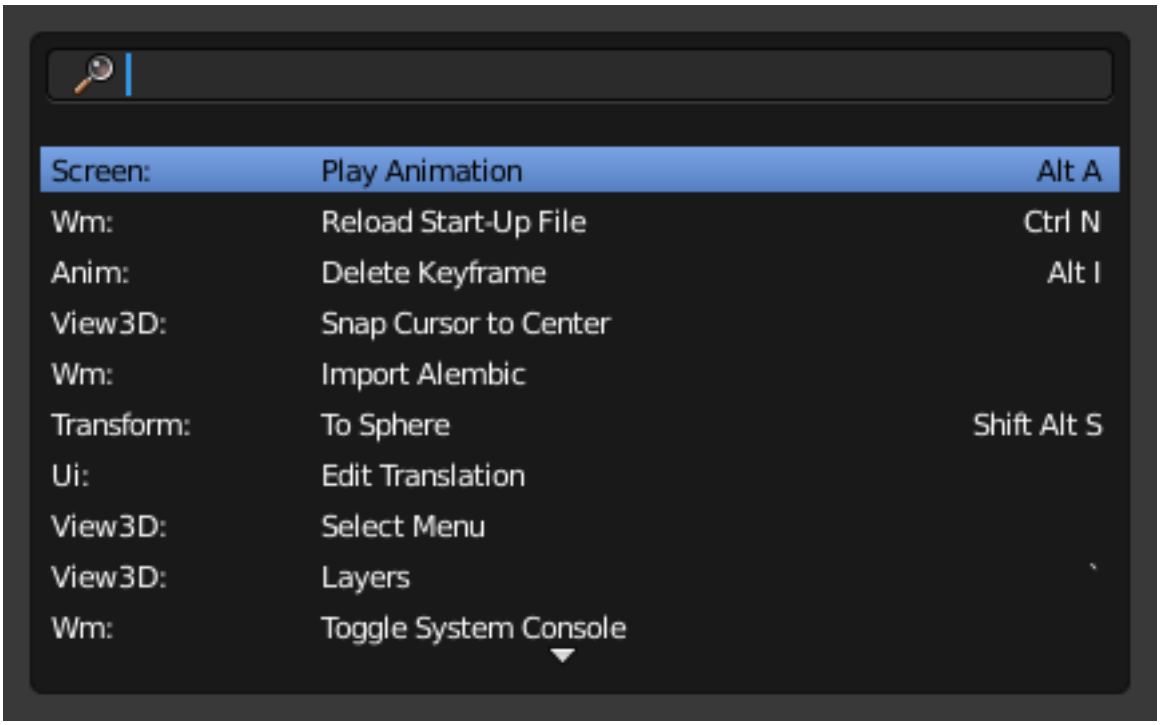


Fig. 2.39: The operator search pop-up.

Common Shortcuts

There are shortcuts shared among many button types.

In Blender the `RMB` (Right Mouse Button) is generally used for Selection and the `LMB` (Left Mouse Button) initiates or confirms actions.

The mouse usage summarized:

`RMB`	To select an item.
`Shift-RMB`	To add more items to the selection.
`LMB`	To perform an action on the selection.

Video: Learn more about Blender's Mouse Button usage.

Note: There are a few corner cases where LMB is used for selection. For example, the *File Browser*.

While Hovering (when the cursor is held over a button).

Properties:

- Ctrl-C – Copy the value of the button.
- Ctrl-V – Paste the value of the button.
- RMB – Open the context menu.
- Backspace – Clears the value (sets to zero or clears a text field).
- Minus – Negate number values (multiply by -1.0).
- Ctrl-Wheel – Changes the value incremental steps.

  For pop-up option menus buttons, this cycles the value.

Animation:

- I – Insert a keyframe.
- Alt-I – Clear the keyframe.
- Alt-Shift-I – Clear all keyframes (removing all F-Curves).
- D – Assign a driver.
- Alt-D – Clear the driver.
- K – Add a Keying Set.
- Alt-K – Clear the Keying Set.

Python Scripting:

- Ctrl-C – Over any *Operation Buttons* copies their Python command into the clipboard.

  This can be used in the Python console or in the text editor when writing scripts.
- Ctrl-Shift-C – Over property buttons copies their data-path for this property (also available from the right-click menu).

  Useful when writing drivers or scripts.
- Ctrl-Alt-Shift-C – Over property buttons copies their *full* data-path for the Data-Block and property.

  Note that in most cases it is best to access values based on the context, instead of by name.

While Dragging

- Ctrl – While dragging snap the discrete steps.
- Shift – Gives precision control over the value.
- Ctrl-Shift – Precise snap. This option will move the object with high precision along with the snapping constraint.

While Editing Text

- Home – Go to the start.
- End – Go to the end.
- Left, Right – Move the cursor a single character.
- Ctrl-Left, Ctrl-Right – Move the cursor an entire word.
- Backspace, Delete – Delete characters.
- Ctrl-Backspace, Ctrl-Delete – Delete words.
- Holding Shift – While moving the cursor selects.
- Ctrl-A – Selects all text.
- Ctrl-C – Copy the selected text.
- Ctrl-X – Cut the selected text.
- Ctrl-V – Paste text at the cursor position.

All Modes

- Esc, RMB – Cancels.
- Return, LMB – Confirms.

2.1.4 Tools

Undo and Redo

The commands listed below will let you roll back an accidental action, redo your last action, or let you choose to recover to a specific point, by picking from a list of recent actions recorded by Blender.

Undo

If you want to undo your last action, just press Ctrl-Z

See also:

Editing Preferences section on undo to change defaults.

Redo

To roll back the Undo action, press Ctrl-Shift-Z.

Redo Last

Redo Last is short for *Redo(ing the) Last (Action)*. F6 after an action will present you a context-sensitive Pop-Up menu based on your last action taken and the Mode and Editor in which Blender is being used.

For example, if your last action was a rotation in *Object Mode*, Blender will show you the last value changed for the angle (see Fig. *Redo last.* left), where you can change your action back completely by typing Numpad0. There are other useful options, based on your action context, and you cannot only Undo actions, but change them completely using the available options.

If you are in *Edit Mode*, Blender will also change its contents based on your last action taken. In our second example (at the right), the last action taken was a Vertex Move; we did a *Scale* on a Face, and, as you can see, the contents of the Pop-Up menu are different, because of your mode (Edit Mode) (See Fig. *Redo last.* right).

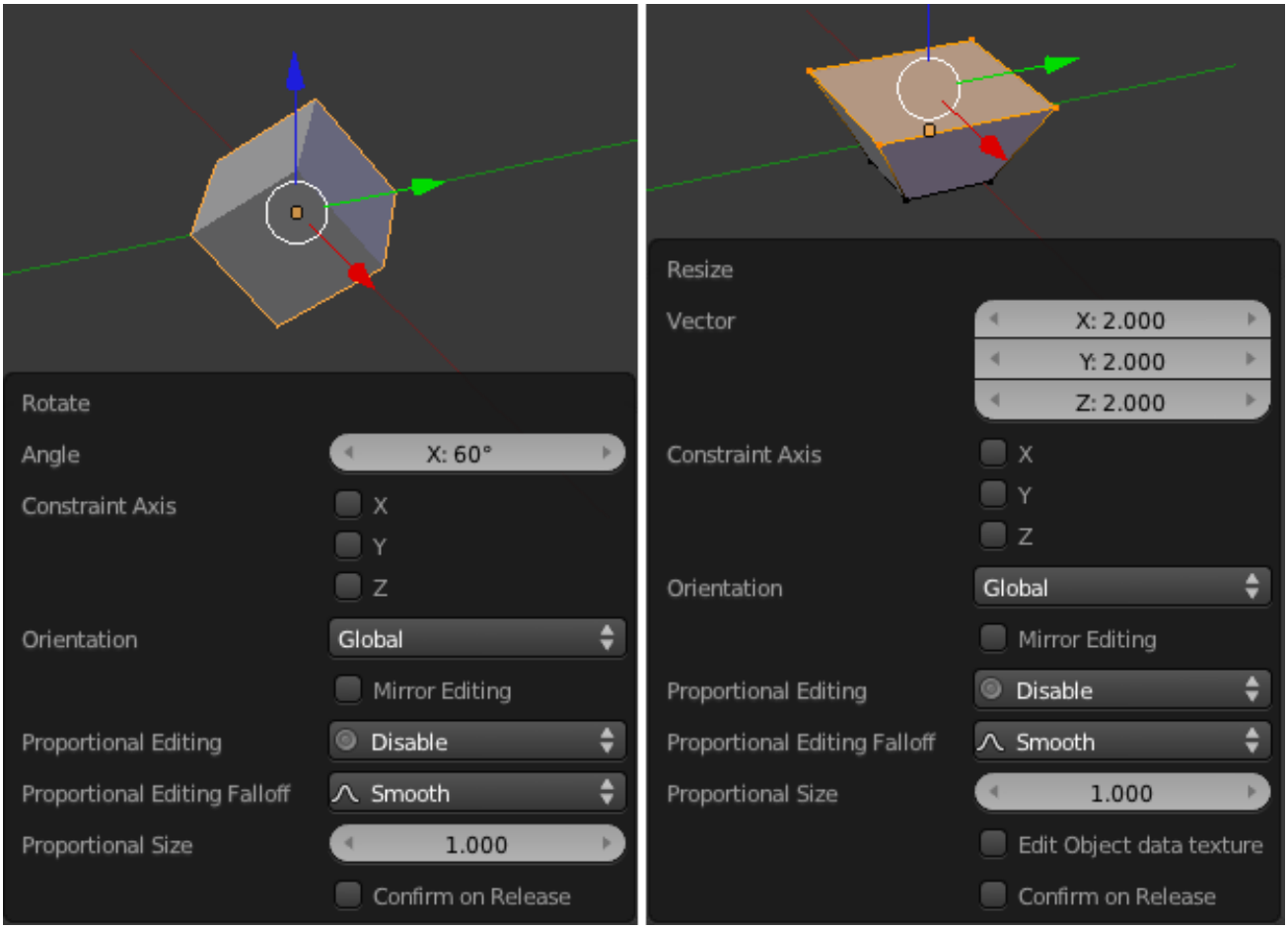


Fig. 2.40: Redo last.
Left Image: Redo Last- Rotation (Object Mode, 60 degrees), Right Image: Redo Last- Scale (Edit Mode, Resize face)

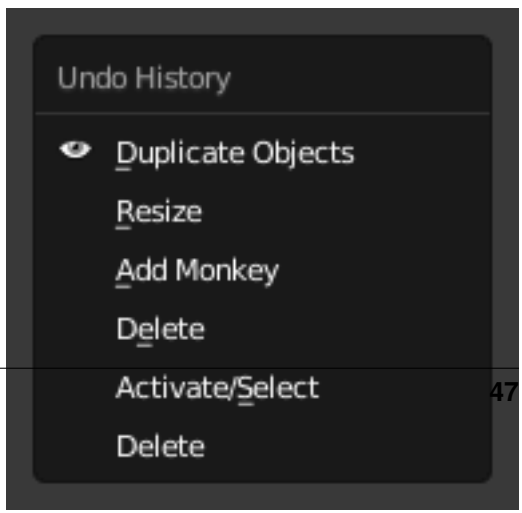
Tip: Operations using Redo Last

Some operations produce particularly useful results if you tweak their parameters with the `F6` Menu. Take, for example, adding a Circle. If you reduce the Vertex count to three, you get a perfect equilateral triangle.

Undo History

There is also an Undo History of your actions, recorded by Blender. You can access the history with `Ctrl-Alt-Z`.

Rolling back actions using the *Undo History* feature will take you back to the action you choose. Much like how you can alternate between going backward in time with `Ctrl-Z` and then forward with `Ctrl-Shift-Z`, you can hop around on the Undo timeline as much as you want as long as you do not make a new change. Once you do make a new change, the Undo History is truncated at that point.



Repeat Last

The Repeat Last feature will repeat your last action when you press Shift-R.

In the example Images below, we duplicated a *Monkey* mesh, and then we moved the Object a bit. Using repeat Shift-R, the *Monkey* was also duplicated and moved.

Fig. 2.42: Suzanne.

Fig. 2.43: After a Shift-D and move.

Fig. 2.44: After a Shift-R.

Repeat History

The *Repeat History* feature will present you a list of the last repeated actions, and you can choose the actions you want to repeat. It works in the same way as the Undo History, explained above, but the list contains only repeated actions. To access Repeat History, use F3.

Note: Blender uses two separate Histories, one dedicated for the *Edit Mode*, and one dedicated for the *Object Mode*.

Important: When you quit Blender, the complete list of user actions will be lost, even if you save your file before quitting.

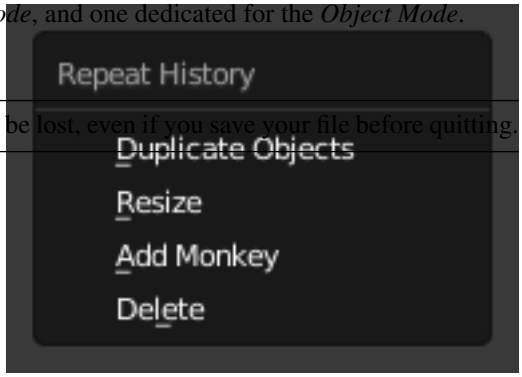


Fig. 2.45: The Repeat History Menu.

See also:

Troubleshooting section on *Recovering your lost work*

Ruler and Protractor

The ruler can be accessed from the Tool Shelf, once activated you can use the ruler to measure lengths and angles in the scene.

Usage

Here are common steps for using the ruler:

1. Activate the Ruler from the Tool Shelf.
2. Click and drag in the view-port to define the initial start/end point for the ruler.
3. Orbit the view and click on either end of the ruler to re-position it. Holding Ctrl enables snap to elements.
4. Click on the middle to measure angles.
5. Press Return to store the ruler for later use or Esc to cancel.

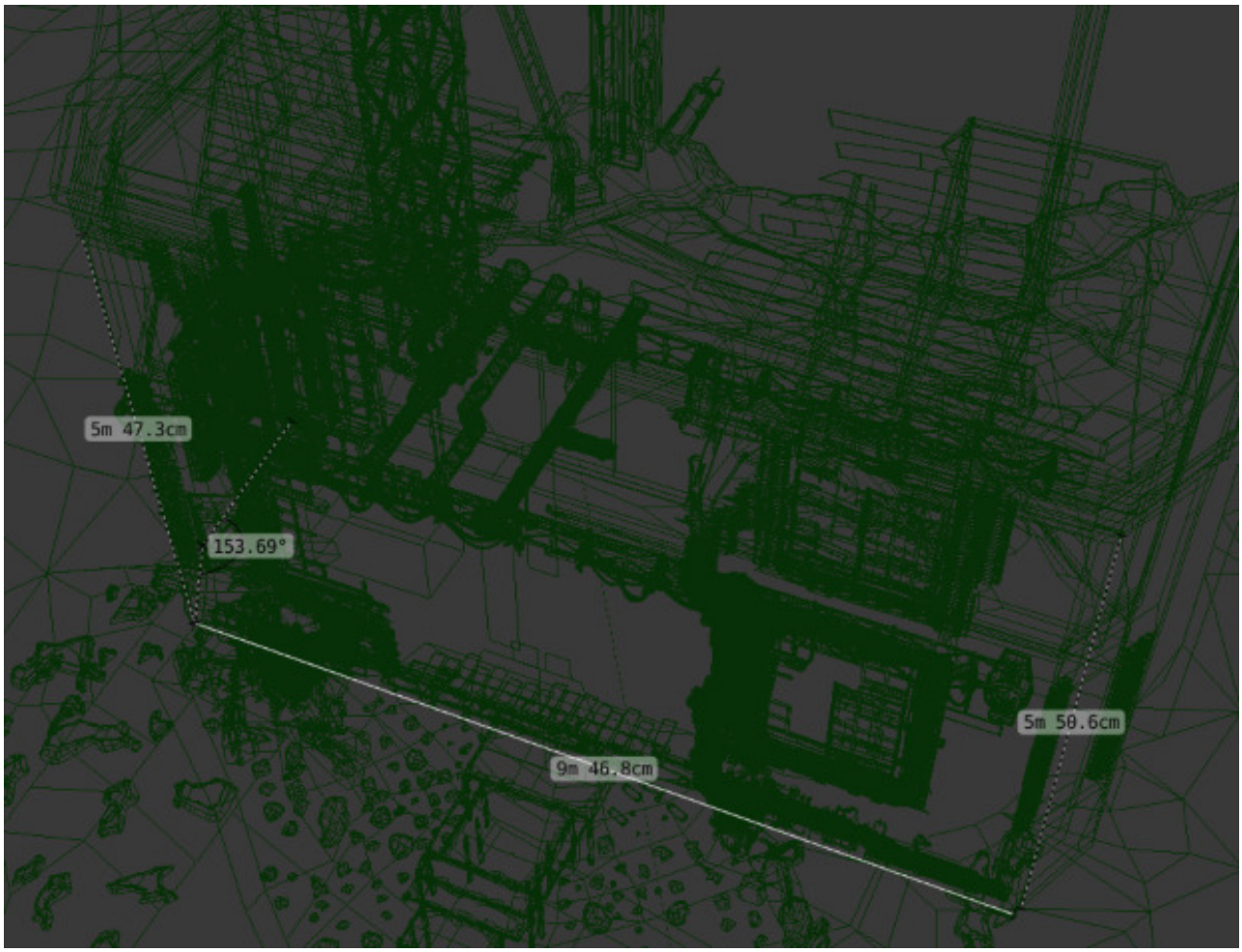


Fig. 2.46: Example of the ruler and protractor.

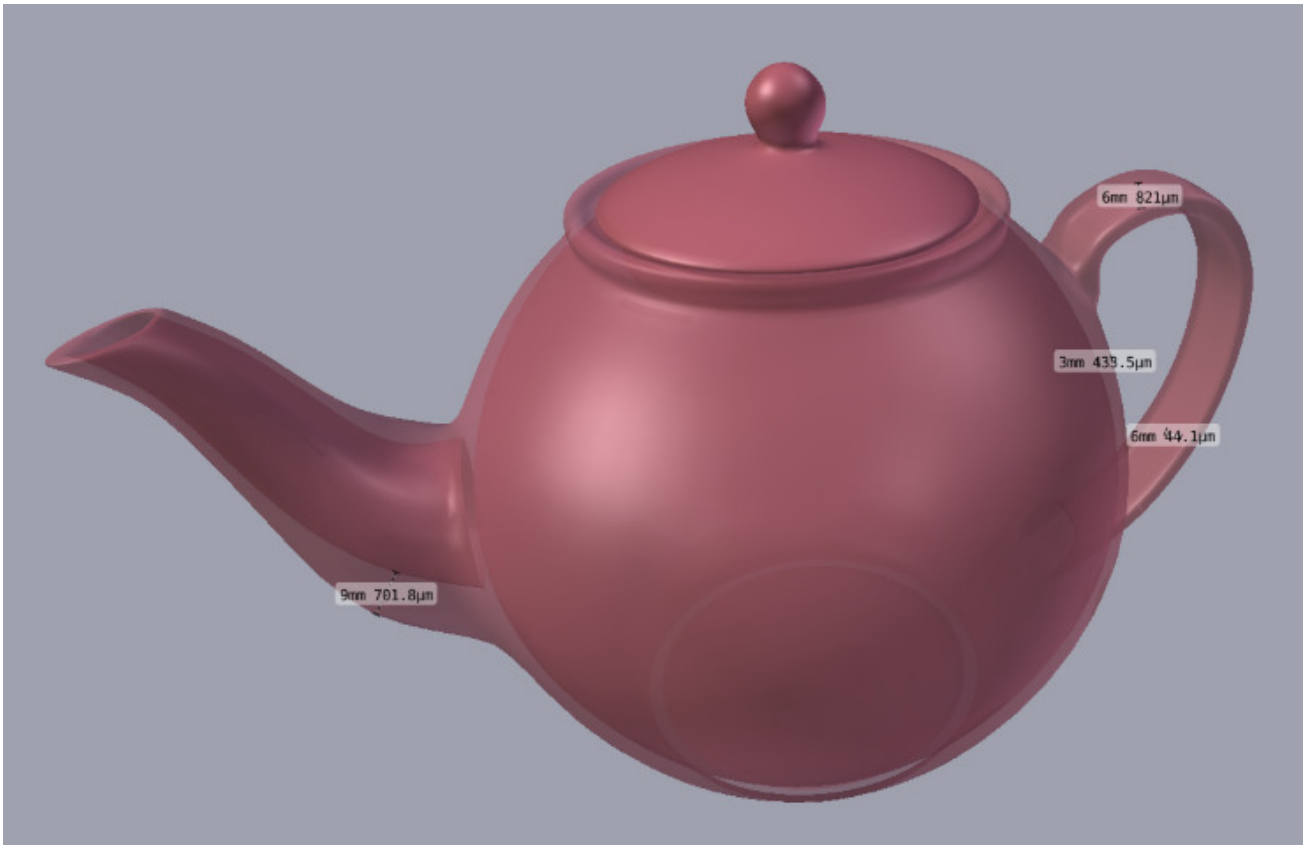
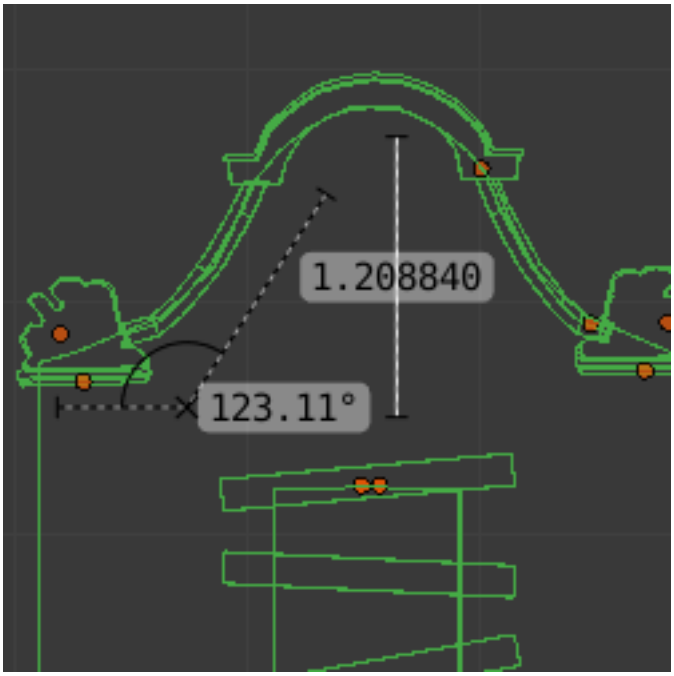


Fig. 2.47: Example using the ruler to measure thickness.



Note: Editing operations can be used while the ruler is running, however, tools like the knife cannot be used at the same time.

Note: Unit settings and scale from the scene are used for displaying dimensions.

Shortcuts

- `Ctrl-LMB` Adds new ruler.
- `LMB` Drag end-points to place them, Hold Ctrl to snap, Hold Shift to measure thickness.
- `LMB` Drag center-point to measure angles, drag out of the view to convert back to a ruler.
- `Delete` Deletes the ruler.
- `Ctrl-C` Copies the rulers value to the clipboard.
- `Esc` Exit tool.
- `Return` Saves the rulers for the next time the tool is activated.

Grease Pencil

Introduction

Years ago people needed a way to quickly draw on their monitors, they did this with a tool called a grease pencil. This is especially helpful for animators who need to add notes directly on their screen. However, not everyone wants to draw on their monitors. So a digital version was made, also called a grease pencil.

You can use the Grease Pencil tool to draw freehand sketches and annotations in most of the *Editors*. The sketches that are made are saved with the blend-file so they can be seen at any time, a disadvantage of the old grease pencil. However, you can also do much more with the digital grease pencil such as:

- Planning animation poses and motion curves.
- Sketching out model topology.
- As director's tool to review shots.

Note: The *Grease Pencil* can also be used by different tools that allow you to draw where the tool is to take effect.

In the next few pages you will learn how to use the different tools such as:

- *Basic Drawing*.
- *Animating Sketches*.
- *Converting Sketches to Geometry*.

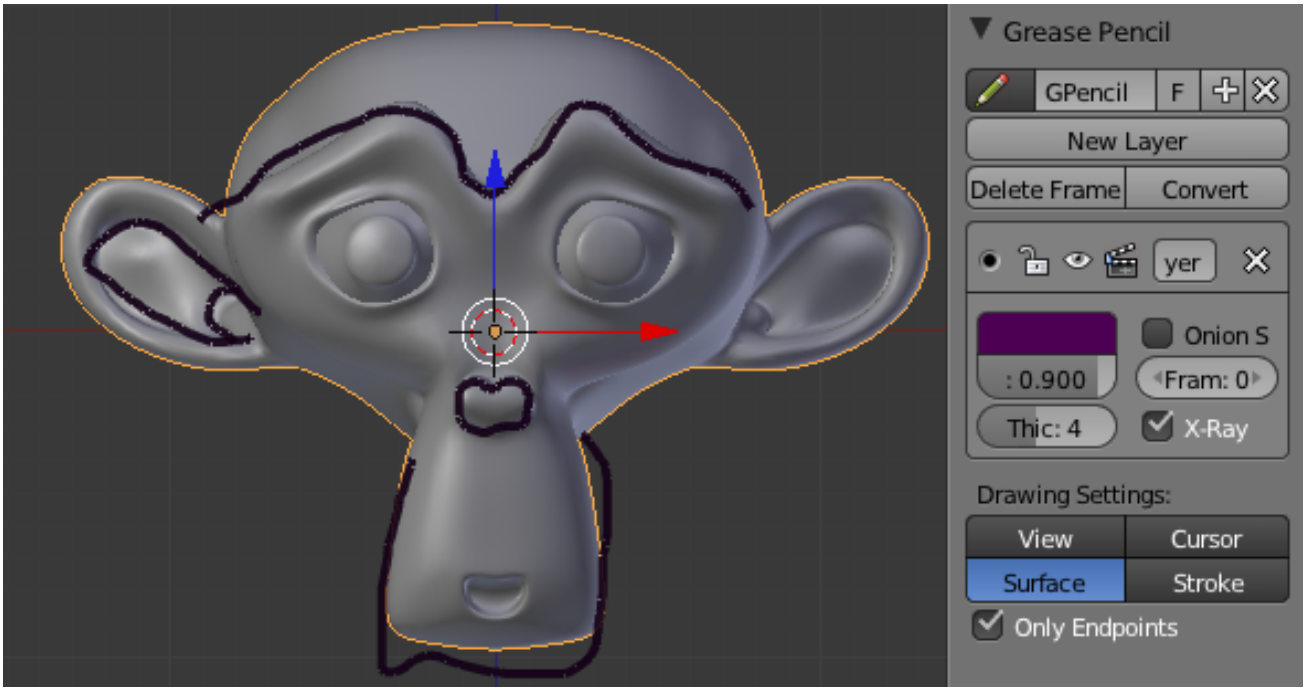


Fig. 2.48: An example of Grease Pencil.

Drawing Strokes

Introduction

Enable the *Grease Pencil* by clicking *Draw, Line, Poly or Erase* from the Tool Shelf `T`. A new layer will be automatically added for you to draw on.

A new layer can be added from the Grease Pencil panel in the Properties region. This panel can also be used to customize the color, opacity and thickness of the pencil lines. Changes to these settings will affect all strokes on the current layer.

Grease Pencil sketches can be converted to editable geometry and used to aid the animation process.

- *Read more about Layers and Animation*
- *Read more about Converting sketches to geometry*

Drawing

The Tool Shelf provides a number of options for drawing with the *Grease Pencil* which are detailed below.

Grease Pencil Mode and Shortcut Summary

Draw `D-LMB` Draw a new stroke (multiple short, connected lines). The stroke will finish when you release the mouse button.

Line `Ctrl-D-LMB` Draw a new line in rubber band mode. The line will finish when you release the mouse button.

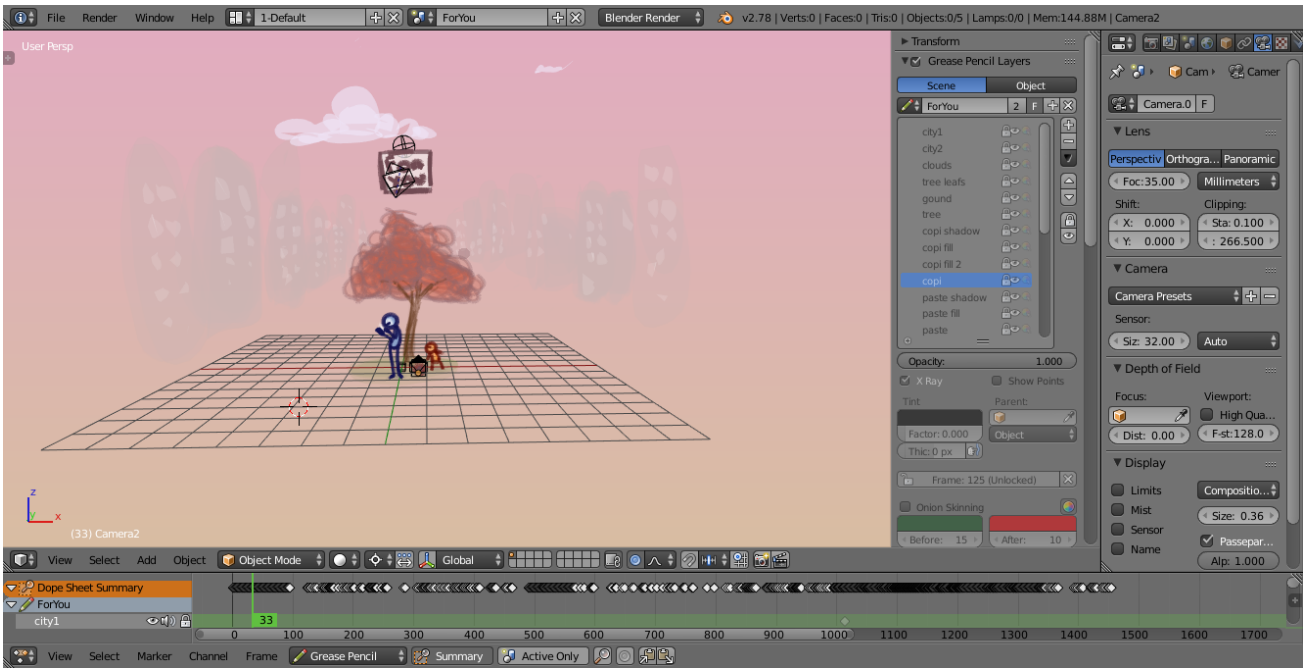


Fig. 2.49: An example of Blender's Grease Pencil.

Fig. 2.50: Grease Pencil Tool Shelf and Properties region.

Poly `Ctrl-D-RMB` Draw connected lines by clicking at various points. Lines will be automatically added to connect the two points.

Erase `D-RMB` Erases segments of strokes that fall within the radius of the eraser "brush". The erasing will continue until the mouse button is released. If begun with *Erase*, either `RMB` or `LMB` will erase strokes. The size of the eraser "brush" can be controlled with `Wheel`, or with `NumpadPlus` and `NumpadMinus`, while still holding `RMB`.

Sketching Sessions

A Sketching Session allows for rapid sketching with the *Grease Pencil* when multiple strokes are desired. With this option set, a sketching session starts when a *Grease Pencil* stroke is made. The type of session (Draw, Line, Poly, Erase) is determined by the first stroke made which can be done via hotkeys or the Tool Shelf. Use `Esc` or `Return` to exit the sketching session. Note that in an Erase Sketching Session both `LMB` or `RMB` can be used once the session has started.

Drawing Settings

In the *Grease Pencil Panel* of the *Tool* shelf `T` there are several choices for *Drawing Settings*.

View New strokes are locked to the view.

Cursor ***(3D View only)*** New strokes are drawn in 3D-space, with position determined by the 3D cursor and the view rotation at the time of drawing. *Cursor* is available as an option in the *UV/Image Editor* but it functions identically to the *View* option. *(3D View only)*

Surface New strokes are drawn in 3D-space, with their position projected onto the first visible surface. *(3D View only)*

Stroke New strokes are drawn in 3D-space, with their position projected onto existing visible strokes. Note that strokes created with *View* are not in 3D-space and are not considered for this projection. *(3D View only)*

Only Endpoints Applies the drawing setting only to the endpoints of the stroke. The part of the stroke between the endpoints is adjusted to lie on a plane passing through the endpoints.

None

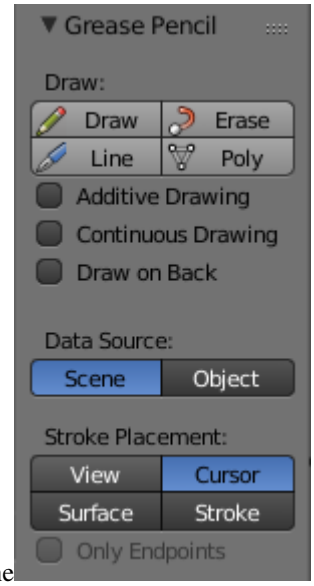


Fig. 2.51: Grease Pencil Drawing Settings.

Tip: Notes For Tablet Users:

- The thickness of a stroke at a particular point is affected by the pressure used when drawing that part of the stroke.
- The "eraser" end of the stylus can be used to erase strokes.

Fig. 2.52: The effect of different Drawing Settings on Grease Pencil strokes.

Layers

Grease Pencil sketches are organized in layers, much like the image layers in the GIMP or Photoshop®. These layers are not related to any of the other layer systems in Blender.

The layers' main purpose is to gather sketches that are related in some meaningful way (i.e. "blocking notes", "director's comments on blocking", or "guidelines"). For this reason, all the strokes on a layer (not just those made after a particular change) are affected by that layer's color, opacity, and stroke thickness settings.

Layers are managed in the *Grease Pencil Panel* of the *Properties region* `N` shown here.

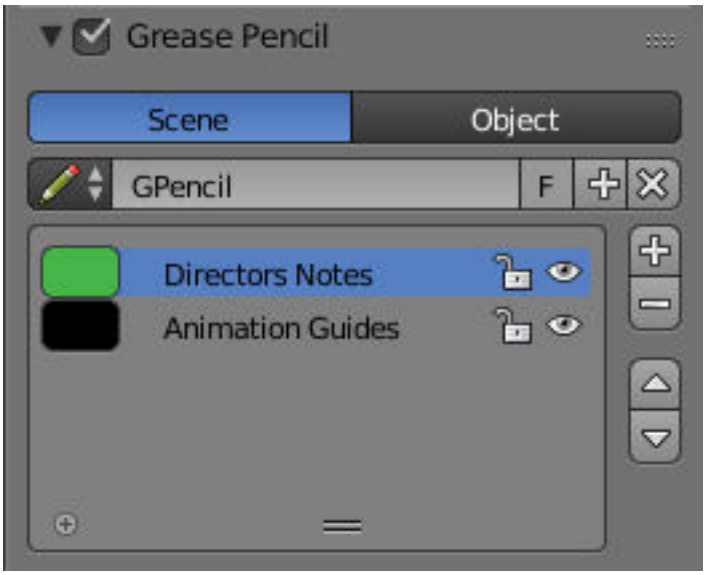


Fig. 2.53: Grease Pencil Panel.

Grease Pencil Data

Use the following controls to Add, Remove or adjust the position of a layer in the list.

Source

Scene Grease Pencil data is attached to the current scene is used, unless the active object already has Grease Pencil data (i.e old files).

Object Grease Pencil data is attached to the active object are used. This is required when using pre 2.73 add-ons.

Data-Block Used to select the Grease Pencil data-block to use for layers. For controls see *Data-Block Menu*.

Layer List There is a list of layers attached to each scene and a list of layers associated with each object.

Lock Locks the ability to edit the current layers layer.

Hide Hides the current layer in the drawing region.

Unlock Color Unprotects selected colors from further editing and/or frame changes.

Layer List Specials

Duplicate Layer Creates a copy of the current layer.

Show All Makes all hidden layers visible

Hide Others Makes all non selected layers hidden.

Lock/Unlock All Locks/Unlocks all of the layers. This can be useful to prevent unwanted editing.

Merge Down Merges the current layer with the layer below it.

Note: By default, most operations occur only on the *active* layer highlighted in the list.

Appearance Settings

These settings can be used to change how the active layer appears.

Opacity The transparency of the layer.

X-Ray Makes the lines visible when they pass behind other objects in the scene.

Show Points Draws the start/end points that make up the stroke.

Tint

Color The color to tint the layer.

Factor The amount that the *Tint Color* has on the layer.

Thickness Change Change in pixels to apply to the stroke in the current layer.

It is also possible to make this be effected by a graphics tablet.

See also:

There are also option to control *Stroke Colors*.

Animation

Parent An *Object Selector* to select the *parent* object.

Type TODO.

Lock Frame Locks the current frame displayed by layer.

Delete Frame Deletes the active frame for the active Grease Pencil Layer.

Onion Skinning

Onion-skinning, also known as ghosting, helps an animator by displaying the neighboring frames as a faded trail. Enable the option with the *Onion Skin* button in the grease pencil properties region (see *Grease Pencil Onion Skinning.* shown below).

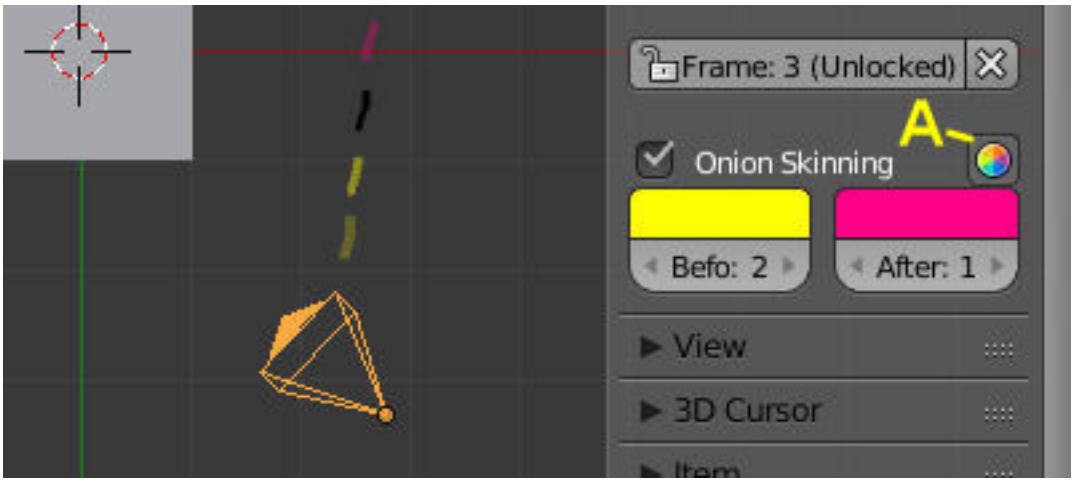


Fig. 2.54: Grease Pencil Onion Skinning.

Use Custom Colors (Marked "A") use the *Before* and *After* controls to change the color of the ghosted frames.

Before

Color The color of the strokes before the current frame.

Before Range The maximum number of frame to show before the current frame. 0 will only show the the previous sketch, and -1 will not show any frames before current.

After

Color The color of the strokes before the current frame.

After Range The maximum number of frame to show after the current frame. 0 will only show the the next sketch, and -1 will not show any frames after current.

See also:

- Grease Pencil mode in the *Dope Sheet* editor.
- Grease Pencil *Animation* page.

Colors

Palette

Colors

Isolate

Lock

Stroke Sets the line color and opacity.

Fill Sets the color of the interior space enclosed by the strokes. Increase the opacity from zero to make the fill visible. Fill works best on convex shapes, unless you are using *High Quality Fill* (see below).

Volumetric Strokes Draw strokes as a series of filled spheres, resulting in an interesting volumetric effect. Get best results with partial opacity and large stroke widths.

High Quality Fill Uses a better fill algorithm that works better for concave drawings.

Brushes

Thickness Number of pixels for full pressure strokes. The thickness can be lower depending of the pressure.

Sensibility Adjust the sensibility of the thickness to the pressure of the pencil on the tablet. This pressure can be disabled using the right small button.

Tip: These properties additionally have a randomness factor which can be enabled using the icon to the right of the properties.

Strength Similar to sensibility, but affect the saturation of the color. This parameter allows to get effects as color fading or watercolor.

Randomness Defines the level of randomness (if enabled) for sensibility and strength.

Jitter Define a jitter randomness in the stroke.

Angle Defines the angle when the thickness of the stroke will be 100%. Any change in the direction will change the thickness.

Factor Defines the effect for drawing angle changes in the thickness.

Tip: The *Angle* and *Angle Factor* parameters allow to create drawing brushes such as markers that change the thickness depending of the angle of drawing. This gets a more artistic drawing and less "computer" lines.

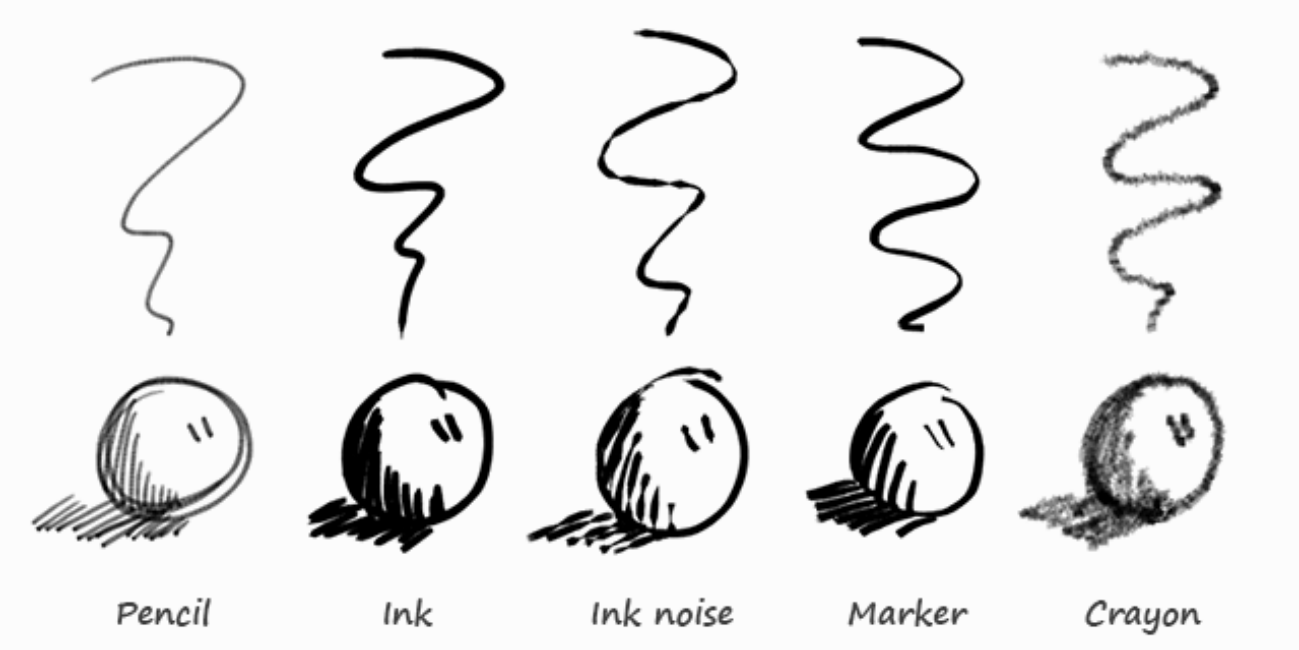


Fig. 2.55: Preset Brushes.

Stoke Quality

Smooth Defines how much smoothing is applied (using the same method as the "Smooth" Brush). It is used to get rid of jagged edges and jitter/hand shake.

Smoothing Iterations Defines how many times smoothing is applied. On each additional round of smoothing performed, the strength of the smoothing applied is halved, i.e. on the first round, it will be 100% of smoothing factor, then 50%, then 25%, etc. This setting is most useful for improving the quality of heavily subdivided strokes, where the multiple rounds of smoothing can help reduce "faceting" artifacts.

Subdivision Steps Defines how many times the stroke will be subdivided. Each time the stroke is subdivided, extra stroke points are added between each pair of existing stroke points. The main use of this setting is to make strokes look less "faceted" (especially large strokes drawn quickly). Strokes are subdivided before smoothing is applied.

Randomness Amount of randomness to add new new strokes after subdivision.

Brush Curves

This panel allows you to adjust the parameters used with tablets to get personal preferences. The available curves that can be edited are:

- Sensitivity
- Strength
- Jitter

Stroke Edit Mode

These tools let you move and reshape grease pencil strokes after they have been drawn.

Open the Grease Pencil tab on the Tool Shelf. Look for the tools in the Edit Strokes panel shown here:

The basic steps are:

1. Enter Stroke Edit Mode.
2. Select some strokes.
3. Move and reshape them.

Selecting

Grease pencil strokes are formed from a series of connected vertex points. To make changes, first select points on the strokes that you want to edit. You can only select points on the active layer. The selected points are highlighted as in the image above.

Hint: Set the layer's *Stroke Thickness* to 1 to make the points more visible.

Use the mouse to select the points, or one of the selection buttons in the panel as detailed in *Basic Selection*.

Various selection functions similar to those available when editing meshes can be used:

Select All	`A`
Border Select	`B`
Circle Select	`C`
Select Linked	`Ctrl-L`
Select More	`Ctrl-NumpadPlus`
Select Less	`Ctrl-NumpadMinus`
Select Stroke	`Alt-LMB`

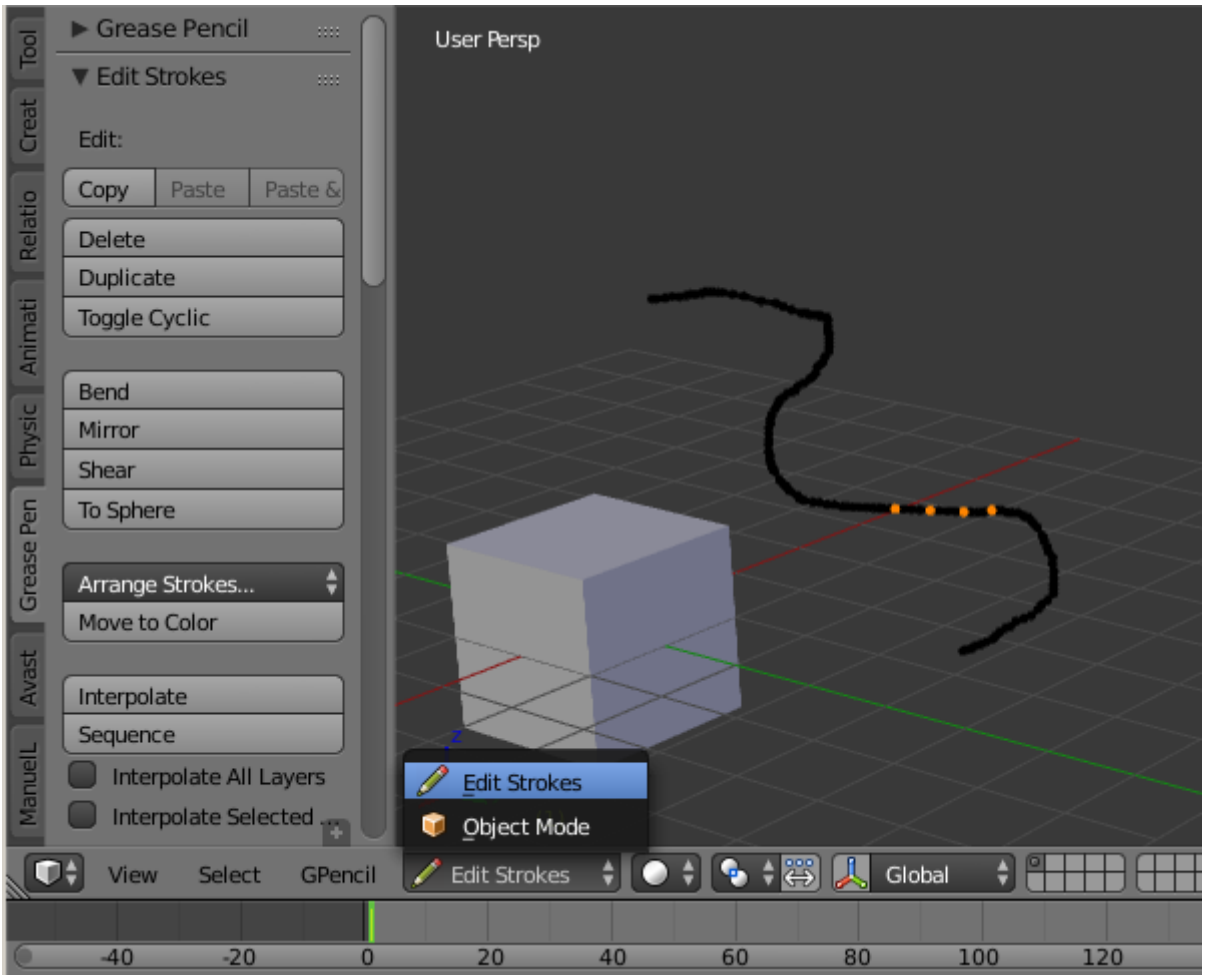


Fig. 2.56: Edit panel with grease pencil strokes.

Edit Strokes Panel

Copy Copies selected Grease Pencil points and strokes.

Paste Pastes the previously copied strokes.

Paste & Merge Pastes the previously copied strokes and merge in active layer.

Delete `X`

Points Delete the selected points, leaving a gap in the stroke.

Dissolve Reconnect the ends so there is no gap in the stroke.

Strokes Delete the entire stroke containing any selected points.

Frame Delete a frame when doing *Animating Sketches*.

Duplicate `Shift-D` Make a copy of the selected points at the same location. Use the mouse to *Translate* them into position. `LMB` places them at their new position. `RMB` cancels and removes the duplicates.

Toggle Cyclic Close or open the selected stroke by adding an edge from the last to first point.

Bend `Shift-W` Bends selected item between the 3D cursor and the mouse.

Mirror `Ctrl-M` Mirrors selected strokes along one or more axises.

Shear `Shift-Ctrl-Alt-S` Shears selected items along the horizontal screen axis.

To Sphere `Shift-Alt-S` Move selected vertieces outward in a spherical shape around the midpoint.

Arrange Strokes Arranges selected strokes up/down in the drawing order of the active layer.

Bring Froward, Send Backward, Bring to Front, Send to Back

Move to Color Moves the selected strokes to active color.

Interpolate Interpolates grease pencil strokes between frames.

Sequence Interpolates full grease pencil strokes sequence between frames.

Interpolate All Layers Interpolates all layers, not only active.

Interpolate Selected Strokes Interpolates only the selected strokes in the original frame.

Join Strokes

Type

Join Joins selected strokes.

Join & Copy Joins selected strokes as a new stroke.

Leave Gaps Leaves gaps between joined strokes instead of linking them.

Flip Direction Flips the start and end of a stroke.

Show Directions Displays stroke drawing direection with a bigger green dot of the start point and a smaller red dot for the end point.

Reproject Strokes Reprojects the selected strokes from the current viewpoint to get all points on the same plane again. This can be useful to fix problem from accidental 3D cursor movement, or viewport changes.

Sculpt Strokes Panel

Several tools for editing Grease Pencil strokes are provided in the form of brushes which you can use to "paint" or "sculpt" the appearance of the strokes without having to keep doing a tedious select-tweak-select-tweak pattern of edits.

Common Options:

Radius The size of the brush.

Strength The Strength off the brush, can be changed by the pressure of the stylus.

Use Falloff Enables a linear falloff to calculate the influence of the brush on a point. That is, a point closer to the midpoint of the brush (i.e. the point under the cursor) will get affected more than the ones at the edges.

The brushes currently implemented are:

Smooth Allows you to selectively smooth out jitter/shake and bumpiness, to tidy up messy parts of your sketches.

Affect Pressure Use this option to perform smoothing on stroke thickness values.

Thickness The Thickness Brush can be used to increase or decrease the thickness of the parts of the stroke under the cursor.

Grab Takes the stroke points which fall within the brush circle when the sculpting action begins, and allows you to translate this set of points.

Push The Push Brush is very similar to the Grab brush, in that it also allows the user to translate stroke points. However, unlike the Grab Brush, the Push Brush is not restricted to operating only on the first set of points which were under the brush when the sculpt action was initiated. Instead, on each brush movement, the points currently under the brush get moved based on the amount the brush has moved since the last time it was evaluated.

Twist Used to twist/rotate points around the cursor, creating a "swirling" effect. It is useful for applying low levels of distortion to stroke points.

Pinch/Inflate Used to draw points away from the cursor, or towards it.

Pinch Draw points towards the cursor.

Inflate Push points away from the cursor.

Clone Brush Used to paste whatever is on the Copy/Paste buffer on the active layer, located at the point where you clicked.

Stamp Mode Moves the newly pasted strokes so that their center follows the movements of the brush/cursor

Stamp + Smudge When the *Use Falloff* option is enabled, instead of moving all the newly pasted strokes by the same amount, only the points that are currently under the cursor get affected. Thus, this in this mode of operation, the brush is closer to a Paste + Push operation instead.

Continuous As the brush moves, repeatedly just paste new copies for where the brush is now. In effect, this treats the contents of the copy buffer as the "brush template/kernel" used for "dabbing" samples all over the canvas.

Selection Mask Used to restrict the brush to only operating on the selected points.

Alpha Alpha value for selected vertices.

Animating Sketches

Use the Grease Pencil to do basic pencil tests (i.e. 2D animation in flipbook style). Sketches are stored on the frame that they were drawn on, as a separate drawing (only on the layer that they exist on). Each drawing is visible until the next drawing for that layer is encountered. The only exception to this is the first drawing for a layer, which will also be visible before the frame it was drawn on.

Therefore, it is simple to make a pencil-test/series of animated sketches:

1. Go to first relevant frame. Draw.
2. Jump to next relevant frame. Draw some more.
3. Keep repeating process, and drawing until satisfied. Voila! Animated sketches.

Converting Sketches to Objects

In the 3D View, sketches on the active layer can be converted to geometry, based on the current view settings, by transforming the points recorded when drawing (which make up the strokes) into 3D-space. Currently, all points will be used, so it may be necessary to simplify or subdivide parts of the created geometry for standard use.

Sketches can currently be converted into curves, as proposed by the *Convert Grease Pencil* menu popped-up by the *Convert* button in the grease pencil properties.

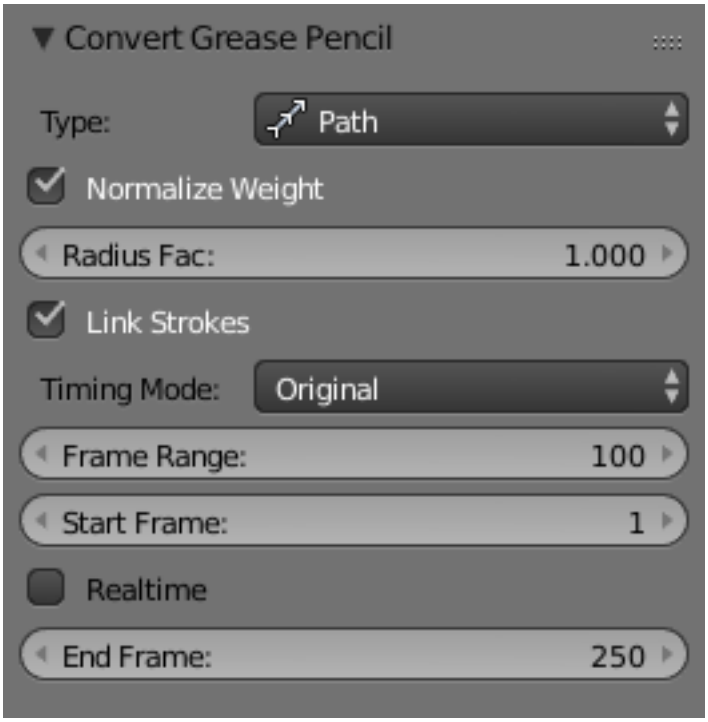


Fig. 2.57: The Convert to Curve options.

Options

Type The type of object to convert to.

Path Create NURBS 3D curves of order 2 (i.e. behaving like polylines).

Bézier Curve Create Bézier curves, with free "aligned" handles (i.e. also behaving like polylines).

Polygon Curve Bézier Curve with strait line segments (auto handles).

Note: Converting to Mesh

If you want to convert your sketch to a mesh, simply choose first *NURBS*, and then convert the created curve to a mesh.

Normalize Weight Will scale weights value so that they tightly fit into the (0.0 to 1.0) range. (enabled by default)

All this means that with a pressure tablet, you can directly control the radius and weight of the created curve, which can affect e.g. the width of an extrusion, or the size of an object through a *Follow Path* constraint or *Curve* modifier!

Link Strokes Will create a single spline, i.e. curve element. (enabled by default) from all strokes in active grease pencil layer. This especially useful if you want to use the curve as a path. All the strokes are linked in the curve by "zero weights/radii" sections.

Timing

Grease pencil stores "dynamic" data, i.e. how fast strokes are drawn. When converting to curve, this data can be used to create an *Evaluate Time* F-Curve (in other words, a path animation), that can be used e.g. to control another object's position along that curve (*Follow Path* constraint, or, trough a driver, *Curve* modifier). So this allows you to reproduce your drawing movements.

Warning: All those "timing" options need *Link Stroke* to be enabled, else they would not make much sense!

Timing Mode This control let you choose how timing data are used.

No Timing Just create the curve, without any animation data (hence all following options will be hidden).

Linear The path animation will be a linear one.

Original The path animation will reflect to original timing, including for the "gaps" (i.e. time between strokes drawing).

Custom Gaps The path animation will reflect to original timing, but the "gaps" will get custom values. This is especially useful if you have very large pauses between some of your strokes, and would rather like to have "reasonable" ones!

Frame Range The "length" of the created path animation, in frames. In other words, the highest value of *Evaluation Time*.

Start Frame The starting frame of the path animation.

Realtime When enabled, the path animation will last exactly the same duration it took you do draw the strokes.

End Frame When *Realtime* is disabled, this defines the end frame of the path animation. This means that the drawing timing will be scaled up or down to fit into the specified range.

Gap Duration *Custom Gaps* only. The average duration (in frames) of each gap between actual strokes. Please note that the value entered here will only be exact if *Realtime* is enabled, else it will be scaled, exactly as the actual strokes' timing is!

Example

Here is a simple "hand writing" video created with curves converted from sketch data:

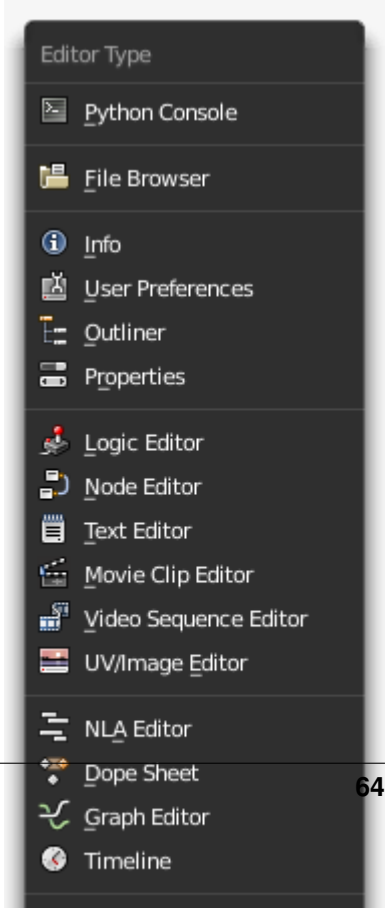
The blend-file from the above example can be found here

2.2 Editors

Blender provides a number of different editors for displaying and modifying different aspects of data.

The *Editor Type* selector, the first button at the left side of a header, allows you to change the editor in that area. Every area in Blender may contain any type of editor and it is also possible to open the same type multiple times.

There is also documentation on the *general interface* of editors.



2.2.1 3D

3D View

Introduction

The 3D View is used to interact with the 3D scene for a variety of purposes, such as modeling, animation, texture painting, etc.

Header

The header contains various menus and controls based on the current *mode*.

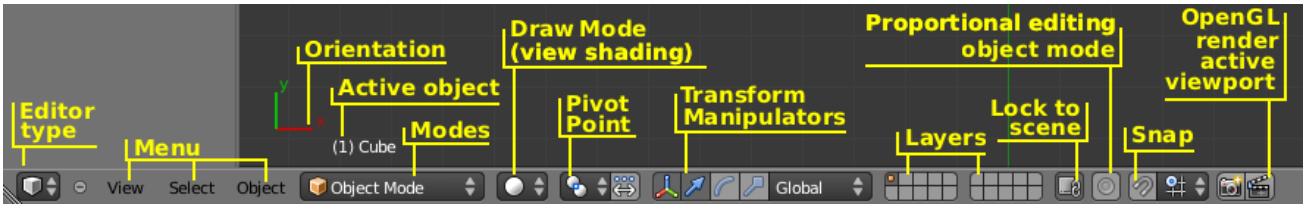


Fig. 2.59: 3D View header.

Menus

View This menu offers tools to *navigate* in 3D space.

Select Contains tools for *selecting* objects.

Add Gives a list of different *objects types* that can be added to a scene.

Object This menu appears when in Object Mode. it contains tools to edit *objects*. In edit mode, it will change to the appropriate menu with *editing tools*.

Controls

Mode The 3D view has *several modes* used for editing different kinds of data:

- Object Mode
- Edit Mode
- Pose Mode
- Sculpt Mode
- Vertex Paint
- Weight Paint
- Texture Paint
- Particle Edit

Viewport Shading Allows you to change the way objects are displayed in the viewport. *Read more about the different shading modes*

Pivot Point Used to change the reference point (or *pivot point*) used by many mesh manipulation tools.

Read more about Pivot Points

Transform Manipulator These handy selectors allow you to rotate or move objects by grabbing (clicking with your mouse) their controls and moving your mouse in the axis.

Read more about Transform Manipulators

Layer The Layers Widget is documented in the *Layers page*.

Lock to Scene By default, the "lock" button to the right of the layer buttons is enabled. This means that in this view, the active layers and camera are those of the whole scene (and those used at render time). Hence, all 3D Views locked this way will share the same active layers and camera. When you change them in one view, all locked others will immediately reflect these changes.

But if you disable this "lock" button, you then can specify different active layers and camera, specific to this view. This might be useful if you do not want to have your working areas (views) cluttered with the whole scene, and

still have an ancillary complete view (which is unlocked with e.g. all layers shown). Or to have several views with different active cameras. Remember that you can use `Ctrl-Numpad0` to make the active object the active camera.

Read more about Scenes

Proportional Edit *Proportional Edit*.

Snap Controls the *snapping tools* that help with transforming and modeling objects.

OpenGL Render The Render Buttons render an OpenGL version of the 3D View. See the *OpenGL Rendering* page for more information.

Tool Shelf

The Tool shelf is a context-sensitive region containing tools depending on the current mode (for example, modeling tools in *Edit Mode*, brush tools in *Sculpt Mode*...).

For more information on specific tools available, see:

- *Transformations*
- *History*
- *Creating Objects*
- *Parents*
- *Groups*
- *Animation*
- *Rigid Body*
- *Grease Pencil*
- *Modeling*
- *Sculpting*
- *Vertex Paint*
- *Weight Paint*
- *Texture Paint*

Properties Region

The Properties Region contains properties of the active object and selected objects (such as their locations), as well as properties of the editor itself:

- *Transform*
- *Grease Pencil*
- *Display & View Panels*
- *Background Images*
- *Transform Orientations*

Startup Scene

After closing the splash the startup scene is displayed in the 3D View, if no other blend-file was loaded.

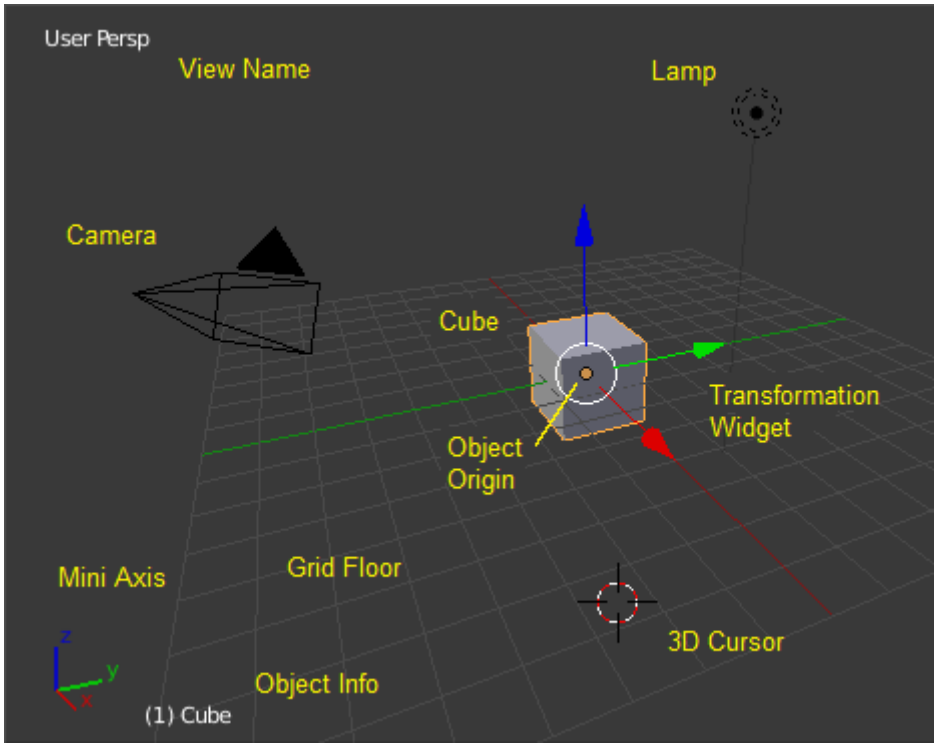


Fig. 2.60: The Startup scene.

Elements

Cube The gray cube in the center of the scene is a *mesh* object. Because the cube is selected it is drawn with an orange outline.

Object Origin The *Origin of the object* is displayed as an orange dot and it marks the cube's (relative) position.

Transformation Widget This *widget* is composed out of a white circle and three colored (red, green, and blue) arrows. It is used to move entities (i.e. the cube) in the scene.

Lamp The circle with a thin line to the bottom is a light source illuminating the cube. Lights in: *Blender Internal*, *Cycles*.

Camera The pyramid with a big triangle pointing upward is the camera used as point of view for rendering. Cameras in *Blender Internal*, *Cycles*.

3D Cursor The *3D cursor*, a cross with a red and white circle, is used for placing objects in the scene.

Grid Floor The gray squares forming a floor mark the zero height of the world. The red and green lines are the axis of the world coordinate system. They meet at the origin, which is also the position of the *Cube*. The Grid Floor settings are in the *Display panel*.

Overlays

The visibility and settings of the overlays can be set in the *User Preferences*.

View Name If the viewport camera is not aligned, the view is named "User" plus the perspective of the viewport camera.

Playback FPS Displays the Frames Per Second screen rate, while playing an animation back.

Mini Axis Shows the axes of world coordinate system as plain lines with name.

Object Info Shown in brackets is the current frame. Followed by the name of the *active object*. And optionally the selected *shape key* and in brackets (<>) the *Markers* name on the current frame. The color of the Object Info is set by the *State Colors* (keyframe only).

Object Modes

Modes are a Blender-level object-oriented feature, which means that whole Blender application is always in a singular mode, and that the available modes vary depending on the selected active object's type – most of them only enable the default *Object Mode* (like cameras, lamps, etc.). Each mode is designed to edit an aspect of the selected object. See Tab. *Blender's Modes* below for details.

You set the current mode in the *Mode* selector of *3D View* header (see Fig. *The Mode select menu.*).

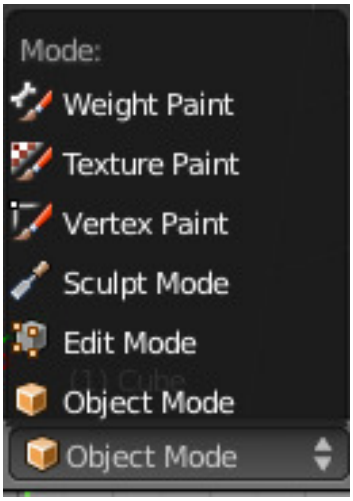


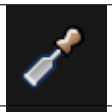
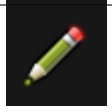
Fig. 2.61: The Mode select menu.

Warning: You can only select objects in *Object Mode*. In all others, the current object selection is "locked" (except, to some extent, with an armature's *Pose Mode*).

Modes might affect many things in Blender:

- They can modify the panels and/or controls available in some Properties editor tabs.
- They can modify the behavior of the whole editor, like e.g. the *UV/Image Editor* and *3D View*.
- They can modify the available header tools (menus and/or menu entries, as well as other controls...). For example, in the *3D View* editor, the *Object* menu in *Object Mode* changes to a *Mesh* menu in *Edit Mode* (with an active mesh object!), and a *Paint* menu in *Vertex Paint Mode*...

Table 2.3: Blender's Modes

Icon	Name	Shortcut	Details
	*Object Mode*	*None* [1]	The default mode, available for all object types, as it is dedicated to *Object* data-block editing (i.e. position/rotation/size).
	*Edit Mode*	`Tab` [1]	A mode available for all renderable object types, as it is dedicated to their "shape" *Object Data* data-block editing (i.e. vertices/edges/faces for meshes, control points for curves/surfaces, etc.)
	*Sculpt Mode*	*None* [1]	A mesh-only mode, that enables Blender's mesh 3D-sculpting tool.
	*Vertex Paint Mode*	*None* [1]	A mesh-only mode, that allows you to set your mesh's vertices colors (i.e. to "paint" them).
	*Texture Paint Mode*	*None* [1]	A mesh-only mode, that allows you to paint your mesh's texture directly on the model, in the 3D Views.
	*Weight Paint Mode*	`Ctrl-Tab` [2]	A mesh-only mode, dedicated to vertex group weighting.
	*Particle Edit Mode*	*None* [1]	A mesh-only mode, dedicated to particle systems, useful with editable systems (hair).
	*Pose Mode*	`Ctrl-Tab` [2]	An armature only mode, dedicated to armature posing.
	Edit Strokes Mode	`D-Tab`	A Grease Pencil only mode, dedicated to editing Grease Pencil strokes.

As you can see, using shortcuts to switch between modes can become quite tricky, especially with meshes.

Note: The cursor becomes a brush in:

- *Vertex Paint* mode
- *Weight Paint* mode
- *Texture Paint* mode.

[1] `Tab` toggles *Edit Mode*.

[2] `Ctrl-Tab` switches between the *Weight Paint Mode* (meshes)/*Pose Mode* (armatures) , and the other current one (by default, the *Object Mode*). However, the same shortcut has other, internal meanings in some modes (e.g. in *Sculpt Mode*, it is used to select the current brush)...

We will not go into any more detail on modes usages here, However, most of them are tackled in the *modeling chapter*, as they are mainly related to this topic. The *Particle Edit Mode* is discussed in the *particle section*, and the *Pose Mode* and *Edit Mode* for armatures, in the *rigging one*.

Note: If you are reading this manual and some button or menu option is referenced that does not appear on your screen, it may be that you are not in the proper mode for that option to be valid.

Navigating

Introduction

Navigating in the 3D space is done with the use of both mouse movement and keyboard shortcuts.

Orbit `MMB` Rotate the view around the point of interest.

Pan `Shift-MMB` Move the view up, down, left and right.

Zoom `Ctrl-MMB`, `Wheel` Move the camera forwards and backwards.

View Menu

The view menu is located in the header of the 3D View.

Camera Switches the view to the current camera view.

Viewing angles: These commands change the view to the default Top/Bottom, Front/Back, or Left/Right views:

- Top `Numpad7`
- Bottom `Ctrl-Numpad7`
- Front `Numpad1`
- Back `Ctrl-Numpad1`
- Right `Numpad3`
- Left `Ctrl-Numpad3`

Cameras Menu:

- Set Active object as camera
- Active camera

Perspective/Orthographic View These commands change the projection of the 3D View

Navigation Menu This sub-menu contains commands for rotating and panning the view. Using these commands through the menu is not that efficient. However, like all Blender menus, the much more convenient keyboard shortcuts are listed next to the commands.

Align View This submenu allows you to align the 3D View in certain ways:

- Align to selected
- Center cursor and view all
- Align active camera to view
- View Selected
- Center View to cursor

Clipping Border Allows you to define a clipping border to limit the 3D View display to a portion of 3D space.

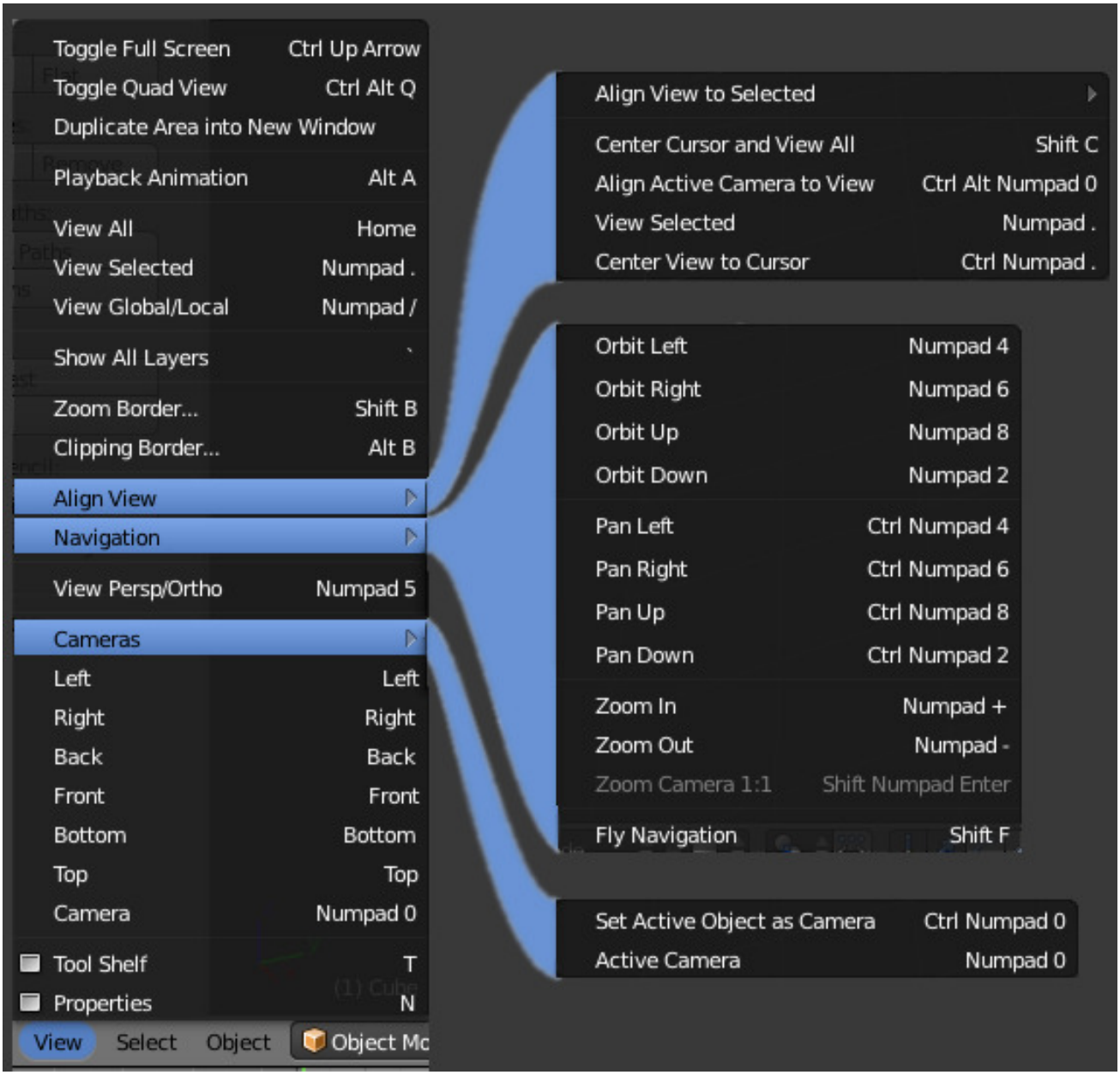


Fig. 2.62: The View menu.

Zoom Border Allows you to define the area you want to zoom into.

Show all Layers Makes all of the display layers visible.

Global View/Local View Global view shows all of the 3D objects in the scene. Local view only displays the selected objects. This helps if there are many objects in the scene, that may be in the way. Accidentally pressing `NumpadSlash` can happen rather often if you are new to Blender, so if a bunch of the objects in your scene seem to have mysteriously vanished, try turning off local view.

View Selected Zooms the 3D View to encompass all the *selected* objects.

View All Zooms the 3D View to encompass *all* the objects in the current scene.

Play Back Animation Plays back the animation from the current frame.

Duplicate area in new window Clones the current 3D View in a new window.

Quad View Toggles a four view 3D View, each showing a different angle of the scene.

Toggle Full Screen Maximizes the *3D View* editor to fill the full screen area.

3D View

To be able to work in the three dimensional space that Blender uses, you must be able to change your viewpoint as well as the viewing direction of the scene. While we will describe the *3D View* editor, most of the other windows have similar functions. For example, it is possible to translate and zoom a Properties editor and its panels.

Tip: Mouse Buttons and Numpad

If you have a mouse with less than three buttons or a keyboard without numpad, see the *Keyboard and Mouse* page of the manual to learn how to use them with Blender.

Perspective and Orthographic Views

Reference

Mode: All modes
Menu: *View* → *Perspective*, *View* → *Orthographic*
Hotkey: `Numpad5`

Each 3D View supports two different types of projection. These are demonstrated in the Fig. below.

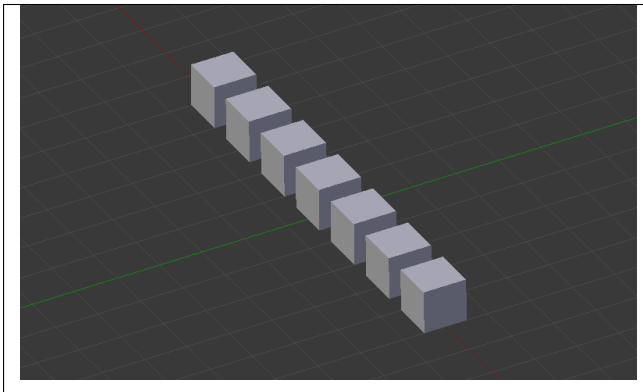


Fig. 2.63: Orthographic projections.

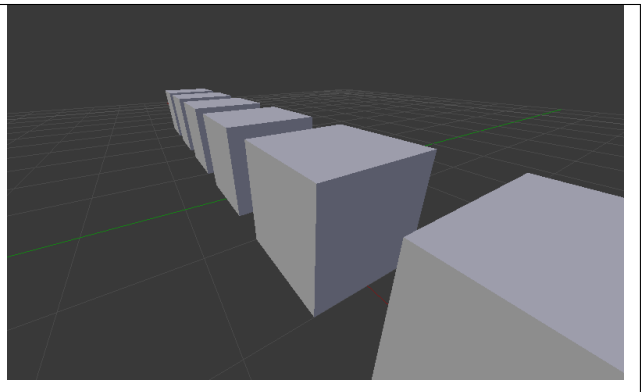


Fig. 2.64: Perspective projections.

Our eye is used to perspective viewing because distant objects appear smaller. Orthographic projection often seems a bit odd at first, because objects stay the same size regardless of their distance. It is like viewing the scene from an infinitely distant point. Nevertheless, orthographic viewing is very useful (it is the default in Blender and most other 3D applications), because it provides a more "technical" insight into the scene, making it easier to draw and judge proportions.

Options

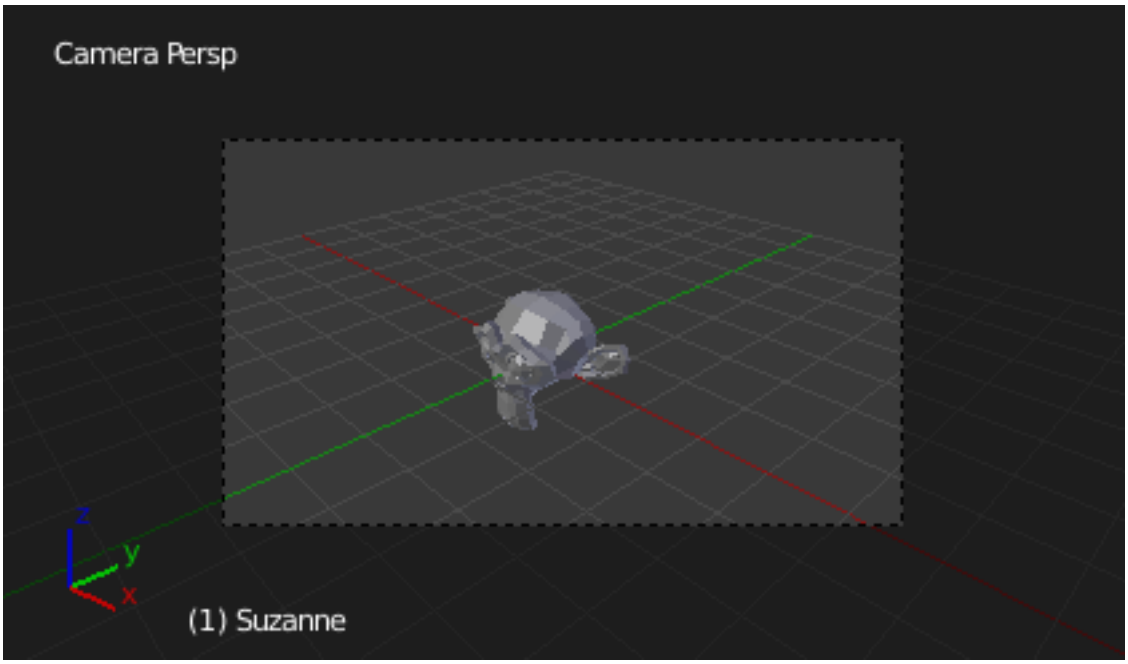


Fig. 2.65: Demonstration of camera view.

To change the projection for a 3D View, choose the *View* → *Orthographic* or the *View* → *Perspective* menu entry. The `Numpad5` shortcut toggles between the two modes. Changing the projection for a 3D View does not affect the way the scene will be rendered. Rendering is in perspective by default. If you need to create an orthographic rendering, select the camera, go to the Camera tab and press the *Orthographic* button in the *Lens* panel.

The *View* → *Camera* menu entry sets the 3D View to camera mode `Numpad0`. The scene is then displayed as it will be rendered later (see Fig. *Demonstration of camera view.*). The rendered image will contain everything within the orange dotted line. Zooming in and out is possible in this view, but to change the viewpoint, you have to move or rotate the camera.

See also:

- *Render perspectives*
- *Camera View*
- *Camera Clipping*
- *Camera Projections*

Rotating the View

Reference

Mode: All modes
Menu: *View* → *Navigation*
Hotkey: `MMB`, `Numpad2`, `Numpad4`, `Numpad6`, `Numpad8`, `Ctrl-Alt-Wheel`

Blender provides four default viewing directions: *Side*, *Front*, *Top* and *Camera* view. Blender uses a right-angled "Cartesian" coordinate system with the Z axis pointing upwards. "Side" corresponds to looking along the X axis, in the negative direction, "Front" along the Y axis, and "top" along the Z axis. The *Camera* view shows the current scene as seen from the camera view point.

Options

You can select the viewing direction for a 3D View with the *View* menu entries, or by pressing the hotkeys `Numpad3` for "side", `Numpad1` for "front", `Numpad7` for "top". You can select the opposite directions if you hold `Ctrl` while using the same numpad shortcuts. Finally `Numpad0` gives access to the "camera" viewpoint.

Apart from these four default directions, the view can be rotated to any angle you wish. Click and drag `MMB` on the viewport's area. If you start in the middle of the area and move up and down or left and right, the view is rotated around the middle of the area. Alternatively, if the *Emulate 3 button mouse* option is select in the *User Preferences* you can press and hold `Alt` while dragging `LMB` in the viewport's area.

To change the viewing angle in discrete steps, use `Numpad8` and `Numpad2` (which correspond to vertical `MMB` dragging, from any viewpoint), or use `Numpad4` and `Numpad6` (or `Ctrl-Alt-Wheel`) to rotate the scene around the Z global axis from your current point of view.

Note: Hotkeys

Remember that most hotkeys affect the **active** area (the one that has focus), so check that the mouse cursor is in the area you want to work in before your use the hotkeys.

Fig. 2.66: A 3D View's View menu.

See also:

- *Orbit Style Preference*
- *Auto-Perspective Preference*

Panning the View

Reference

Mode: All modes
Menu: *View* → *Navigation*
Hotkey: `Shift-MMB` , `Ctrl-Numpad2`, `Ctrl-Numpad4`, `Ctrl-Numpad6`, `Ctrl-Numpad8`, `Shift-Alt-LMB`

To pan the view, hold down Shift and drag MMB in the 3D View. For discrete steps, use the hotkeys Ctrl-Numpad8, Ctrl-Numpad2, Ctrl-Numpad4 and Ctrl-Numpad6 as with rotating (note: you can replace Ctrl by Shift). For those without a middle mouse button, you can hold Shift-Alt while dragging with LMB.

Zooming the View

Reference

Mode: All modes
Menu: *View* → *Navigation*
Hotkey: Ctrl-MMB, Wheel, NumpadPlus, NumpadMinus

You can zoom in and out by holding down Ctrl and dragging MMB. The hotkeys are NumpadPlus and NumpadMinus. The *View* → *Navigation* sub-menu holds these functions too as well. Refer to the 3D View's *View* menu image above for more information.

If you have a wheel mouse, you can perform all of the actions in the 3D View that you would do with NumpadPlus and NumpadMinus by rotating the Wheel. To zoom a Properties editor, hold Ctrl-MMB and move your mouse up and down.

Hint: If You Get Lost

If you get lost in 3D space, which is not uncommon, two hotkeys will help you: Home changes the view so that you can see all objects *View* → *View All*, while NumpadPeriod zooms the view to the currently selected objects when in perspective mode *View* → *View Selected*.

Zoom Border

Reference

Mode: All modes
Menu: *View* → *Zoom Border*
Hotkey: Shift-B

The *Zoom Border* tool allows you to specify a rectangular region and zoom in so that the region fills the 3D View.

You can access this through the *View* menu, or the shortcut Shift-B, then LMB click and drag a rectangle to zoom into.

Alternatively you can zoom out using the MMB.

Dolly the View

Reference

Mode: All modes

Hotkey: Ctrl-Shift-MMB

In most cases its sufficient to zoom the view to get a closer look at something, however, you may notice that at a certain point you cannot zoom any closer.

This is because Blender stores a view-point thats used for orbiting and zooming. It works well in many cases, but sometimes you want to move the view-point to a different place. This is what Dolly supports, allowing you to transport the view from one place to another.

You can dolly back and fourth by holding down Ctrl-Shift and dragging MMB.

Aligning the View

Align View

These options allow you to align and orient the view in different ways. They are found in the *View Menu*

Align View to Selected menu These options align your view with specified local axes of the selected object, bone or in *Edit Mode*, with the normal of the selected face.

Hold down Shift while using the numpad to set the view axis.

Center Cursor and View All Shift-C moves the cursor back to the origin and zooms in/out so that you can see everything in your scene.

Align Active Camera to View, Ctrl-Alt-Numpad0 Gives your active camera the current viewpoint.

View selected, NumpadPeriod Focuses view on currently selected object/s by centering them in the viewport, and zooming in until they fill the screen.

Center view to cursor, Alt-Home Centers view to 3D-cursor.

View Selected See above.

View All Home Frames all the objects in the scene, so they are visible in the viewport.

Local View

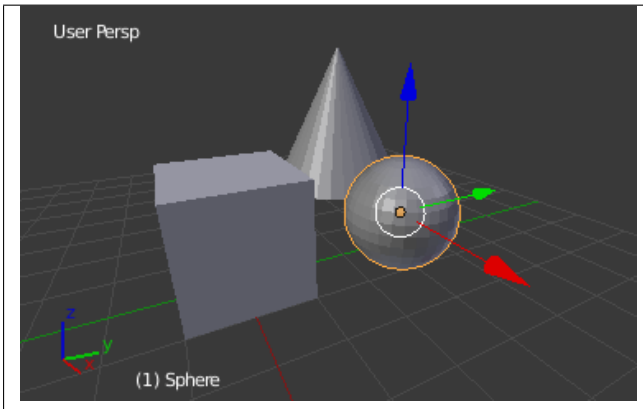


Fig. 2.67: Global View.

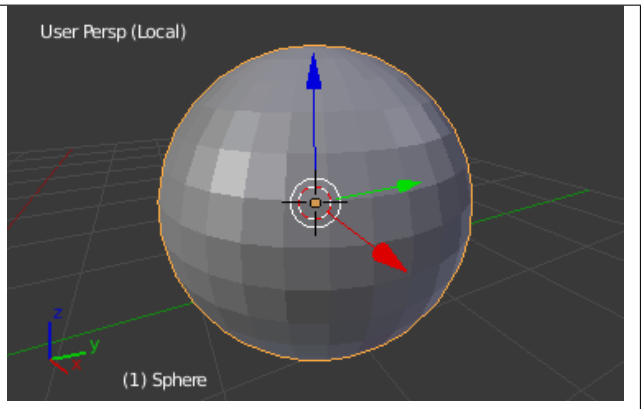


Fig. 2.68: Local View.

You can toggle *Local View* by selecting the option from the *View Menu* or using the shortcut NumpadSlash.

Local view isolates the selected object or objects, so that they are the only ones visible in the viewport. This is useful for working on objects that are obscured by other ones, or have heavy geometry. Press NumpadSlash to exit *Local View*.

This allow you to focus on a specific object without others getting in your way, and can be used to speed up viewport performance in heavy scenes.

Note: These notes cover changes in local-view which are not immediately obvious.

3D Cursor In local-view the 3D cursor is not locked to the scene. Instead, each view has an independent cursor location.

Layers Local-view bypasses layers, using only the selected objects when entering local-view. Although new objects may be added while in local-view.

Its also possible to send objects out of local view, using *Object* → *Move Objects out of Local View*, which can be useful to further isolate a selection.

Preview Renders Preview renders will still use lamps outside the local-view, this allows you to quickly render previews without having to remember to select all lamps when entering local-view.

Quad View

Reference

Mode: All modes
Menu: *View* → *Toggle Quad View*
Panel *Properties region* → *Display* → *Toggle Quad View*
Hotkey: `Ctrl-Alt-Q`

Toggling Quad View will split the 3D View into four views: 3 *Orthographic* "side views" and a *Camera/User View*. This view will allow you to instantly see your model from a number of view points. In this arrangement, you can zoom and pan each view independently but you cannot rotate the view.

Note: Quad View is different from *splitting the area* and aligning the view manually. In Quad View, the four views are still part of a single 3D View. So they share the same draw options and layers.

Options

These options can be found in *Properties region* → *Display*.

Lock If you want to be able to rotate each view, you can un-check the *Locked* option.

Box Syncs view position between side views.

Clip Clip objects based on what is visible in other side views.

View Clipping Border

Reference

Mode: All modes
Menu: *View* → *Set Clipping Border*
Hotkey: `Alt-B`

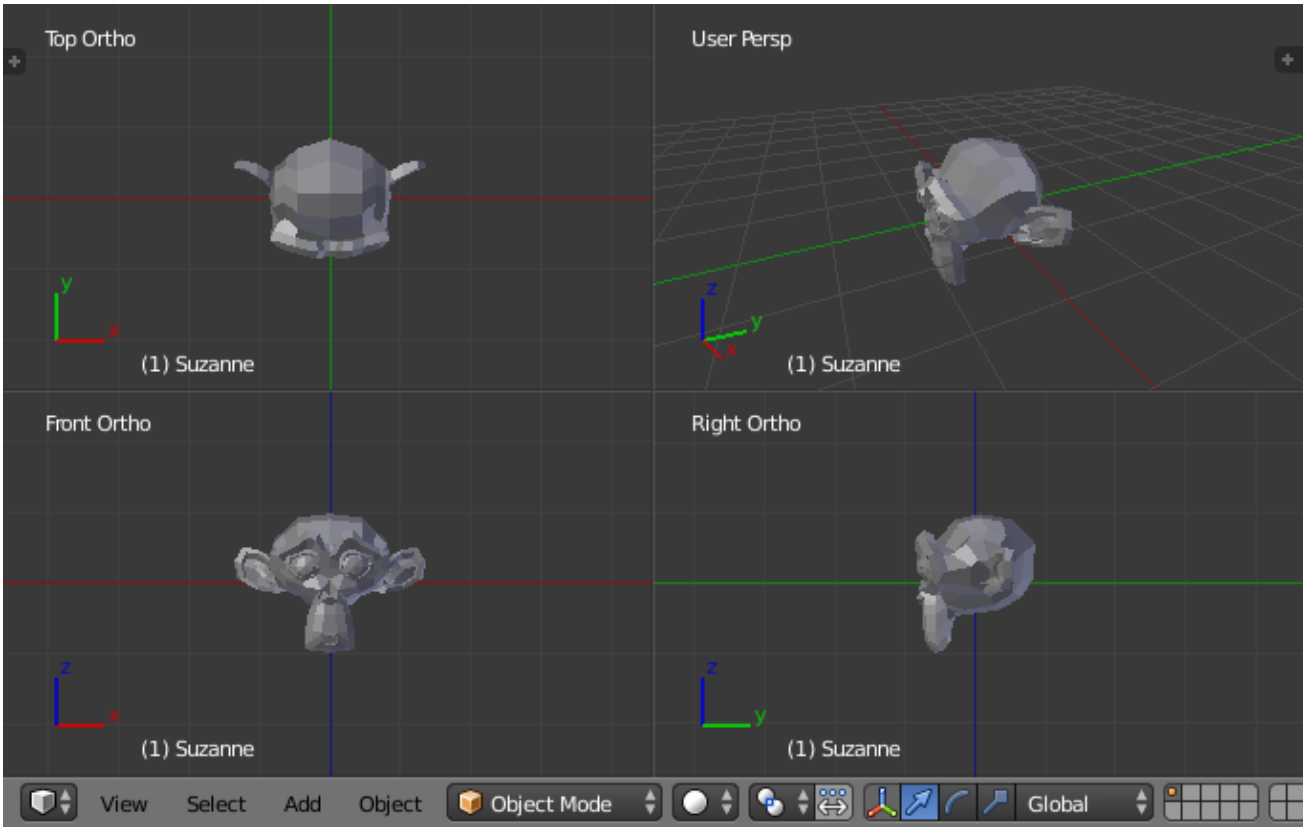


Fig. 2.69: Quad View.

Table 2.4: View rotated.

To assist in the process of working with complex models and scenes, you can set the view clipping to visually isolate what you are working on.

Once clipping is used, you will only see whats inside a volume you have defined. Tools such as paint, sculpt, selection, transform-snapping, etc. will also ignore geometry outside the clipping bounds.

Once activated with `Alt-B`, you have to draw a rectangle with the mouse, in the wanted 3D View. The created clipping volume will then be:

- A right-angled parallelepiped (of infinite length) if your view is orthographic.
- A rectangular-based pyramid (of infinite height) if your view is in perspective.

To delete this clipping, press `Alt-B` again.

Example

The *Region/Volume clipping* image shows an example of using the clipping tool with a cube. Start by activating the tool with Alt-B (upper left of the image). This will generate a dashed cross-hair cursor. Click with the LMB and drag out a rectangular region shown in the upper right. Now a region is defined and clipping is applied against that region in 3D space. Notice that part of the cube is now invisible or clipped. Use the MMB to rotate the view and you will see that only what is inside the pyramidal volume is visible. All the editing tools still function as normal but only within the pyramidal clipping volume.

The dark gray area is the clipping volume itself. Once clipping is deactivated with another Alt-B, all of 3D space will become visible again.

Walk/Fly Mode

When you have to place the view, normally you do as described above.

However, there are cases in which you really prefer to just navigate your model, especially if it is very large, like environments or some architectural model. In these cases viewing the model in perspective mode has limitations, for example after zooming a lot of panning is extremely uncomfortable and difficult, or you apparently cannot move the camera any nearer. As an example, try to navigate to a very distant object in the view with traditional methods (explained above) and see what you can get.

With walk/fly modes you move, pan, tilt, and dolly the camera around without any of those limitations.

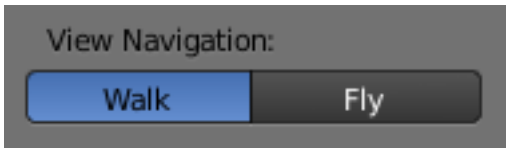


Fig. 2.73: View Navigation.

In the *User Preferences editor* select the navigation mode you want to use as default when invoking the View Navigation operator. Alternatively you can call the individual modes from the View Navigation menu.

Note: This mode actually *moves* the camera used by the view. This means that when you are in camera view, it moves the active camera, which is another way to place and aim it.

Walk Mode

Reference

Mode: All modes
Menu: *View* → *View Navigation* → *Walk Navigation*
Hotkey: Shift-F

On activation the mouse pointer will move at the center of the view, and a cross marker will appear...

This navigation mode behaves similar to the first person navigation system available in most 3D world games nowadays. It works with a combination of keyboard keys W, A, S, D and mouse movement. By default the navigation is in the "free" mode, with no gravity influence. You can toggle between gravity and free mode during the navigation Tab.

To move to places more quickly you can teleport Spacebar around your scene. If there is an object in front of the walk cross/aim you will move in that direction until you reach the point (offset by the *camera height* value set in the *User Preferences*).

Shortcuts

- Move the mouse left/right to pan the view left/right or move it up/down to tilt the view up/down.
- Move the camera forward/backward W, S.
- Strafe left/right A, D.
- Jump V - only in *gravity* mode.
- Move up and down Q, E - only in *free* mode.
- Alternate between *free* and *gravity* modes Tab.
- Change the movement speed: - WheelUp or NumpadPlus to increase the movement speed for this open session. - WheelDown or to NumpadMinus to decrease the movement speed for this open session. - Shift (hold) - to speed up the movement temporarily. - Alt (hold) - to slow down the movement temporarily.

When you are happy with the new view, click LMB to confirm. In case you want to go back from where you started, press Esc or RMB, as usual.

If the defaults values (speed, mouse sensitivity, ...) need adjustments for your project, in the *User Preferences* you can select a few options for the navigation system:

Fly Mode

Reference

Mode: All modes
Menu: *View* → *View Navigation* → *Fly Navigation*
Hotkey: Shift-F

On activation the mouse pointer will move at the center of the view, and a squared marker will appear – a sort of HUD...

Some of the options of Fly mode are influenced by the position of the mouse pointer relative to the center of the view itself, and the squared marker around this center provides a sort of "safe region" where you can place the mouse for it to have no effect on the view. The more you take the mouse pointer away from the marker, the more you pan, or track, etc.

Shortcuts

- Move the mouse left/right to pan the view left/right or move it up/down to tilt the view up/down.
- Move the view forward/backward: - WheelUp or NumpadPlus to accelerate the movement forward. - WheelDown or to NumpadMinus to accelerate the movement backward.

  So if the view is already moving forward, WheelDown, NumpadMinus will eventually stop it and then move it backward, etc.
- Drag the MMB to dolly. In this case the view can move laterally on its local axis at the moment you drag the mouse – quite obviously, dragging left/right/up/down makes the view dolly on the left/right/up/down respectively.

When you are happy with the new view, click LMB to confirm. In case you want to go back from where you started, press Esc or RMB, as usual.

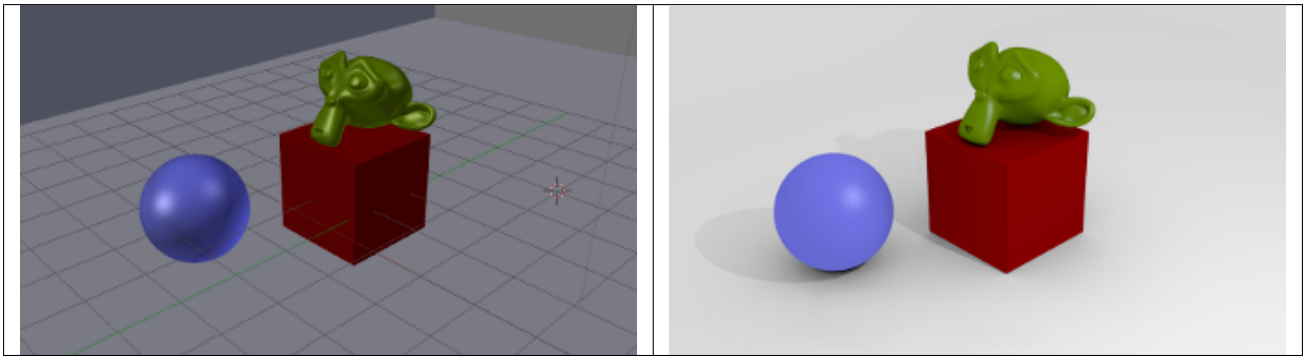
Camera View

Reference

Mode: All modes
Menu: *View* → *Camera* → *Active Camera*
Hotkey: `Numpad0`

Cameras View can be used to virtually compose shots and preview how the scene will look when rendered. Pressing `Numpad0` will show the scene as viewed from the currently active camera. In this view you can also set the *Render Border* which defines the portion of the 3D View to be rendered.

Table 2.5: Rendered image.



There are several different ways to navigate and position the camera in your scene, some of them are explained below.

Camera Navigation

There are several different ways to navigate and position the camera in your scene, some of them are explained below.

Note: Remember that the active "camera" might be any kind of object. So these actions can be used, for example, to position and aim a lamp.

Move Active Camera to View

Reference

Mode: Object Mode
Hotkey: `Ctrl-Alt-Numpad0`

This feature allows you to position and orient the active camera to match your current viewport.

Select a camera and then move around in the 3D View to a desired position and direction for your camera (so that you are seeing what you want the camera to see). Now press `Ctrl-Alt-Numpad0` and your selected camera positions itself to match the view, and switches to camera view.

Camera View Positioning

By enabling *Lock Camera to View* in the View panel of the Properties region, while in camera view, you can navigate the 3D View as usual, while remaining in camera view. Controls are exactly the same as when normally moving in 3D.

Roll, Pan, Dolly, and Track

To perform these camera moves, the camera must first be *selected*, so that it becomes the active object (while viewing through it, you can RMB – click on the solid rectangular edges to select it). The following actions also assume that you are in camera view Numpad0! Having done so, you can now manipulate the camera using the same commands that are used to manipulate any object:

Roll Press R to enter object rotation mode. The default will be to rotate the camera in its local Z-axis (the axis orthogonal to the camera view), which is the definition of a camera "roll".

Vertical Pan or Pitch This is just a rotation along the local X-axis. Press R to enter object rotation mode, then X twice (the first press selects the *global* axis, pressing the same letter a second time selects the *local* axis – this works with any axis; see the *axis locking page*).

Horizontal Pan or Yaw This corresponds to a rotation around the camera's local Y axis. Press R, and then Y twice.

Dolly To dolly the camera, press G then MMB (or Z twice).

Sideways Tracking Press G and move the mouse (you can use X twice or Y to get pure-horizontal or pure-vertical sideways tracking).

See also:

Fly/Walk Mode When you are in walk/fly mode, navigation actually moves your camera:

Lock Camera to View **When enabled,** performing typical view manipulation operations will move the camera object.

3D Cursor

The 3D Cursor is simply a point in 3D space which can be used for a number of purposes.

Placement

There are a few methods to position the 3D cursor.

Direct Placement with the Mouse

Using LMB in the 3D View will place the 3D cursor directly under your mouse pointer.

For accuracy you should use two perpendicular orthogonal 3D Views, i.e. any combination of top Numpad7, front Numpad1 and side Numpad3. That way you can control the positioning along two axes in one view and determine depth in the second view.

To place the 3D Cursor on the surface of geometry, enable *Cursor Depth* in the *User Preferences*.

Using the Snap Menu

The *Snap* menu *Object/Mesh* → *Snap*, Shift-S will allow you to snap the cursor in the following ways:

Cursor to Selected Snaps the cursor to the center of the current selection.

Cursor to Center Snaps the cursor to the origin of the scene (location 0, 0, 0).

Cursor to Grid Snaps the cursor to the nearest *visible* grid line.

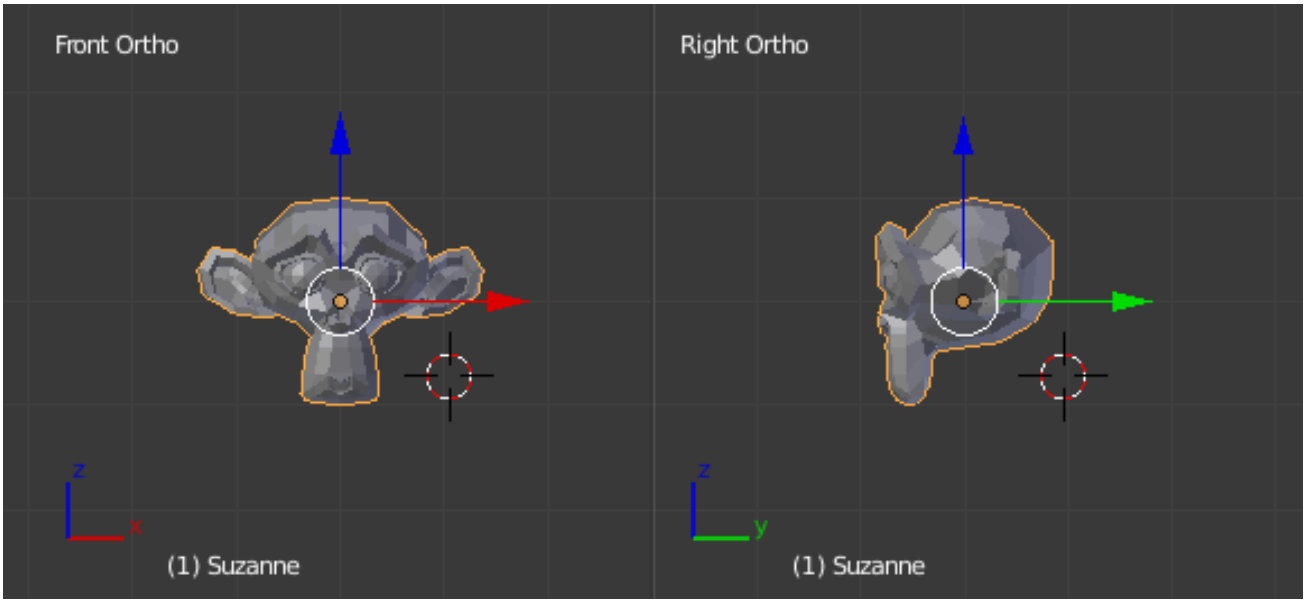


Fig. 2.76: Positioning the 3D cursor with two orthogonal views.

Cursor to Active Snaps the cursor to the *active* (last selected) object, edge, face or vertex.

The *Cursor to Selected* option is also affected by the current *Pivot Point*. For example:

- With the *Bounding Box Center* pivot point active, the *Cursor to Selected* option will snap the 3D cursor to the center of the bounding box surrounding the objects' centers.
- When the *Median Point* pivot point is selected, *Cursor to Selected* will snap the 3D cursor to the median of the object centers.

Numeric Input

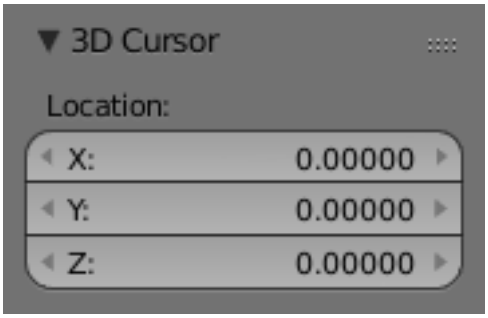


Fig. 2.77: The 3D Cursor panel of the Properties region.

The 3D cursor can also be positioned by entering Numeric location values into the *3D cursor* panel of the *Properties* region N.

Usage

Todo.

Objects

Introduction

The geometry of a scene is constructed from one or more Objects. These objects can range from lamps to light your scene, basic 2D and 3D shapes to fill it with models, armatures to animate those models, to cameras to take pictures or video of it all.

Instancing

Each Blender object type (mesh, lamp, curve, camera, etc.) is composed from two parts: an *Object* and *Object Data* (sometimes abbreviated to *ObData*):

Object Holds information about the position, rotation and size of a particular element.

Object Data Holds everything else. For example:

Meshes Store geometry, material list, vertex groups... etc.

Cameras Store focal length, depth of field, sensor size... etc.

Each object has a link to its associated *object-data*, and a single object-data may be shared by many objects.

Object Types

New objects can be created with the *Add* menu in the 3D Views header.

Mesh *Meshes* are objects composed of Polygonal Faces, Edges and/or Vertices, and can be edited extensively with Blender's mesh editing tools. See *Mesh Primitives*.

Curve *Curves* are mathematically defined objects which can be manipulated with control handles or control points (instead of vertices), to manage their length and curvature. See *Curves Primitives*.

Surface *Surfaces* are patches that are also manipulated with control points. These are useful for simple rounded forms and organic landscapes. See *Surfaces Primitives*.

Metaball *Meta Objects* (or Metaballs) are objects formed by a mathematical function (with no control points or vertices) defining the 3D volume in which the object exists. Meta Objects have a liquid-like quality, where when two or more Metaballs are brought together, they merge by smoothly rounding out the connection, appearing as one unified object. See *Meta Primitives*.

Text *Text objects* create a two dimensional representation of a string of characters.

Armature *Armatures* are used for *rigging* 3D models in order to make them poseable and animateable.

Lattice *Lattices* are non-renderable wireframes, commonly used for taking additional control over other objects with help of the *Lattice Modifier*.

Empty *Empties* are null objects that are simple visual transform nodes that do not render. They are useful for controlling the position or movement of other objects.

Speaker *Speaker* brings to scene source of sound.

Camera This is the virtual camera that is used to determine what appears in the render. See Cameras in *Blender Internal*, *Cycles*.

Lamp These are used to place light sources in the scene. See Lamps in *Blender Internal*, *Cycles*.

Force Field *Force Fields* are used in physical simulations. They give simulations external forces, creating movement, and are represented in the 3D View editor as small control objects.

Group Instance Lets you select from a list of existing object groups. Once selected, an Empty object will be created, with an instance of the selected group (group duplication active). See *DupliGroup*.

Common Options

You can change the options of the object in the Operator panel just after creating it:

Type Some objects let you change its type after creation with a selector.

Align to View By default objects are aligned to the global space axes. This option rotates the new object so that it is aligned to the view space.

Location Objects are placed, by default, at the position of the 3D Cursor. These values let you place the object in an other position.

Rotation Values let you rotate the object so that default rotation is overridden.

Selecting

Selection determines which elements will be the target of our actions. Blender has advanced selection methods. Both in *Object Mode* and in *Edit Mode*.

Selections and the Active Object

Blender distinguishes between two different states of selection:

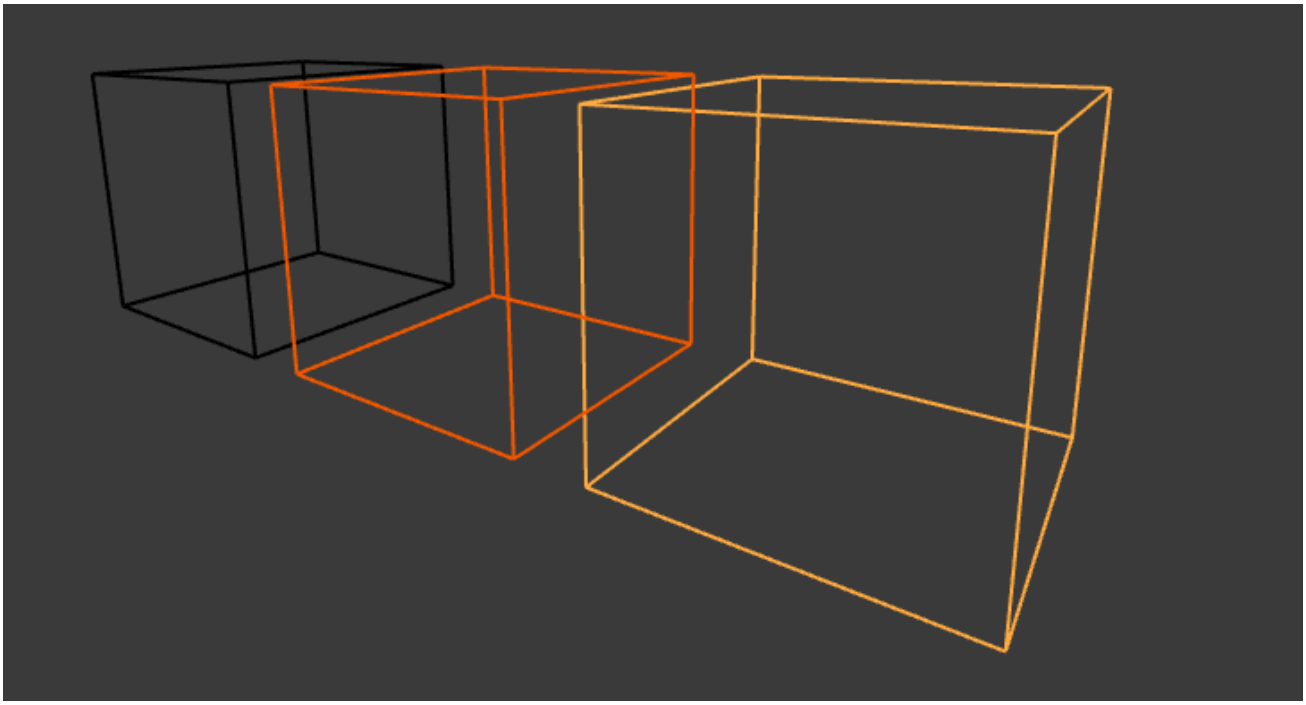


Fig. 2.78: Unselected object in black, selected object in orange, and active object in yellow.

- In *Object Mode* the last (de)selected item is called the "Active Object" and is outlined in yellow (the others are orange). There is exactly one active object at any time (even when nothing is selected).

  Many actions in Blender use the active object as a reference (for example linking operations). If you already have a selection and need to make a different object the active one, simply re-select it with `Shift-RMB`.

- All other selected objects are just selected. You can select any number of objects.

Point Selection

The simplest form of object *selection* consists of using RMB on it.

To *add to the selection*, use Shift−RMB on more objects.

If the *objects are overlapping* in the view, you can use Alt−RMB to cycle through possible choices.

If you want *to add to a selection* this way then the shortcut becomes Shift−Alt−RMB.

To *activate an object* that is already selected, click Shift−RMB on it.

To *deselect* an active object, click Shift−RMB one time and hence, two clicks if the object is not active. Note that this only works if there are no other objects under the mouse. Otherwise it just adds those to the selection. There appears to be no workaround for this bug.

Border Select

Reference

Mode: Object Mode and Edit Mode
Menu: *Select* → *Border Select*
Hotkey: B

With *Border Select* you draw a rectangle while holding down LMB. Any object that lies even partially within this rectangle becomes selected.

For deselecting objects, use MMB or *Border Select* again with holding Shift.

To cancel the selection use RMB.

Example

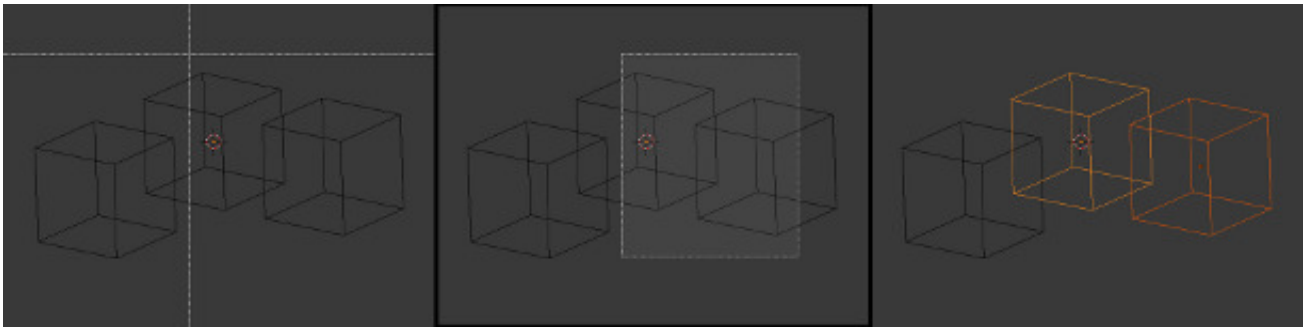


Fig. 2.79: Border selecting in three steps.

Border Select has been activated in the first image and is indicated by showing a dotted cross-hair cursor. In the second image, the *selection region* is being chosen by drawing a rectangle with the LMB. The rectangle is only covering two cubes. Finally, in the third image, the selection is completed by releasing LMB.

Notice in the third image, the bright color of left-most selected cube. This means it is the "active object", the last selected object prior to using the *Border Select* tool.

Hint: *Border Select* adds to the previous selection, so in order to select only the contents of the rectangle, deselect all with A first.

Lasso Select

Reference

Mode: Object Mode and Edit Mode
Menu: no entry in the menu
Hotkey: Ctrl-LMB

Lasso select is used by drawing a dotted line around the pivot point of the objects, in *Object Mode*.

Usage

While holding Ctrl down, you simply have to draw around the pivot point of each object you want to select with LMB.

Lasso select adds to the previous selection. For deselection, use Ctrl-Shift-LMB.

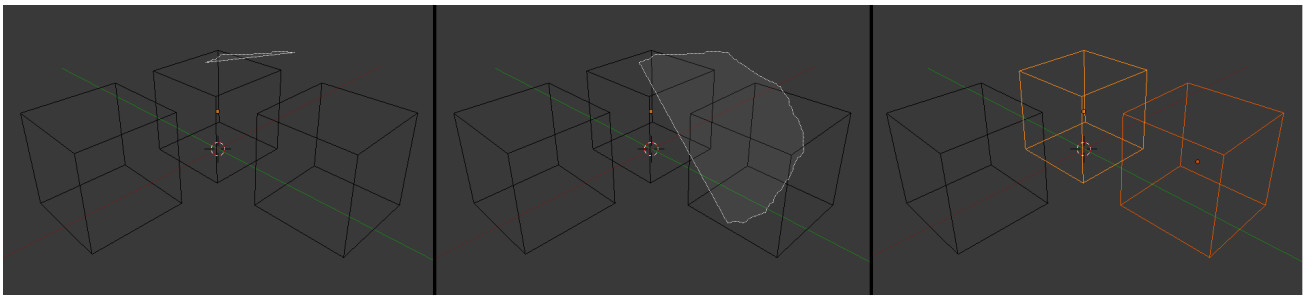


Fig. 2.80: Lasso selection example.

Circle Select

Reference

Mode: Object Mode and Edit Mode
Menu: *Select* → *Circle Select*
Hotkey: C

Circle Select is used by moving with dotted circle through objects with LMB. You can select any object by touching of circle area. It is possible to dynamically change the diameter of circle by scrolling MMB as seen in pictures below. Deselection is under the same principle - MMB. To cancel the selection use RMB or key Esc.

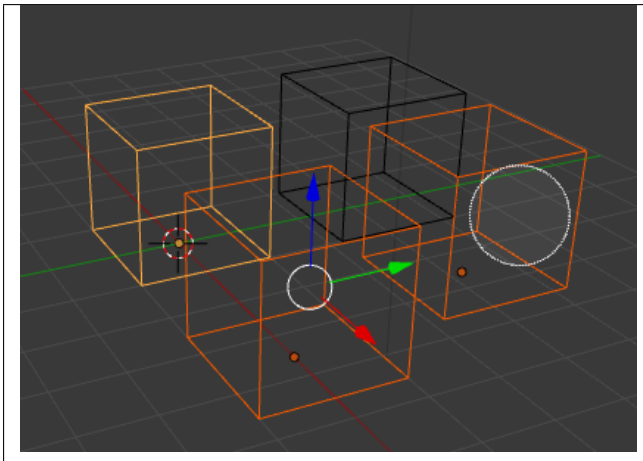


Fig. 2.81: Circle selection.

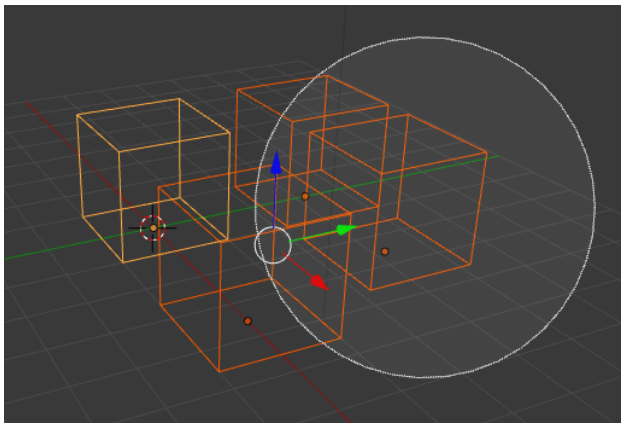


Fig. 2.82: ...with huge circle.

Menu Selection

The selection methods described above are the most common. There are also many more options accessible through the *Select* menu of the 3D View.

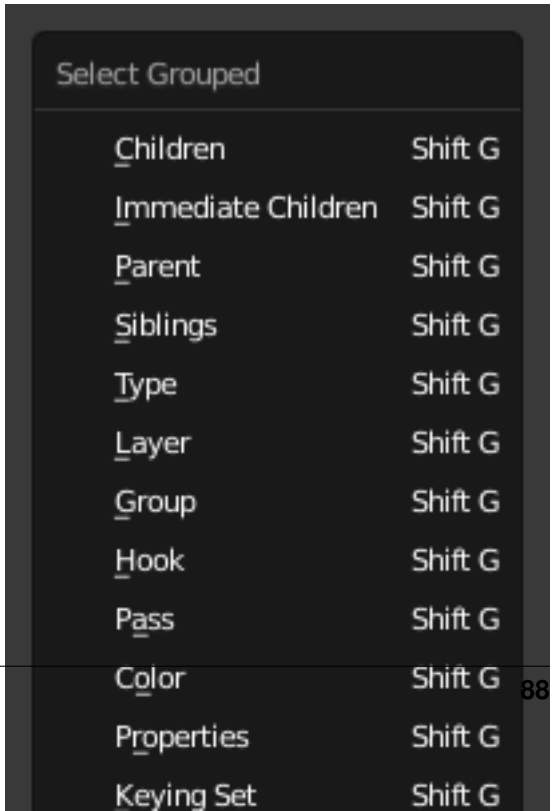
Each is more adapted to certain operations.

Select Grouped

Reference

Mode: Object Mode
Menu: *Select* → *Grouped*
Hotkey: `Shift-G`

There are two ways to organize the objects in relation to one another. The first one is *parenting*, and the second is simple *grouping*. These relationships to an artist's advantage by selecting members of respective families or groups. *Select Grouped* uses the active object as a basis to select all others.



Options

Children Selects all children of the active object recursively.

Immediate Children Selects all direct children of the active object.

Parent Selects the parent of this object if it has one.

Siblings Select objects that have the same parent as the active object. This can also be used to select all root level objects (objects with no parents).

Type Select objects that are the same type as the active one.

Layer Objects that have at least one shared layer.

Group Objects that are part of a group (rendered green with the default theme) will be selected if they are in one of the groups that the active object is in.

Object Hooks Every hook that belongs to the active object.

Pass Select objects assigned to the same *render pass*.

Color Select objects with same *Object Color*.

Properties Select objects with same *Game Engine Properties*.

Keying Set Select objects included in the active *Keying Set*.

Lamp Type Select matching lamp types.

Pass Index Select matching object pass index.

Select linked

Reference

Mode: Object Mode
Menu: *Select* → *Linked*
Hotkey: `Shift-L`

Selects all objects which share a common data-block with the active object. *Select linked* uses the active object as a basis to select all others.

Options

Object Data Selects every object that is linked to the same Object Data, i.e. the data-block that specifies the type (mesh, curve, etc.) and the build (constitutive elements like vertices, control vertices, and where they are in space) of the object.

Material Selects every object that is linked to the same material data-block.

Texture Selects every object that is linked to the same texture data-block.

Dupligroup Selects all objects that use the same *Group* for duplication.

Particle System Selects all objects that use the same *Particle System*

Library Selects all objects that are in the same *Library*

Library (Object Data) Selects all objects that are in the same *Library* and limited to *object data*.

Select All by Type

Reference

Mode: Object Mode
Menu: *Select* → *Select All by Type*
Hotkey: None

With this tool it becomes possible to select every **visible** object of a certain type in one go.

Options

The types are *Mesh*, *Curve*, *Surface*, *Meta*, *Font*, *Armature*, *Lattice*, *Empty*, *Camera*, *Lamp*, *Speaker*.

Select All by Layer

Reference

Mode: Object Mode
Menu: *Select* → *Select All by Layer*
Hotkey: None

Layers are another means to regroup your objects to suit your purpose.

This option allows the selection of every single object that belongs to a given layer, visible or not, in one single command.

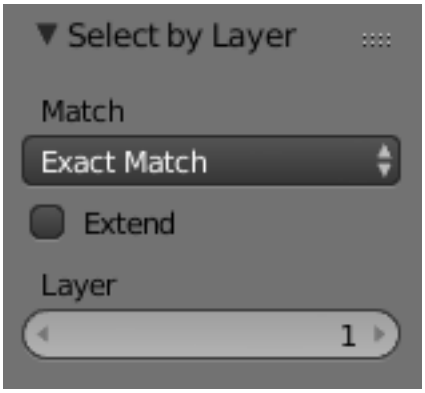


Fig. 2.84: All by Layer selection menu.

Options

Match The match type for selection.

Extend Enable to add objects to current selection rather than replacing the current selection.

Layer The layer on which the objects are.

Tip: Selection of Objects

Rather than using the *Select All by Layer* option, it might be more efficient to make the needed layers visible and use `A` on them. This method also allows objects to be deselected.

Other Menu Options

Available options on the first level of the menu are:

Select Pattern Selects all objects whose name matches a given pattern. Supported wildcards: * matches everything, ? matches any single character, [abc] matches characters in "abc", and [!abc] match any character not in "abc". The matching can be chosen to be case sensitive or not. As an example *house* matches any name that contains "house", while floor* matches any name starting with "floor".

Select Camera Select the active camera.

Mirror `Shift-Ctrl-M` Select the Mirror objects of the selected object eg. *L.sword* → *R.sword*.

Random Randomly selects unselected objects based on percentage probability on currently active layers. On selecting the command a numerical selection box becomes available in the *Tool Shelf*. It is important to note that the percentage represents the likelihood of an unselected object being selected and not the percentage amount of objects that will be selected.

Inverse `Ctrl-I` Selects all objects that were not selected while deselecting all those which were.

(De)select All `A` If anything was selected it is first deselected. Otherwise it toggles between selecting and deselecting every visible object.

Transform

Introduction

Transformations refer to a number of operations that can be performed on a selected Object or Mesh that alters its position or characteristics.

Each object can be moved, rotated and scaled in *Object Mode*. However, not all of these transformations have an effect on all objects. For example, scaling a camera has no effect on the render dimensions.

Basic transformations include:

- Grabbing (moving)
- Rotating
- Scaling

These three transforms are the three big ones however, more, advanced transformations can be found in the *Advanced Transformations* section.

For making other changes to the geometry of editable objects, you should use *Edit Mode*.

Once you have added a basic object, you remain in *Object Mode*.

You can switch between *Object Mode* and *Edit Mode* by pressing `Tab`.

The object's wireframe should now appear orange. This means that the object is now selected and active.

The Fig. Selected object image shows both the solid view and wireframe view of the default cube. To switch between wireframe and solid view, press `Z`.

Basic Transformations

Grab/Move

Reference

Mode: Object Mode, Edit Mode, and Pose Mode for the 3D View;
Menu: Context sensitive, *Object Based* → *Transform* → *Grab/Move*
Hotkey: `G`

In Object Mode, the grab/move option lets you translate (move) objects. It also lets you translate any elements that make up the object within the 3D space of the active 3D View. Grab/Move works similarly here as it does in the Node Editor, Graph Editor, UV/Image Editor, Sequencer, etc.

Pressing G activates "Grab/Move" transformation mode. The selected object or element then moves freely according to the mouse pointer's location and camera.

You can also move an object by clicking and holding RMB on the object to move it. To confirm the action, press LMB.

While Grab/Move is active, the amount of change in the X, Y, and Z coordinates is displayed at the bottom left corner of the 3D View editor.

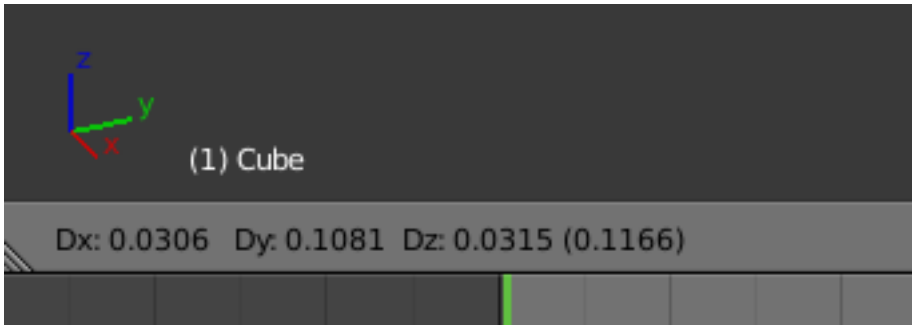


Fig. 2.85: Translation Display.

Tip: Moving an object in Object Mode changes the object's origin. Moving the object's vertices/edges/faces in Edit Mode does not change the object's origin.

Rotate

Reference

Mode: Object and Edit Modes
Menu: *Object/Mesh/Curve/Surface* → *Transform* → *Rotate*
Hotkey: R

Rotation is also known as a spin, twist, orbit, pivot, revolve, or roll and involves changing the orientation of elements (vertices, edge, face, Object etc) around one or more axes or the element's *Pivot Point*.

The amount of rotation will be displayed in the footer of the 3D View editor.

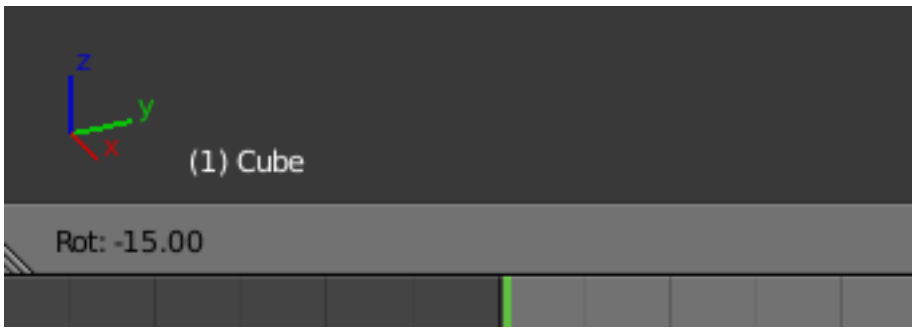


Fig. 2.86: Rotation values.

Trackball Rotation

A free rotation mode. Press R, R to enable Trackball rotation.

Scale

Reference

Mode: Object and Edit Modes
Menu: *Object/Mesh/Curve/Surface* → *Transform* → *Scale*
Hotkey: S

Pressing S will enter the *Scale* transformation mode where the selected element is scaled inward or outward according to the mouse pointer's location. The element's scale will increase as the mouse pointer is moved away from the Pivot Point and decrease as the pointer is moved towards it. If the mouse pointer crosses from the original side of the *Pivot Point* to the opposite side, the scale will continue in the negative direction and flip the element.

Fig. 2.87: Basic scale usage. From left to right, the panels show: the original Object, a scaled down Object, a scaled up Object and a scale-flipped Object.

The amount of scaling will be displayed in the footer of the 3D View editor.

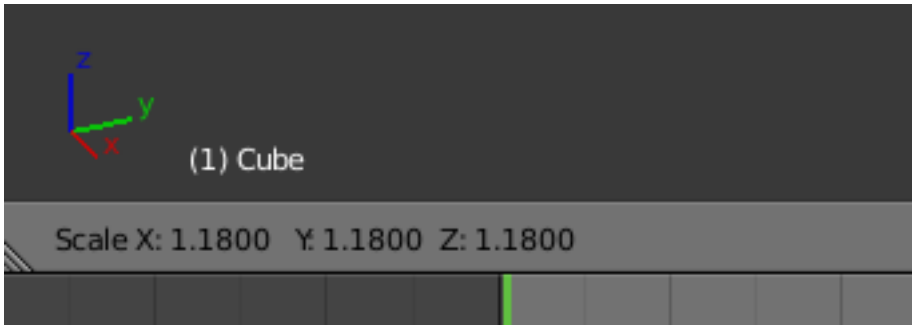


Fig. 2.88: Scale values.

Common Options

There are multiple ways to transform an element which include:

- The keyboard shortcut.
- The menu in the header or the Transform panel in the Tool Shelf.
- The *3D Transform Manipulator* widget.
- The *Transform panel* in the Properties region or the Object tab.

Confirm and Cancel

LMB click to accept changes. This behavior can be changed globally by activating *Release Confirms* in the *User Preferences*, so that a single RMB drag can be used to move and confirm.

To cancel the transformation press RMB or Esc instead. This will reset the object or element to its original state.

See also:

Using combination of shortcuts gives you more control over your transformation. See *Transform Control*.

Operator Panel

In the case of the 3D View, there is the possibility to tweak the operation once accepted, using the specific Operator panel corresponding to the tool.

Value The amount of the transformation.

Vector, Angle

Constrain Axis Used to constraint the transformation to one or more axes.

X, Y, Z

Orientation Shows the *Orientations* of the constraint axes.

Proportional Editing, Falloff, Size Activates/deactivates *Proportional Editing* and configures the type *Falloff* and *Size* of the *Proportional Edit* tool.

Edit Grease Pencil ToDo.

Edit Texture Space This checkbox lets you apply the transformation on the *texture space*, instead of the object or element itself. Only available in translation and scale.

Confirm on Release Shows if either the operation was drag-and-release or move-and-confirm.

Workflow

Using Keyboard Shortcuts

1. Use RMB to select the elements you want to transform.
2. Tap G, or R, or S once to enter the transformation mode.
3. Transform the elements by moving the mouse.
4. LMB click to accept changes.

Mirror

Reference

Mode: Object and Edit Modes

Menu: *Object/Mesh → Mirror*
Hotkey: `Ctrl-M`

Mirroring an Object or Mesh selection will create a reversed version of the selection. The position of the mirrored version of the selection is determined by the *Pivot Point*. A common use of mirroring is to model half an object, duplicate it and then use the mirror transform to create a reversed version to complete the model.

Note: Mirrored duplicates can also be created with a *Mirror Modifier*.

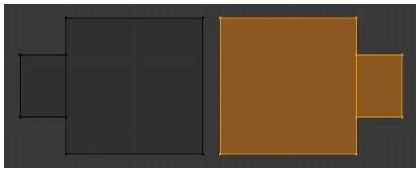


Fig. 2.89: Mirroring a Selection.

Usage

To mirror a selection along a particular global axis press: `Ctrl-M`, followed by `X`, `Y` or `Z`. The image *Mirroring a Selection* shows the results of this action after a mesh element has been duplicated.

In Mesh mode, you can mirror the selection on the currently selected *Transform Orientations* by pressing the appropriate axis key a second time. For example, if the Transform Orientation is set to *Normal*, pressing: `Ctrl-M`, followed by `X` and then `X` again will mirror the selection along the X-axis of the *Normal Orientation*.

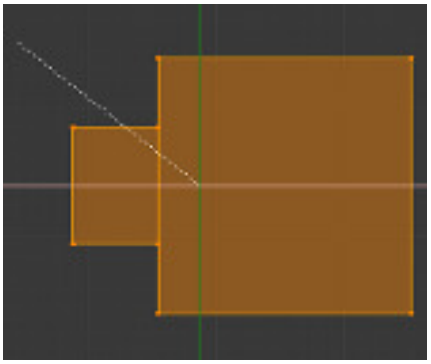


Fig. 2.90: Interactive Mirror.

You can alternatively hold the `MMB` to interactively mirror the object by moving the mouse in the direction of the mirror axis.

Duplication

There are two types of object duplication, being *Duplicate* and *Linked Duplicates* which instance their Object Data.

Duplicate

Reference

Mode: Edit and Object Modes
Menu: *Object* → *Duplicate*
Hotkey: `Shift-D`

This will create a visually-identical copy of the selected object(s). The copy is created at the same position as the original object and you are automatically placed in *Grab* mode. See the examples below.

This copy is a new object, which shares some data-blocks with the original object (by default, all the Materials, Textures, and F-Curves), but which has copied others, like the mesh, for example. This is why this form of duplication is sometimes called "shallow link", because not all data-blocks are shared; some of them are "hard copied"!

Tip: You can choose which types of data-block will be linked or copied when duplicating: in the *User Preferences*.

Examples

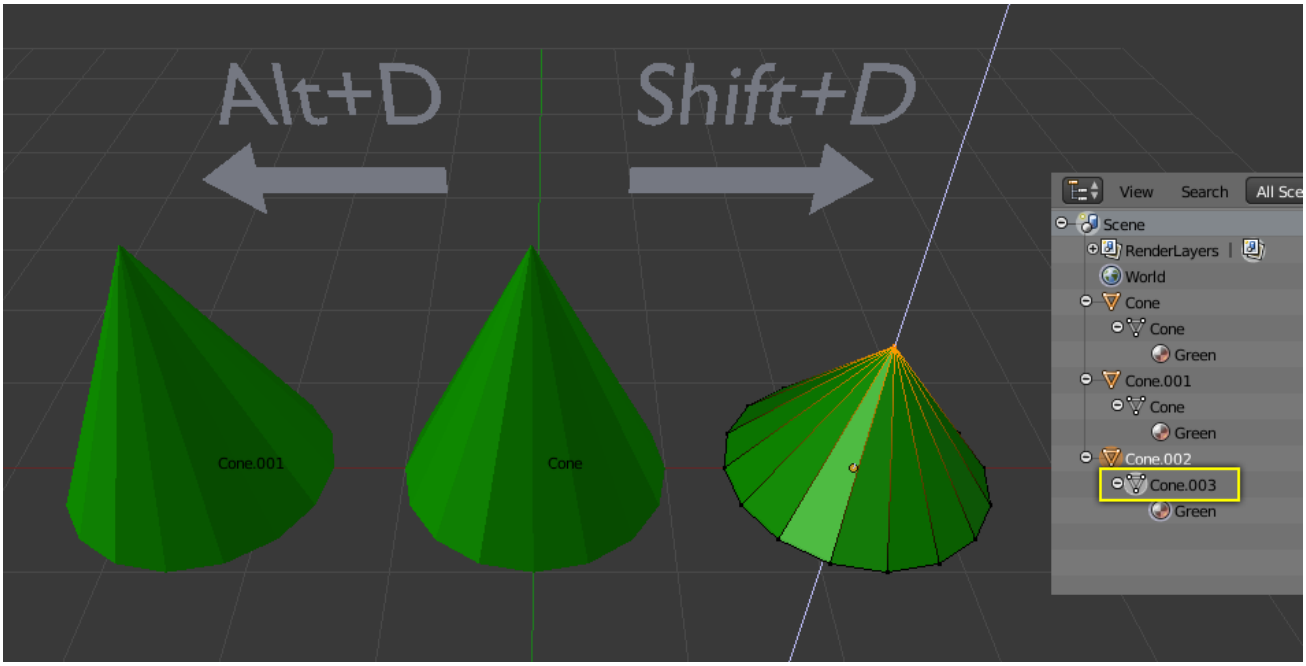


Fig. 2.91: The mesh `Cone.006` of object `Cone.002` is being edited. The mesh's Unique data-block ID name is highlighted in the Outliner.

The cone in the middle has been (1) link duplicated to the left and (2) duplicated to the right.

- The duplicated right cone is being edited, the original cone in the middle remains unchanged. The mesh data has been copied, not linked.
- Likewise, if the right cone is edited in object mode, the original cone remains unchanged. The new object's transform properties or data-block is a copy, not linked.
- When the right cone was duplicated, it inherited the material of the middle cone. The material properties were linked, not copied.

See above if you want separate copies of the data-blocks normally linked.

Linked Duplicates

Reference

Mode: Object Mode
Menu: *Object* → *Duplicate Linked*
Hotkey: `Alt-D`

You also have the choice of creating a *Linked Duplicate* rather than a *Duplicate*; this is called a deep link. This will create a new object with **all** of its data linked to the original object. If you modify one of the linked objects in *Edit Mode*, all linked copies are modified. Transform properties (object data-blocks) still remain copies, not links, so you still can rotate, scale, and move freely without affecting the other copies. Reference Expl. *Duplicate Example* for the discussions below.

Hint: If you want to make changes to an object in the new linked duplicate independently of the original object, you will have to manually make the object a "single-user" copy by `LMB` the number in the *Object Data* panel of the Properties editor. (See *Data-Block Menu*.)

Examples

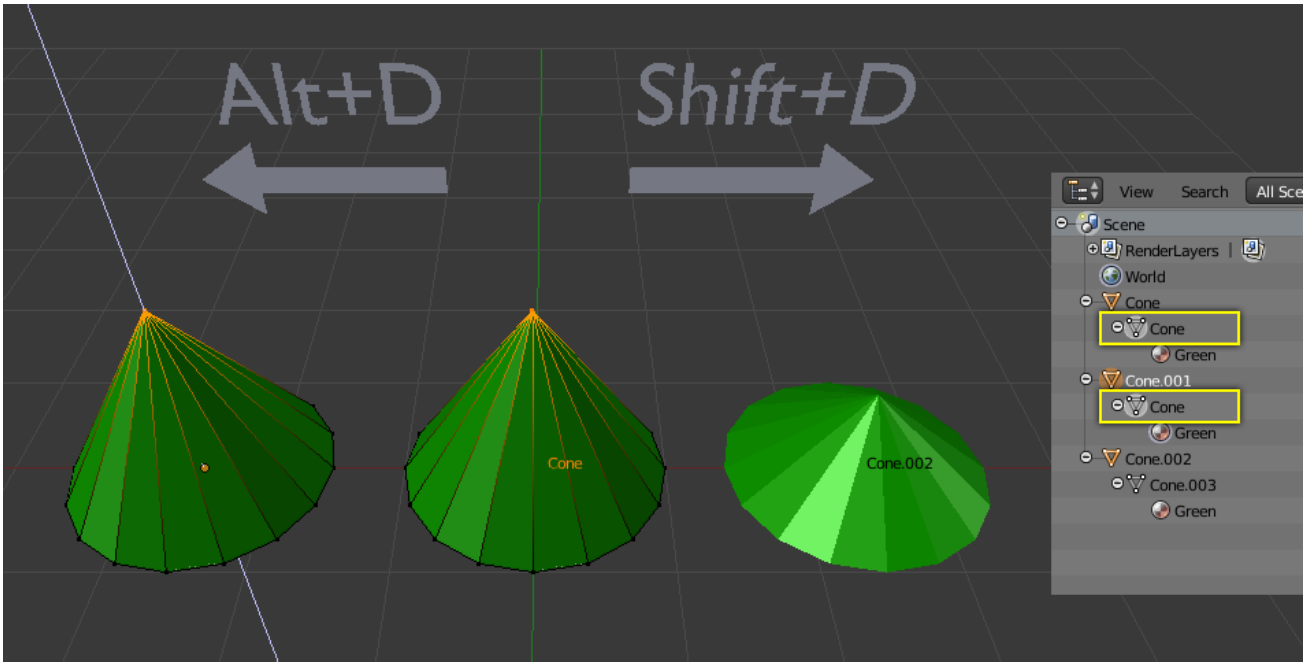


Fig. 2.92: The object `Cone.001` was linked duplicated. Though both these cones are separate objects with unique names, the single mesh named Cone, highlighted in the Outliner, is shared by both.

The left cone is a *Linked Duplicate* of the middle cone (using `Alt-D`).

- As a vertex is moved in *Edit Mode* in one object, the same vertex is moved in the original cone as well. The mesh data are links, not copies.
- In contrast, if one of these two cones is rotated or rescaled in object mode, the other remains unchanged. The transform properties are copied, not linked.
- As in the previous example, the newly created cone has inherited the material of the original cone. The material properties are linked, not copied.

A common table has a top and four legs. Model one leg, and then make linked duplicates three times for each of the remaining legs. If you later make a change to the mesh, all the legs will still match. Linked duplicates also apply to a set of drinking glasses, wheels on a car... anywhere there is repetition or symmetry.

Copying & Linking Objects Between Scenes

Sometimes you may want to link or copy objects between scenes. This is possible by first selecting objects you want to link and then using: *Object* → *Make Links* → *Object to Scene*.

This makes the same object exist in two different scenes at once, including its position and animation data. You can tell this is a *multi-user* object by the blue color of its center-circle.

If you do not want the objects to be shared between the scenes, you can make them *Single-User* by using: *Object* → *Make Single User* → *Object*.

Further information on working with scenes can be found *here*.

Linked Library Duplication

Reference

Menu: *File* → *Link Append*
Hotkey: `Shift-F1`

Linked Libraries are also a form of duplication. Any object or data-block in other blend-files can be reused in the current file.

Hint: If you want transform properties (i.e. object data-blocks) to be "linked", see the page on *parenting*.

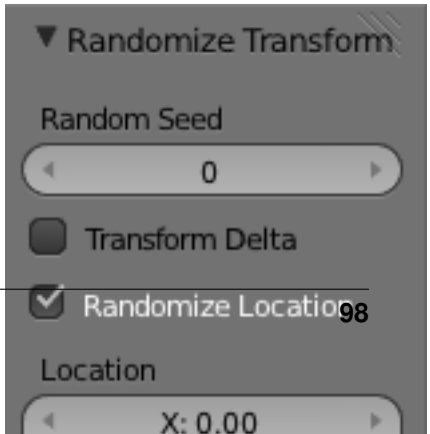
Transform Tools

Randomize Transform

Reference

Mode: Object Mode and Edit Mode
Menu: *Object* → *Transform* → *Randomize Transform*, *Mesh* → *Transform* → *Randomize*

This tool allows you to apply a randomized transformation.



Object Mode

In Object Mode it randomizes the translate, rotate, and scale values to an object or multiple objects. When applied on multiple objects, each object gets its own seed value, and will get different transform results from the rest.

Options

Random Seed The random seed is an offset to the randomized transformation. A different seed will produce a new result.

Transform Delta Randomize *Delta Transform* values instead of regular transform.

Randomize Location Randomize Location values.

Location The maximum distances the objects can move along each axis.

Randomize Rotation Randomize rotation values.

Rotation The maximum angle the objects can rotate on each axis.

Randomize Scale Randomize scale values.

Scale Even Use the same scale for each axis.

Scale The maximum scale randomization over each axis.

Edit Mode

The *Randomize* tool in Edit Mode allows you to displace the vertices of a mesh along their normal.

Options

Amount Distance of the displacement.

Uniform Adds a random offset of the amount.

Normal Adds a random offset to the displacement normal.

Random Seed The random seed is an offset to the random transformation. A different seed will produce a new result.

Align Objects

Reference

Mode: Object Mode
Menu: *Object* → *Transform* → *Align Objects*

The align tool is used to align multiple selected objects so they line up on a specified axis.

Options

High Quality Uses more precise math to better determine the locations for the objects.

Align Mode TODO.

Relative To TODO.

Align X, Y, Z Chooses which axis to align the selected objects on.

Object Origin

Each object has an origin point. The location of this point determines where the object is located in 3D space. When an object is selected, a small circle appears, denoting the origin point. The location of the origin point is important when translating, rotating or scaling an object. See *Pivot Points* for more.

The color of the origin is set by the *State Colors* and the size can be changed in the *Interface tab* of the User Preferences.

Moving Object Centers

Object Centers can be moved to different positions through *3D View editor* → *Transform* → *Origin* (press `T` to open panel):

Geometry to Origin Move model to origin and this way origin of the object will also be at the center of the object.

Origin to Geometry Move origin to the center of the object and this way origin of the object will also be at the center of the object.

Origin to 3D Cursor Move origin of the model to the place of the 3D cursor.

Origin to Center of Mass Move origin to calculated center of mass of model.

Transform Control

Transform controls can be used to modify and control the effects of the available transformations.

The following pages detail the available control options:

Precision

Introduction

Reference

Mode: Object and Edit Modes
Hotkey: `Ctrl` and/or `Shift`

Holding `Ctrl` during a transform operation (such as grab, rotate or scale) will toggle *Transform Snapping*. When the *Snap Element* is set to *Increment*, this allows the transformation to be performed in discrete amounts.

Holding `Shift` during a transform operation will transform the object at 1/10th the speed, allowing much finer control.

The magnitude of the transformation can be viewed in the 3D View header in the bottom left hand corner. Releasing `Ctrl` or `Shift` during the transformation will cause the movement to revert back to its normal mode of operation.

Holding both `Ctrl-Shift` enables precise snap. This option will move the object with high precision along with the snapping constraint.

Note: The snapping behaviors described on this page **only** apply when *Increment Snap* is selected.

Usage

With hotkeys

Press `G`, `R` or `S` and then hold either `Ctrl`, `Shift` or `Ctrl-Shift`.

With the Transform Manipulator

Hold `Ctrl`, `Shift` or `Ctrl-Shift` and click on the appropriate manipulator handle. Then move the mouse in the desired direction. The reverse action will also work i.e. clicking the manipulator handle and then holding the shortcut key for precision control.

See also:

Read more about the Transform Manipulator

Tip: Combining with other controls

All of the precision controls detailed on the page can be combined with the *Axis Locking* controls and used with the different *Pivot Points*.

Holding CTRL

Grab/move transformations

For grab/move operations at the default zoom level, holding `Ctrl` will cause your selection to move by increments of 1 Blender Unit (1 BU) (i.e. between the two light gray lines). Zooming in enough to see the next set of gray lines will now cause `Ctrl` movements to occur by 1/10 of a BU. Zooming in further until the next set of gray lines becomes visible will cause movement to happen by 1/100 of a BU and so on until the zoom limit is reached. Zooming out will have the opposite effect and cause movement to happen by increments of 10, 100 etc BU.

See also:

Read more about Zooming

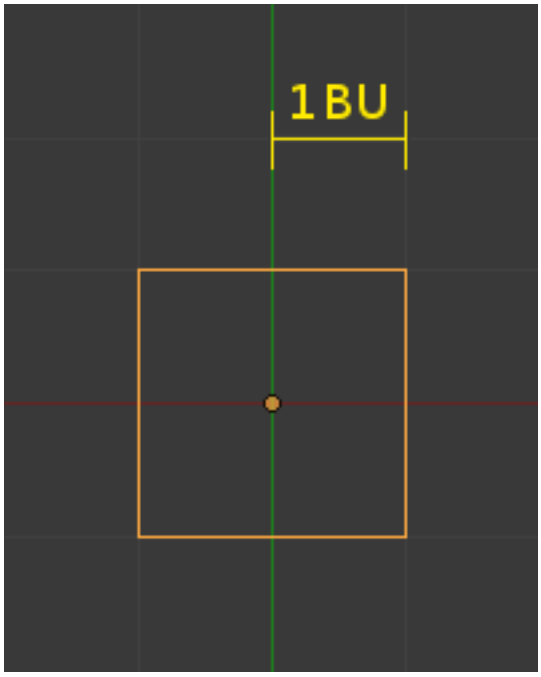


Fig. 2.94: 1 Blender Unit (default zoom level).

Rotation transformations

Holding `Ctrl` will cause rotations of 5 degrees.

Scale transformations

Holding `Ctrl` will cause size changes in increments of 0.1 BU.

Note: Snapping modes

Note that if you have a *Snap Element* option enabled, holding `Ctrl` will cause the selection to snap to the nearest element.

Read more about Snapping

Holding SHIFT

Holding `Shift` during transformations allows for very fine control that does not rely on fixed increments. Rather, large movements of the mouse across the screen only result in small transformations of the selection.

In rotation mode the selected element will be rotate in 0.01 degree increments.

Holding CTRL and SHIFT

Grab/move transformations

For grab/move operations at the default zoom level, holding `Ctrl-Shift` will cause your selection to move by increments of 1/10 Blender Units. Holding `Ctrl-Shift` at any zoom level will cause the transformation increments to always be 1/10 of the increment if you were only holding `Ctrl`.

Rotation transformations

Holding `Ctrl-Shift` will cause rotations of 1 degree.

Scale transformations

Holding `Ctrl-Shift` will cause size changes in 0.01 BU increments.

Numeric Input

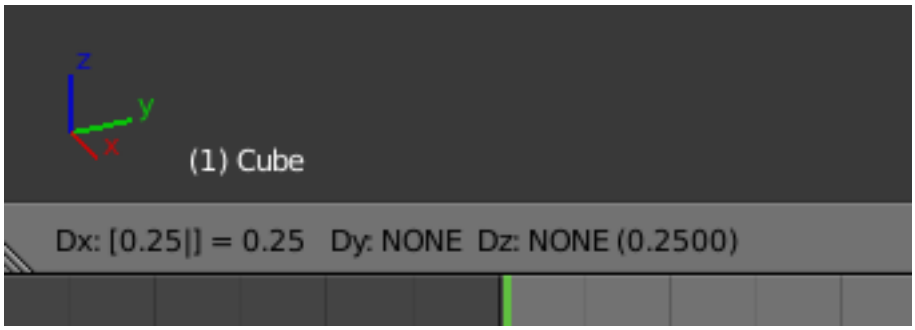


Fig. 2.95: Numeric input displayed in the 3D View footer.

Using the mouse for transformations is convenient, but if you require more precise control, you can also enter numeric values. After pressing `G`, `R`, `S` type a number to indicate the magnitude of the transformation.

You can see the numbers you enter in the bottom left hand corner of the 3D View header. Negative numbers and decimals can be entered by pressing `Minus` and `.` respectively.

- Hitting `Backspace` during number entry and deleting the number removes the numerical specification option but the object will remain constrained to the same axis.
- Hitting `/` during number entry switches the number being entered to its reciprocal, e.g. `2 /` results in 0.5 (1/2); `2/0` results in 0.05 (1/20).

Translation

To move Objects, vertices, faces or edges select the element, press `G` and then type a number. By default and with no other key presses, movement will occur along the X-axis. To confirm the movement, press `Return` or `LMB`. To cancel the movement, press `Esc` or `RMB`.

Chaining

To enter numeric values for multiple axes, use `Tab` after entering a value for the axis. e.g. To move an Object, one (1) Blender unit on all three axes press: `G 1` and `Tab 1` and `Tab 1`. This will move the element one unit along the X-axis, followed by the Y-axis and then the Z-axis.

You can also combine numeric input with *Axis Locking* to limit movement to a particular axis.

Rotation

To specify a value for clockwise rotation, press `R`, (`0 - 9`), then `Return` to confirm. To specify counter-clockwise rotation press `R`, `Minus`, (`0 - 9`), then `Return` to confirm. Note that 270 degrees of clockwise rotation is equivalent to -90 degrees of counter-clockwise rotation.

Scaling

Objects, faces and edges can be scaled by pressing `S`, (`0 - 9`), then `Return` to confirm. Scaling transformations can also be constrained to an axis by pressing `X`, `Y`, `Z` after pressing `S`. Essentially, scaling with numeric values works in almost identical fashion to translation. The primary difference is that by default, scaling applies equally to all three axes. e.g. pressing `S 0 5`, `Return` will scale an Object by 0.5 on all three axes.

Tip: Numeric input can also be inputed in the *Properties* region.

Axis Locking

This option limits the transformation to the specified axis.

Transformations (translation/scale/rotation) in *Object Mode* and *Edit Mode*, as well as extrusion in *Edit Mode*) can be locked to particular axis relative to the current *transform orientation*. By locking a transformation to a particular axis you are restricting transformations to a single dimension.

Usage

A locked axis will display in a brighter color than an unlocked axis. For example in the image to the right, the Z axis is drawn in light blue as movement is constrained to this axis. This example, can be achieved in two ways:

Hotkey

The axis of movement can be changed at any time during transformation by typing `X`, `Y`, `Z`.

None

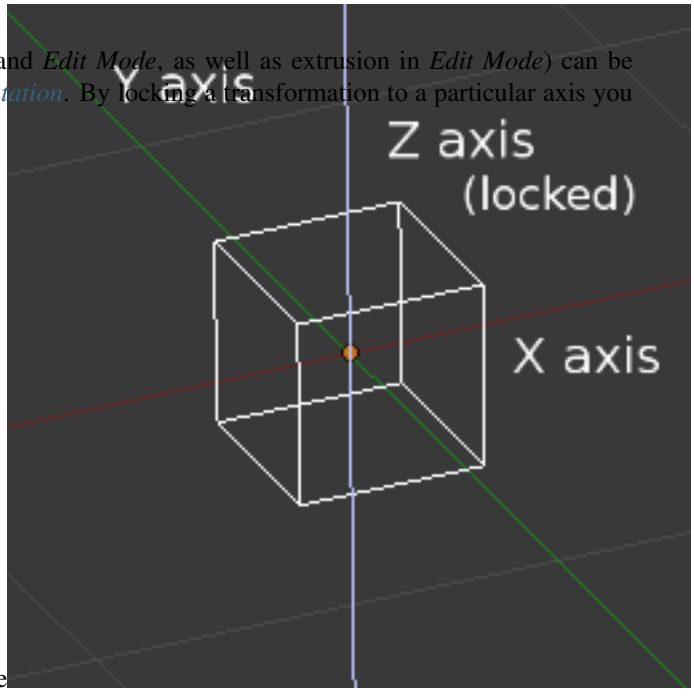


Fig. 2.96: Axis locking.

Pointing

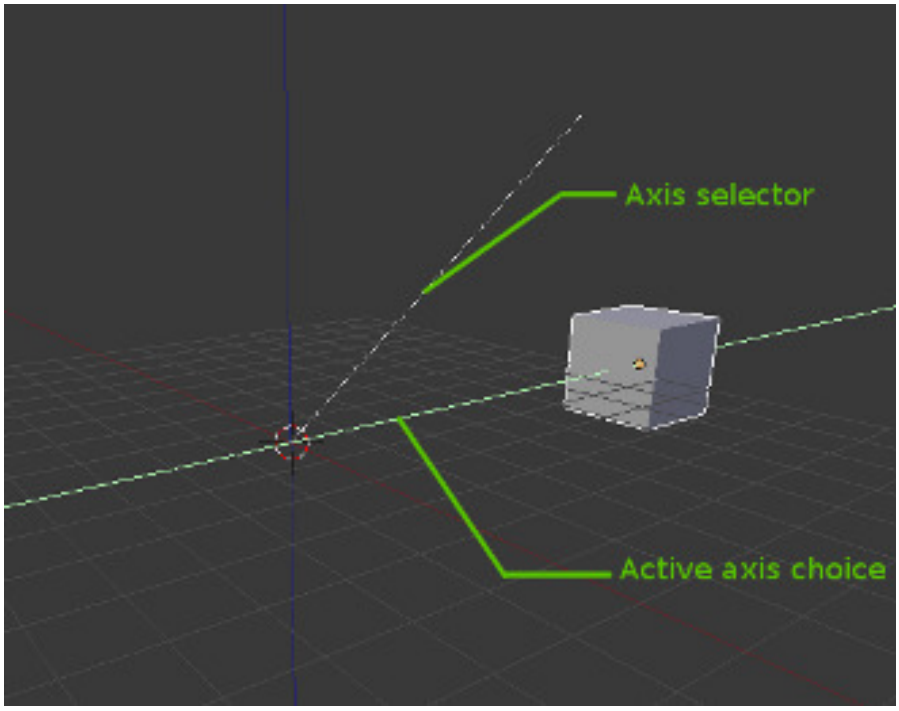


Fig. 2.97: Axis-Constraint in action.

Holding MMB after starting a transformation lets you select an axis to constrain to. A visual option to constrain the translation will be available, showing the three axes in the 3D View space. A dotted white line is used as a pointer. The axis of choice to confirm the operation will depend on the highlighted axis about which the MMB is released.

When you already moved the mouse in the desired direction, pressing MMB will lock to the axis in which was pointed at.

Axis locking types

Axis locking

Reference

Mode: Object and Edit Modes (translate, rotate, scale, extrude)
Hotkey: X, Y, Z or MMB after moving the mouse in the desired direction.

Axis locking limits the transformation to a single axis (or forbids transformations along two axes). An object, face, vertex or other selectable item will only be able to move, scale or rotate in a single dimension.

Plane locking

Reference

Mode: Object and Edit Modes (translate, scale)
Hotkey: `Shift-X`, `Shift-Y`, `Shift-Z` or `Shift-MMB` after moving the mouse in the desired direction.

Plane locking locks the transformation to *two* axes (or forbids transformations along one axis), thus creating a plane in which the element can be moved or scaled freely. Plane locking only affects translation and scaling.

Note that for rotation, both axis and plane locking have the same effect because a rotation is always constrained around one axis. *Trackball* type rotations `R-R` cannot be locked at all.

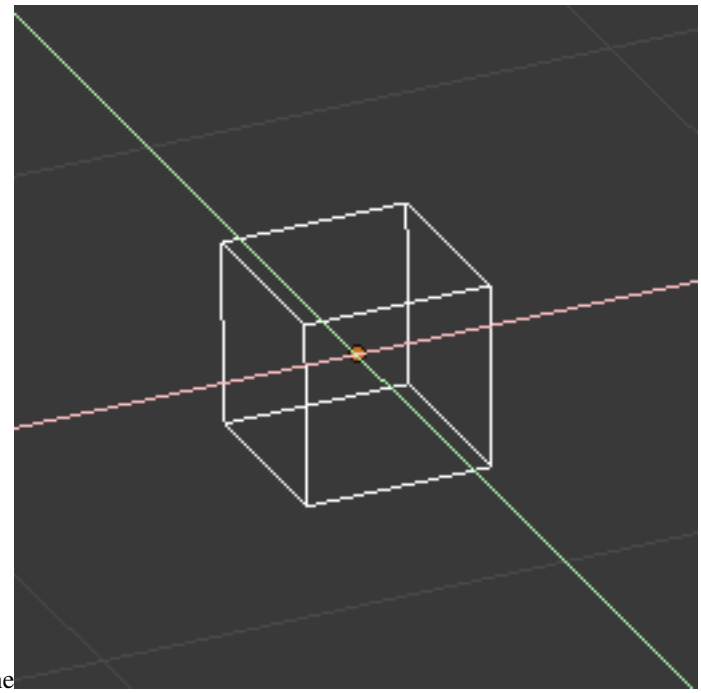

None

Fig. 2.98: Plane locking.

Axis locking modes

A single key press constrains movement to the corresponding *Global* axis. A second key press of the *same* key constrains movement to the current transform orientation selection (except if it is set to *Global*, in which case the *Local* orientation is used). Finally, a third key press of the same key removes constraints.

The orientation can be set in the *Transform Orientation* selector of the 3D View header.

For example, if the current transform orientation is set to *Normal*, pressing `G` to start translation, followed by `Z` will lock translation in the Z direction relative to the *Global* orientation, pressing `Z` again will lock translation to the Z axis relative to the *Normal* orientation. Pressing `Z` again will remove all constraints. The current mode will be displayed in the left hand side of the *3D View header*.

As can be seen in the *Axis locking modes* image, the direction of the transform also takes into account the selection.

Note that using a locked axis does not prevent you from using the keyboard to enter *numeric transformation* values.

Snapping

There are two types of snap operations that you can use in Blender. The first type snaps your selection or cursor to a given point while the second type is used during transformations (translate, rotate, scale) and snaps your selection to elements within the scene.

Reference

Mode: Object and Edit Mode

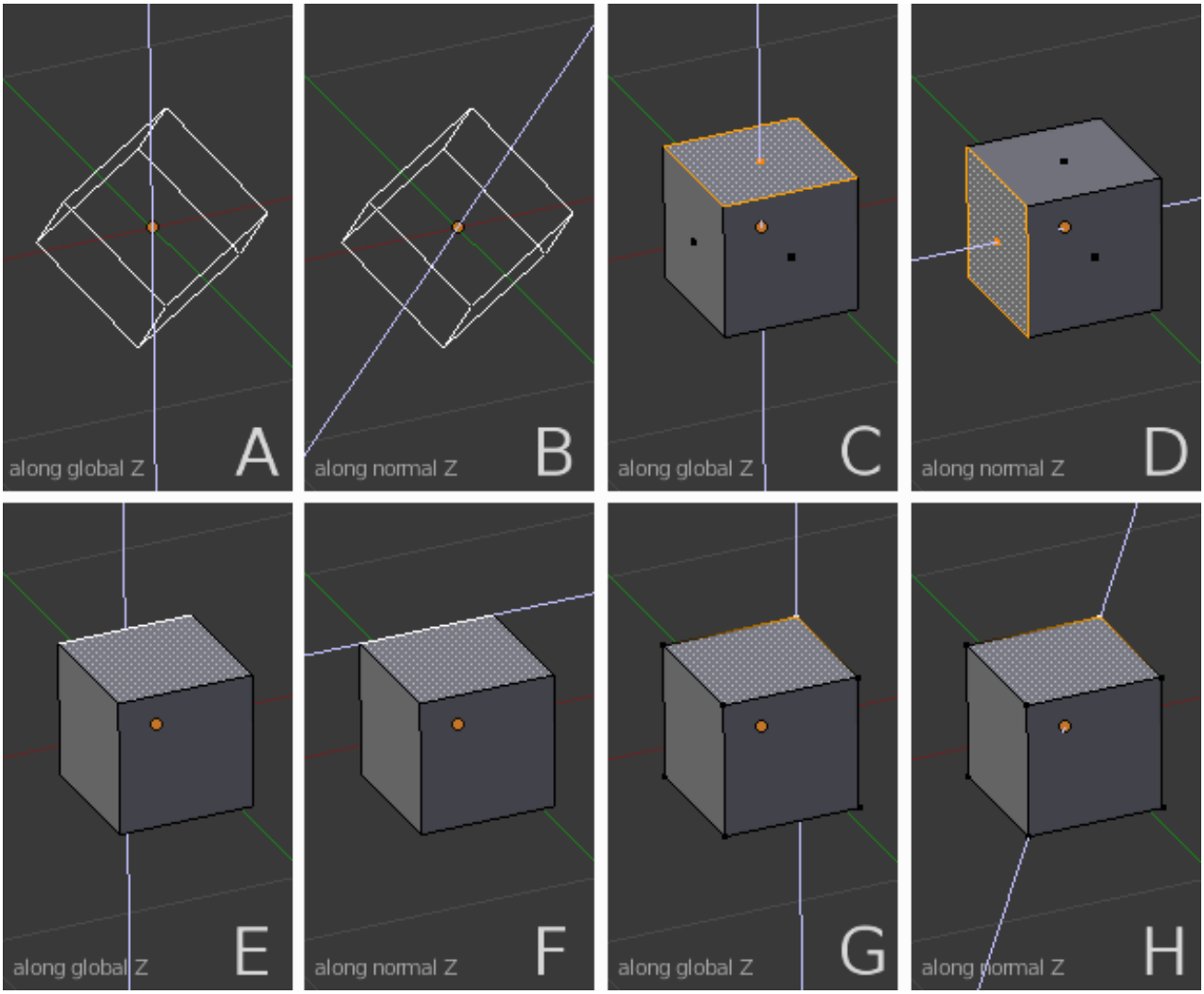


Fig. 2.99: Axis locking modes.
A and B show Z axis locking in *Global* and *Normal* orientations respectively. C and D show the same situation with face selection, E and F with edge selection and G and H with vertex selection.

Hotkey: `Shift-S`

The *Snap* menu (also available from the 3D header in both *Object Mode* and *Edit Mode Object* → *Snap* and *Mesh* → *Snap*). This menu provides a number of options to move the cursor or your selection to a defined point (the cursor, selection or the grid).

Selection to Grid Snaps the currently selected object(s) to the nearest grid point.

Selection to Cursor Snaps the currently selected object(s) to the cursor location.

Cursor to Selected Moves the cursor to the center of the selected object(s).

Cursor to Center Moves the cursor to the center of the grid.

Cursor to Grid Moves the cursor to the nearest grid point.

Cursor to Active Moves the cursor to the center of the active object.

Transform Snapping

The ability to snap Objects and Mesh element to various types of scene elements during a transformation is available by toggling the magnet icon (which will turn red) in the 3D View's header buttons.

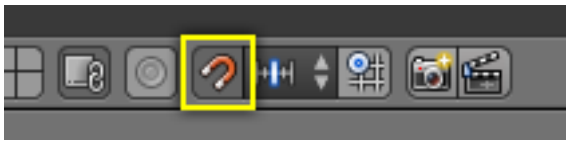


Fig. 2.100: Magnet icon in the 3D View header (red when enabled).

Snap Element

Volume Snaps to regions within the volume of the first Object found below the mouse cursor. Unlike the other options, this one controls the depth (i.e. Z-coordinates in current view space) of the transformed element. By toggling the button that appears to the right of the snap target menu (see below), target objects will be considered as a whole when determining the volume center.

Face Snap to the surfaces of faces in mesh objects. Useful for retopologizing.

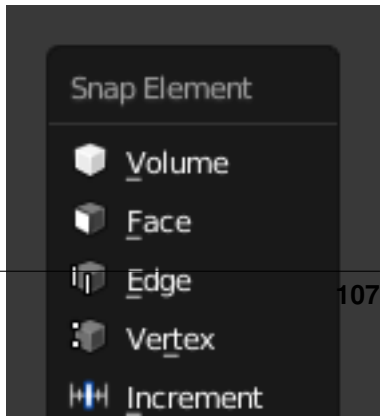
Edge Snap to edges of mesh objects.

Vertex Snap to vertices of mesh objects.

Increment Snap to grid points. When in Orthographic view, the snapping increment changes depending on zoom level.

> **Note:** In this context the grid does not mean the visual grid cue displayed. Snapping will use the resolution of the displayed grid, but all transformations are relative to the initial position (before the snap operation).

Snap Target



Snap target options become active when either *Vertex*, *Edge*, *Face*, or *Volume* is selected as the snap element. These determine what part of the selection snaps to the target objects.

Active Moves the active element (vertex in Edit Mode, object in Object Mode) to the target.

Median Moves the median of the selection to the target.

Center Moves the current transformation center to the target. Can be used with 3D cursor to snap with an offset.

Closest Moves the closest point of the selection to the target.

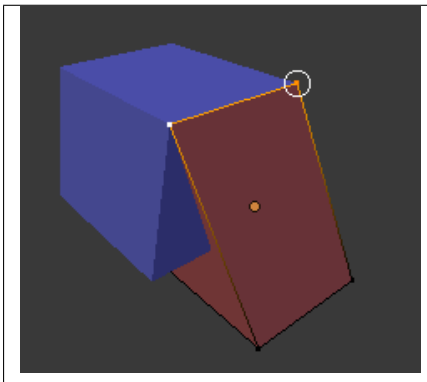


Fig. 2.102: Closest.

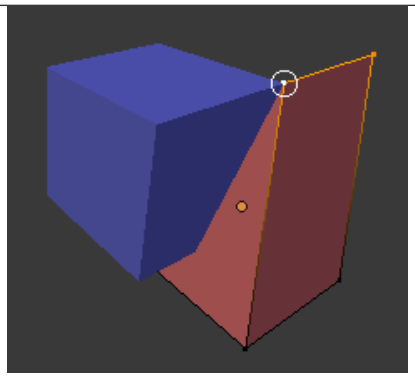


Fig. 2.103: Active.

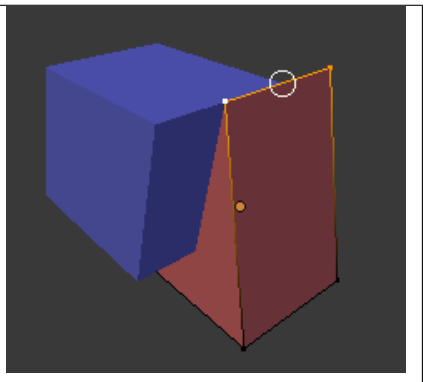


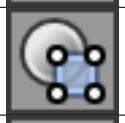
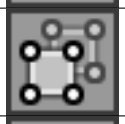
Fig. 2.104: Median.

Additional Snap Options

Fig. 2.105: Object Mode.

Fig. 2.106: Edit Mode.

As seen by the yellow highlighted areas in the image above, additional controls are available to alter snap behavior. These options vary between mode (Object and Edit) as well as Snap Element. The four options available are:

Icon	Details
	Align rotation with the snapping target.
	Project individual elements on the surface of other objects.
	Snaps elements to its own mesh.
	Consider Objects as whole when finding volume center.
	Snap to grid, instead of snapping in increments relative to the current location.

Multiple Snap Targets

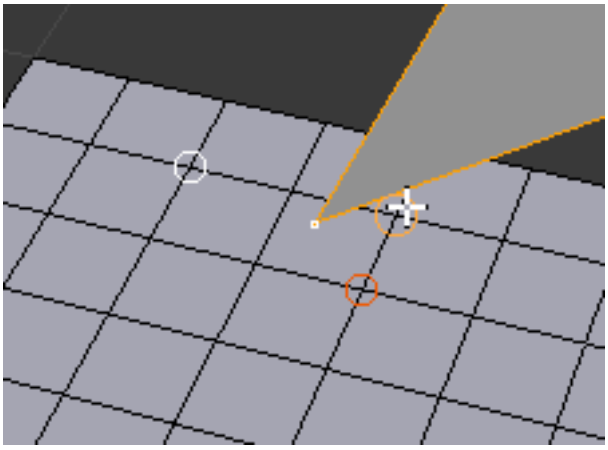


Fig. 2.107: Multiple snapping targets.

Once transforming a selection with Snapping on (not just when holding `Ctrl`), you can press `A` to mark the current snapping point, then proceed to mark as many other snapping points as you wish and the selection will be snapped to the average location of all the marked points.

Marking a point more than once will give it more weight in the averaged location.

Transformation Manipulators

Reference

Mode: Object and Edit Modes

Menu: 

Hotkey: `Ctrl-Spacebar`

The Transformation manipulator widgets allow mouse controlled translation, rotation and scaling in the 3D View. There is a separate manipulator for each operation. Each manipulator can be used separately or in combination with the others.

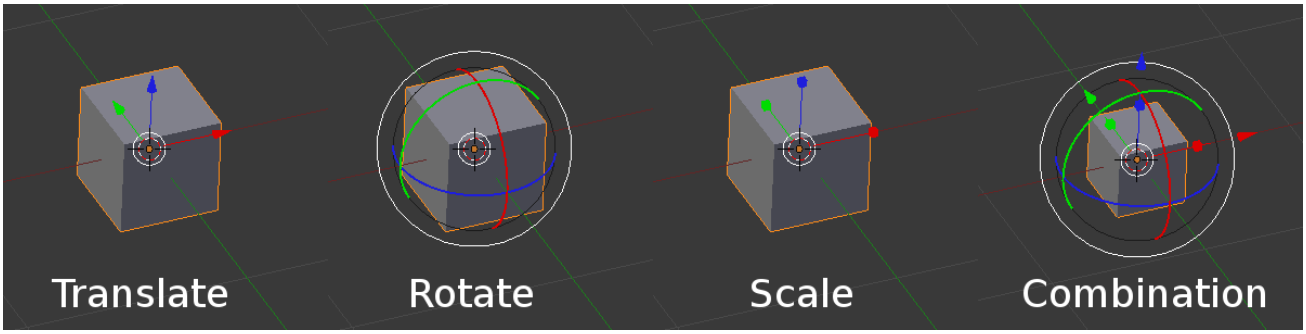


Fig. 2.108: The different Manipulators.

Header Controls

Manipulators can be accessed through the header of the *3D View*.

Axis Enable/disable the manipulators `Ctrl-Spacebar`.

Manipulators Toggles each of the manipulators. Clicking with `Shift-LMB` on multiple manipulator icons will combine the manipulators.

Arrow Translation.

Arc Rotation.

Box Scale.

Transform Orientation A select menu of the *Transform Orientations*.

Manipulator Controls

Basic

You can use the widget by dragging `LMB` one of the three colored axes. The transformation will be locked to the clicked axis.

Dragging the small white circle allows free transformation. In case of the rotation manipulator this starts a *trackball rotation*. The rotation manipulator contains another big outer white circle to activate free transformation.

Releasing the mouse confirms the operation (*confirm on release*).

Extended

The operations work in same way as described in *precision control* except:

Holding down `Shift` *after* you `LMB` the manipulator handle will constrain the action to smaller increments. Holding down `Shift` *before* you `LMB` click on one of the handles will cause the manipulator action to be performed relative to the other two axes. See *Plane locking*.

See also:

The *Manipulator Preferences*.

Clear Object transformations

Clearing transforms simply resets the transform values. The objects location and rotation values return to 0, and the scale returns to 1.

Reference

Mode: Object Mode
Menu: *Object* → *Clear* → *Clear Location/Clear Scale/Clear Rotation/Clear Origin*
Hotkey: `Alt-G`, `Alt-S`, `Alt-R`, `Alt-O`

Clear Options

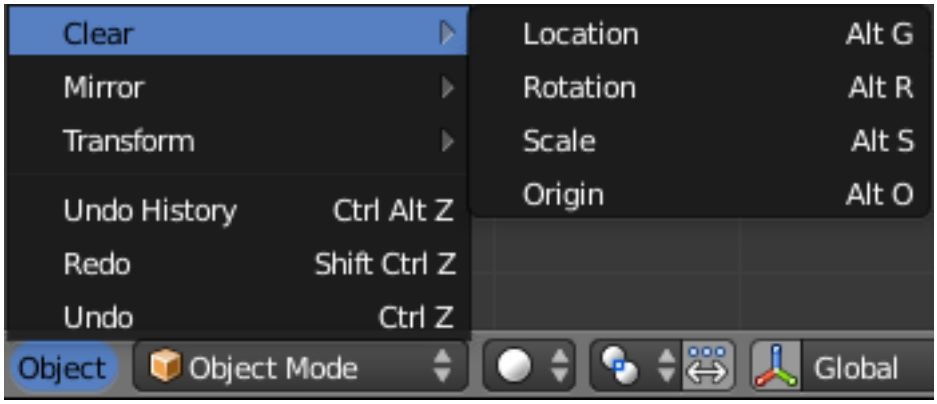


Fig. 2.109: Clear Transformation menu.

Clear Location **`Alt-G`** Clear (reset) the location of the selection. This will move the selection back to the coordinates (0, 0, 0).

Clear Scale **`Alt-S`** Clear (reset) the scale of the selection. This will resize the selection back to the size it was when created.

Clear Rotation **`Alt-R`** Clear (reset) the rotation of the selection. This will set the rotation of the selection to 0 degrees in each plane.

Clear Origin **`Alt-O`** Clear (reset) the origin of the Child objects. This will cause Child objects to move to the coordinates of the parent.

Apply Object Transformations

Reference

Mode: Object Mode
Menu: *Object* → *Apply* →
Hotkey: `Ctrl-A`

Applying transform values essentially resets the values of object's position, rotation, or scale, but does not actually do anything to the object. The center point is moved to the origin and the transform values are set to zero. In terms of scale, the scale values return to 1.

To apply a transform select the *Apply* sub-menu from the *Object menu* or use the shortcut `Ctrl-A` and select the appropriate transform to apply.

Make Duplicates Real unlinks linked duplicates so each duplicate now has its own data-block.

Apply Options

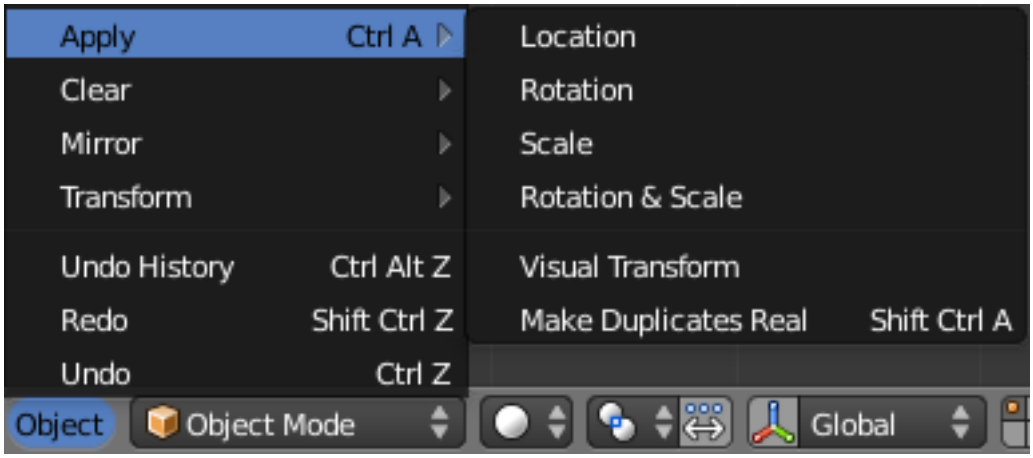


Fig. 2.110: Apply Transformation menu.

Apply Location `Ctrl-A` Apply (set) the location of the selection. This will make Blender consider the current location to be equivalent to 0 in each plane i.e. the selection will not move, the current location will be considered to be the "default location". The Object Center will be set to actual (0, 0, 0) (where the colored axis lines intersect in each view).

Apply Rotation `Ctrl-A` Apply (set) the rotation of the selection. This will make Blender consider the current rotation to be equivalent to 0 degrees in each plane i.e. the selection will not rotated, the current rotation will be considered to be the "default rotation".

Apply Scale `Ctrl-A` Apply (set) the scale of the selection. This will make Blender consider the current scale to be equivalent to 0 in each plane i.e. the selection will not scaled, the current scale will be considered to be the "default scale".

Apply Rotation and Scale `Ctrl-A` Apply (set) the rotation and scale of the selection. Do the above two applications simultaneously.

All Transforms to Deltas Converts all "normal" transformations to *Delta transforms*.

Animated Transform to Deltas Converts the "normal" transformation animations (animations done to the translation, scale, and rotation values) to *Delta transforms*. To use this tool simply select the object with the animations that you want to convert press `Ctrl-A` and select *Animated Transform to Deltas*.

Apply Visual Transform `Ctrl-A` Apply (set) the result of a constraint and apply this back to the Object's location, rotation and scale.

Make Duplicate Real `Shift-Ctrl-A` Make any duplicates attached to this Object real so that they can be edited.

Proportional Edit

Proportional Edit is a way of transforming selected elements (such as vertices) while having that transformation affect other nearby elements. For example, having the movement of a single vertex cause the movement of unselected vertices within a given range. Unselected vertices that are closer to the selected vertex will move more than those farther from it (i.e. they will move proportionally relative to the location of the selected element). Since proportional editing affects the nearby geometry, it is very useful when you need to smoothly deform the surface of a dense mesh.

Note: Sculpting

Blender also has *Sculpting* that contains brushes and tools for proportionally editing a mesh without seeing the individual vertices.

Object Mode

Reference

Mode: Object Mode

Menu: Via the  icon in the header indicated by the yellow square in the below image.

Hotkey: O

Proportional editing is typically used in *Edit Mode*, however, it can also be used in *Object Mode*. In *Object Mode* the tool works on entire objects rather than individual mesh components. In the image below, the green cube is the active Object, while the red and blue cubes are located within the proportional edit tool's radius of influence. When the green cube is moved to the right, the other two cubes follow the movement.

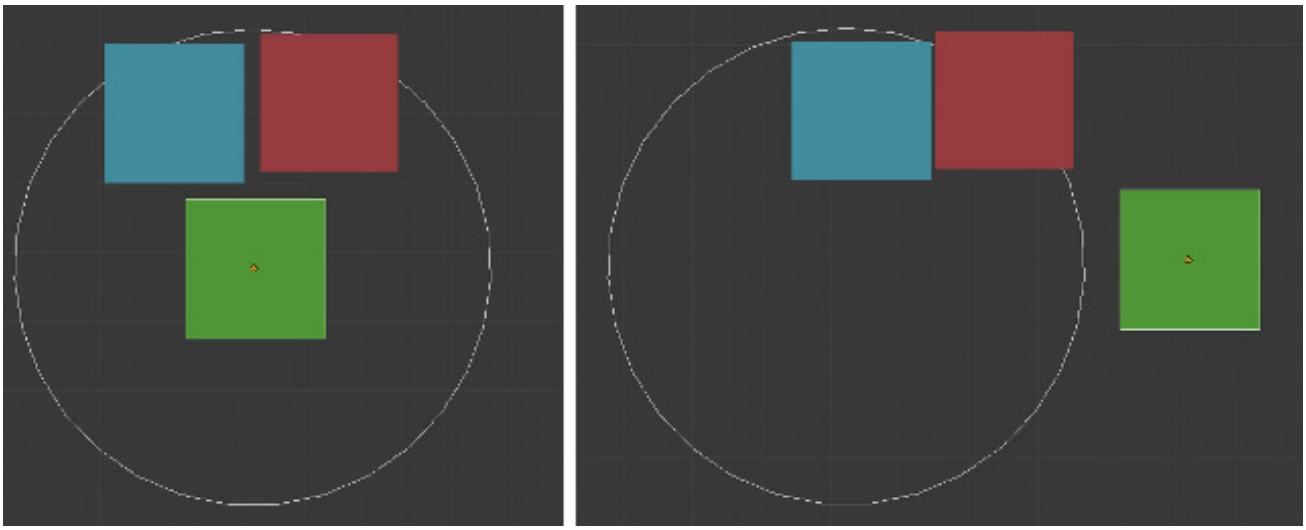


Fig. 2.111: Proportional editing in Object Mode.

Edit Mode

Reference

Mode: Edit Mode
Menu: *Mesh* → *Proportional*
Editing and via the  highlighted icon
Hotkey: `O`, `Alt-O`, `Shift-O`

When working with dense geometry, it can become difficult to make subtle adjustments to the vertices without causing visible lumps and creases in the model's surface. When you face situations like this the proportional editing tool can be used to smoothly deform the surface of the model. This is done by the tool's automatic modification of unselected vertices within a given range.

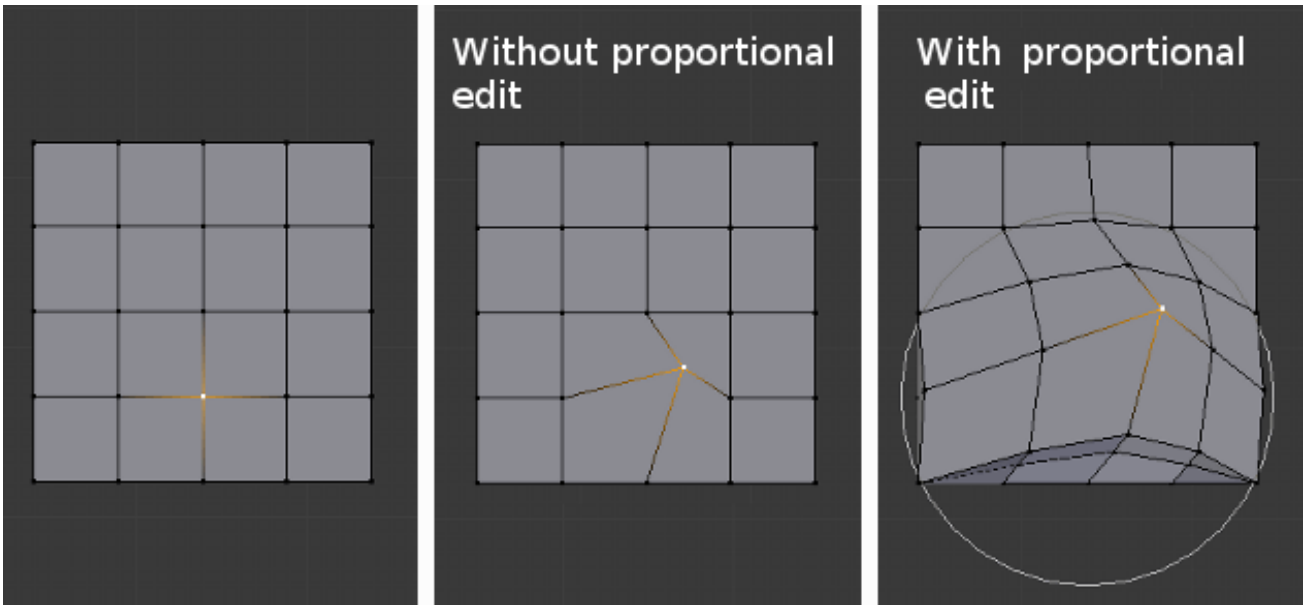


Fig. 2.112: Proportional editing in Edit Mode.

Influence

You can increase or decrease the radius of the proportional editing influence with the mouse wheel `WheelUp`, `WheelDown` or `PageUp`, `PageDown` respectively. As you change the radius, the points surrounding your selection will adjust their positions accordingly.

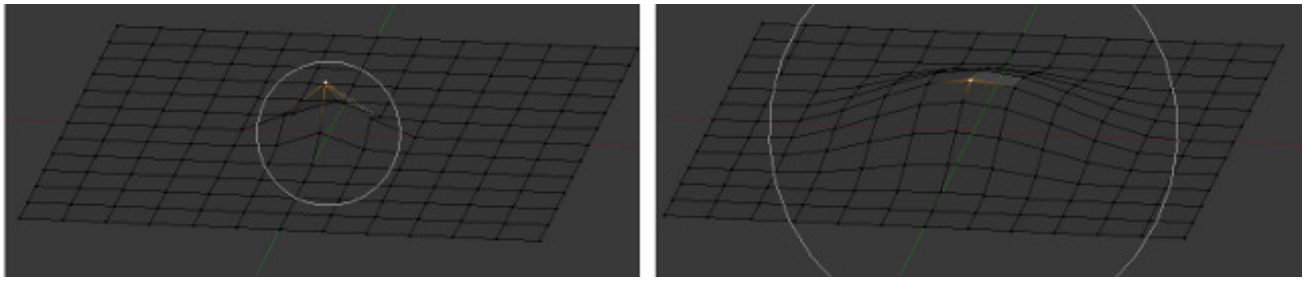


Fig. 2.113: Influence circle.

Options

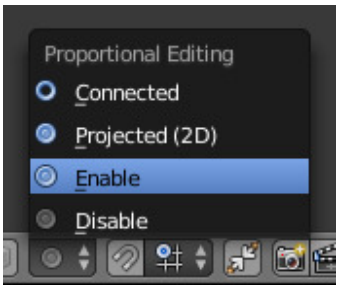


Fig. 2.114: Proportional Editing tool.

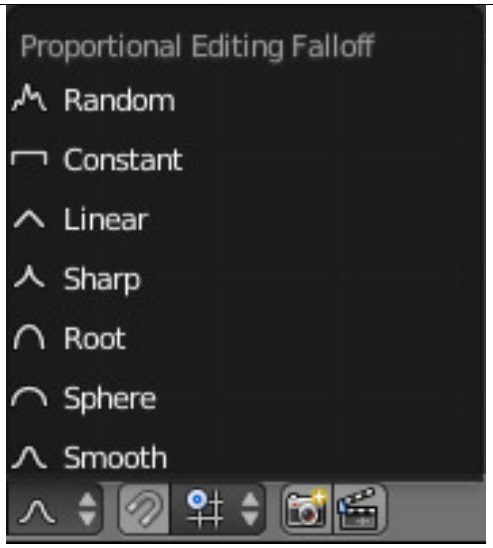


Fig. 2.115: Falloff menu.

The *Proportional Editing* mode menu is on the *3D View* header.

Disable `O`**,** `Alt-O` Proportional Editing is Off, only selected vertices will be affected.

Enable `O`**,** `Alt-O` Vertices other than the selected vertex are affected, within a defined radius.

Projected (2D) Depth along the view is ignored when applying the radius.

Connected `Alt-O` Rather than using a radius only, the proportional falloff spreads via connected geometry. This means that you can proportionally edit the vertices in a finger of a hand without affecting the other fingers. While the other vertices are physically close (in 3D space), they are far away following the topological edge connections of the mesh. The icon will have a gray center when *Connected* is active. This mode is only available in *Edit Mode*.

Falloff While editing, you can change the curve profile used by either using the *Mesh* → *Proportional Falloff* submenu, using the header icon *Falloff menu*, or by pressing `Shift-O` to toggle between the various options.

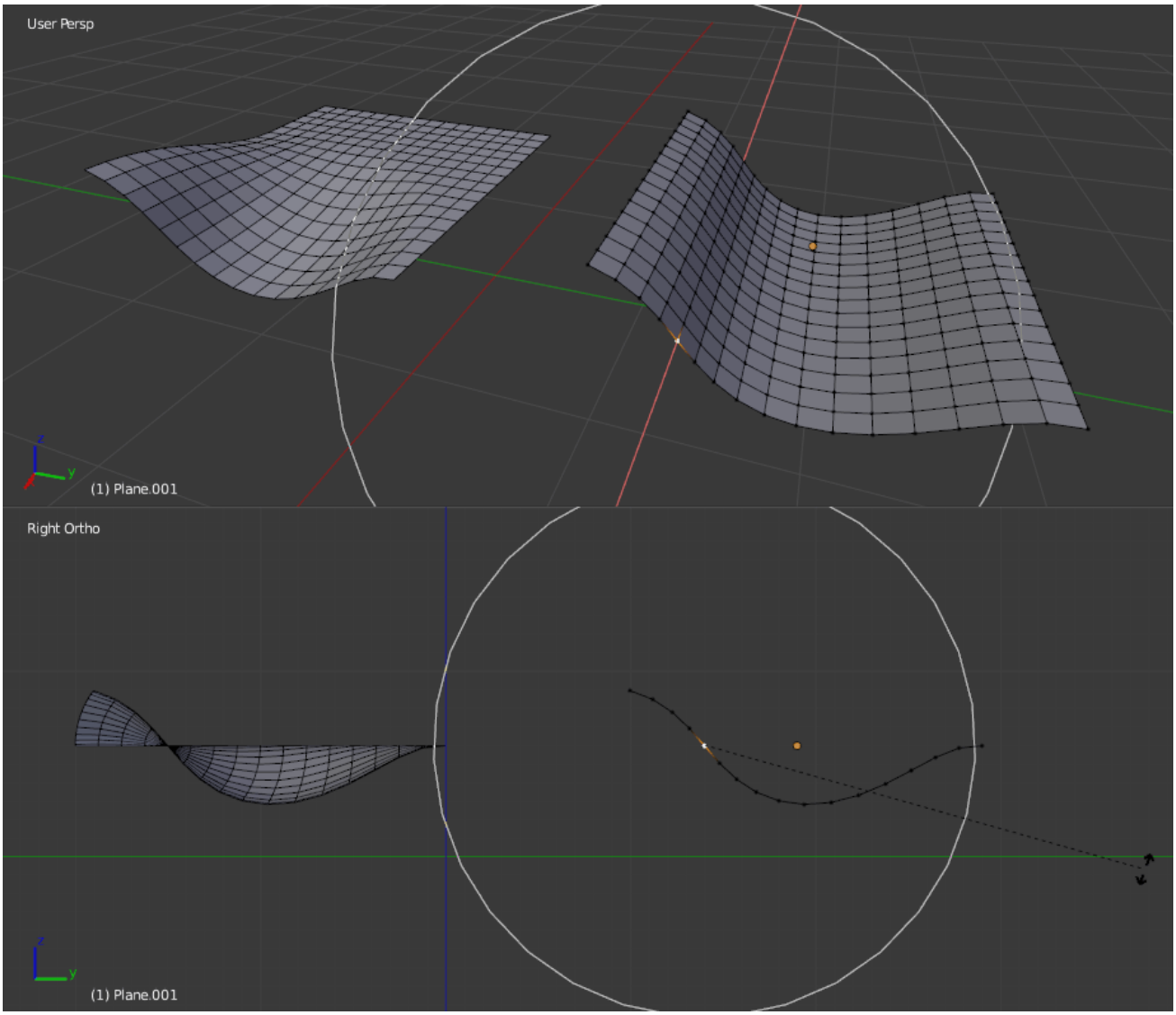


Fig. 2.116: The difference between regular and Projected (2D) proportional option (right).

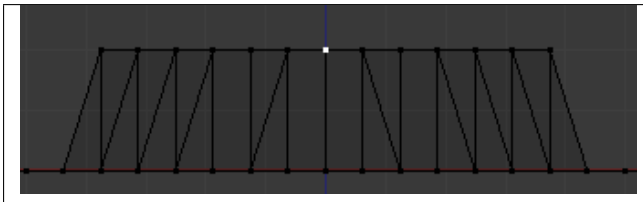


Fig. 2.117: Constant, No Falloff.

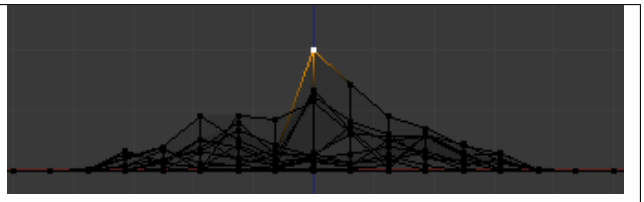


Fig. 2.118: Random Falloff.

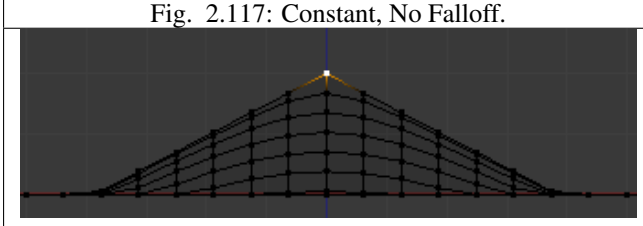


Fig. 2.119: Linear Falloff.

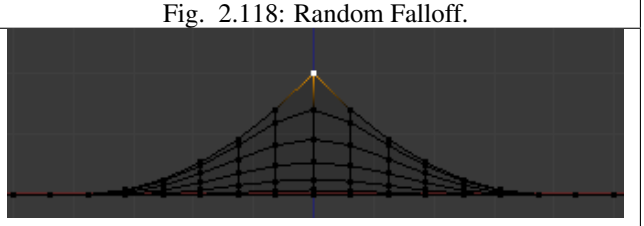


Fig. 2.120: Sharp Falloff.

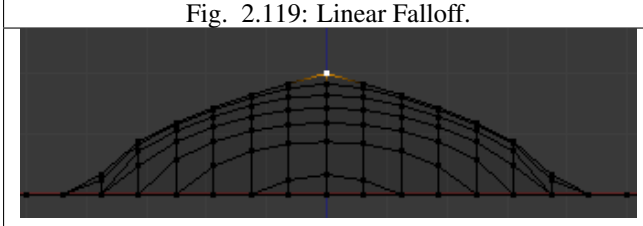


Fig. 2.121: Root Falloff.

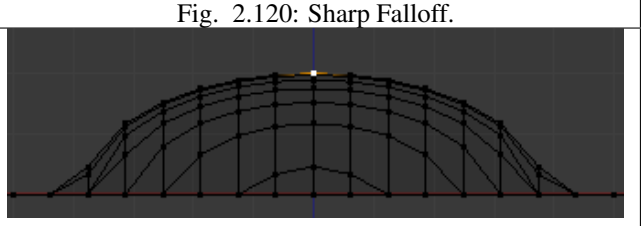


Fig. 2.122: Sphere Falloff.

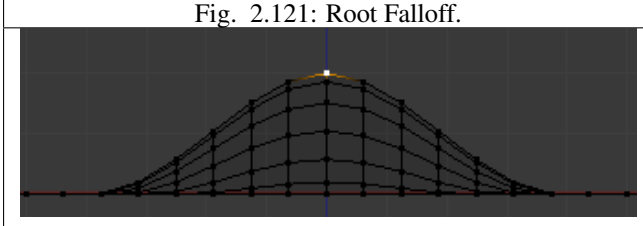


Fig. 2.123: Smooth Falloff.

Examples

Switch to a front view `Numpad1` and activate the grab tool with `G`. As you drag the point upwards, notice how nearby vertices are dragged along with it. When you are satisfied with the placement, click `LMB` to fix the position. If you are not satisfied, cancel the operation and revert your mesh to the way it looked before with `RMB`, `Esc`.

You can use the proportional editing tool to produce great effects with the scaling `S` and rotation `R` tools, as Fig. *A landscape obtained via proportional editing.* shows.

Combine these techniques with vertex painting to create fantastic landscapes. The Fig. *Final rendered landscape.* below shows the results of proportional editing after the application of textures and lighting.

Transform Orientations

Reference

Mode: Object and Edit Modes
Hotkey: `Alt-Spacebar`

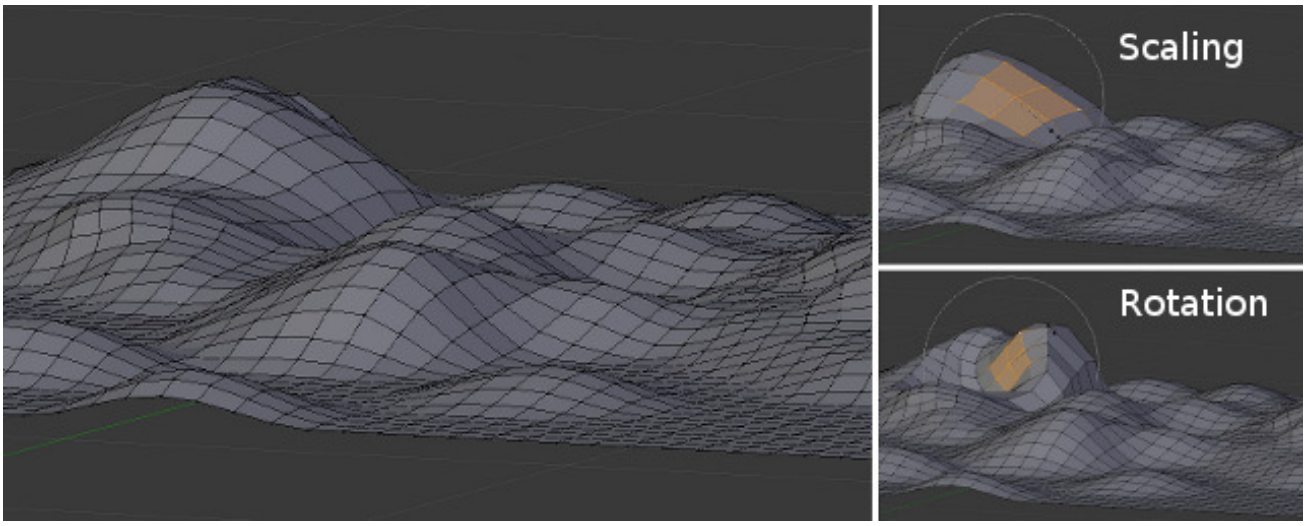


Fig. 2.124: A landscape obtained via proportional editing.

Fig. 2.125: Final rendered landscape.

Orientations affect the behavior of Transformations: Location, Rotation, and Scale. You will see an effect on the 3D Manipulator (the widget in the center of the selection), as well as on transformation constraints (like *axis locking*). This means that, when you press `G-X`, it will constrain to the *global* X-axis, but if you press `G-X-X` it will constrain to your *Transform Orientation* s X-axis.

Fig. 2.126: Transform Orientations Menu.

The Orientations options can be set on the 3D View's header, with `Alt-Spacebar`, or through the *Orientation* menu in a 3D View header.

In addition to the four built-in options, you can define your own custom orientation (see *Custom Orientations* below).

Orientations

Global The manipulator matches the global axis.

When using the Global orientation, the orientation's XYZ matches world's XYZ axis. When this mode is selected, the local coordinates of the object are subjected to the Global coordinates. This is good to place objects in the scene. To constrain an axis, press `G` and the desired axis. To constrain to a local axis, press the desired axis two times. The difference between Global and Local, is more noticeable when you have an object in which the origin is not located at the exact center of the object, and does not match the Global coordinates.

Local The manipulator matches the object axis.

Notice that, here, the Manipulator is at a slight tilt (it is most visible on the object's Y-axis, the green arrow). This is due to our 15° rotation of the object. This demonstrates the difference between local coordinates and global coordinates. If we had rotated the object 90° along its X-axis, we would see that the object's "Up" is the world's "Forward" – or the object's Z-axis would now be the world's Y-axis. This orientation has an effect on many parts of the interface, so it is important to understand the distinction.

Normal The Z-axis of the manipulator will match the normal vector of the selection.

In Object Mode, this is equivalent to Local Orientation, in Edit Mode, it becomes more interesting.

As you see, the light blue lines indicate the faces' normals, and the darker blue lines indicate the vertex normals (these were turned on in the `N` Properties region under *Mesh Display* → *Normals* → *Face* and *Vertex*). Selecting any given face will cause our Manipulator's Z-axis to align with that normal. The same goes for Vertex Select Mode. Edge Select is different – A selected Edge has the Z-axis aligned with it (so you will have to look at the Manipulator widget to determine the direction of X and Y). If you select several elements, it will orient towards the average of those normals.

A great example of how this is useful is in Vertex Select Mode: Pick a vertex and then do `G,Z,Z` to tug it away from the mesh and shove it into the mesh. To make this even more useful, select a nearby vertex and press `Shift-R` to repeat the same movement – except along that second vertex's normal instead.

Gimbal Uses a *Gimbal* behavior that can be changed depending on the current *Rotation Mode*.

View The manipulator will match the 3D View:

- Y: Up/Down
- X: Left/Right
- Z: Towards/Away from the screen.

  This way you can constrain movement to one View axis with `G-X-X`.

Custom Orientations

Reference

Mode: Object and Edit Modes
Hotkey: `Ctrl-Alt-Spacebar`

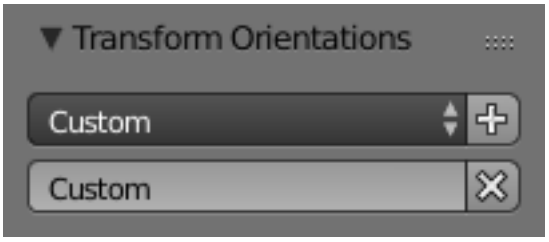


Fig. 2.127: Custom orientation.

You can define custom transform orientations, using object or mesh elements. Custom transform orientations defined from objects use the local orientation of the object whereas those defined from selected mesh elements (vertices, edges, faces) use the normal orientation of the selection.

The *Transform Orientations* panel, found in the Properties region, can be used to manage transform orientations: selecting the active orientation, adding and deleting custom orientations.

The default name for these orientations comes from whatever you have selected. If an edge, it will be titled, "Edge," if an object, it will take that object's name, etc. The Tool shelf (`T` in the 3D View) allows you to rename the custom orientation after you press `Ctrl-Alt-Spacebar`.

The technique of creating custom orientations can become important in creating precise meshes. In Fig. *Custom Extrusion.*, to achieve this effect:

1. Select the object's sloping top edge
2. Create a Custom Orientation with `Ctrl-Alt-Spacebar` and rename it "Top Edge".
3. Select the objects's bottom, right edge.
4. Extrude with `E`.
5. Cancel the extrusion's default movement by pressing `RMB` or `Esc`.
6. Hit `G` to reinitiate movement.
7. Hit `Z-Z` to constrain to the "Top Edge" orientation.

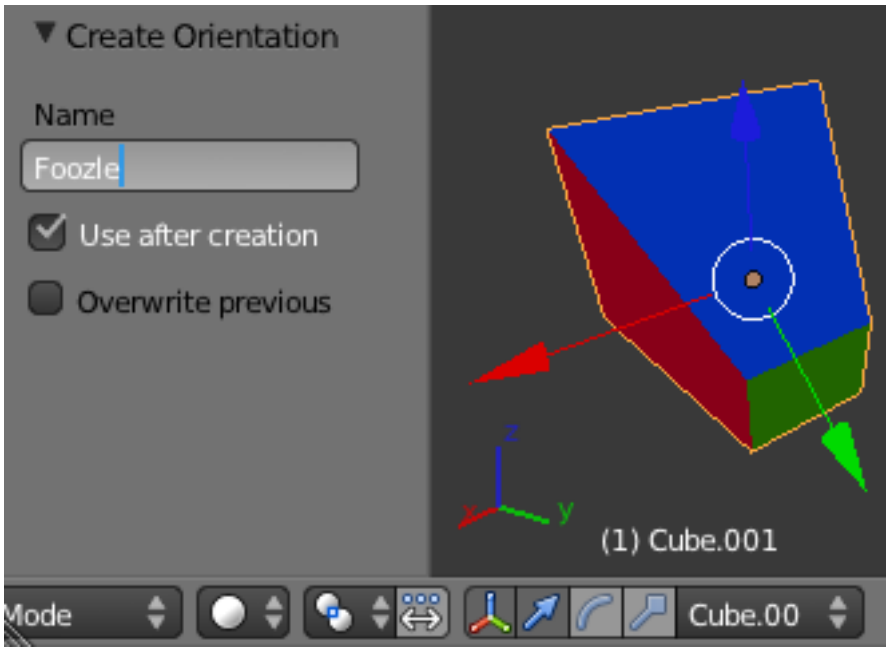


Fig. 2.128: Renaming a Custom Orientation.

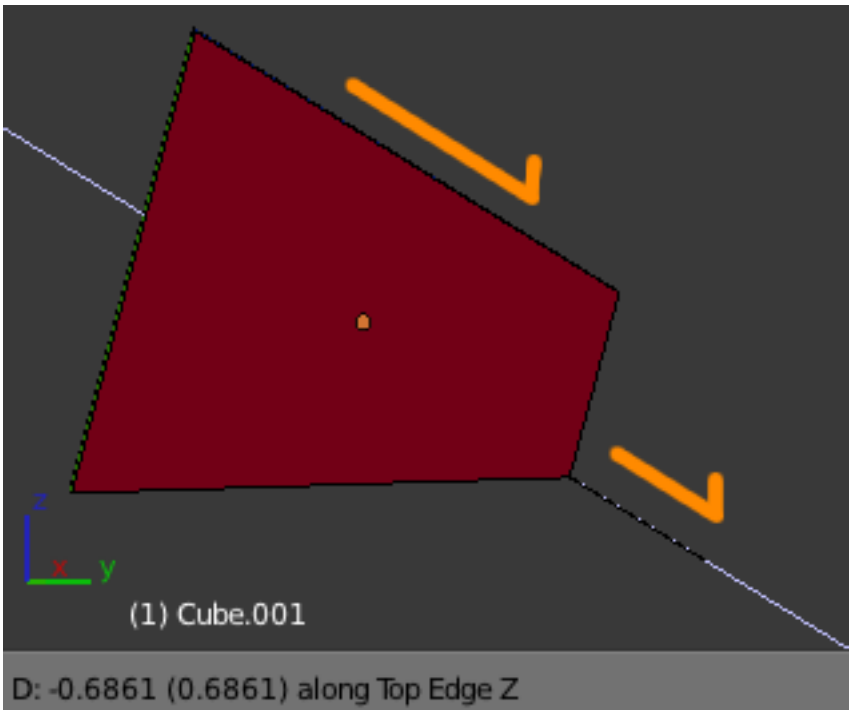


Fig. 2.129: Custom Extrusion.

Examples

Demo Cube

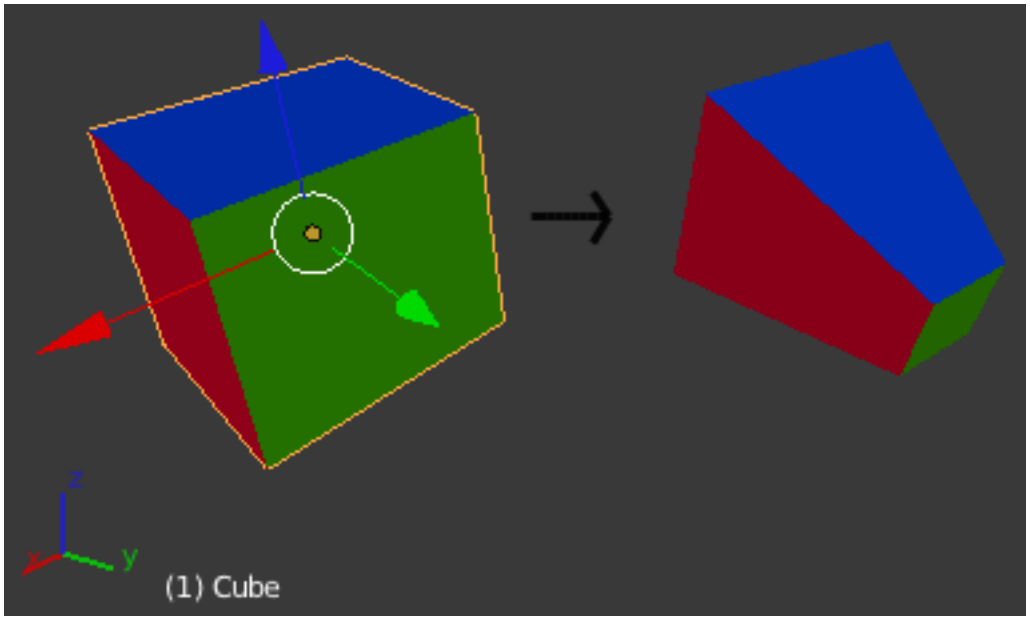


Fig. 2.130: To demonstrate the various behaviors, we add some colors to the default cube, rotate it -15° along its local Z- and X Axes, and we scale its "y" face down.

Please note two things:

- The "Mini-axis" in the lower-left corner, which represents the Global X, Y, Z orientation.
- The *"Object Manipulator"* widget emanating from the selection, which represents the current Transform Orientation.

– If you click on one of the axes of the Manipulator with `LMB`, it will allow you to constrain movement to only this direction. An example of a keyboard equivalent is `G,Z,Z`.

– If you `Shift-LMB` click, it will lock the axis you clicked on and allow you to move in the plane of the two remaining axes. The keyboard analogue is `G,Shift-Z,Shift-Z`.

Effect on Manipulators

The image below shows a cube with the rotation manipulator active in multiple transform orientations. Notice how the manipulator changes depending on the orientation selected (compare A with F).

Similarly, notice how when normal orientation (F and G) is selected the manipulator changes between *Object Mode* and *Edit Mode*. The normal orientation manipulator will also change depending on what is selected in *Edit Mode* i.e. the orientation is based on the normal of the selection which will change depending on how many and which faces, edges or vertices are selected.

Pivot Point

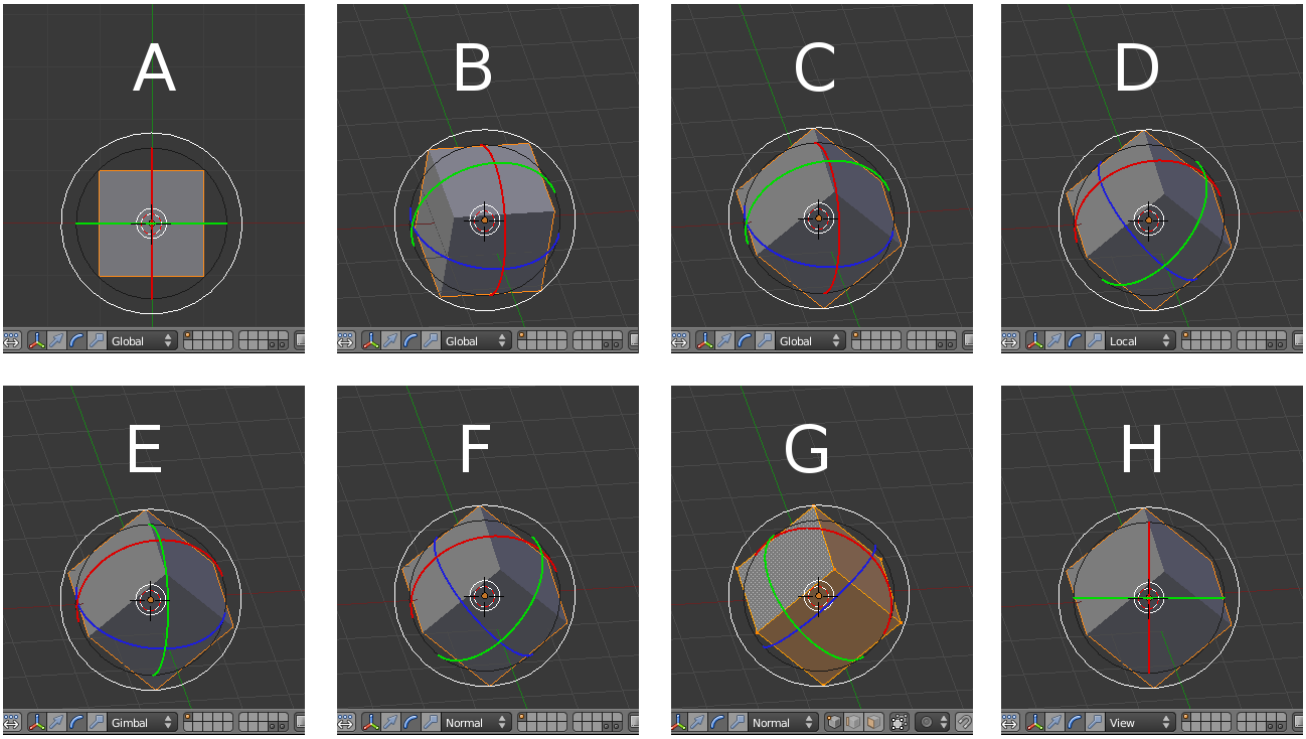


Fig. 2.131: Transform manipulator orientation options.

1. Standard cube in default top view with *global* orientation selected
2. Standard cube with view rotated and *global* orientation selected
3. Randomly rotated cube with view rotated and *global* orientation selected
4. Randomly rotated cube with *local* orientation selected
5. Randomly rotated cube with *gimbal* orientation selected
6. Randomly rotated cube with *normal* orientation selected
7. Randomly rotated cube, vertices selected with *normal* orientation selected
8. Randomly rotated cube with *view* orientation selected

Reference

Mode: Object Mode and Edit Mode
Menu: Droplist in the header of the 3D View

When rotating or scaling an object or group of vertices/edges/faces, you may want to shift the *pivot point* to make it easier to manipulate an object. Using this selector, you can change the pivot point to the location of the:

- Active Element
- Median Point (the average center spot of the selected items)
- Individual Origins
- 3D Cursor
- Bounding Box Center

Each of these can be chosen from the selector in the header of any 3D View, as seen here in Fig. *Pivot Point modes.*. The pages linked below describe each *Pivot Point* mode in more detail.

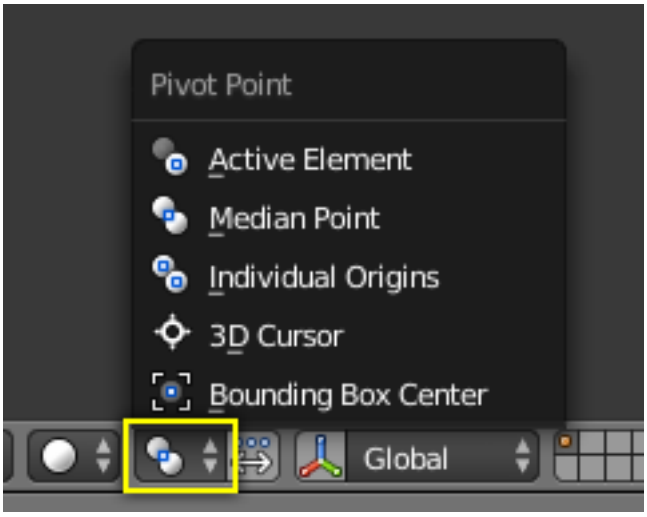


Fig. 2.132: Pivot Point modes.

Active Element

Reference

Mode: Object Mode and Edit Mode
Menu: Select
from the  icon in the 3D View header.
Hotkey: `Alt-.`

The *active* element can be an Object, vertex, edge or a face. The active element is the last one to be selected and will be shown in a lighter orange color when in *Object Mode* and white when in *Edit Mode*. With *Active element as Pivot* set to active, all transformations will occur relative to the active element.

In Object Mode

When in *Object Mode*, rotation and scaling happen around the active Object's center. This is shown by the figure to the below where the active Object (the cube) remains in the same location (note its position relative to the 3D cursor) while the other Objects rotate and scale in relation to the active element.

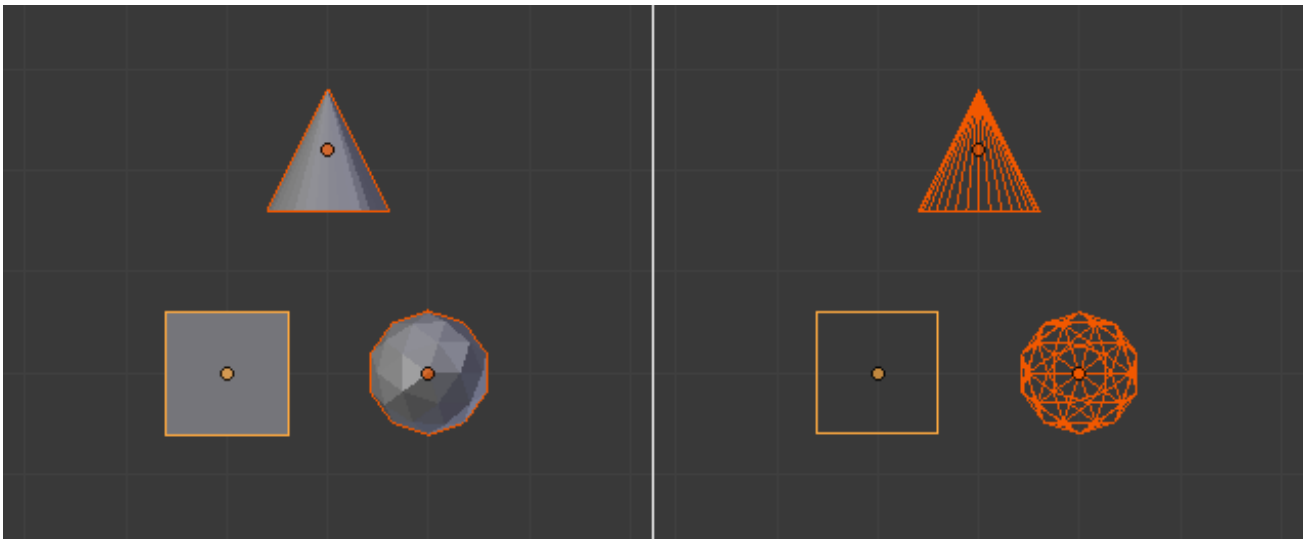


Fig. 2.133: Display of active elements in Object Mode where the active element (cube) is a lighter orange.

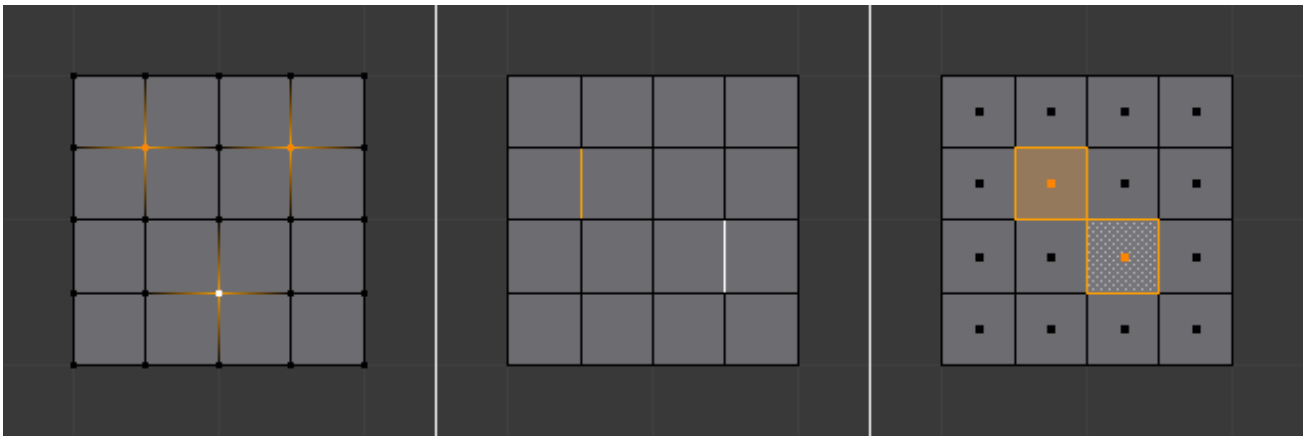


Fig. 2.134: Active elements for vertices, edges and faces in Edit Mode are displayed in white.

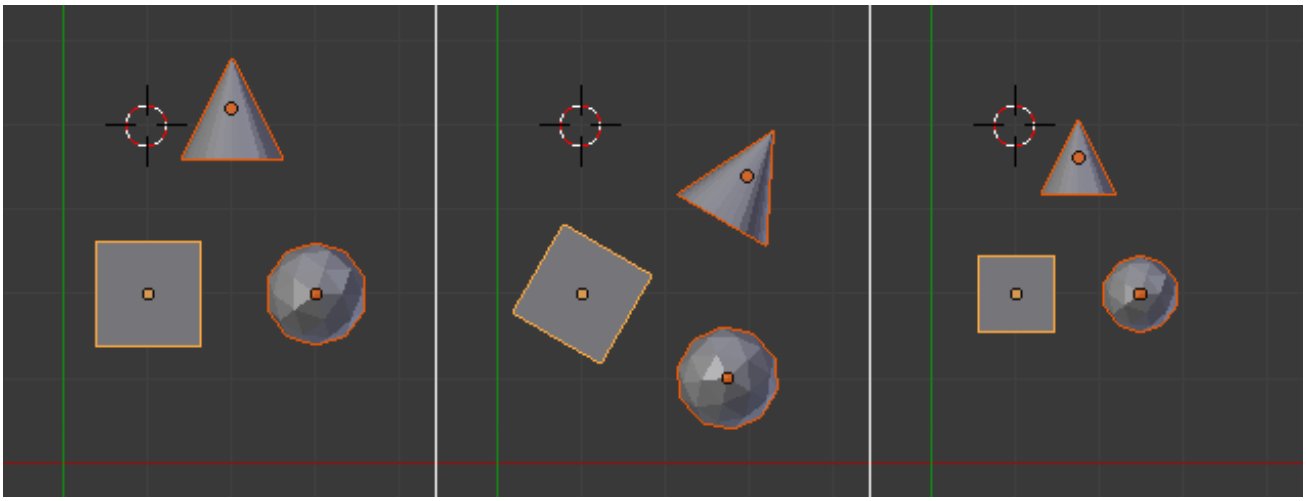


Fig. 2.135: Rotation and scaling with the cube as the active element.

In Edit Mode

Using the active element as a pivot point in *Edit Mode* may seem complex but all the possible transformations follow a few rules:

- The pivot point is always at the median of the active element(s).
- The transformations occur by transformation of the *vertices* of the selected element(s). If an unselected element shares one or more vertices with a selected element then the unselected one will get some degree of transformation also.

Let us examine the following examples: in each case we will see that the two rules apply.

Single selection

When one single element is selected it becomes automatically active. In the image below, you can see that when it is transformed its vertices move, with the consequence that any adjacent element which shares one or more vertices with the active element is also transformed.

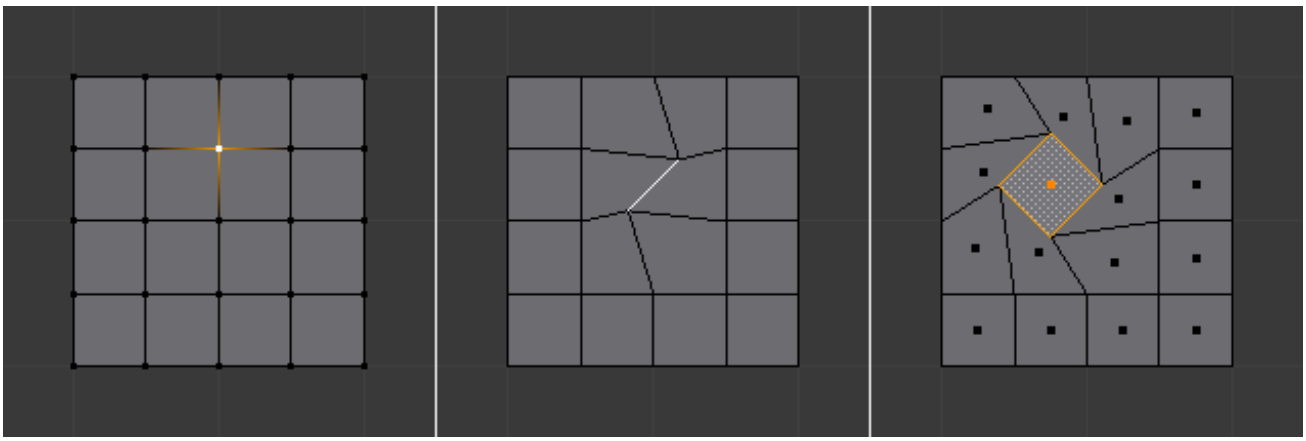


Fig. 2.136: Edit Mode and only one element selected.

Let us review each case:

- *Faces* have their pivot point where their selection dot appears, which is where the median of their vertices is.
- *Edges* have their pivot point on their middle since this is always where the median of an edge is.
- A single *Vertex* has no dimensions at all so it cannot show any transformation (except translation, which is not affected by the pivot point).

Multiple selection

When multiple elements are selected they all transform. The pivot points stay in the same place as what we have seen above, with only one exception for Fgons. In the image below, the selected elements have been rotated.

- For *Faces* the transformation occurs around the selection dot of the active face.
- *Edges* also keep the same behavior with their pivot point at their median.
- There is a case for *Vertices* this time: the active Vertex is where the pivot point resides. All other vertices are transformed relative to it.

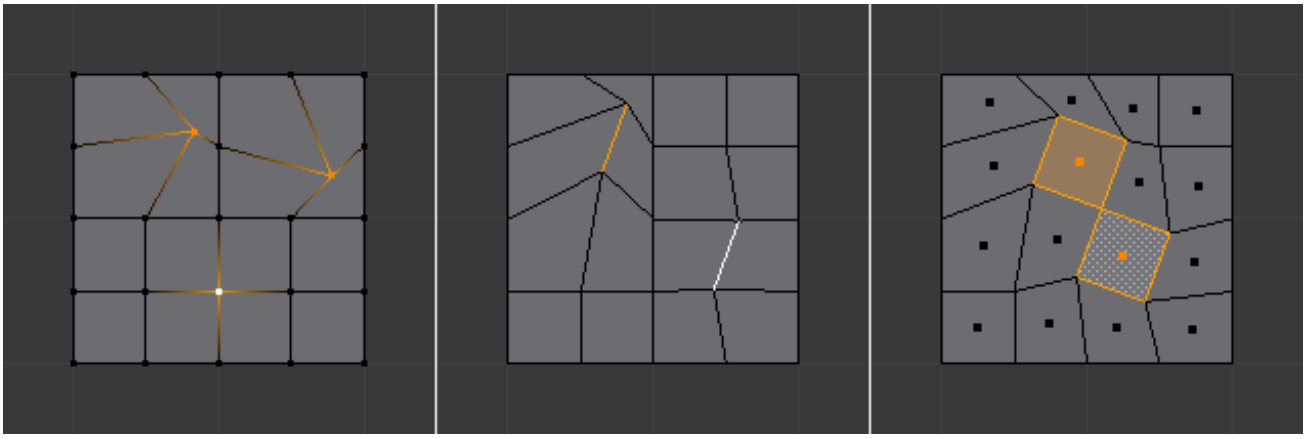


Fig. 2.137: Edit Mode and multiple selections.

Median Point

Reference

Mode: Object Mode and Edit Mode

Menu: Select from the  icon in the 3D View header.
Hotkey: `Ctrl-,`

The Median Point can be considered to be broadly similar to the concept of Center of Gravity (COG). If we assume that every element (Object, face, vertex etc) of the selection has the same mass, the median point would sit at the point of equilibrium for the selection (the COG).

In Object Mode

In Object Mode, Blender only considers the Object centers when determining the median point. This can lead to some counterintuitive results. In the Fig. *Median points in Object Mode.* below, you can see that the median point is between the Object centers and can be nowhere near the Objects' mesh.

In Edit Mode

In Edit Mode, the median point is determined via the part of the selection that has the most elements. For example, in the Fig. *Median points in Edit Mode.*, when there are two cubes with an equal number of vertices, the median point lies directly between the two cubes. However, if we subdivide one cube multiple times so that it has many more vertices, you can see that the median point has shifted to the region with the most vertices.

Individual Origins

Reference

Mode: Object Mode and Edit Mode

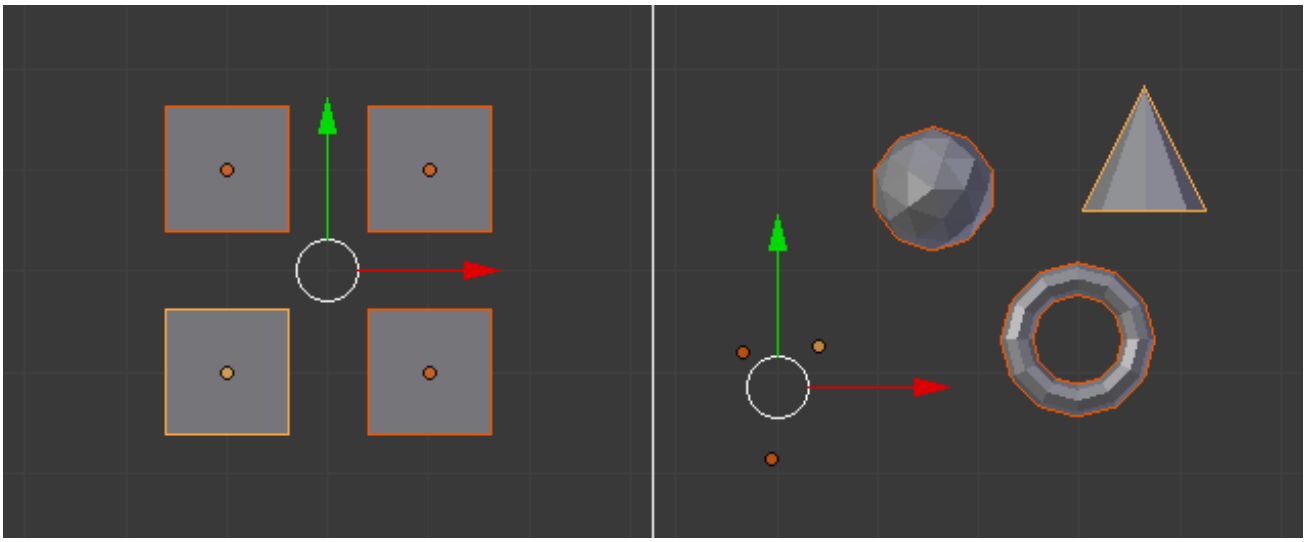


Fig. 2.138: Median points in Object Mode.

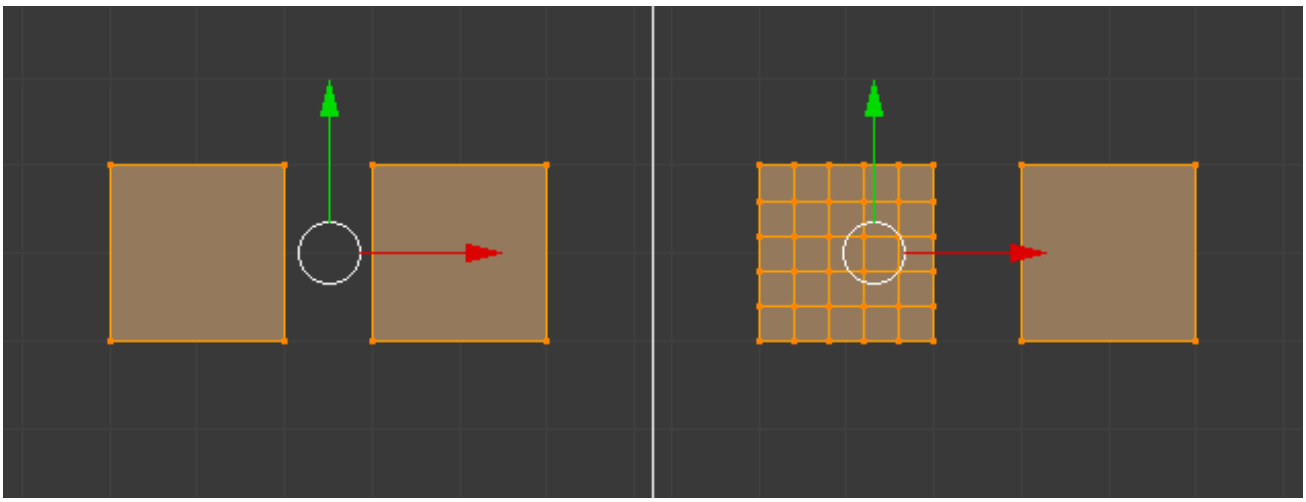


Fig. 2.139: Median points in Edit Mode.

Menu: Select from the  icon in the 3D View header.
Hotkey: `Ctrl-.`

In Object Mode

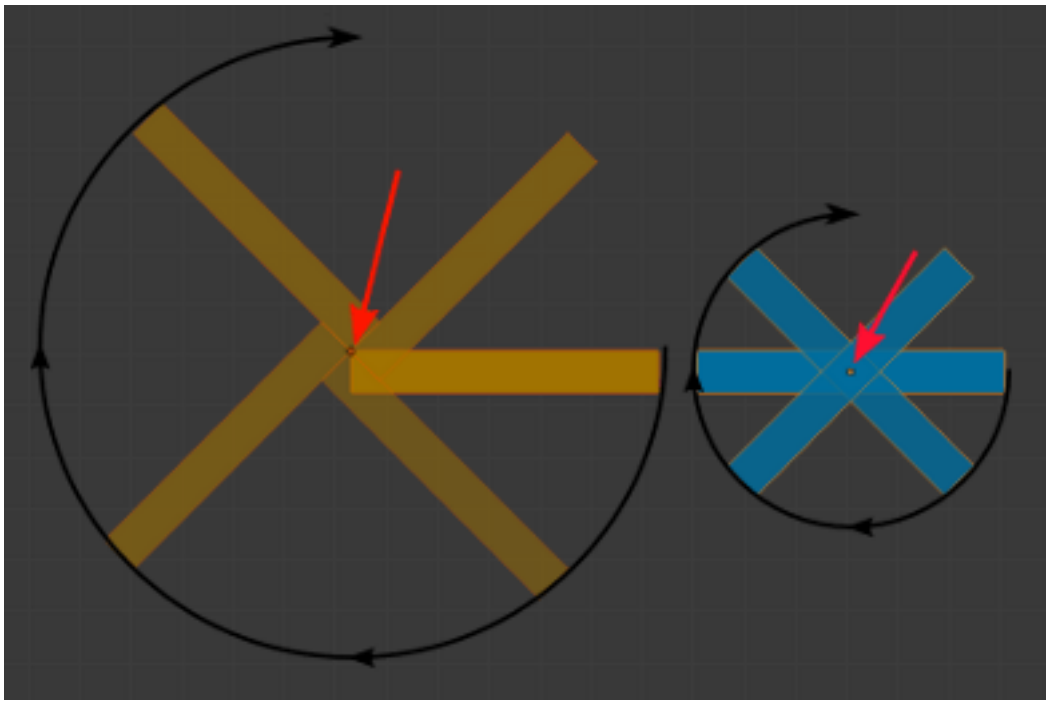


Fig. 2.140: Rotation around individual origins.

The Origin of an object is shown in the 3D View by a small orange circle. This is highlighted in the image to the right by the red arrow. The origin tells Blender the relative position of that object in 3D space. What you see in the 3D View (vertices, edges etc) is what makes up the object.

While the Origin is equivalent to the center of the object, it does not have to be located in the center of the *Mesh*. This means that an object can have its center located on one end of the mesh or even completely outside the mesh. For example, the orange rectangle in the image has its Origin located on the far left of the mesh.

Now let us examine: Rotation around the individual origins:

- The blue rectangle has its Origin located in the center of the mesh, while the orange rectangle has its Origin located on the left hand side.
- When the Pivot Point is set to *Individual Origins*, the center of each object (indicated by the red arrow) remains in place while the object rotates around it in the path shown by the black arrow.

In Edit Mode

In Edit Mode, setting the Pivot Point to Individual Origins produces different results when the selection mode is set to Vertex, Edge or Face. For example, Vertex mode produces results similar to setting the pivot point to median and Edge mode often produces distorted results. Using Individual Origins in Face mode produces the most predictable results.

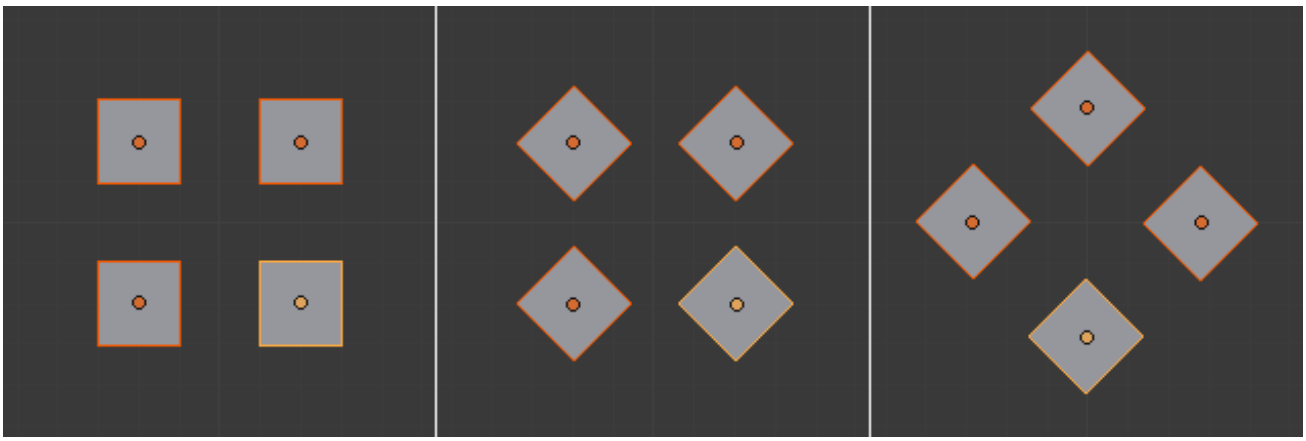


Fig. 2.141: Rotation around individual origins (middle) compared to the median point (right).

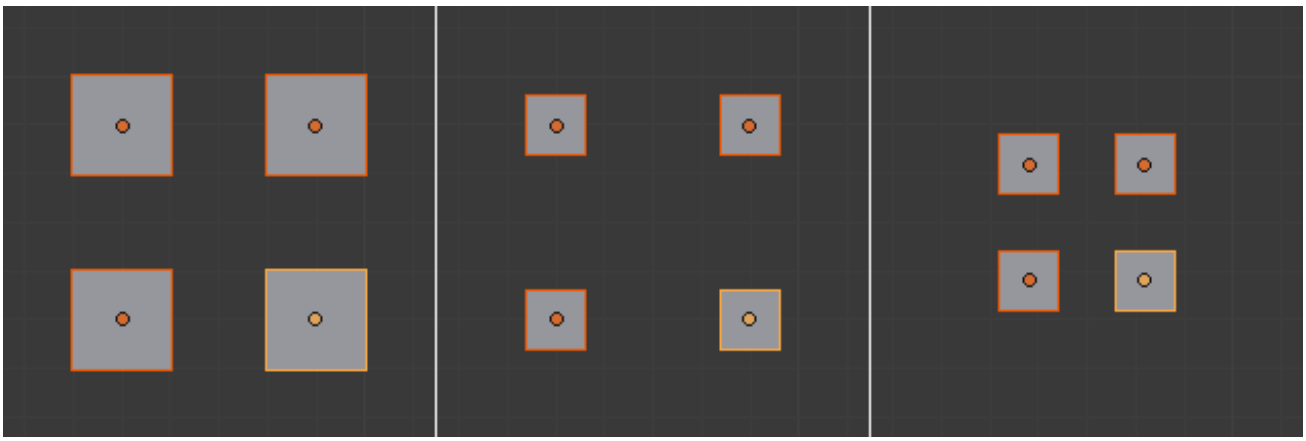


Fig. 2.142: Scaling around individual origins (middle) compared to the median point (right).

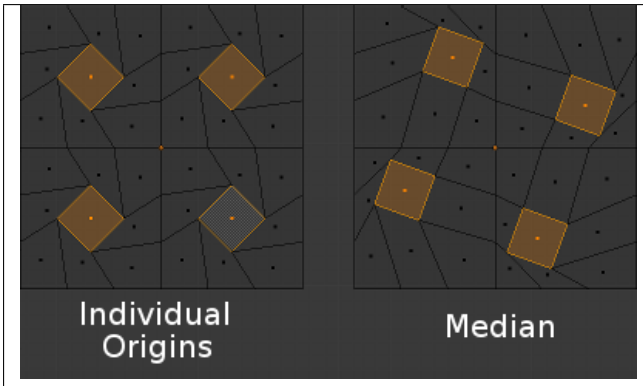


Fig. 2.143: Rotation of individual faces with the pivot point indicated by the image text.

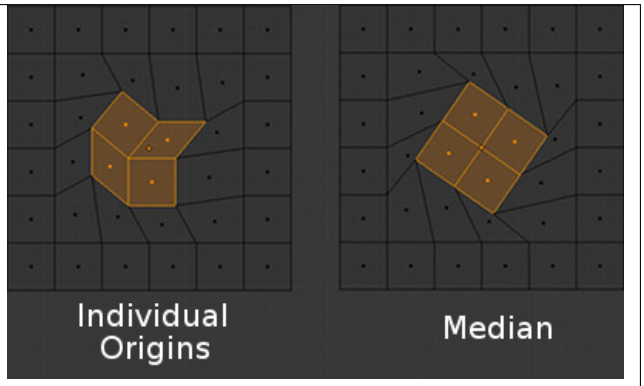


Fig. 2.144: Rotation of grouped faces with the pivot point indicated by the image text.

As can be seen in the images above, faces that touch each other will deform when rotated when the pivot point is set to Individual Origins. Faces that do not touch will rotate around their Individual Origins (their center).

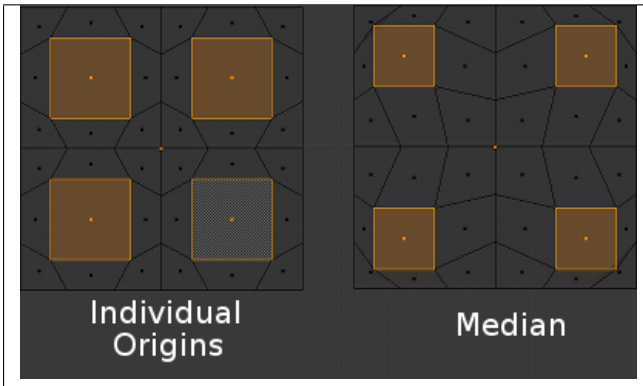


Fig. 2.145: Scaling with non-touching faces.

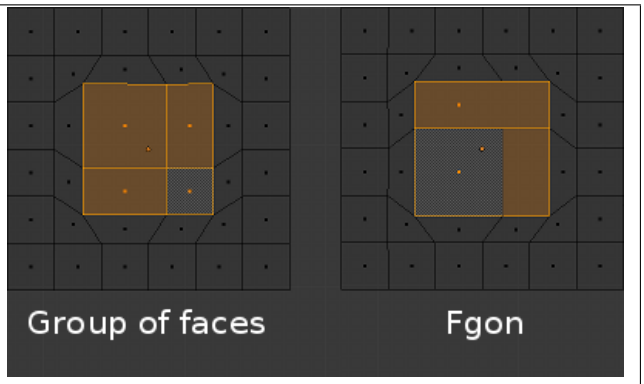


Fig. 2.146: Scaling with touching faces.

Groups of faces and Fgons can be scaled without their outside perimeter being deformed. However, the individual faces inside will not be scaled uniformly.

Once you are aware of its limitations and pitfalls, this tool can save a lot of time and lead to unique shapes. This "anemone" was modeled from a 12 sided cylinder in about 10 minutes by repeatedly using this workflow: extrusions of individual faces, scaling with *median as a pivot point*, and scaling and rotations of those faces with *Individual Origins as pivot points*.

3D Cursor as Pivot

Reference

Mode: Object Mode and Edit Mode

Menu: Select from the  icon in the 3D View header.
Hotkey: .

The 3D cursor is the most intuitive of the pivot points. With the 3D cursor selected as the active pivot point (from either the *Editors Header* or via .), simply position the 3D cursor and then do the required transformation. All rotation and scaling transformations will now be done relative to the location of the 3D cursor.

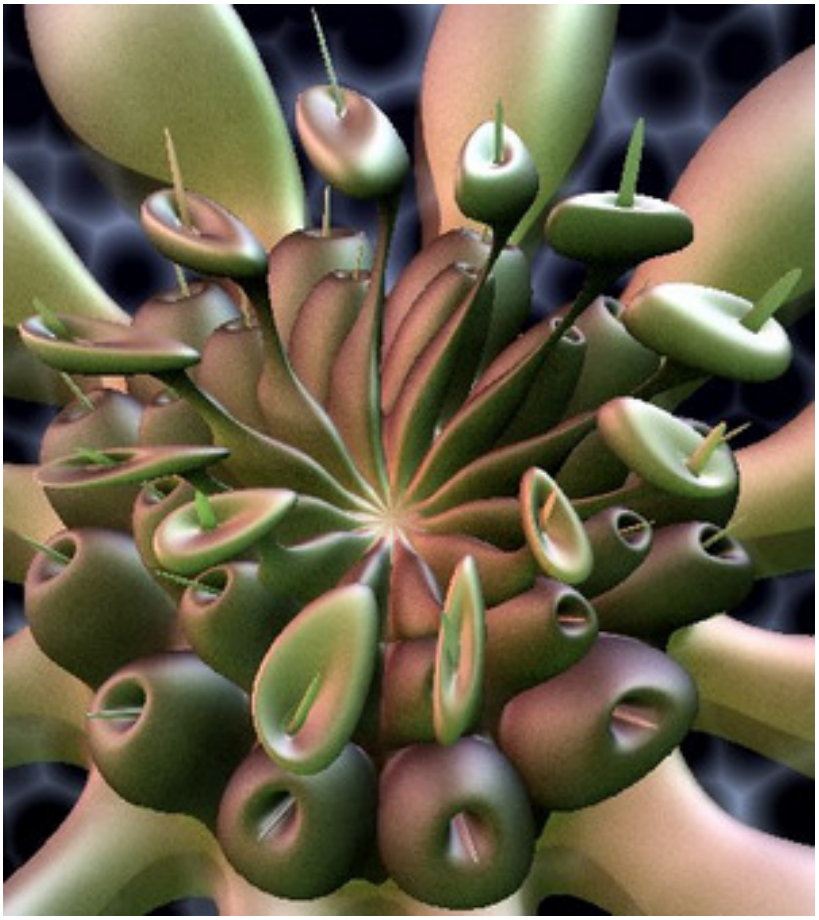


Fig. 2.147: Modeling with faces and individual origins as the pivot point.

Example

The image below shows the difference when rotating an Object around the median point (left) and around the 3D cursor (right).

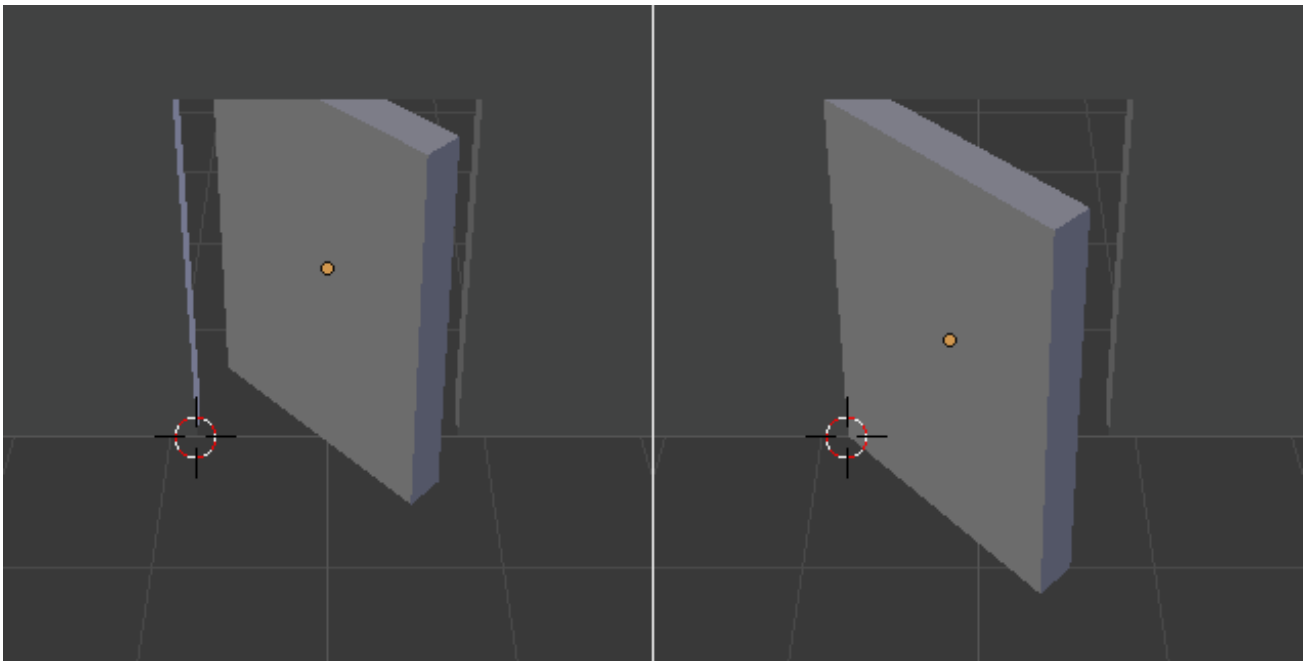


Fig. 2.148: Rotation around the 3D cursor compared to the median point.

Bounding Box Center

Reference

Mode: Object Mode and Edit Mode

Menu: Select from the  icon in the 3D View header.
Hotkey: `,`

The bounding box is a rectangular box that is wrapped as tightly as possible around the selection. It is oriented parallel to the world axes. In this mode the pivot point lies at the center of the bounding box. You can set the pivot point to bounding box with `,` or via the menu in the editors header. The image below shows how the Object's Bounding Box size is determined by the size of the Object.

In Object Mode

In *Object Mode*, the bounding box is wrapped around the Object and transformation takes place relative to the location of the Object center (indicated by the yellow circle). The image below shows the results of using the Bounding Box as the pivot point in a number of situations.

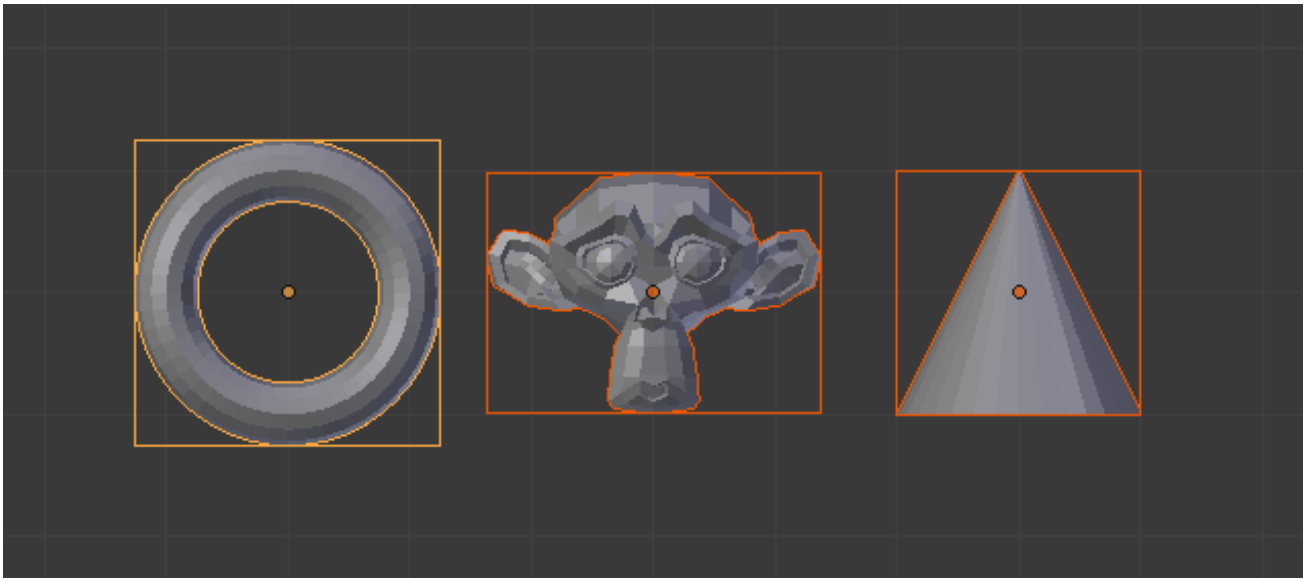


Fig. 2.149: Relationship between an Object and its Bounding Box.

For example, images A (before rotation) and B show rotation when the Object center is in its default position, while images C (before rotation) and D shows the result when the Object center has been moved. Image E shows that when multiple Objects are selected, the pivot point is calculated based on the Bounding Box of all the selected Objects.

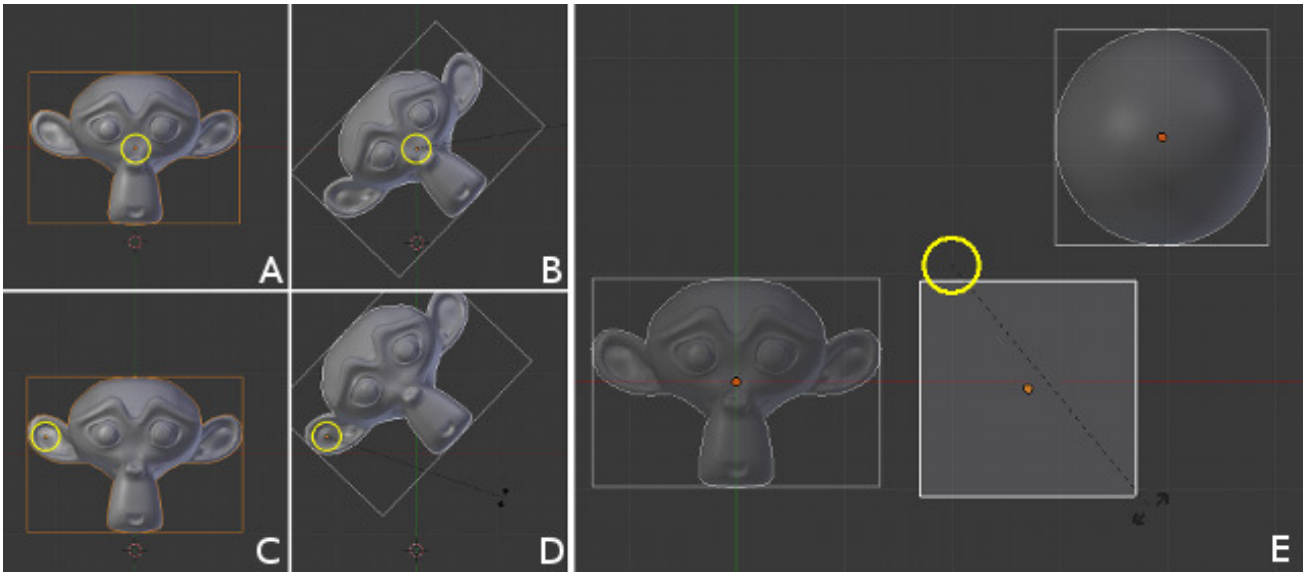


Fig. 2.150: The grid of four images on the left (ABCD) shows the results of Object rotation when the pivot point is set to Bounding Box. The image to the right (E) shows the location of the Bounding Box pivot point when multiple Objects are selected. The pivot point is shown by a yellow circle.

In Edit Mode

This time it is the Object Data that is enclosed in the bounding box. The bounding box in *Edit Mode* takes no account of the Object(s) centers, only the center of the selected vertices.

Manipulate Center Points

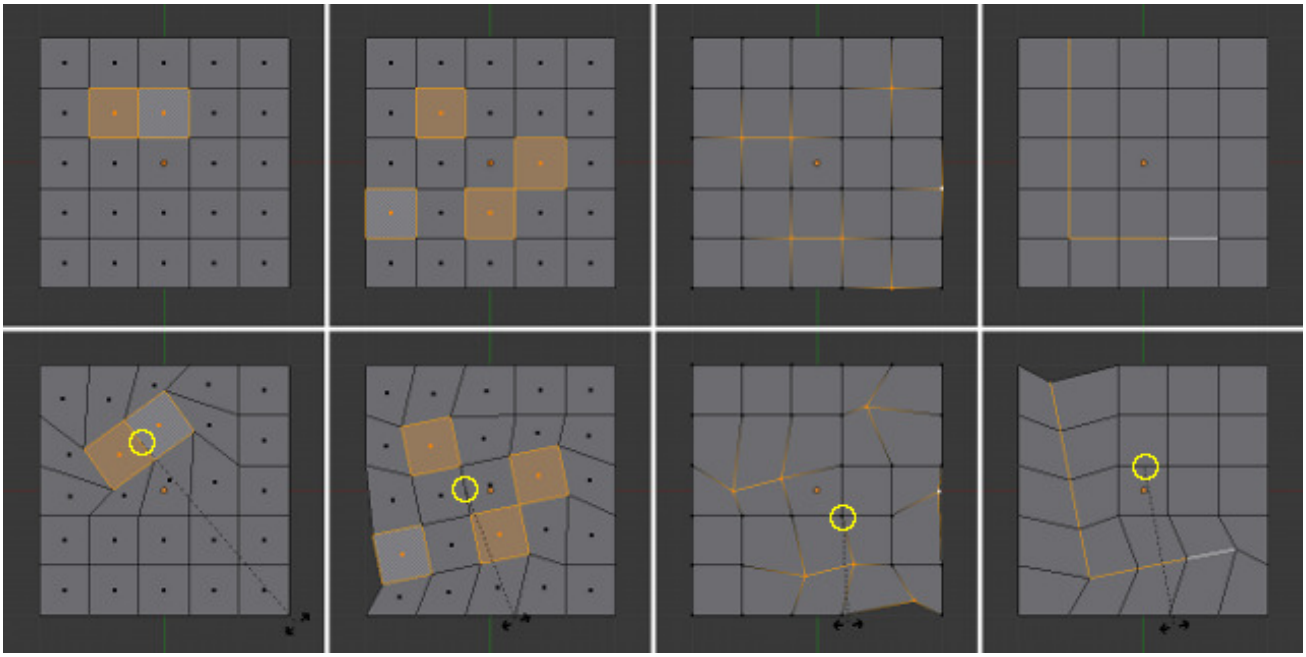


Fig. 2.151: The effects of rotation in different mesh selection modes when the bounding box is set as the pivot point. The pivot point is shown by a yellow circle.

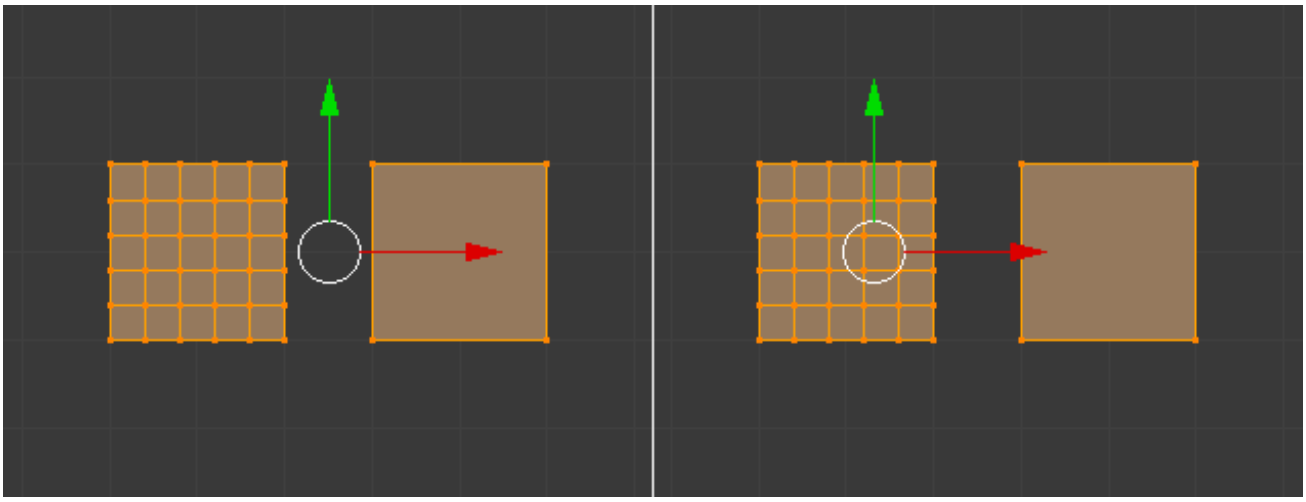


Fig. 2.152: The bounding box center compared to the median point.

Reference

Mode: Object Mode and Pose Mode

Menu: Select from the  icon in the 3D View header.
Hotkey: `Alt-,`

When this option is enabled, the transformation will change the positions of the object's origins, but will not affect the object itself.

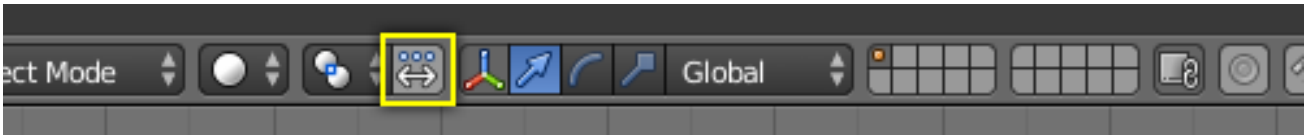


Fig. 2.153: Manipulate Center Points button.

In the examples below, a comparison of the scaling and rotation of objects, when *Manipulate Center Points* is enabled (middle) and disabled (right).

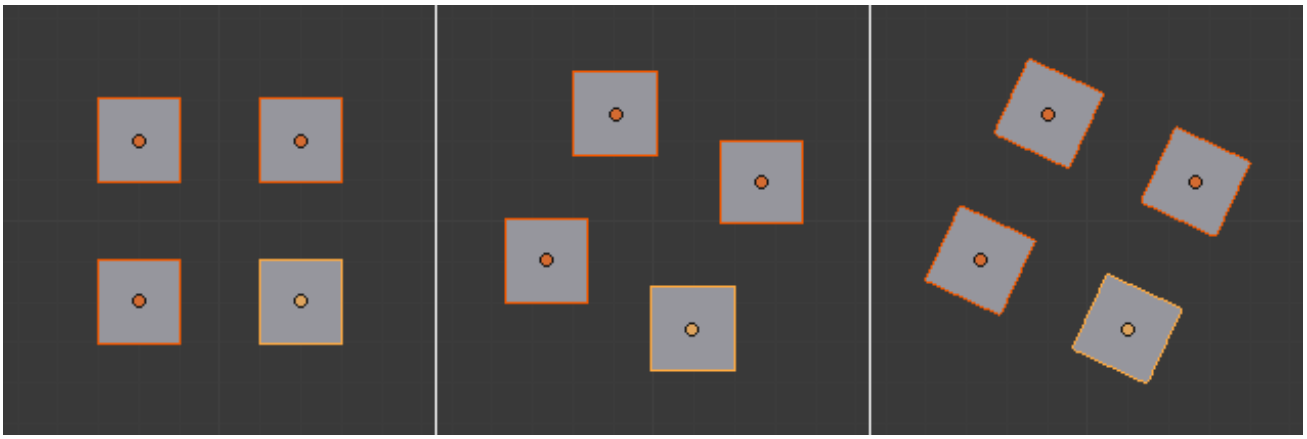


Fig. 2.154: Rotation example.

Objects Properties

Introduction

TODO.

Transform Properties

Each object stores its position, orientation, and scale values. These may need to be manipulated numerically, reset, or applied.

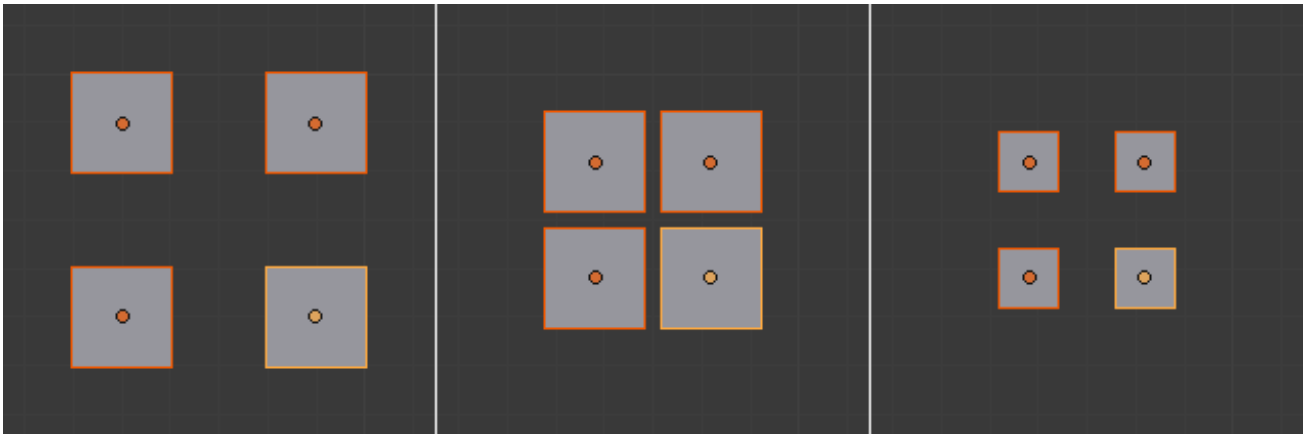


Fig. 2.155: Scaling example.

Transform Panel

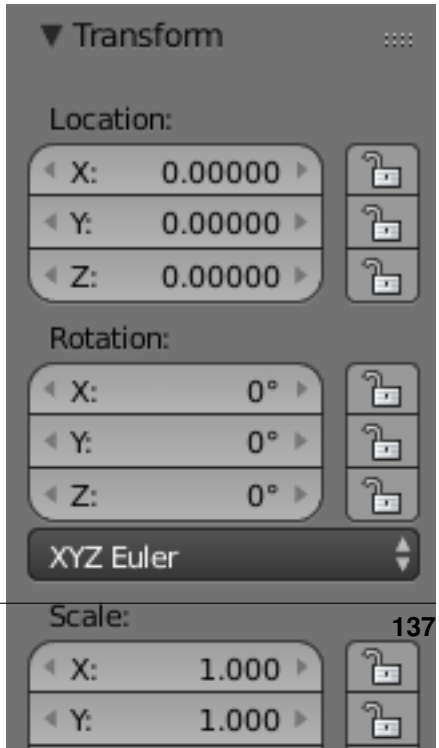
Reference

Mode: Edit and Object Modes
Menu: *Object* → *Transform*
Panel: *Properties region* → *Transform*

The *Transform* panel in the Properties region allows you to view and manually/numerically control the position, rotation, and other properties of an object, in *Object Mode*. In *Edit Mode*. It mainly allows you to enter precise coordinates for a vertex, or median position for a group of vertices (including an edge/face). As each type of object has a different set of options in its *Transform* panel in *Edit Mode*, see their respective descriptions in the *Modeling chapter*.

Options in Object Mode

Use this panel to either edit or display the object's transform properties such as position, rotation and/or scaling. These fields change the object's center and then affect the aspect of all of its *vertices* and faces.



Location The object's center location in global coordinates.

Rotation The object's orientation, relative to the global axes and its own center.

Rotation Mode Method for calculating rotations, additional information can be found here.

Euler The manipulator handles are aligned to the *Euler* axis, allowing you to see the discreet XYZ axis underlying the euler rotation, as well as possible *gimbal lock*.

Axis Angle The X, Y, and Z coordinates define a point relative to the object origin through which an imaginary "skewer" passes. The "W" value is the rotation of this skewer, in radians. Here, the Manipulator's Z-axis stays aligned with this skewer.

Quaternion X, Y, Z and W correspond to the *Quaternion* components.

Scale The object's scale, relative to its center, in local coordinates (i.e. the *Scale X* value represents the scale along the local X-axis). Each object (cube, sphere, etc.), when created, has a scale of one Blender unit in each local direction. To make the object bigger or smaller, you scale it in the desired dimension.

Dimensions The object's basic dimensions (in Blender units) from one outside edge to another, as if measured with a ruler. For multi-faceted surfaces, these fields give the dimensions of the bounding box (aligned with the local axes – think of a cardboard box just big enough to hold the object).

Transform Properties Locking

The locking feature of the Location, Rotation and Scale fields allows you to control a transform property solely from the properties region. Once a lock has been activated any other methods used for transformation are blocked. For example, if you locked the *Location X* field then you cannot use the mouse to translate the object along the global X axis. However, you can still translate it using the *Location X* edit field. Consider the locking feature as a rigid constraint only changeable from the panel.

To lock a field, click the padlock icon next to the field. The field is unlocked if the icon appears as "open padlock", and it is locked if the icon appears as "closed padlock".

Delta Transforms

Delta Transforms are simply transformations that are applied on top of the transforms described above. They can be found in the *Properties Editor* → *Object* → *Delta Transforms*.

Usage

Delta Transforms are particularly useful in animations. For example, you can animate an object with the "normal" transforms then move them around with Delta Transforms.

Object Relations

Layers

Reference

Mode: Object Mode
Panel: *Object* → *Relations*
Menu: *Object* → *Move to Layer...*
Hotkey: `M`

3D scenes often become exponentially more confusing as they grow more complex. Sometimes the artist also needs precise control over how individual objects are lit, and does not want lights for one object to affect nearby objects. For this and other reasons below, objects can be placed into one or more "layers". Using object layers, you can:

- Selectively display objects from certain layers in your 3D View, by selecting those layers in the *3D View* header. This allows you to speed up interface redrawing, reduce virtual-world clutter, and help improve your workflow.
- Control *which lights illuminate an object*, by making a light illuminate only the objects on its own layer(s).
- Control which forces affect which *particle systems*, since particles are only affected by forces and effects on the same layer.
- Control which layers are rendered (and hence, which objects), and which properties/channels are made available for compositing by using *render layers*.

Armatures can also become very complex, with different types of bones, controllers, solvers, custom shapes, and so on. Since armatures are usually located close together, this can quickly become cluttered. Therefore, Blender also provides layers just for armatures. Armature layers are very similar to object layers, in that you can divide up an armature (rig) across layers and only display those layers you wish to work on.

See also:

Armature Layers

Working with Layers

3D layers differ from the layers you may know from 2D graphics applications as they have no influence on the drawing order and are there (except for the special functions listed above) mainly to allow you to organize your scene.

When rendering, Blender only renders the selected layers. If all your lights are on a layer that is *not selected*, you will not see anything in your render except for objects lit by ambient lighting.

Groups and *Parents* are other ways to logically group related sets of objects.

Viewing layers

Blender provides twenty layers whose visibility can be toggled with the small unlabeled buttons in the header (see *3D View layer buttons*). To select a single layer, click the appropriate button with `LMB`; to select more than one, use `Shift-LMB` - doing this on an already active layer will deselect it.

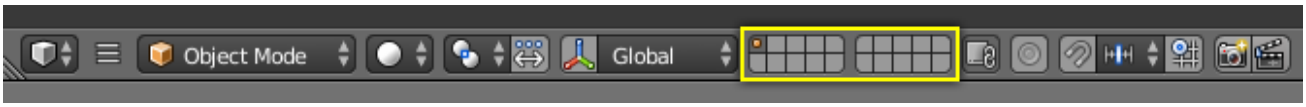


Fig. 2.157: 3D View layer buttons.

To select layers via the keyboard, press `1` to `0` (on the main area of the keyboard) for layers 1 through 10 (the top row of buttons), and `Alt-1` to `Alt-0` for layers 11 through 20 (the bottom row). Use `Shift` for multiple (de)selection works for these shortcuts too.

You can select or deselect all Scene Layer buttons at once by pressing `\`.

Locking to the scene

By default, the lock button directly to the right of the layer buttons is enabled. This means that changes to the viewed layers affect all other 3D Views locked to the scene. See the *navigating the 3D View options page* for more information.

Multiple Layers

An object can exist on multiple layers. For example, a lamp that only lights objects on a shared layer could "be" on layers 1, 2, and 3. An object on layers 3 and 4 would be lit, whereas an object on layers 4 and 5 would not. There are many places where layer-specific effects come into play, especially lights and particles.

Moving objects between layers

Fig. 2.158: Layer selection.

To move selected objects to a different layer, press `M` and then select the layer you want from the pop-up menu. Objects can also be on more than one layer at a time. To have an object on multiple layers, hold `Shift` while clicking.

Fig. 2.159: Selection in the Object tab.

Another way to view or change a selected object layer is via the *Relations* panel, in the *Object* tab.

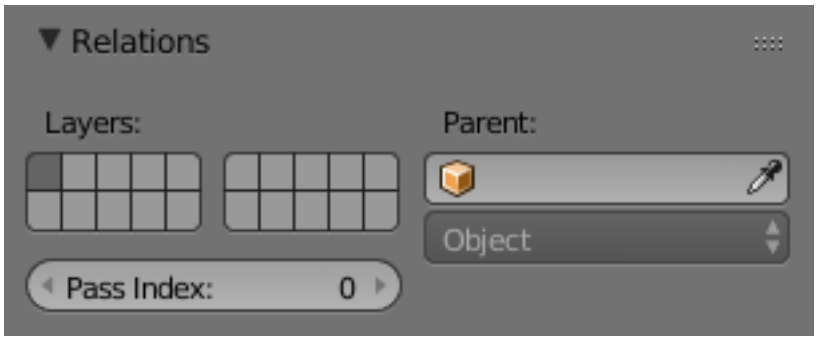


Fig. 2.160: Layers in Object tab, Relations panel.

You will then see the layer buttons in the *Relations* panel – as before – the object can be displayed on more than one layer by clicking `Shift-LMB`.

Parenting Objects

When modeling a complex object, such as a watch, you may choose to model the different parts as separate objects. However, all of the parts may be attached to each other. In these cases, you want to designate one object as the parent of all the children. Movement, rotation or scaling of the parent also affects the children.

To parent objects, select at least two objects (select the *Child Objects* first, and select the *Parent Object* last), and press `Ctrl-P`. The *Set Parent To* menu will pop up allowing you to select from one of several possible different parenting types. Selecting one of the entries in *Set Parent To* confirms, and the child/children to parent relationship is created.

The last object selected will be the *Active Object* (outlined in light orange), and will also be the *Parent Object*. If you selected multiple objects before selecting the parent, they will all be children of the parent and will be at the same level of the hierarchy (they are "siblings").

The *Set Parent To* pop-up menu is context sensitive, which means the number of entries it displays can change depending on what objects are selected when the `Ctrl-P` shortcut is used.

For non-inverse-mode, press `Shift-Ctrl-P` instead. This creates an alternative parent-child-relationship where child-objects exist entirely in the parent's coordinate system. This is the better choice for CAD purposes, for example.

Moving, rotating or scaling the parent will also usually move/rotate/scale the child/children. However, moving/rotating/scaling the child/children of the parent will not result in the parent moving/rotating/scaling. In other words, the direction of influence is from parent to child and not child to parent.

In general when using `Ctrl-P` or *3D View Header* → *Object* → *Parent* to parent objects, the *Child Objects* can only have one *Parent Object*. If a *Child Object* already has a *Parent Object* and you give it another parent then Blender will automatically remove the previous parent relationship.

Blender supports many different types of parenting, listed below:

- Object
- Armature Deform
- Bone
- Curve Deform
- Path Constraint
- Lattice Deform
- Vertex
- Vertex (Triangle)

Object Parent

Object Parent is the most general form of parenting that Blender supports. If will take selected objects and make the last selected object the *Parent Object*, while all other selected objects will be *Child Objects*.

Object (Keep Transform) Parent

Object (Keep Transform) Parent works in a very similar way to *Object Parent* the major difference is in whether the *Child Objects* will remember any previous transformations applied to them from the previous *Parent Object*.

Since explaining this in an easy to understand technical way is hard, lets instead use an example to demonstrate.

Assume that we have a scene consisting of three objects, those being two Empty Objects named "EmptyA" and "EmptyB", and a Monkey object. Fig. *Scene with no parenting.* shows the three objects with no parenting relationships active on them.

Fig. 2.161: Scene with no parenting.

If you select the Monkey object by `RMB` click and then `Shift-RMB` click "EmptyA" object and press `Ctrl-P` and then select *Object* from the *Set Parent To* pop-up menu. This will result in "EmptyA" object being the *Parent Object* of the

Monkey object. With only "EmptyA" selected rotating/scaling/moving it will result in the Monkey object being altered respectively.

Scale the "EmptyA" object, so that the Monkey becomes smaller and moves to the left a little.

Fig. 2.162: The monkey is the child object of "EmptyA".

If you select only the Monkey object by `RMB` click and then `Shift-RMB` click "EmptyB" object and press `Ctrl-P` and select *Object* from the *Set Parent To* pop-up menu. This will result in "EmptyB" object being the *Parent Object* of the Monkey object. Notice that when you change the parent of the Monkey the scale of the Monkey changed.

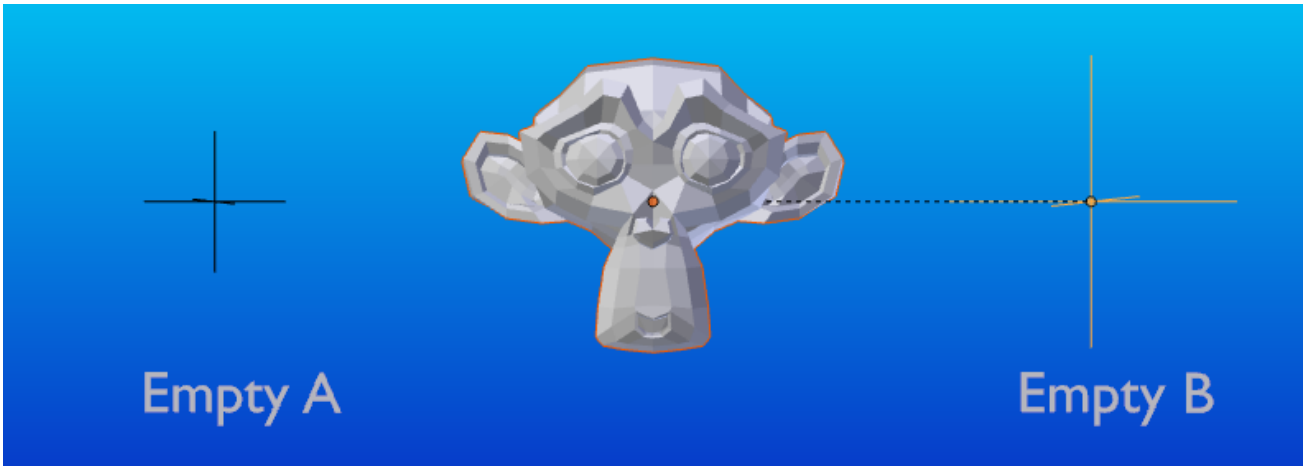


Fig. 2.163: The monkey is the child object of "EmptyB".

This happens because the Monkey object never had its scale altered directly, the change came about because it was the child of "EmptyA" which had its scale altered. Changing the Monkey's parent to "EmptyB" resulted in those indirect changes in scale being removed, because "EmptyB" has not had its scale altered.

This is often the required behavior, but it is also sometimes useful that if you change your *Parent Object* that the *Child Object* keep any previous transformations it got from the old *Parent Object*; If instead when changing the *Parent Object* of the Monkey from "EmptyA" to "EmptyB" we had chosen parenting type *Object (Keep Transform)*, the Monkey would keep its scale information it obtained from the old parent "EmptyA" when it is assigned to the new parent "EmptyB";

If you want to follow along with the above description here is the blend-file used to describe *Object (Keep Transform)* parenting method:

File:Parent_-_Object_(Keep_Transform)_(Demo_File).blend.

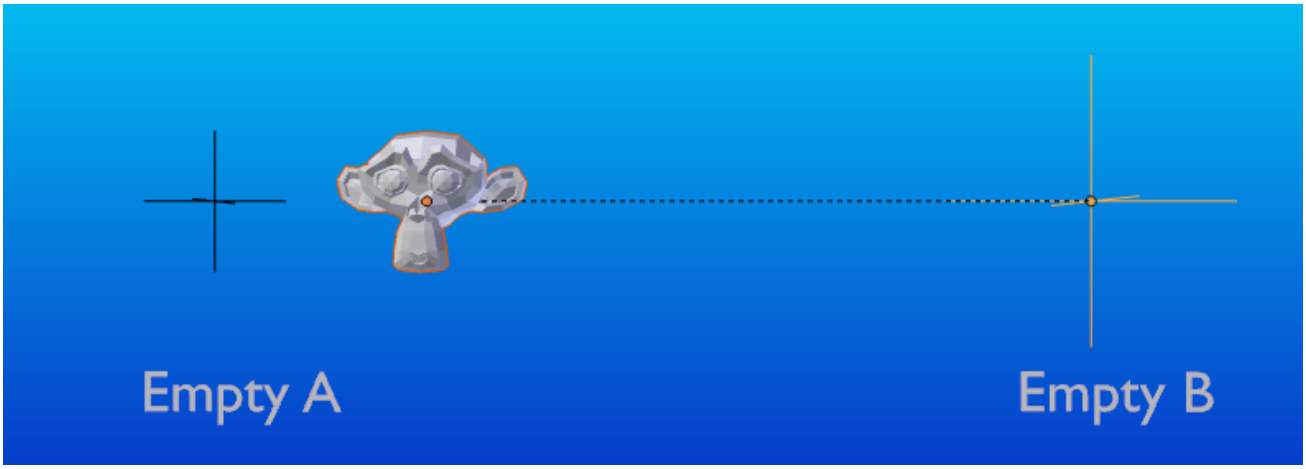


Fig. 2.164: The Object (Keep Transform) parent method.

Bone Parent

Bone parenting allows you to make a certain bone in an armature the Parent Object of another object. This means that when transforming an armature the Child Object will only move if the specific bone it is the Child Object of moves.

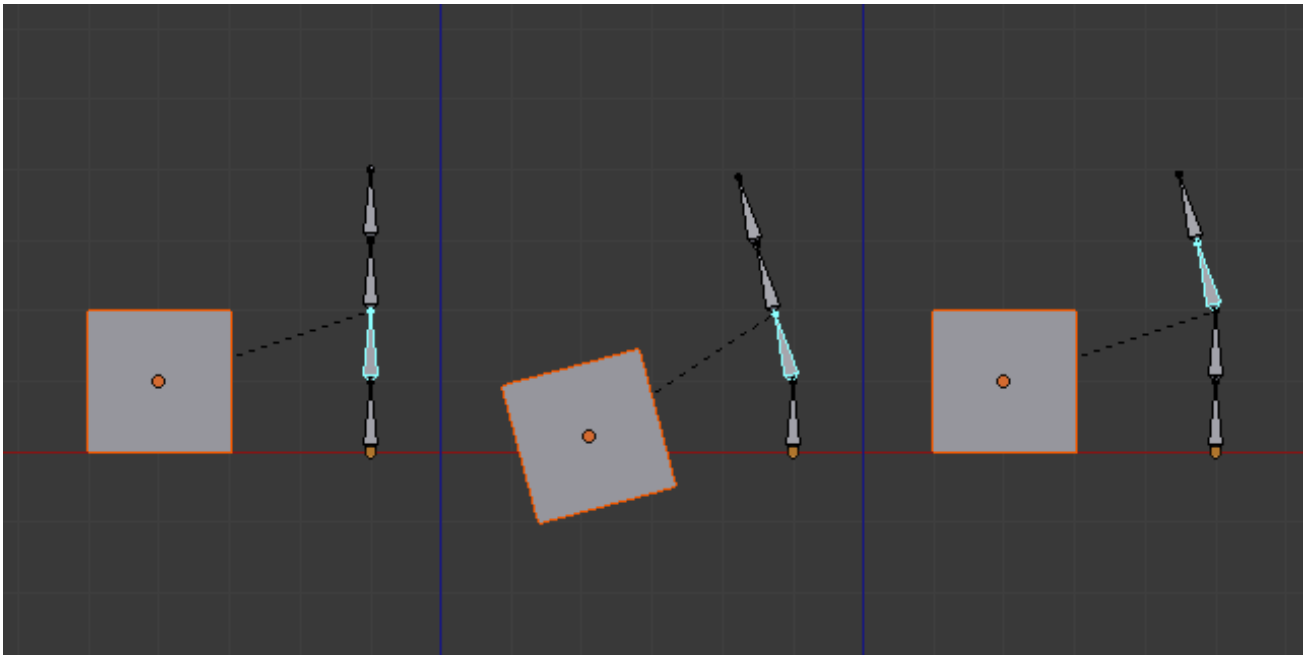


Fig. 2.165: Three pictures of Armatures with four Bones.

In Fig. *Three pictures of Armatures with four Bones.* with the 2nd bone being the Bone Parent of the Child Object Cube. The Cube is only transformed if the 1st or 2nd bones are. Notice altering the 3rd and 4th bones has no effect on the Cone.

To use Bone Parenting, you must first select all the Child Objects you wish to parent to a specific Armature Bone, then `Shift-RMB` select the Armature Object and switch it into Pose Mode and then select the specific bone you wish to be the Parent Bone by `RMB` selecting it. Once done press `Ctrl-P` and select Bone from the Set Parent To pop-up menu.

Now transforming that bone in Pose Mode will result in the Child Objects also transforming.

Relative Parenting

Bone Relative parenting is an option you can toggle for each bone. This works in the same way as Bone parenting with one difference.

With Bone parenting if you have parented a bone to some Child Objects and you select that bone and switch it into Edit Mode and then translate that bone; When you switch back into Pose Mode on that bone, the Child Object which is parented to that bone will snap back to the location of the bone in Pose Mode.

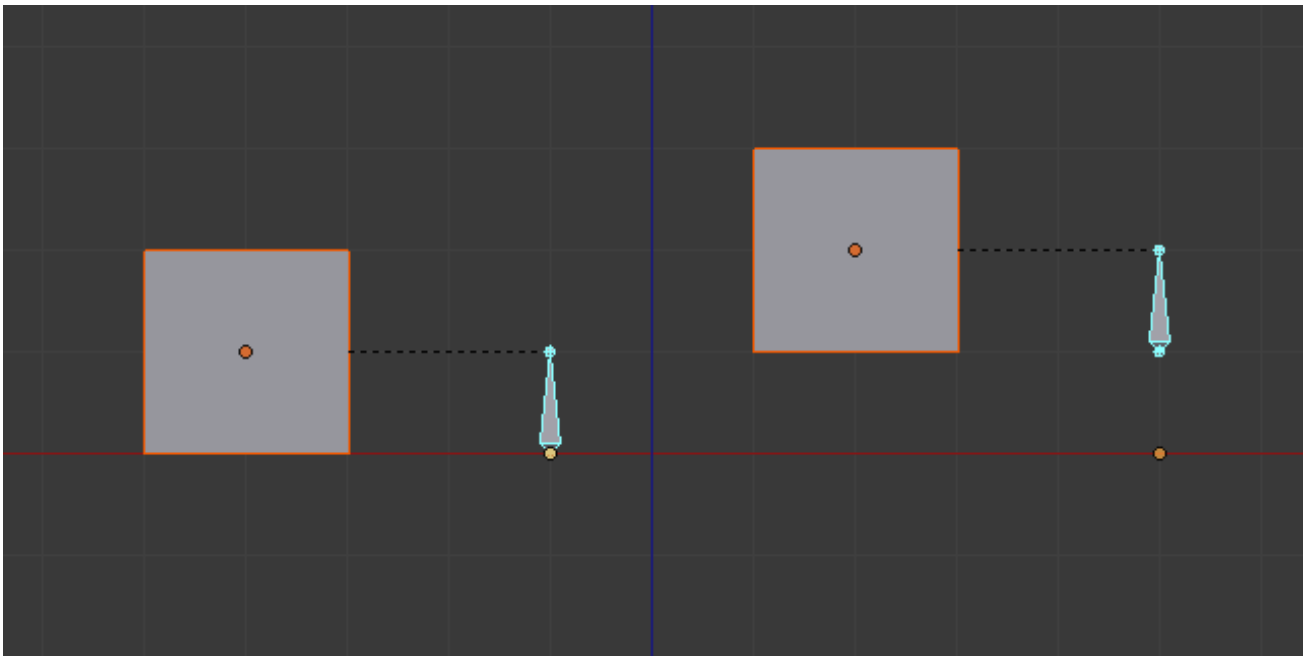


Fig. 2.166: Single Armature Bone which has a Child Object cube parented to it using Bone parenting.

In Fig. *Single Armature Bone which has a Child Object cube parented to it using Bone parenting.* the 1st picture shows the position of the cube and armature before the bone is moved in Edit Mode. 2nd picture shows the position of the cube and armature after the bone was selected in Edit Mode, moved and switched back into Pose Mode. Notice that the Child Object moves to the new location of the Pose Bone.

Bone Relative parenting works differently; If you move a Parent Bone in Edit Mode, when you switch back to Pose Mode, the Child Objects will not move to the new location of the Pose Bone.

In Fig. *Single Bone with Bone Relative parent to a cube.* the 1st picture shows the position of the cube and armature before the bone is moved in Edit Mode. 2nd picture shows the position of the cube and armature after the bone was selected in Edit Mode, moved and switched back into Pose Mode. Notice that the Child Object does not move to the new location of the Pose Bone.

Vertex Parent

You can parent an object to a single vertex or a group of three vertices as well; that way the child/children will move when the parent mesh is deformed, like a mosquito on a pulsing artery.

Vertex Parent from Edit Mode

In *Object Mode*, select the child/children and then the parent object. `Tab` into *Edit Mode* and on the parent object select either one vertex that defines a single point, or select three vertices that define an area (the three vertices do not have to form a complete face; they can be any three vertices of the parent object), and then press `Ctrl-P` and confirm.

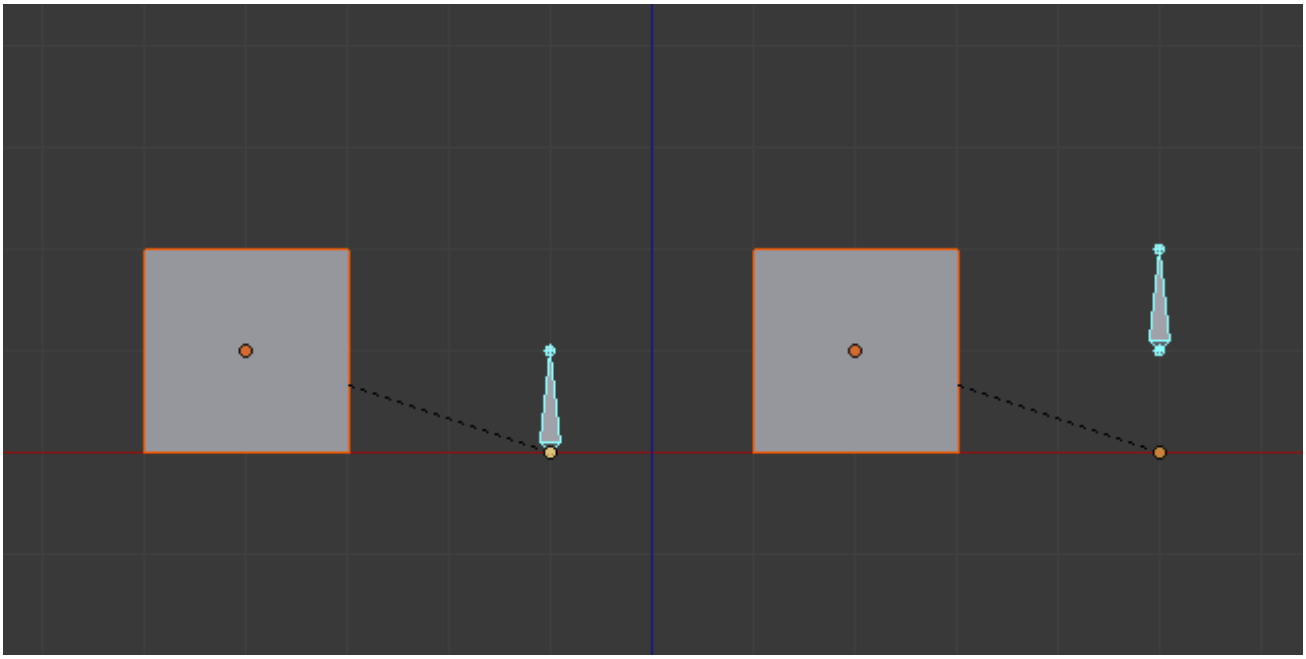


Fig. 2.167: Single Bone with Bone Relative parent to a cube.

At this point, if a single vertex was selected, a relationship/parenting line will be drawn from the vertex to the child/children. If three vertices were selected then a relationship/parenting line is drawn from the averaged center of the three points (of the parent object) to the child/children. Now, as the parent mesh deforms and the chosen parent vertex/vertices move, the child/children will move as well.

Vertex Parent from Object Mode

Vertex parenting can be performed from object mode, This is done like regular object parenting, Press `Ctrl-P` in object mode and select *Vertex* or *Vertex (Triangle)*.

The nearest vertices will be used from each object which is typically what you would want.

The parent context menu item means users can rapidly set up a large number of vertex parent relationships, and avoid the tedious effort of establishing each parent-child vertex relationship separately.

Note: It is in fact a sort of "reversed" *hook*

Options

Move child

You can *move* a child object to its parent by clearing its origin. The relationship between the parent and child is not affected. Select the child object and press `Alt-O`. By confirming the child object will snap to the parent's location. Use the *Outliner* view to verify that the child object is still parented.

Remove relationship/Clear Parent

You can *remove* a parent-child relationship via `Alt-P`

The menu contains:

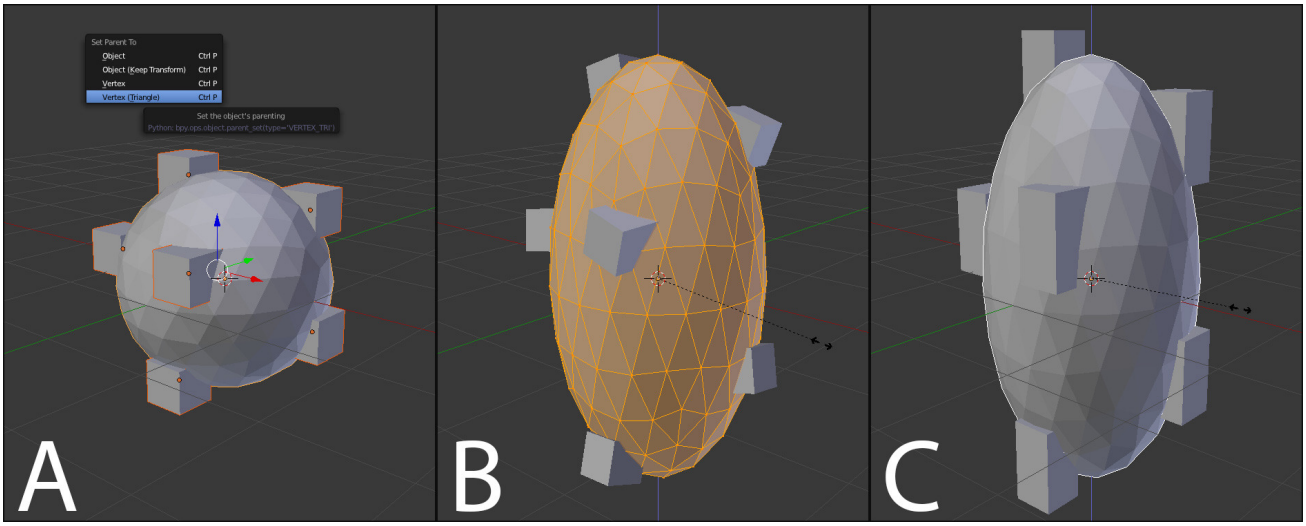


Fig. 2.168: Vertex Parent example.

1. The small cubes can each be automatically parented to a triad of nearby vertices on the icosphere using the "Vertex (Triangle)" in the set parent context menu.
2. Reshaping the object in edit mode then means each of the cubes follows their vertex parent separately.
3. Re-scaling the parent icosphere in object mode means the child cubes are also rescaled as expected.

Clear Parent If the parent in the group is selected nothing is done. If a child or children are selected they are disassociated from the parent, or freed, and they return to their *original* location, rotation, and size.

Clear and Keep Transformation Frees the children from the parent, and *keeps* the location, rotation, and size given to them by the parent.

Clear Parent Inverse Places the children with respect to the parent as if they were placed in the Global reference. This effectively clears the parent's transformation from the children. For example, if the parent is moved 10 units along the X axis and *Clear Parent Inverse* is invoked, any selected children are freed and moved -10 units back along the X axis. The "Inverse" only uses the last transformation; if the parent moved twice, 10 units each time for a total of 20 units, then the "Inverse" will only move the child back 10 units, not 20.

Hints

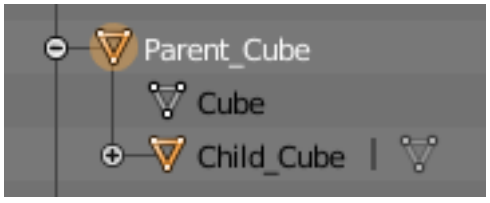


Fig. 2.169: Outliner view.

There is another way to see the parent-child relationship in groups and that is to use the *Outliner* view of the *Outliner editor*. Fig. *Outliner view.* is an example of what the *Outliner* view looks like for the figurers in the *Object Parent* example. Cube A's object name is "Cube_Parent" and cube B is "Cube_Child".

Grouping objects

There can be many objects in a scene: A typical stage scene consists of furniture, props, lights, and backdrops. Blender helps you keep everything organized by allowing you to group like objects together.

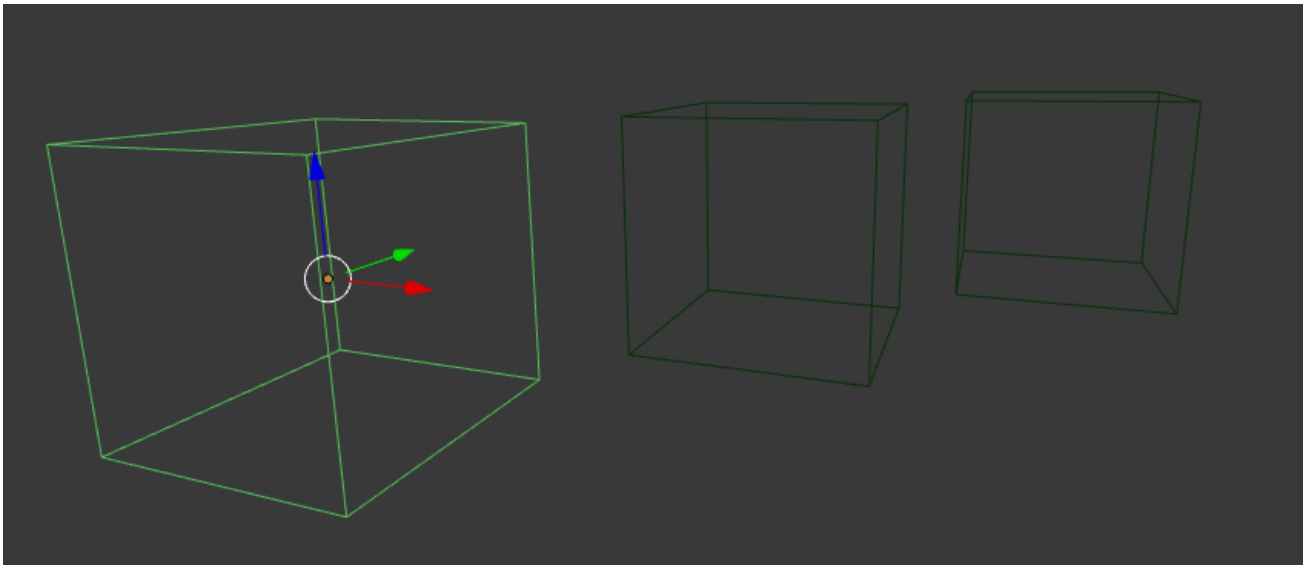


Fig. 2.170: Grouped objects.

Group objects together without any kind of transformation relationship. Use groups to just logically organize your scene, or to facilitate one-step appending or linking between files or across scenes. Objects that are part of a group always shows as light green when selected. See Fig. *Grouped objects.*.

Options

Creating a Group `Ctrl-G` creates a new group and adds the selected object(s) to it.

Naming a Group All groups that an object has been assigned to are listed in the Properties editor *Object tab* → *Relations panel*. To rename a group, simply click in the groups name field. To name groups in the *Outliner* editor, select *Groups* as the outliner display from the header combo box, and `Ctrl-LMB` click on the group name. The name will change to an editable field; make your changes and press `Return`.

Restricting Group Contents via Layers The cluster of layer buttons attached to each group determines from which layers the group objects will be included when duplicated. If your group contains objects on layers 10, 11 and 12, but you disable the layer 12 button in the group controls, duplicates of that group (using the *Dupligroup* feature) will only show the portions of the group that reside in layers 10 and 11.

Appending or Linking Groups To append a group from another blend-file, consult *this page*. In summary, *File* → *Link/Append Link* Select a blend-file and, and then the group.

Removing Groups To remove an object from a group, in the Properties editor *Object* tab, open the "Groups" panel. Find the name of the group from which you wish to remove the object, and click the x to the right of the group name.

Tip: Selecting Groups

Groups can be selected, see *Select Grouped* for more information.

Relations Extras

Tracking Axes TODO.

Axis Axis that points in the "forward" direction. (Applies to *DupliFrame* when parent *Follow* is enabled).

Up Axis Axis that points in the upward direction. (Applies to *DupliFrame* when parent *Follow* is enabled).

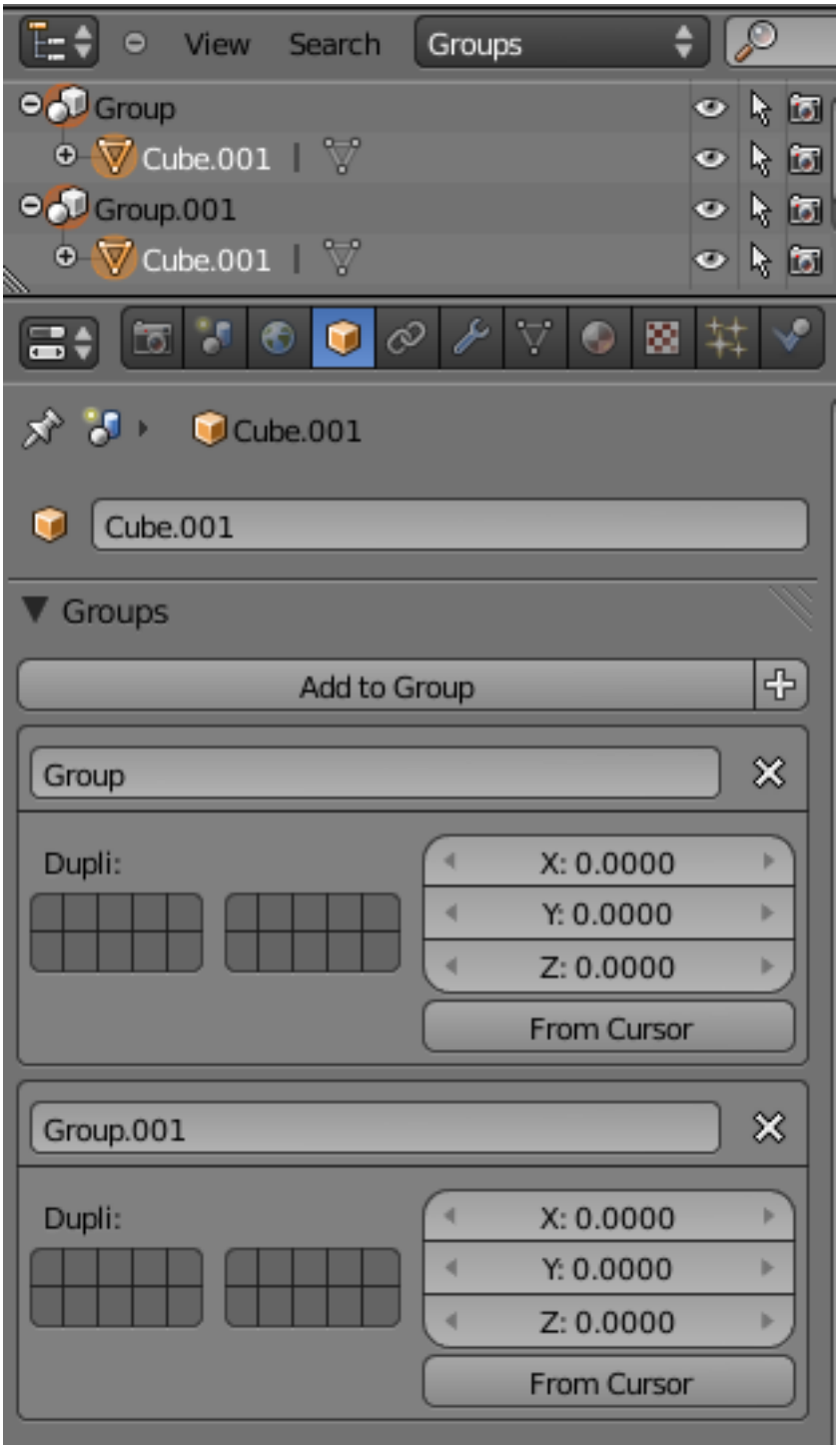


Fig. 2.171: Naming a Group.

Show Parent Create a delay in the parent relation.

> **Warning:** This may be unsafe for renderfarms as it may be invalid after jumping around the time line.

Offset Delay in the parent offset.

Extra Object Update Refresh the object agian on frame changes.

Extra Data Update Refresh the object's data again on frame changes.

Display

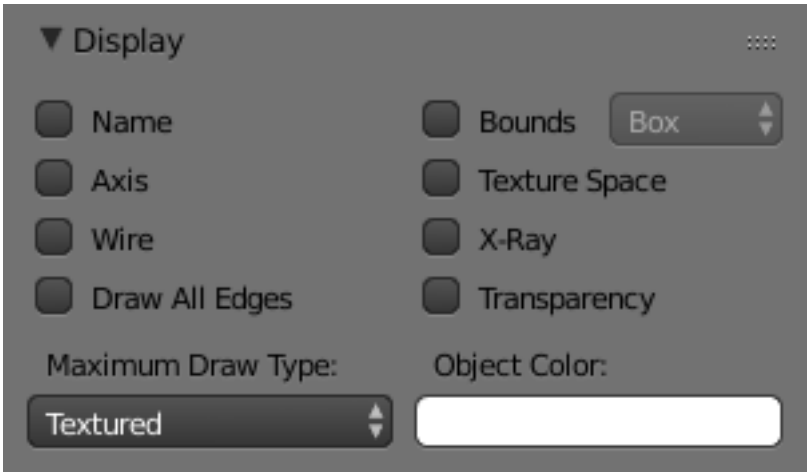


Fig. 2.172: Display panel.

The *Display* panel is used to enable extra drawing or viewing options for the 3D View.

Name Displays the name of the object in the 3D View.

Axis Displays a object similar to an *Empty* that shows the object's axis.

Wire Draws an object's wire frame on top of the solid drawing.

Draw All Edges Displays all edges for mesh objects.

Bounds Displays a bounding box around an object.

Draw Bounds Type TODO.

Texture Space Displays the objects *Texture Space*.

X-Ray Makes the object draw in front of others. (Unsupported for duplicator drawing).

Transparency Displays material transparency in the object. (Unsupported for duplicator drawing).

Maximum Draw Type The Maximum shading mode to display in the 3D View. This can be useful if you have a high poly object that is slowing down performance.

Object Color Allows to specify the color used when the *Object Color* in *Material Options* is enabled.

Duplication

There are currently four ways in Blender to procedurally duplicate objects. These options are located in the *Object* menu.

Verts This creates an instance of all children of this object on each vertex (for mesh objects only).

Faces This creates an instance of all children of this object on each face (for mesh objects only).

Group This creates an instance of the group with the transformation of the object. Group duplicators can be animated using actions, or can get a *Proxy*.

Frames For animated objects, this creates an instance on every frame. As you will see on this topic's subpage, this is also a *very* powerful technique for arranging objects and for modeling them.

DupliFrames

DupliFrames is a tool to duplicate objects at frames distributed along a path. This is a useful tool to quickly arrange objects.

Examples

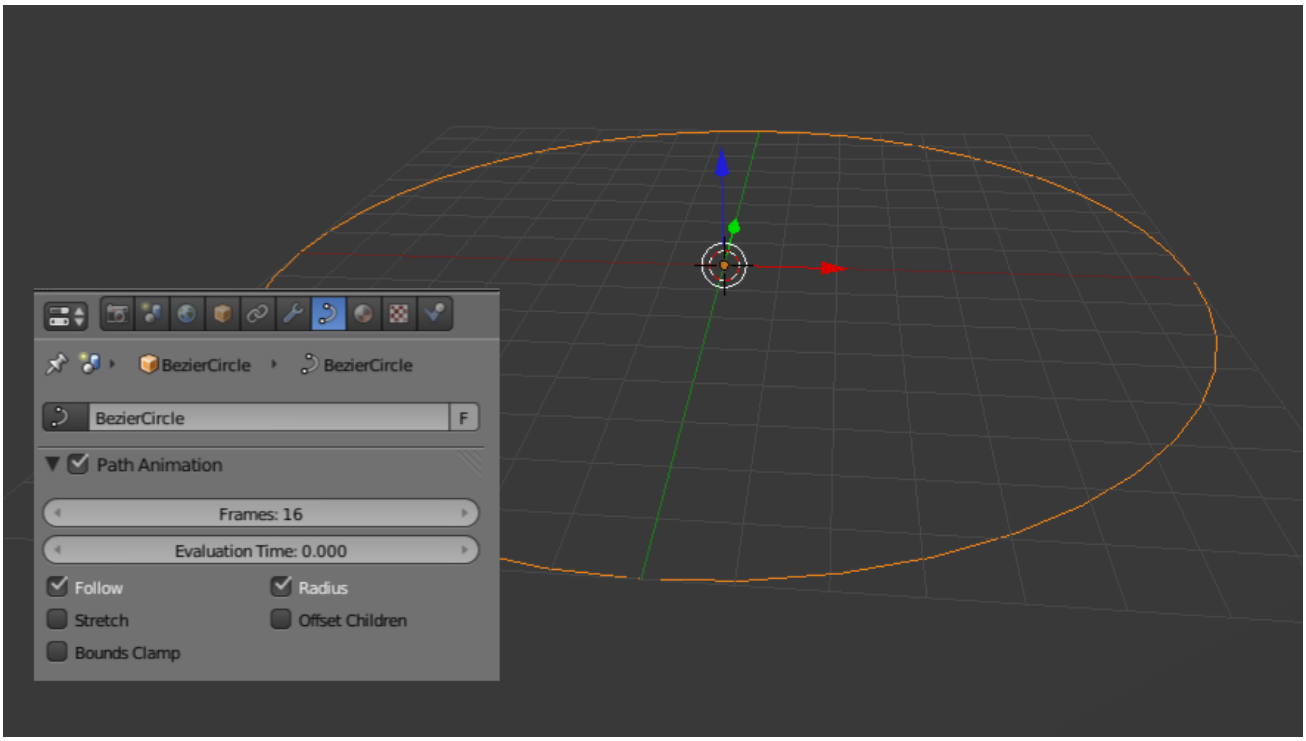


Fig. 2.173: Settings for the curve.

`Shift-A` to add a *Bézier Circle* and scale it up. In the *Curve* menu under *Path Animation* enable *Follow* and set *Frames* to something more reasonable than 100 (say 16).

Add a *Monkey*. In the *Object* menu under *Duplication* enable *Frames* and disable *Speed*.

Note: Speed

The *Speed* option is used when the parent-child relationship is set to *Follow Path* (see below). In this example, the monkey will then travel along the circle over 16 frames.

To parent the monkey to the Bézier circle, first select the monkey then the curve (so that the curve is the active object) and `Ctrl-P`. Select the monkey and `Alt-O` to reset its origin.

You can now change the orientation of the monkey by either rotating it (either in *Edit Mode* or *Object Mode*) or by changing the *Tracking Axes* under *Relations Extras* (with the monkey selected). The arrangement of monkeys can, of course, be further enhanced by editing the curve.

To transform all monkeys into real objects, first `Ctrl-Shift-A` to *Make Duplicates Real*. All monkeys are now real objects, but still linked copies. To change this, *Object* → *Make Single User* → *ObjectData* → *All*.

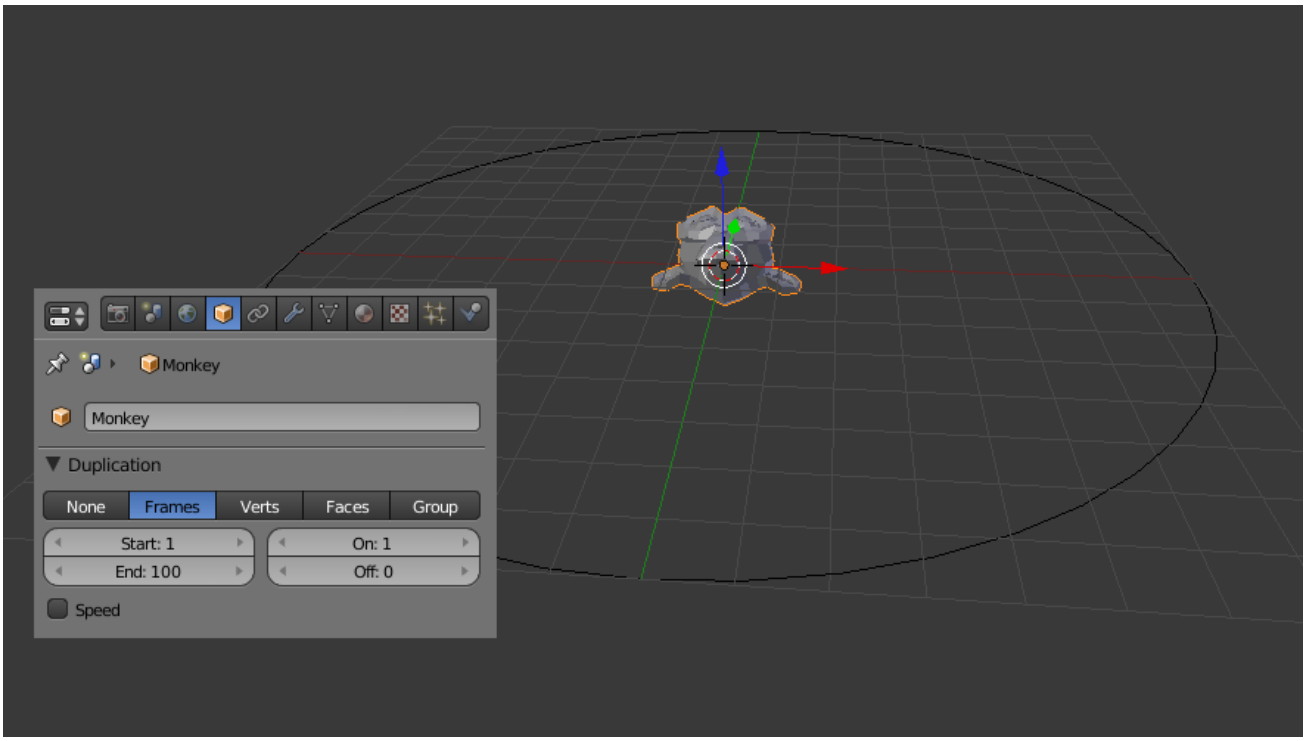


Fig. 2.174: Settings for the object.

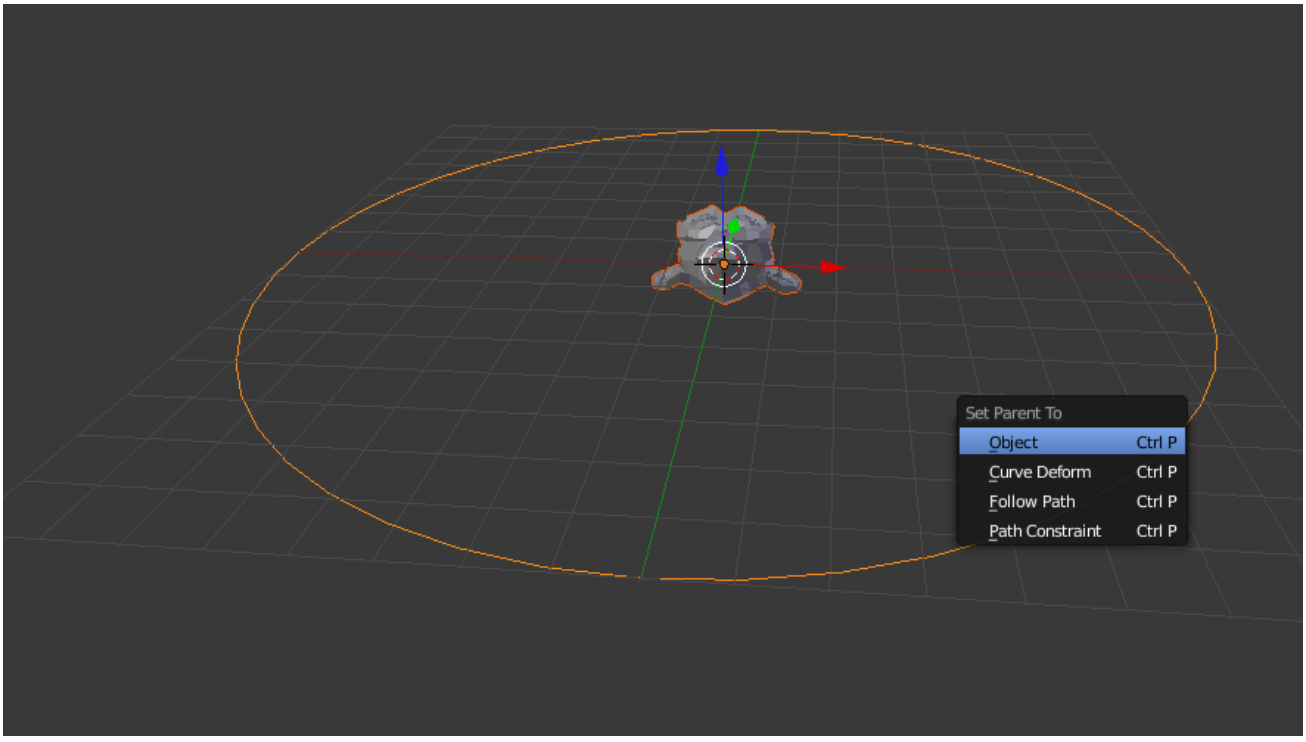


Fig. 2.175: Parenting.

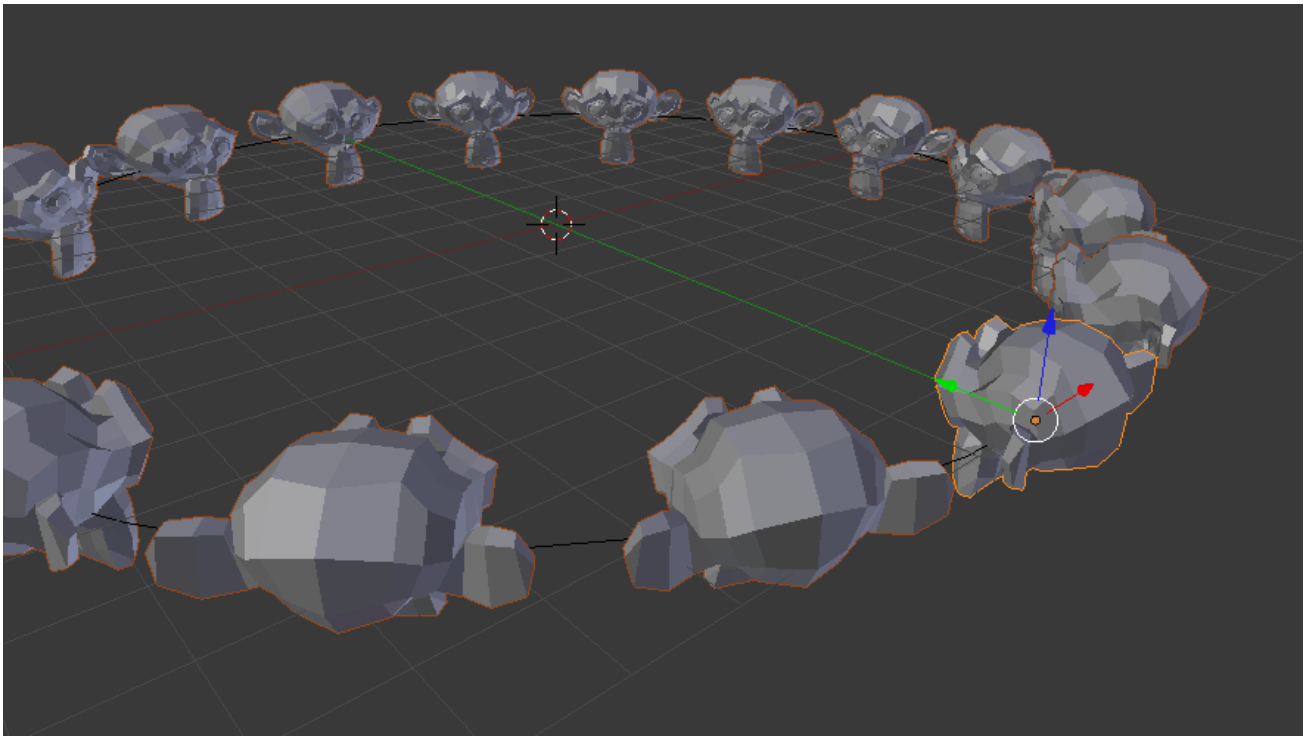


Fig. 2.176: Orientation tweaks.

Note: There are many alternatives to Dupliframes. Which tool to use depends on context:

- To use a small curve as a profile and a larger curve as a path, simply use the former as a *Bevel Object* to the latter.
- To arrange objects along a curve, combining an *Array Modifier* and a *Curve Modifier* is often useful.
- Dupliverts can be used to arrange objects, for example, along a circle or across a subdivided plane.

See also:

Blender Artists: Dupliframes in 2.5

DupliVerts

Reference

Mode: Object Mode
Panel: *Object* → *Duplication*

Duplication Vertices or *DupliVerts* is the duplication of a base object at the location of the vertices of a mesh. In other words, when using *DupliVerts* on a mesh, an instance of the base object is placed on every vertex of the mesh.

There are actually two approaches to modeling using *DupliVerts*. They can be used as an arranging tool, allowing to model geometrical arrangements of objects (e.g. the columns of a Greek temple, the trees in a garden, an army of robot soldiers, the desks in a classroom). The object can be of any object type which Blender supports. The second approach is to use them to model an object starting from a single part of it (e.g. the spikes in a club, the thorns of a sea-urchin, the tiles in a wall, the petals in a flower).

Note: Download example blend-file

You can download a file with the examples described on this page. In this blend, the first example, a monkey parented to a circle is on layer 1; while a tentacle parented to an icosphere is on layer 2.

DupliVerts as an Arranging Tool

Setup

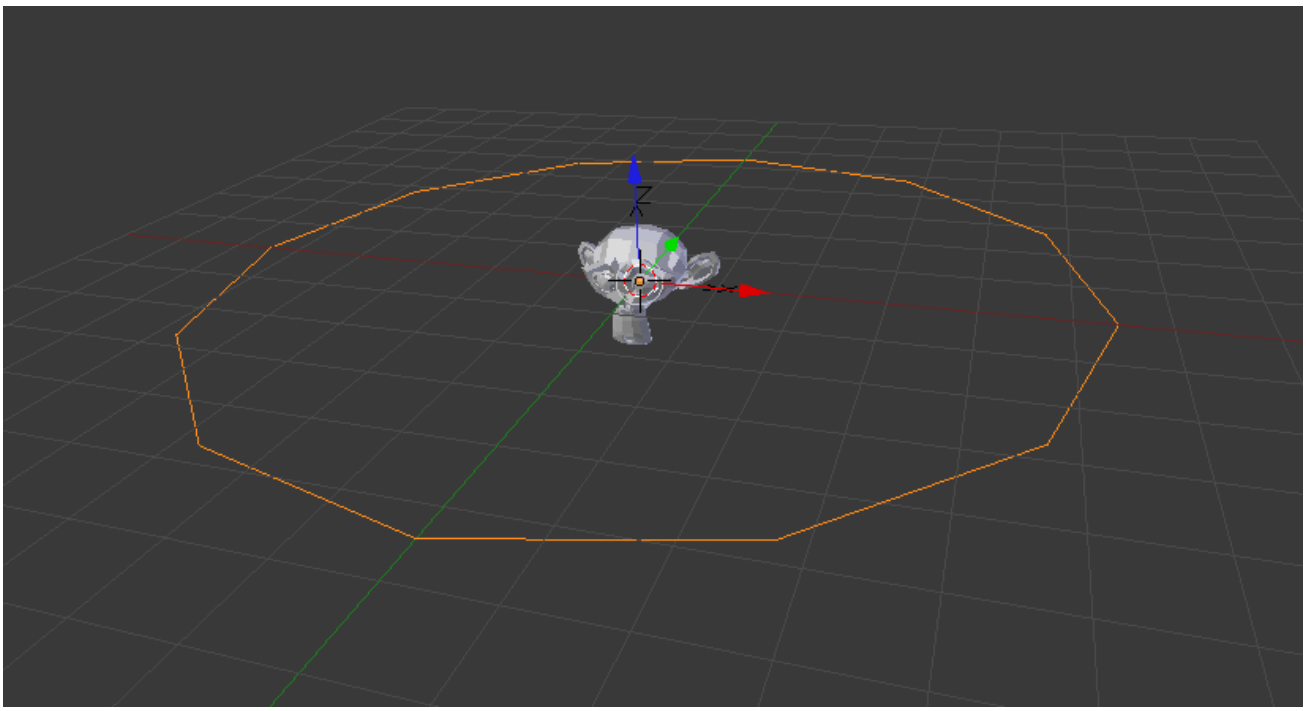


Fig. 2.177: A monkey head and a circle.

All you need is a base object (e.g. the *tree* or the *column*) and a pattern mesh with its vertices following the pattern you have in mind. In this section, we will use a simple scene for the following part. We will be using a monkey head located at the origin of the coordinate system as our base object and a circle at the same location as our parent mesh.

First, in *Object Mode*, select the base object and `Shift-RMB` to add the circle to the selection (order is very important here), and `Ctrl-P` to parent the base object to the circle. Now, the circle is the parent of the monkey; if you move the circle, the monkey will follow it.

With only the circle selected, enable *Duplication vertices* in the *Object panel* → *Duplication* → *Verts*. A monkey head should be placed at every vertex of the circle.

The original monkey head at the center and the parent mesh are still shown in the 3D View but neither will be rendered. If the placement and rotation of your monkey head is odd, you might need to clear its rotation `Alt-R`, scale `Alt-S`, location `Alt-G`, and origin `Alt-O`.

Rearranging

If you now select the base object and modify it in either object or edit mode, all changes will also affect the shape of all duplicate objects. You can also select the parent mesh to modify the arrangement of the duplicates; adding vertices will also add more base objects.

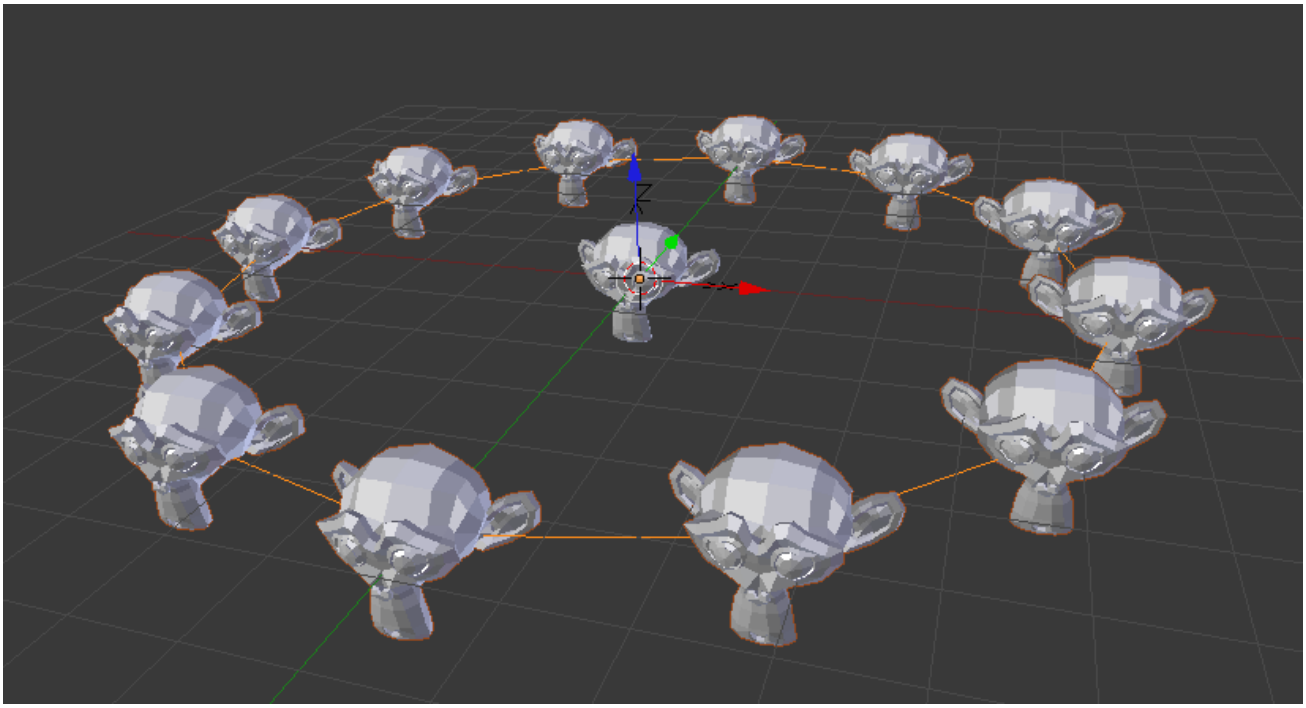


Fig. 2.178: Dupliverted monkeys.

Note that the base objects will inherit changes made to the parent mesh in Object Mode, but not in Edit Mode. So scaling the circle up in object mode will enlarge the monkey head, while scaling the circle up in edit mode will only increase the distance between the base objects.

Orientation

The orientation of the base objects can be controlled by enabling *Rotation* in the *Duplication* panel. This will rotate all base objects according to the vertex normals of the parent mesh.

To change the orientation of the duplicated objects, select the base object and in the *Object* → *Relations extras* panel change the *Tracking Axes*.

Output of various orientations:

Note: The axes of an object can be made visible in the *Object* → *Display* panel. To display the vertex normals of the parent mesh, tab into edit mode and enable this function in *Properties* → *Display* panel where you can also resize the displayed normals as necessary.

DupliVerts as a Modeling Tool

Very interesting models can be made using DupliVerts and a standard primitive. In this example, a simple tentacle was made by extruding a cube a couple of times. The tentacle object was then parented to an icosphere. With dupli *Rotation* enabled for the parent mesh (the icosphere), the orientation of the base object (the tentacle) was adapted to the vertex normals of the parent mesh

(in this case the tentacle was rotated -90° about the X axis in edit mode).

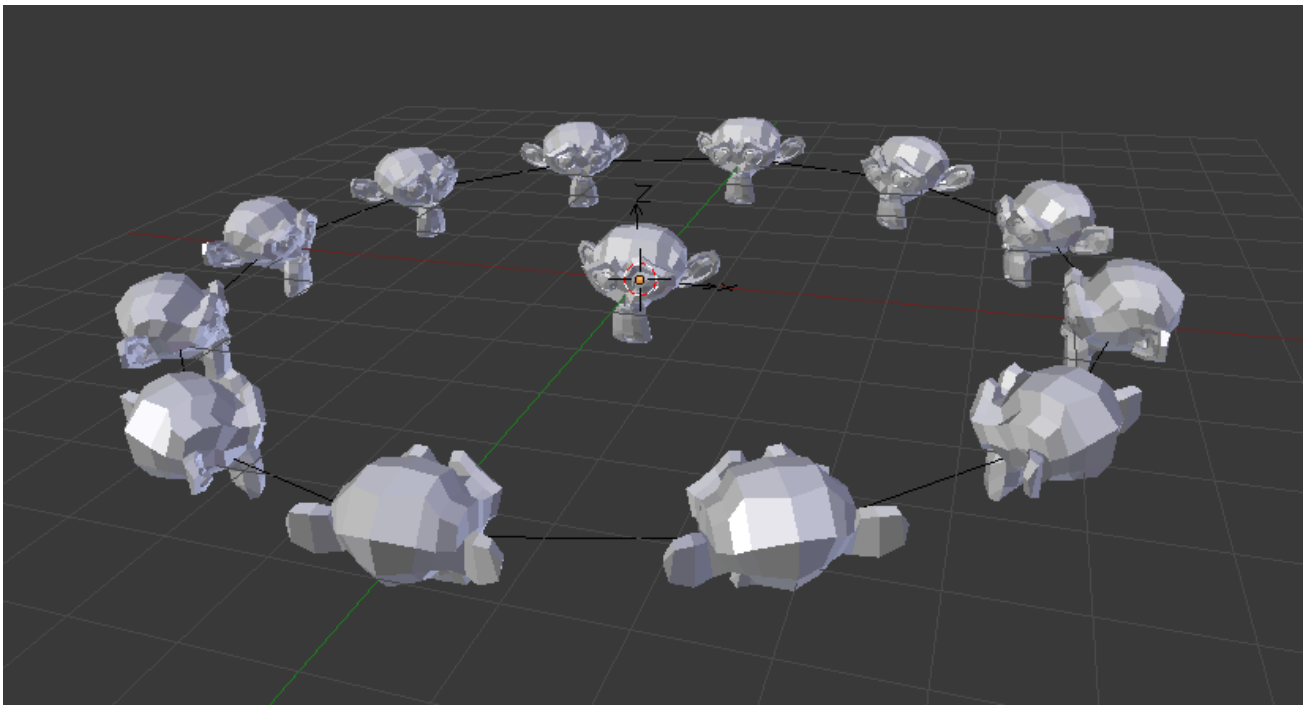


Fig. 2.179: Orientation enabled, orientation +Y.

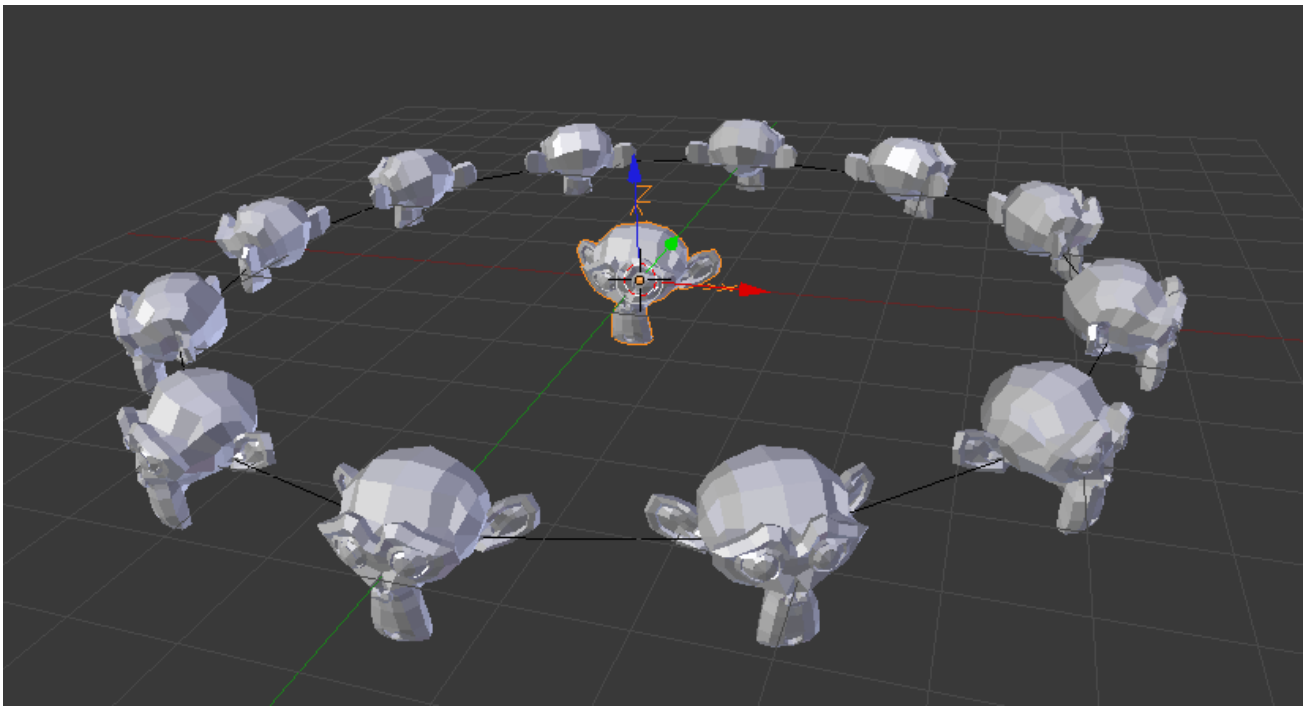


Fig. 2.180: Negative Y.

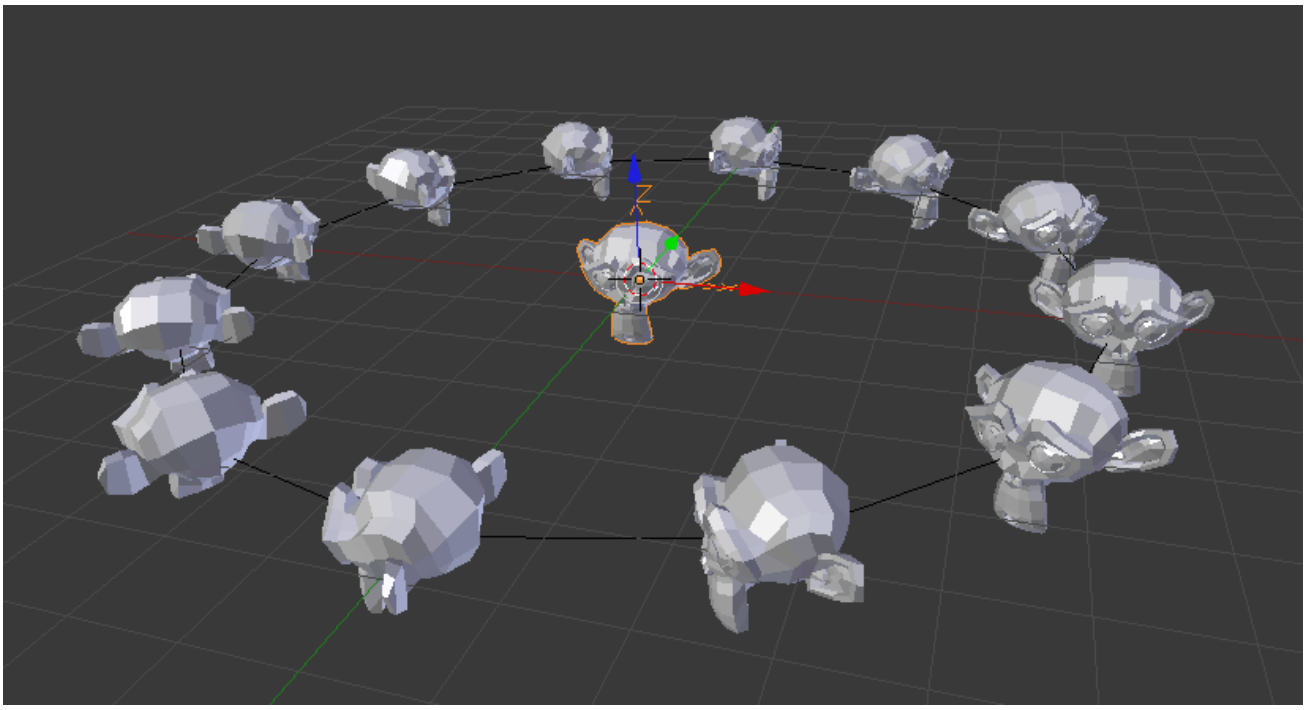


Fig. 2.181: Positive X.

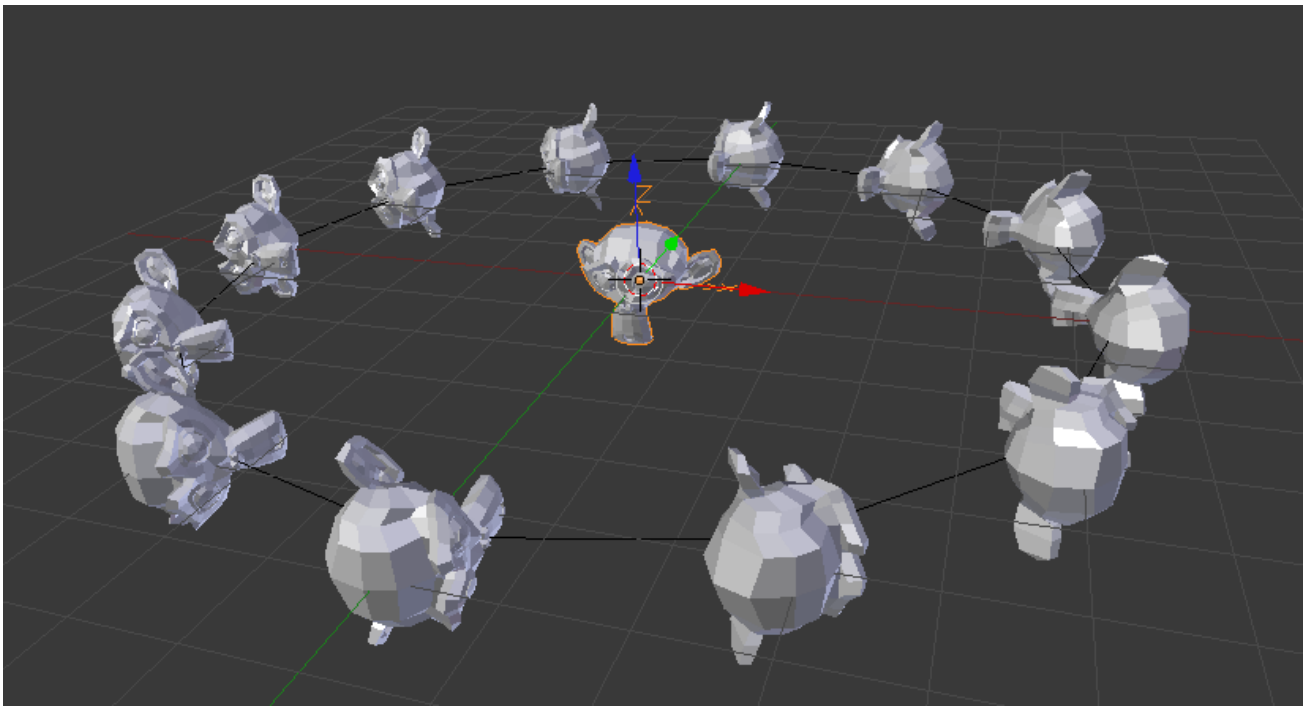


Fig. 2.182: Positive Z, up X.

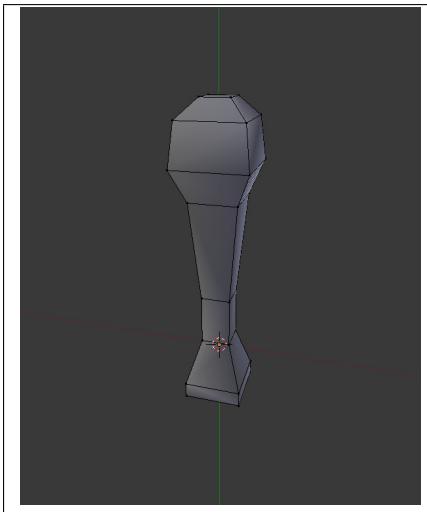


Fig. 2.183: A simple tentacle set to smooth.

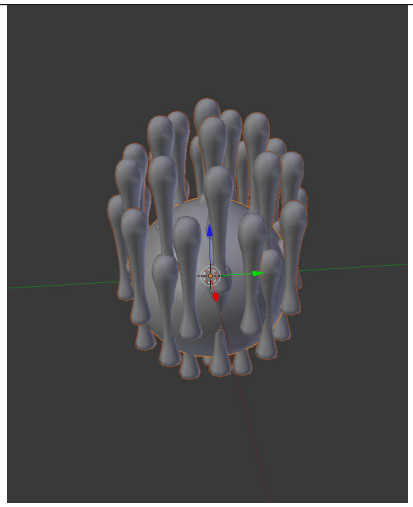


Fig. 2.184: Tentacle dupliverted onto the parent mesh.

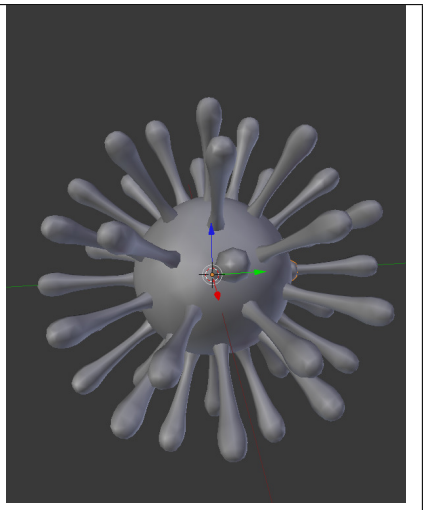


Fig. 2.185: Rotation enabled to align duplicates.

As in the previous example, the shape and proportions of the arrangement can now be tweaked.

To turn all duplicates into real objects, simply select the icosphere and *Object* → *Apply* → *Make Duplicates Real*, `Ctrl-Shift-A`. To make the icosphere and the tentacle a single object, make sure they are all selected and go to *Object* → *Join*, `Ctrl-J`.

See also:

Other duplication methods are listed *here*.

DupliFaces

Reference

Mode: Object Mode
Panel: *Object* → *Duplication*

Duplication Faces or *DupliFaces* is the capability to replicate an object on each face of a parent object. One of the best ways to explain this is through an example illustration.

Note: Example blend-file

Download the blend-file used for the examples on this page here.

Basic Usage

In this example we will use a UV sphere with an extruded "north pole" as our base object and cube as our parent mesh. To parent the sphere to the cube, in *Object Mode*, first `RMB` select the sphere, then `Shift-RMB` select the cube (order is very important here), and finally `Ctrl-P` to parent.

Next, in the *Object tab* → *Duplication panel*, enable *Faces*. The sphere is duplicated one for each face of the cube.

Note: Inherited properties

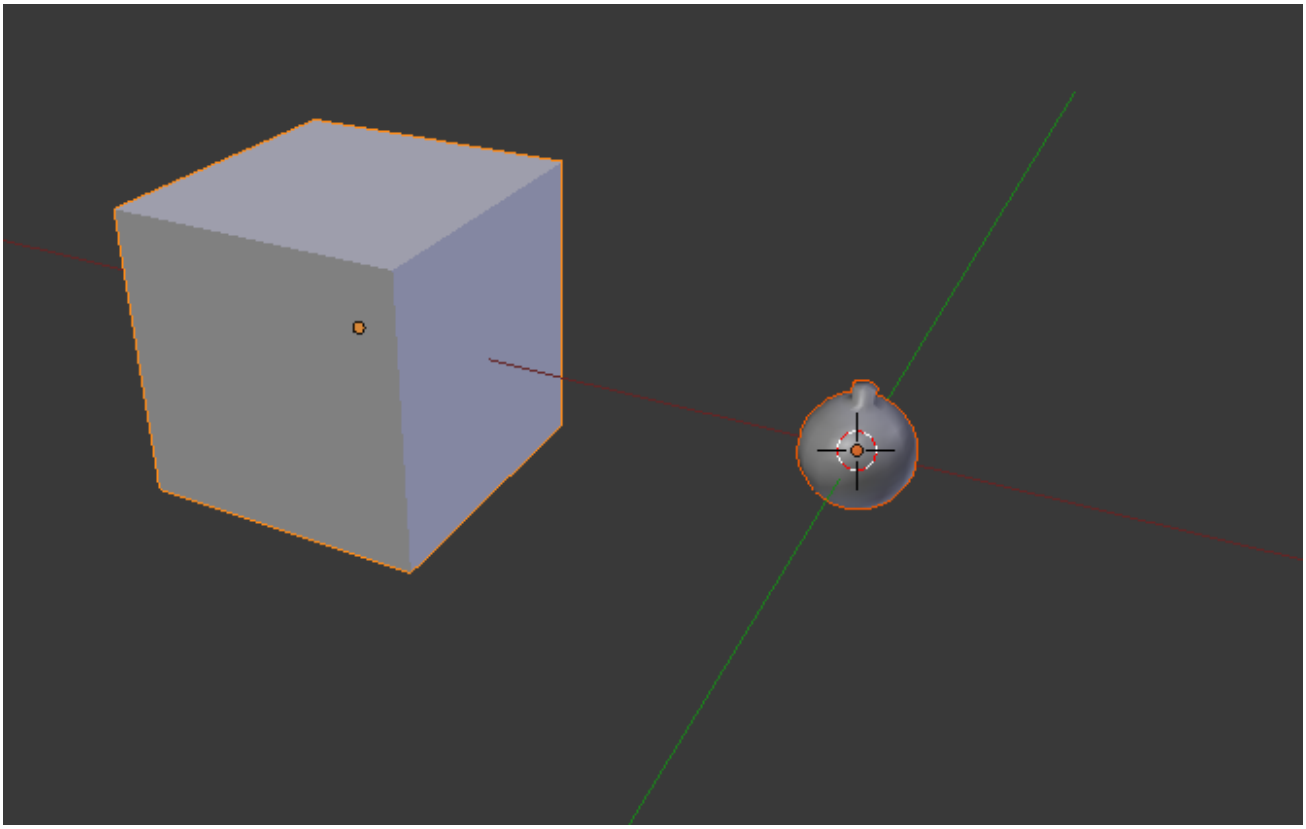


Fig. 2.186: A cube and a sphere.

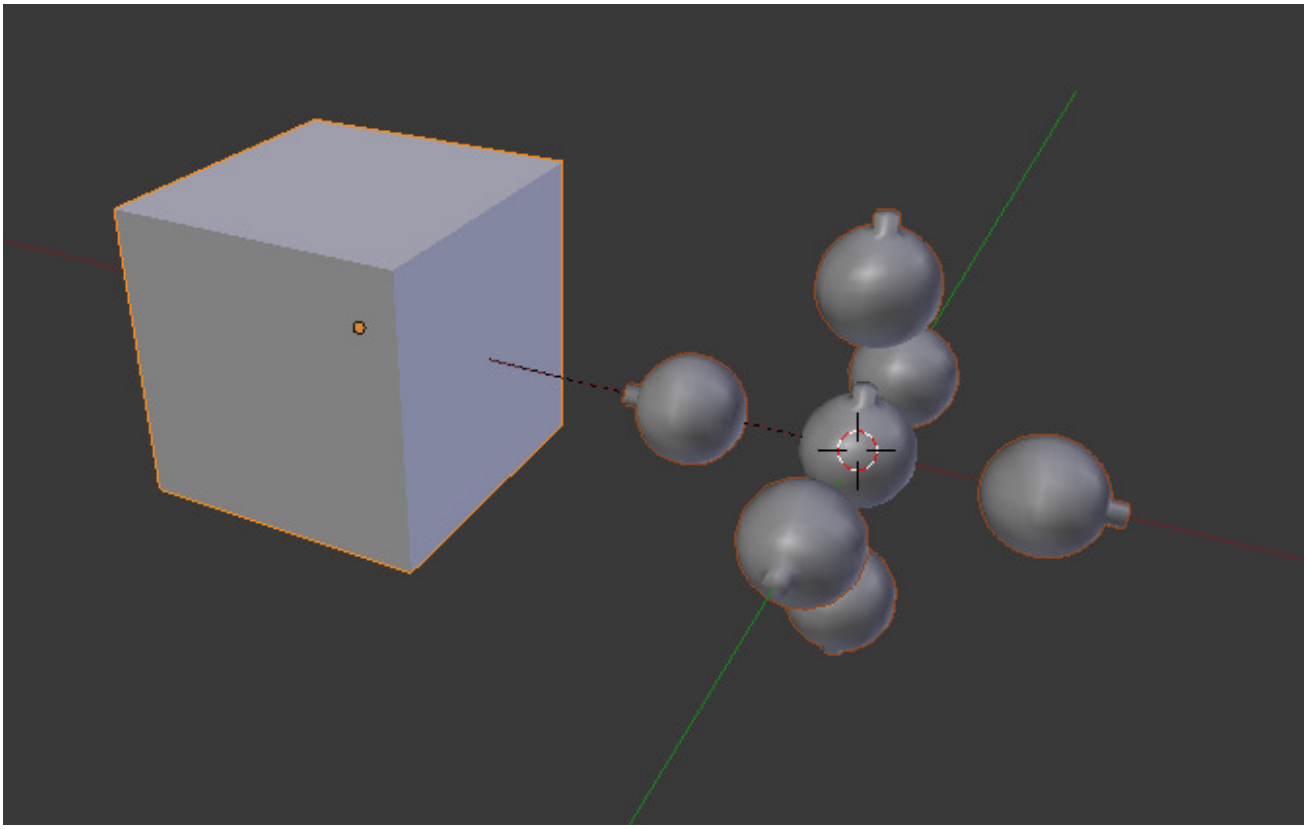


Fig. 2.187: Duplication Faces applied to the cube.

The location, orientation, and scale of the duplicated child(ren) matches that of the faces of the parent. So, if several objects are parented to the cube, they will all be duplicated once for each face on the cube. If the cube is subdivided (in *Edit Mode* W), every child will be duplicated for each face on the cube.

Both the parent object and original are displayed as editable "templates" in 3D View, but neither is rendered.

Scale

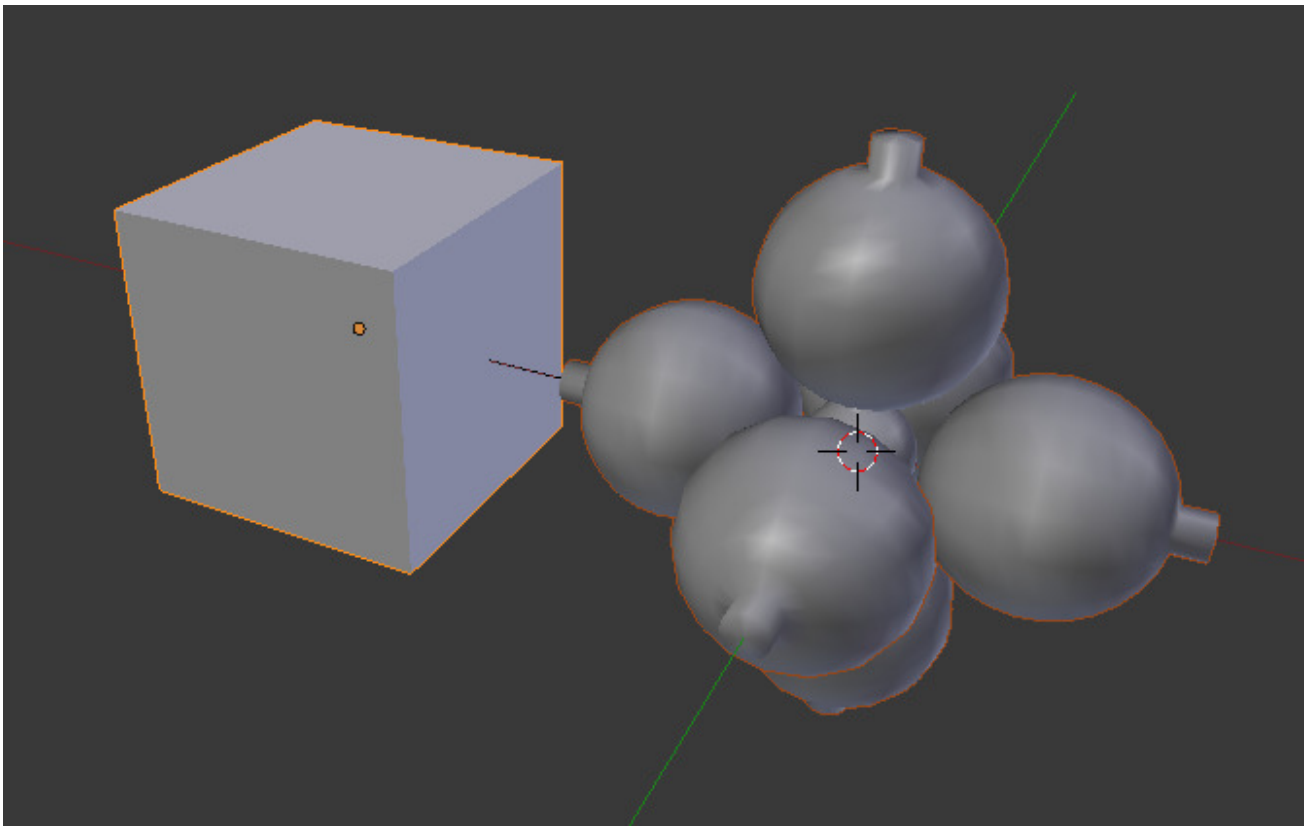


Fig. 2.188: Scale enabled.

By enabling *Scale* for the parent object, the scale of the child objects will be adapted to the size of each face in the parent object.

Thus, by rescaling the face of the parent object, the size of the duplicated object will change accordingly.

Limitations/Considerations

The positioning of the duplicated geometry relative to the face is dependent upon the position of the child objects relative to the duplicator's origin. This can lead to some visual artifacts in the editor as the geometry of the original objects overlaps or intersects with the duplicates. One workaround is to move the origin of the duplicator mesh off of the plane of the faces.

If the geometry of the children is not symmetrical then the orientation of the face (as determined by the order of its vertices) could matter. As of 2.70 Blender does not have tools which allow you to adjust the ordering of the vertices on a face.

However, there is a workflow that lets you control for this. Make a single square and enable the Duplication/Faces so you can see the duplicated geometry in your editor. If the orientation is not what you want, rotate the face until it is how you want. Typically you want to do the rotation in Edit Mode, not Object Mode, but this is not a hard rule.

Once you have the orientation correct, Duplicate the face and move the duplicate where you want it. Repeat this process until you have enough faces. Since it is common for these faces to butt up against one another, your geometry will have lots of duplicate vertices. Use the Remove Doubles button in the Tools panel.

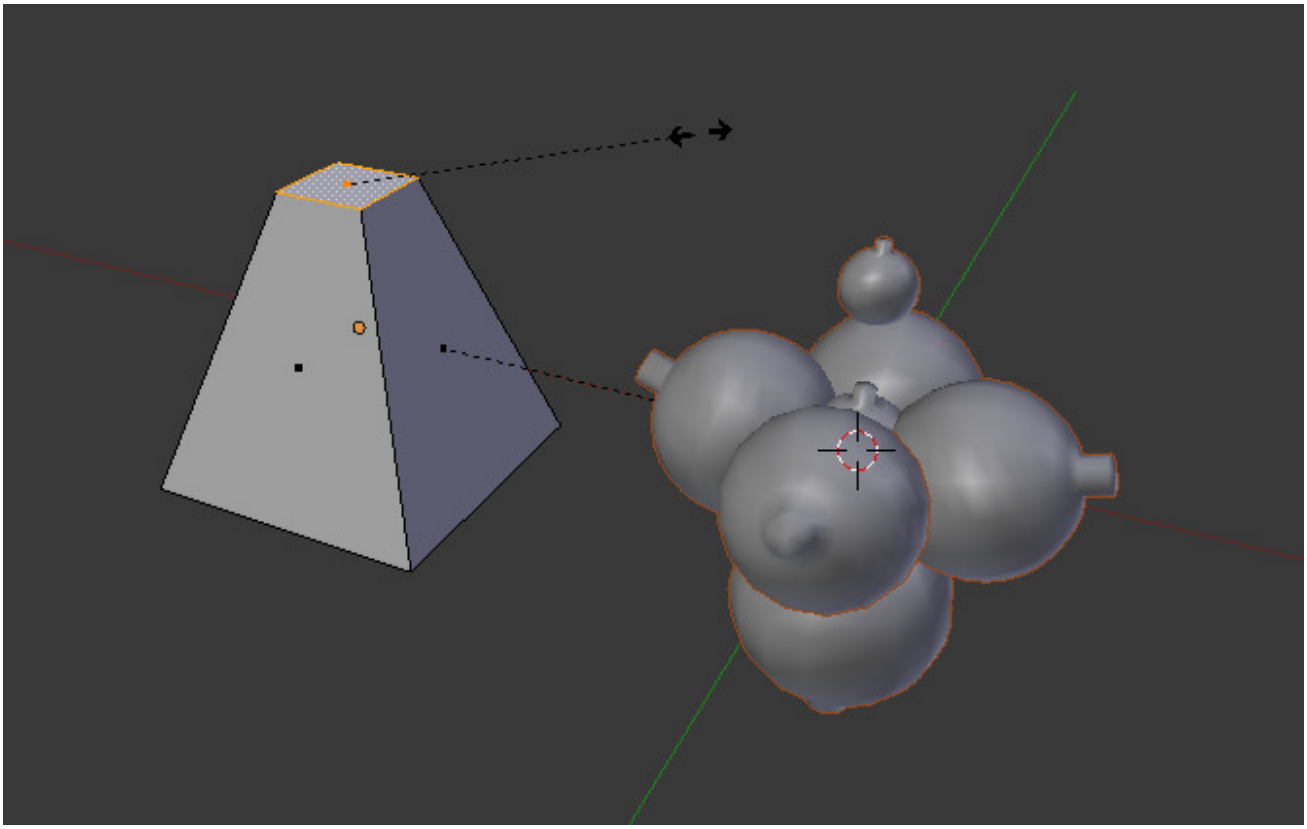


Fig. 2.189: Top face of cube scaled down.

A short video illustrating this workflow:

DupliGroup

Reference

Mode: Object Mode
Panel: *Object* → *Duplication* → *Group*

Duplication Group or *DupliGroup* allows you to create an instance of a group for each instance of another object.

DupliGroups may contain animations, objects with physics simulations and even other nested *DupliGroups*.

Basic Usage

Create a Group:

- Selecting the objects to be grouped.
- Create a new group *Object* → *Group* → *Create New Group*
- Rename your group in the properties editor: *Object* –> *Groups*

Create a new Group Instance:

- *Add* → *Group Instance*

Change the Group Instance of existing objects:

- In the properties editor: *Object* → *Duplication*, enable *Group*.
- Select the name of your newly created group.

At this point, an instance of the group will appear. You can duplicate the empty, and the DupliGroup settings will be preserved for each empty. This way, you can get multiple copies of linked data very easily.

DupliGroup and Dynamic Linking

See *Appending and Linking* to understand how to dynamically link data from another blend-file into the current file. You can dynamically link groups from one blend-file to another. When you do so, the linked group does not appear anywhere in your scene until you create an object controlling where the group instance appears.

Warning: Material Transparency will not display when instancing dupli-groups; this is a known limitation of Blender's viewport.

Making a DupliGroup Object Real

Say you want to make further edits on an DupliGroup instance:

Simply select your DupliGroup and press `Ctrl-Shift-A` to convert the DupliGroup into regular objects that can be transformed and animated normally.

Note: Note that if the DupliGroup was linked from an external file the Object Data (mesh, materials, textures, transforms) will also still be linked from the original group. However, the various object's parent-child relationships do not carry over.

Display Options

Display and View Panels

Display Panel

Only Render Displays only items that will be rendered.

This can be is useful to preview how animations look without being distracted by rigs, empties, lights & cameras.

Useful to enable for *OpenGL Render*.

Note: While the option displays the regular view-port without distracting elements, the objects displayed are **not** matching the final render output.

Options such as restrict-render, modifiers render option, dupli-parents and render layers are not taken into account.

Outline Selected If disabled, the pink outline around your selected objects in *Solid*, *Shaded*, *Textured* draw types will no longer be displayed.

All Object Origins If enabled, the center dot of objects will always be visible, even for non-selected ones (by default, unselected centers might be hidden by geometry in solid/shaded/textured shadings).

Relationship Lines Controls whether the dashed parenting, constraining, hooking, etc., lines are drawn.

Grid Floor If disabled, you have no grid in other views than the orthographic top/front/side ones.

X Axis, Y Axis, Z Axis Control which axes are shown in other views than the orthographic top/front/side ones.

Lines Controls the number of lines that make the grid in non-top/front/side orthographic views, in both directions.

Scale Control the scale of the grid floor

Subdivisions Controls the number of sub-lines that appear in each cell of the grid when you zoom in, so it is a setting specific to top/front/side orthographic views.

Toggle Quad View Toggles the four view 3D View. *Read more about arranging areas*

View Panel

The *View Properties* panel lets you set other settings regarding the 3D View. You can show it with the *View* → *View Properties...* menu entry.

Lens Control the focal length of the 3D View camera in millimeters, unlike a *rendering camera*

Lock to Object By entering the name of an object in the *Object* field, you lock your view to this object, i.e. it will always be at the center of the view (the only exception is the active camera view, `Numpad0`). If the locked object is an armature, you can further center the view on one of its bones by entering its name in the *Bone* field.

Lock to Cursor Lock the center of the view to the position of the 3D cursor.

Lock Camera to View When in camera view, use this option to move the camera in 3D space, while continuing to remain in camera view.

Clip Start and Clip End Adjust the minimum and maximum distances to be visible for the view-port.

Note: A large clipping range will allow you to see both near and far objects, but reduces the depth precision resulting in artifacts.

See *Troubleshooting Depth Buffer Glitches* for more information.

Local Camera Active camera used in this view.

Render Border Use a Render Border when not looking through a camera. Using `Ctrl-B` to draw a border region will automatically enable this option.

Shading

Shading Modes

Shading refers to the way objects are drawn and lit in the 3D View.

Rendered An accurate representation using the selected *Render Engine* and lit with the visible scene lights.

Material A fast approximation of the applied material.

Textured Shows meshes with an image applied using the mesh's active UV Map. For Cycles materials, the image is the last one selected in the *Node Editor*. For other render engine's, the UV Map's applied face texture will be shown.

Solid The default drawing mode using solid colored surfaces and simple lighting.

Wireframe Objects appear as a mesh of lines representing the edges of faces and surfaces.

Bounding Box Only shows rectangular boxes that outline an object's size and shape.

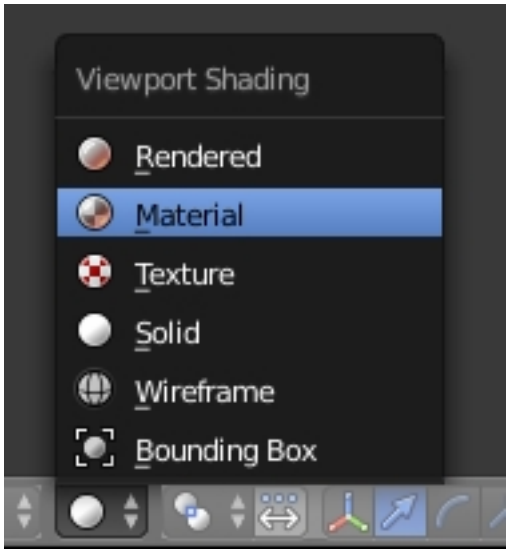


Fig. 2.190: The Viewport Shading menu.

Except for *Rendered*, these shading modes are not dependent on light sources in the scene. Instead they use a simple default lighting adjusted by the *Solid OpenGL Lights* controls on the *System* tab of the *User Preferences* editor.

The viewport shading controls the appearance of all objects in a scene, but this can be overridden for individual objects using the *Display panel* in their *Object Properties*.

Keyboard Shortcuts

Switches between *Wireframe* and *Solid* draw modes	`Z`
Switches between the current and *Rendered* draw modes.	`Shift-Z`
Switches between *Solid* and *Textured* draw modes.	`Alt-Z`

Shading Panel

The shading panel in the Properties Region provides additional control over the way objects in the 3D View appear.

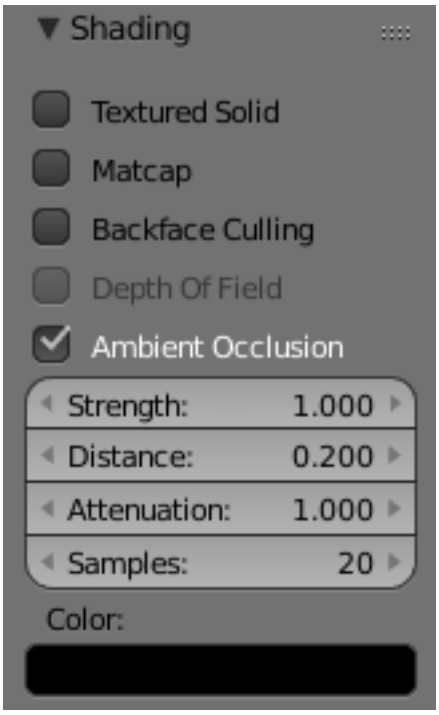


Fig. 2.191: 3D View Shading Panel.

Textured Solid Display assigned *face textures* in the *Solid* shading mode. (*Not* available in the Cycles Render Engine).

Matcap A selection of preset shader effects, (overriding regular materials) which can help visualize your models while editing or sculpting, without having to set up complex materials first.

Backface Culling Only show the front side of faces. Use this to find faces flipped the wrong way, especially when exporting to programs that use single sided drawing.

Depth of Field Simulates a camera's focal blur effect in the 3D View. This is only visible in a camera view. Control the effect using these options in the *Properties Tab* of the active camera: Focal Length, Sensor Size, Focus Object or Focus Distance, and Viewport F-stop.

Ambient Occlusion Improves the realism of the viewport image by simulating the darkening effect that occurs in crevices and corners. Typically such effects are rendered at higher quality, but this is a quick real-time preview which can help when modeling or sculpting.

These settings control the AO effect.

Strength A higher number makes the corners darker.

Distance How far out of the corners does the effect extend.

Attenuation How strongly the effect attenuates with distance. Increasing this makes far away surfaces contribute less to the effect. Use this to get rid of some banding artifacts.

Samples The number of samples used for the effect. Low numbers produce a grainy effect, but the actual number used is squared so use high numbers with caution.

Color Color of the effect, can be modified to give a different feel, from ambient lighting to dirt/rust.

Background Images

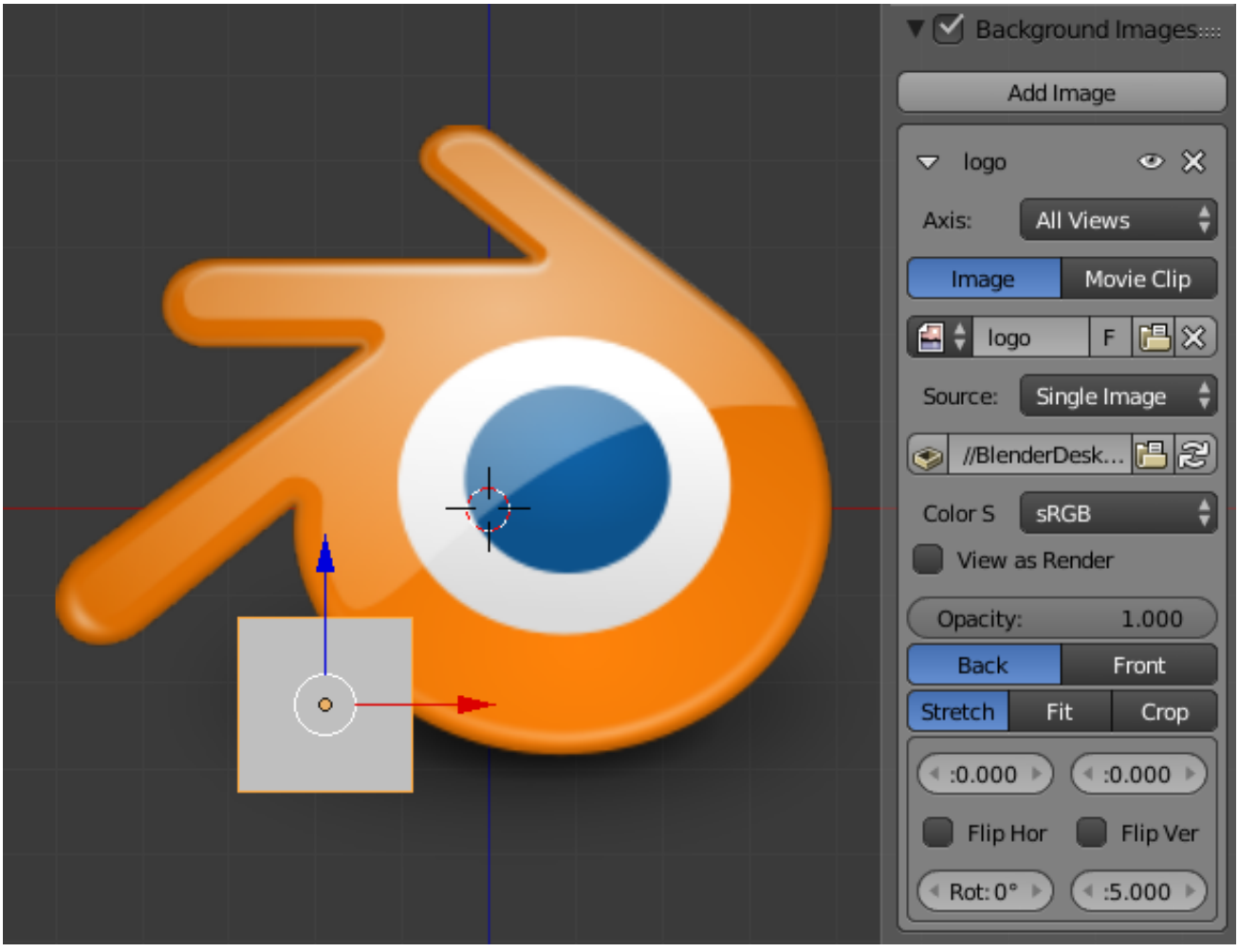
Reference

Editor: *3D View*
Panel: *Background Image*

A background picture in your 3D View is very helpful in many situations: modeling is obviously one, but it is also useful when painting (e.g. you can have reference pictures of faces when painting textures directly on your model...), or animation (when using a video as background), etc.

Note: Background images are only available for orthographic views.

Settings



Axis Choose which views the image is visible from. This is helpful when you have several reference images from different views (e.g. top, front and side).

Data Source The source of the background image.

Image Use an external image, image sequence, video file or generated texture.

Movie Clip Use one of the Movie Clip data-blocks.

Opacity Controls the transparency of the background image.

Front/Back Choose whether the image is shown behind all objects, or in front of everything.

Stretch/Fit/Crop Controls how the image is placed in the camera view.

Stretch Forces the image dimensions to match the camera bounds (may alter the aspect ratio).

Fit Scales the image down to fit inside the camera view without altering the aspect ratio.

Crop Scales the image up so that it fills the entire camera view, but without altering the aspect ratio (some of the image will be cropped)

X/Y Position the background image using these offsets.

In orthographic views, this is measured in the normal scene units. In the camera view, this is measured relative to the camera bounds (0.1 will offset it by 10% of the view width/height)

Flip Horizontally Swap the image around, such that the left side is now on the right, and the right now on the left.

Flip Vertically Swap the image around, such that the top side is now on the bottom, and the bottom now on the top.

Rotation Rotate the image around its center.

Size Scale the image up or down from its center.

2.2.2 Animation

Timeline Editor

The *Timeline* editor, identified by a clock icon, is shown by default at the bottom of the screen.

Fig. 2.192: The Timeline.

The *Timeline* is not much of an editor, but more of an information and control.

Here you can have an overview of the animation part of your scene. What is the current time frame, either in frames or in seconds, where are the keyframes of the active object, the start and end frames of your animation, markers, etc...

The *Timeline* has *Player Controls*, to play, pause the animation, and to skip though parts of the scene.

It also has some tools for *Keyframes*, *Keying Sets*, and *Markers*.

Main View

The main *Timeline* region displays the animation frames over time.

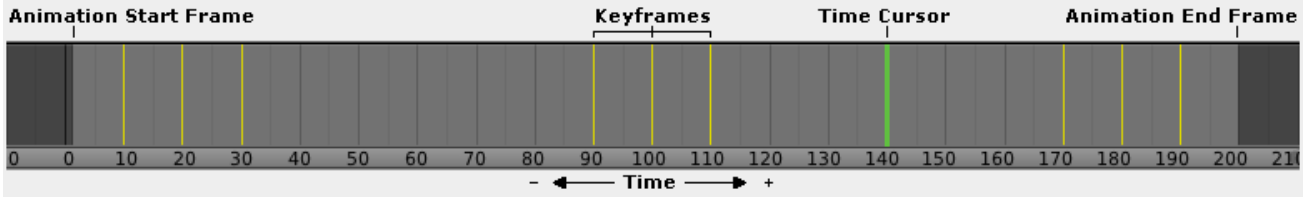


Fig. 2.193: Timeline Main Area.

Adjusting the View

The *Timeline* can be panned by holding `MMB`, then dragging the area left or right.

You can zoom the *Timeline* by using `Ctrl-MMB`, the mouse `Wheel`, or pressing `Minus` and `Plus` on the numpad.

Time Cursor

The *Time Cursor* is the green line, it is used to set and display the current time frame.

Fig. 2.194: Time Cursor.

The *Time Cursor* can be set or moved to a new position by pressing or holding `LMB` in the Timeline editor.

The current frame or second can be displayed on the *Time Cursor*, check the View menu for settings.

The *Time Cursor* can be moved in steps by pressing `Left` or `Right`, or in steps of 10 frames by pressing `Shift-Up` or `Shift-Down`.

Playback/Rendering Range

By default, the *Playback/Rendering Range* (Frame Start 1 to Frame End 200) is a lighter shade of gray. The start and end frame can be set to the *Time Cursor* by pressing `S` or `E`. The *Playback Range* can also be set by pressing `P` then drawing a box.

Keyframes

For the active and selected objects, keyframes are displayed as a yellow line. For *Armatures*, the object keyframes and the pose bones keyframes are drawn.

Only Selected Channels can be enabled. *Timeline* → *View* → *Only Selected Channels*. For *Armatures*, this will draw the object keyframes, and the keyframes for the active and selected pose bones.

Markers

Markers are the small triangles, with their name near them. Markers are usually used to identify key parts of the animation.

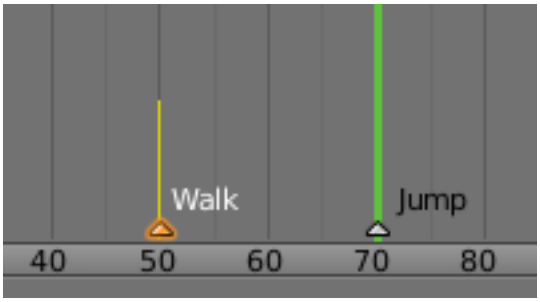


Fig. 2.195: Markers.

See the *Markers page* for more information.

Header

Menus

View Menu

The *View Menu* controls what you see, and what it looks like.

Toggle Full Screen Maximize or minimize the *Timeline* editor. `Ctrl-Up` or `Ctrl-Down`

Duplicate Area into New Window This creates a new window, and sets it to the *Timeline* editor.

Bind Camera to Markers This is used switch cameras during animation. It binds the active camera to the selected markers. First select a camera. Then select the marker(s). Then use the function. `Ctrl-B`

Cache

Show Cache Show all enabled types.

Softbody, Particles, Cloth, Smoke, Dynamic Paint, Rigid Body.

Fig. 2.196: Timeline Cache.

Only Selected Channels For *Armatures*, this will draw the object keyframes, and the keyframes for the active and selected pose bones.

Show Frame Number Indicator This will draw the current frame or seconds on the *Time Cursor*.

View All Maximize the *Timeline* area based on the Animation Range. `Home`

Show Seconds Show time in seconds for the *Timeline* and the *Time Cursor* based on the FPS. `Ctrl-T`

Marker Menu

See the *Markers page* for more information.

Frame Menu

Auto-Keyframing Mode This controls how the Auto Keyframe mode works. Only one mode can be used at a time.

Add & Replace Add or Replace existing keyframes.

Replace Only Replace existing keyframes.

Playback Menu

Audio Scrubbing If your animation has sound, this option plays bits of the sound wave while you move the time cursor with `LMB` or keyboard arrows.

Audio Muted Mute the sound from Sequence Editors.

AV-sync Play back and sync with audio clock, dropping frames if frame display is too slow. See *Synchronize Playback* for more info.

Frame Dropping Play back dropping frames if frame display is too slow. See *Synchronize Playback* for more info.

Clip Editors While playing, updates the Movie Clip Editor.

Node Editors While playing, updates the Node properties for the Node Editor.

Sequencer Editors While playing, updates the Video Sequence Editor.

Image Editors

Property Editors When the animation is playing, this will update the property values in the UI.

Animation Editors While playing, updates the Timeline, Dope Sheet, Graph Editor, Video Sequence Editor.

All 3D View Editors While playing, updates the 3D View and the Timeline.

Top-Left 3D Editor While playing, updates the Timeline, if Animation Editors and All 3D View Editors disabled.

Header Controls

The Timeline header controls.

Fig. 2.197: Timeline header controls.
1. Range Control, 2. Frame Control, 3. Player Control, 4. Synchronize Playback, 5. Keyframe Control.

Range Control

Use Preview Range This is an alternative range used to preview animations. This works for the UI playback, this will not work for rendering an animation.

Lock Time Cursor to Playback Range This limits the *Time Cursor* to the *Playback Range*.

Frame Control

Start Frame The start frame of the animation/playback range.

End Frame The end frame of the animation/playback range.

Current Frame The current frame of the animation/playback range. Also the position of the *Time Cursor*.

Player Control

These buttons are used to set, play, rewind, the *Time Cursor*.

Jump to start This sets the cursor to the start of frame range. `Shift-Ctrl-Down` or `Shift-Left`

Jump to previous keyframe This sets the cursor to the previous keyframe. `Down`

Rewind This plays the animation sequence in reverse. `Shift-Alt-A` When playing the play buttons switch to a pause button.

Play This plays the animation sequence. `Alt-A` When playing the play buttons switch to a pause button.

Jump to next keyframe This sets the cursor to the next keyframe. `Up`

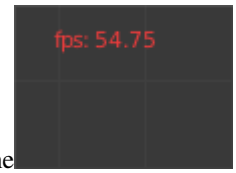
Fig. 2.198: Player Controls.

Jump to end This sets the cursor to the end of frame range. `Shift-Ctrl-Up` or `Shift-Right`

Pause This stops the animation. `Alt-A`

Synchronize Playback

When you play an animation, the FPS is displayed at the top left of the 3D View. If the scene is detailed and playback is slower than the set *Frame Rate* (see *Dimensions panel*, these options are used to synchronize the playback.



None

Fig. 2.199: 3D View Red FPS. 60:54.75

No Sync Do not sync, play every frame.

Frame Dropping Drop frames if playback is too slow. This enables *Frame Dropping* from the *Playback Menu*.

AV-sync Sync to audio clock, dropping frames if playback is slow. This enables *AV-sync* and *Frame Dropping* from the *Playback Menu*.

Keyframe Control

Auto Keyframe

The "Record" red-dot button enables something called *Auto Keyframe* : It will add and/or replace existing keyframes for the active object when you transform it in the 3D View.

For example, when enabled, first set the *Time Cursor* to the desired frame, then move an object in the 3D View, or set a new value for a property in the UI.

When you set a new value for the properties, Blender will add keyframes on the current frame for the transform properties.

Auto Keying Set When enabled *Auto Keyframe* will insert new keyframes for the properties in the active *Keying Set*.

Layered Adds a new NLA Track and strip for every loop/pass made over the animation to allow non-distructive tweaking.

Note: Note that *Auto Keyframe* only works for transform properties (objects and bones), in the 3D Views (i.e. you can't use it e.g. to animate the colors of a material in the Properties editor...).

Keyframe Type See *Keyframe Types*.

Active Keying Set

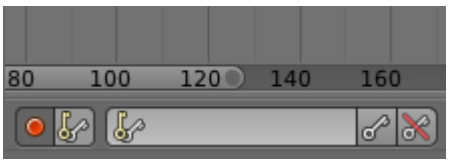


Fig. 2.200: Timeline Auto Keyframe.

Keying Sets are a set of keyframe channels in one.

They are made so the user can record multiple properties at the same time.

With a keying set selected, when you insert a keyframe, Blender will add keyframes for the properties in the active *Keying Set*.

There are some built in keying sets, *LocRotScale*, and also custom keying sets.

Custom keying sets can be defined in the panels *Properties* → *Scene* → *Keying Sets* + *Active Keying Set*.

Insert Keyframes Insert keyframes on the current frame for the properties in the active *Keying Set*.

Delete Keyframes Delete keyframes on the current frame for the properties in the active *Keying Set*.

Graph Editor

Introduction

The graph editor is the main animation editor. It allows you to modify the animation for any properties using *F-Curves*.

The graph editor has two modes, *F-Curve* for *Actions*, and *Drivers* for *Drivers*. Both are very similar in function.

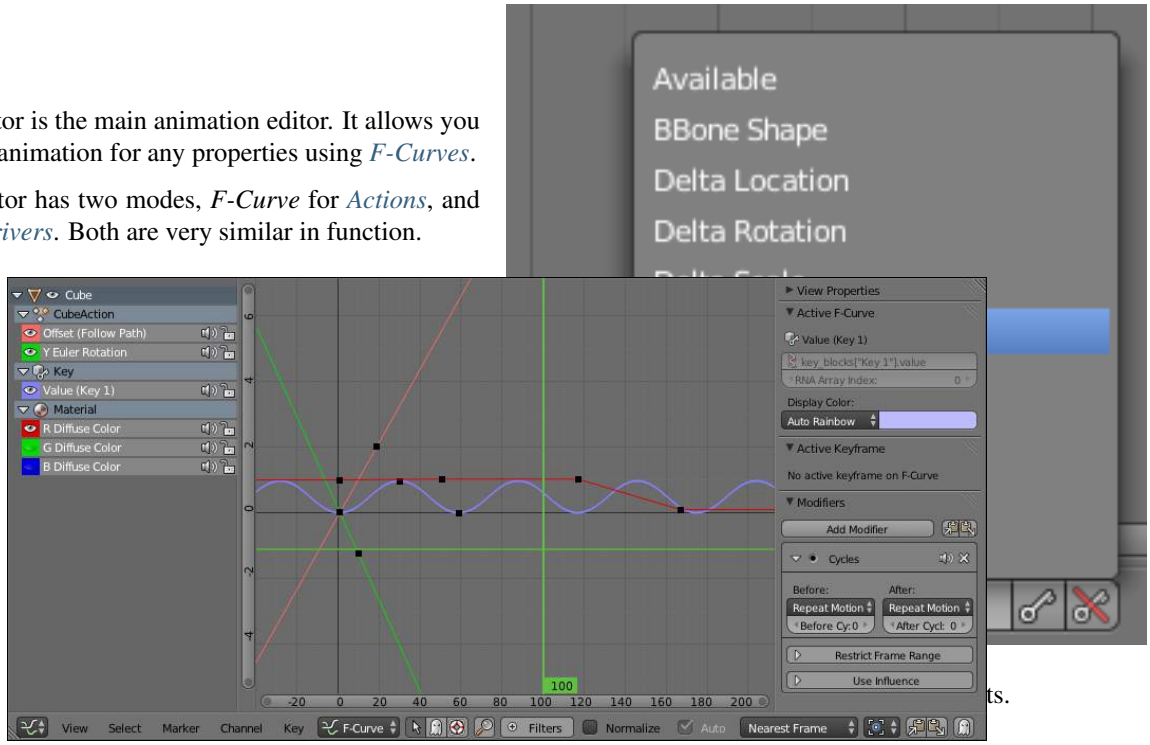


Fig. 2.202: The Graph Editor.

Curve View

Here you can see and edit the curves and keyframes.

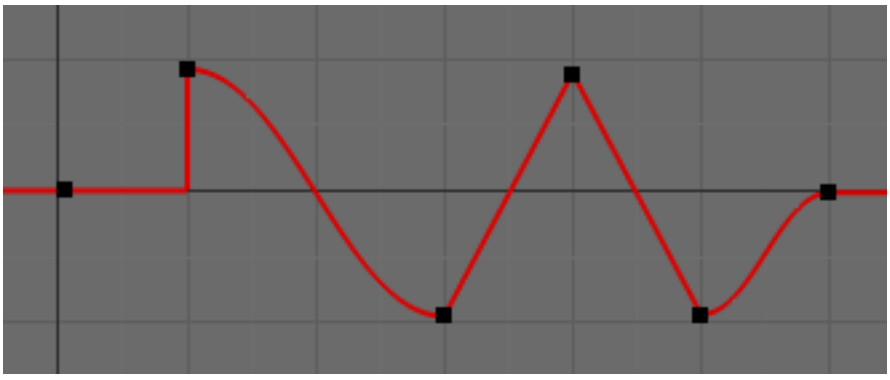


Fig. 2.203: A curve with different types of interpolation.

See *F-Curves* for more info.

Navigation

As with most editors, you can:

Pan Pan the view vertically (values) or horizontally (time) with click and drag (MMB).

Zoom Zoom in and out with the mouse wheel (Wheel).

Scale View Scale the view vertically or horizontally (Ctrl−MMB).

These are some other useful tools.

View All Reset viewable area to show all keyframes (Home).

View Selected Reset viewable area to show selected keyframes (NumpadPeriod).

2D Cursor

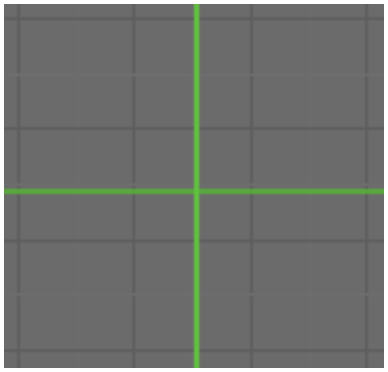


Fig. 2.204: Graph Editor 2D Cursor.

The current frame is represented by a green vertical line called the *Time Cursor*.

As in the *Timeline*, you can change the current frame by pressing or holding LMB.

The green horizontal line is called the *Cursor*. This can be disabled via the *View Menu* or the *View Properties* panel.

The *Time Cursor* and the *Cursor* make the *2D Cursor*. The *2D Cursor* is mostly used for editing tools.

View Axes

For *Actions* the X-axis represents time, the Y-axis represents the value to set the property.

For *Drivers* the X-axis represents the *Driver Value*, the Y-axis represents the value to set the property.

Depending on the selected curves, the values have different meaning: For example rotation properties are shown in degrees, location properties are shown in Blender Units. Note that *Drivers* use radians for rotation properties.

Markers

Like with most animation editors, markers are shown at the bottom of the editor.

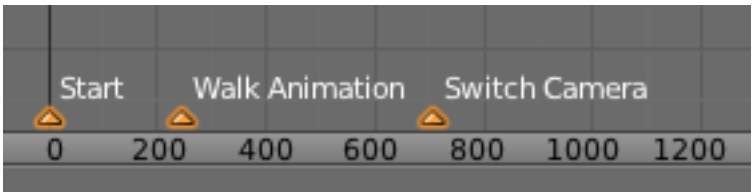


Fig. 2.205: Graph Editor Markers.

Markers can be modified in the *Graph Editor* though it's usually best to use the *Timeline*.

See *Markers* for more info.

Header

Here you will find:

- The menus.
- Graph Editor mode.
- View controls.
- Curve controls.

Header Controls

Fig. 2.206: Graph Mode.

Mode F-Curve for *Actions*, and Drivers for *Drivers*.

Fig. 2.207: View Controls.

View controls

Show Only Selected Only include curves related to the selected objects and data.

Show Hidden Include curves from objects/bones that are not visible.

Show Only Errors Only include curves that are disabled or have errors.

Search Filter Only include curves with keywords contained in the search field.

Type Filter Filter curves by property type.

Normalize Normalize curves so the maximum or minimum point equals 1.0 or -1.0.

Auto Automatically recalculate curve normalization on every curve edit.

Fig. 2.208: Curve Controls.

Curve controls

Auto Snap Auto snap the keyframes for transformations.

- *No Auto-Snap*
- *Time Step*
- *Nearest Frame*
- *Nearest Marker*

Pivot Point Pivot point for rotation.

Bounding Box Center Center of the selected keyframes.

2D Cursor Center of the *2D Cursor*. *Time Cursor + Cursor*.

Individual Centers Rotate the selected keyframe *Bézier* handles.

Copy Keyframes Copy the selected keyframes to memory (`Ctrl-C`).

Paste Keyframes Paste keyframes from memory to the current frame for selected curves (`Ctrl-V`).

Create Snapshot Creates a picture with the current shape of the curves.

Channels Region

The channels region is used to select and manage the curves for the graph editor.

Hide curve Represented by the eye icon.

Deactivate/Mute curve Represented by the speaker icon.

Lock curve from editing Represented by the padlock icon.

Channel Editing

- Select channel: `LMB`
- Multi Select/Deselect: `Shift-LMB`

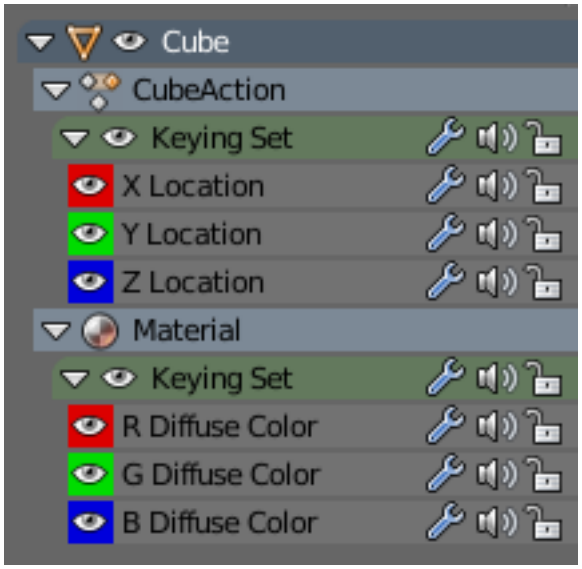


Fig. 2.209: Channels Region.

- Toggle Select All: `A`
- Border Select: (`LMB` drag) or `B` (`LMB` drag)
- Border Deselect: (`Shift-LMB` drag) or `B` (`Shift-LMB` drag)
- Delete selected: `X` or `Delete`
- Lock selected: `Tab`
- Make only selected visible: `V`
- Enable Mute Lock selected: `Shift-Ctrl-W`
- Disable Mute Lock selected: `Alt-W`
- Toggle Mute Lock selected: `Shift-W`

Properties Region

The panels in the *Properties Region*.

View Properties Panel

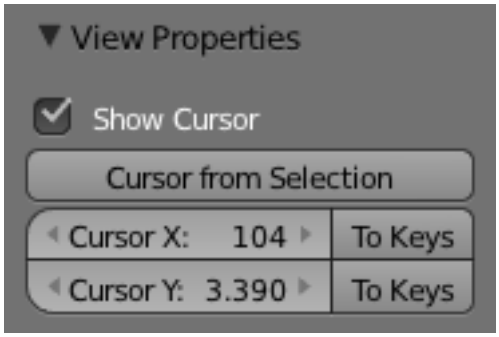


Fig. 2.210: View Properties Panel.

Show Cursor Show the vertical *Cursor*.

Cursor from Selection Set the *2D cursor* to the center of the selected keyframes.

Cursor X *Time Cursor* X position.

To Keys Snap selected keyframes to the *Time Cursor*.

Cursor Y Vertical *Cursor* Y position.

To Keys Snap selected keyframes to the *Cursor*.

Active F-Curve Panel

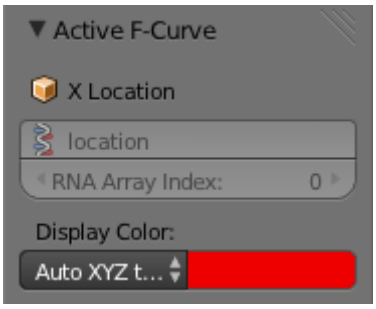


Fig. 2.211: Active F-Curve Panel.

This panel displays properties for the active *F-Curve*.

Channel Name *ID Type* + Channel name (X Location).

RNA Path *RNA Path* to property + Array index.

Color Mode *Color Mode* for the active *F-Curve*.

Auto Rainbow Increment the *HUE* of the *F-Curve* color based on the channel index.

Auto XYZ to RGB For property sets like location xyz, automatically set the set of colors to red, green, blue.

User Defined Define a custom color for the active *F-Curve*.

Active Keyframe Panel

Interpolation Set the forward interpolation for the active keyframe.

Constant Keep the same value till the next keyframe.

Linear The difference between the next keyframe.

Bézier Bézier interpolation to the next keyframe.

Key

Frame Set the frame for the active keyframe.

Value Set the value for the active keyframe.

Left Handle Set the position of the left interpolation handle for the active keyframe.

Right Handle Set the position of the right interpolation handle for the active keyframe.

Fig. 2.212: Active Keyframe Panel.

Drivers Panel

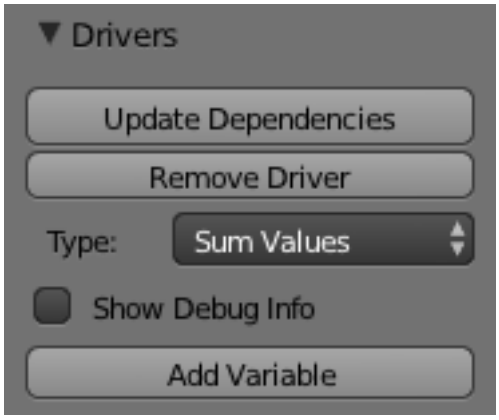


Fig. 2.213: Drivers Panel.

See *Drivers Panel* for more info.

Modifiers Panel

See *F-Modifiers* for more info.

See also:

- *Graph Editor - F-Curves*
- *Graph Editor - F-Modifiers*
- *Actions*
- *Drivers*

Fig. 2.214: Modifiers Panel.

F-Curves

Introduction

After animating some property in Blender using keyframes you can edit their corresponding curves. When something is "animated," it changes over time. This curve in shown as something called an F-Curve. Basically what an F-Curve does is an interpolation between two animated properties. In Blender, animating an object means changing one of its properties, such as the object's location, or its scale.

As mentioned, Blender's fundamental unit of time is the "frame", which usually lasts just a fraction of a second, depending on the *frame rate* of the scene. As animation is composed of incremental changes spanning multiple frames, usually these properties are **not** manually modified *frame by frame*, because:

- It would take ages!
- It would be very difficult to get smooth variations of the property (unless you compute mathematical functions and type a precise value for each frame, which would be crazy).

This is why nearly all direct animation is done using *interpolation*.

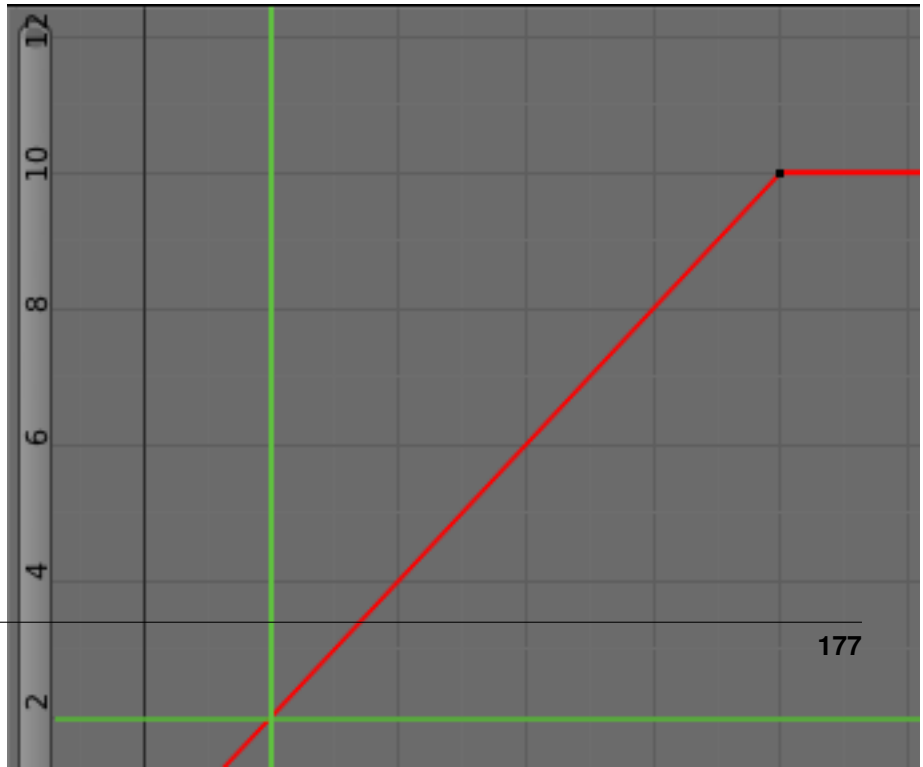
The idea is simple: you define a few Keyframes, which are multiple frames apart. Between these keyframes, the properties' values are computed (interpolated) by Blender and filled in. Thus, the animators' workload is significantly reduced.

For example, if you have:

- A control point of value 0 at frame 0,
- another one of value 10 at frame 25,
- and you use linear interpolation,

then, at frame 5 we get a value of 2.

The same goes for all intermediate frames: with just two points, you get a smooth growth from (0 to 10) along the 25 frames. Obviously, if you would like the frame 15 to have a value of 9, you would



have to add another control point (or keyframe)...

Settings

F-Curves have three additional properties, which control the interpolation between points, extension behavior, and the type of handles.

Interpolation Mode

Reference

Menu:
Curve → *Interpolation Mode*
Hotkey: `T`

You have three choices:

Constant There is no interpolation at all. The curve holds the value of its last keyframe, giving a discrete (stairway) "curve". Usually only used during the initial "blocking" stage in pose-to-pose animation workflows.

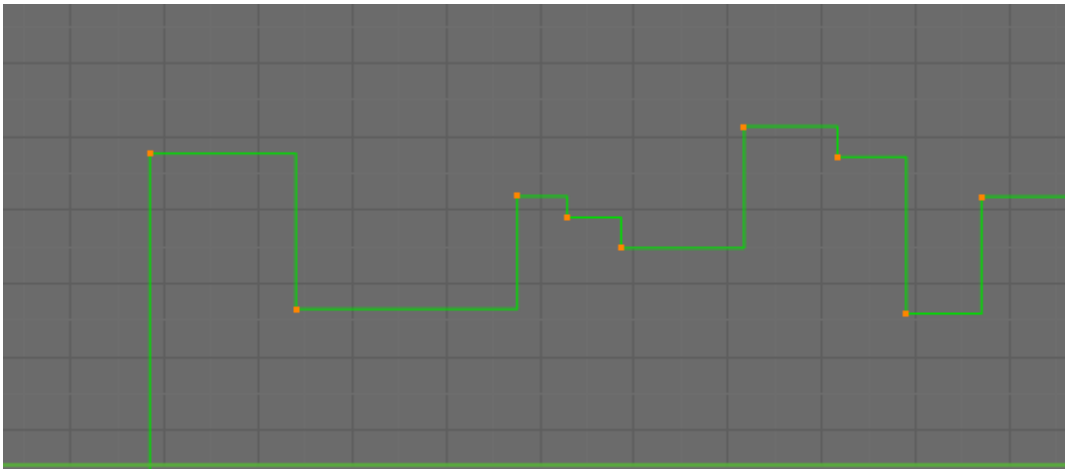


Fig. 2.216: Constant.

Linear This simple interpolation creates a straight segment between each neighbor keyframes, giving a broken line. It can be useful when

using only two keyframes and the *Extrapolation* extend mode, to easily get an infinite straight line (i.e. a linear curve).

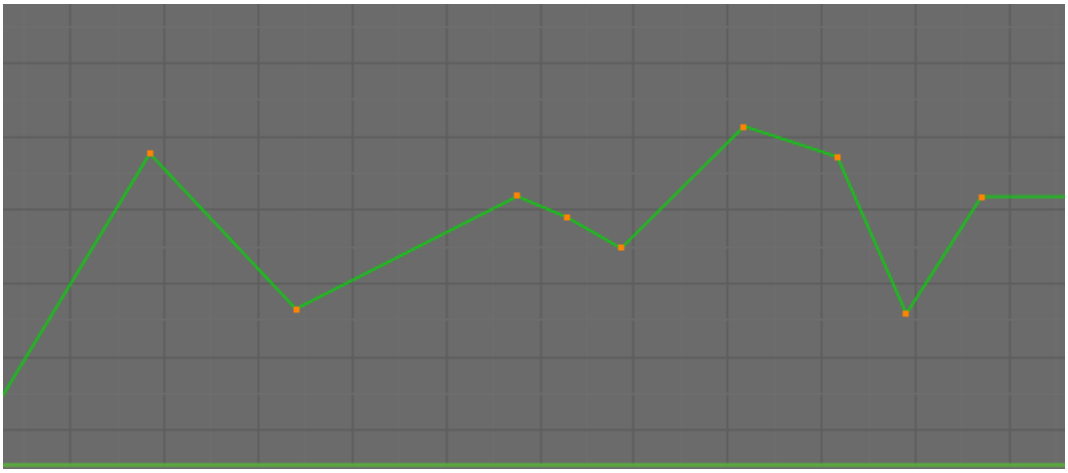


Fig. 2.217: Linear.

Bézier The more powerful and useful interpolation, and the default one. It gives nicely smoothed curves, i.e. smooth animations!

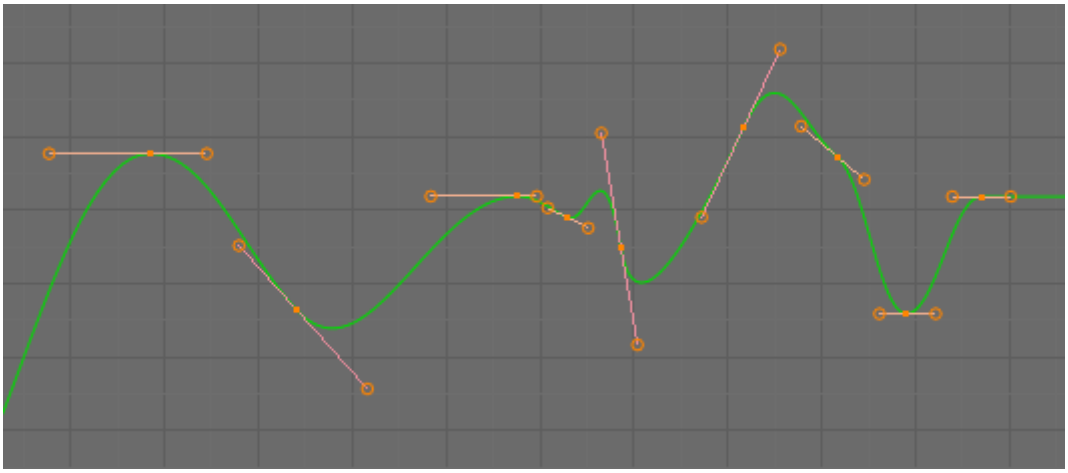


Fig. 2.218: Bézier.

Remember that some F-Curves can only take discrete values, in which case they are always shown as if constant interpolated, whatever option you chose.

Extrapolation

Reference

Menu: *Channel* → *Extrapolation Mode*

Hotkey: `Shift-E`

Extrapolation defines the behavior of a curve before the first and after the last keyframes.

There are two basic extrapolation modes:

Constant The default one, curves before their first keyframe and after their last one have a constant value (the one of these first and last keyframes).

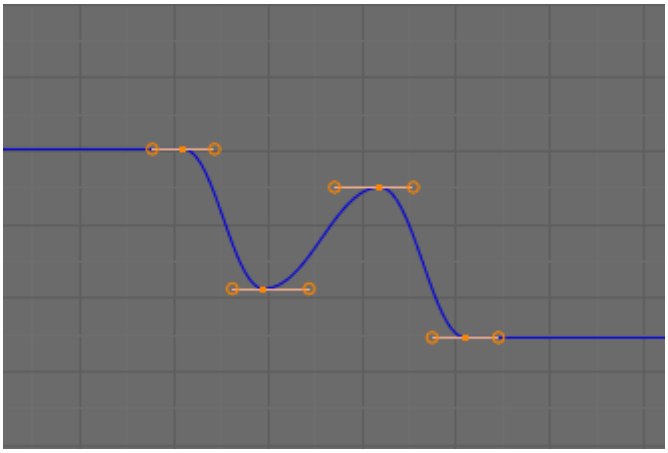


Fig. 2.219: Constant extrapolation.

Linear Curves ends are straight lines (linear), as defined by their first two keyframes (respectively their last two keyframes).

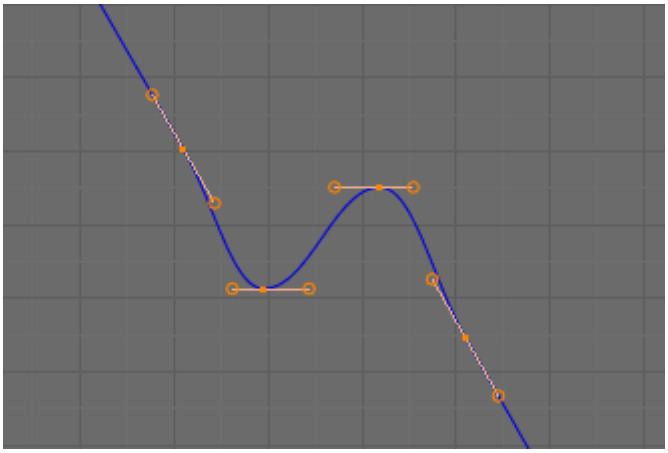


Fig. 2.220: Linear extrapolation.

Additional extrapolation tools (e.g. the "Cycles" F-Modifier) are located in the *F-Curve Modifiers*

Handle Types

There is another curve option quite useful for Bézier-interpolated curves. You can set the type of handle to use for the curve points `V`

Automatic Keyframes are automatically interpolated.

Vector Creates linear interpolation between keyframes. The linear segments remain if keyframe centers are moved. If handles are moved, the handle becomes Free.

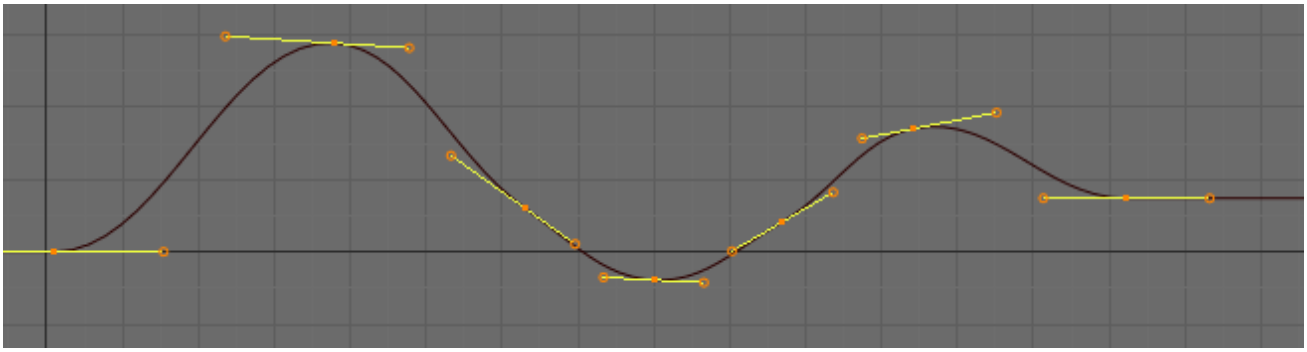


Fig. 2.221: Auto handles.

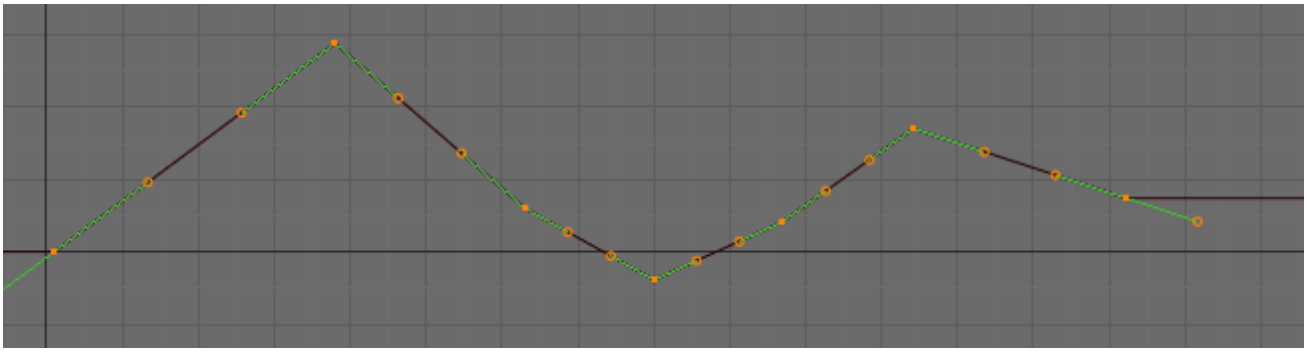


Fig. 2.222: Vector handles.

Aligned Handle maintain rotation when moved, and curve tangent is maintained.

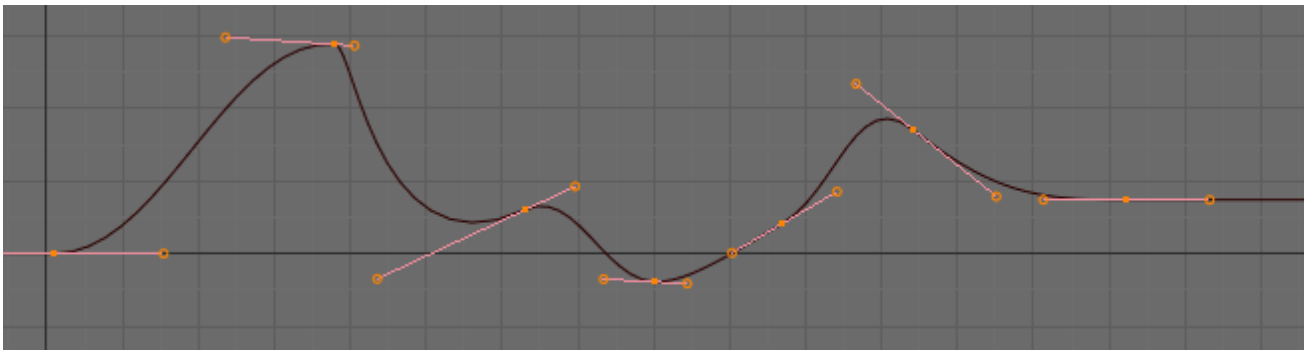


Fig. 2.223: Aligned handles.

Free Breaks handles tangents.

Auto Clamped Auto handles clamped to not overshoot.

Direction of Time

Although F-Curves are very similar to *Bézier Curves*, there are some important differences.

For obvious reasons, a property represented by a Curve cannot have more than **one** value at a given time, hence:

- when you move a control point ahead of a control point that was previously ahead of the point that you are moving, the two control points switch their order in the edited curve, to avoid that the curve goes back in time.
- for the above reason, it is impossible to have a closed F-Curve.

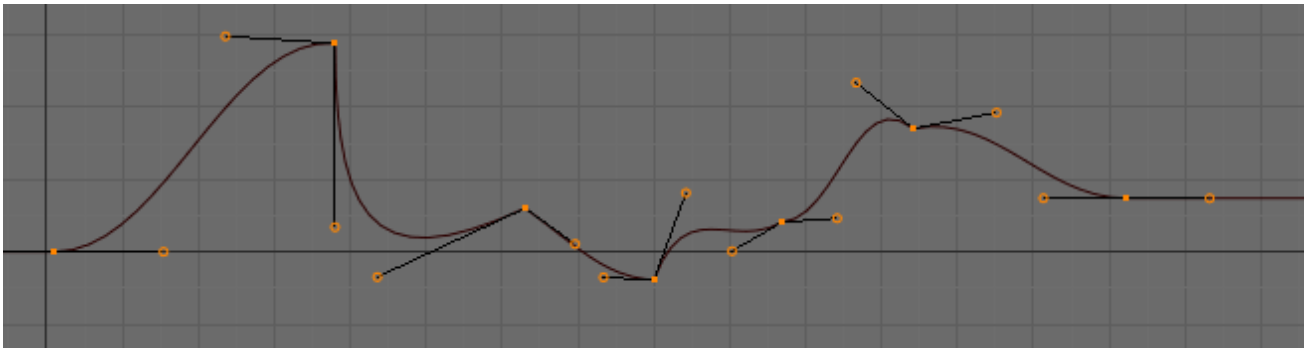


Fig. 2.224: Free handles.

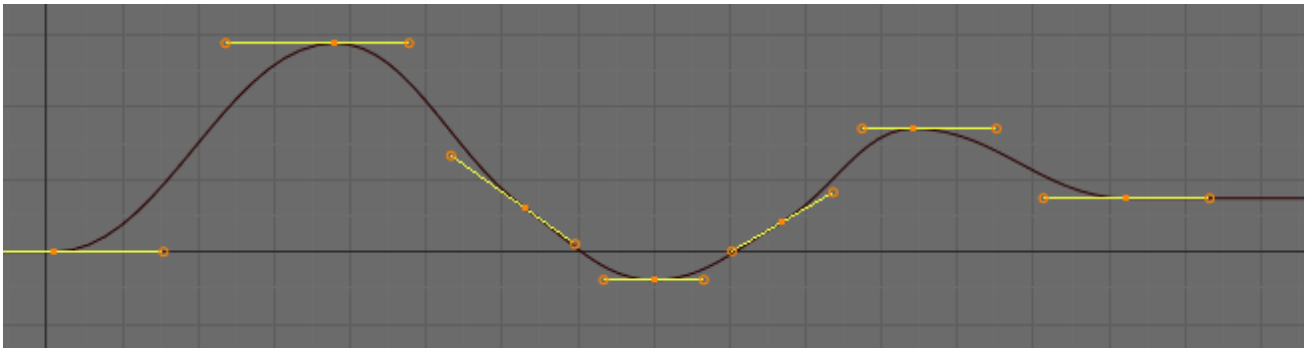
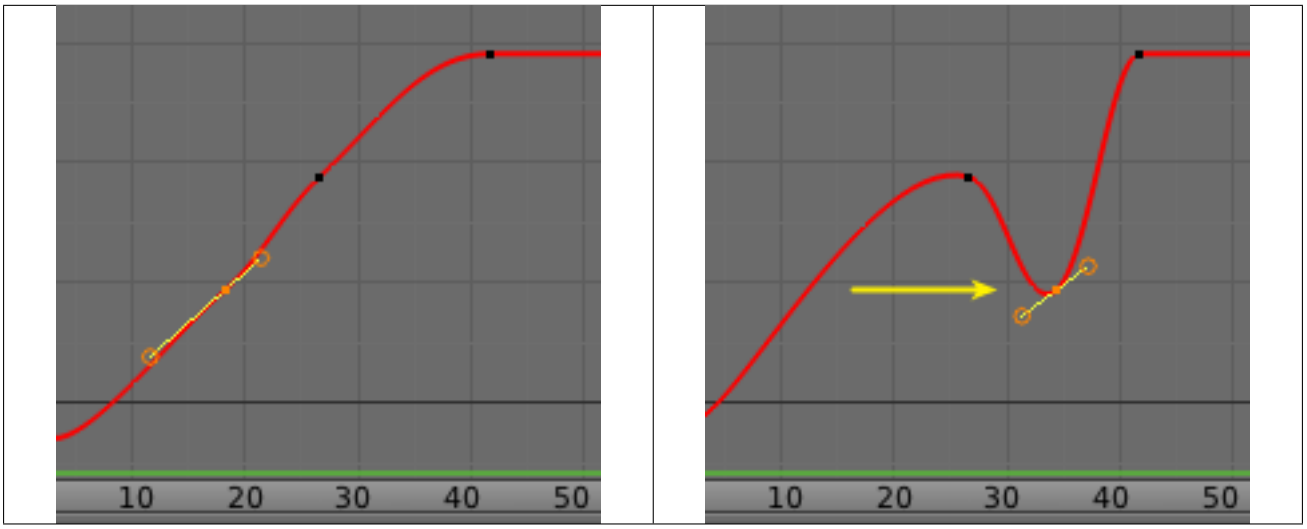


Fig. 2.225: Auto clamped handles.

Table 2.6: After moving the second keyframe.



Editing

By default, when new channels are added, the *Graph Editor* sets them to *Edit Mode*. Selected channels can be locked by pressing `Tab`.

Many of the hotkeys are the same as the viewport ones, for example:

- `G` to grab
- `R` to rotate

- S to scale
- B for border select/deselect

And of course you can lock the transformation along the X (time frame) or Y (value) axises by pressing X or Y during transformation.

For precise control of the keyframe position and value, you can set values in the *Active Keyframe* of the Properties Region.

Transform Snapping

When transforming keyframes with G, R, S, the transformation can be snapped to increments.

Snap Transformation to 1.0 Ctrl

Divide Transformation by 10.0 Shift

Keyframes can be snapped to different properties by using the *Snap Keys* tool.

Snap Keys Shift-S

Current Frame Snap the selected keyframes to the *Time Cursor*.

Cursor Value Snap the selected keyframes to the *Cursor*.

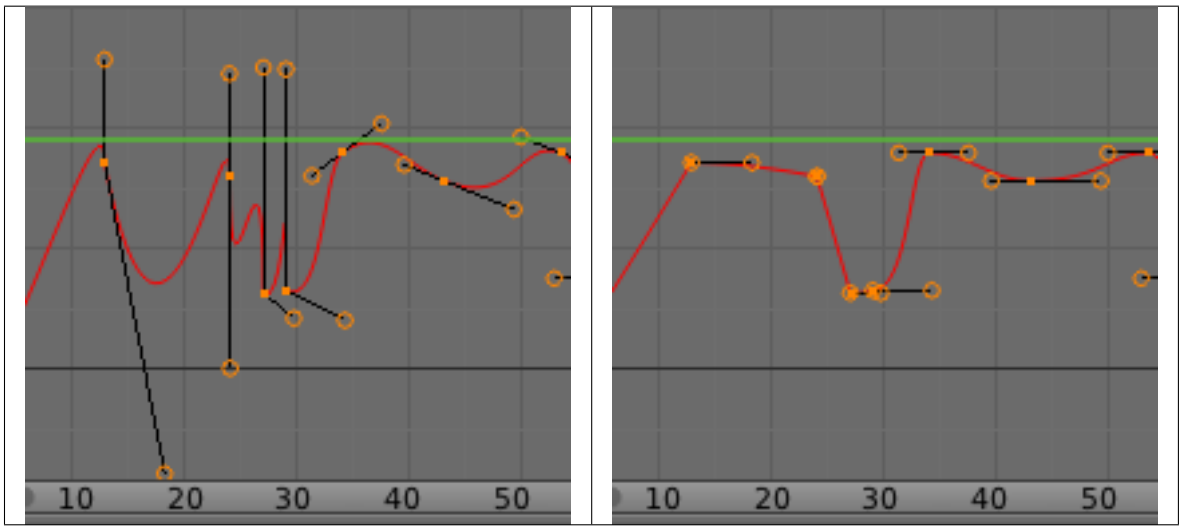
Nearest Frame Snap the selected keyframes to their nearest frame individually.

Nearest Second Snap the selected keyframes to their nearest second individually, based on the *FPS* of the scene.

Nearest Marker Snap the selected keyframes to their nearest marker individually.

Flatten Handles Flatten the *Bézier* handles for the selected keyframes.

Table 2.7: After Flatten Handles.



Mirror

Selected keyframes can be mirrored over different properties using the *Mirror Keys* tool.

Mirror Keys Shift-M

By Times Over Current Frame Mirror horizontally over the *Time Cursor*.

By Values over Cursor Value Mirror vertically over the *Cursor*.

By Times over Time 0 Mirror horizontally over frame 0.

By Values over Value 0 Mirror vertically over value 0.

By Times over First Selected Marker Mirror horizontally the over the first selected *Marker*.

Clean Keyframes

Clean Keyframes resets the keyframe tangents to their auto-clamped shape, if they have been modified. *Clean Keyframes* O

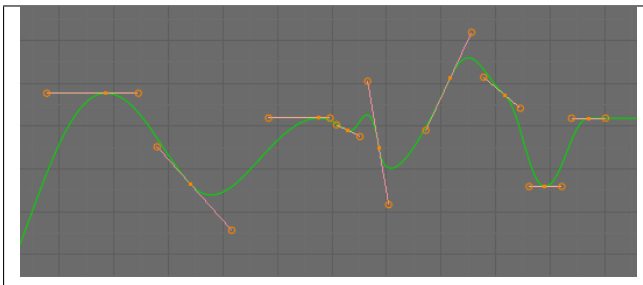


Fig. 2.230: F-Curve before cleaning.

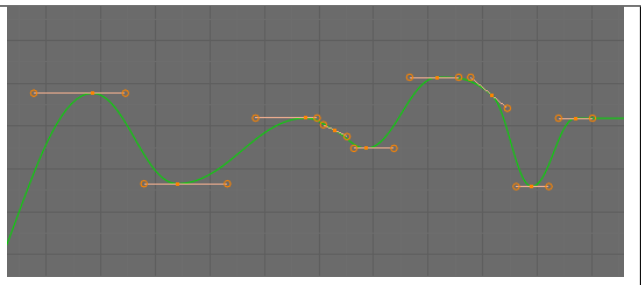


Fig. 2.231: F-Curve after cleaning.

Smoothing

Reference

Menu: *Key* → *Smooth Keys*
Hotkey: Alt-O

There is also an option to smooth the selected curves , but beware: its algorithm seems to be to divide by two the distance between each keyframe and the average linear value of the curve, without any setting, which gives quite a strong smoothing! Note that the first and last keys seem to be never modified by this tool.

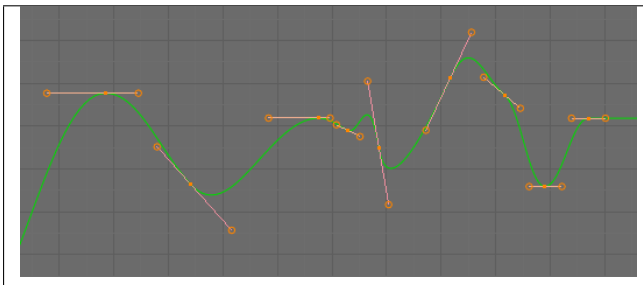


Fig. 2.232: F-Curve before smoothing.

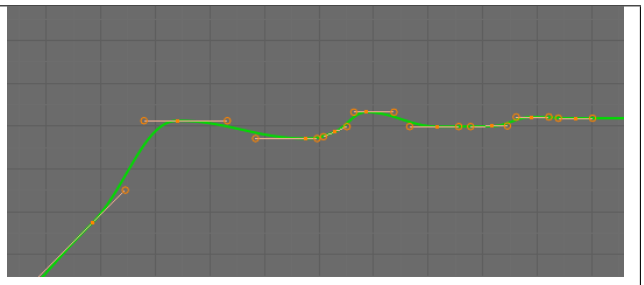


Fig. 2.233: F-Curve after smoothing.

Sampling and Baking Keyframes

Sample Keyframes **Shift-O** Sampling a set a keyframes replaces interpolated values with a new keyframe for each frame.

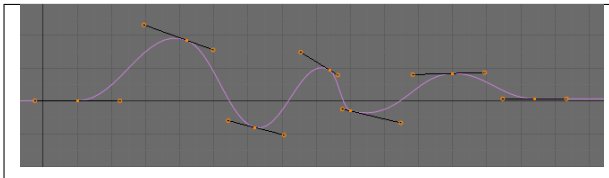


Fig. 2.234: F-Curve before sampling.

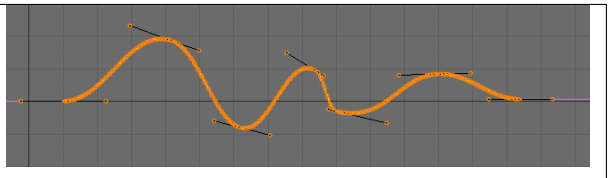


Fig. 2.235: F-Curve after sampling.

Bake Curves `Alt-C` Baking a curve replaces it with a set of sampled points, and removes the ability to edit the curve.

Bake Sound to F-Curves

The *Bake Sound to F-Curves* tool takes and sound file and uses its sound wave to create the animation data.

Lowest frequency Cutoff frequency of a high-pass filter that is applied to the audio data.

Highest frequency Cutoff frequency of a low-pass filter that is applied to the audio data.

Attack time Value for the hull curve calcuclation that tells how fast the hull curve can rise. The lower the value the steeper it can rise.

Release time Value for the hull curve calculation that tells how fast the hull curve can fall. The lower the value the steeper it can fall.

Threshold Minimum amplitude value needed to influence the hull curve.

Accumulate Only the positive differences of the hull curve amplitudes are summarized to produce the output.

Additive The amplitudes of the hull curve are summarized. If *Accumulate* is enabled, both positive and negative differences are accumulated.

Square Gives the output as a square curve. Negative values always result in -1, and positive ones in 1.

Square Threshold All values lower than this threshold result in 0.

F-Curve Modifiers

F-Curve modifiers are similar to object modifiers, in that they add non-destructive effects, that can be adjusted at any time, and layered to create more complex effects.

Adding a Modifier

The F-Curve modifier panel is located in the Properties region. Select a curve by selecting one of its curve points, or by selecting the channel list. Click on the *Add Modifier* button and select a modifier.

To add spin to an object or group, select the object/group and add a keyframe to the axis of rotation (X, Y, or Z)

To add a modifier, go to *Properties* → *Add Modifier*.

Types of Modifiers

Generator

Generator creates a Factorized or Expanded Polynomial function. These are basic mathematical formulas that represent lines, parabolas, and other more complex curves, depending on the values used.

Additive This option causes the modifier to be added to the curve, instead of replacing it by default.

Poly Order Specify the order of the polynomial, or the highest power of `X` for this polynomial. (number of coefficients: 1).

Change the Coefficient values to change the shape of the curve.

See also:

The Wikipedia Page for more information on polynomials.

Built-in Function

These are additional formulas, each with the same options to control their shape. Consult mathematics reference for more detailed information on each function:

- Sine
- Cosine
- Tangent
- Square Root
- Natural Logarithm
- Normalized Sine (`sin(x)/x`)

Amplitude Adjusts the Y scaling.

Phase Multiplier Adjusts the X scaling.

Phase Offset Adjusts the X offset.

Value Offset Adjusts the Y offset.

Envelope

Allows you to adjust the overall shape of a curve with control points.

Reference Value Set the Y value the envelope is centered around.

Min Lower distance from Reference Value for `1:1` default influence.

Max Upper distance from Reference Value for `1:1` default influence.

Add Point Add a set of control points. They will be created at the current frame.

Fra Set the frame number for the control point.

Min Specifies the lower control point's position.

Max specifies the upper control point's position.

Cycles

Cycles allows you add cyclic motion to a curve that has two or more control points. The options can be set for before and after the curve.

Cycle Mode

Repeat Motion Repeats the curve data, while maintaining their values each cycle.

Repeat with Offset Repeats the curve data, but offsets the value of the first point to the value of the last point each cycle.

Repeat Mirrored Each cycle the curve data is flipped across the X-axis.

Before/After Cycles Set the number of times to cycle the data. A value of 0 cycles the data infinitely.

Noise

Modifies the curve with a noise formula. This is useful for creating subtle or extreme randomness to animated movements, like camera shake.

Blend Type

Replace Adds a -0.5 to 0.5 range noise function to the curve.

Add Adds a 0 to 1 range noise function to the curve.

Subtract Subtracts a 0 to 1 range noise function to the curve.

Multiply Multiplies a 0 to 1 range noise function to the curve.

Scale Adjust the overall size of the noise. Values further from 0 give less frequent noise.

Strength Adjusts the Y scaling of the noise function.

Phase Adjusts the random seed of the noise.

Depth Adjusts how detailed the noise function is.

Limits

Limit curve values to specified X and Y ranges.

Minimum/Maximum X Cuts a curve off at these frames ranges, and sets their minimum value at those points.

Minimum/Maximum Y Truncates the curve values to a range.

Stepped Interpolation

Gives the curve a stepped appearance by rounding values down within a certain range of frames.

Step Size Specify the number of frames to hold each frame.

Offset Reference number of frames before frames get held. Use to get hold for (1-3) vs (5-7) holding patterns.

Use Start Frame Restrict modifier to only act before its "end" frame.

Use End Frame Restrict modifier to only act after its "start" frame.

Dope Sheet

Introduction

Classical hand-drawn animators often made a chart, showing exactly when each drawing, sound and camera move would occur, and for how long. They nicknamed this the "dopesheet". While CG foundations dramatically differ from classical hand-drawn animation, Blender's *Dopesheet* inherits a similar directive. It gives the animator a "birds-eye-view" of every thing occurring within a scene.

Dope Sheet Modes

There are four basic views for the Dopesheet.These all view different contexts of animation:

DopeSheet The dopeSheet allow you to edit multiple actions at once.

Action Editor *Action Editor* is the default, and most useful one. It is here that you can define and control your actions.

Shape Key Editor *ShapeKey Editor* is dedicated to the *Shapekey* data-blocks. It uses/edits the same action data-block as the previous mode. It seems to be an old and useless thing, as the *Action Editor* mode handles *Shape* channels very well, and this mode adds nothing...

Grease Pencil *Grease Pencil* is dedicated to the *grease pencil tool's* keyframes- for each grease pencil layer, you have a strip along which you can grab its keys, and hence easily re-time your animated sketches. As it is just another way to see and edit the grease pencil data, this mode uses no data-block (and hence has nothing to do with actions).

Note: Note that you will have as much top-level grease pencil channels as you have sketched areas (3D Views, *UV/Image Editor*, etc.)

Fig. 2.236: The DopeSheet.

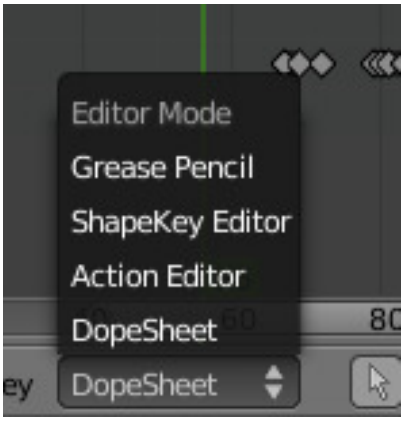


Fig. 2.237: Dope Sheet Modes.

Interface

The *Action Editor* interface is somewhat similar to the *Graph Editor* one, it is divided in three regions:

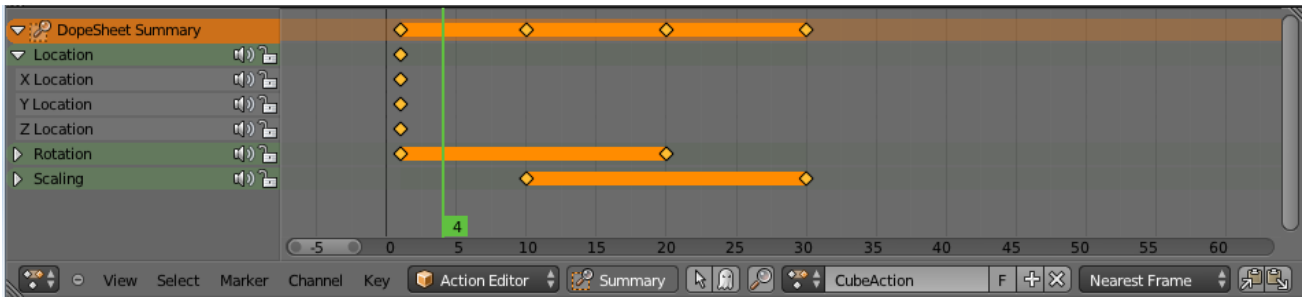


Fig. 2.238: The Action Editor, Action Editor mode, with an Object and Shape channels.

Header

Here you find the menus, a first block of controls related to the editor "mode", a second one concerning the action data-blocks, and a few other tools (like the copy/paste buttons, and snapping type).

View Menu

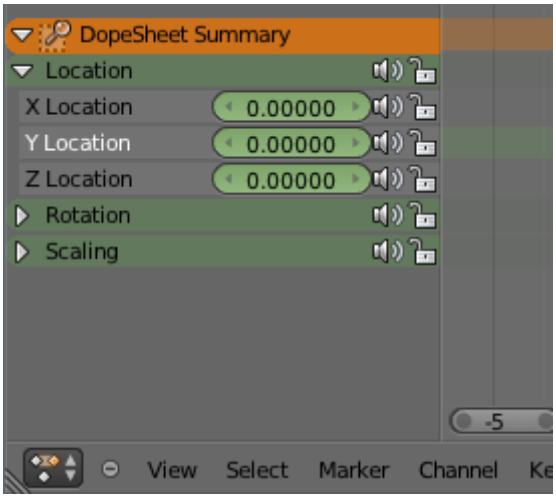


Fig. 2.239: the action editor showing sliders.

Realtime Updates When transforming keyframes, changes to the animation data are flushed to other views.

Show Frame Number Indicator Show frame number beside the current frame indicator line.

Show Sliders A toggle option that shows the value sliders for the channels. See the Fig. *The Action Editor, Action Editor mode, with an Object and Shape channels.*.

Use Group Colors Draw groups and channels with colors matching their corresponding groups.

AutoMerge Keyframes Automatically merge nearby keyframes.

Sync Markers Sync Markers with keyframe edits.

Show Seconds Whether to show the time in the X-axis as frames or as seconds.

Set Preview Range `P` Interactively define frame range used for playback. Allow you to define a temporary preview range to use for the `Alt-A` realtime playback (this is the same thing as the *Playback Range* option of the *Tmeline editor header*).

Clear Preview Range `Alt-P` Clears the preview range.

Auto-Set Preview Range Automatically sets the preview range to playback the whole action.

Marker Menu

See the *Markers page*.

Main Region

It contains the keyframes for all visible action channels. As with the other "time" editor, the X-axis represents time. The Y-axis has no mean in itself, unlike with the *Graph Editor*, it is just a sort of "stack" of action channels. Each one being shown as an horizontal colored strip (of a darker shade "during" the animated/keyed period). On these channel strips lay the keyframes, visualized as light-gray (unselected) or yellow (selected) diamonds. One of the key feature of this editor is that it allow you to visualize immediately which channel (i.e. F-Curve) is *really* affected. When the value of a given channel does not change at all between two neighboring keyframes, a gray (unselected) or yellow (selected) line is drawn between them.

"List-tree" Region

This part shows the action's channel "headers" and their hierarchy. Basically, there are:

- "Top-level" channels, which represent whole F-Curve data-blocks (so there is one for *Object* one, one for *Shape* one, etc.). They gather *all* keyframes defined in their underlying F-Curve data-block.
- "Mid-level" channels, which seem currently to have no use (there iss one per top-level channel, they are all named *F-Curves*, and have no option at all).
- "Low-level" channels, which represent individual F-Curve, with their own keyframes (fortunately, only keyed frames are shown!).

Each level can be expended/collapsed by the small arrow to the left of its "parent" channel. To the right of the channel's headers, there are some channel's setting controls:

Mute (Speaker icon) Will allow you to mute that channel (and all its "children" channels, if any!).

Lock (Lock icon) Will allow you to prevent this channel and its children to be edited.

> **Note:** This is also working inside the NLA, but that it does not prevent edition of the underlying F-Curve).

F-Modifier (Wrench icon) Disables the F-Modifiers.

A channel can be selected (text in white, strip in gray-blue color) or not (text in black, strip in pink-brown color.), use `LMB` clicks to toggle this state. You can access some channel's properties by clicking `Ctrl-LMB` on its header. Finally, you can have another column with value-sliders, allowing you to change the value of current keyframes, or to add new ones. These are obviously only available for low-level channels (i.e. individual F-Curve). See *View Menu* above for how to show these sliders.

Action Editor

The *Action Editor* enables you to see and edit the F-Curve data-blocks you defined as *Actions* in the *F-Curve Editor*. So it takes place somewhere in-between the low-level *F-Curves*, and the high-level *NLA editor*.

It gives you a slightly simplified view of the F-Curve data-blocks (somewhat similar to F-Curve drawn without handles). The editor can lists all Action data-blocks of an object at once.

Each Action data-block forms a top-level channel (see below). Note that an object can have several *Constraint* (one per animated constraint) and *Pose* (for armatures, one per animated bone) F-Curve data-blocks, and hence an action can have several of these channels.

Header

Action A *Data-block menu*.

Add When an action is created it is stored in a NLA Action Stash.

Channel Menu

Delete `X`

Deletes the whole channel from the current action (i.e. unlink the underlying F-Curve data-block from this action data-block).

> **Warning:** The `X` shortcut is area-dependent: if you use it in the left list part, it will delete the selected channels, whereas if you use it in the main area, it will delete the selected keyframes.

***Settings* → *Toogle/Enable/Disable a Setting*, `Shift-W` or `Ctrl-Shift-W` or `Alt-W`** Enable/disable a channel's setting (selected in the menu that pops-up) - currently, "lock" and/or "mute" only.

Toggle Channel Editability `Tab` Locks or unlocks a channel for editing

Extrapolation Mode Change the extrapolation between selected keyframes. More options are available in the Graph Editor.

Expand Channels, Collapse Channels `NumpadPlus`, `NumpadMinus` Expands or collapses selected channels.

Move... This allows you to move top-level channels up/down `Shift-PageUp`, `Shift-PageDown`, or directly to the top/bottom `Ctrl-Shift-PageUp`, `Ctrl-Shift-PageDown`.

Revive Disabled F-Curves Clears "disabled" tag from all F-Curves to get broken F-Curves working again.

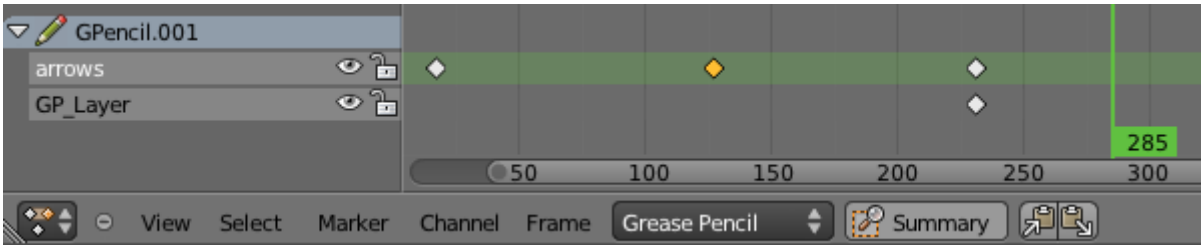
Grease Pencil

Adjusting Timing of Sketches

It is possible to set a *grease pencil* block to be loaded up in the *DopeSheet* for editing of the timings of the drawings. This is especially useful for animators blocking out shots, where the ability to re-time blocking is one of the main purposes of the whole exercise.

1. In a *Dope Sheet* editor, change the mode selector (found beside the menus) to *Grease Pencil* (by default, it should be set to *DopeSheet*).
2. At this point, the *DopeSheet* should now display a few "channels" with some "keyframes" on them. These "channels" are the layers, and the "keyframes" are the frames at which the layer has a sketch defined. They can be manipulated like any other data in the *DopeSheet* can be.

All the available Grease-Pencil blocks for the current screen layout will be shown. The Area/Grease-Pencil data-blocks are drawn as green channels, and are named with relevant info from the views. They are also labeled with the area (i.e. "window") index (which is currently not shown anywhere else though).



Copying Sketches

It is possible to copy sketches from a layer/layers to other layers in the *Action Editor*, using the "Copy"/"Paste" buttons in the header. This works in a similar way as the copy/paste tools for keyframes in the *Action Editor*.

Sketches can also be copied from one screen (or view) to another using these tools. It is important to keep in mind that keyframes will only be pasted into selected layers, so layers will need to be created for the destination areas too.

Shape Key

Todo.

Non-Linear Animation Editor

Introduction

The NLA editor can manipulate and repurpose actions, without the tedium of keyframe handling. Its often used to make broad, significant changes to a scene's animation, with relative ease. It can also re purpose, and "layer" actions, which make it easier to organize, and version-control your animation.

Usage

Tracks

Tracks are the layering system of the NLA. At its most basic level, it can help organize strips. But it also layers motion much like an image editor layers pixels – the bottom layer first, to the top, last.

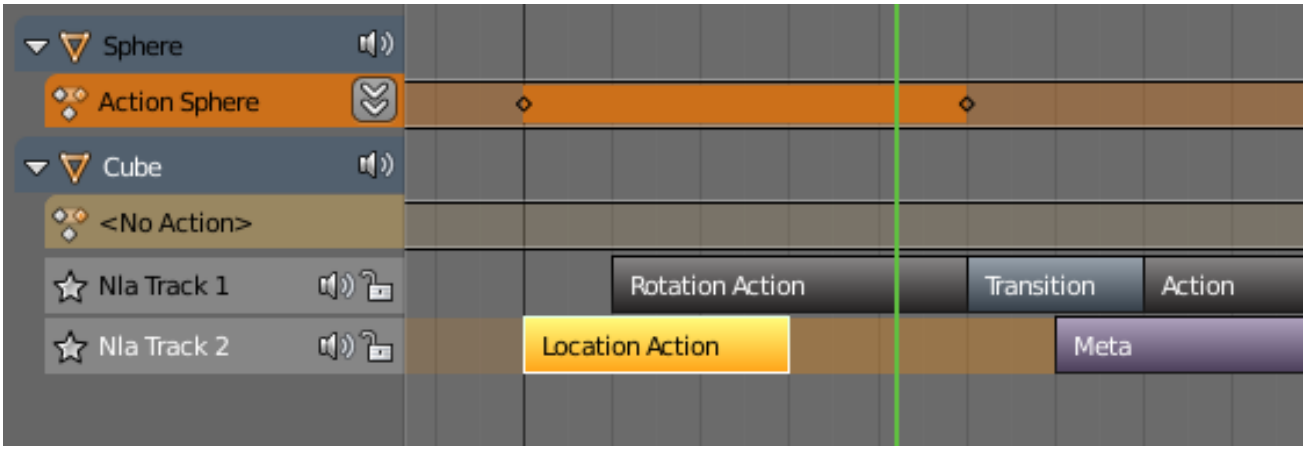


Fig. 2.240: NLA Tracks and Strips.

Solo (Star icon) Toggling *Solo Track* causes only the selected tracks effects to be visible when animating.

Mute (Speaker icon) Keeps the track from having an effect on the animation.

Lock (Lock icon) Prevents changes from being made to this layer.

Action Track

Push Down (double down arrow peak icon) Turns the action into a new action strip.

Push Down action button.

Pin (Pin icon) If you try moving the strip, while in *Tweak Mode*, you will notice that the keys will go along with it. On occasion, you will prefer the keys to remain on their original frames, regardless of where the strip is. To do so, hit the *unpin* icon, next to the strip.

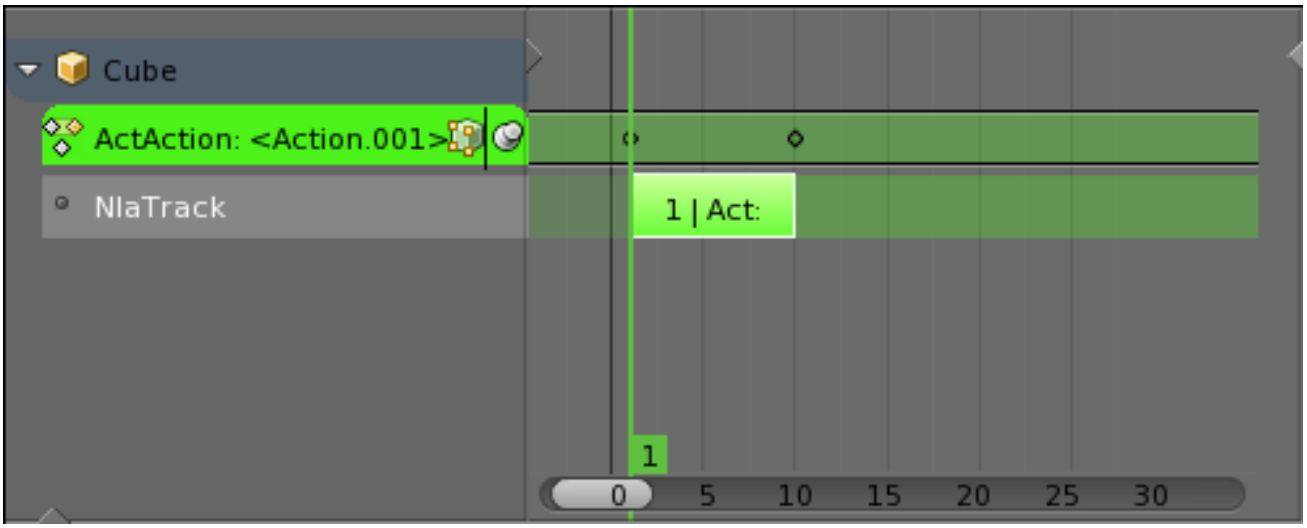


Fig. 2.241: Nla strip with pinned keys.

Fig. 2.242: Strip moved, notice the keys move with it.

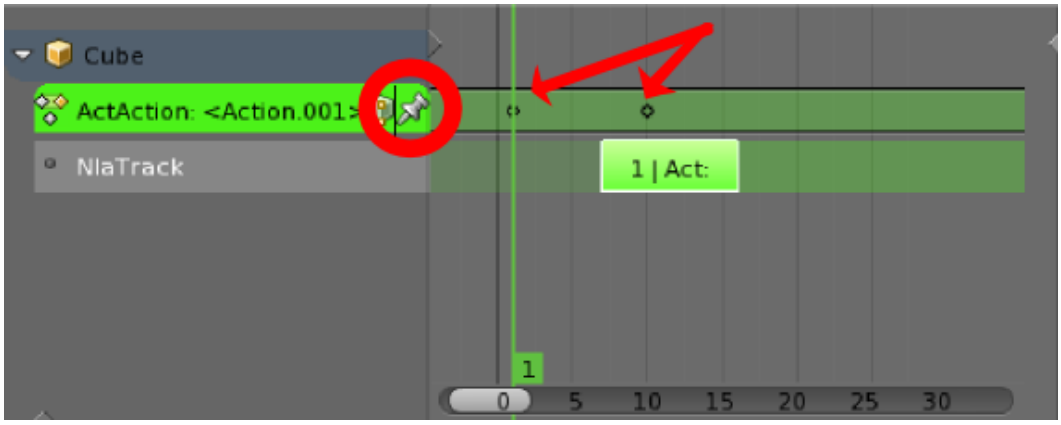


Fig. 2.243: The unpinned keys return to their original frames.

Strips

There are four kinds of strips: Action, Transition, Sound clip and Meta.

Action Strips

An Action Strip is a container of keyframe data of an action. Any action used by the NLA first must be turned into an Action strip. This is done so by clicking the Push Down action button see above. Alternatively, you can go to *Add* → *Action*.

Transition Strips

Transitions interpolate between Actions. They must be placed in between other strips. Select two or more strips on the same track, and go to: *Add* → *Transition*.

Fig. 2.244: Transition Strip.

Sound Clip Strips

Controls when a speaker plays a sound clip. *Add* → *Sound Clip*.

Meta Strips

Meta strips group strips together as a whole, so you can move them as one. If you find yourself moving a lot of strips together, you can group them into a Meta strip. A meta strip can be moved and duplicated like a normal strip.

Reference

Menu: *Add* → *Add Meta-Strips*

Hotkey: `Shift-G`

Fig. 2.245: Shift-select two or more strips..

Fig. 2.246: Combine them into a meta strip.

A meta strip still contains the underlying strips. You can ungroup a Meta strip.

Reference

Menu: *Add* → *Remove Meta-Strips*
Hotkey: `Alt-G`

Editing Strips

Start Tweaking Strips Action

Reference

Menu: *Edit* → *Start Tweaking Strips Action*
Hotkey: `Tab`

The contents of Action strips can be edited, but you must be in *Tweak Mode* to do so. The keyframes of the action can then be edited in the Dope Sheet.

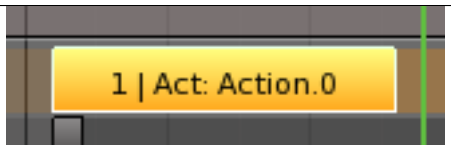


Fig. 2.247: Strip in NLA mode..

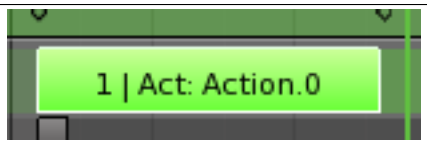


Fig. 2.248: Strip in Tweak mode.

When your finished editing the strip, simply go to *Edit* → *Tweaking Strips Action* or press `Tab`.

Linked Duplicate

Reference

Menu: *Edit* → *Linked Duplicate*
Hotkey: `Alt-D`

The contents of one Action strip can be instanced multiple times. To instance another strip, select a strip, go to *Edit* → *Linked Duplicate*

Now, when any strip is tweaked, the others will change too. If a strip other than the original is tweaked, the original will turn to red.

Fig. 2.249: Linked duplicated strip being edited.

Properties & Modifiers

Properties

Strip properties can be accessed via the NLA Properties region.

Animation Data

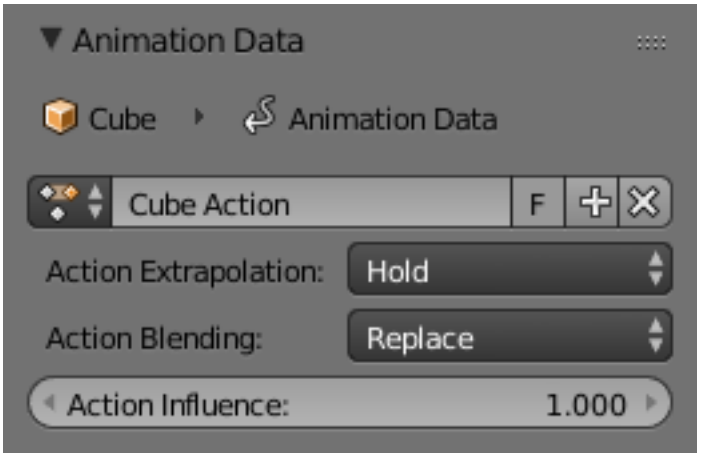


Fig. 2.250: Animation Data panel

Context

Action *Data-Block* allows you to edit actions shown in the action track.

Action Extrapolation Action to take for gaps past the strip extents.

Hold Affects both sides of the strip.

Hold Forward Affects the region after the clip, only.

Nothing Neither.

Action Blending Affects the behavior when two tracks simultaneously have a curve affecting the same property.

Replace Causes the top strip to take precedence according to the parameters of the Blend In/Out (see next option, below).

Multiply, Subtract, Add

Action Influence

Active Track

Name Name of the track which the strip currently belongs to.

Active Strip

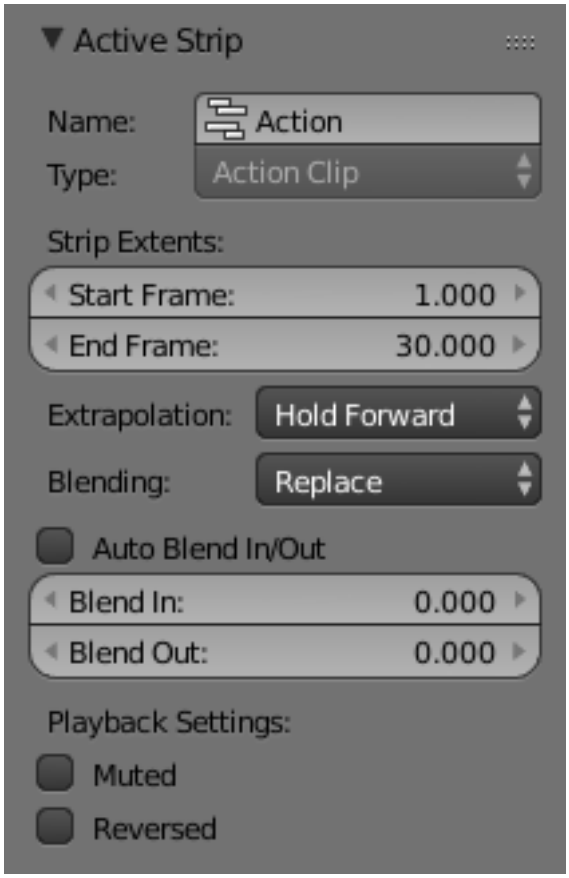


Fig. 2.251: Active Strip panel

Options of the strip itself.

Name Renames the strips.

Type Will either say “Action Clip”, “Transition”, or “Meta”, according to the three types of strips.

Strip Extents The boundaries of the strip itself. Note that this will stretch the duration of the Action, it will not cause greater or fewer keyframes from the Actions to play (see below for that option).

Extrapolation See *Action Extrapolation* above.

Blending See *Action Blending* above.

Auto Blend In/Out Creates a ramp starting at the overlap of the strips. The first strip has full control, and it ramps linearly giving the second strip full control by the end of the overlapping time period.

Blend In Set the frame that represents when this strip will have full influence.

Blend Out Set the last frame of this strip's full influence.

Muted Mute a single strip (like muting the track, above). Causes the track outline to be dashed.

Reversed Cause this strip to be played completely backwards.

Action Clip

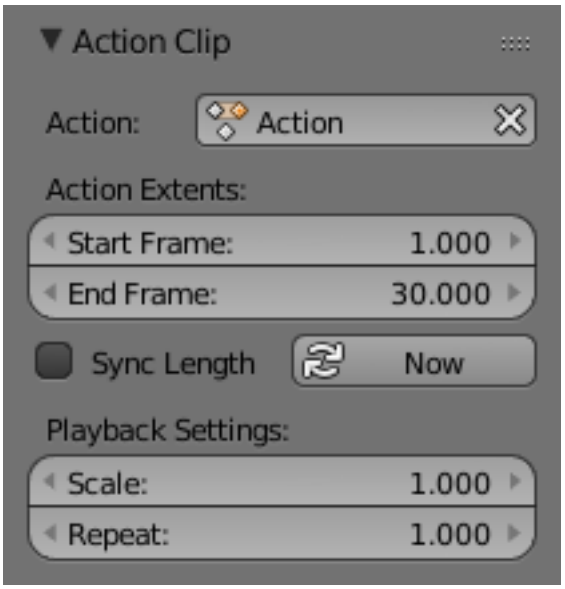


Fig. 2.252: Action Clip panel

This represents the 'object data' of the strip. Much like the transform values of an object.

Action A reference to the Action contained within the strip. Can be changed to replace the current strip's value with another Action.

Action Extents How much of the Action to use.

> Note: If you select values that are above or below the actual keyframe count of the Action, then the F-Curve Extrapolation will be consulted. Which can be changed in the Graph Editor, under *Channel → Extrapolation Mode*.

Sync Length Causes the "Start" and "End" Frames, above, to be reset to the first and last keyframed frames of the Action.

Sync Action Length "Now" Causes the "Start" and "End" Frames, above, to be reset to the first and last keyframed frames of the Action.

Playback Settings

> **Scale** Stretches strip, another way of increasing the *Strip Extents: End Frame*, above.
>
> **Repeat** Also expands the strip, but by looping from the first keyframe and going forward.

Evaluation

This determines the degree of influence the strip has, and over what time.

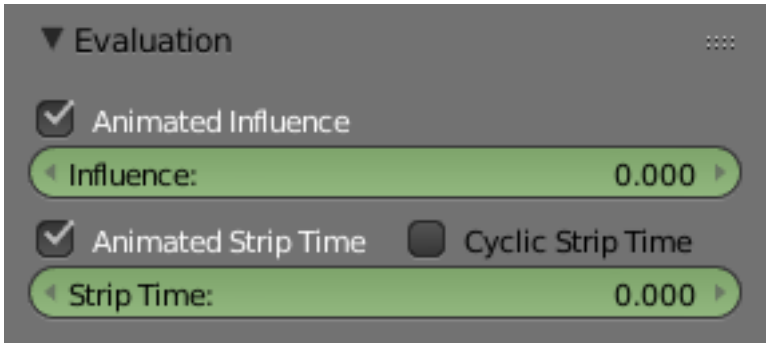


Fig. 2.253: Evaluation panel

Animated Influence Enabling alteration of the degree of influence this strip has as a keyframable value. If influence isn't animated, the strips will fade linearly, during the overlap.

Animated Strip Same as *Animated Influence*, but with *Strip Time*.

Cyclic Strip Time Cycle the animated time within the action start and end.

Modifiers

Like its close cousins in mesh and graph editing, Modifiers can stack different combinations of effects for strips.

See *F-Curve Modifiers*.

2.2.3 Image/Video

UV/Image Editor

Introduction

TODO see https://developer.blender.org/T46878

The UV/Image Editor is where you can edit 2D assets like images/textures and UVs.

Header

View Tools for controlling how the content is displayed in the editor. See *Navigating*.

Select Tools for *Selecting UVs*

Image This contains options for *Image*.

UVs Contains tools for *Unwrapping Meshes* and *Editing UVs*.

Modes

View Images and UV maps.

Paint *Texture Paint*.

Mask *Masking*.

Properties Region

Grease Pencil See the *Grease Pencil* Docs.

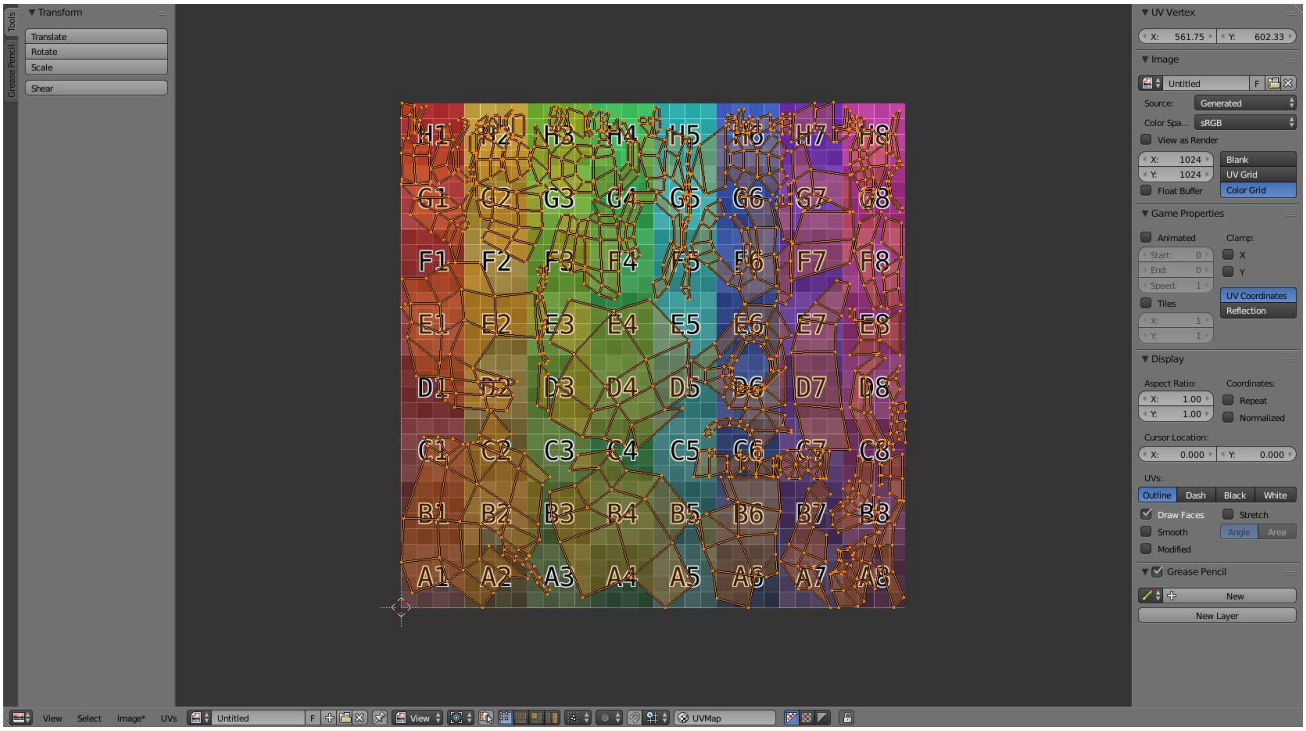


Fig. 2.254: UV/Image Editor with a UV map and a test grid texture.

Display Controls display options.

Navigating

2D View

Panning can be done by clicking the `MMB` and dragging.

Zooming can be done by scrolling `Wheel` up or down. Also, as in the 3D View, you can use `NumpadPlus` or `NumpadMinus` to zoom.

View Menu

Update Automatically Update the view in multiple areas.

UV Local View

Draw Other Objects Draws the UVs of selected objects (Object Mode) in the background.

Show Metadata Draws the Metadata if they were set in the render tabs *Metadata* panel.

View Zoom In/Out Adjusts the Zoom level `Wheel`.

Zoom Ratio

- Zoom 1:8 `Numpad8`
- Zoom 1:4 `Numpad4`
- Zoom 1:2 `Numpad2`
- Zoom 1:1 `Numpad1`
- Zoom 2:1 `Shift-Numpad2`

- Zoom 4:1 `Shift-Numpad4`
- Zoom 8:1 `Shift-Numpad8`

View Center Center the view to the entire UVs `NumpadPeriod`.

View All Center the view to the entire image `Home`.

View Fit Fit the view to the image dimensions `Shift-Home`.

Image

Introduction

Image Menu

New Image Creates a new *Generated* Image.

Open Image Load image from a file.

Read Render Layers

Save All Images

Replace Image Replaces the current image, while preserving the link to UV maps, with an selected file.

Reload Image Reloading the image from an external file.

Save Image Save the image, if the image is already a file `Alt-S`.

Save As Image Save the (rendered) image in a separate file `F3` or you want to save it under a different name.

Save a Copy Using *Save as Copy* will save the file to a specified name, but will keep the old one open in the UV/Image editor.

Edit Externally Using the *Edit Externally* tool Blender will open an external image editor, as specified in the *User Preferences* and load in the image to be edited.

Invert

Invert Image Colors Invert the colors of an image.

Invert Channel Red, Green, Blue, Alpha

Pack

Pack Image

Pack As PNG Packs the image inside the blend-file.

See also:

Pack and Unpack Data.

> **Warning:** Rendered images had to be saved externally.

Header Controls

Image Data-block menu used for selecting images. When an image has been loaded or created in the UV/Image editor, the Image panel appears in the *Properties region*. See *Image Settings*.

- Render Result
- Viewer Node

Pin Image Displays current image regardless of selected object.

Multi-Layer

When a rendered image is displayed in the UV/Image Editor, several new menu items become available.

Slot You can save successive renders into the render buffer by selecting a new slot before rendering. If an image has been rendered to a slot, it can be viewed by selecting that slot. Empty slots appear as blank grids in the UV/Image editor. Use the `J` and `Alt-J` to cycle forwards and backwards through saved renders.

Render Layer If you are using *Render Layers*, use this menu to select which layer is displayed.

Render Pass If you are using *Render Passes*, use this menu to select which pass is displayed.

Channels

Draw Channels The radio buttons set which channels of the image are displayed.

RGBA Replaces transparent pixels with background checkerboard, denoting the alpha channel.

RGB Draw the colored image, without alpha channel.

Alpha Displays the Alpha channel a gray-scale image. White areas are opaque, black areas have an alpha of 0.

Z-Buffer Display the depth from the camera, from Clip Start to Clip End, as specified in the *Camera settings*.

Red, Green, Blue Single Color Channel visualized as a gray-scale image.

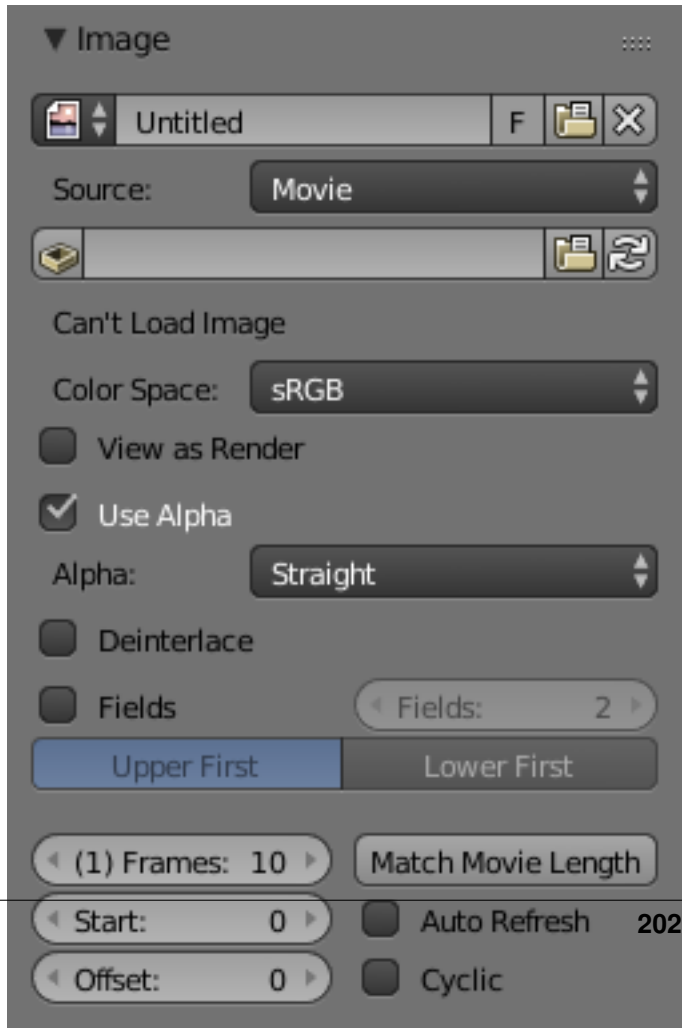
Image Settings

Image Data-block menu.

New + The *New Image* button opens a pop-up to configure a *Generated* image.

Source

See about supported *Supported Graphics Formats*.



Single Image

Still image or a single frame.

Image Sequence

Each frame is stored in a separate file. How to load a *Opening an Image Sequence*.

Frame A label showing the current frame.

further options See *Movie* below.

Movie

Frames packed into a container.

Deinterlace TODO

Fields Sets the number of fields per rendered frame (2 fields is 1 frame). Used with Fields and interlaced video, it says whether each image has both odd and even, or just one.

Frame

Frames Sets the range of frames to use.

Start Global starting frame of the sequence, when the playback should start.

Offset Offsets the first frame of the clip.

Match Movie Length This button set image's user's length to the one of selected movie.

Auto Refresh Automatically refresh images on frame changes.

Cyclic Start over and repeats after the last frame to create a continuous loop.

Generated

Image generated in Blender or preloaded.

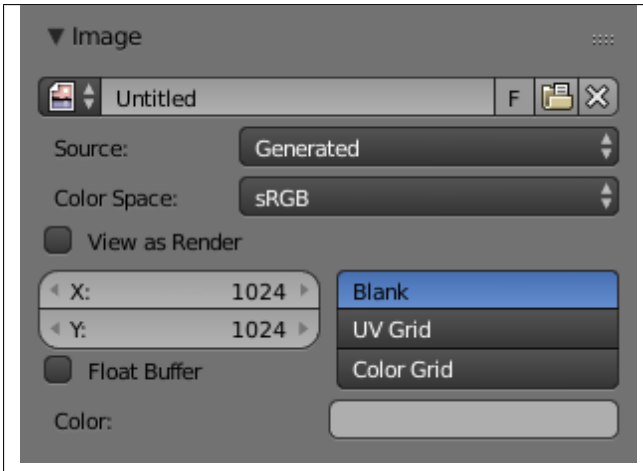


Fig. 2.256: Image panel for Generated source.

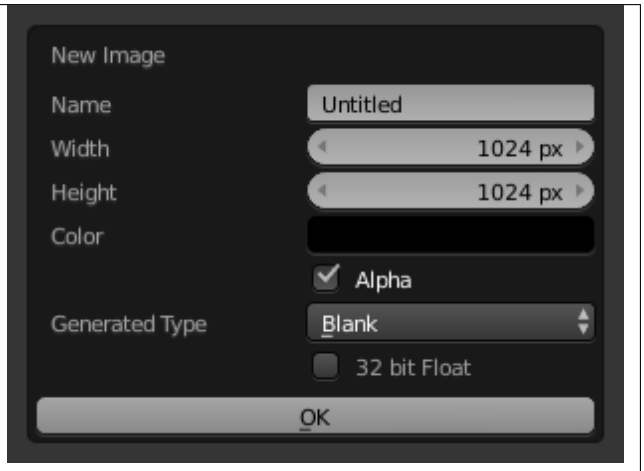


Fig. 2.257: The New Image pop-up menu.

Width, Height The size of image in pixels.

Color Sets the fill color if creating a blank image.

32 bit Float/ Float Buffer Creates a 32 bit image. This is a larger file size, but holds much more color information than the standard 8 bit image. For close ups and large gradients, it may be better to use a 32 bit image.

Type

Blank Creates a Blank image of a single specified color.

UV Grid Creates a checkerboard pattern with a colored cross (+) in each square.

Color Grid Creates a more complex colored grid with letters and numbers denoting locations in the grid. It could be used for testing how the UVs have been mapped and to reduce stretching or distortion.

Common Options

File Use for replacing or packing files.

Pack Embed the resource into the current blend-file.

Path Path to the linked file.

Open Opens the *File Browser* to select a file from the disk.

Reload Reloads the file. Useful when an file has been rework in an external application.

Color Space *Color Space*.

XYZ XYZ space.

VD16 The simple video conversion from a gamma 2.2 sRGB space.

sRGB Standard RGB display space.

Raw Raw space.

Non-Color Color space used for images which contains non-color data (i.e. normal maps).

Linear ACES ACES linear space.

Linear 709 (full range). Blender native linear space.

View as Render Apply render part of display transformation when displaying this image on the screen.

Use Multi-View See *Multi-View*.

Use Alpha Determines whether the alpha channel of the image is used.

Alpha Mode *Alpha Channel*.

Straight, Premultiplied

Fields Work with *Fields* images. Video frames consist of two different images (fields) that are merged. This option ensures that when fields are rendered, the correct field of the image is used in the correct field of the rendering. *MIP Mapping* cannot be combined with *Fields*. Order of video fields:

Upper First, Lower First.

Scopes

Histogram

This mode displays a graph showing the distribution of color information in the pixels of the currently displayed image. The X-axis represents values of pixel, from 0 to 1 (or 0 to 255), while the Y-axis represents the number of pixels in that tonal range. A predominantly dark image would have most of its information toward the left side of the graph.

Use this mode to balance out the tonal range in an image. A well balanced image should a nice smooth distribution of color values.

Luma Shows the luminacity of an image.

RGB Shows the RGB (Red, Green, Blue) channels stacked on top of each other.

R/G/B/A Depending on the channel you choose the scope will show the appropriate channel.

Show line Displays lines rather then filled shapes.

Waveform

Waveform Opacity Opacity of the points.

Waveform Mode TODO.

Vectorscope

Vectorscope Opacity Opacity of the points.

Sample Line

The *Sample Line* scope is the same as the *Histogram* but allows you to get the sample data from a line.

Sample Line Used to draw a line to use to read the sample data from.

Scope Samples

Full Sample Sample every pixle.

Accuracy Proportion of original image source pixel lines to sample.

UV Editing

Introduction

Header

Fig. 2.258: UV/Image Editor Header.

The header contains several menus and options for working with UVs.

Select Tools for *Selecting UVs*.

UVs Contains tools for *Unwrapping Meshes* and *Editing UVs*.

Pivot Point Selector Similar to working with Pivot Points in the 3D View.

Sync Selection Keeps UV and Mesh component selections in sync.

Selection Modes

- Vertex
- Edge
- Face
- Island

Sticky Selection Mode When Sync Selection is disabled, these options control how UVs are selected.

Proportional Editing See *Proportional Editing*.

UV Snapping Similar to Snapping in the 3D View.

Active UV Texture Map Selector Select which UV texture to use.

Auto Update Other Affected Windows Update other affected windows automatically to reflect changes during interactive operations e.g. transforms.

Properties Region

UV Vertex Panel

UV Vertex Transform Properties *Selecting UVs*.

Overview

The most flexible way of mapping a 2D texture over a 3D object is a process called "UV mapping". In this process, you take your three-dimensional (X, Y & Z) mesh and unwrap it to a flat two-dimensional (X & Y ... or rather, as we shall soon see, "U & V") image. Colors in the image are thus mapped to your mesh, and show up as the color of the faces of the mesh. Use UV texturing to provide realism to your objects that procedural materials and textures cannot do, and better details than Vertex Painting can provide.

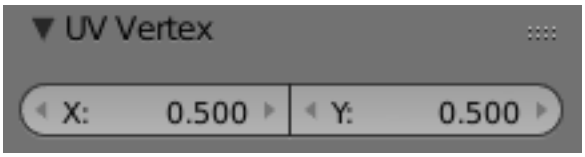


Fig. 2.259: UV Vertex Panel.

UVs Explained

Fig. 2.260: Box being inspected.

Fig. 2.261: Box mapped flat.

The best analogy to understanding UV mapping is cutting up a cardboard box. The box is a three-dimensional (3D) object, just like the mesh cube you add to your scene.

If you were to take a pair of scissors and cut a seam or fold of the box, you would be able to lay it flat on a tabletop. As you are looking down at the box on the table, we could say that U is the left-right direction, is V is the up-down direction. This image is thus in two dimensions (2D). We use U and V to refer to these "texture-space coordinates" instead of the normal X and Y, which are always used (along with Z) to refer to the three dimensional space (3D).

When the box is reassembled, a certain UV location on the paper is transferred to an (X, Y, Z) location on the box. This is what the computer does with a 2D image in wrapping it around a 3D object.

During the UV unwrapping process, you tell Blender exactly how to map the faces of your object (in this case, a box) to a flat image in the UV/Image Editor. You have complete freedom in how to do this. (Continuing our previous example, imagine that, having initially laid the box flat on the tabletop, you now cut it into smaller pieces, somehow stretch and/or shrink those pieces, and then arrange them in some way upon a photograph that is also lying on that tabletop).

Cartography Example

Cartographers (map makers) have been dealing with this problem for millennia. A cartography (map-making) example is creating a projection map of the whole world. In cartography, we take the surface of the earth (a sphere) and make a flat map that can be folded up into the glove compartment aboard the space shuttle. We "fill in" spaces toward the poles, or change the outline of the map in any of several ways:

Fig. 2.262: Mercator Projection.

Fig. 2.263: Mollweide Projection.

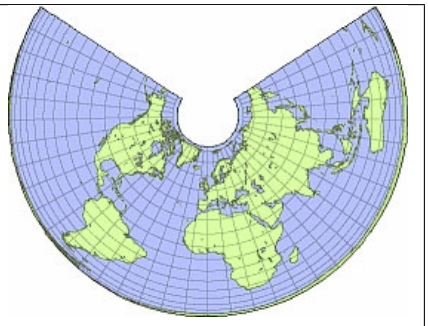


Fig. 2.264: Albers-equal Projection.

Each of these is an example of a way to UV map a sphere. Each of the hundred or so commonly accepted projections has its advantages and disadvantages. Blender allows the same thing anyway we want to, on the computer.

On more complex models (like seen in the earth map above) there pops up an issue where the faces cannot be cut, but instead they are stretched in order to make them flat. This helps making easier UV maps, but sometimes adds distortion to the final mapped texture. (Countries and states that are closer to the North or the South Pole look smaller on a flat map than do ones which are close to the Equator.)

Half-Sphere Example

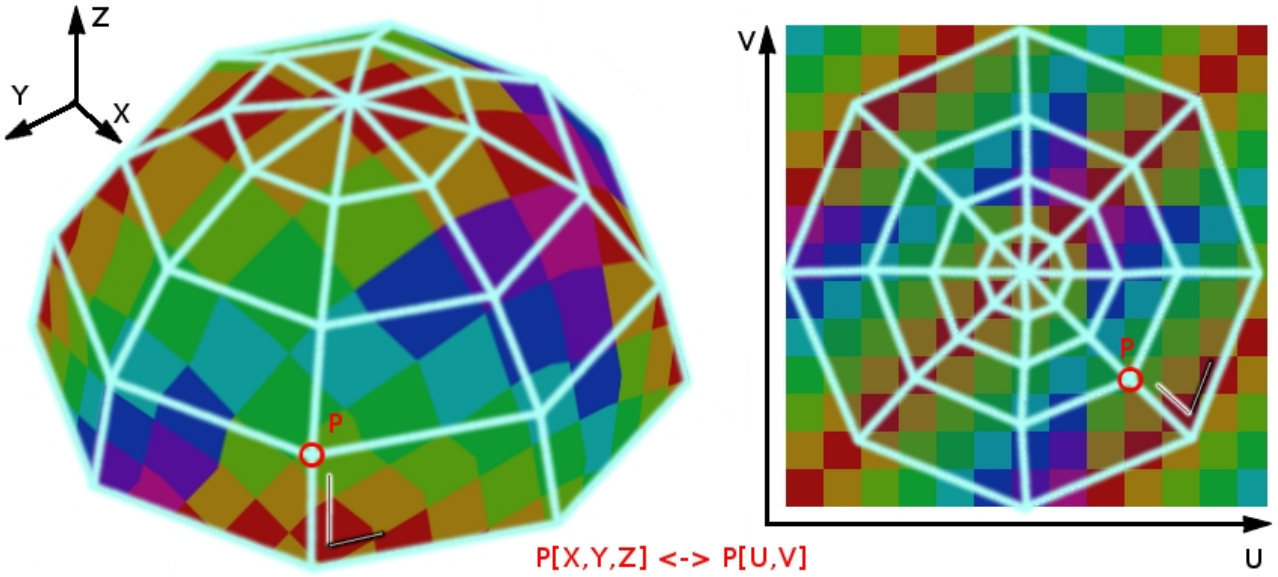


Fig. 2.265: 3D Space (XYZ) versus UV Space (click to enlarge).

In this image you can easily see that the shape and size of the marked face in 3D space is different in UV space.

This difference is caused by the "stretching" (technically called mapping) of the 3D part (XYZ) onto a 2D plane (i.e the UV map).

If a 3D object has a UV map, then, in addition to the 3D-coordinates X, Y, and Z, each point on the object will have corresponding U and V coordinates. (*P* in the image above is an

example of how a point on a 3D object might be mapped onto a 2D image.)

The UV/Image Editor

To learn about the functionalities for UV mapping see the *UV/Image Editor* section for details.

Advantages of UVs

While procedural textures (described in the previous chapters) are useful-they never repeat themselves and always "fit" 3D objects-they are not sufficient for more complex or natural objects. For instance, the skin on a human head will never look quite right when procedurally generated. Wrinkles on a human head, or scratches on a car do not occur in random places, but depend on the shape of the model and its usage. Manually-painted images, or images captured from the real world gives more control and realism. For details such as book covers, tapestry, rugs, stains, and detailed props, artists are able to control every pixel on the surface using a UV Texture.

A UV map describes what part of the texture should be attached to each polygon in the model. Each polygon's vertex gets assigned to 2D coordinates that define which part of the image gets mapped. These 2D coordinates are called UVs (compare this to the XYZ coordinates in 3D). The operation of generating these UV maps is also called "unwrap", since it is as if the mesh were unfolded onto a 2D plane.

For most simple 3D models, Blender has an automatic set of unwrapping algorithms that you can easily apply. For more complex 3D models, regular Cubic, Cylindrical or Spherical mapping, is usually not sufficient. For even and accurate projection, use seams to guide the UV mapping. This can be used to apply textures to arbitrary and complex shapes, like human heads or animals. Often these textures are painted images, created in applications like the Gimp, Photoshop, or your favorite painting application.

Note: Games

UV mapping is also essential in the *Game Engine*, or any other game. It is the de facto standard for applying textures to models; almost any model you find in a game is UV mapped.

Unwrapping

Introduction

The first step is to unwrap your mesh. You want to unwrap when you feel your mesh is complete with respect to the number of faces it needs to have. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you, but you may need to do additional mapping or editing. In this fashion, you can use the UV Texture image to guide additional geometry changes.

This section covers techniques for Mapping Uvs. The next sections cover *Editing UVs*, followed by methods of *Managing UV Layouts*, and *Applying Images to UVs*.

About UVs

Every point in the UV map corresponds to a vertex in the mesh. The lines joining the UVs correspond to edges in the mesh. Each face in the UV map corresponds to a mesh face.

Each face of a mesh can have many UV Textures. Each UV Texture can have an individual image assigned to it. When you unwrap a face to a UV Texture in the UV/Image Editor, each face of the mesh is automatically assigned *four UV coordinates:* These coordinates define the way an image or a texture is mapped onto the face. These are 2D coordinates, which is why they are called UV, to distinguish them from XYZ coordinates. These coordinates can be used for rendering or for real-time OpenGL display as well.

Every face in Blender can have a link to a different image. The UV coordinates define how this image is mapped onto the face. This image then can be rendered or displayed in real time. A 3D View has to be in "Face Select" mode to be able to assign Images or change UV coordinates of the active Mesh Object. This allows a face to participate in many UV Textures. A face at the hairline of a character might participate in the facial UV Texture, *and* in the scalp/hair UV Texture.

These are described more fully in the next sections.

Getting Started

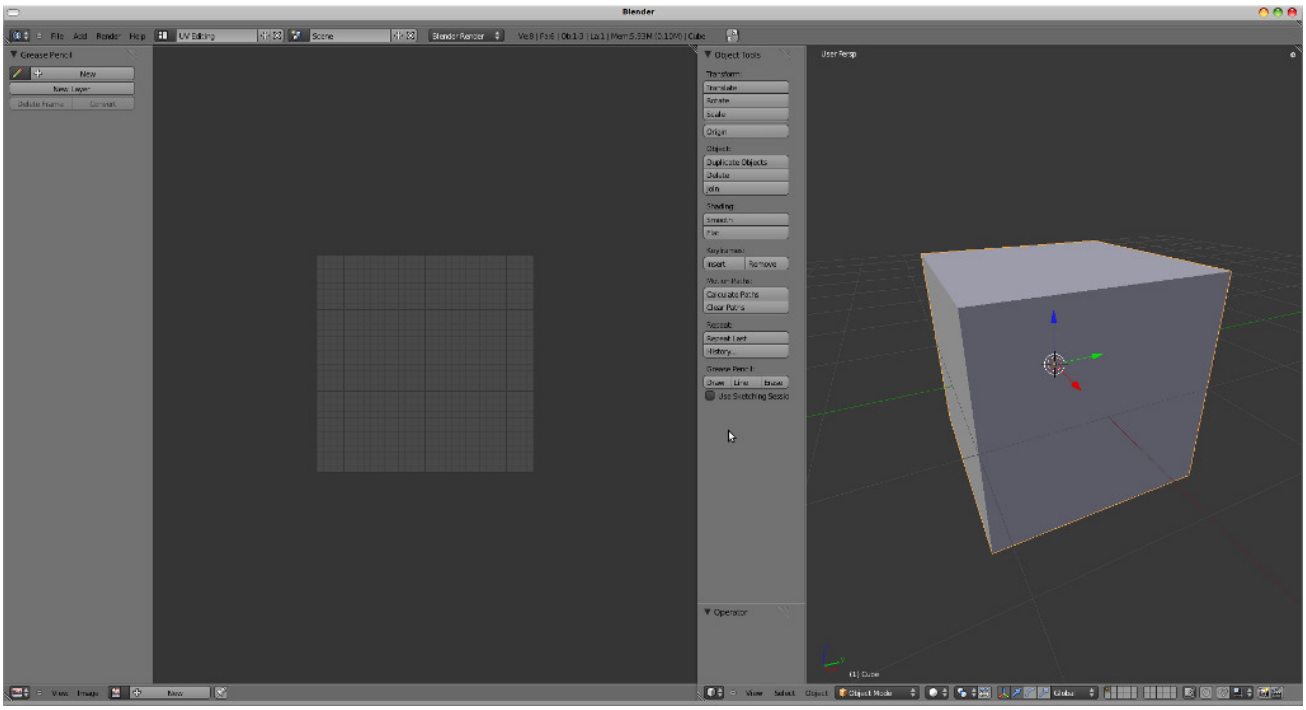


Fig. 2.266: UV Editing screen layout.

By default, meshes are not created with UVs. First you must map the faces, then you can *edit them*. The process of unwrapping your model is done within Edit Mode in the 3D View editor. This process creates one or more UV Islands in the *UV/Image Editor*.

To begin, choose the *UV Editing* *screen layout* from the selection list at the top of your screen in the User Preferences header. This sets one of the area to show you the UV/Image Editor `Shift-F10`, and the other area to the 3D View `Shift-F5`.

Enter *Edit Mode*, as all unwrapping is done in Edit Mode. You can be in vertex, face, or edge selection mode.

Workflow

The process for unwrapping is straightforward, but there are tons of options available, each of which dramatically affect the outcome of the unwrap. By understanding the meaning behind the options, you will become more efficient at unwrapping. The process is:

- Mark Seams if necessary
- Select all of the mesh components
- Select a UV mapping method from the UV Unwrap menu
- Adjust the unwrap settings
- Add a test image to see if there will be any distortion. See *Applying Images to UVs*
- Adjust UVs in the UV/Image editor. See *Editing UVs*

Mapping Types

Blender offers several ways of mapping UVs. The simpler projection methods use formulas that map 3D space onto 2D space, by interpolating the position of points toward a point/axis/plane through a surface. The more advanced methods can be used with more complex models, and have more specific uses.

Basic:

Cube Projection Maps the mesh onto the faces of a cube, which is then unfolded.

Sphere Projects the UVs onto a spherical shape. Useful only for spheres or spherical shapes, like eyes, planets, etc.

Cylinder Projects UVs onto a cylindrical surface.

Project from View Takes the current view in the 3D View and flattens it as it appears.

Advanced:

Unwrap Useful for organic shapes. Smooths the mesh into a flat surface by cutting along seams.

Smart UV Project Breaks the mesh into islands based on an angle threshold.

Lightmap Pack Separates each face and packs them onto the UV grid.

Follow Active Quads Follow UV from active quads along continuous face loops.

You can also *Reset* UVs, which maps each face to fill the UV grid, giving each face the same mapping.

If we were to use an image that was tileable, the surface would be covered in a smooth repetition of that image, with the image skewed to fit the shape of each individual face. Use this unwrapping option to reset the map and undo any unwrapping (go back to the start).

Unwrap

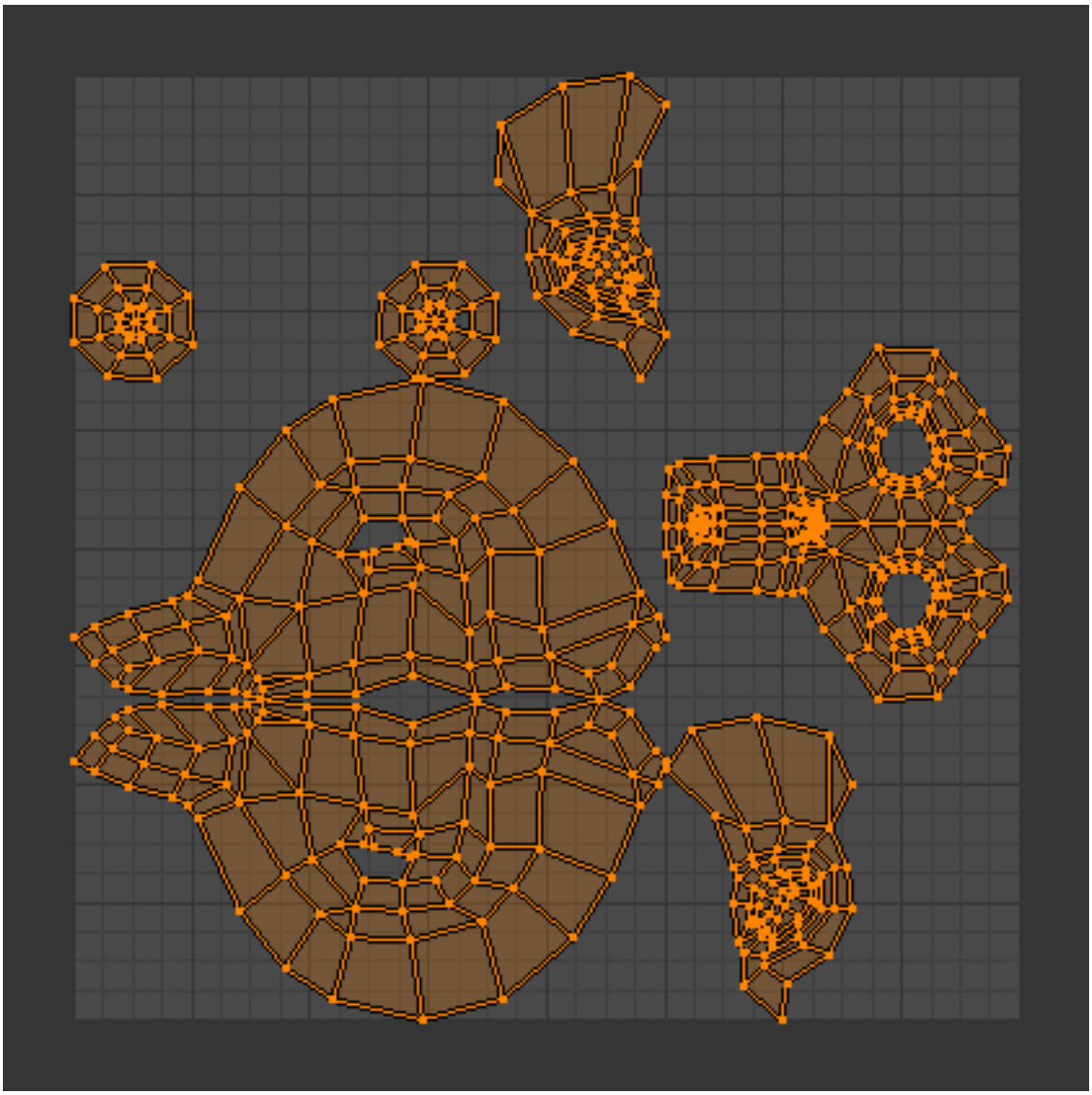


Fig. 2.267: Result of unwrapping Suzanne.

Begin by selecting all faces to be unwrapped in the 3D View. With our faces selected, it is now time to unwrap them. In the 3D View, select *Mesh* → *UV Unwrap* → *Unwrap* or U and select Unwrap.

You can also do this from the UV/Image Editor with *UVs* → *Unwrap* or E. This method will unwrap all of the faces and reset pre-

vious work. The UVs menu will appear in the UV/Image Editor after unwrapping has been performed once.

This tool unwraps the faces of the object to provide the "best fit" scenario based on how the faces are connected and will fit within the image, and takes into account any seams within the selected faces. If possible, each selected face gets its own different area of the image and is not overlapping any other faces UV's. If all faces of an object are selected, then each face is mapped to some portion of the image.

Operator panel

Blender has two ways of calculating the unwrapping. They can be selected in the tool setting in the tool panel in the 3D View.

Angle Based This method gives a good 2D representation of a mesh.

Conformal Uses LSCM (Least Squared Conformal Mapping). This usually gives a less accurate UV mapping than Angle Based, but works better for simpler objects.

Fill Holes Activating Fill Holes will prevent overlapping from occurring and better represent any holes in the UV regions.

Correct Aspect Map UVs taking image aspect into account.

Use Subsurf Modifier Map UVs taking vertex position after subsurf modifier into account.

Margin Space between UV islands.

Tip: A face's UV image texture only has to use *part* of the image, not the *whole* image. Also, portions of the same image can be shared by multiple faces. A face can be mapped to less and less of the total image.

Smart UV Project

Smart UV Project, (previously called the Archimapper) gives you fine control over how automatic seams should be created, based on angular changes in your mesh. This method is good for simple and complex geometric forms, such as mechanical objects or architecture.

This function examines the shape of your object, the faces selected and their relation to one another, and creates a UV map based on this information and settings that you supply.

In the example to the right, the Smart Mapper mapped all of the faces of a cube to a neat arrangement of three sides on top, 3 sides on the bottom, for all six sides of the cube to fit squarely, just like the faces of the cube.

For more complex mechanical objects, this tool can very quickly and easily create a very logical and straightforward UV layout for you.

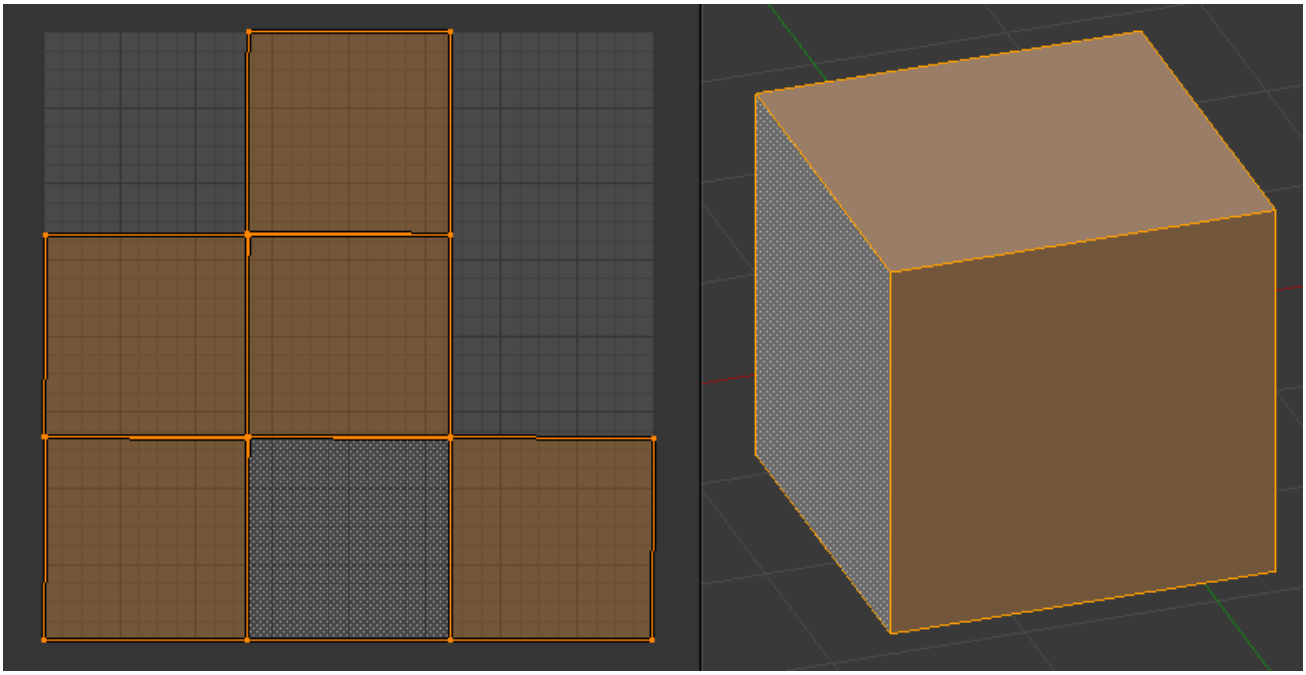


Fig. 2.268: Smart UV project on a cube.

Operator panel

The Tool Settings panel in the Tool Shelf allows the fine control over how the mesh is unwrapped:

Angle Limit This controls how faces are grouped: a higher limit will lead to many small groups but less distortion, while a lower limit will create fewer groups at the expense of more distortion.

Island Margin This controls how closely the UV islands are packed together. A higher number will add more space in between islands.

Area Weight Weight projection's vector by faces with larger areas.

Lightmap Pack

Lightmap Pack takes each of a mesh's faces, or selected faces, and packs them into the UV bounds. Lightmaps are used primarily in gaming contexts, where lighting information is baked onto texture maps, when it is essential to utilize as much UV space as possible. It can also work on several meshes at once. It has several options that appear in the Tool Shelf:

You can set the tool to map just *Selected Faces* or *All Faces* if working with a single mesh.

The *Selected Mesh Object* option works on multiple meshes. To use this, in *Object Mode* select several mesh objects, then go into *Edit Mode* and activate the tool.

Operator panel

Share Tex Space This is useful if mapping more than one mesh. It attempts to fit all of the objects' faces in the UV bounds without

overlapping.

New UV Layer If mapping multiple meshes, this option creates a new UV layer for each mesh. See *Managing the Layout*.

New Image Assigns new images for every mesh, but only one if *Shared Tex Space* is enabled.

Image Size Set the size of the new image.

Pack Quality Pre-packing before the more complex Box packing.

Margin This controls how closely the UV islands are packed together. A higher number will add more space in between islands.

Follow Active Quads

The *Face* → *Unwrap* → *Follow Active Quads* takes the selected faces and lays them out by following continuous face loops, even if the mesh face is irregularly shaped. Note that it does not respect the image size, so you may have to scale them all down a bit to fit the image area.

Operator panel

Edge Length Mode:

Even Space all UVs evenly.

Length Average space UVs edge length of each loop.

Note: Please note that it is the shape of the active quad in UV space that is being followed, not its shape in 3D space. To get a clean 90-degree unwrap make sure the active quad is a rectangle in UV space before using "Follow active quad".

Cube Projection

Cube mapping projects s mesh onto six separate planes, creating six UV islands. In the UV/Image editor, these will appear overlapped, but can be moved. See *Editing UVs*.

Basic Mapping

Based on the fundamental geometry of the object, and how it is being viewed, the *Mesh* → *UV Unwrap* → *Cube, Cylinder and Sphere* UV Calculations attempt to unfold the faces for you as an initial best fit. Here, the view from the 3D View is especially important. Also, the settings for cube size or cylinder radius (Editing buttons, UV Calculation panel) should be set (in Blender units) to encompass the object.

Operator panel

Cube Size Set the size of the cube to be projected onto.

Common

The following settings are common for the Cube, Cylinder, and Sphere mappings:

Correct Aspect Map UVs taking image aspect ratios into consideration. If an image has already been mapped to the texture space that is non-square, the projection will take this into account and distort the mapping to appear correct.

Clip to Bounds Any UVs that lie outside the (0 to 1) range will be clipped to that range by being moved to the UV space border it is closest to.

Scale to Bounds If the UV map is larger than the (0 to 1) range, the entire map will be scaled to fit inside.

Cylinder and Sphere Projection

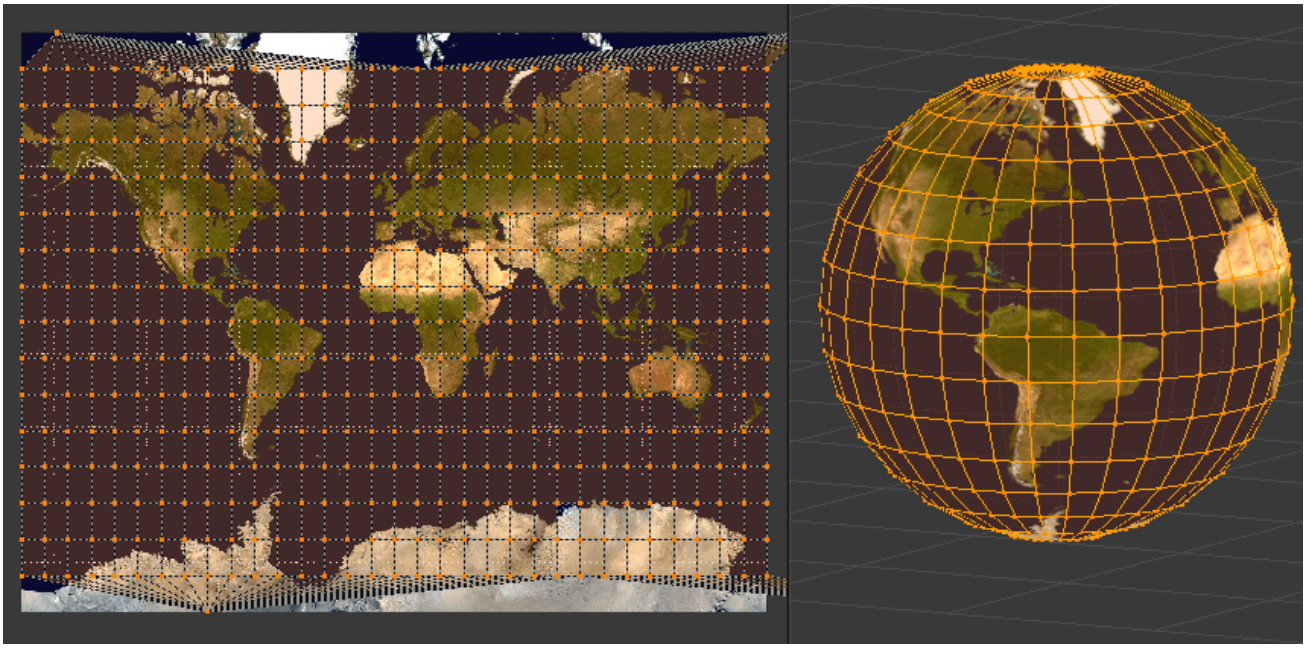


Fig. 2.269: Using a Mercator image with a Sphere Projection.

Cylindrical and Spherical mappings have the same settings. The difference is that a cylindrical mapping projects the UVs on a plan toward the cylinder shape, while a spherical map takes into account the sphere's curvature, and each latitude line becomes evenly spaced.

Normally, to unwrap a cylinder (tube) as if you slit it lengthwise and folded it flat, Blender wants the view to be vertical, with the tube standing "up". Different views will project the tube onto the UV map differently, skewing the image if used. However, you can set the axis on which the calculation is done manually. This same idea works for the sphere mapping:

Recall the opening cartographer's approaching to mapping the world? Well, you can achieve the same here when unwrapping a sphere from different perspectives. Normally, to unwrap a sphere,

view the sphere with the poles at the top and bottom. After unwrapping, Blender will give you a Mercator projection; the point at the equator facing you will be in the middle of the image. A polar view will give a very different but common projection map. Using a Mercator projection map of the earth as the UV image will give a very nice planet mapping onto the sphere.

Operator panel

Direction

View on Poles Use when viewing from the top (at a pole) by using an axis that is straight down from the view.

View on Equator Use if view is looking at the equator, by using a vertical axis.

Align to Object Uses the object's transform to calculate the axis.

Align Select which axis is up.

Polar ZX Polar 0 is on the X axis.

Polar ZY Polar 0 is on the Y axis.

Radius The radius of the cylinder to use.

Project From View

In the 3D View, the *Face* → *Unwrap UVs* → *Project from View* option maps the face as seen through the view of the 3D View it was selected from. It is almost like you had x-ray vision or squashed the mesh flat as a pancake onto the UV map. Use this option if you are using a picture of a real object as a UV Texture for an object that you have modeled. You will get some stretching in areas where the model recedes away from you.

Project From View (Bounds)

Using *Project from View (Bounds)* will do the same as above, but scales the UVs to the bounds of the UV space.

Reset

In the 3D View, *Face* → *Unwrap* → *Reset* maps each selected face to the same area of the image, as previously discussed. To map all the faces of an object (a cube, for example) to the same image, select all the faces of the cube, and unwrap them using the Reset menu option.

Seams

Introduction

For many cases, using the Unwrap calculations of Cube, Cylinder, Sphere, or best fit will produce a good UV layout. However,

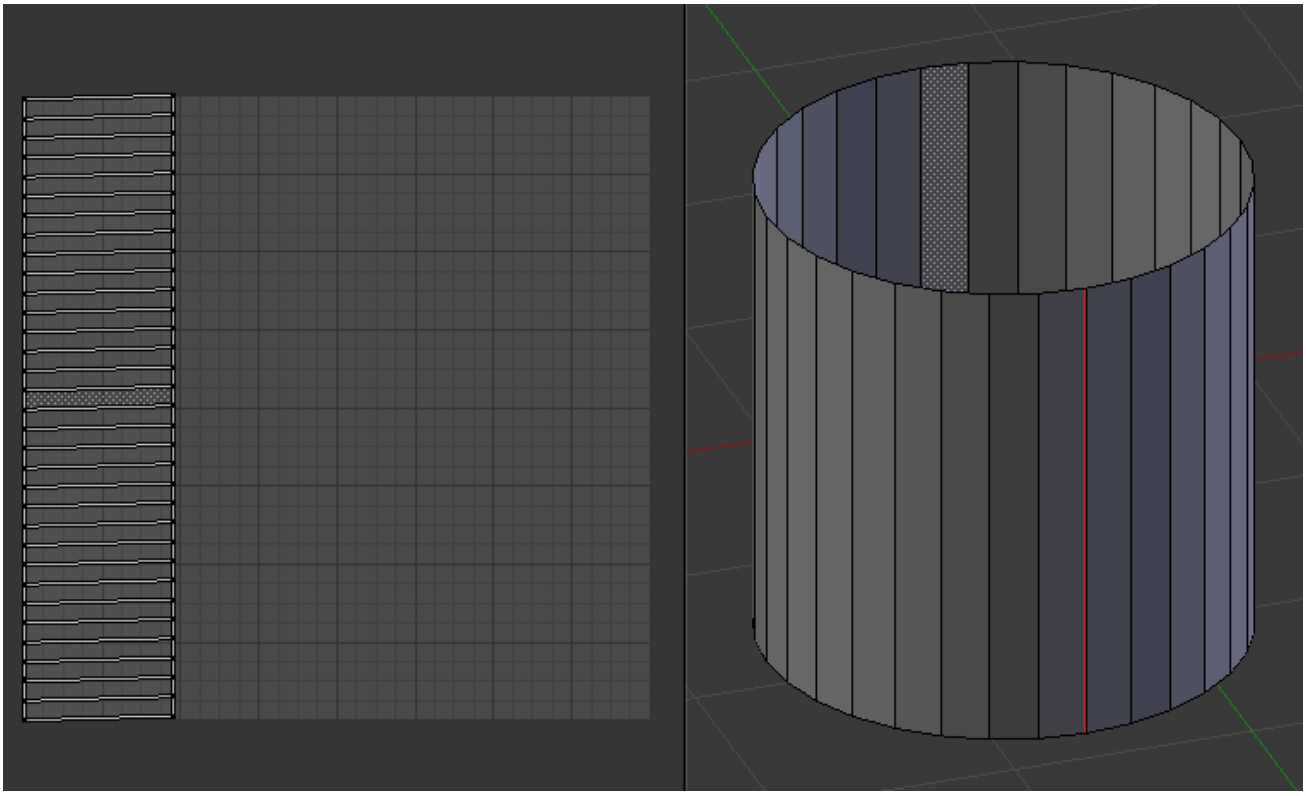


Fig. 2.270: Simple Seam on a Cylinder.

for more complex meshes, especially those with lots of indentations, you may want to define a *seam* to limit and guide any of the unwrapping processes discussed above.

Just like in sewing, a seam is where the ends of the image/cloth are sewn together. In unwrapping, the mesh is unwrapped at the seams. Think of this method as peeling an orange or skinning an animal. You make a series of cuts in the skin, then peel it off. You could then flatten it out, applying some amount of stretching. These cuts are the same as seams.

When using this method, you need to be aware of how much stretching there is. The more seams there are, the less stretching there is, but this is often an issue for the texturing process. It is a good idea to have as few seams as possible while having the least amount of stretching. Try to hide seams where they will not be seen. In productions where 3D paint is used, this becomes less of an issue, as projection painting can easily deal with seams, as opposed to 2D texturing, where it is difficult to match the edges of different UV islands.

The workflow is the following:

- Create seams. A seam is marked in Edit Mode by selecting edges to make the seam and then issuing the command to Mark Seam.
- Unwrap.
- Adjust seams and repeat.
- Manually adjust UVs. See the next section on Editing UVs.

Marking Seams

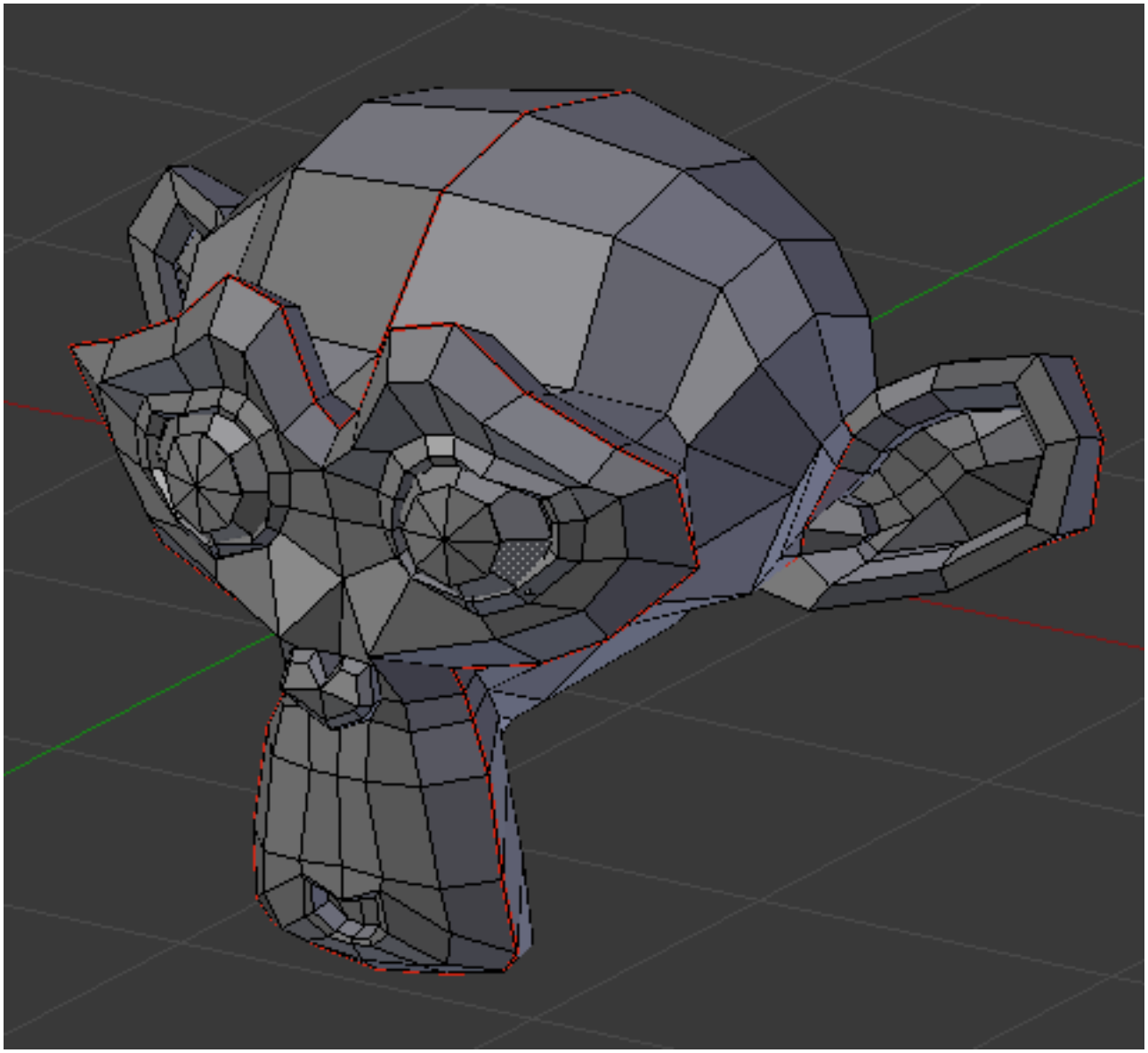


Fig. 2.271: Seamed Suzanne.

To add an edge to a seam, simply select the edge and `Ctrl-E` *Mark Seam*. To take an edge out of a seam, select it, `Ctrl-E` and *Clear Seam*.

In the example to the right, the back-most edge of the cylinder was selected as the seam (to hide the seam), and the default unwrap calculation was used. In the UV/Image Editor, you can see that all the faces are nicely unwrapped, just as if you cut the seam with a scissors and spread out the fabric.

When marking seams, you can use the *Select* → *Linked Faces* or `Ctrl-L` in Face Select Mode to check your work. This menu option selects all faces connected to the selected one, up to a seam. If faces outside your intended seam are selected, you know that your seam is not continuous. You do not need continuous seams, however, as long as they resolve regions that may stretch.

Just as there are many ways to skin a cat, there are many ways to go about deciding where seams should go. In general though, you should think as if you were holding the object in one hand, and a pair of sharp scissors in the other, and you want to cut it apart and spread it on the table with as little tearing as possible. Note that we seamed the outside edges of her ears, to separate the front from the back. Her eyes are disconnected sub-meshes, so they are automatically unwrapped by themselves. A seam runs along the back of her head vertically, so that each side of her head is flattened out.

Another use for seams is to limit the faces unwrapped. For example, when texturing a head, you do not really need to texture the scalp on the top and back of the head since it will be covered in hair. So define a seam at the hairline. Then, when you select a frontal face, and then select linked faces before unwrapping, the select will only go up to the hairline seam, and the scalp will not be unwrapped.

When unwrapping anything that is bilateral, like a head or a body, seam it along the mirror axis. For example, cleave a head or a whole body right down the middle in front view. When you unwrap, you will be able to overlay both halves onto the same texture space, so that the image pixels for the right hand will be shared with the left; the right side of the face will match the left, etc.

Note: You **do not** have to come up with "one unwrapping that works perfectly for everything everywhere." As we will discuss later, you can easily have multiple UV unwrappings, using different approaches in different areas of your mesh.

Managing UV Maps

After you finish editing a UV map, you may need to create additional maps on the same object, or transfer a UV map to another mesh.

Transferring UV Maps

You can copy a UV Map from one mesh to another Mesh provided both meshes have the same geometry/vertex order. This is useful for example when you want to recreate a UV map from an earlier version of your model with intact UVs.

Workflow

- `RMB` Select the target mesh (to which you want to copy the UV Map)
- `Shift` select the source mesh (that contains the intact UV map)
- *Object menu* → *Make Links...* → *Transfer UV Layouts* (Shortcut: `Ctrl-L` ...)

The target Mesh will now have a UV map that matches the original mesh.

Multiple UV Maps

You are not limited to one UV Map per mesh. You can have multiple UV maps for parts of the mesh by creating new UV maps. This can be done by clicking the *Add* button next to UV maps list (in *Object Data* tab in the Properties Editor) and unwrapping a different part of the mesh. UV maps always include the whole mesh.

UV Maps Panel

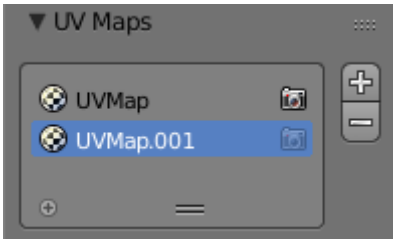


Fig. 2.272: The UV Maps panel in the Mesh tab.

In the Mesh tab the UV maps panel contains a *List Views & Presets* that lists the UV maps created for this mesh. The selected map is displayed in the UV/Image Editor.

Active Render Click the camera icon to enable that UV texture for rendering. If no other map is explicitly specified.

Add + Clicking the *Add* button duplicates the selected UV map.

See also:

Note that each texture can be mapped to a specific UV texture. See the *Mapping* panel of the texture tab.

Selecting

Selection tools are available in the *Select Menu* in the header, and the shortcuts listed below:

Menu

Border Select Use the box lasso to select UV coordinates `B`. See *Border Select*.

Border Select Pinned Use the box lasso to select only pinned UV coordinates `Shift-B`.

Circle Select See *Circle Select*.

Select/Deselect All Selects or de-selects all UV coordinates `A`.

Inverse Inverts the current selection `Ctrl-I`.

Select Pinned Selects all pinned UVs `Shift-P`. See Pinning.

Select Linked This operator selects all UVs that are connected to currently selected UVs `Ctrl-L`. This works similarly to the command in 3D View.

More

Less

Unlink Selection `Alt-L` Cuts apart the selected UVs from the map. Only those UVs which belong to fully selected faces remain selected following this command. As the name implies, this is particularly useful to unlink faces and move them elsewhere. The hotkey is analogous to the mesh Separate command.

Header

Sync Selection Turning on the *Sync Selection* button causes selection of components in the 3D View to sync with their corresponding elements in the UV/Image editor. If off only the selected faces are displayed in the UV/Image editor. These two modes have very different results when transforming components in the UV/Image editor.

Selection Modes

Select Modes dependent on the Sync Selection.

Sync Selection Off

Vertex Select individual vertices.

Edge Select edges.

Face Select faces.

Island Select contiguous groups of faces.

Sticky Selection Mode This selector lets you enable automatic additional selection.

Shared Vertex Selects UVs that share a mesh vertex, even if they are in different UV locations.

Shared Location Selects UVs that are in the same UV location and share a mesh vertex.

Disabled Disables Sticky Selection. When you move a UV in this mode, each face owns its own UVs, allowing them to be separated.

Sync Selection On

When selecting UVs or Edges, it behave like *Shared Vertex* mode above. When selecting Faces, it behaves as in *Disabled Stick Selection* above.

- Vertex
- Edge
- Face

Editing UVs

After unwrap, you will likely need to arrange the UV maps into something that can be logically textured or painted. Your goals for editing are:

- Stitch some pieces (UV maps) back together.
- Minimize wasted space in the image.
- Enlarge the faces where you want more detail.
- Re-size/enlarge the faces that are stretched.
- Shrink the faces that are too grainy and have too much detail.

With a minimum of dead space, the most pixels can be dedicated to giving the maximum detail and fineness to the UV Texture. A UV face can be as small as a pixel (the little dots that make up an image) or as large as an entire image. You probably want to make some major adjustments first, and then tweak the layout.

Menu

Snap to pixel Will force the UVs to snap to the nearest pixels of an image if loaded.

Constraining to Image Bounds Turning on *Constrain to Image Bounds* will prevent UVs from being moved outside the 0 to 1 UV range.

UV Sculpt

Live Unwrap

Unwrap

Pin and Unpin

You can pin UVs so they do not move between multiple unwrap operations.

When Unwrapping a model it is sometimes useful to "Lock" certain UVs, so that parts of a UV layout stay the same shape, and/or in the same place.

Pinning is done selecting a UV, then by selecting *Pin* from the *UVs* menu, or the shortcut `P`. You can *Unpin a UV* with the shorctut `Alt-P`

Pinning is most effective when using the Unwrap method of UV mapping, for organic objects. An example is when you are modeling a symmetrical object using the *Mirror Modifier*. Some of the UVs on the mirror axis may be shared across the mirrored counterparts. You could pin the UVs that correspond to the midline, then align them on the X axis, and they will stay in that location.

Pinning also work great with the Live Unwrap tool. If you pin two or more UVs, with Live Unwrap on, dragging pinned UVs will interactively unwrap the model. This helps with fitting a UV island to a certain shape or region.

Pack Islands

The *Pack Islands* tool, shortcut `Ctrl-P`, will uniformly scale, then individually transform each Island so that they fill up the UV space as much as possible. This is an important tool for efficiently making use of the texture space.

Average Island Scale

Using the *Average Island Scale* tool, shortcut `Ctrl-A`, will scale each UV island so that they are all approximately the same scale.

Minimize Stretch

The *Minimize Stretch* tool, `Ctrl-V`, reduces UV stretch by minimizing angles. This essentially relaxes the UVs.

Stitch

Stitch, `V`, will join selected UVs that share vertices. You set the tool to limit stitching by distance in the Tool Settings, by activating *Use Limit* and adjusting the *Limit Distance*

Seams

Mark Seam

Clear Seam

Seams From Island

Copy Mirrored UV coords

Transform

- Translate `G`
- Rotate `R`
- Scale `S`
- Shear `Shift-Ctrl-Alt-S`

Axis Locking

Transformations can be locked to an axis by pressing `X` or `Y` after one of the transform tools. Also, holding the `MMB` will constrain movement to the X or Y axis.

Mirror

UVs can be mirrored on the Y axis or the X axis:

- Mirror X
- Mirror Y

You can also use the hotkey `Ctrl-M`, then enter `X` or `Y`, or hold the `MMB` and drag in the mirror direction.

Snap

Snapping in UV/image editor is similar to *Snapping in 3D*. For the snap to pixel options to work an image has to be loaded.

Selected to Pixels Moves selection to nearest pixel. See also *Snap to pixel* above.

Selected to Cursor Moves selection to 2D cursor location.

Selected to Cursor (Offset) Moves selection center to 2D cursor location, while preserving the offset of the vertices from the center.

Selected to Adjacent Unselected Moves selection to adjacent unselected element.

Cursor to Pixels Snaps the cursor to the nearest pixels.

Cursor to Selected Moves the Cursor to the center of the selection.

Weld/Align

The *Weld or Align* tool, `W`.

Weld The *Weld* tool will move selected UVs to their average position.

Remove Doubles UV

Straighten Auto, X, Y

Align Will line up the selected UVs on the X axis, Y axis, or automatically chosen axis.

Auto, X, Y

Proportional Editing

Proportional Editing is available in UV editing. The controls are the same as in the 3D View. See *Proportional Editing in 3D* for full reference.

Show/Hide Faces

- Reveal Hidden `Alt-H`
- Hide Select `H`
- Hide Unselect `Shift-H`

Export UV Layout

Using your favorite image painting program, you could use an exported UV layout to create a texture. Then save your changes, and back in Blender, use the *Image* → *Open* menu command to load it as your UV image for the mesh in Edit Mode for the desired (and active) UV Texture layer.

As a way of communicating to an artist who is painting your UV Texture for you, Blender has a tool called *Save UV Face Layout* (located in the UV/Image Editor, *UVs* → *Save UV Face Layout*) that saves an image as a `Targa` (`.tga`), `EPS`, or an `SVG` format for the object you have selected.

The image is an outline of the UV face mapping. Activating the tool brings up the File Browser with options for saving the layout:

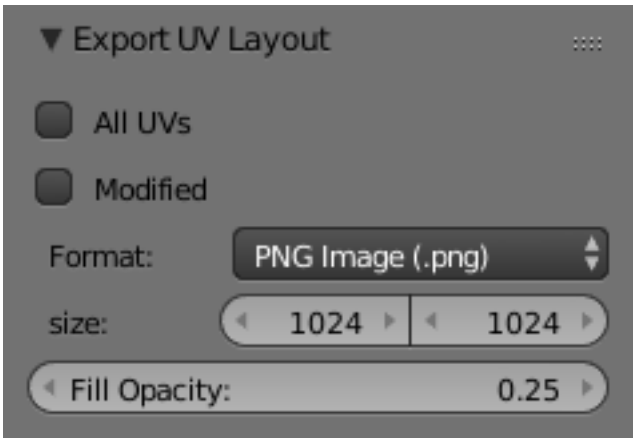


Fig. 2.273: Export Options.

All UVs if disabled, then only the UV faces selected will be outlined

Modified Export UVs from the modified mesh.

Format Select the type of image file to save (`.png`, `.eps`, `.svg`)

Size select the size of the image in pixels. The image be square.

Fill Opacity Set the opacity of the fill.

The image will be lines defining the UV edges that are within the image area of the UV mapping area. Edges outside the boundary, even if selected, will not be shown in the saved graphic.

The artist will use this as a transparent layer in their paint program as a guide when painting your texture. The example below shows Blender in the background, and the Gimp working on the texture, using the saved layout as a guide. Note that targa format supports the Alpha channel, so you can paint transparent areas of the mesh.

For using images as textures, see the page on *Image Textures*.

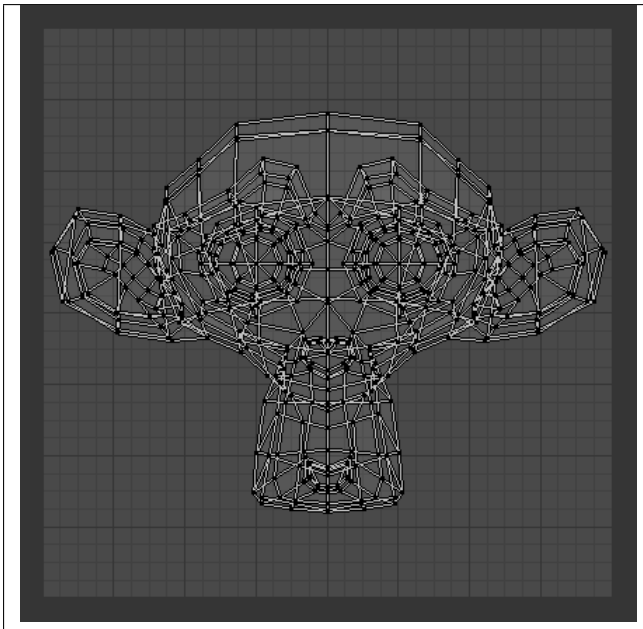


Fig. 2.274: A UV Layout in the UV/Image Editor.

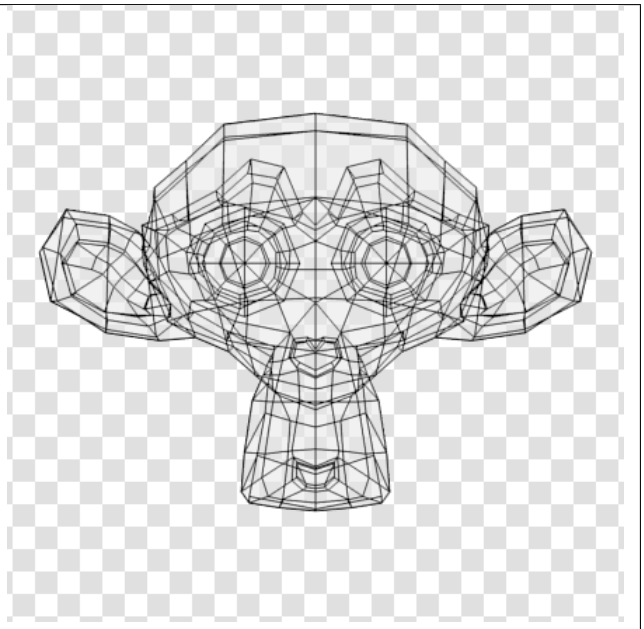


Fig. 2.275: A UV Layout in an paint program.

Header

Pivot Point

The UV/Image editor has a 2D cursor. Its position can be changed by LMB clicking in the UV/Image editor. You can also manually adjust its position in the Properties region. The range by default is from 0 to 256 starting from the lower left corner. By enabling *Normalized* under *Coordinates*, the range changes from 0 to 1.

The Pivot Point can be changed to:

- Bounding Box Center
- Median Point
- 2D Cursor Location

Proportional Editing

Proportional Editing is available in UV editing. The controls are the same as in the 3D View. See *Proportional Editing in 3D* for full reference.

Snap

UV Data

3D View

Face Mirror and Rotate UVs

The orientation of the UV Texture is defined by each face. If the image is, for example, upside down or laying on its side, use the

Face → *Rotate UVs* (in the 3D View in Face Select mode) menu to rotate the UVs per face in 90-degree turns.

The *Face* → *Mirror UVs* tool mirrors the UVs per face, which flips the image over, showing you the image reversed.

Layout Worflow

Optimizing the UV Layout

When you have unwrapped, possibly using seams, your UV layout may be quite disorganized and chaotic. You may need to proceed with the following tasks: Orientation of the UV mapping, arranging the UV maps, stitching several maps together.

The next step is to work with the UV layouts that you have created through the unwrap process. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you. In this fashion, you can use the UV Texture image to guide additional geometry changes.

When arranging, keep in mind that the entire view is your workspace, but only the UV coordinates within the grid are mapped to the image. So, you can put pieces off to the side while you arrange them. Also, each UV unwrap is its own linked set of coordinates.

You can lay them on top of one another, and they will onionskin (the bottom one will show through the top one). To grab only one though, `RMB` select one of the UV coordinates, and use *Select* → *Linked UVs*, `Ctrl-L` to select connected UVs, not border select because UVs from both will be selected.

Combining UV Maps

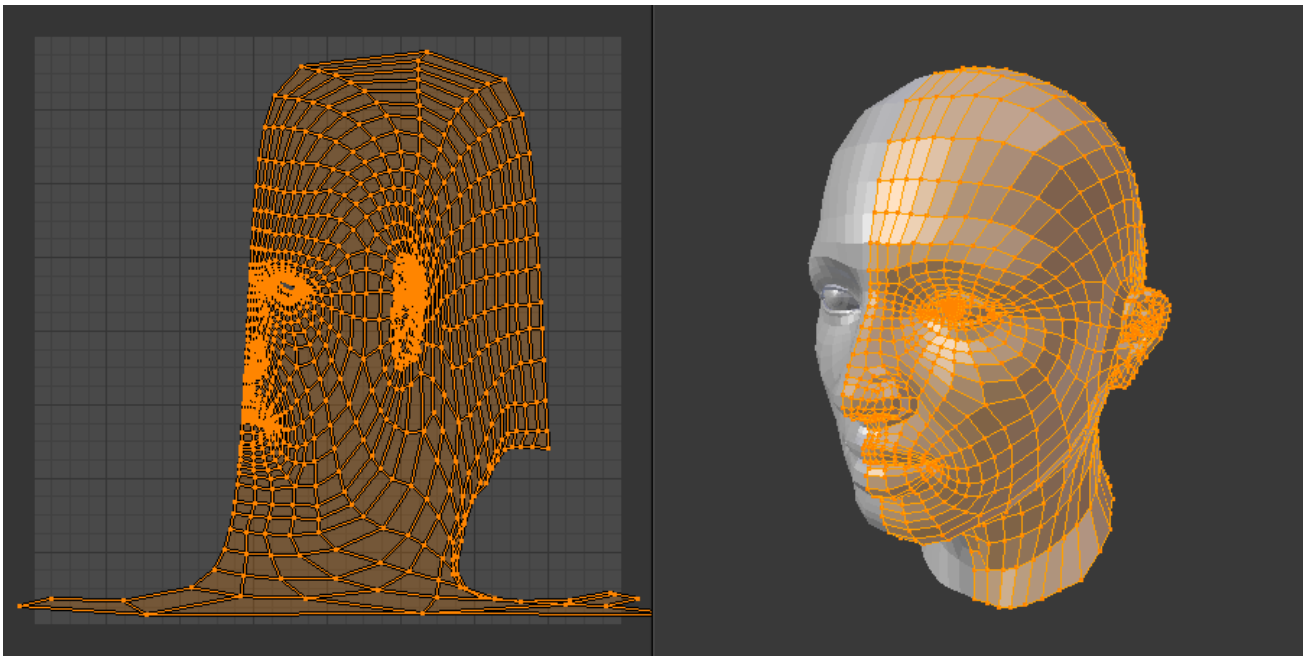


Fig. 2.276: Bad unwrap, note ear and neck.

Very often you will unwrap an object, such as the face example we have been using, and get it "mostly right" but with parts of the mesh that did not unwrap properly, or are horribly confusing. The picture to the right shows an initial unwrap of the face using the Unwrap from sphere option. The issues are with the ear; it is just a mush of UVs, and the neck, it is stretched and folded under. Too much work to clean up.

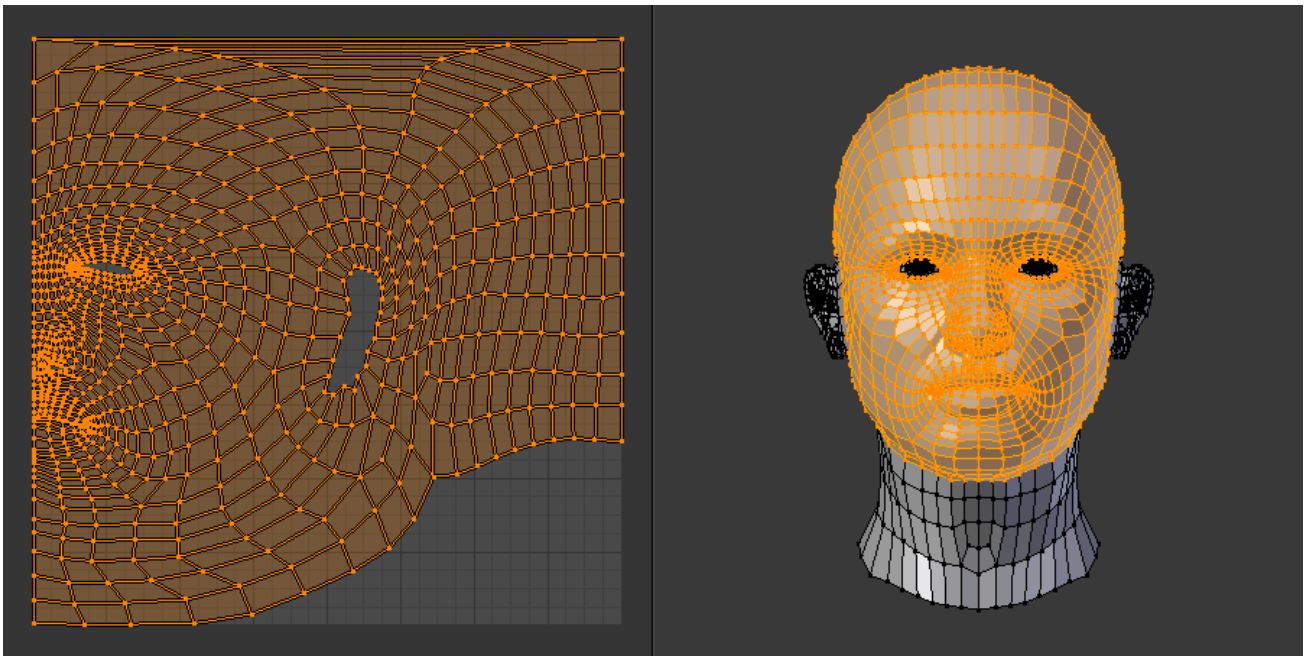


Fig. 2.277: Unwrap face only, without ear or neck.

We can tell that the ear would unwrap nicely with just a straightforward projection from the side view, and the neck with a tubular unwrap. So, our general approach will be to unwrap different parts of the object (face, ears, and so on) using different unwrap calculations, selecting each calculation according to whatever works best for that piece. So let us begin: We select only the "face" faces, unwrap them using the *Sphere* calculation, and scale and rotate them somewhat to fit logically within the image area of the UV/Image Editor.

Once we are satisfied with the face, it is time to turn our attention to the ear. First, unselect the faces you were working with. Their UVs will disappear from the UV/Image Editor, but they are still there, just not shown. (To verify this, you can select a few faces in 3D View and it will show up in the UV/Image Editor.)

To work on the ear, in the 3D View, we now select only the "ear" faces. You can use Vertex Groups to select the ear faces. Selecting sub-meshes is easy too, since they are not connected to the rest of the mesh. Simply selecting Linked vertices will select that entire submesh. Basically, since you are in edit mode, all of the selecting/unselecting features are available to you.

Now re-unwrap the ear using the *Project* calculation from side view, and scale and rotate them somewhat (discussed in the next section), and place them off to the side. You can do this repetitively, using different UV calculations; each re-calculation just puts those UVs for the selected faces somewhere else. Choose

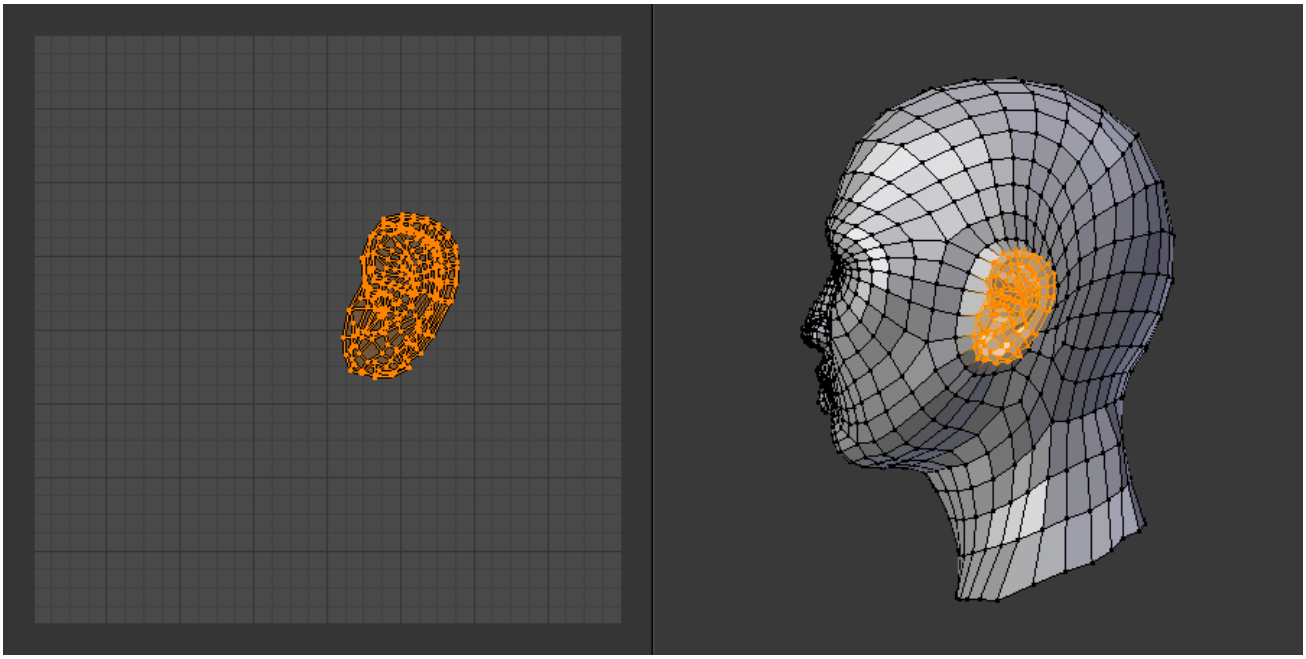


Fig. 2.278: Unwrap ear only, using the *Project From View*.

the calculation for each piece that gives you the best fit and most logical layout for subsequent painting of that piece.

When all of the pieces of the mesh have been unwrapped using the various calculations, you should end up with something that looks like to the Example to the right. All of the sections of the mesh have been mapped, and all those maps are laid out in the same UV Texture map. Congratulations! From here, it is a simple matter of "stitching" (discussed in the next section) to construct the entire UV Map as a single map.

When you have completed arranging and stitching, you will end up with a consolidated UV Map, like that shown to the right, arranged such that a single image will cover, or paint, all of the mesh that needs detailed painting. All of the detailed instructions on how to do this are contained in the next section. The point of this paragraph is to show you the ultimate goal. Note that the mesh shown is Mirrored along the Z axis, so the right side of the face is virtual; it is an exact copy of the right, so only one set of UVs actually exist. (If more realism is desired, the *Mirror* modifier would be applied, resulting in a physical mirror and a complete head. You could then make both side physically different by editing one side and not the other. Unwrapping would produce a full set of UVs (for each side) and painting could thus be different for each side of the face, which is more realistic.)

Iteration and Refinement

At least for common people, we just do not "get it right the first time." It takes building on an idea and iterating our creative process until we reach that magical milestone called "Done." In software development, this is called the Spiral Methodology.

Applied to Computer Graphics, we cycle between modeling, texturing, animating, and then back to making some modifications to mesh, re-UV mapping, tweaking the animation, adding a bone or two, finding out we need a few more faces, so back to modeling,

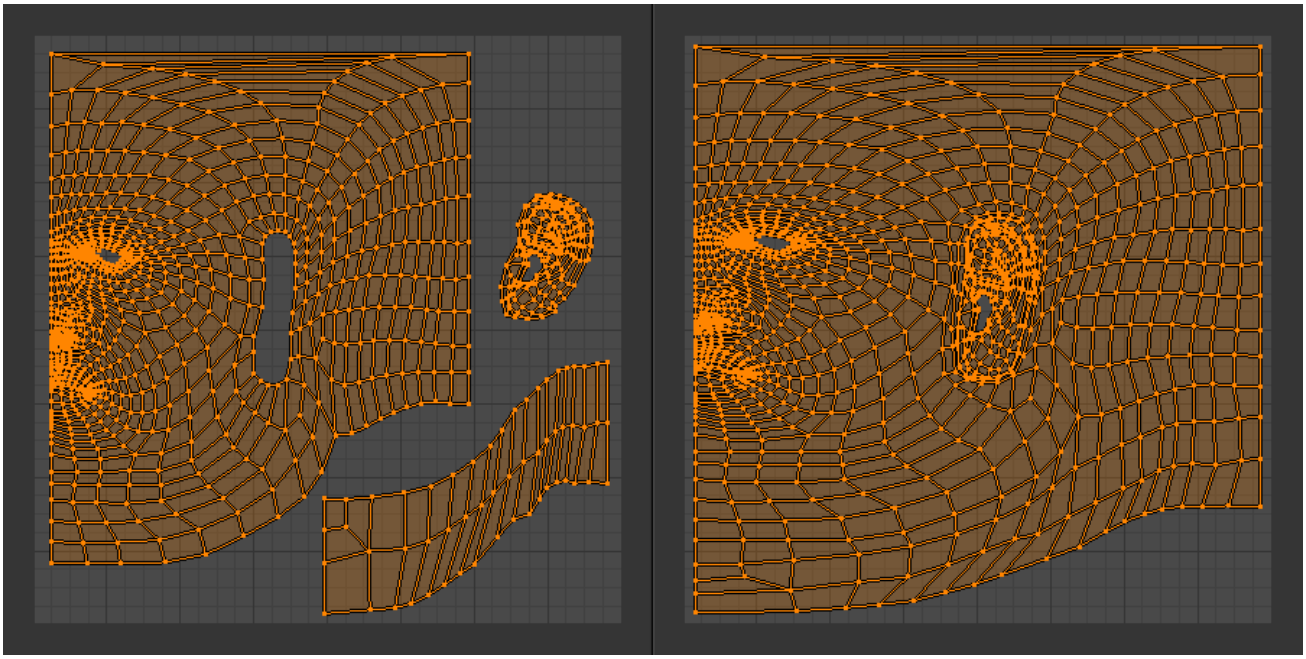


Fig. 2.279: UV Maps arranged together and stitched.

etc. We continue going round and round like this until we either run out of time, money, or patience, or, in some rare cases, are actually happy with our results.

Refining the Layout

Refinement comes into play when we finally look at our character, and realize that we need more detail in a particular spot. For example, areas around the eyes might need crow's feet, or we need to add a logo to the vest. As you start to edit the image, you realize that there just are not enough pixels available to paint the detail that you want.

Your only choice is to expand the size (scale out) that UV face. Using the minimize stretch or scale commands, you expand the UV faces around the eyes or chest, allocating more pixels to those areas, but at the same time taking away pixels (detail) from something else, like the back of the head. After refining the UV map, you then edit the image so that it looks right and contains the details you want.

Reusing Textures

Another consideration is the need to conserve resources. Each image file is loaded in memory. If you can re-use the same image on different meshes, it saves memory. So, for example, you might want to have a generic face painting, and use that on different characters, but alter the UV map and shape and props (sunglasses) to differentiate.

You might want to have a "faded blue jeans" texture, and unwrap just the legs of characters to use that image. It would be good to have a generic skin image, and use that for character's hands,

feet, arms, legs, and neck. When modeling a fantasy sword, a small image for a piece of the sword blade would suffice, and you would Reset Unwrap the sword faces to re-use that image down the length of the blade.

Applying Textures

Sooner or later, you may want to use an image texture on your model. If you are using an external application, you need to know where on the mesh you are painting. You may also need to test your UV mapping with a test image. This section covers how to export an outline of your UV map, and how to load images into the UV/Image editor.

Applying Textures to UVs

The UV/Image Editor allows you to map textures directly to the mesh faces. The 3D View editor shows you the object being textured. If you set this editor into Textured viewport shading, you will immediately see any changes made in the UV/Image and this editor, and vice versa.

You can edit and load images, and even play a game in the Blender Game Engine with UV textures for characters and object, without a material, and still see them in the 3D View. This is because no real rendering is taking place; it is all just viewport shading. If you were to apply an image to UVs then render, the texture would not show up by default.

To render an image however, you must:

- Create a Material for the object, and
- tell Blender to use the UV Textures on faces when rendering.

To create a Material, you have to click *Add New* Material in the Shading context.

There are two ways to tell Blender to use the UV Texture when rendering: the Proper way and the Quick Way:

Use UV Coordinates

In the Texture channel panel, Add a New Texture and define the texture as an image and load the image you want to use. In the Mapping section, choose UV from the Coordinates menu, and select the UV layer to use.

Make sure it is mapped to Color in the Influence section as well (it will be mapped to Color by default, and the UV Texture is named "UVTex" by default). If the image has an alpha channel and you want to use it, click "Use Alpha" in the Map Image panel.

Full details of using Image textures are on the *Image Textures* page.

Note: Material is Required for Rendering

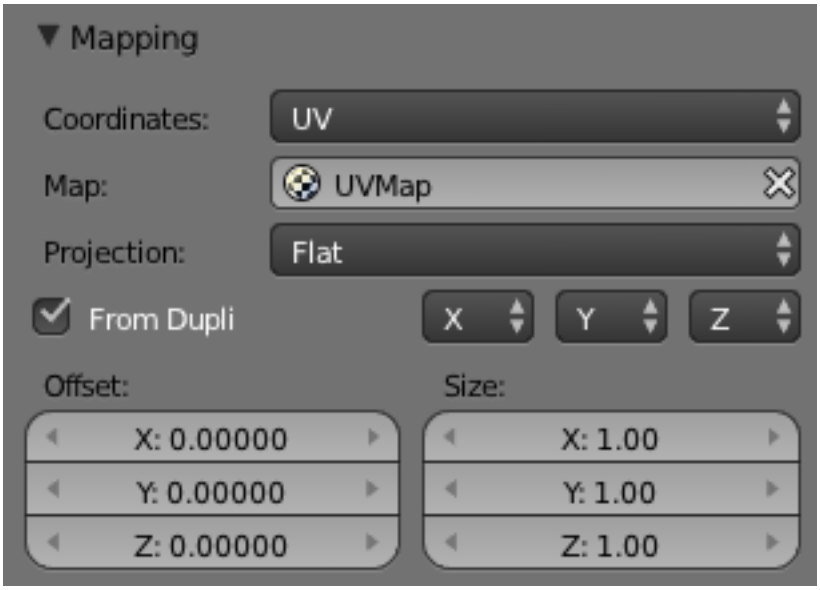


Fig. 2.280: A texture setup to map using its UV coordinates.

You can perform UV Texturing on a mesh within Blender without assigning a material, and you will even see it in your 3D View in textured viewport mode. However, when you render, you will just get a default gray if the object does not have a Material assigned. You will get a black if you do not load an image. If you do not create a texture that uses the image, or enable *Face Texture*, your object will render according to the procedural material settings.

Face Textures

Fig. 2.281: The Material panel with activated Face Textures button.

An alternate way is to set up a Face Textures Material as shown. To do so, with the Properties editor displayed, press `F5` to display the Shader Buttons. In the Properties editor, Material settings, click *Add New* material.

On the Options panel, enable *Face Textures*. This way is quick, but bypasses the normal rendering system for fast results, but results which do not respect transparency and proper shading.

Using the Test Grid

If your image is a base uniform pattern and you want the application of that image to your model to look like cloth, you do **not** want any stretching (unless you want the cloth to look like spandex).

Fig. 2.282: The test grid applied to the UVs.

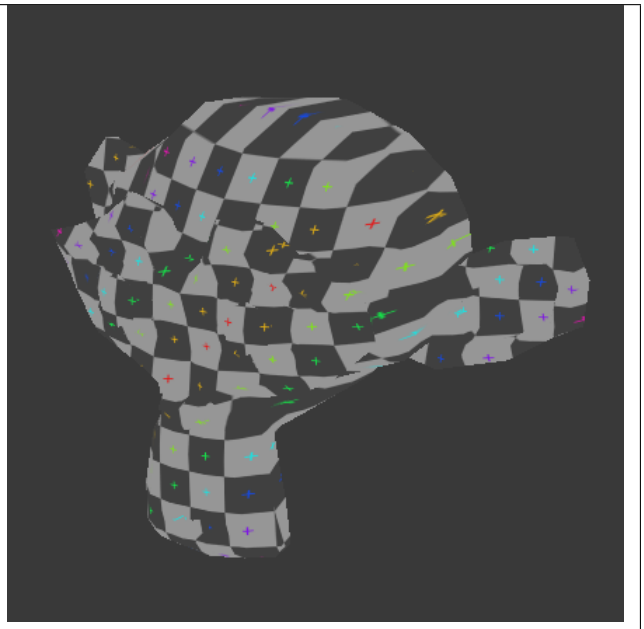


Fig. 2.283: A preview of the texture on the geometry.

When you render, the mesh will have the test grid as its colors, and the UV Texture will be the size image you specified.

Modifying your Image Texture

See also:

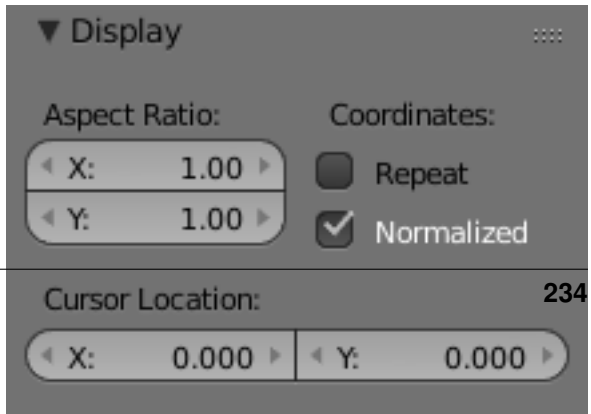
- *Render Bake*
- *Texture Paint*.

The advantage to saving as a separate file is that you can easily switch textures just by copying other image files over it, and you can use external editing programs to work on it. The advantage of packing is that your whole project is kept in the blend-file, and that you only have to manage one file.

Display Panel

You can set the editors display options in the this panel.

Image



Aspect Ratio Display Aspect for this image. Does not affect rendering.

Coordinates

Repeat Draw the image repeated outside of the main view.

UV

Coordinates

Normalized Display UV coordinates from 0.0 to 1.0 rather than in pixels.

Cursor Location 2D cursor location for this view.

UVs

Edge Draw Type Sets how UV edges are displayed.

Outline, Dash, Black, White

Draw Faces Draw faces over the image.

Smooth Makes edges appeared anti-aliased.

Modified Show results of modifiers in the UV display.

Stretch Shows how much of a difference there is between UV coordinates and 3D coordinates. Blue means low distortion, while Red means high distortion. Choose to display the distortion of *Angles* or the *Area*.

Painting

TODO see https://developer.blender.org/T46878

Masking

Introduction

Masks have many purposes. They can be used in a motion tracking workflow to mask out, or influence a particular object in the footage. They can be used for manual rotoscoping to pull a particular object out of the footage, or as a rough matte for green screen keying. Masks are independent from a particular image of movie clip, and so they can just as well be used for creating motion graphics or other effects in the compositor. These masks can also be used in other places in Blender.

Editing Masks

Masks can be created in the image and movie clip editors, by changing the mode from View to Mask in the header. This will add various tools and properties to the editor panels, while hiding others that are not needed for interacting with masks. The tools and panels available to edit masks are the same in both editors, with the exception that linking masks to motion tracking data is only possible in the movie clip editor.

Once set to Mask mode, a Mask data-block can be added. Any image, movie clip, render or compositing result can be used as a backdrop to draw masks over. To get interactive feedback on the resulting mask, a Mask node can be connected directly to a Viewer node in the compositor, which will then keep updating the compositing result while editing.

S-Curves

The curve type used for creating mask splines is almost a Bézier curve, but with some differences. The curve needed to support feathering in a way that stuck to the curve as you edited it, for ease of editing an animation. These are called S-Curves.

Besides the handles, every control point also has points that define the feather between the current point and the next point on the spline. Each feather point is stored in UV space, where U means position across spline segment, and V means distance between main spline and feather points.

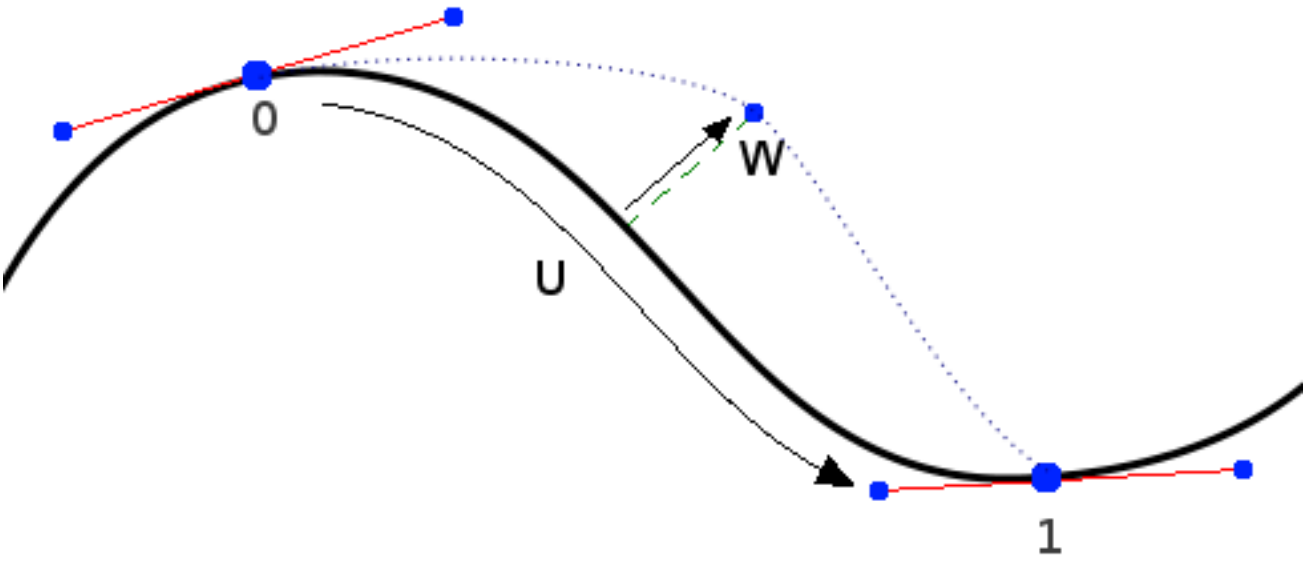


Fig. 2.285: S- Curve Explained.

This allows for deforming the main spline in almost any way, and the feather will be updated automatically to reflect that change. For example if there is just rotation of the spline, feather would stay completely unchanged. If one point's feather is moved, the other feathers will be automatically stretched uniformly along that segment and the overall shape will be almost the same as artists would want it to be.

Control Points

Editing of mask splines happens in a way similar to editing Bézier curves or paths in GIMP or other curve editors: control points are added to define the spline itself, and handles of different types are used to create smooth bends. This makes it possible to define a mask with few points to easily follow an object in footage.

- `Ctrl-LMB` is used to place new control points and define handle orientations.
- `Alt-C` to close the mask by joining the last control point to the first.

- Existing control points can be translated, scaled and rotated with the usual G, S, R shortcuts.

- X or Delete removes control points.

Selection

The usual selection and hide/reveal tools are available:

- A toggle select all.
- B , C border and circle Select.
- Ctrl-L select linked from selection, L: select linked with mouse.
- Ctrl-Alt-LMB lasso select.
- H hide selected, Shift-H hide unselected, Alt-H reveal.

Curve Handles

- Alt-C cycle toggle spline, to create a close curve or open it again.
- V set handle type for selected spline points.
- Ctrl-N make normals (handle directions) consistent.

- Switch Direction handle directions in/out.

Feather

It is possible to control feather of mask, including a way to define non-linear feather. Linear feather is controlled by a slider, non-linear feather is controlled in the same curve-based way to define feather falloff.

- Shift-LMB is used to define a feathering outline curve. To create an initial feather, sliding from a spline control point outside or inside will create and position feather points. After this Shift-LMB will insert new feather point and mouse sliding can be used to move them around.
- Alt-S will scale the feather size.

Animating

Masks can be driven over the time so that they follow some object from the footage, e.g. a running actor. This animation can be done in several ways:

- Control points can be parented to motion tracks. This way is the main way to interact with masks in a motion tracking workflow.
- Keyframe animation of control points using a shape keying system. This can be useful when there are not enough good feature points to track in the footage, or the mask is not based on footage.

  For animation more complex mask shapes, it is also possible to do more high level animation:

- Splines and mask layers can be animated as a whole, instead of individual control points.
- Masks can be parented to motion tracking data. Works for both individual mask point parenting and for overall spline. To select motion track to be parented to use Ctrl-RMB. To parent selected mask points to active motion track use Ctrl-P.

- Mask animation timing can be edited from the Dope Sheet. Here there is a mask mode where mask keyframes can be selected and edited.

Shape Keys

Masks can be animated with shape keyframing. This works on the level of mask layers, so inserting a shape key will keyframe all the splines and points contained in it.

  - `I` will insert a shape key for the active mask layer at the current frame
  - `Alt-I` will clear the shape key for the active mask layer at the current frame.

- Feather Reset Animation: Resets the feather offset across all animated frames.
- Re-Key Points of Selected Shapes: Re-interpolate selected points on across the range of keys selected in the dope sheet.

Video Sequence Editor

Introduction

In addition to modeling and animation, Blender can be used to edit video. There are two possible methods for this one being the *Compositor*. However, this chapter is on the other, the Video Sequence Editor (VSE) and some time shorten to just Sequencer. The Sequencer within Blender is a complete video editing system that allows you to combine multiple video channels and add effects to them. You can use these effects to create powerful video edits (especially when you combine it with the animation power of Blender!).

To use the VSE, you load multiple video clips and lay them end-to-end (or in some cases, overlay them), inserting fades and transitions to link the end of one clip to the beginning of another. Finally, you can audio and synchronize the timing of the video sequence to match it.

The Video Sequence Editor has a header (where the menu and view modes are shown) and a workspace, and works in one of several view modes. The Marker menu allows you to add markers in the VSE. Markers are shared across animation editors. See *Markers*

The sequencer workspace is horizontally striped into channels and each video strip will go in a horizontal channel. Each channel is numbered on the left-hand side, starting from zero and going up.

Note: The first channel 0 is unusable as a place to put strips. This is because it is used by the *Sequencer Display* to show a composite of all strips above channel 0.

Stripes toward the bottom are more dominant, which we will get to in a minute. In the x direction, seconds of animation or frames of animation, `Ctrl-T` to choose, are used as the measure of time (seconds 1 through 7 are shown). You can scale the time using the zoom keys or mouse actions (see the Reference for more info).

Fig. 2.286: Default Video Editing screen layout.

Fig. 2.287: Video Sequencer Header.

Note: By default the Sequencer is enabled however, it can be disabled in the *Post Processing Panel*.

Navigating

Header

View Menu

As usual, the View Menu controls what and how you view in the workspace.

View all Sequences `Home` Zooms the display to show all strips.

View Selected `NumpadPeriod` Zooms in the display to fit only the selected strips.

View Frame `Numpad0` ToDo.

Fit preview in window `Home` ToDo.

Zoom 1:1 `Numpad1` Resizes preview to a 1:1 scale (actual size).

Show Seconds `Ctrl-T` Displays the time instead of the frame number, in the Frame Number Indicator.

Show Frame Number Indicator Toggles the units of measure across the bottom of the workspace between seconds or frames.

Sync Markers Transform Markers as well as Strips.

View Types

The icons in the header allow to change the view of the VSE. By default, only the sequencer is displayed. The second button displays only the Preview region, and the third button displays both the Sequencer and the Preview.

When the preview is enabled, you have several options to change what type pf preview to display. They are explained in the *Display Modes Page*.

Refresh View

Certain operations, like moving an object in 3D View, may not force the *Sequencer* to call for a refresh of the rendered image (since the movement may not affect the rendered image). If an image or video, used as a strip, is changed by some application outside of Blender, Blender has no real way of being notified from your operating system. To force Blender to re-read in files, and to force a re-render of the 3D View, click the *Refresh* button to force Blender to clear all cached images and compute the current frame.

Main View

Adjusting the View

Use these shortcuts to adjust the sequence area of the VSE: Pan `MMB` Zoom `Wheel` Vertical Scroll use `Shift-Wheel`, or drag on the left scroll bar. Horizontal Scroll use `Ctrl-Wheel`, or drag on the lower scroll bar. Scale View Vertically, drag on the circles on the vertical scroll bar. Scale View Horizontally, drag on the circles on the horizontal scroll bar.

Scrubbing

To move back and forth through your movie, use the Timeline editor. `LMB` click and drag left/right in the Timeline editor, moving the vertical bar which indicates the current frame. As you do, the image for that frame is displayed in the VSE editor.

When you `LMB` directly on a sequence strip, this will show the strip *solo*, (temporarily disregarding effects and other strips, showing only this strips output).

Real-time scrubbing and image display is possible on reasonable computers when viewing an image sequence or movie (avi/mov) file.

Scene strips can use OpenGL previews or proxies for realtime playback, otherwise displaying rendered frame is supported, but typically too slow for real-time playback.

Selecting

The Select Menu helps you select strips in different ways.

Strips to the Left Select all strips to the left of the currently selected strip.

Strips to the Right Select all strips to the right of the currently selected strip.

Select Linked Time **`Ctrl-RMB`** Selects the strip under the cursor as well as all strips with the same start/end.

Select Surrounding Handles **`Alt-RMB`** Selects the strip under the cursor as well as the handles of neighboring strips.

Note: Select with this method to move a strip that is between to others without affecting the selected strip's length.

Select Both Handles **`Alt-RMB`** Select the handle under the cursor as well as the handles of the adjacent strip.

Note: Select with this method when you want to change the timing of a cut.

Linked **`L`** Select all strips linked to the currently selected strip

Select All `A` Selects all the strips loaded.

Select Inverse `Ctrl-I` Inverts the current selection.

Border Select `B` Begins the *Box* mode select process. Click and drag a rectangular lasso around a region of strips in your Sequence workspace. When you release the mouse button, the additional strips will be selected.

Frame

Set Preview Range TODO.

Clear Preview Range TODO.

Jump to end of strip `PageUp` Current frame will jump to end of strip.

Jump to beginning of strip `PageDown` Current frame will jump to beginning of strip.

Editing

Moving and Modifying Strips

`G` Moves the selected strip(s) in time or in channels. Move your mouse horizontally (left/right) to change the strip's position in time. Move vertically (up/down) to change channels.

- To snap while dragging hold `Ctrl`
- To "ripple edit" (Make room for strips you drag) hold `Alt` when placing a strip.

If you have added a strip by mistake or no longer want it, delete it by pressing *X* or using this menu option.

Duplicate a strip to make an unlinked copy; drag it to a time and channel, and drop it by `LMB` click.

The Strip Menu contains additional tools for working with strips:

- *Grab/Move*
- *Grab/Extend from Frame*
- *Cut (hard) at frame*
- *Cut (soft) at frame*
- *Separate Images*
- *Deinterlace Movies*
- *Duplicate Strips*
- *Erase Strips*
- *Set Render Size*
- *Make Meta Strip*
- *UnMeta Strip*
- *Reload Strips*
- *Reassign Inputs*
- *Swap Inputs*

- *Lock Strips*
- *UnLock Strips*
- *Mute Strips*
- *Un-Mute Strips*
- *Mute Deselected Strips*
- *Snap Strips*
- *Swap Strips*

Snap to Frame

`Shift-S` Position your cursor (vertical green line) to the time you want. Snap to current frame to start a strip exactly at the beginning of the frame. If your Time display is in seconds, you can get to fractional parts of a second by zooming the display; you can get all the way down to an individual frame.

Separate Images to Strips

`Y` Converts the strip into multiple strips, one strip for each frame. Very useful for slide shows and other cases where you want to bring in a set on non-continuous images.

Editing Strips

- The *entire* strip could be selected by clicking `RMB` in the middle of the strip; holding it down (or pressing `G` rab) and then moving the mouse drags a strip around.
- The *start frame offset* for that strip could be selected by clicking `RMB` on the left arrow of the strip; holding it down (or pressing `G` rab and then moving the mouse left/right changes the start frame within the strip by the number of frames you move it:

– If you have a 20-image sequence strip, and drag the left arrow to the right by 10 frames, the strip will start at image 11 (images 1 to 10 will be skipped). Use this to clip off a rollup or useless lead-in.

– Dragging the left arrow left will create a lead-in (copies) of the first frame for as many frames as you drag it. Use this when you want some frames for transitions to the this clip.

- The *end frame* of the strip could be selected by clicking `RMB` on the right arrow of the strip; holding it down (or pressing `G` rab) and then moving the mouse changes the ending frame within the strip:

– Dragging the right arrow to the left shortens the clip; any original images at the tail are ignored. Use this to quickly clip off a rolldown.

– Dragging the right arrow right extends the clip. For movies and images sequences, more of the animation is used until exhausted. Extending a clip beyond its end results in Blender making a copy of the last image. Use this for transitions out of this clip.

Note: Multiple selection

You can select several (handles of) strips by Shift−RMB clicking: when you press G, everything that is selected will move with your mouse- this means that, for example, you can at the same time move a strip, shorten two others, and extend a forth one.

- Strip Extend

  With a number of strips selected, pressing E lets you interactively extend the strips. This is is similar to grabbing but is useful for extending (or shortening) time around the current frame.

  All selected strip handles to the "mouse side" of the current frame indicator will transform together, so you can change the duration of the current frame.

  While splicing two strips happens just by placing them finish-to-start, cut a strip by pressing K to cut. At the selected frame for the selected strips, K cuts them in two. Use Cut to trim off roll-ups or lead-ins, or roll-downs or extra film shot.

Note: Note on the *Cut*

When you cut a strip, you do not really make a cut like it cutting a real of film. In fact, you make a copy of the strip: the end of the original one is "winded" to the cut point, as with the beginning of the new copy.

For example, imagine that you have a strip of 50 frames, and that you want to delete the first ten ones. You have to go to frame 11, and press K; the cut divides your strip in two parts. You now can select the first small part (frame 1 to frame 10), and delete it press X.

You might think that you have really erased the frames (1 to 10), but there are still there, winded, as in a film reel, under your frame 11: you just have deleted one of the two copies of your strip created by the cut. And you can at any time get your lost frames back (just RMB -click on the left arrow of the strip, then G grab it to the left to display the desired number of frames again (or to the right to hid more frames – this is another way to remove frames at the beginning/end of a strip!).

This is at the heart of nearly every editor solution, and that is quite handy!

Note: Action Stops

When extending the start beyond the beginning or end after the ending, keep in mind that only the last image copies, so when viewed, action will stop on that frame. Start your transition (fade, cross) a little early while action is still happening so that the stop action is not that noticeable.

Change the length of an effect strip by changing the start/end frame of the origin strips.

Copy and Paste

You can copy a clip and paste it using the two header buttons.

Strips

Introduction

The Add menu is the main menu you will be using to add content to the VSE. In general, you load up your strips, create strips of special transition effects, and then animate out your sequence by selecting "Do Sequence" and clicking the *Animation* button. You can use the Add menu in the header, or hover your mouse cursor over the Sequence workspace and press `Shift-A`.

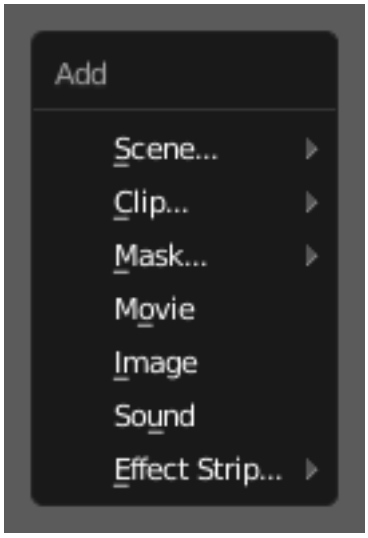


Fig. 2.288: The Add Menu.

Note: Clips can be Huge

A three minute quicktime `.mov` file can be 140Megs. Loading it, even over a high-speed LAN can take some time. Do not assume your computer or Blender has locked up if nothing happens for awhile.

First, let us add a clip:

- A movie clip in the Audio-Video Interleaved format (`*.avi` file).
- A movie clip in the Apple QuickTime format (`*.mov`).
- A single still image to be repeated for a number of frames (`*.jpg`, `*.png`, etc.).
- A numbered sequence of images (`*-0001.jpg`, `*-0002.jpg`, `*-0003.jpg`, etc, of any image format).
- One or more images from a directory.
- A Scene in your blend-file.

Blender does not care which of these you use; you can freely mix and match any of them. They all become a color-coded strip in the VSE:

- Blue is used for Avi/mov codec strips.
- Grey is a single image that is repeated/copied.
- Purple is an image sequences or group of images played one after the other.
- Green is an Audio track.

When you choose to add one of these, the VSE editor will switch to a file browser for you to select what you want to add. Supported files have a little rectangle next to their name (blue for images, green for clips) as a visual cue that you can pick them successfully:

Strip Properties

Edit Strip Panel

The *Edit Strip* panel is used to control placement and properties of strips.

Name You can name or rename your strips here.

Type Displays the type of strip selected.

Blend Mode Controls how the strip affects other strips. See *Color Blend Modes* for details on each blending mode.

Opacity Set the opacity of the strip.

Mute Hides the strip so that it does not participate in the final image computation

Lock Prevents the strip from being moved.

Channel Changes the channel number, or row, of the strip.

Start Frame Changes the starting frame number of the strip, which is the same as grabbing and moving the strip.

> **Tip:** When you add a strip, just drop the strip and then use *Start Frame* to place it at the desired frame. This helpful when it is hard to drag and drop in exactly the right place.

Length Specify the number of frames to use for the strip.

Strip Input Panel

The Strip Input panel is used to controls the duration of the strip along with some basic transforms.

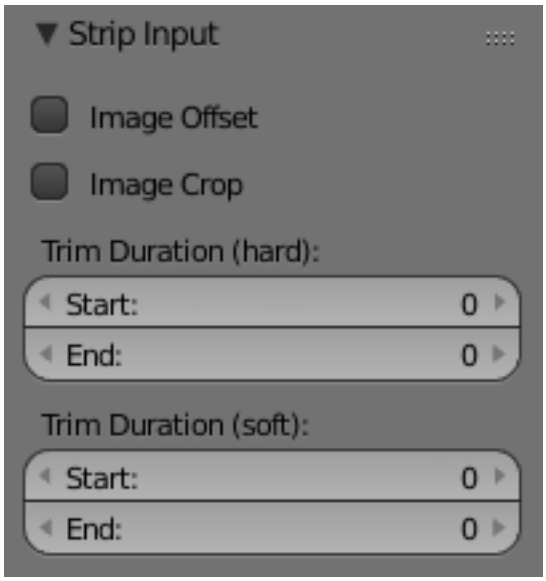


Fig. 2.289: Strip Input Settings.

Image Offset Used to transform the strip by moving it in the X and Y direction.

Image Crop Used to crop the strip by stretching the image, use *Top*, *Left*, *Bottom*, and *Right* to control which part of the image is cropped.

Trim Duration (hard) Controls at what frame the source of the strip starts and ends at.

Trim Duration (soft) Can be used to either extend the strip beyond the end frame by repeating the last frame. Or it can be used to shorten the strip, as if you were cropping the end frame. This is the same has adjusting the strip handles.

Filter Panel

Enables you to quickly set common image pre-processing options.

Strobe To display only a defined number of images. For example, if you set this to 10, the strip will only display frames 1, 11, 21, 31, 41... of the source.

Flip X flips (reverses) the image left-to-right, Y reverses top-to-bottom.

Backwards Reverses strip image sequence

De-Interlace Removes fields in a video file.

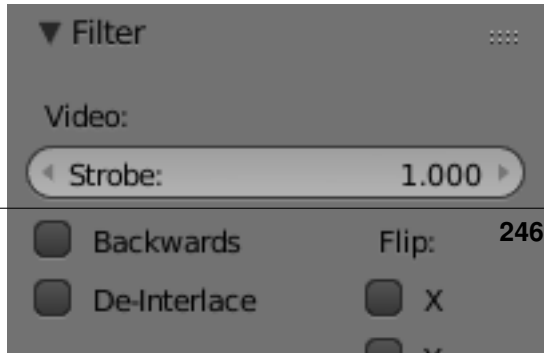
Saturation Increase or decrease the saturation of an image.

Multiply Multiplies the colors by this value.

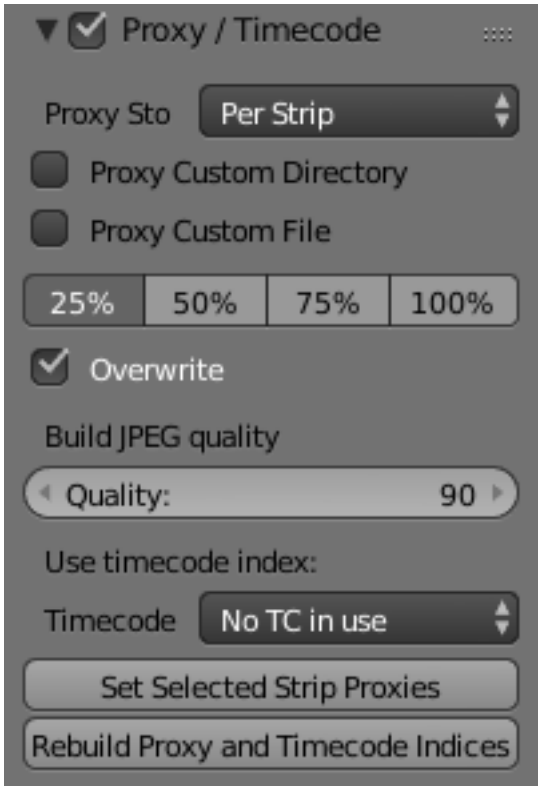
Convert Float Converts input to float data.

Proxy/Timecode Panel

Once you have chosen the Proxy/Timecode parameters, you need to use *Strip* → *Rebuild Proxy and Timecode indices* to generate the proxy clip and it will be available after Blender makes it.



Proxy



A proxy is a smaller image (faster to load) that stands in for the main image. When you Rebuild proxy Blender computes small images (like thumbnails) for the big images and may take some time. After computing them, though, editing functions like scrubbing and scrolling and compositing functions like cross using these proxies is much faster but gives a low-res result. Disable proxies before final rendering.

In order to actually use the proxies, the proper Proxy Render Size selector value must be selected in the Properties region of the Sequencer View (where the edit plays back).

Proxy Storage Defines whether the proxies are for individual strips or the entire sequence.

Per Strip Proxies are stored in the directory of the input.

Proxy Custom Directory By default, all generated proxy images are storing to the <path of original footage> /BL_proxy/<clip name> folder, but this location can be set by hand using this option.

Proxy Custom File Allows you to use pre-existing proxies.

Project All proxies are stored in one directory.

Proxy Directory The location to to store the proxies for the project.

Proxy Size Buttons to control how big the proxies are. The available options are 25%, 50%, 75%, 100 percent of original strip size.

Overwrite Saves over any existing proxies in the proxy storage directory.

Quality Defines the quality of the JPEG images used for proxies.

Timecode See *Timecode*.

Set Selected Strip Proxies Same as choosing the *Proxy Size* and *Overwrite*.

Rebuild Proxy and Timecode Indices Generates Proxies and Timecodes, same as doing *Strip* → *Rebuild Proxy and Timecode indices*.

Timecode

When you are working with footage directly copied from a camera without pre-processing it, there might be bunch of artifacts, mostly due to seeking a given frame in sequence. This happens because such footage usually does not have correct frame rate values in their headers. So, for Blender to calculate the position of a needed frame in the stream works inaccurately and can give errant result. There are two possible ways to avoid this:

- Preprocess your video with, say, mencoder to repair file header and insert correct keyframes.
- Use Proxy/Timecode option in Blender.

Options

Timecode Timecode to use on the selected movie strip.

The following timecodes are supported:

- No TC in use- do not use any timecode
- Record Run
- Free Run
- Free Run (rec date)
- Record Run No Gaps

Note: Record Run is the timecode which usually is best to use, but if the clip's file is totally damaged, *Record Run No Gaps* will be the only chance of getting acceptable result.

Modifiers Panel

Modifiers are used to make adjustments on the image, like contrast, brightness, saturation, color balance and applying masks.

You can add these modifiers directly to the selected strip, or you can use it within an "Adjustment Layer" effect strip, which allows you to apply these modifiers onto several strips the same time.

Use Linear Modifiers Calculate modifiers in linear space instead of sequencer space.

Copy to Selected Strips Allows you to copy the modifiers to selected strips. This works two ways, you can either replace the old modifiers or append/add to the previous modifiers.

Each modifiers have several buttons at their top:

- The "eye" is to disable the modifier. Very useful to compare the image, with / without modifications.
- The next two buttons (up and down arrows) are used to change the modifier's position in the stack.
- The cross is to delete the modifier from the stack.

Strip Use this to apply the modification on the whole image, or to use another strip's image (with alpha channel) for masking the modifier (and only this modifier), by choosing it in the "Mask" select menu.

Mask This allows you to choose a Mask created in the Mask editor which will limit the modification to the masked image's zones.

Currently, the following modifiers are supported:

Color Balance Color balance adjustments, through Lift, Gamma, and Gain.

Note: This modifier works the same as the *Color Balance Node*

Curves C/RGB curves.

Note: This modifier works the same as the *Curves Node*

Hue Correct HSV multi points curves.

Note: This modifier works the same as the *Curves Node*

Bright/Contrast Adjusts the brightness and contrast of the modifier input.

Mask Use it for masking the other modifiers in the stack which are below.

For example, to correct the brightness only on a certain zone of the image, you can filter the Bright/Contrast modifier by placing a Mask modifier, just before it in the stack. You can choose to use a Mask created in the Mask editor, or to use another strip as a mask (the image of this strip must have an alpha channel). This mask will be applied on all the others modifiers below it in the stack.

White Balance Use it to adjust the white balance by choosing the color that should be white.

Tone Map Used to map one set of colors to another in order to approximate the appearance of high dynamic range images in a medium that has a more limited dynamic range.

Note: This modifier works the same as the *Tone Map Node*

Types

Scene Strip

Scene strips are a way to insert the render output of a scene into your sequence. Instead of rendering out a video, then inserting the video file, you can insert the scene directly.

The strip length will be determined based on the animation settings in that scene.

Use Sequence Expand the scenes sequence strips, allowing one scene to re-use another scenes edit, (instead of taking the render output from the scene).

This is similar to how *Meta Strips* work, with the added advantage of supporting multiple instances of the same data.

Camera Override This can be used to override the scenes camera with any other object.

It is useful to support switching views within a single scene.

Show Grease Pencil Shows *Grease Pencil* in OpenGL preview.

Audio Volume Volume of the audio taken from the chosen scene.

Hint: Its best not add a scene strip for the scene you are currently editing. While this is supported, it can be confusing when changing the start and end frame.

Mask Strip

A Mask Strip is used to select a mask data-block generated from *Movie Clip Editor*. This works similar to the *Mask Node* but without the options available for finer control. The mask image is always generated at the render resolution, scaling along with different proxy levels.

Image and Movie Strips

When adding a Movie or Movie with Audio `LMB` to put the name of the file into the text field at the top; this selects a *single* file (like a movie)

In the case of (numbered) *image sequences*, you have a choice:

Directory `RMB` right-click on a directory name, and all files in that directory will be brought in as part of the image, in sort order, one image per frame.

Range Navigate into the directory and right-click and drag over a range of names to highlight multiple files. You can page down and continue right-click-dragging to add more to the selection

Batch Shift-right-click selected non-related stills for batch processing; each image will be one frame, in sort order, and can be a mix of file types (jpg, png, exr, etc.)

All Press `A` to select/deselect All files in the directory.

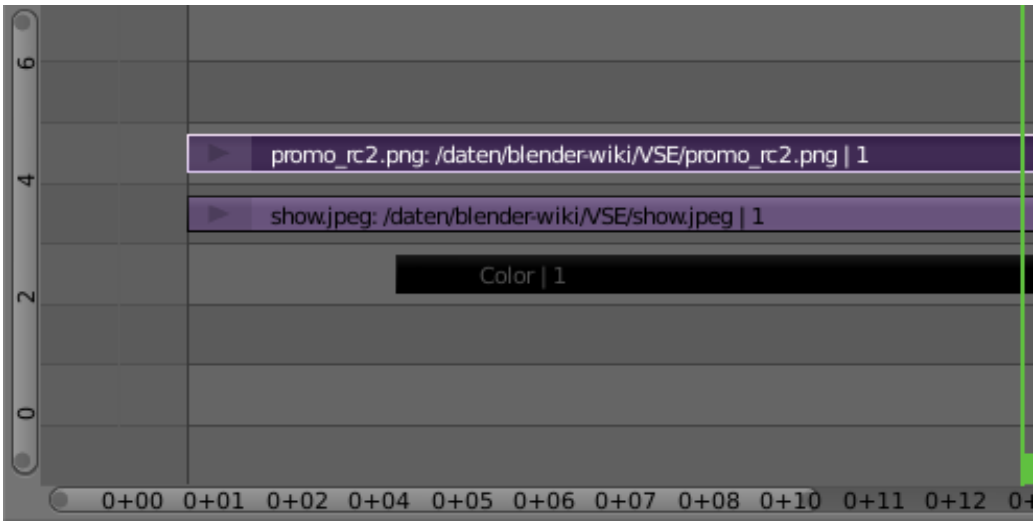
When you click the *Select ...* button, the area will switch back to VSE, and the strip will be rubber-banded to your mouse. You cannot load multiple movies at the same time by right-clicking them; no movies load if you right click them. Right-clicking only works for images.

In order to add items to the VSE, left-click for movies, left-click for single images, or right-click and drag for image sequences. Move your mouse to the frame/time and stripe you want, and click to break the rubberband and drop the strip in place (in a channel and starting at a frame).

When you add an image, Blender makes it into a 50-frame strip, which means that image will be in your video for two seconds (at 25 fps PAL). Aside from re-positioning it, you will want to scale it by `RMB` on either the start or end arrow, and dragging left or right. As you move, the frame number updates to say where the arrow is. Use `LMB` to validate, or `RMB` to cancel the modification.

Tip: Dealing with Different Sizes

Dealing with different sized images and different sized outputs is tricky. If you have a mis-match between the size of the input image and the render output size, the VSE will try to auto-scale the image to fit it entirely in the output. This may result in clipping. If you do not want that, use *Crop* and/or *Offset* in the Input panel to move and select a region of the image within the output. When you use *Crop* or *Offset*, the auto-scaling will be disabled and you can manually re-scale by adding the Transform effect.



If you scroll up the workspace, you will see an information channel (at vertical location channel 0) that gives you some helpful hints about the active strip. The example above shows a color strip from frames 1 to 25, then a mov file, and then an image strip. The info channel shows handy information about the image strip, whose name has been scrunched in the strip display, but is clearly spelled out in the information strip.

Effect Strips

Introduction

Blender offers a set of effects that can be added to your sequence. Each effect is explained in the next pages individually, but they all are added and controlled in the same way. To add an effect strip, select one base strip (image, movie, or scene) by `RMB` clicking on it. For some effects, like the Cross transition effect, you will need to `Shift-RMB` a second overlapping strip (it depends on the effect you want). Then select *Add* → *Effect* and pick the effect you want from the pop-up menu. When you do, the Effect strip will be shown above the source strips. If it is an independent effect, like the *Color Generator*, it will be rubberbanded to your mouse; click to drop the strip.

Note: Since most Effects strips depend on one or two source strips, their frame location and duration depends on their source strips. Thus, you may not be able to move it; you have to move the source strips in order to affect the effect strip.

To use an effect that combines or makes a transition between (or composites) two strips, you must Box select `B` or shift-right-click two of them. When you add the effect strip, it will be placed in a channel above the two in Grab mode (click to drop it on a channel). Its duration will be the overlap between the two strips as a maximum.

With some effects, like the *Alpha Over*, the order in which you select the strips is important. You can also use one effect strip as the input or source strip with another strip, thus layering effects on top of one another.

Note: The only exception is the *Color Generator* effect. It does not depend on a base strip; you can add and position it independent of any other strip. Change the length as you would any strip.

If you picked the wrong effect from the menu, you can always change it by selecting the strip `RMB` and using the *Strip* → *Change Effect* selection. Or, you can press `C` to switch effects on a selected Effects strip.

Add Effect

Fig. 2.290: Can you hear the thunder?

The Add effect adds two colors together. Red and Cyan (Green and Blue) make White. Red and Blue make "Magenta" (i.e. Purple!). Red and Green make Yellow.

The Add Effect adds the colors of two strips together, Use this effect with a base image strip, and a modifier strip. The modifier strip is either a solid color or a black-and-whte mask, or another image entirely. The example to the right shows what happens when you add gray to an image, and animate the effect over time. The image gets bright because we are adding gray (R:.5, G:.5, B:.5) to say, a blue color (R.1, G:.1, B:.5) resulting in (R:.6, G:.6, B:1.0) which retains the original hue (relationship between the colors) but is much brighter (has a higher value). When applied to the whole image like this, the whole image seems to flash.

You can use this effect to increase the brightness of an image, or if you use a BW mask, selectively increase the brightness of certain areas of the image. The Mix node, in Add mode, does exactly the same thing as the Add sfx strip here, and is controlled the same way by feeding the Factor input.

Adjustment Layer

The Adjustment Layer strip works like a regular input file strip except for the fact, that it considers all strips below it as its input.

Real world use cases, you want to add some last finishing color correction on top of parts of your final sequence, timeline without messing with metastrips around. Just add an adjustment layer on top and activate the color balance.

Orm you can stack a primary color correction and several secondary color correction on top of each other (probably using the new mask input for area selection).

Alpha Over, Under, and Over Drop

Using the alpha (transparency channel), this effect composites a result based on transparent areas of the dominant image. If you use a Scene strip, the areas of the image where there is not anything solid are transparent; they have an alpha value of 0. If you use a movie strip, that movie has an alpha value of 1 (completely opaque).

So, you can use the *Alpha Over* / *Alpha Under* effect to composite the CGI Scene on top of your movie. The result is your model doing whatever as if it was part of the movie. The Factor curve controls how much the foreground is mixed over

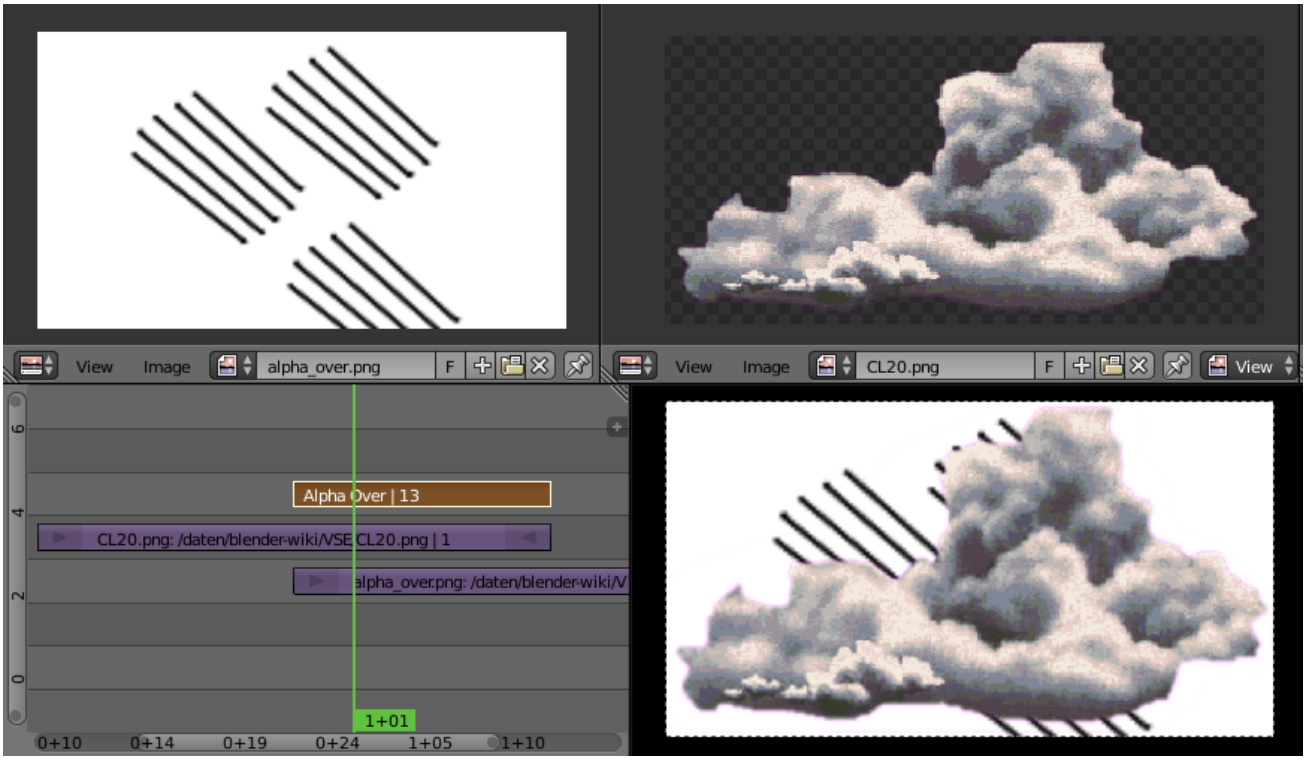


Fig. 2.291: Alpha Over Effect.

the background, fading in the foreground on top of the background. The colors of transparent foreground image areas is ignored and does not change the color of the background.

Select two strips `Shift-RMB`:

- With *Alpha Over*, the strips are layered up in the order selected; the first strip selected is the background, and the second one goes *over* the first one selected. The *Factor* controls the transparency of the *foreground*, i.e. *Factor* of 0.0; will only show the background, and a *Factor* of 1.0 will completely override the background with the foreground (except in the transparent areas of this one, of course!)
- With *Alpha Under*, this is the contrary: the first strip selected is the foreground, and the second one, the background. Moreover, the *Factor* controls the transparency of the *background*, i.e. a *Factor* of 0.0; will only show the foreground (the background is completely transparent), and a *Factor* of 1.0 will give the same results as with *Alpha Over*.
- *Alpha Over Drop* is between the two others: as with *Alpha Under*, the first strip selected will be the foreground, but as with *Alpha Over*, the *Factor* controls the transparency of this foreground.

The example shows layering of Alpha Over effects. The very bottom channel is red, and an arrow is on top of that. Those two are Alpha Over to Channel 3. My favorite toucan is Channel 4, and Channel 5 alpha over composes the toucan on top of the composited red arrow. The last effect added is tied to Channel 0 which will be rendered.

By clicking the Premultiply Alpha button in the properties panel of the foreground strip, the Alpha values of the two strips are not multiplied or added together. Use this effect when adding a foreground strip that has a variable alpha channel (some opaque areas, some transparent, some in between) over a strip that has a flat opaque (Alpha=1.0 or greater) channel. If you notice a glow around your foreground objects, or strange transparent areas of your foreground object when using Alpha Over, enable *Premultiply*. The Alpha Over Drop effect is much like the Cross, but puts preference to the top or second image, giving more of a gradual overlay effect than a blend like the Cross does. Of course, all of the Alpha effects respect the alpha (transparency) channel, whereas Cross does not.

The degree of Alpha applied, and thus color mixing, can be controlled by an F-Curve. Creating a Sine wave could have the effect of the foreground fading in and out.

Gaussian Blur

The Gaussian Blur strip is used to blur the input strip in a defined direction. This can be used to blur a background or to blur a transition strip. For example, in the image below it shows an example of this strip being used to blur a transition. In this set up the *Gaussian Blur Strip* is modifying a *Adjustment Layer Strip* where the curve defines the amount of blur over the length of the *Adjustment Layer Strip*.

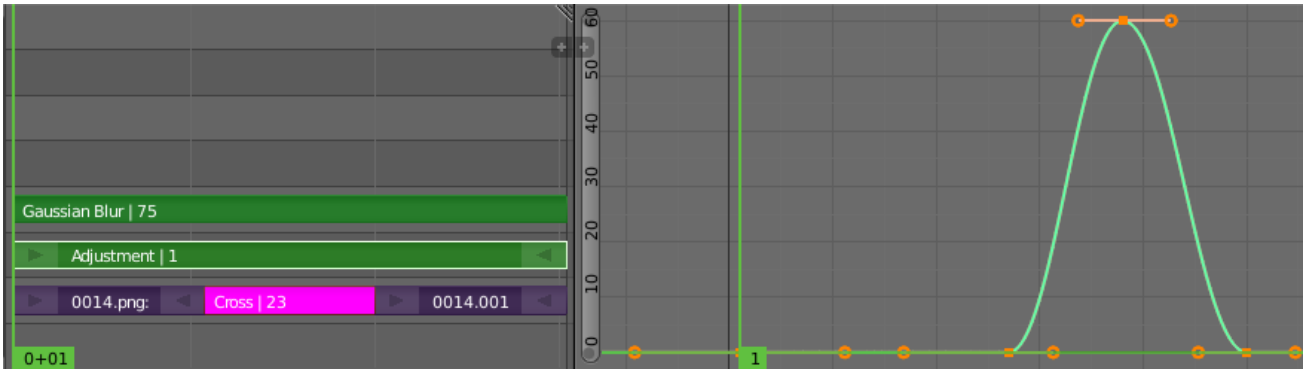


Fig. 2.292: Example of Blurring a Transition.

Options

Size X Distance of the blur effect on the X axis.

Size Y Distance of the blur effect on the X axis.

Color

This effect works by itself to create a color strip. By default, when it is created, it is 50 frames long, but you can extend it by grabbing and moving one of the ends. Click on the color button in the Effect panel under Sequencer buttons, which is under the Scene tab, to pick a different color (by default, it is gray). Use this strip crossed with your main movie to provide a fade-in or fade-out.

Cross and Gamma Cross

This effect fades from one strip to another, based on how many frames the two strips overlap. This is a very useful strip that blends the whole image from one to the other.

Gamma Cross uses color correction in doing the fade, resulting in a smooth transition that is easier on the eye.

Glow

Example of a Glow effect applied to a picture.

Top left Base picture (Lofoten Islands, Norway – source: wikipedia.fr);

Top right Result of the effect;

Bottom left Effect settings;

Bottom right Result with the Only boost button activated.

This effect makes parts of an image glow brighter by working on the luminance channel of an image. The *Glow* is the superposition of the base image and a modified version, where some areas (brighter than the *Threshold:*) are blurred. With the *Glow* strip properties, you control this *Threshold:*, the maximum luminosity that can be added (*Clamp:*), a *Boost*

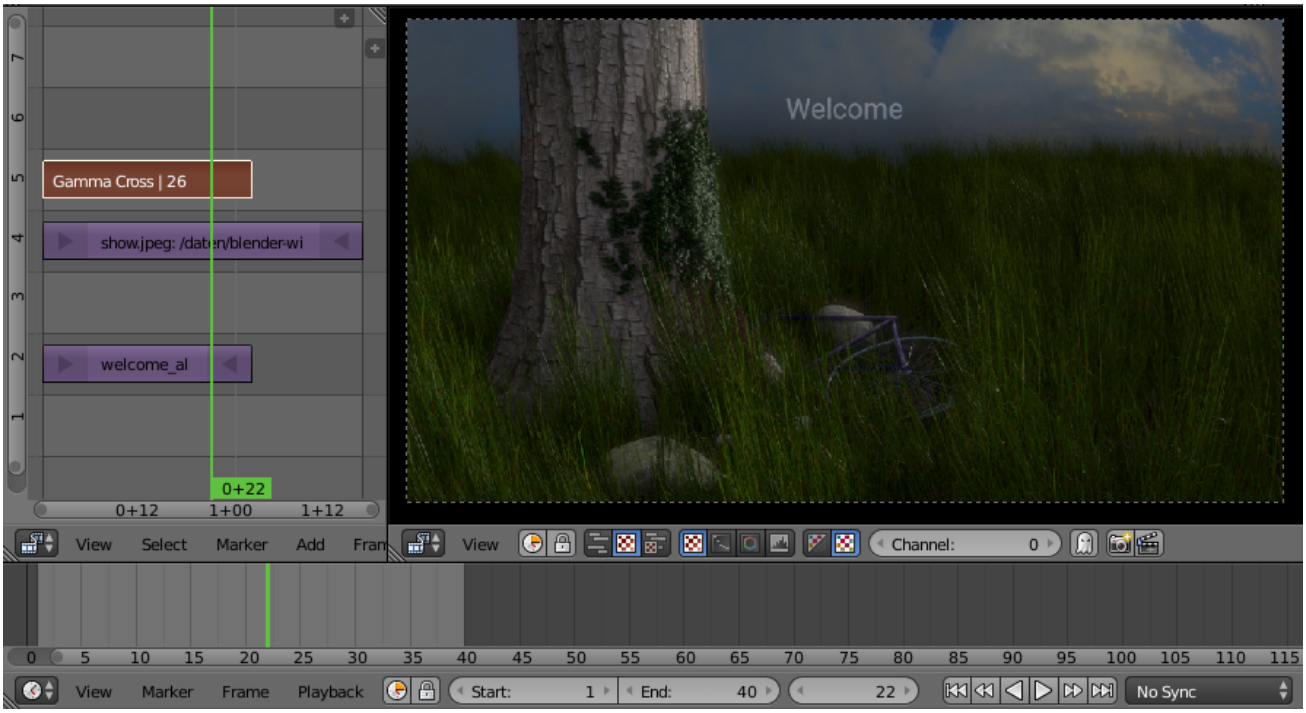


Fig. 2.293: Cross Effect.

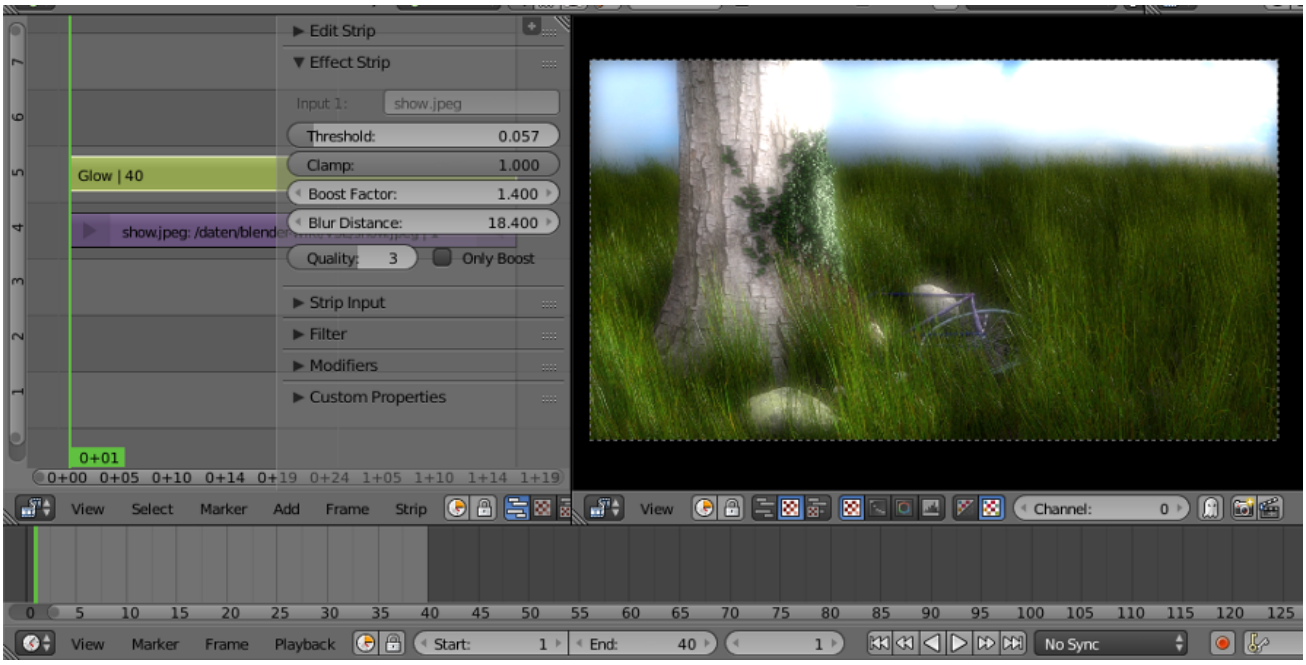


Fig. 2.294: Glow Effect.

factor: for it, the size of the blur (*Blur distance:*), and its *Quality:*. The *Only boost* button allows you to only show/use the "modified" version of the image, without the base one. To "animate" the glow effect, mix it with the base image using the Gamma Cross effect, crossing from the base image to the glowing one.

Multicam Selector

The Multicam Selector strip is used for multi camera editing. Multicam editing is for when you have multiple cameras recording the same scene from different angles. To edit these in the VSE (Video Sequence Editor) can be easy if you do it right.

1. First your going to want to add in each of your video strips.
2. Next make sure to sync them to each other using there audio waveform see the *Audio Docs* or by the movement of objects.
3. If you are using any effects on you strips it may helpful to use *Meta Strips*.
4. Add a viewer region for every input channel and put it into 25% proxy display mode.
5. Add a multicam selector effect strip *above* all the channel tracks.

   After completing these steps you should get something similar to the image below:

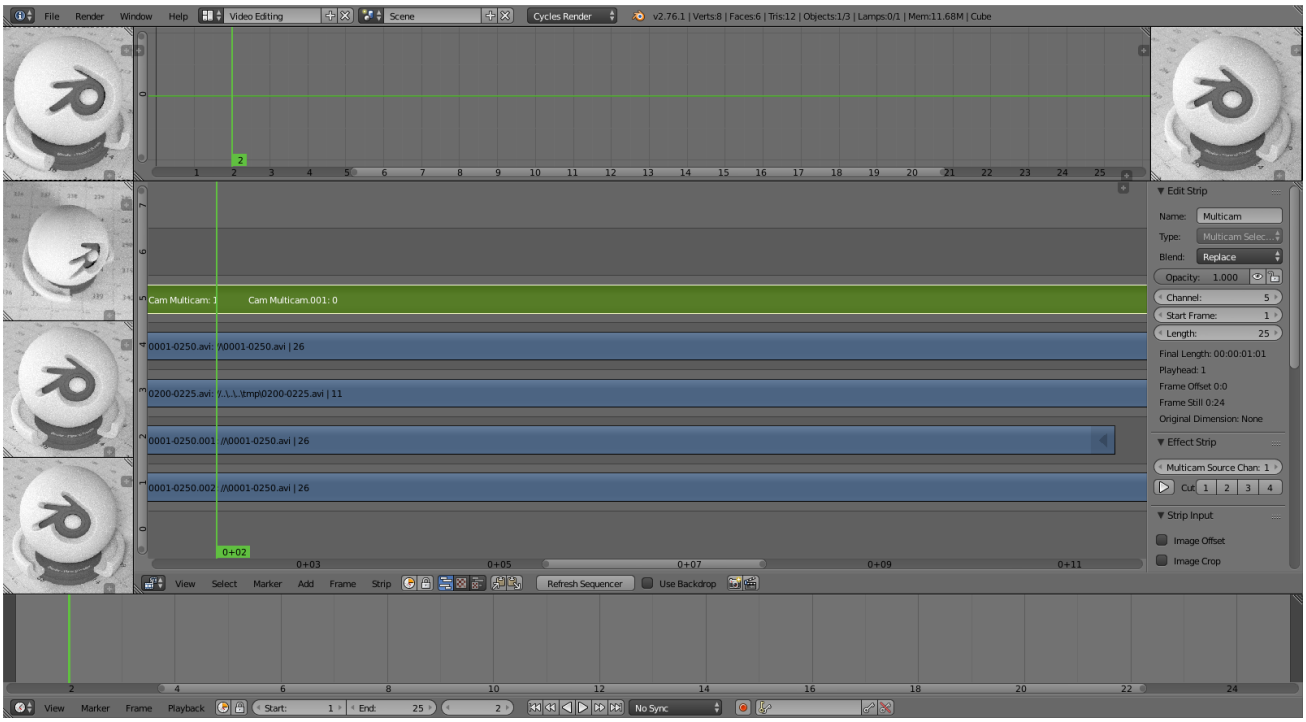


Fig. 2.295: Multi camera editing setup.

6. Now select the multicam strip, if you take a look at the strip options (Properties region), you will notice, that multicam is a rather simple effect strip: It just takes a selected channel as its input. That is all. The magic comes with the convenient keyboard layout.
7. When you select the multicam strip, the keys 1-9 are mapped to a Python handler, that does a cut on the multicam and changes its input.
8. So: you select the multicam strip, you start playback and press the keys for the correct input while watching your show.
9. You will end up with a small multicam selector strip for every cut.

In reality, it boils down to: watch a few seconds to see, what is coming, watch it again and do a rough cut using the number keys, do some fine tuning by selecting the outer handles of two neighboring multicam for A/B rolling.

Multiply

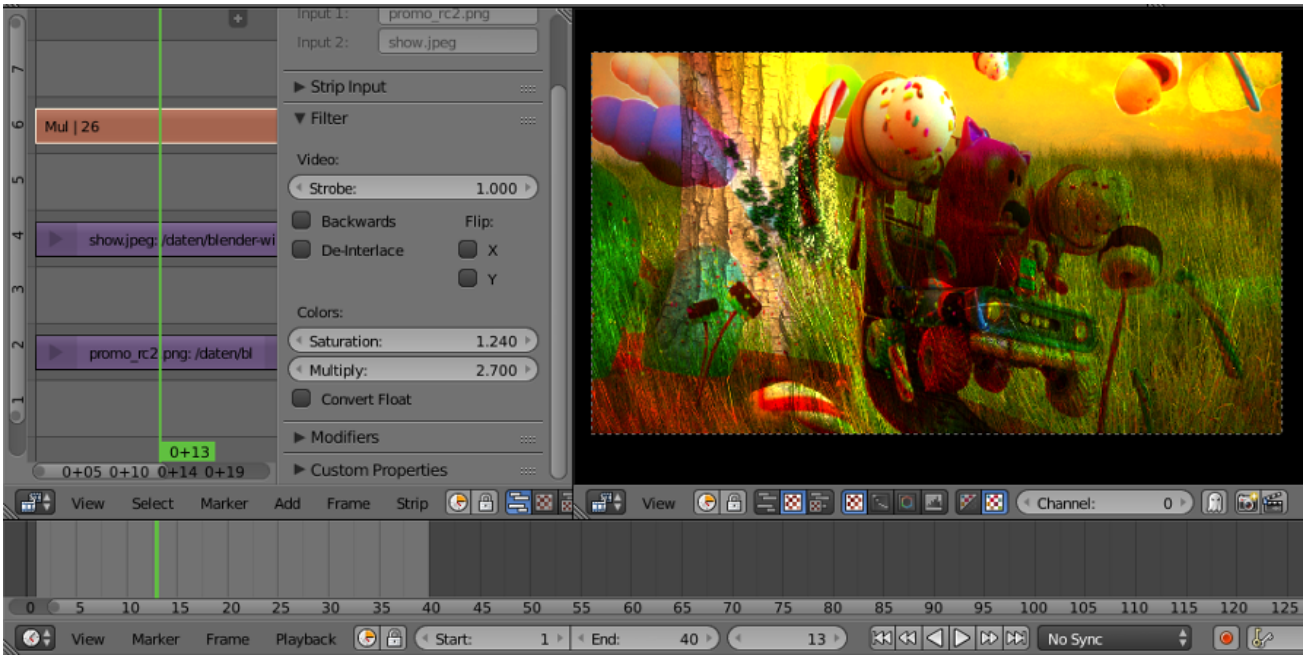


Fig. 2.296: Multiply Effect.

The *Multiply* effect multiplies two colors. Blender uses values between (0.0 to 1.0) for the colors, he does not have to normalize this operation, the multiplication of two terms between (0.0 to 1.0) always gives a result between (0.0 to 1.0).

(with the "traditional" representation of three bytes, like RGB(124, 255, 56) , the multiplications give far too high results, like RGB(7316, 46410, 1848), that have to be, normalized (brought back) by dividing them by 256 to fit in the range of (0 to 255) ...).

This effect has two main usages:

With a mask A mask is a B&W picture witch, after multiplication with a "normal" image, only show this one in the white areas of the mask (everything else is black).

The opening title sequence to James Bond movies, where the camera is looking down the barrel of a gun at James, is a good example of this effect.

With uniform colors Multiplying a color with a "normal" image allows you to soften some hues of this one (and so – symmetrically – to enhance the others).

For example, if you have a brown pixel RGB(0.50, 0.29, 0.05), and you multiply it with a cyan filter (uniform color RGB(0.0, 1.0, 1.0), you will get a color RGB(0.0, 0.29, 0.5). Visually, the result is to kill the reds and bring up (by "symmetry" – the real values remain unchanged!) the blues an greens. Physically, it is the same effect as shining a cyan light onto a chocolate bar. Emotionally, vegetation becomes more lush, water becomes more Caribbean and inviting, skies become friendlier.

Note: This effect reduces the global luminosity of the picture (the result will always be smaller than the smallest operand). If one of the image is all white, the result is the other picture; if one of the image is all black, the result is all black!

Speed Control

Speed Control time-warps the strip, making it play faster or slower than it normally would. A *Global Speed* less than 1.0 makes the strip play slower; greater than 1. 0 makes it play faster. Playing faster means that some frames are skipped, and the strip will run out of frames before the end frame. When the strip runs out of frames to display, it will just keep repeating

the last one; action will appear to freeze. To avoid this, position the next strip under the original at a point where you want motion to continue.

Creating a Slow-Motion Effect

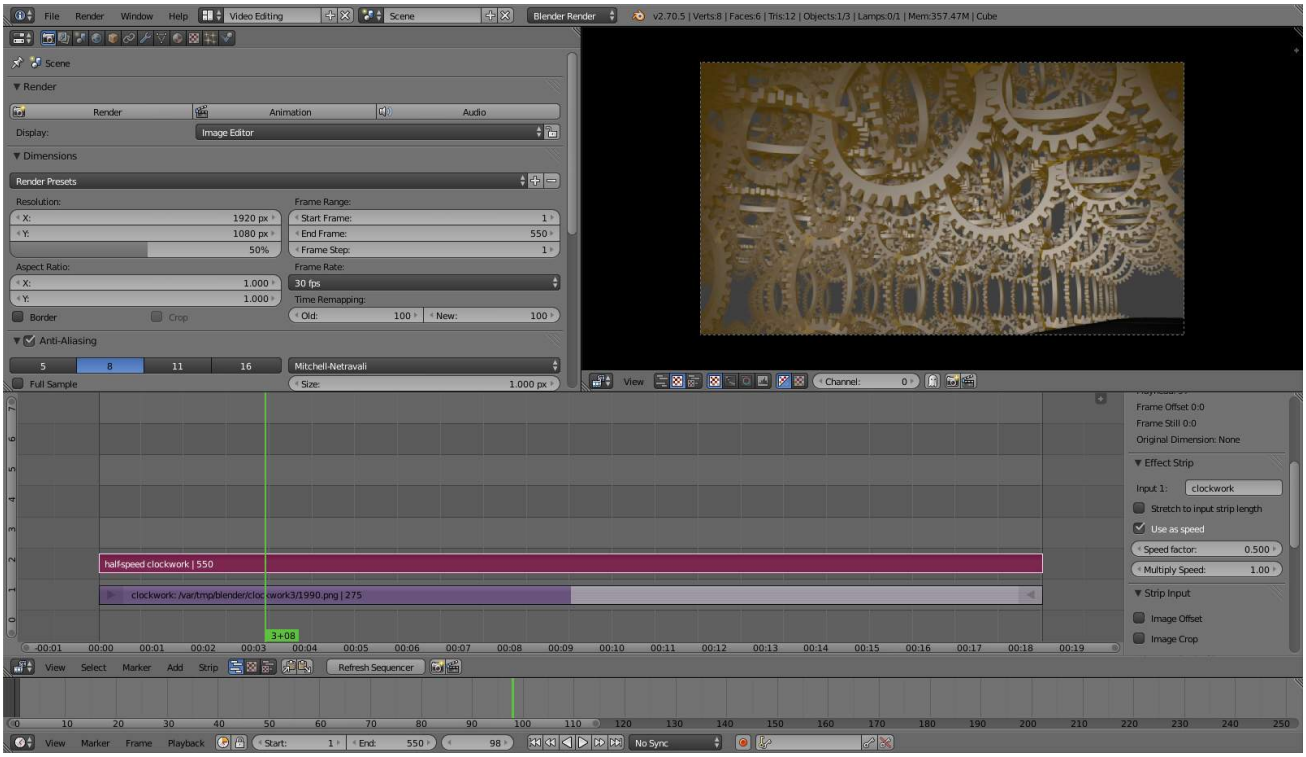


Fig. 2.297: 50% Slow motion using Speed Control.

Suppose you want to slow your strip down. You need to affect the speed of the video clip without affecting the overall frame rate. Select the clip and *Add* → *Effect* → *Speed Control* effect strip. Click to drop it and press N to get the Properties. Uncheck the *Stretch to input strip length* option in the Effect Strip section. Set the Speed factor to be the factor by which you want to adjust the speed. To cut the displayed speed by 50%, enter 0.5 . Now, a 275-frame clip will play at half speed, and thus display only the first 137 frames.

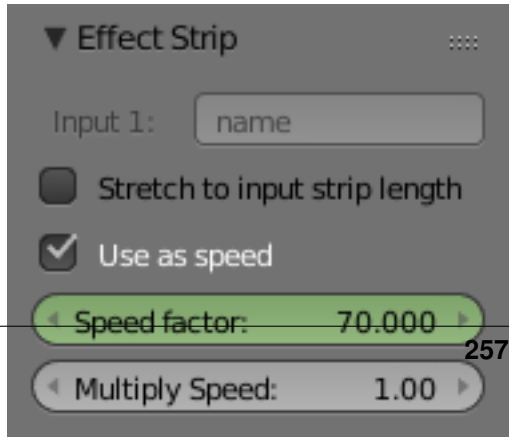
If you want the remaining frames to show in slo-mo after the first set is displayed, double the Length of the source strip (since effects strip bounds are controlled by their source strips). If you are using a speed factor other than 0.5 then use the formula:

```
new_length = real_length / speed_factor
```

That is it, set your render to animate (in this example) all 550 frames.

Keyframing the Speed Control

To get even finer control over your clip timing, you can use curves! While it is possible to keyframe the Speed factor, usually you want to keyframe the Frame number directly.



Uncheck *Stretch to input strip length* and uncheck *Use as speed*. You now have a Frame number button which you can keyframe. If you want the strip to animate **at all** you will have to insert some keyframes, otherwise it will look like a still. In most cases you will want to use the Graph editor view to set the curve interpolation to Linear since the default Bézier will rarely be what you want.

If you do choose to keyframe the Speed factor instead, remember to click the Refresh Sequencer button in the header of the Video Sequence Editor's strip view or your changes will not take effect.

Changing Video Frame Rates

You can use the speed control to change the frames per second (fps), or framerate, of a video. If you are rendering your video to a sequence set, you can effectively increase or decrease the number of individual image files created, by using a Global Speed value less than or greater than one, respectively. For example, if you captured a five-minute video at 30 fps and wanted to transfer that to film, which runs at 24 fps, you would enter a Global Speed of 30/24, or 1.25 (and Enable Frame Blending to give that film blur feel). Instead of producing `5 × 60 × 30 = 9000` frames, Blender would produce `9000/ 1.25 = 7200 = 5 × 60 × 24` frames. In this case, you set a *start* = 1 and *end* = 7200, set your Format output to `jpeg` 30fps, and image files `0001.jpg` through `7200.jpg` would be rendered out, but those images cover the entire 9000 frames. The image file `7200.jpg` is the same a frame 9000. When you read those images back into your film blend-file at 24 fps, the strip will last exactly 5 minutes.

Subtract Effect

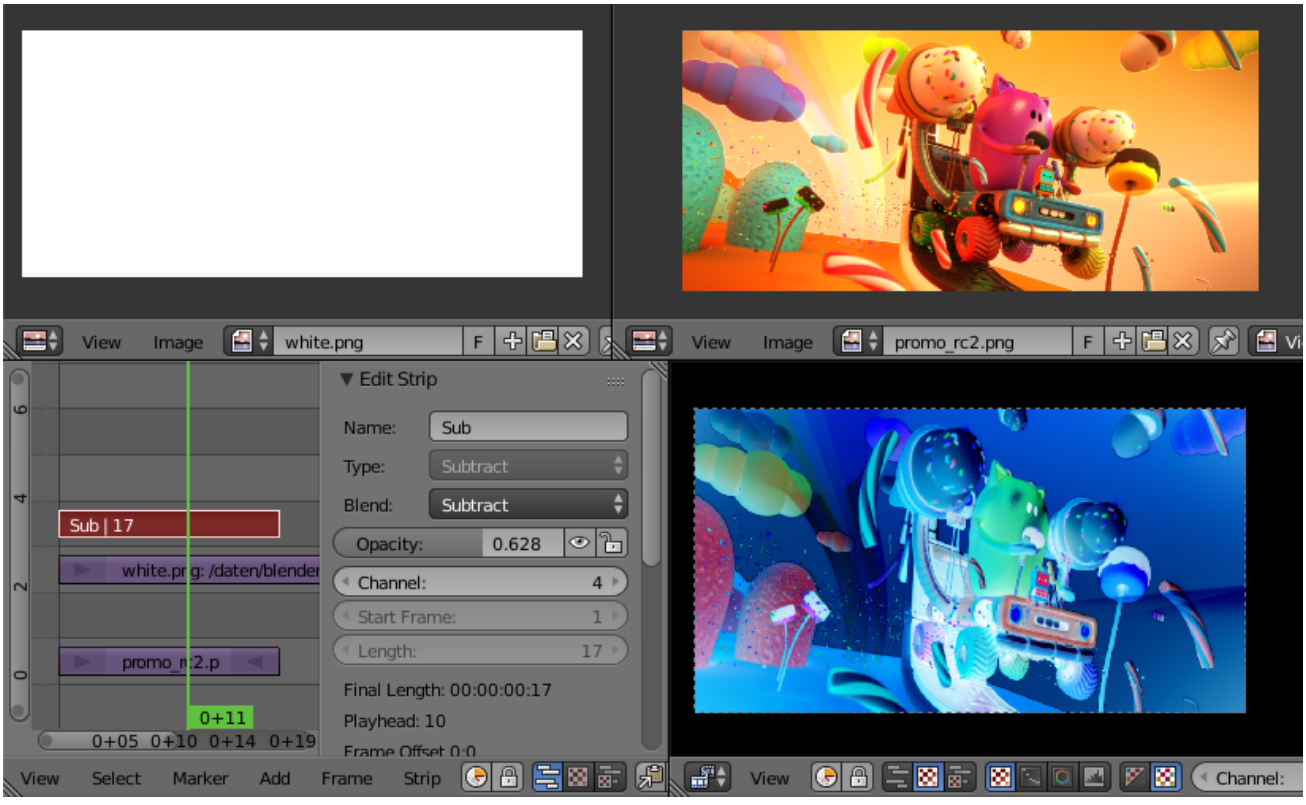


Fig. 2.299: Subtract Effect.

This effect takes away one strip's color from the second. Make a negative of an image using this effect, or switch the order of the strips and just darken the strip. Subtracting a hue of blue from a white image will make it yellow, since red and green make yellow.

Text Effect

The text effect strip allows you to directly displaying text in the sequence editor. The strip will display the text inserted in its text field on the final sequence.

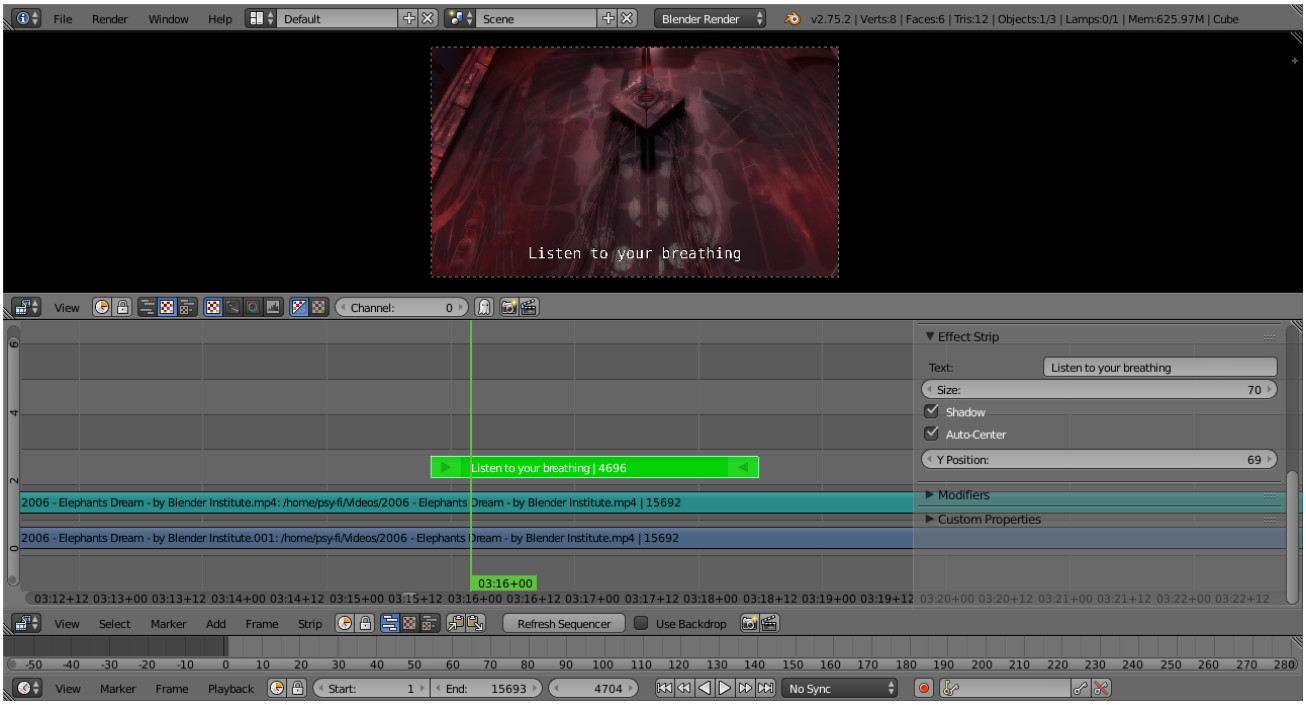


Fig. 2.300: Text Effect.

Options

Text The actual text displayed.

Size Size of the text.

Shadow Creates a shadow under the text.

Auto Center Centers the text on the x axis.

X Position Positions the text on the x axis. Only appears if auto center is off.

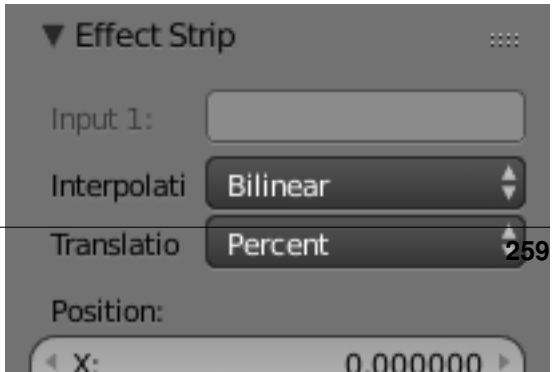
Y Position Positions the text on the y axis.

Export Subtitles

Exporting subtitles in .srt format is also supported. The exported subtitles contain all text strips in the sequence editing.

Transform

Transform is a swiss-army knife of image manipulation. It scales, shifts, and rotates the images within a strip.



Options

Interpolation

None No interpolation, uses nearest neighboring pixel.

Bilinear Simple interpolation between adjacent pixels.

Bicubic Highest quality interpolation.

Translation Unit Control whether the input values are in *Percent* or *Pixels*

Uniform Scale Scale the input evenly along the X and Y axis.

Scale Scale the image on the X and Y axis

Rotation Rotates the input two-dimensionally along the Z axis.

Wipe

The wipe effect is a type of transition strip. It can be used to transition from one strip to the next. The wipe will have no effect if created from a single strip instead of two strips. The duration of the wipe is the intersection of the two source strips and cannot be adjusted. To adjust the start and end of the wipe you must adjust the temporal bounds of the source strips in a way that alters their intersection.

Options

Transition The type of transition used.

Clock Like the hands of an analog clock, it sweeps clockwise or (if Wipe In is enabled) counterclockwise from the 9:00 position. As it sweeps, it reveals the next strip.

Iris Like the iris of a camera or eye, it reveals the next strip through an expanding (or contracting) circle. You can blur the transition, so it looks like ink bleeding through a paper.

Double Similar to *Single* but uses two lines either starting from the middle of the image or the outside. Unlike the other transitions you can control the angle of the line using the angle controls.

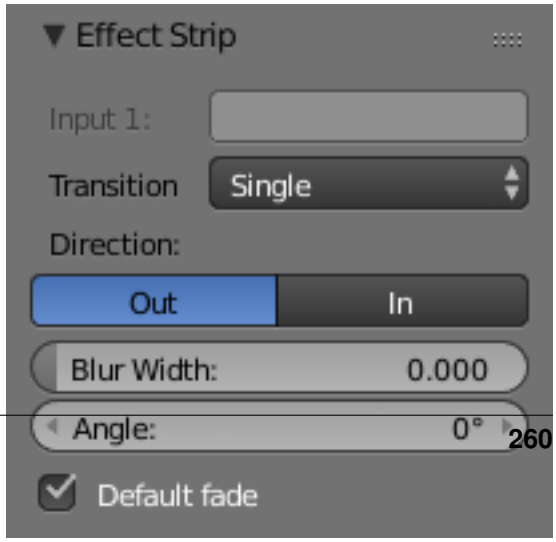
Single Reveals the next strip by uncovering it in a strait line moving across the image. This transition also allows you to control the angle of the transition.

Direction Control whether to fade *In* or *Out*.

Blur Width The width of the blur used to blur the transition.

Sound Strips

As well as images and movies the VSE can also edit audio tracks. You can add `WAV`, `mp3` and other audio formates files from your hard disk as a file, or as encoded within a movie, and mix them using an F-Curve as a volume control.



Options

Pack This allows you to save the audio file into the blend-file.

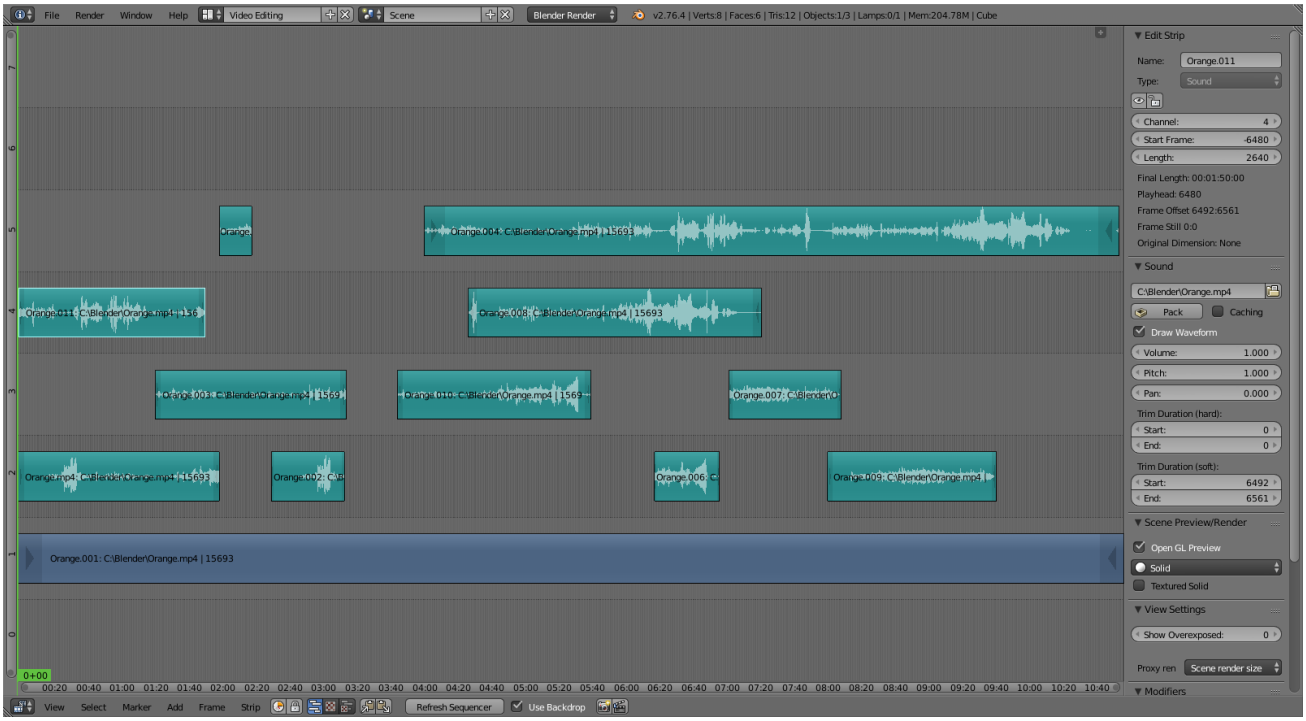


Fig. 2.302: Example of Sound Editing.

Caching Caching loads a file into ram and plays it from there, apposed to reading it for the hard drive.

Draw Waveform Draws a waveform over top of the sequence strip. This can be useful for syncing two or more audio strips.

Volume Changes the loudness of the audio.

Pitch Changes the frequency of the audio.

Pan Used to pan the audio from left an right channels -2 being hard left, 2 being hard right.

Working with Audio Tracks

An audio track (strip) is just like any other strip in the VSE. You can grab and move it, adjust its starting offset using RMB over the arrow end handles, and K cut it into pieces. A useful example is cutting out the "um's" and dead voice time.

You can have as many Audio strips as you wish and the result will be the mixing of all of them. You can give each strip its own name and volume via the N menu.

Overlapping strips are automatically mixed down during the rendering process. For example, you can have the announcer on channel 5, background music on channel 6, and Foley sound effects on channel 7.

Animating Audio Track Properties

To animate audio strips simply hit I over any of its values. Examples of animating an audio strip are to fade in/out background music or

to adjust volume levels. Layered/crossed audio strips are added together; the lower channel does not override and cut out higher channels (unlike image and video strips). This makes Blender an audio mixer. By adding audio tracks and using the curves to adjust each tracks' sound level, you have an automated dynamic multi-track audio mixer!

Output

There are two ways to render out your audio. You can either have it encoded with a video file or in its own audio file. To render your audio in an video file make sure to use a video format as the output with an audio codec and hit the render *Animation* button in the properties editor. Read more on how to do this *here*. To render as an audio file simple use the *Audio* button. Read more on how to do this *here*.

Known Limitations

Hiss, Crackle and Pop

In some cases when *Caching* is disabled, playback noise/hiss is introduced.

If you hear pops and crackles, usually that is a sign that your hardware cannot keep up in real-time playback. They will not be present in your final rendered animation output.

Also, static hiss can occur whenever two or more audio strips are overlapping in the timeline.

Meta Strips

Meta-Strips are a kind of organization tool. For example, if you are using a lot of strips and they are complicated the the interface you can group them together using Meta-Strips. A Meta-Strip spans from the beginning of the first strip to the end of the last one, and condenses all channels into a single strip. Separating (ungrouping) them restores them to their relative positions and channels. To create a Meta-Strip select all the strips you want to group, and `Ctrl-G` to group them. If you choose to delete a Meta-Strip and want to keep the strips inside, use `Alt-G`.

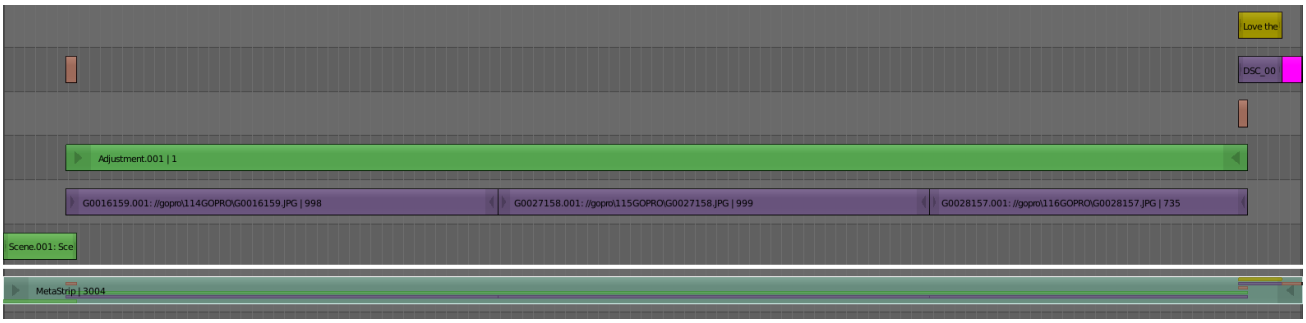


Fig. 2.303: Example of Meta-Strip.

After creating a Meta-Strip it is also possible to edit the contents inside a Meta-Strip. To do this select the desired Meta-strip and press `Tab`. Once you are done editing the contents inside a Meta-Strip press `Tab` again to exit the Meta-Strip. Meta-Strips can also be nested, which make editing them a little confusing. To exit out one level of Meta-Strip make sure you do not have a Meta-Strip selected when you press `Tab`.

Note: The default blend mode for a Meta strip is Replace. There are many cases where this alters the results of the animation so be sure to check the results and adjust the blend mode if necessary.

One convenient use for Meta-Strips is when you want to apply the same effect to multiple strips. For example: if you have a video that was recorded in different files and want to add an effect strip. It is much more convenient to apply a single set of effects to one Meta-Strip then applying it to each individual strip.

See also:

It is also possible to do the similar task described above with a *Adjustment Layer* effect strip.

Sequence Display Modes

By default, the VSE only displays the strips, however, there are a few ways to preview the result of your sequence. The first is the preview mode, this can be enable by hitting the texture button ( ).

Several options in the header allow you change the editor to display the sequence in real time, and in various ways.

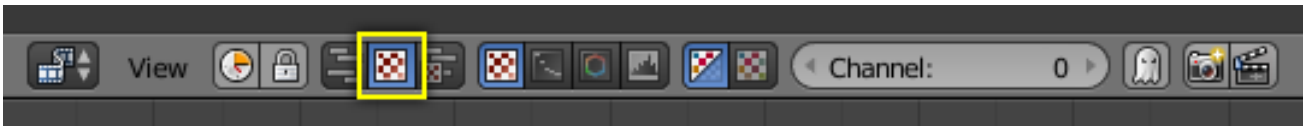


Fig. 2.304: Sequencer Display Header.

The second button will change the editor to display only the preview, and the third button displays both the sequencer and the preview.

The VSE workspace can show you different aspects of the composite result, for the current frame:

Image/Sequence Colors (what you see).

Chroma Color hue and saturation.

Luma Brightness/contrast.

Histogram Levels of red, green, and blue.

In the Chroma, Luma, and Image modes, a channel selector appears; channel 0 is the result of compositing the strips with their special effects strips. Channel 1 is what the current frame's image from the strip in channel 1 looks like (channel 1 is at the bottom of the heap). The display of these modes is either the composite (channel 0) or the frame from the strip (channels 1 through n).

Properties Region

Scene Preview/Render

OpenGL Preview When enabled *Scene Strips* use a quick OpenGL preview (see *OpenGL render* for more on this subject).

Otherwise a full render is used, which can be very slow.

Sequencer Preview Shading Method for rendering OpenGL renders.

Textured Solid Display textures even when in solid mode.

Settings used by OpenGL Previews:

- The anti-alias setting from the active scene is used for all scenes.
- The alpha setting is taken from each scene, where *Sky* fills in a solid background and *Transparent* has a transparent background.

View Settings

Show Overexposed Shows overexposed (bright white) areas using a zebra pattern. The threshold can be adjust with the slider.

Proxy Render Size Size to display proxies at in the preview region. Using a smaller preview size will increase speed.

Safe Areas

Shows guides used to position elements to ensure that the most important parts of the video can be seen across all screens.

See also:

See *Safe Areas* in the camera docs.

Grease Pencil

Allows you to use *Grease Pencil* in the sequencer.

Previews

The are an array of different display modes available, each having a specific purpose. You can adjust the view by zooming in with `Plus` and zoom out with `Minus`. You can also reset the view with `Home`.

Image Preview

In the upper area of the Sequence screen layout is another VSE editor, this one set to Image Preview mode. It shows you what the resulting video will look like when saved. This is the main working mode for adding strips and moving them around, cutting, grouping (making meta) and splicing them through special effects.

Luma Waveform

For the selected channel, brightness, or luminosity, is mapped with this display.

A luma waveform allows you to judge the quality of the luminance distribution across the video signal, you can view a luma-waveform instead of the usual output display on every control monitor.

The display plots for every scanline the luminance value. The lines are all drawn on top of each other. The points get brighter if the lines cross (which is very likely with several hundred scanlines). You will understand the picture most easily if you plug an oscilloscope to the Luma-video-output of your television set. It will basically look the same.

In this mode, the vertical axis represents the luminosity: 0 at the bottom, 1 at the top; the horizontal axis is a mapping from the horizontal axis of the frame. There are as many curves as scanlines in the frame: each one of this curves represents the luminosity of the pixels of one line. Moreover, the color of a pixel in this mode represents the number of pixels from the matching column of the frame sharing the same luminosity, i.e. the number of curves that cross at this point (black/transparent, for no pixel, white/opaque for at least three pixels).

Separate Colors Separates RGB channels into separate graphs.

This mode is good for:

- If the waveform does not fill the whole picture you might want to play with the “setup” and “gain” master-sliders in the “gamma”-plugin until it fills the whole picture (contrast autostretch).
- With the more advanced gamma-plugin you can decide where you have to desaturated (especially in dark regions).
- You can judge if you want to dump the whole thing since it is completely distorted and clips at the top or the bottom.

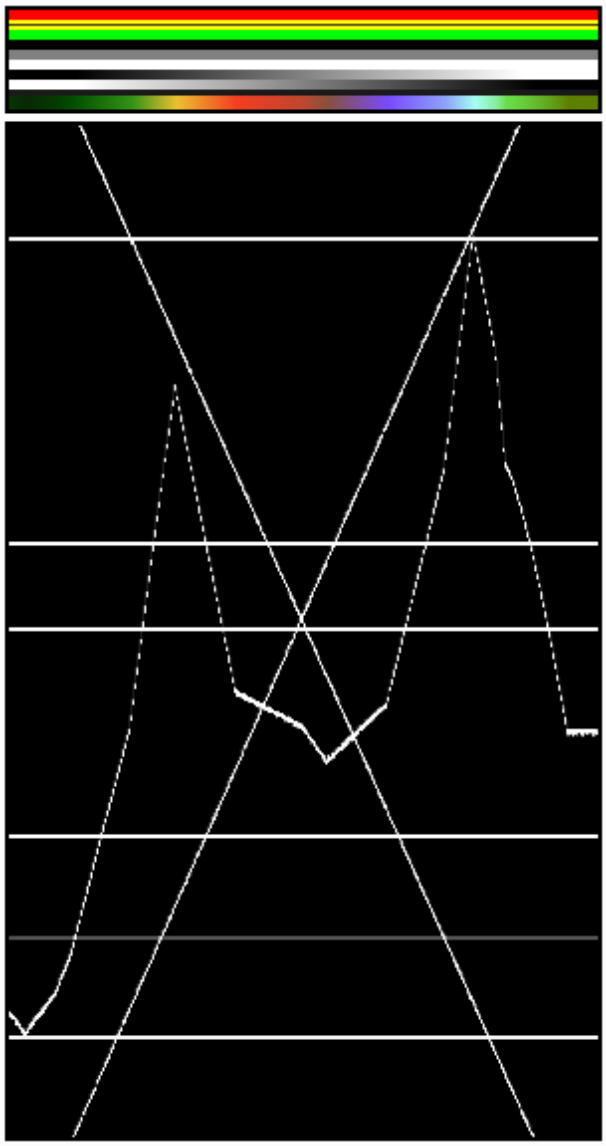


Fig. 2.305: The various horizontal lines in the Luma waveform match the uniform-color lines of the picture. Note that the 'gray 20%' one-pixel width line (inside the yellow strip) is represented in the Luma waveform by a gray line. The two lines drawing an "X" are from the two linear tone shades (white –> black and black –> white). Finally, the broken line matches the complex tone shade at the bottom of the picture.

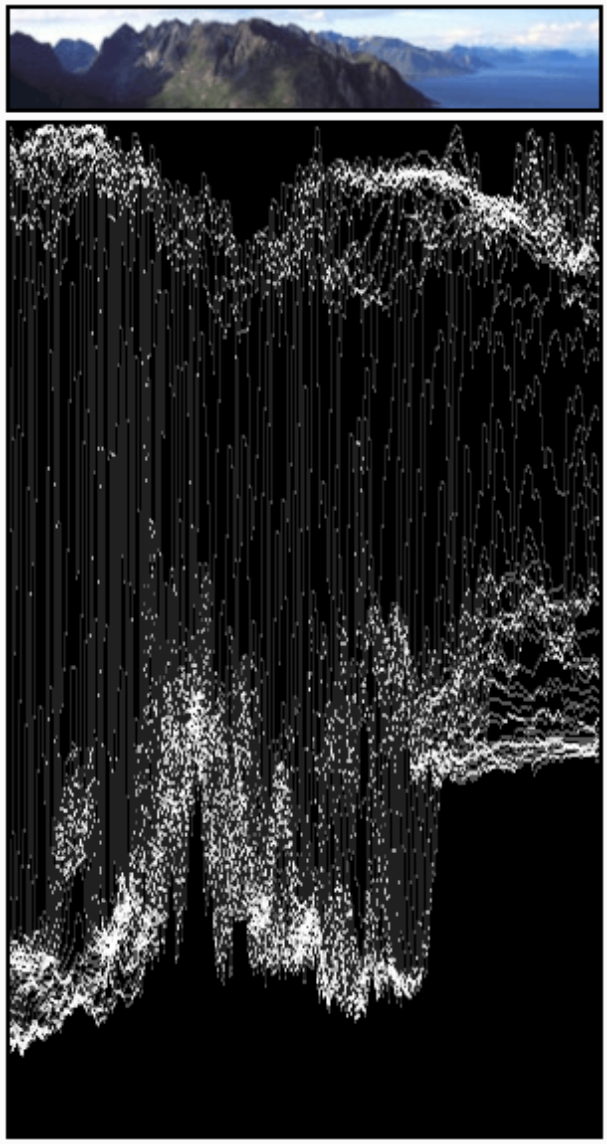


Fig. 2.306: The curves are quite visible. We found a luma of 80-100% for the sky, a luma around 40% for the sea, and a luma of 10-20% for the mountains, growing around 40% for the sunny part.

Note: Note that the pictures (first green frame, at the top) are only 50px high, to limit the number of curves displayed in the *Luma waveform*

Use this display to check for appropriate contrast and luminosity across all frames in the channel. When spots in the film that should have even illumination do not, it looks like a flashbulb went off or an extra light was suddenly turned on. This can happen if two strips were rendered or shot under different lighting conditions but are supposed to be contiguous.

Chroma Vectorscope

Fig. 2.307: Example image.

Use this mode judge the quality of the color-distribution and saturation, you can also view a U/V scatter-plot.

The picture is converted to YUV-format. The U- and V-values represent the angle of the color. For pixel of the picture, one point is plotted in the display at the U and V-value-position. If several pixels happen to have the same U/V-value the pixel in the plot gets brighter.

To help you understand what color is meant, a hexagram marking the extreme positions (red, magenta, blue, cyan, green, yellow) is drawn and a red cross to mark the origin.

In other words, for the selected channel, this display shows the color space of the image inside a hexagon. Each point of the hexagon is a primary color: red, magenta, blue, cyan, green, and yellow. Black is at the center, and overall saturation is scaled as dots closer to the outside. The example to the right shows that the image has a lot of red (50% saturation) and small amount of blue, with no green.

Always: remember to activate an additional control monitor of the end result. Color calibration is a matter of taste and depends on what you want.

Use this display to check for too much color saturation. While oversaturated images look great for op-art and computer displays, they stink when shown on the big screen TV. Use `Alt-A` to scrub the video; this display will update with a new/revised map for each frame. Just like watching the Image preview to see what it looks like, watch the Chroma Vectorscope to watch for color use.

This mode is good for:

- If you picture looks very moody or desaturated you might want to take a look at the U/V-plot. You will most likely see all pixels building a crowd at the origin. If you add saturation using the "gamma"-plugin you can see in the U/V-plot if you distort the color.
- If you do color-matching on a by hand basis you can match the angle you see of different channels monitors.

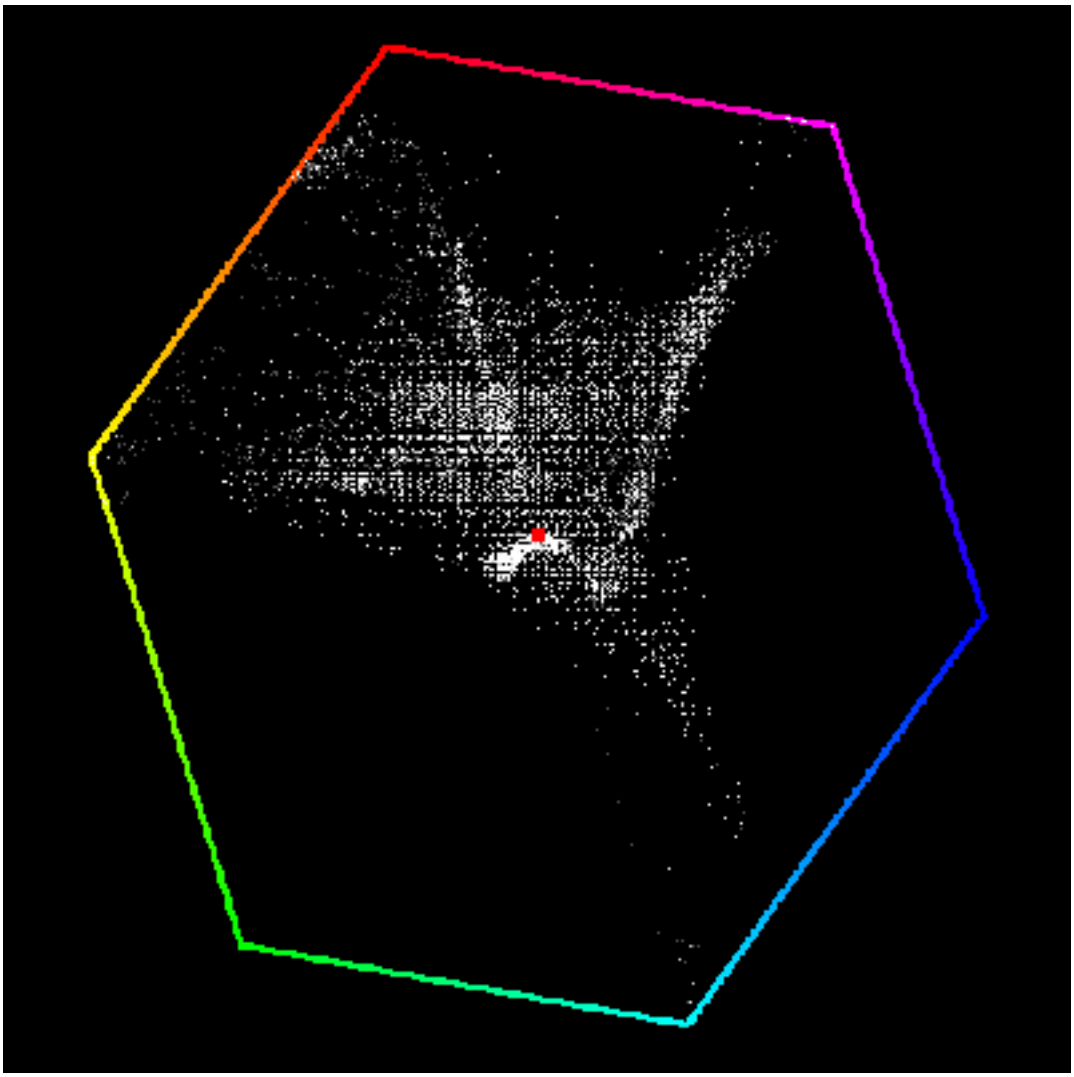


Fig. 2.308: Example of Chroma Vectorscope Preview.

Histogram

Fig. 2.309: Example image.

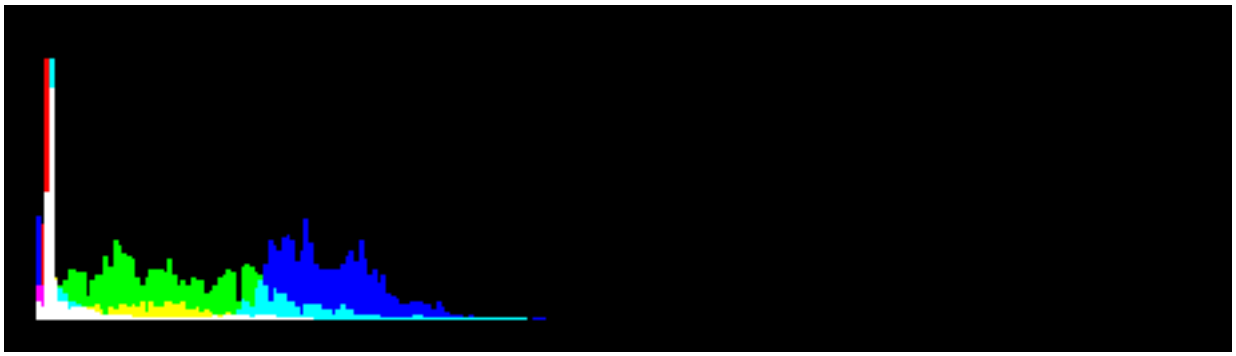


Fig. 2.310: Example of Histogram Preview.

This mode displays a graph showing the distribution of color information in the pixels of the currently displayed image. The X-axis represents values of pixel, from 0 to 1 (or 0 to 255), while the Y-axis represents the number of pixels in that tonal range. A predominantly dark image would have most of its information toward the left side of the graph.

Use this mode to balance out the tonal range in an image. A well balanced image should a nice smooth distribution of color values.

Movie Clip Editor

Introduction

The Movie Clip Editor has two main purposes, it can be used for for tracking or masking movies. The default layout looks like the image below.

In order to get to either the masking or the tracking tools you must first add a movie file. There are several ways to to this:

- Use *Open* button from movie editor header.

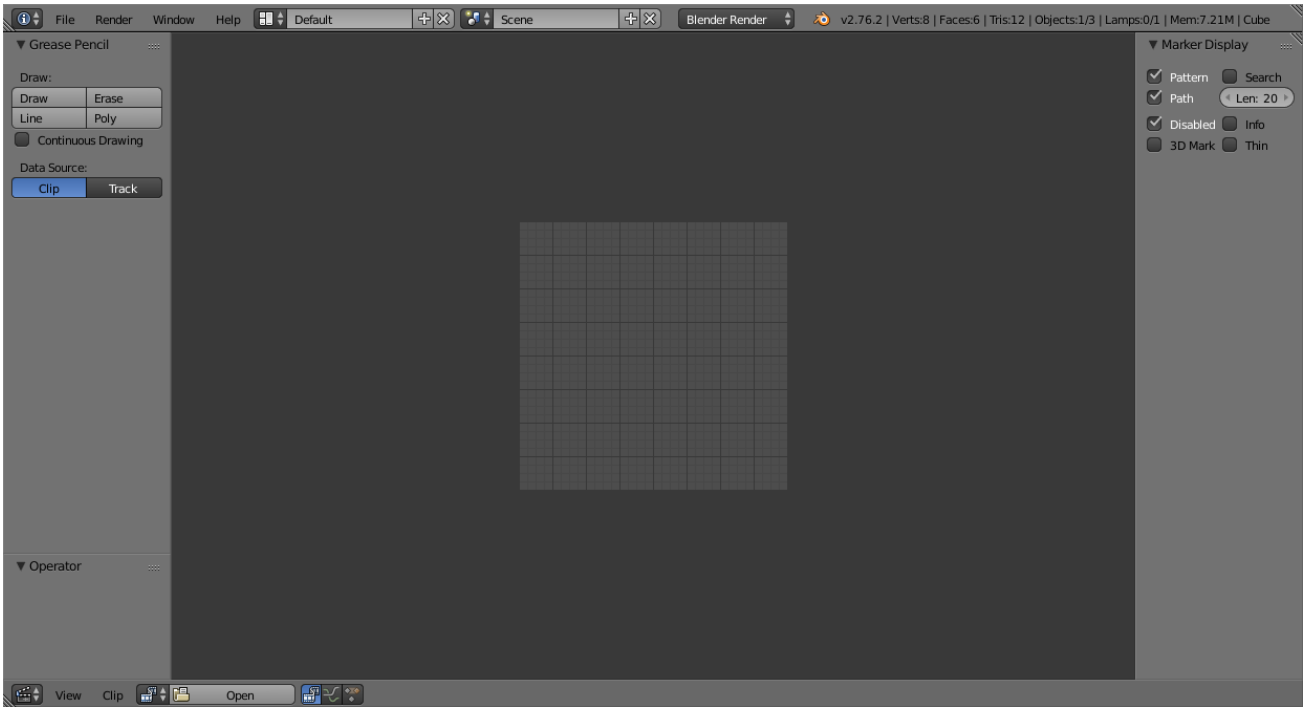


Fig. 2.311: Movie Clip Editor interface.

- Use *Clip* → *Open* menu.
- Use `Alt-O` shortcut.

Both movie files and image sequences can be used in the clip editor. If you are using an image sequence there is one limitation on naming of files: the numbers at the end of the image name should be increasing continuously. After that you will then be able to choose to options in the select menu for what you want to do.

So, when a movie clip is loaded into the clip editor, extra panels are displayed in the interface.

- *Motion Tracking*
- *Masking*

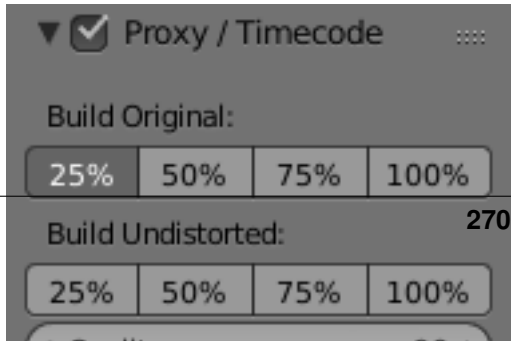
Movie Clip Properties

Proxy/Timecode Panel

Once you have chosen the Proxy/Timecode parameters, you need to use *Clip* → *Proxy* → *Rebuild Proxy and Timecode indices* to generate the proxy clip and it will be available after Blender makes it.

Proxy

A proxy is a smaller image (faster to load) that stands in for the main image. When you rebuild proxies Blender computes small images (like thumbnails) for the big images and may take some time. After computing them, though, editing functions like scrubbing and scrolling is much faster but gives a low-res result. Make sure to disable proxies before final rendering.



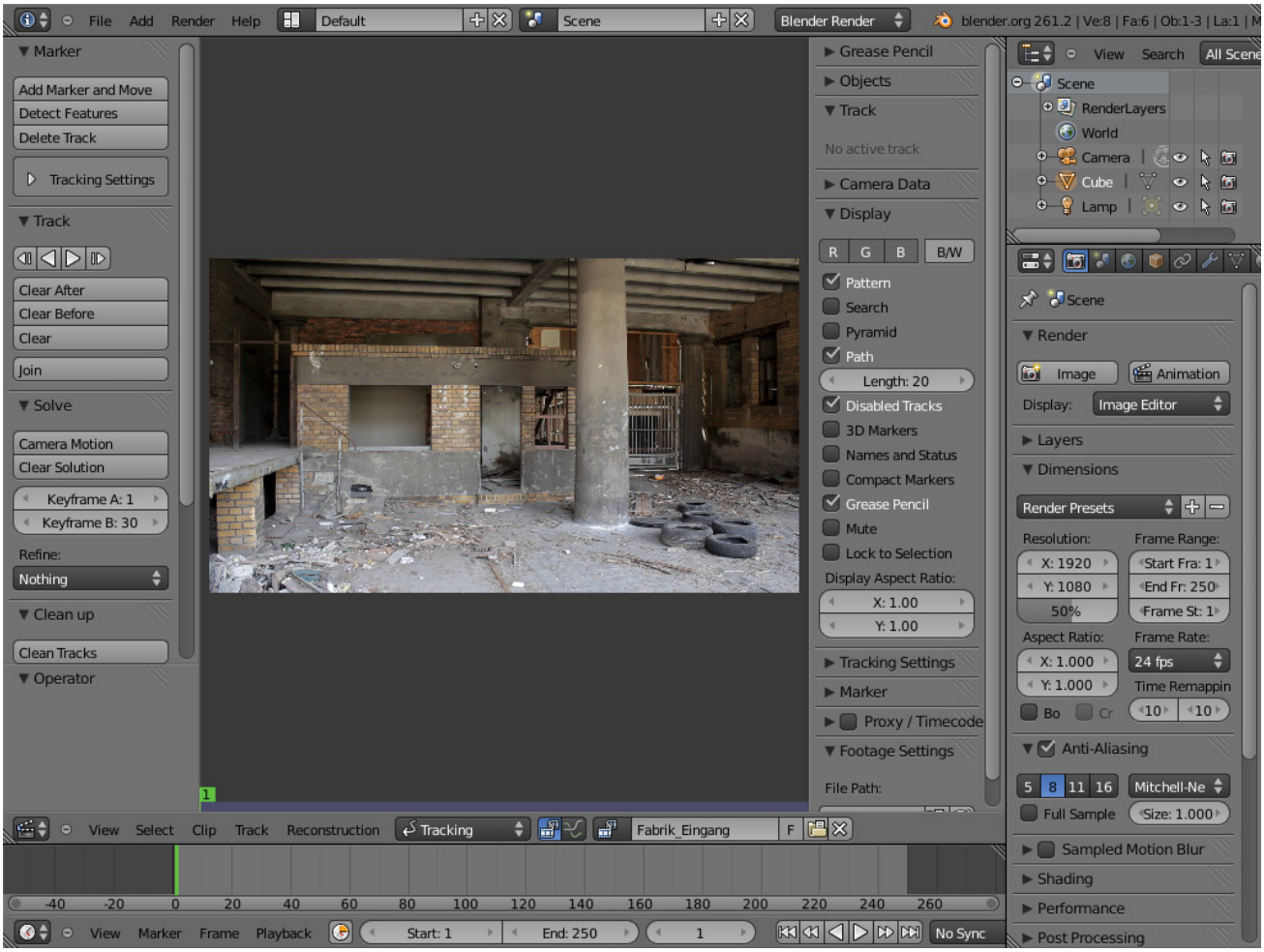


Fig. 2.312: Movie Clip Editor with an Opened Clip.

Build Original Used to define which resolutions of proxy images should be built.

Build Undistorted Builds images from undistorted original images for the sizes set above. This helps provide faster playback of undistorted footage.

Quality Defines the quality of the JPEG images used for proxies.

Proxy Custom Directory By default, all generated proxy images are storing to the `<path of original footage>/BL_proxy/<clip name>` folder, but this location can be set by hand using this option.

Rebuild Proxy Regenerates proxy images for all sizes set above and regenerate all timecodes which can be used later.

Timecode See *Timecode*.

Proxy Render Size defines which proxy image resolution is used for display. If *Render Undistorted* is set, then images created from undistorted frames are used. If there is no generated proxies, render size is set to "No proxy, full render", and render undistorted is enabled, undistortion will happen automatically on frame draw.

Timecode

When you are working with footage directly copied from a camera without pre-processing it, there might be bunch of artifacts, mostly due to seeking a given frame in sequence. This happens because such footage usually does not have correct frame rate values in their headers. So, for Blender to calculate the position of a needed frame in the stream works inaccurately and can give errant result. There are two possible ways to avoid this:

- Preprocess your video with, say, mencoder to repair file header and insert correct keyframes.
- Use Proxy/Timecode option in Blender.

Options

Timecode Timecode to use on the selected movie strip.

The following timecodes are supported:

- No TC in use- do not use any timecode
- Record Run
- Free Run
- Free Run (rec date)
- Record Run No Gaps

Note: Record Run is the timecode which usually is best to use, but if the clip's file is totally damaged, *Record Run No Gaps* will be the only chance of getting acceptable result.

Motion Tracking

Introduction

Motion Tracking is used to track the motion of objects and applying that data to 3D object through the compositor. Blender's motion tracker supports a couple of very powerful tools for 2D tracking and 3D motion tracking, including camera tracking and object tracking, as well as some special features like the plane track for compositing. Tracks can also be used to move and deform masks for rotoscoping in the Mask Editor, which is available as a special mode in the Movie Clip Editor.

Manual Lens Calibration

All cameras record distorted video. Nothing can be done about this because of the manner in which optical lenses work. For accurate camera motion, the exact value of the focal length and the "strength" of distortion are needed.

Currently, focal length can be automatically obtained only from the camera's settings or from the EXIF information. There are some tools which can help to find approximate values to compensate for distortion. There are also fully manual tools where you can use a grid which is getting affected by distortion model and deformed cells defines straight lines in the footage.

You can also use the grease pencil for this – just draw a line which should be straight on the footage using poly line brush and adjust the distortion values to make the grease pencil match lines on the footage.

To calibrate your camera more accurately, use the grid calibration tool from OpenCV. OpenCV is using the same distortion model, so it should not be a problem.

Camera and Object Motion Solving

Blender not only supports the solving of camera motion, including tripod shots, but also the solving of object motion in relation to the motion of the camera. In addition to that there is the Plane Track, which solves the motion of all markers on one plane.

There are also plans to add more tools in the future, for example more automatic tracking and solving, multi-camera solving and constrained solutions.

Tools for Scene Orientation and Stabilization

After solve, you need to orient the real scene in the 3D scene for more convenient compositing. There are tools to define the floor, the scene origin, and the X/Y axes to perform scene orientation.

Sometimes, the video footage includes spurious jumps and tilting movements, like e.g. when using a hand held camera. Based on some tracked image elements, the *2D Stabilization* is able to detect and compensate such movements to improve the quality of the final result.

Clip View

Introduction

The clip view is used is the main part of the of the movie clip editor. Almost all motion tracking tools are concentrated in the Movie Clip Editor.

It should be mentioned that the camera solver consists of three quite separate steps:

1. 2D tracking of footage.
2. Camera intrinsics (focal length, distortion coefficients) specification/estimation/calibration.
3. Solving camera, scene orientation, and scene reconstruction.

Tools in the clip editor are split depending on which step they are used in, so the interface is not cluttered up with scene orientation tools when only 2D tracking can be done. The currently displayed tool category can be changed using the Mode menu, which is in the editor header.

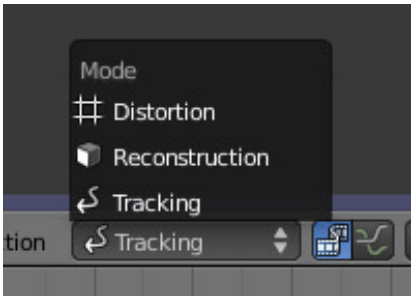


Fig. 2.313: Movie Clip Editor Mode Menu.

But almost all operators can be called from menus, so it is not necessary to change the mode every time you want to use a tool which is associated with a different editor mode.

In tracking mode only tools which are related to tracking and camera solving are displayed. Camera solving tools are included here because it is after solving you will most probably want to re-track existing tracks or place new tracks to make solving more accurate.

Tracking Settings Panel

This panel contains all settings for the 2D tracking algorithms. Depending on which algorithm is used, different settings are displayed, but there are a few that are common for all tracker settings:

Adjust Frames controls which patterns get tracked; to be more precise, the pattern from which frame is getting tracked. Here is an example which should make things clearer.

The tracker algorithm receives two images inside the search area and the position of a point to be tracked in the first image. The tracker tries to find the position of that point from the first image in the second image.

Now, this is how tracking of the sequence happens. The second image is always from a frame at which the position of marker is not known (next tracking frame). But a different first image (instead of the one that immediately precedes the second image in the footage) can be sent to the tracker. Most commonly used combinations:

- An image created from a frame on which the track was keyframed. This configuration prevents sliding from the original position (because the position which best corresponds to the original pattern is returned by the tracker), but it can lead to small jumps and can lead to failures when the feature point is deformed due to camera motion (perspective transformation, for example). Such a configuration is used if *Adjust Frames* is set to 0.
- An image created from the current frame is sent as first image to the tracker. In this configuration the pattern is tracking between two neighboring frames. It allows dealing with cases of large transformations of the feature point but can lead to sliding from the original position, so it should be controlled. Such a configuration is used if *Adjust Frames* is set to 1.

If *Adjust Frames* is greater than 1, the behavior of tracker is: keyframes for tracks are creating every *Adjust Frames* frames, and tracking between keyframed image and next image is used.

Speed can be used to control the speed of sequence tracking. This option does not affect the quality of tracking; it just helps to control if tracking happens accurately. In most cases tracking happens much faster than real time, and it is difficult to notice when a track began to slide out of position. In such cases *Speed* can be set to Double or Half to add some delay between tracking two frames, so slide-off would be noticed earlier and the tracking process can be canceled to adjust positions of tracks.

Frames Limit controls how many frames can be tracked when the Track Sequence operator is called. So, each Track Sequence operation would track maximum *Frames Limit* frames. This also helps to notice slide-off of tracks and correct them.

Margin can be used disable tracks when they become too close to the image boundary. This slider sets "too close" in pixels.

KLT Tracker Options

The KLT tracker is the algorithm used by default. It allows tracking most kinds of feature points and their motion. It uses pyramid tracking which works in the following way. The algorithm tracks an image larger than the defined pattern first to find the general direction of motion. Then it tracks a slightly smaller image to refine the position from the first step and make the final position more accurate. This iterates several times. The number of steps of such tracking is equal to the *Pyramid Level* option and we tell that on first step tracking happens for highest pyramid level. So Pyramid Level=1 is equal to pattern itself, and each next level doubles tracking image by 2.

The search area should be larger than the highest pyramid level and the "free space" between the search area and highest pyramid level defines how much the feature can move from one frame to another and still be tracked.

Default settings should work in most general cases, but sometimes the pyramid level should be changed. For example, when footage is blurry, adding extra pyramid levels helps to track them.

This algorithm can fail in situations where a feature point is moving in one direction and the texture around that feature point is moving in another direction.

SAD tracker options

On each step, the SAD tracker reviews the whole search area and finds the pattern on the second image which is most like the pattern which is getting tracking. This works pretty quickly, but can fail in several cases. For example, when there is another feature point which looks like the tracking feature point in the search area. In this case, SAD will tend to jump off track from one feature to another.

Correlation defines the threshold value for correlation between two patterns which is still considered successful tracking. 0 means there is no correlation at all, 1 means correlation is full.

There is one limitation currently, it works for features of size 16×16 pixels only.

Movie Clip Properties

Objects Panel

This panel contains a list of all objects which can be used for tracking, camera or object solving. By default there is only one object in this list which is used for camera solving. It cannot be deleted and other objects cannot be used for camera solving; all added objects are used for object tracking and solving only. These objects can be referenced from Follow Track and Object Solver constraints. Follow Track uses the camera object by default.

New objects can be added using `Plus` and the active object can be deleted with the `Minus` button. Text field at the bottom of this panel is used to rename the active object.

If some tracks were added and tracked to the wrong object, they can be copied to another object using *Track* → *Copy Tracks* and *Track* → *Paste Tracks*.

The usage for all kind of objects (used for camera and object tracking) is the same: track features, set camera data, solve motion. Camera data is sharing between all objects and refining of camera intrinsics happens when solving camera motion only.

None

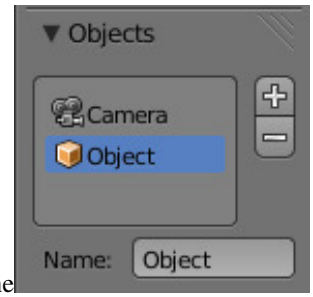


Fig. 2.314: Objects Panel.

Track Panel

First of all, track name can be changed in this panel. Track names are used for linking tracking data to other areas, like a Follow Track constraint.

The next thing which can be controlled here is the marker's enabled flag (using the button with the eye icon). If a marker is disabled, its position is not used either by solver nor by constraints.

The button with the lock icon to the right of the button with the eye controls whether the track is locked. Locked tracks cannot be edited at all. This helps to prevent accidental changes to tracks which are "finished" (tracked accurate along the whole footage).

None

Fig. 2.315: Track Panel.

The next widget in this panel is called "Track Preview" and it displays the content of the pattern area. This helps to check how accurately the feature is being tracked (controlling that there is no sliding off original position) and also helps to move the track back to the correct position. The track can be moved directly using this widget by mouse dragging.

If an anchor is used (the position in the image which is tracking is different from the position which is used for parenting), a preview widget will display the area around the anchor position. This configuration helps in masking some things when there is no good feature at position where the mask corner should be placed. Details of this technique will be written later.

There is small area below the preview widget which can be used to enlarge the vertical size of preview widget (the area is highlighted with two horizontal lines).

The next setting is channels control. Tracking happens in gray-scale space, so a high contrast between the feature and its background yields more accurate tracking. In such cases disabling some color channels can help.

When several tracks are used for 3D camera reconstruction or for 2D stabilization, it is possible to assign a reduced weight to some tracks to control their influence on the solution result. The *Weight* parameter is used for 3D reconstruction, while the *Stab Weight* parameter is used to control 2D stabilization. This parameter can (and often need to be) animated.

The last thing is custom color, and the preset for it. This setting overrides the default marker color used in the clip editor and 3D View, and it helps to distinguish different type of features (for example, features in the background vs. foreground and so on). Color also can be used for "grouping" tracks so a whole group of tracks can be selected by color using the Select Grouped operator.

Tip: To select good points for tracking, use points in the middle of the footage timeline and track backwards and forwards from there. This will provide a greater chance of the marker and point staying in the camera shot.

Plane Track Panel

Its properties are shown only when a plane track is selected. Firstly, the name of the selected plane track is shown. It can also be changed from here.

Auto Keyframe Toggles the auto-keyframing for corners of the plane track. With this enabled, keyframes will automatically get inserted when any corner is moved.

None

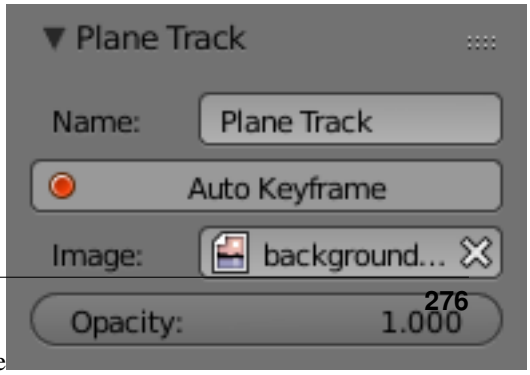


Fig. 2.316: Plane Track Panel

Note: Corners can be moved using `LMB`.

Image Used to select an image which will be inside the plane track.

Note: This image is for viewing purposes in movie clip editor only. To include it in your final render, see *Plane Track Deform node*.

Opacity Used to set the opacity of this image. Again, this is for display purposes only, and will not affect your final render.

Camera Data Panel

This panel contains all settings of the camera used for filming the movie which is currently being edited in the clip editor.

Camera Presets Predefined settings can be used here. But such settings as distortion coefficients and principal point are not included in the presets and should be filled in even if camera presets are used.

Focal Length Is self-explanatory; it is the focal length with which the movie was shot. It can be set in millimeters or pixels. In most cases focal length is given in millimeters, but sometimes (for example in some tutorials on the Internet) it is given in pixels. In such cases it is possible to set it directly in the known unit.

Sensor Width Is the width of the CCD sensor in the camera. This value can be found in camera specifications.

Pixel Aspect Ratio Is the pixel aspect of the CCD sensor. This value can be found in camera specifications, but can also be guessed. For example, you know that the footage should be 1920×1080, but the images themselves are 1280×1080. In this case, the pixel aspect is: 1920 / 1280 = 1.5 .

Optical Center Is the optical center of the lens used in the camera. In most cases it is equal to the image center, but it can be different in some special cases. Check camera/lens specifications in such cases. To set the optical center to the center of image, there is a `Return` button below the sliders.

Undistortion K1, K2 and K3 Are coefficients used to compensate for lens distortion when the movie was shot. Currently these values can be tweaked by hand only (there are no calibration tools yet) using tools available in Distortion mode. Basically, just tweak K1 until solving is most accurate for the known focal length (but also take grid and grease pencil into account to prevent "impossible" distortion).

Display Panel

This panel contains all settings which control things displayed in the clip editor.

R, G, B And *B/W* buttons at the top of this panel are used to control color channels used for frame preview and to make the whole frame gray scale. It is needed because the tracking algorithm works with gray-scale images and it is not always obvious to see which channels disabled will increase contrast of feature points and reduce noise.

Pattern Can be used to disable displaying of rectangles which correspond to pattern areas of tracks. In some cases it helps to make the clip view cleaner to check how good tracking is.

Search Can be used to disable displaying of rectangles which correspond to search areas of tracks. In some cases it helps to make the clip view cleaner to check how good tracking is. Only search areas for selected tracks will be displayed.

Pyramid Makes the highest pyramid level be visible. Pyramids are defined later in the Tracking Settings panel section, but basically it helps to determine how much a track is allowed to move from one frame to another.

Track Path And *Length* control displaying of the paths of tracks. The ways tracks are moving can be visible looking at only one frame. It helps to determine if a track jumps from its position or not.

Disabled Tracks Makes it possible to hide all tracks which are disabled on the current frame. This helps to make view more clear, to see if tracking is happening accurately enough.

Bundles Makes sense after solving the movie clip, and it works in the following way: the solved position of each track gets projected back to the movie clip and displayed as a small point. The color of the point depends on the distance between the projected coordinate and the original coordinate: if they are close enough, the point is green, otherwise it will be red. This helps to find tracks which were not solved nicely and need to be tweaked.

Track Names and Status Displays information such as track name and status of the track (if it is keyframed, disabled, tracked or estimated). Names and status for selected tracks are displayed.

Compact Markers The way in which markers are displayed (black outline and yellow foreground color) makes tracks visible on all kind of footage (both dark and light). But sometimes it can be annoying and this option will make the marker display more compactly - the outline is replaced by dashed black lines drawn on top of the foreground, so that marker areas are only 1px thick.

Grease pencil Controls if grease pencil strokes are allowed to be displayed and made.

Mute Changes displaying on movie frame itself with black square, It helps to find tracks which are tracked inaccurately or which were not tracked at all.

Grid Displays a grid which is originally orthographic, but is affected by the distortion model (available in distortion mode only). This grid can be used for manual calibration – distorted lines of grids are equal to straight lines in the footage.

Manual Calibration Applies the distortion model for grease pencil strokes (available in distortion mode only). This option also helps to perform manual calibration. A more detailed description of this process will be added later.

Display Stabilization This option makes the displayed frame be affected by the 2D stabilization settings (available in reconstruction mode only). It is only a preview option, which does not actually change the footage itself.

Lock to Selection Makes the editor display selected tracks at the same screen position along the whole footage during playback or tracking. This option helps to control the tracking process and stop it when the track is starting to slide off or when it jumped.

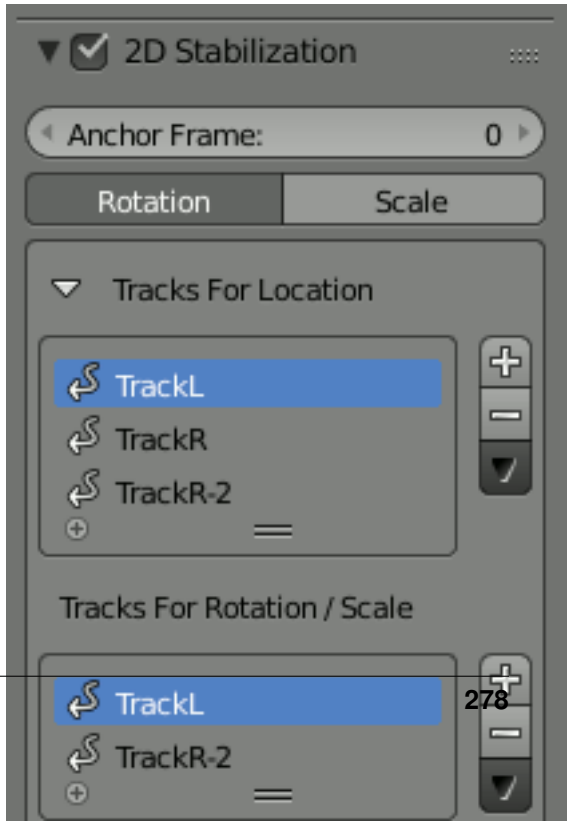
Display Aspect Ratio Changes the aspect ratio for displaying only. It does not affect the tracking or solving process.

Marker Panel

This panel contains numerical settings for marker position, pattern and search area dimensions, and offset of anchor point from pattern center. All sliders are self-explanatory.

2D Stabilization Panel

There is one extra panel which is available in reconstruction mode – 2D Stabilization Panel. The purpose of this feature is to smooth out jerky camera handling on existing real world footage. To activate the 2D stabilizer, you need to set the toggle in the panel, and additionally you need to enable *Display Stabilization* in the Display panel. Then you'll need to set up some tracking points to detect the image movements.



The 2D Stabilization panel is used to define the data used for 2D stabilization of the shot. Several options are available in this panel: you may add a list of tracks to determine lateral image shifts and another list of tracks to determine tilting and zooming movements. Based on the average contribution of these tracks, a compensating movement is calculated and applied to each frame.

When the footage includes panning and traveling movements, the stabilizer tends to push the image out of the visible area. This can be compensated by animating the parameters for the intentional, "expected" camera movement.

See the *Stabilization* page for detailed description of image stabilization parameters and workflow.

Grease Pencil Panel

It is a standard grease pencil panel where new grease pencil layers and frames can be controlled. There is one difference in the behavior of the grease pencil from other areas – when a new layer is created "on-demand" (when making a stroke without adding a layer before this) the default color for the layer is set to pink. This makes the stroke easy to notice on all kinds of movies.

2D Stabilization

The 2D video stabilization is a feature built on top of Blender's image feature tracking abilities: we use some *tracking points* to remove shakiness, bumps and jerks from video footage. Typically, image stabilization is part of a **2D workflow** to prepare and improve footage prior to further processing or modeling steps. This page helps to understand how it works, introduces related terms and concepts, describes the available interface controls in detail and finally gives some hints about usage in practice.

Typical **usage scenarios** of the stabilizer:

- fix minor deficiencies (shaky tripod, jerk in camera movement)
- "poor man's steadycam" (when a real steadycam was not available, affordable or applicable)
- as preparation for masking, matching and rotoscoping

It is not uncommon for 2D stabilization to have to deal with somewhat imperfect and flawed footage.

How it works

To detect spurious movement in a given shot, we'll assume a simplified model about this movement. We then try to fit the movement of tracked features with this simplified model to derive a compensation. Of course, this works only to the degree our model is adequate – yet in practice, this simplified approach works surprisingly well even with rather complicated shots, where our basic assumption was just an approximation of much more elaborate movements.

This simplified model underlying the 2D stabilization as implemented here assumes movement by an **affine-linear transform**:

- the camera is pushed up/down/sideways by some **translation component**
- the image is then **tilted** and **scaled** around a **pivot point** (rotation center)

To compensate movement according to this simplified model, the 2D stabilizer proceeds in two steps. First we try to detect the translation offset from the weighted average of all *translation tracking points*. After compensating this translation component, we then use additional *rotation/scale tracking points* to detect rotation around a given pivot point. Again, we detect rotation and scale changes through a weighted average of all the rotation/scale tracking points given.

In the current version, the **pivot point** is anchored to the *weight center of the translation tracking points*. So effectively the detected translation is already factored out. In some cases this is not optimal, especially when tracks have gaps or do not cover the whole duration of the footage – we plan further options to better control the pivot point in future releases.

Stabilization tracks

Thus, as foundation for any image stabilization, we need tracked image features to derive the movements. These *tracking points* or "tracks" can be established with Blender's *image feature tracking component* The right choice of points to track is somewhat tricky, yet crucial for successful image stabilization. Often, we're here because we'll have to deal with imperfect footage. In such cases, the *averaging of tracks* helps to work around image or tracking errors at some point. Moreover, when the footage contains *perspective induced movements,* symmetrically placed tracking points above and below the horizon can be used to cancel out spurious movement and get stabilization to the focal area in between.

Fig. 2.318: Diverging movements caused by perspective.

Tracks can be added in two groups

- first of all is the list of tracks to be used to compensate for jumps in the camera location. From all the tracking points added to this group, we calculate a *weighted average.* We then try to keep this average location constant during the whole shot. Thus it is a good idea to use tracking markers close to and centered around the most important subject.
- a second selection of tracks is used to keep the rotation and scale of the image constant. You may use the same tracks for both selections. But usually it is best to use tracking points with large distance from the image center, and symmetrically, on both sides, to capture the angular movements more precisely. Similar to the "location" case, we calculate an *average angular contribution* and then try to keep this value constant during the whole shot.

Footage, image and canvas

When talking about the movement stabilization of video, we have to distinguish several frames of reference. The image elements featured by the footage move around irregularly within the footage's **original image boundaries** – this is the very reason why we are using the stabilizer. When our attempt at stabilization was successful, the image elements can be considered *stable* now, while in exchange the footage's image boundaries have taken on irregular movement and jump around in the opposite way. This is the immediate consequence of the stabilizer's activity.

Since the actual image elements, i.e the subject of our footage can be considered stable now, we may use these as a new frame of reference: we consider them attached to a fixed backdrop, which we call the **canvas**. Introducing this concept of a "canvas" helps to deal with deliberate movements of the camera. And beyond that, it yields an additional benefit: It is very frequent for the pixels of video footage to be *non square.* So we have to stretch and expand those pixels, before we're able to preform any sensible rotation stabilization. Thus the canvas becomes, by definition, the reference for an undistorted display of the image contents.

But when the camera was *moved intentionally,* we have to consider yet another frame of reference beyond the canvas: namely the frame (or *"cadre"*) of the **final image** we want to create. To understand this distinction, let's consider a hand-held, panning shot to the right: Since our camera was turned towards the right side, the actual image contents move towards the left side *within* the original image frame. But let's assume the stabilizer was successful with "fixing" any image contents relative to the *canvas* – which in turn means, that the original image boundaries start to move irregularly towards the right side, and the *contents* of the image will begin to disap-

pear gradually behind the left boundary of the original image. After some amount of panning, we'll have lost all of our original contents and just see an empty black image backdrop. The only solution to deal with that problem is to *move the final image frame along to the right,* thus following the originally intended panning movement. Of course, this time, we do want to perform this newly added panning movement in a smooth and clean way.

Fig. 2.319: Stabilizing a panning shot.

To allow for such a compen-

sation and to reintroduce deliberate panning, or tilting and zoom of the resulting image, the stabilizer offers a dedicated set of controls: *Expected position*, *Expected rotation* and *Expected scale*. These act like the controls of a virtual camera filming the contents we have fixed onto the canvas. By *animating* those parameters, we're able to perform all kinds of deliberate camera movements in a smooth fashion.

The "dancing" black borders

As explained above, when we succeed with stabilizing the image contents, the boundaries of the original footage start to jump around in the opposite direction of the movements compensated. This is inevitable – yet very annoying, since due to the irregular nature of these movements, these "dancing black borders" tend to draw away attention from the actual subject and introduce an annoying restlessness. Thus our goal must be to hide those dancing borders as good as possible. A simple solution is to add a small amount of zoom. Sometimes we'll also need to animate the parameter *Expected position* in order to keep the image centered as good as we can – this helps to reduce the amount of zoom necessary to remove those annoying borders.

The **Autoscale function** can be used to find the minimal amount of zoom just sufficient to remove those black borders completely. However, if the camera jumps a lot, the autoscale function often zooms in too much, especially since this calculation aims at finding a single, static zoom factor for the whole duration of the footage. When this happens, you'll typically get overall better results with animating both the zoom factor and the expected position manually.

2D Stabilizer Panel

Note: To *activate* the 2D stabilizer, you need to set the toggle in the panel, and additionally you need to enable *Display Stabilization* in the *Display* panel.

Anchor Frame Reference point to anchor stabilization: other frames will be adjusted relative to this frame's position,

orientation and scale. You might want to select a frame number where your main subject is featured in an optimal way.

Stabilization Type

Rotation In addition to location, stabilizes detected rotation around the *rotation pivot point,* which is the weighted average of all location tracking points.

Scale Compensates any scale changes relative to center of rotation.

Tracks For Stabilization

Location List of tracks to be used to compensate for camera jumps, or location movement.

Rotation/Scale List of tracks to be used to compensate for camera tilts and scale changes.

Autoscale Finds smallest scale factor which, when applied to the footage, would eliminate all empty black borders near the image boundaries.

Max Limits the amount of automatic scaling.

Expected Position X/Y Known relative offset of original shot, will be subtracted, e.g. for panning shots

Expected Rotation Rotation present on original shot, will be compensated, e.g. for deliberate tilting.

Expected Zoom Explicitly scale resulting frame to compensate zoom of original shot.

Influence The amount of transformation applied to the footage can be controlled. In some cases it is not necessary to fully compensate camera jumps. The amount of stabilization applied to the footage can be controlled. In some cases you may not want to fully compensate some of the camera's jumps. Please note that these "* *Influence*" parameters do control only the *compensation movements* calculated by the stabilizer, not the deliberate movements added through the "*Expected* *"-parameters.

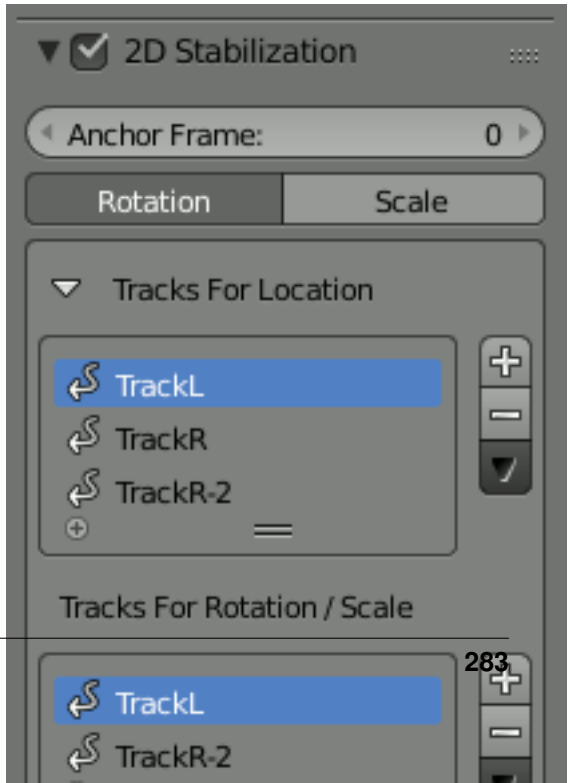
Interpolate The stabilizer calculates compensation movements with sub pixel accuracy. Consequently, a resulting image pixel needs to be derived from several adjacent source footage pixels. Unfortunately, any interpolation causes some minor degree of softening and loss of image quality.

Nearest No interpolation, uses nearest neighboring pixel. No interpolation, use nearest neighboring pixel. This setting basically retains the original image's sharpness. The downside is we also retain residual movement below the size of one pixel, and compensation movements are done in 1 pixel steps, which might be noticeable as irregular jumps.

Bilinear Simple linear interpolation between adjacent pixels.

Bicubic Highest quality interpolation, most expensive to calculate

Stabilization Workflow



Depending on the original footage's properties, achieving good stabilization results might be simple and easy, or it might require some work, dedication and careful planning. This section covers some practical considerations to help improving the results.

The simple case

Whenever the camera is basically fixed, or at least "almost" stationary, and the footage is crisp and without motion blur, perfect stabilization is easy to achieve. This might be the case when a tripod was used, but wind or vibrations on the floor (e.g. on a stage) caused some minor shakes. Shoulder camera shots done by an experienced operator also frequently fall into this category.

- Use as few points as possible. Start with a single point right on the main subject.

- Track this single point as accurate as possible. Beware of movements and shape changes of the tracked feature. Proceed in small increments (e.g. 50 frames), zoom in and readjust the target point manually when it drifts away. Another option is to use a larger target area for tracking; since we're tracking only a single point, the slower tracking speed might be acceptable.
- After enabling the basic (location) stabilization, consider if you really need rotation stabilization. Often, some minor, slow swinging movements are not really noticeable and do not warrant the additional working time and quality loss caused by rotation and scale stabilization.
- For rotation, start with one extra point, well spaced but preferably still attached to the main subject.
- Consider to fix some slow residual motion by manually animating the "*Expected **" parameters, before you even think of adding more tracking markers. Because doing so is often not worth the effort.
- If you need to add more points, the most important goal is to achieve *symmetry.* Place location tracking points symmetrically above and below the horizon. Place rotation tracking points into diagonally opposed direction, always centered around the main focal area.

Avoid problematic footage

The 2D stabilizer can not work miracles; some flaws simply can not be fixed satisfactory. Notorious issues are motion blur, rolling shutter, pumping autofocus and moving compression artifacts. Especially if you do succeed with basic stabilization, such image flaws become yet the more noticeable and annoying. When on set or on location, it might be tempting to "fix matters in postpro". Resist that deception, it rarely works out well.

- Prefer a short exposure time to avoid motion blur. While motion blur is good to render filmed movements more smooth and natural, it seriously impedes the ability to track features precisely. As a guideline, try to get at least to 1/250 s
- Prefer higher frame rates. The more *temporal resolution* the stabilizer has to work on, the better the results. If you have the option to choose between progressive and interlaced modes, by all means use interlaced and de-interlace the footage to the *doubled frame rate.* This can be done with the yadif filter of FFmpeg: use the mode 1 (`send_field`).
- Beware of Rolling Shutter. Avoid fast lateral movements. If you can, prefer a camera which produces less rolling shutter. Also, using a higher frame rate reduces the amount of rolling shutter; another reason to prefer interlaced over progressive for the purpose at hand.
- Switch off autofocus. Better plan your movement beforehand, set a fixed focus and rely on depth-of-field through using a small aperture. Pumping movements might not be so noticeable to the human observer, but the feature tracking tends to slide away on defocused image elements; fixing this manually after the fact can cause a huge waste of time.
- Increase the lighting level, at least use a higher sensitivity. This helps to set a fast shutter speed plus a small aperture. Better lighting

and good exposure also help to reduce the impact of compression artifacts. If you can, also select a codec with less data reduction, better color space etc. Inevitably, we're loosing some quality through the interpolation necessary for stabilization. Plus we're loosing some quality due to color space conversion.

Elaborate movements

When the footage builds on elaborate intended movement of the camera, the process of stabilization becomes more involved – especially when there is a shift in the main area of interest within the shot. When working with many tracks and fine grained animation, it is easy to get into a situation where additional manipulations actually decrease the quality, while it might be hard to spot and locate the root cause of problems. Recommendation is to proceed systematically, starting from the general outline down to tweaking of specific aspects.

1. Understand the nature of the movements in the shot, both the intended and the accidental.
2. Track some relevant features for location.
3. Establish the basic location stabilization. This includes the decision, which feature to use for what segment of the shot. Work with the track weights to get an overall consistent movement of the weight center, in accordance with the inherent focus of the shot.
4. Define the panning movements of the virtual camera (through animation of the *Expected Position* parameter)
5. Add tracking for rotation and zoom stabilization
6. Fine tuning pass:

   Break down the whole duration of the shot into logical segments to define the intended camera movement. Then refine those segments incrementally step by step, until the overall result looks satisfactory...

Animating stabilization parameters

Animating some parameters over duration of the shot is often necessary, at least to get the final touch, including control of the scale factor to hide the dancing black borders. Unfortunately there is a **known limitation** in the current version: it is not possible to open the generic animation editors (F-curve and dope sheet) for animation data beyond the 3D scene. So, while it *is possible* to set key frames *right within the UI controls* of the stabilizer (either through pressing the `I` key or with the help of the context menu), it is not possible to manipulate the resulting curves graphically. The only way to readjust or remove a misguided keyframe is to locate the timeline to the very frame and then use the context menu of the animated UI control. (Hint: the color of the UI control changes when you have located at precisely the frame number of the keyframe)

Irregular track setup

It might not be possible to track a given feature over the whole duration of the shot. The feature might be blurred or obscured; it might even move out of sight entirely, due to deliberate camera movement. In such a situation, we need *another tracked feature* to take on it's role, and we need some *overlap time* to get a smooth transition without visible jump.

The stabilizer is able to deal with gaps and partial coverage within the given tracks. However, the basic assumption is that each track covers a single, fixed reference point whenever there is any usable/enabled data. Thus, you must not "re-use" a given track to follow several different points, rather you should disable and thus end one track, when tracking this feature is no longer feasible. You may include "gaps", when a tracking point is temporarily disabled or unavailable, but you should start a new track for each distinct new feature to be tracked.

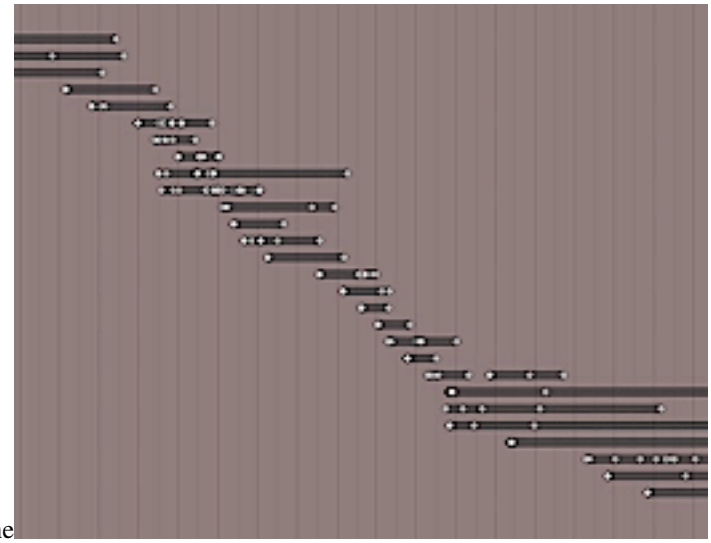
None 

Fig. 2.322: Irregular Tracks.

Each track contributes to the overall result by the degree controlled through its *Stab Weight* parameter. It is evaluated on a per frame base, which enables us to control the influence of a track by *animating* this *Stab Weight*. You may imagine the overall working of the stabilizer as if each tracking point "drags" the image through a flexible spring: When you turn down the *Stab Weight* of a tracking point, you decrease the amount of "drag" it creates. Sometimes the contribution of different tracks has to work partially counter each other. This effect might be used to cancel out spurious movement, e.g. as caused by perspective. But when, in such a situation, one of the involved tracks suddenly goes away, a jump in image position or rotation might be the result. Thus, whenever we notice a jump at the very frame where some partially covered track starts or ends, we need to soften the transition. We do so by animating the *Stab Weight* gradually down, so that it reaches zero at the boundary point. In a similar vein, when we plan a "handover" between several partially covered tracks, we define a *cross-fade* over the duration where the tracks overlap, again by animating the *Stab Weight* parameters accordingly. But even with such cross-fade smoothing, some residual movement might remain, which then needs to be corrected with the *Expected Position* or *Expected rotation* parameters. It is crucial to avoid "overshooting" movements in such a situation – always strive at setting the animation keyframes onto precisely the same frame number for all the tracks and parameters involved.

Movie Clip Tools

Track

Clip Panel

This panel currently contains the single operator *Set as background* which sets the clip currently being edited as the camera background for all visible 3D Views. If there is no visible 3D Views or the Clip Editor is open in full screen, nothing will happen.

Marker panel

Add Marker and Move Places a new marker at the position of the mouse (which is under the button in this case, not ideal but it is just how things work) and then it can be moved to the needed location. When it is moved to the desired position, `LMB` can be used to finish placing the new marker. Also, `Return` and `Spacebar` can be used to finish

placing the marker. But it is faster to use `Ctrl-LMB` to place markers directly on the footage. This shortcut will place the marker in the place you have clicked. One more feature here: until you have released the mouse button, you can adjust the marker position by moving the mouse and using the track preview widget to control how accurately the marker is placed.

Detect Features Detects all possible features on the current frame and places markers at these features. This operator does not take into account other frames, so it can place markers on features which belong to moving objects, and if camera is turning away from this shot, no markers would be placed on frames after the camera moved away.

There are several properties for this operator:

Placement Used to control where to place markers. By default, they will be added through the whole frame, but you can also outline some areas with interesting features with grease pencil and place markers only inside the outlined area. That is how the "Inside Grease Pencil" placement variant works. You can also outline areas of no interest (like trees, humans and so) and place markers outside of these areas. That is how the "Outside Grease Pencil" placement variant works.

Margin controls the distance from the image boundary for created markers. If markers are placed too close to the image boundary, they will fail to track really quickly and they should be deleted manually. To reduce the amount of manual clean-up, this parameter can be used.

Trackability Limits minimal trackability for placing markers. This value comes from the feature detection algorithm and basically it means: low values means most probably this feature would fail to track very soon, high value means it is not much such track. Amount of markers to be added can be controlled with this value.

Distance Defines the minimal distance between placed markers. It is needed to prevent markers from being placed too close to each other (such placement can confuse the camera solver).

Delete Track is a quite self-explaining operator which deletes all selected tracks.

Track panel

The first row of buttons is used to perform tracking of selected tracks (i.e. following the selected feature from frame to frame). Tracking can happen (in order of buttons):

- Backward one frame
- Backward along the sequence
- Forward along the whole sequence
- Forward one frame

This operator depends on settings from the Tracking Settings panel, which will be described later. If during sequence tracking the algorithm fails to track some markers, they will be disabled and tracking will continue for the rest of the markers. If the algorithm fails when tracking frame-by-frame, the marker is not disabled, and the most likely position of the feature on the next frame is used.

Clear After deletes all tracked and keyframed markers after the current frame for all selected tracks.

Clear Before deletes all tracked and keyframed markers before the current frame for all selected tracks.

Clear clears all markers except the current one from all selected tracks.

Join operator joins all selected tracks into one. Selected tracks should not have common tracked or keyframed markers at the same frame.

Solve

Plane Track Panel

Create Plane Track operator creates a new plane track. Four markers are needed to be selected which will form the four corners of the plane.

Solve Panel

Camera Motion operator solves the motion of camera using all tracks placed on the footage and two keyframes specified on this panel. There are some requirements:

- There should be at least eight common tracks on the both of the selected keyframes.
- There should be noticeable parallax effects between these two keyframes.

If everything goes smoothly during the solve, the average reprojection error is reported to the information space and to the clip editor header. Reprojeciton error means the average distance between reconstructed 3D position of tracks projected back to footage and original position of tracks. Basically, reprojection error below 0.3 means accurate reprojection, (0.3 - 3.0) means quite nice solving which still can be used. Values above 3 means some tracks should be tracked more accurately, or that values for focal length or distortion coefficients were set incorrectly.

The *Refine* option specifies which parameters should be refined during solve. Such refining is useful when you are not sure about some camera intrinsics, and solver should try to find the best parameter for those intrinsics. But you still have to know approximate initial values – it will fail to find correct values if they were set completely incorrectly initially.

Cleanup Panel

This panel contains a single operator and its settings. This operator cleans up bad tracks: tracks which are not tracked long enough or which failed to reconstruct accurately. Threshold values can be specified from sliders below the button. Also, several actions can be performed for bad tracks:

- They can simply be selected
- Bad segments of tracked sequence can be removed
- The whole track can be deleted

Graph View

Fig. 2.323: Graph View.

Introduction

The graph or curves view has numerous purposes based on the color of the lines. The red and green lines on the graph show you the speed of the trackers at a given frame. Green is vertical movement, Red is horizontal. Therefore the first frames will always be at zero. The blue line is the line that comes out when you click on the film strip is the average per frame error. This curve is available only after pressing camera solve and is not editable. This is the one line that you want to be as flat as possible and as closer to zero as you can. The high points will show you where in your shot you are having inaccurate tracking.

Usage

The curves are useful to see if particular trackers are moving differently than the average. A line that spikes from the rest of the curve usually means a tracking error.

You can manually edit the curve by selecting a point in the curve and dragging it or deleting, that will affect the corresponding tracker on that particular frame.

Dope Sheet View

Introduction

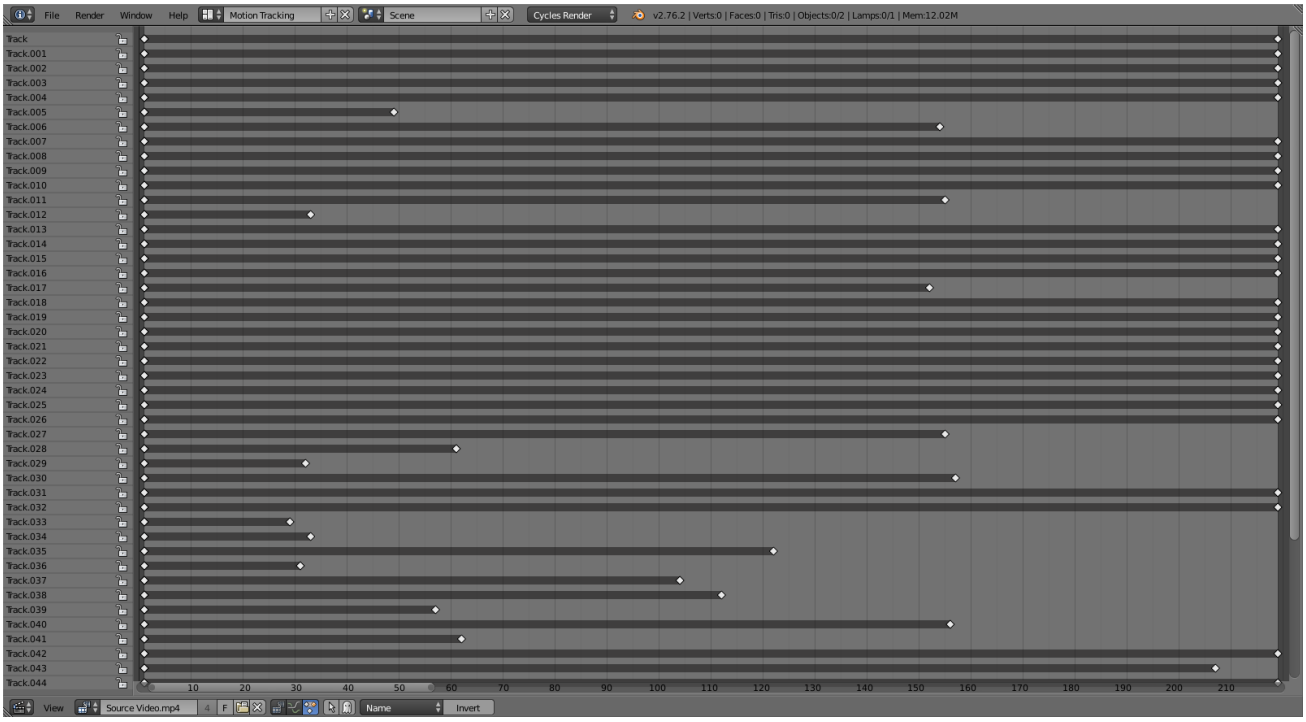


Fig. 2.324: Dope Sheet View.

The dope sheet view is used to visualize motion tracking data, it implemented as separate view of the movie clip editor just like the *Graph View*. To support this in a nice way, you must toggle between modes specified to a view in the whole area of the movie clip editor. Hence, to display a curve or dope sheet view, the editor must be split into two, with one switched to the curve or dope sheet view.

Usage

Currently the dope sheet view is for visualization and does not have any tools to actually edit data. It displays channels for selected tracks and each channel visualizes tracked segments of tracks as dark bars and keyframed positions of tracks as small diamonds.

By default, this view sorts tracks in alphabetical order, but here is list of all possible sort orders with the Sort Order option:

- Name: sort selected tracks in alphabetical order based on their names.
- Longest: sort tracks by longest tracked segment length.
- Total: sort tracks by overall amount of frames.
- Average Error: sort tracks by their average reprojection error after solving camera or object motion.

There is also an option called *Invert* to change the sort order from ascending to descending, next to the sort order method option.

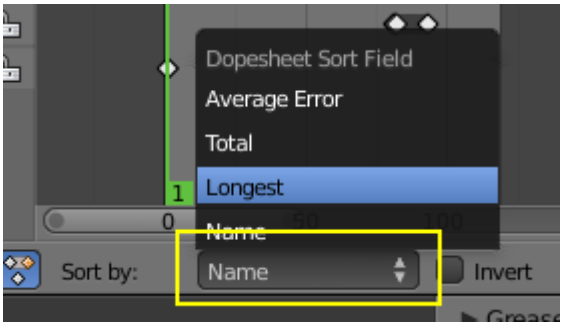


Fig. 2.325: Sort Channels Order.

Masking

Introduction

Masks have many purposes. They can be used in a motion tracking workflow to mask out, or influence a particular object in the footage. They can be used for manual rotoscoping to pull a particular object out of the footage, or as a rough matte for green screen keying. Masks are independent from a particular image of movie clip, and so they can just as well be used for creating motion graphics or other effects in the compositor. These masks can also be used in other places in Blender.

Editing Masks

Masks can be created in the image and movie clip editors, by changing the mode from View to Mask in the header. This will add various tools and properties to the editor panels, while hiding others that are not needed for interacting with masks. The tools and panels available to edit masks are the same in both editors, with the exception that linking masks to motion tracking data is only possible in the movie clip editor.

Once set to Mask mode, a Mask data-block can be added. Any image, movie clip, render or compositing result can be used as a backdrop to draw masks over. To get interactive feedback on the resulting mask, a Mask node can be connected directly to a Viewer node in the compositor, which will then keep updating the compositing result while editing.

S-Curves

The curve type used for creating mask splines is almost a Bézier curve, but with some differences. The curve needed to support feathering in a way that stuck to the curve as you edited it, for ease of editing an animation. These are called S-Curves.

Besides the handles, every control point also has points that define the feather between the current point and the next point on the spline. Each feather point is stored in UV space, where U means position across spline segment, and V means distance between main spline and feather points.

This allows for deforming the main spline in almost any way, and the feather will be updated automatically to reflect that change. For example if there is just rotation of the spline, feather would stay completely unchanged. If one point's feather is moved, the other feathers will be automatically stretched uniformly along that segment and the overall shape will be almost the same as artists would want it to be.

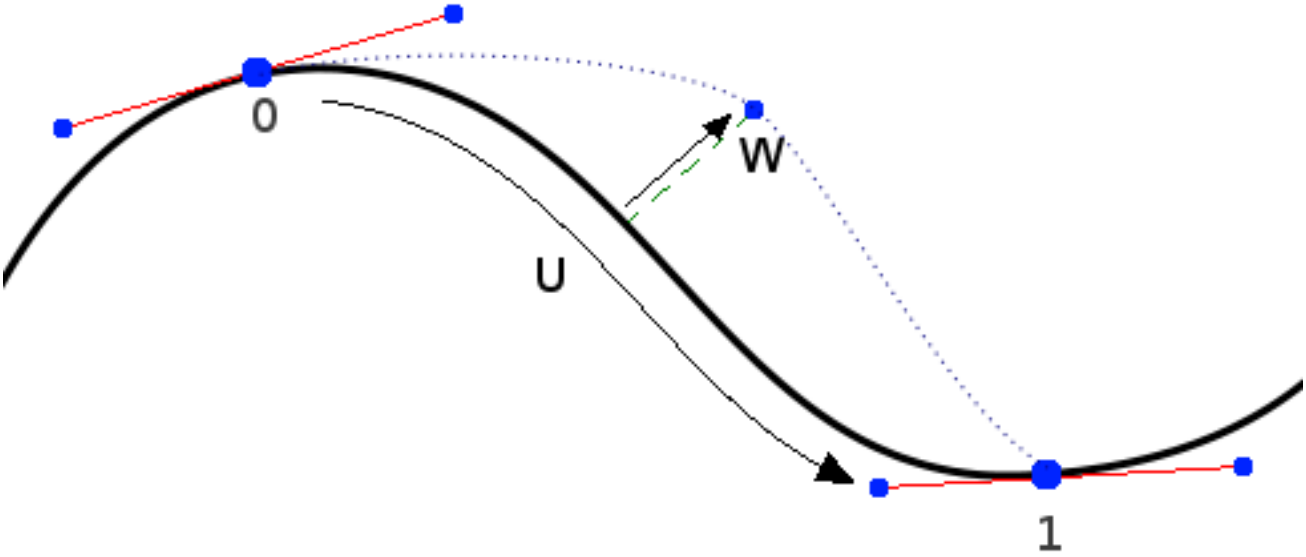


Fig. 2.326: S- Curve Explained.

Control Points

Editing of mask splines happens in a way similar to editing Bézier curves or paths in GIMP or other curve editors: control points are added to define the spline itself, and handles of different types are used to create smooth bends. This makes it possible to define a mask with few points to easily follow an object in footage.

- `Ctrl-LMB` is used to place new control points and define handle orientations.
- `Alt-C` to close the mask by joining the last control point to the first.
- Existing control points can be translated, scaled and rotated with the usual G, S, R shortcuts.
- `X` or Delete removes control points.

Selection

The usual selection and hide/reveal tools are available:

- `A` toggle select all.
- `B` , `C` border and circle Select.
- `Ctrl-L` select linked from selection, L: select linked with mouse.
- `Ctrl-Alt-LMB` lasso select.
- `H` hide selected, `Shift-H` hide unselected, `Alt-H` reveal.

Curve Handles

- `Alt-C` cycle toggle spline, to create a close curve or open it again.
- `V` set handle type for selected spline points.
- `Ctrl-N` make normals (handle directions) consistent.
- Switch Direction handle directions in/out.

Feather

It is possible to control feather of mask, including a way to define non-linear feather. Linear feather is controlled by a slider, non-linear feather is controlled in the same curve-based way to define feather falloff.

- `Shift-LMB` is used to define a feathering outline curve. To create an initial feather, sliding from a spline control point outside or inside will create and position feather points. After this `Shift-LMB` will insert new feather point and mouse sliding can be used to move them around.
- `Alt-S` will scale the feather size.

Animating

Masks can be driven over the time so that they follow some object from the footage, e.g. a running actor. This animation can be done in several ways:

- Control points can be parented to motion tracks. This way is the main way to interact with masks in a motion tracking workflow.
- Keyframe animation of control points using a shape keying system. This can be useful when there are not enough good feature points to track in the footage, or the mask is not based on footage.

For animation more complex mask shapes, it is also possible to do more high level animation:

- Splines and mask layers can be animated as a whole, instead of individual control points.
- Masks can be parented to motion tracking data. Works for both individual mask point parenting and for overall spline. To select motion track to be parented to use `Ctrl-RMB`. To parent selected mask points to active motion track use `Ctrl-P`.
- Mask animation timing can be edited from the Dope Sheet. Here there is a mask mode where mask keyframes can be selected and edited.

Shape Keys

Masks can be animated with shape keyframing. This works on the level of mask layers, so inserting a shape key will keyframe all the splines and points contained in it.

- `I` will insert a shape key for the active mask layer at the current frame
- `Alt-I` will clear the shape key for the active mask layer at the current frame.
- Feather Reset Animation: Resets the feather offset across all animated frames.
- Re-Key Points of Selected Shapes: Re-interpolate selected points on across the range of keys selected in the dope sheet.

2.2.4 Nodes/Logic

Text Editor

Blender has a *Text Editor* among its editor types, accessible via the *Editor type* menu, or the shortcut `Shift-F11`.

Header

The newly opened Text editor is gray and empty, with a very simple header Fig. *Text header.*

Editor type The standard editor selection button.

Menus Editors *Menus*.

Fig. 2.327: Text header.

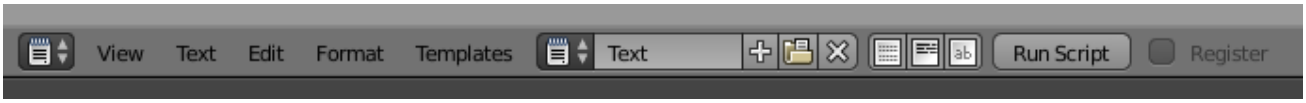


Fig. 2.328: Text header with a text loaded.

Text Data-block menu. Once a text is selected or newly created, the header changes. Fig. *Text header with a text loaded.*

Show The following three buttons toggle display options.

Line numbers, word-wrap text, syntax highlighting

Run Script/ Script Node Update Executes the text as a Python script `Alt-P`. See *Script and Templates*.

Register Todo.

Label This Label shows, if the text is saved internal or external and if there are unsaved changes to an external file.

Menus

View

Bottom of File Moves the view and cursor to the end of the text.

Top of File Moves the view and cursor to the start of the text.

Text

Create Text Block Creates a new internal text.

Open Text Block Loads a text, a *File Browser* appears `Alt-O`.

Reload Reopens (reloads) the current buffer (all non-saved modifications are lost) `Alt-R`.

Save Saves an already open file `Alt-S`.

Save As Saves unsaved text as a text file, a *File Browser* appears `Shift-Ctrl-Alt-S`.

Make Internal Stores the text inside the blend-file.

Run Script Executes the text as a Python script `Alt-P`. See *Script and Templates*.

Edit

Undo `Ctrl-Z`.

Redo `Ctrl-Shift-Z`.

Cut Cuts out the marked text into the text clipboard `Ctrl-X`.

Copy Copies the marked text into the text clipboard `Ctrl-C`.

Paste Pastes the text from the clipboard at the cursor location in the Text editor `Ctrl-V`.

Duplicate Line Duplicates the current line `Ctrl-D`.

Move line(s) up Swaps the current line with the above.

Move line(s) down Swaps the current line with the below.

Select Select Line, Select All.

Jump Shows the Jump pop-up, which lets you select a line number where to jump to.

Find... Shows the Find panel in the Properties Region.

Text Auto Complete Shows a selectable list of Python commands and words already used in the text.

Text To 3D Object One Object, One Object per line

Format

Indent Indents the selection `Tab`.

Unindent Un-indents the selection `Shift-Tab`.

Comment Turns the selected lines into a Python comment.

Uncomment Uncomments the selected lines.

Convert Whitespace To Space, To Tab.

Template See *Script and Templates*.

Python, OpenShading Language

Script and Templates

The most notable keystroke is `Alt-P` which makes the content of the buffer being parsed by the internal Python interpreter built into Blender. Before going on it is worth noticing that Blender comes with a fully functional Python interpreter built in, and with a lots of Blender-specific modules, as described in the *Scripting & Extending Blender* section.

The *Text Editor* has now also some dedicated Python scripts, which add some useful writing tools, like a class/function/variable browser, completion... You can access them through the Template menu in the header.

Main View

Typing on the keyboard produces text in the text buffer. As usual, pressing, dragging and releasing `LMB` selects text.

Tip: Usages for the Text editor

The Text editor is handy also when you want to share your blend-files with others. The Text editor can be used to write in a `README` text explaining the contents of your blend-file. Be sure to keep it visible when saving!

Node Editor

Introduction

The *Node Editor* is used to work with node-based work flows. The node tree type can be changed using the buttons in the node editor header. However, here we will only give an overview of what the *Node Editor* is. In the list below it shows a list of different types of node trees and where each is documented.

Icon	Name	Documentation
	Material Nodes	Because there are two different render engines documentation is split between *Blender Internal* and *Cycles*.
	Composite Nodes	Documentation can be found in the *Compositing* section.
	Texture Nodes	Texture Nodes are covered in the *Blender Internal* docs.

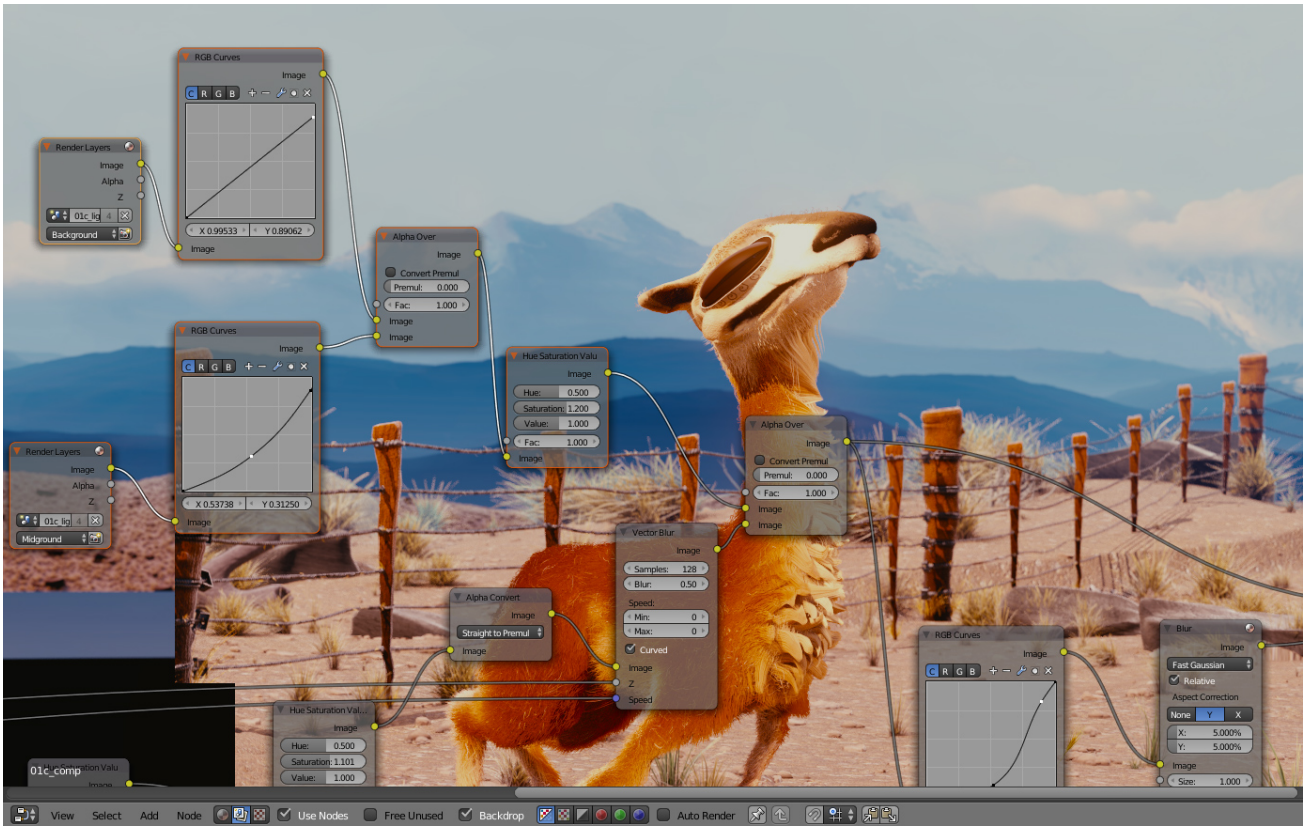


Fig. 2.329: The Node Editor.

After choosing what node context you want to use, you have to enable nodes with the *Use Nodes* button.

Interface

Header

The *Header* contains various menus, buttons and options, partially based on the current node tree type.

Fig. 2.330: Common Node Header Options.

View This menu changes your view of the editor.

Select This menu allows you to select a node or groups of nodes.

Add This menu allows you to add nodes.

Node To do things with selected nodes, akin to vertices.

Material, Compositing or Texture buttons Nodes are grouped into three categories, to see the list see *Node Tree Types*.

Use Nodes Tells the render engine to use the node map in computing the material color or rendering the final image, or not. If not, the map is ignored and the basic render of the material tabs or scene is accomplished.

Use Pinned This button tells the render engine to use pinned node tree.

Go to Parent button This button allows you go to parent node tree.

Snap Toggle snap mode for node in the Node Editor.

Snap Node Element Selector This selector provide the follow node elements for snap:

Grid (default) Snap to grid of the Node Editor.

Node X Snap to left/right node border.

Node Y Snap to top/bottom node border.

Node X/Y Snap to any node border.

Snap Target Which part to snap onto the target.

Closest Snap closest point onto target.

Center Snap center onto target.

Median Snap median onto target.

Active Snap active onto target.

Copy Nodes This button allows you copy selected nodes to the clipboard.

Paste Nodes This button allows you paste nodes from the clipboard to the active node tree.

Tool Shelf

The *Tool Shelf* is a context-sensitive region, natively containing tools for the Grease Pencil and buttons for adding nodes. The Tool Shelf is organized using tabs.

Properties Region

The *Properties Region* contains properties for the current selected node as well as node editor specific settings.

Navigating

Navigating the node editor is done with the use of both mouse movement and keyboard shortcuts.

Pan **`MMB`** Move the view up, down, left and right

Zoom **`Ctrl-MMB`**, **`Wheel`** Move the camera forwards and backwards.

View Selected **.**

View All `Home`

Node Editor Actions

When the cursor is in the area, several standard Blender hotkeys and mouse actions are available, including:

Search... (add menu) Brings up a pop-up menu, allowing you to search the available nodes.

Undo `Ctrl-Z`

Redo `Ctrl-Y` or `Ctrl-Shift-Z` – You can use this if you used "undo" a bit too often.

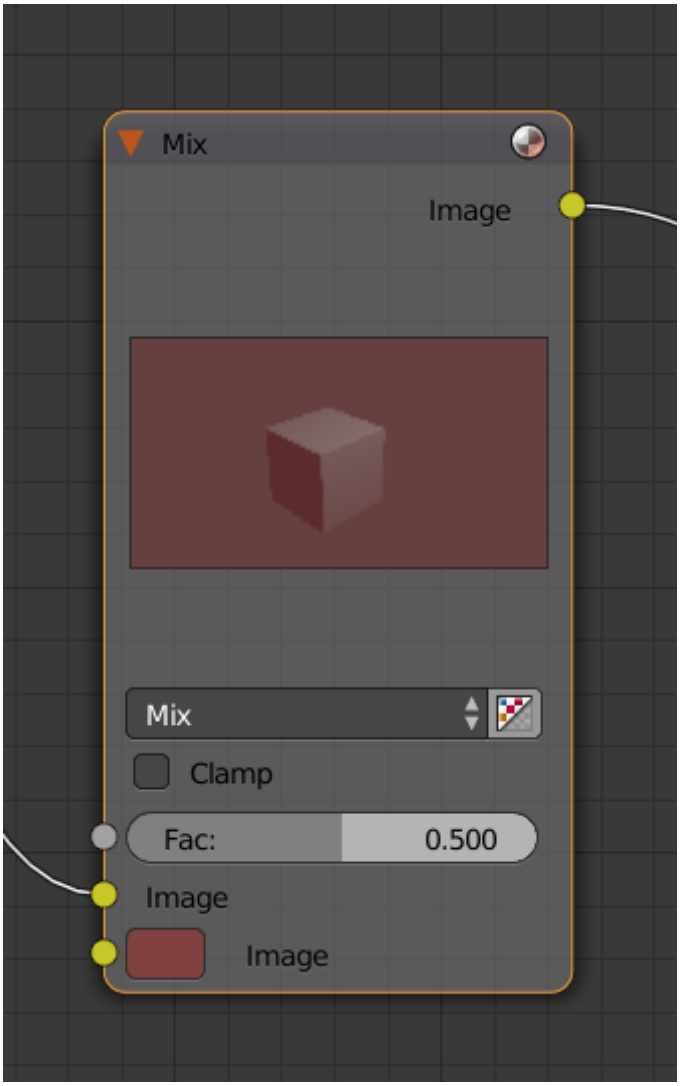
Nodes

Introduction

Todo.

Node Parts

All nodes in Blender are based off of a similar construction. This applies to *any type of node*. These parts include the Title, Sockets, Preview and more.



Title

The *Title* shows the name/type of the node. It can be overridden by changing the value of Label in the *Node* section of the *Properties Region* `N`. On the left side of the title is the *collapse toggle* which can be used to collapse the node this can also be done with `H`.

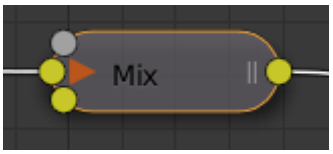


Fig. 2.331: How a node appears when collapsed.

Sockets

The *Sockets* input and output values from the node. They appear as little colored circles on either side of the node. Unused sockets can be hidden with `Ctrl-H`. There are two functions of sockets; *inputs* and *outputs*.

Each socket is color-coded depending on what type of data it handles.

Color (Yellow) Indicates that color information needs to be input or will be output from the node. This may or may not include an alpha channel.

Numeric (Grey) Indicates numeric values information. It can either be a single numerical value or a so-called "value map". (You can think of a value map as a grayscale-map where the different amount of bright/dark reflects the value for each point). If a single value is used as an input for a "value map" socket, all points of the map are set to this same value. Common use: Alpha maps and value options for a node.

Vector (Blue) Indicates vector, coordinate and normal information.

Shader (Green) Used for shaders in *Cycles*

Inputs

The *Inputs* are located on bottom left side of the node, and provide the data the node needs to perform its function. Each input socket, except for the green shader input, when disconnected, has a default value which can be edited via a color, numeric, or vector interface input. In the screen shot of the node above, the second color option is set by a color interface input.

Properties

Many nodes have settings which can affect the way they interact with inputs and outputs. Node settings are located below the outputs and above any inputs.

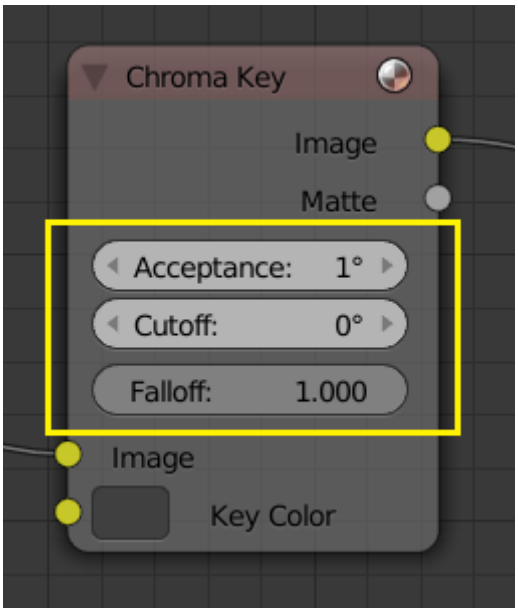


Fig. 2.332: An example of the controls on the chroma key node.

Preview

On some nodes this shows a preview image of how the output data for a certain channel will appear. Usually it shows color data.

The preview can be toggled using the icon on the very top right hand corner of the node, next to the title.

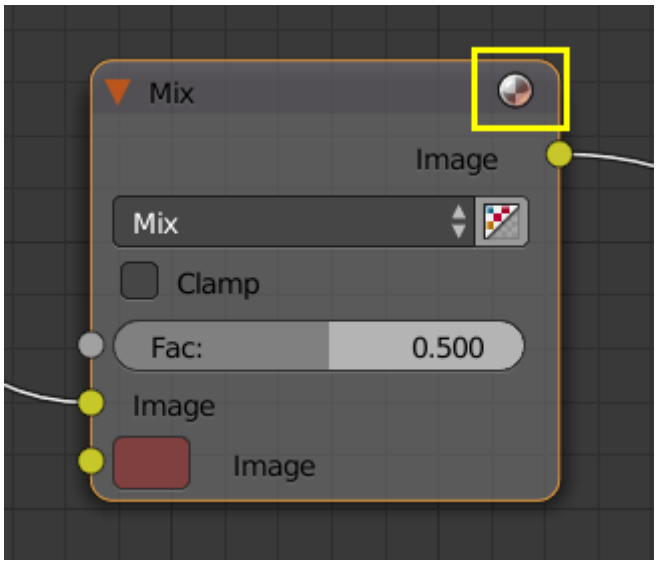


Fig. 2.333: How a node appears without the preview.

Outputs

The *Outputs* are located on the top right side of the node, and can be connected to the input of nodes further down the node tree.

Selecting

Border Select `B` starts the bounding box selection process. Position your cursor and `LMB` click & drag to select a set of nodes.

Cut connections (lasso) `Ctrl-Alt-LMB` click & drag starts a lasso selection, **but** when you let up the mouse button, all threads (connections) within the lasso are broken.

(De)select All `A`

Inverse `Ctrl-I`

Select Linked From `L`

Select Linked To `Shift-L`

Select Groued `Shift-G`

Activate Same Type Previous `Shift-]`

Activate Same Type Next `Shift-[`

Find Node `Ctrl-F`

Select multiple `Shift-LMB` or `Shift-RMB` used for multiple node selection.

Editing

Translate `G` to move the current selection around.

Rotate `R`

Resize `S`

Duplicate `Shift-D`

Delete `X` or `Delete` deletes the selected node(s).

Delete with Reconnect `Ctrl-X`

Join in new Frame `Ctrl-J`

Remove from Frame `Alt-P`

Make Links `F`

Make and Replace Links `Shift F`

Cut Links `Ctrl-LMB`

Detach Links

Edit Group `Tab`

Ungroup `Alt-G`

Make Group `Ctrl-G`

Group Insert

Hide `H`

Toggle Node Mute `M`

Toggle Node Preview `Shift-H`

Toggle Hidden Node Sockets `Ctrl-H`

Toggle Node Options

Collaps and Hide Unused Sockets

Read Render-Layers `Ctrl-R`

Read Full Sample Layers `Shift-R`

Using Nodes

Adding Nodes

Nodes are added in two ways to the node editor:

- By using the Tool Shelf which has buttons for adding nodes, organized with tabs.
- By using the *Add* menu `Shift-A`.

Arranging Nodes

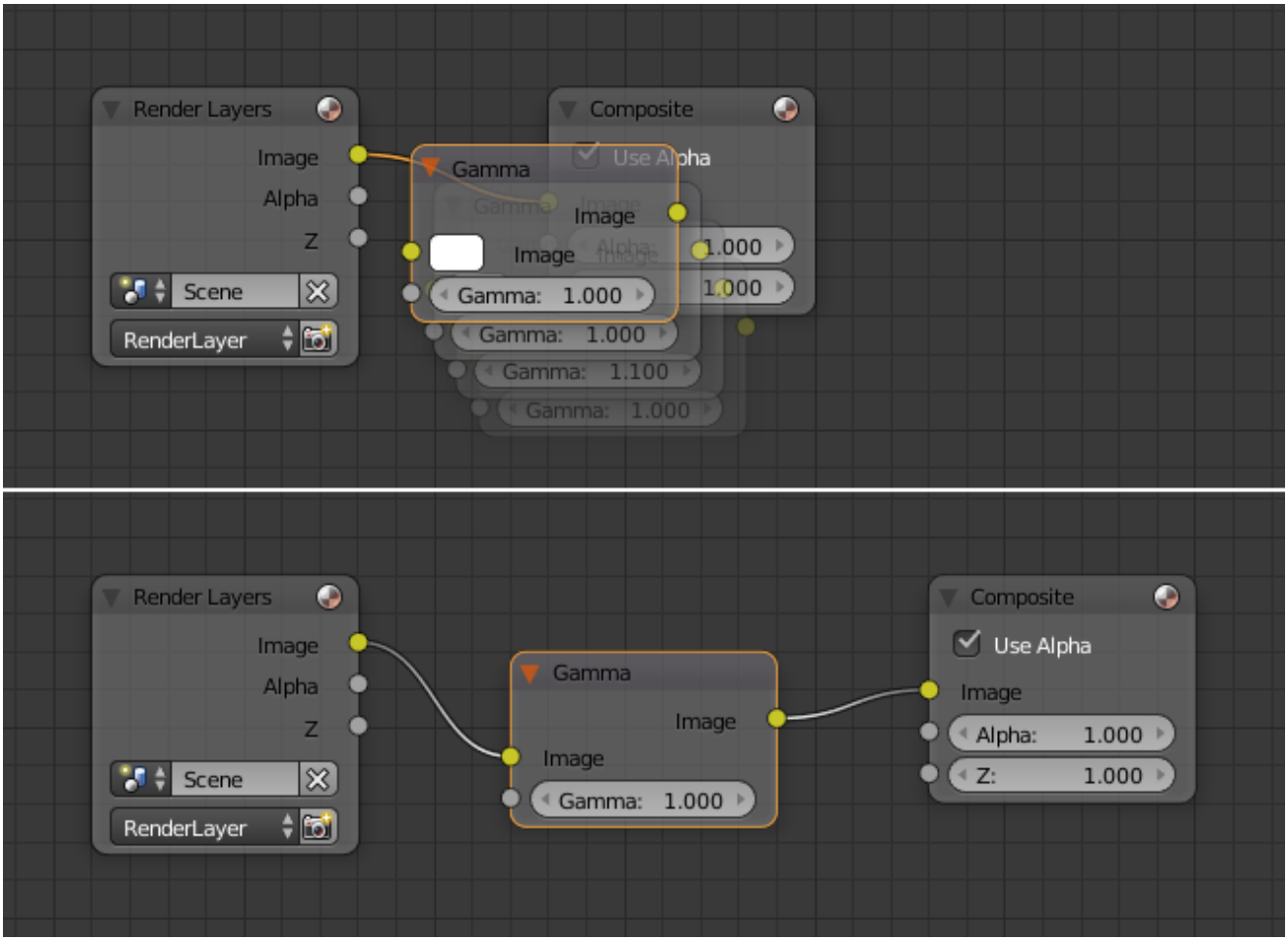
In general, try to arrange your nodes within the view such that the image flows from left to right, top to bottom. Move a node by click and drag it around. A node can be clicked almost anywhere to start dragging.

Auto-offset

Auto-offset is a feature that helps organizing node layouts interactively without interrupting the user workflow. When you drop a node with at least one input and one output socket onto an existing connection between two nodes, auto-offset will, depending on the direction setting, automatically move the left or right node away to make room for the new node.

Auto-offset is enabled by default, but it can be disabled from the node editor header.

You can toggle the offset direction while you are moving the node by pressing `T`.


Render Layers
Image
Alpha
Z
Scene
RenderLayer
Gamma
Image
Image
Gamma: 1.000
Gamma: 1.000
Gamma: 1.100
Gamma: 1.000
Composite
Use Alpha
Image
1.000
1.000
Render Layers
Image
Alpha
Z
Scene
RenderLayer
Gamma
Image
Image
Gamma: 1.000
Composite
Use Alpha
Image
Alpha: 1.000
Z: 1.000

The offset margin can be changed using the *Auto-offset Margin* setting in the editing section of the User Preferences.

Example Video:

Connecting nodes

LMB-click on a socket and drag. You will see a line coming out of it: This is called a *link*.

Keep dragging and connect the link to an input socket of another node, then release the LMB.

While multiple links can route out of an output socket, only a single link can be attached to an input socket.

To reposition the outgoing links of a node, rather than adding a new one, hold Ctrl while dragging from an output socket. This works for single as well as for multiple outgoing links.

Disconnecting nodes

To break a link between sockets Ctrl–LMB-click in an empty area, near the link you want to disconnect, and drag: You will see a little cutter icon appearing at your mouse pointer. Move it over the link itself, and release the LMB.

Duplicating a node

Click LMB or RMB on the desired node, press Shift–D and move the mouse away to see the duplicate of the selected node appearing under the mouse pointer.

Note: When you duplicate a node, the new node will be positioned *exactly* on top of the node that was duplicated. If you leave it there (and it is quite easy to do so), you can **not** easily tell that there are *two* nodes there! When in doubt, grab a node and move it slightly to see if something's lurking underneath.

Properties

In the properties region.

Node Panel

Name A unique node identifier.

Label Nodes can be given a title by modifying the text field.

Color Panel

Color Presets Colors saved as a preset for re-use in other nodes.

Color Color of the node background. Node colors can can be used to provide a visual cue.

Node Groups

Both material and composite nodes can be grouped. Grouping nodes can simplify the node network layout in the node editor, making your material or composite node setup easier to work with. Grouping nodes also creates what are called NodeGroups (inside a blend-file) or NodeTrees (when appending).

Conceptually, "grouping" allows you to specify a *set* of nodes that you can treat as though it were "just one node". You can then re-use it one or more times in this or some other blend-file(s).

As an example: If you have created a material using nodes that you would like to use in another blend-file, you could simply append the material from one blend-file to another. However, what if you would like to create a new material, and use a branch from an existing material node network? You could re-create the branch. Or you could append the material to the new blend-file, then cut and paste the branch that you want into the new material. Both of these options work, but are not very efficient when working across different blend-files. A better method of re-use, for either material node branches or composite node networks, would be to create groups of nodes.

Once a group has been defined, it becomes an opaque object; a reusable software component. You can (if you choose) ignore exactly how it is "defined", and simply use it as many times as you like.

Grouping Nodes

To create a node group, in the node editor, select the nodes you want to include, then press `Ctrl-G`, *Group* → *Make Group*, `Shift-A`. A node group will have a green title bar. All of the selected nodes will now be contained within the group node. Default naming for the node group is "NodeGroup", "NodeGroup.001" etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful. When appending node groups from one blend file to another, Blender does not make a distinction between material node groups or composite node groups, so it is recommended some naming convention, that will allow you to easily distinguish between the two types.

Note: What **not** to include in your groups (all types of Node editors)

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not** include:

Input nodes If you include a source node in your group, you will end up having the source node appearing *twice*: once inside the group, and once outside the group in the new material node-network.

Output node If you include an output node in the group, there will not be an output socket available *from* the group!

Editing Node Groups

With a group node selected, `Tab` expands the node to a frame, and the individual nodes within it are shown. You can move them around, play with their individual controls, re-thread them internally, etc. just like you can if they were a normal part of the editor view. You will not be able, though, to thread them to a node outside the group; you have to use the external sockets on the side of the group node. To add or remove nodes from the group, you need to ungroup them.

Ungrouping Nodes

The `Alt-G` command removes the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread internal nodes to other nodes in your workspace.

Appending Node Groups

Once you have appended a NodeTree to your blend-file, you can make use of it in the node editor by pressing `Shift-A`, *Add* → *Group*, then select the appended group. The "control panel" of the Group is the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

Logic Editor

The Logic Editor provides the main method of setting up and editing the game logic for the various actors (i.e. objects) that make up the game. The logic for the objects which are currently selected in the associated 3D View are displayed as logic bricks, which are shown as a table with three columns, showing sensors, controllers, and actuators, respectively. The links joining the logic bricks conduct the pulses between sensor-controller and controller-actuator.

To give you a better understanding of the Logic Editor, the image below shows a typical editor content in which the major components have been labeled. We will look at each one individually.

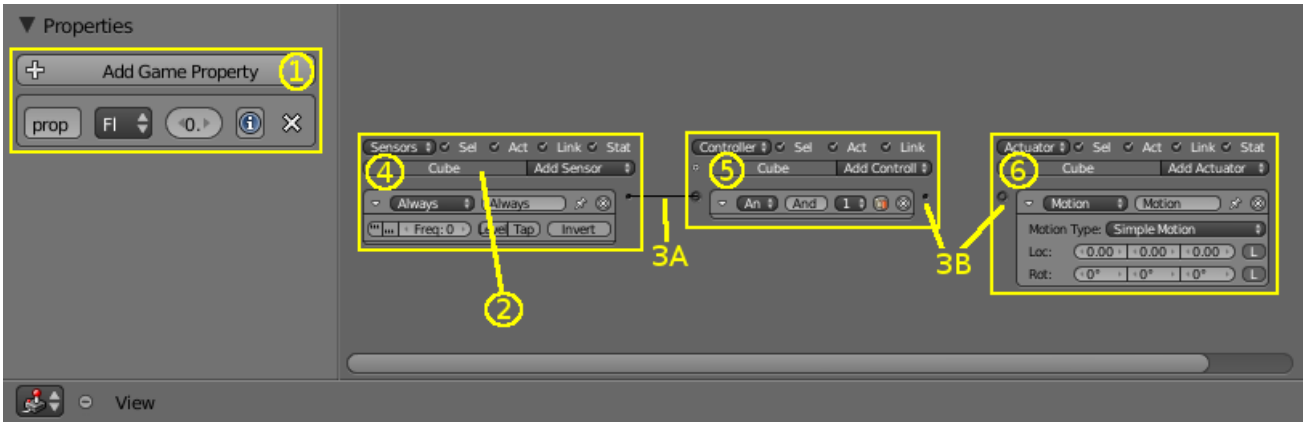


Fig. 2.334: The different parts of the Logic Editor.
1) Property Region, 2) Object Name, 3a) Links, 3b) Link socket, 4) Sensor column, 5) Controller Column, 6) Actuator Column.

Main view

Object Name This box shows the name of the object which owns the logic bricks below.

Links Links (3A) indicate the direction of logical flow between objects. Link lines are drawn by LMB dragging from one Link socket (3B) to another. Links can only be drawn from Sensors to Controllers, or from Controllers to Actuators. You cannot directly link Sensors to Actuators; likewise, Actuators cannot be linked back to Sensors (however, special actuator and sensor types are available to provide these connections).

Sending nodes (the black circles found on the right-hand side of Sensors and Controllers) can send to multiple Reception nodes (the white circles found on the left-hand side of Controllers and Actuators). Reception nodes can likewise receive multiple links.

Links can be created between logic bricks belonging to different objects. To delete a link between two nodes, LMB drag between the two nodes.

Sensor Column

This column contains a list of all sensors owned by the active object (and any other selected objects). New sensors for the active object are created using the "Add Sensor" button. For a more in-depth look at the content, layout and available operations in this area, see *Sensors*.

Controller Column

This column contains a list of all controllers owned by the active object (and any other selected objects). New controllers for the active object are created using the "Add Controller" button, together with the creation of states for the active object. For a more in-depth look at the content, layout, and available operations in this area, see *Controllers*.

Actuator Column

This column contains a list of all actuators owned by the active object (and any other selected objects). New actuators for the active object are created using the "Add Actuator" button. For a more in-depth look at the content, layout, and available operations in this area, see *Actuators*.

Property Region

Game properties are like variables in other programming languages. They are used to save and access data associated with an object. Several types of properties are available. Properties are declared by clicking the *Add Game Property* button in this region. For a more in-depth look at the content, layout and available operations in this region, see *Properties*.

2.2.5 Settings

Properties Editor

The *Properties Editor* is used to edit data and properties for the *Active Scene* and the *Active Object*.

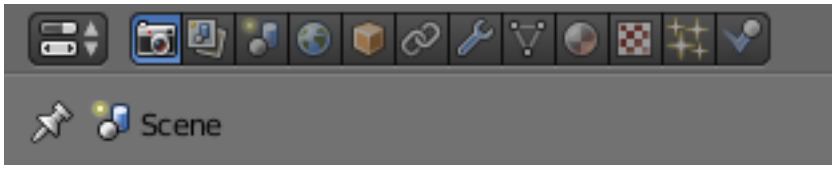


Fig. 2.335: Properties editor top part.

Tabs

The Properties editor shows several tabs, which can be chosen via the icon row in the header. The tabs are documented in their own manual sections, the links are listed below.

Scene/Render

These tabs are used to add features, and to change properties for the Active Scene.

- *Render* and Settings: *Blender Internal*, *Cycles*
- *Render Layers*
- *Scene*
- World: *Blender Internal*, *Cycles*

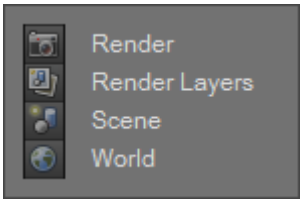


Fig. 2.336: Scene/Render tabs.

Object & Object Data

These tabs are used to add features, and to change properties for the Active Object (and other active elements, material, curve, etc.).

The Object Data tabs shown depend on what type of object was selected last (The Active Object).

- *Object*
- *Constraints*
- *Modifiers*

´ *Mesh Curve Surface Metaball Text Empty*

´ *Armature Bones Bone Constraints Lattice*

´ *Speaker* Camera: *Blender Internal*, *Cycles* Lamp: *Blender Internal*, *Cycles*

´ Material: *Blender Internal*, *Cycles* Texture: *Blender Internal*, *Cycles Particles Physics*

Main View

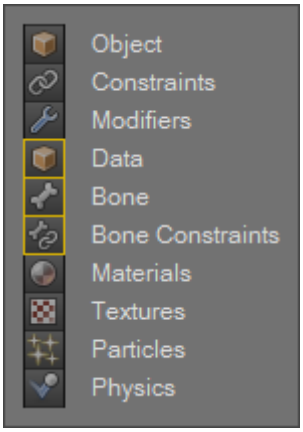


Fig. 2.337: Object Data tabs.

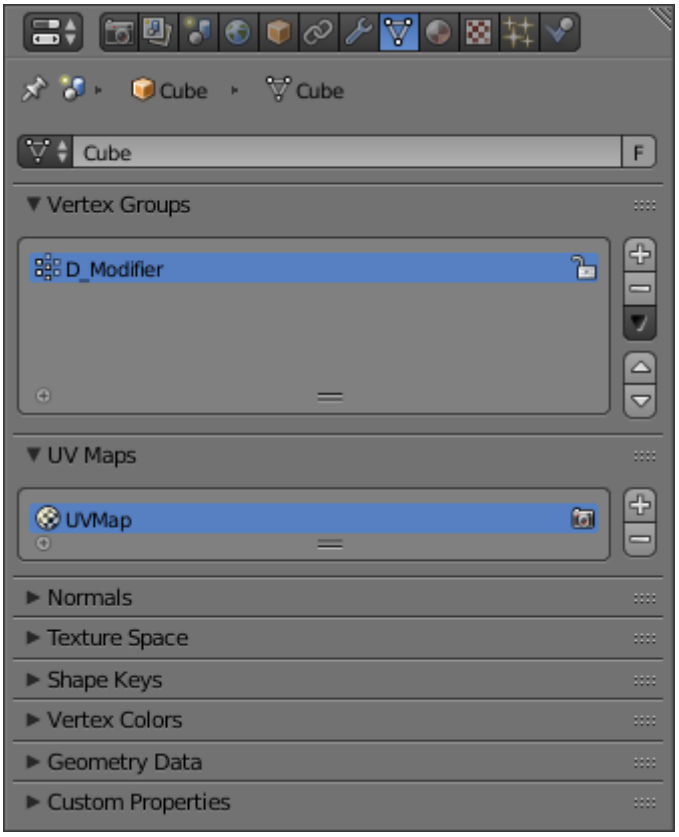


Fig. 2.338: The Properties Editor with the Mesh tab selected.

At the top of the each tab a list of icons explains the context in which the properties are being edited. In the example above, the mesh *Cube* is linked to the object *Cube* which is linked to the scene *Scene*.

By toggling the pin symbol on the left side on and off, Blender can be told to display only the selected property or to follow context.

Outliner

Fig. 2.339: The Outliner editor.

The *Outliner* is a list that organizes data in the blend-file. i.e. the scene data and also the User Preferences.

Usage

- View the data in the scene.
- Select and deselect objects in the scene.
- Hide or show an object in the scene.
- Enable or disable selection (to make an object "unselectable" in the 3D View).
- Enable or disable the rendering of an object.
- Delete objects from the scene.
- Unlink data (equivalent to pressing the *X* button next to the name of a data-block).
- Easily select which render layer to render.
- Easily select which render pass to render (for example, you can choose to render just the *Specular* pass).

Tree View

Each row in the *Outliner* shows a data-block. You can click the plus-sign to the left of a name to expand the current data-block and see what other data-blocks it contains.

You can select data-blocks in the *Outliner*, but this will not necessarily select the data-block in the scene. To select the data-block in the scene, you have to activate it.

Selecting and Activating

Single selection does not require any pre-selection: just work directly with `LMB` (and/or `RMB` - contextual menu, see below) *inside* the name/icon area.

When you select an object in the list this way, it is selected and becomes the active object in all other 3D Views.

Activating a data-block

To "activate" the data-block with `LMB` on the *icon* of the data-block. Activating the data-block will automatically switch to the relevant mode. For example, activating the mesh data of the cube will select the cube and enter *Edit Mode* while activating the object data of the cube will select the cube and enter *Object Mode* (see right).

Selecting a group of data-blocks

Useful when you want to select/deselect a whole bunch of data-blocks. For this you must prepare the selection using, to your liking:

- `RMB` or `LMB`,
- `Shift-RMB` or `Shift-LMB`,
- `RMB` and drag or `LMB` and drag,

all *outside* the name/icon area. Those pre-selected have their line in a lighter color. You then can (de)select them with a `RMB` *on* the name/icon area, which brings on a context menu (see bellow).

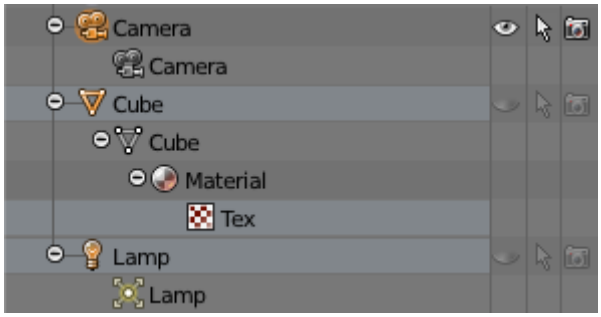


Fig. 2.340: Selection of a data-block.

Context menu

Show the context menu for a data-block with RMB on the icon or name. Depending on the type of the pre-selected data-block(s), you will have all or part of the following options:

- *Select*.
- *Deselect*.
- *Delete* X.
- *Unlink* - To unlink a data-block from its "owner" (e.g., a material from its mesh).
- *Make Local* - To create a "local" duplicate of this data-block.

Note: Some data-block types will not have a context menu at all!

Object-level Restrictions

The three following toggles, in the right side of the *Outliner* editor, are only available for objects:

Visibility (eye icon) Toggles the visibility of the object in the 3D View. V will toggle this property for any objects that are selected in the *Outliner*.

Selectability (mouse cursor icon) This is useful for if you have placed something in the scene and do not want to accidentally select it when working on something else. S will toggle this property for any objects that are selected in the *Outliner*.

Rendering (camera icon) This will still keep the object visible in the scene, but it will be ignored by the renderer. R will toggle this property for any objects that are selected in the *Outliner*.

Header

View Menu

Sort Alphabetically Sort the entries alphabetically.

Show Restriction Columns Toggles the three columns of *Object-level Restrictions*.

Show Active Centers the Tree View to selected object ..

Show/Hide One Level Expand one level down in the tree NumpadPlus and NumpadMinus to collapse.

Show Hierarchy To collapse all levels of the tree Home.

Display Mode

The editors header has a select menu that let you filter what the Outliner should show. It helps to narrow the list of objects so that you can find things quickly and easily.

All Scenes Shows *everything* the *Outliner* can display (in all scenes, all layers, etc.)

Current Scene Shows everything in the current scene.

Visible Layers Shows everything on the visible (currently selected) layers in the current scene. Use the *layer* buttons to make objects on a layer visible in the 3D View.

Selected Lists the object(s) that are currently selected in the 3D View. See *selecting in the 3D View* for more information.

Active Lists only the active (often last selected) object.

Same Types Lists only those objects in the current scene that are of the same types as those selected in the 3D View.

Groups Lists only *Groups* and their members.

Sequence Lists *data-block* that are used by the *Sequencer*.

Blender File Lists all data in the current blend-file.

Data-Blocks Lists every *data-block* along with any properties that they might have.

User Preferences Lists options that can be found in the *User Preferences* along with some other settings.

Orphan Data Lists *data-blocks* which are unused and/or will be lost when the file is reloaded.

Searching

You can search the view for data-blocks, by using Search field in the header of the *Outliner*, The *Search* menu lets you toggle the following options:

- Case Sensitive Matches Only
- Complete Matches Only

Example

User Preferences

Introduction

This chapter explains how to change Blender's default configuration with the *User Preferences* editor.

The Blender *User Preferences* editor contains settings to control how Blender behaves.

Open User Preferences

To open the *User Preferences* editor go to *File* → *User Preferences*.

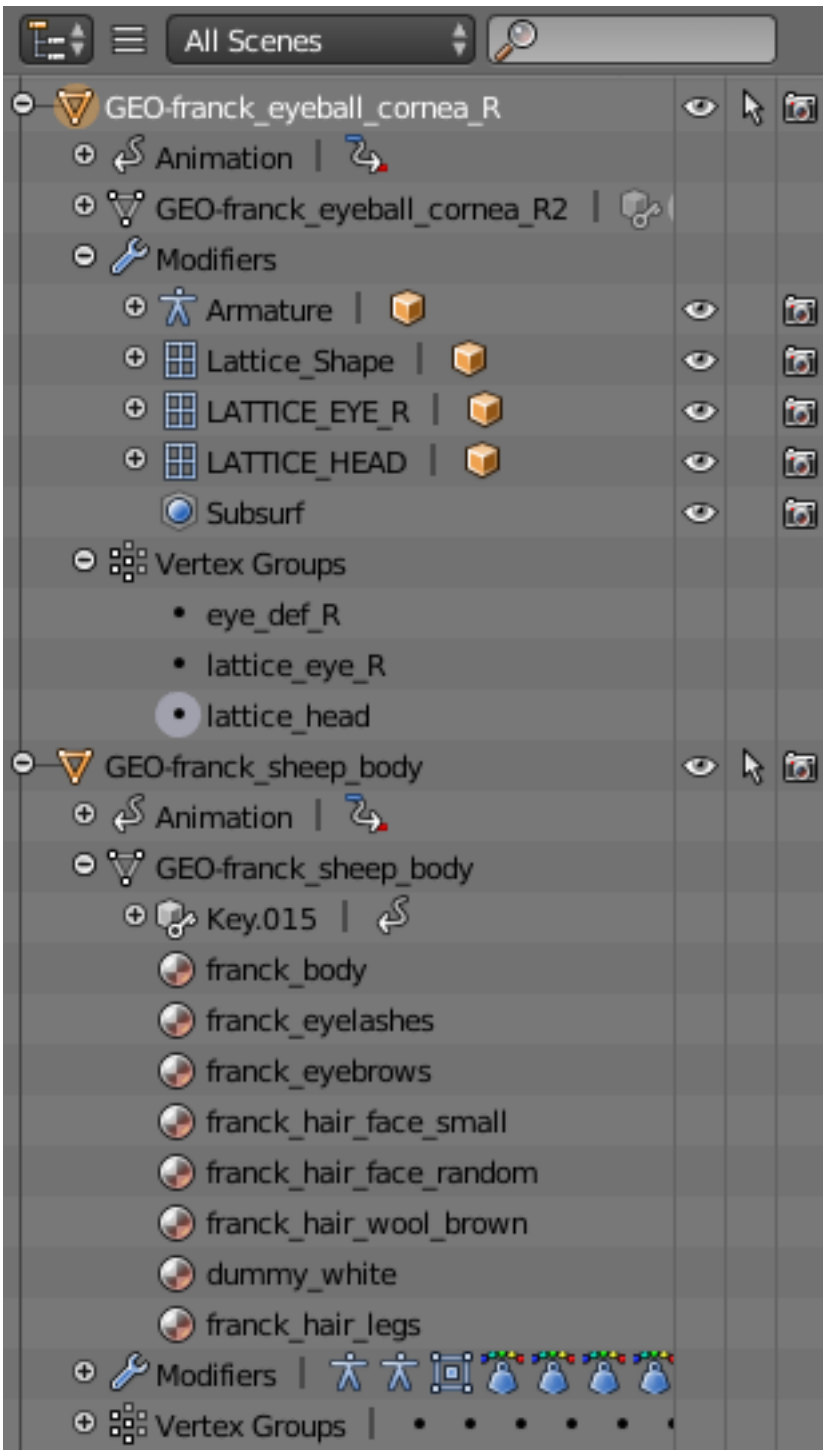
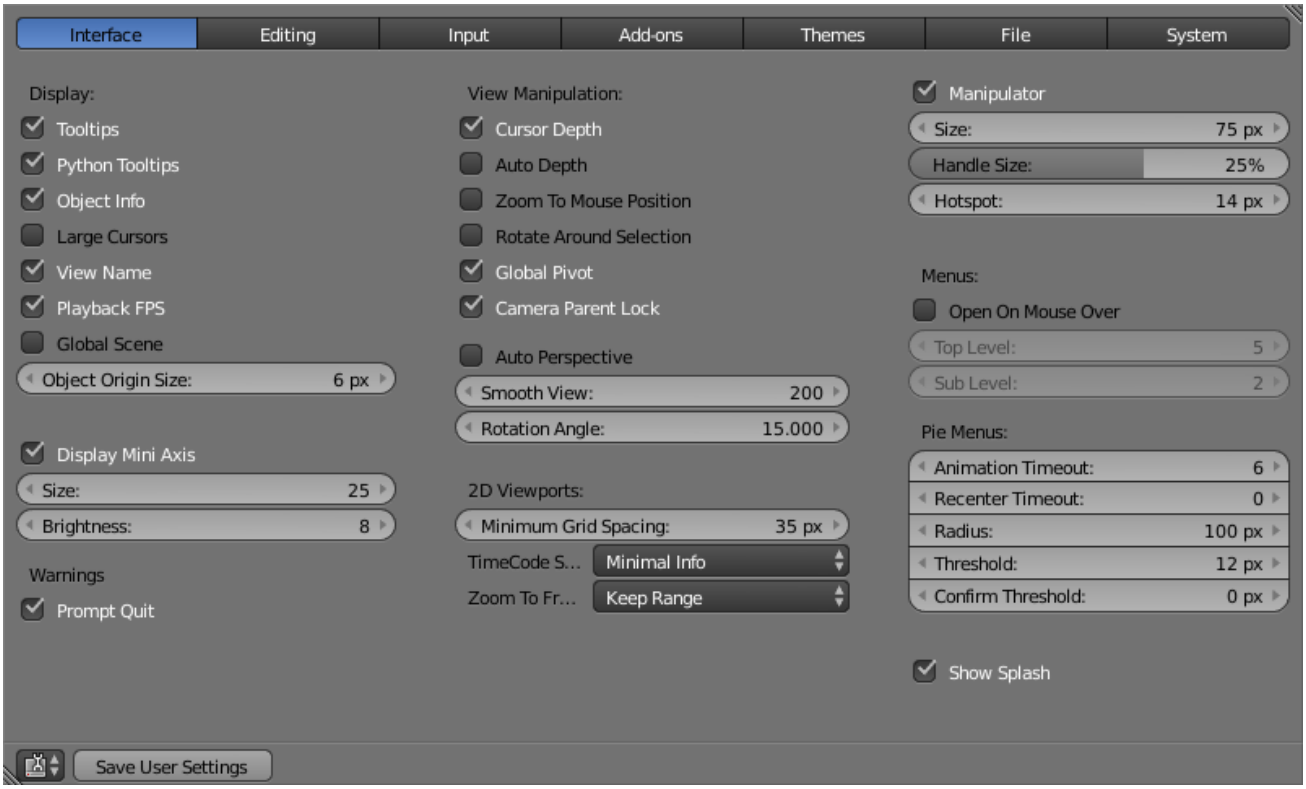


Fig. 2.341: The Outliner with different kind of data.



Configure

Now that you have opened the User Preferences editor, you can configure Blender to your liking. At the top of the editor, the available options are grouped into seven tabs:

Interface Change how UI elements are displayed and how they react.

Editing Control how several tools will interact with your input.

Input Customize how Blender reacts to the mouse and keyboard as well as define your own keymap.

Add-ons Manage Blender's *Add-ons*, allowing you to access features not built-in as well as install new features.

Themes Customize interface appearance and colors.

File Configure auto-save preferences and set default file paths for blend-files, rendered images, and more.

System Set resolution, scripting console preferences, sound, graphics cards, and internationalization.

Save the new preferences

Once you have set your preferences, you will need to manually save them, otherwise the new configuration will be lost after a restart. Blender saves its preferences to *userpref.blend* in your user folder (see next section, "Load Factory Settings", for details).

In the *User Preferences* editor, click on the *Save User Settings* button in the bottom left. This will save all of the new preferences.

Load Factory Settings

Go to *File* → *Load Factory Settings* then save the preferences via the *User Preferences* editor.

Hint: It can be valuable to make a backup of your preferences in the event that you lose your configuration.

See the *directory layout* section to see where your preferences are stored.

Startup File

Reference

Mode: All modes
Menu: *File* → *Save Startup File*
Hotkey: `Ctrl-U`

When you start Blender or start a new project with the menu entry *File* → *New*, a new scene is created from the default scene included with Blender.

This default scene can instead be your own customized setup.

To change the default scene, make all of the desired changes to the current scene or current file and *File* → *Save Startup File*.

Tabs

Interface

Interface configuration lets you change how UI elements are displayed and how they react.

Display

Tooltips When enabled, a tooltip will appear when your mouse pointer is over a control. This tip explains the function of what is under the pointer, gives the associated hotkey (if any) and the Python function that refers to it.

Python Tooltips Displays a property's Python information below the tooltip.

Object Info Display the active Object name and frame number at the bottom left of the 3D View.

Large Cursors Use large mouse cursors when available.

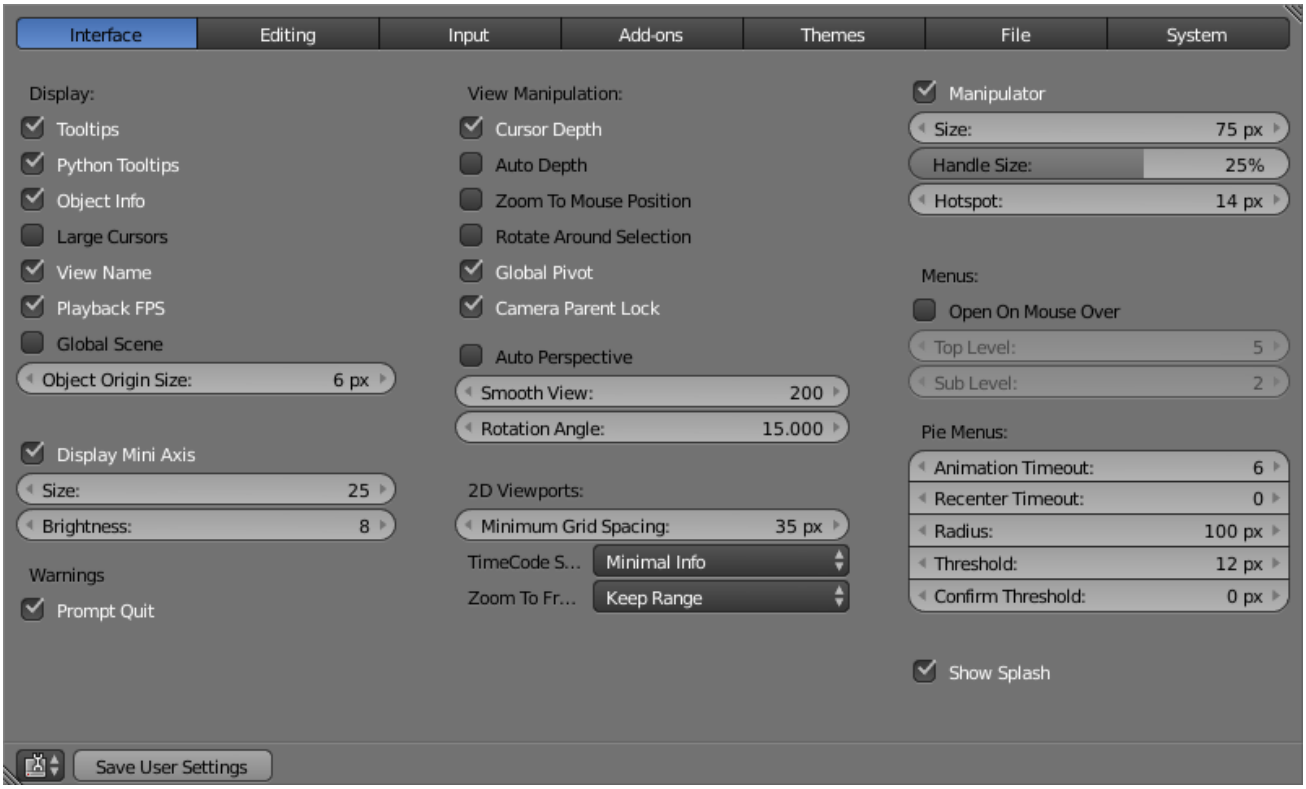
View Name Display the name and type of the current view in the top left corner of the 3D View. For example: *User Persp* or *Top Ortho*.

Playback FPS Show the frames per second screen refresh rate while an animation is played back. It appears in the viewport corner, displaying red if the frame rate set cannot be reached.

Global Scene Forces the current scene to be displayed in all screens (a project can consist of more than one scene).

Object Origin Size Diameter of 3D Object centers in the view port (value in pixels from 4 to 10).

Display Mini Axis Show the mini axis at the bottom left of the viewport.



Size Size of the mini axis.

Brightness Adjust brightness of the mini axis.

Warnings

Prompt Quit When exiting Blender, a pop-up will ask you weather or not you really want to quit (currently only available on MS-Windows).

View Manipulation

Cursor Depth Use the depth under the mouse when placing the cursor.

Auto Depth Use the depth under the mouse to improve view pan, rotate, zoom functionality. Useful in combination with *Zoom To Mouse Position*.

Zoom to Mouse Position When enabled, the mouse pointer position becomes the focus point of zooming instead of the 2D window center. Helpful to avoid panning if you are frequently zooming in and out.

Rotate Around Selection The selected object becomes the rotation center of the viewport. When there is no selection the last selection will be used.

Hint: This may seem ideal behavior, however, it can become problematic with larger objects such as a terrain-mesh, where the center is not necessarily your point of interest.

Global Pivot Lock the same rotation/scaling pivot in all 3D Views.

Camera Parent Lock When the camera is locked to the view and in fly mode, transform the parent rather than the camera.

Auto Perspective Automatically to perspective Top/Side/Front view after using User Orthographic. When disabled, Top/Side/Front views will retain Orthographic or Perspective view (whichever was active at the time of switching to that view).

Smooth View Length of time the animation takes when changing the view with the numpad (Top/Side/Front/Camera...). Reduce to zero to remove the animation.

Rotation Angle Rotation step size in degrees, when `Numpad4`, `Numpad6`, `Numpad8`, or `Numpad2` are used to rotate the 3D View.

2D Viewports

Minimum Grid Spacing The minimum number of pixels between grid lines in a 2D (i.e. top orthographic) viewport.

Time Code Style Format of Time Codes displayed when not displaying timing in terms of frames. The format uses '+' as separator for sub-second frame numbers, with left and right truncation of the timecode as necessary.

Zoom To Frame Type How zooming to frame focuses around current frame.

Keep Range Todo.

Seconds Todo.

Keyframes Todo.

Manipulator Turns manipulators on and off.

Size Diameter of the manipulator.

Handle Size Size of manipulator handles, as a percentage of the manipulator radius (*size*/ 2).

Hotspot Hotspot size (in pixels) for clicking the manipulator handles.

Menus

Open on Mouse Over Select this to have the menu open by placing the mouse pointer over the entry instead of clicking on it.

Menu Open Delay Time for the menu to open.

Top Level Time delay in 1/10 second before a menu opens (*Open on Mouse Over* needs to be enabled).

Sub Level Same as above for sub menus (for example: *File* → *Open Recent*).

Pie Menus

Animation Timeout Length of animation when opening Pie Menus.

Recenter Timeout The window system tries to keep the pie menu within the window borders. Pie menus will use the initial mouse position as center for this amount of time, measured in 1/100ths of a second. This allows for fast dragged selections.

Radius Size of the Pie Menu.

Threshold Distance from center before a selection can be made.

Confirm Threshold Distance threshold after which selection is made (zero disables).

Splash

Show Splash Display the *Splash Screen* when starting Blender.

Editing

These preferences control how several tools will interact with your input.

Link Materials To

Fig. 2.342: Example for a Mesh.

To understand this option properly, you need to understand how Blender works with Objects. Almost everything in Blender is organized in a hierarchy of data-blocks. A data-block can be thought of as containers for certain pieces of information. For example, the Object data-block contains information about the Object's location while the Object Data "ObData" data-block contains information about the mesh.

A material may be linked in two different ways:

Object Data Any created material will be created as part of the Object Data data-block.

Object Any created material will be created as part of the Object data-block.

Read more about Blender's Data System

Fig. 2.343: A material linked to Object Data (left) and Object (right).

New Objects

Enter Edit Mode If selected, Edit Mode is automatically activated when you create a new object.

Align To

World New objects align with world coordinates.

View New object align with view coordinates.

Undo

Global Undo This enables Blender to save actions done when you are **not** in *Edit Mode*. For example, duplicating Objects, changing panel settings or switching between modes.

> **Warning:** While disabling this option does save memory, it stops the *Redo Panel* from functioning, also preventing tool options from being changed in some cases.
>
> For typical usage, its best to keep this enabled.

Steps Number of Undo steps available.

Memory Limit Maximum memory usage in Mb (0 is unlimited).

Read more about Undo and Redo options

Grease Pencil

Eraser Radius The size of the eraser used with the grease pencil.

Manhattan Distance The minimum number of pixels the mouse should have moved either horizontally or vertically before the movement is recorded. Decreasing this should work better for curvy lines.

Euclidian Distance The minimum distance that mouse has to travel before movement is recorded.

Default Color The default color for new Grease Pencil layers.

Simplify Stroke This turns on the post-processing step of simplifying the stroke to remove about half of current points in it. It is only relevant when not drawing straight lines.

Read more about Grease Pencil

Playback

Allow Negative Frame Time Cursor can be set to negative frames with mouse or keyboard. When using *Use Preview Range*, this also allows playback.

Node Editor

Auto-offset Margin Margin to use for *offsetting nodes*.

Animation Editors

F-Curve Visibility Opacity that un-selected *F-Curves* stand out from the *Graph Editor*.

Keyframing

In many situations, animation is controlled by keyframes. The state of a value (e.g. location) is recorded in a keyframe and the animation between two keyframes is interpolated by Blender.

Visual Keying When an object is using constraints, the objects property value does not actually change. *Visual Keying* will add keyframes to the object property, with a value based on the visual transformation from the constraint.

Only Insert Needed This will only insert keyframes if the value of the propery is different.

Auto Keyframing Enables *Auto Keyframe* by default for new scenes.

Show Auto Keying Warning Displays a warning at the top right of the *3D View*, when moving objects, if *Auto Keyframe* is on.

Only Insert Available This will only add keyframes to channel F-Curves that already exist.

New F-Curve Defaults

Interpolation Controls the default *Interpolation* for newly created keyframes.

Handles Controls the default *Handle* for newly created F-Curves.

XYZ to RGB Color for X, Y or Z animation curves (location, scale or rotation) are the same as the color for the X, Y and Z axis.

Transform

Release confirm Dragging LMB on an object will move it. To confirm this (and other) transforms, a LMB is necessary by default. When this option is activated, the release of LMB acts as confirmation of the transform.

Sculpt Overlay Color

This color button allows the user to define a color to be used in the inner part of the brushes circle when in sculpt mode, and it is placed as an overlay to the brush, representing the focal point of the brush influence. The overlay color is visible only when the overlay visibility is selected (clicking at the *eye* to set its visibility), and the transparency of the overlay is controlled by the alpha slider located at the *Option tab* → *Overlay panel* in the tool shelf.

Duplicate Data

The *Duplicate Data* check-boxes define what data is copied with a duplicated Object and what data remains linked. Any boxes that are checked will have their data copied along with

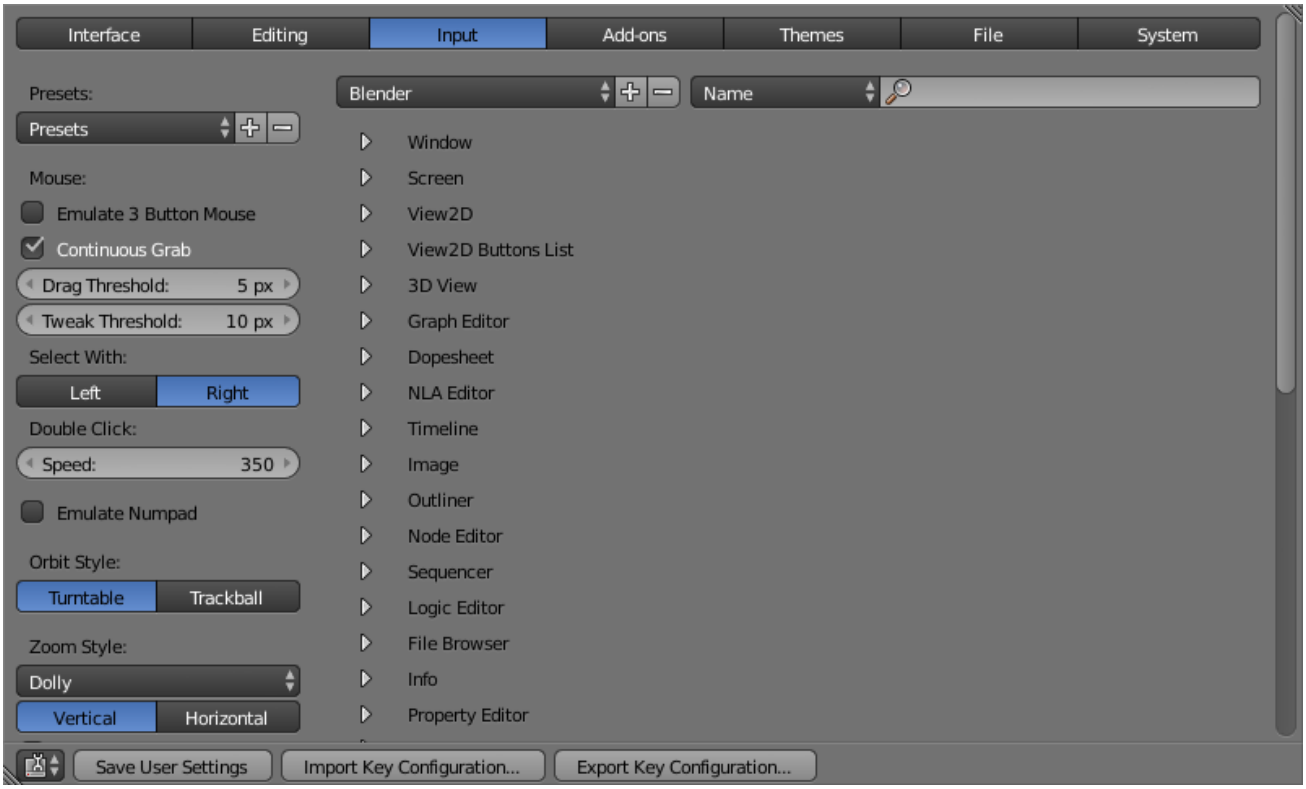
the duplication of the Object. Any boxes that are not checked will instead have their data linked from the source Object that was duplicated.

For example, if you have Mesh checked, then a full copy of the mesh data is created with the new Object, and each mesh will behave independently of the duplicate. If you leave the mesh box unchecked then when you change the mesh of one object, the change will be mirrored in the duplicate Object.

The same rules apply to each of the check-boxes in the 'Duplicate Data' list.

Input

In the Input preferences, you can customize how Blender reacts to the mouse and keyboard as well as define your own keymap.



Interaction

Interaction Presets Presets that allow Blender to act like other software or you personal preference.

Mouse

Emulate 3 Button Mouse Blender can be configured to work with pointing devices which do not have a MMB (such as a two-button mouse, Apple's single-button mouse, or laptop touch-pad). The functionality of the three mouse buttons will then be emulated with key/mouse button combinations as shown in the table below.

Table 2.8: Shortcuts for supported mouse hardware

3-button Mouse	2-button Mouse	Apple Mouse
LMB	LMB	LMB (mouse button)
MMB	Alt-LMB	Alt-LMB (Option/Alt key + mouse button)
RMB	RMB	Cmd-LMB (Command/Apple key + mouse button)

Mouse/Keyboard combinations referenced in this manual can be expressed with the combinations shown in the table. For example:

- MMB drag becomes Alt-LMB drag.
- Shift-Alt-RMB becomes Shift-Alt-Cmd-LMB on a single-button mouse.

Continuous Grab This feature is used to prevent the problem where an action such as grabbing or panning a view, is limited by your screen bounds.

This is done by warping the mouse within the view.

Note: Cursor warping is only supported by *relative* input devices (mouse, trackball, trackpad).

Graphics tablets, however, typically use *absolute* positioning, this feature is disabled when a tablet is being used.

This is detected for each action, so the presence of a tablet will not disable *Continuous Grab* for mouse cursor input.

Drag Threshold The number of pixels that a User Interface element has to be moved before it is recognized by Blender.

Select With You can choose which button is used for selection (the other one is used to place the 3D cursor).

Double Click The time in ms for a double click to be recognized.

Note: The Mouse emulate option is only available if *Select With* is set to *Right*.

Numpad Emulation

The Numpad keys are used quite often in Blender and are not the same keys as the regular number keys. If you have a keyboard without a Numpad (e.g. on a laptop), you can tell Blender to treat the standard number keys as Numpad keys. Just check *Emulate Numpad*.

View Manipulation

Orbit Style Select how Blender works when you rotate the 3D View by default when holding MMB.

Turntable Rotates the view keeping the horizon horizontal.

This behaves like a potter's wheel or record player where you have two axes of rotation available, and the world seems to have a better definition of what is "Up" and "Down" in it.

The drawback to using the *Turntable* style is that you lose some flexibility when working with your objects. However, you gain the sense of "Up" and "Down" which can help if you are feeling disoriented.

Trackball Is less restrictive, allowing any orientation.

Zoom Style Choose your preferred style of zooming in and out with `Ctrl-MMB`

Scale *Scale* zooming depends on where you first click in the view. To zoom out, hold `Ctrl-MMB` while dragging from the edge of the screen towards the center. To zoom in, hold `Ctrl-MMB` while dragging from the center of the screen towards the edge.

Continue The *Continue* zooming option allows you to control the speed (and not the value) of zooming by moving away from the initial click point with `Ctrl-MMB`. Moving up from the initial click-point or to the right will zoom out, moving down or to the left will zoom in. The further away you move, the faster the zoom movement will be. The directions can be altered by the *Vertical* and *Horizontal* radio buttons and the *Invert Zoom Direction* option.

Dolly *Dolly* zooming works similarly to *Continue* zooming except that zoom speed is constant.

Zoom Axis The axis of the `MMB` to use for zooming.

Vertical Moving up zooms out and moving down zooms in.

Horizontal Moving left zooms in and moving right zooms out.

Invert Zoom Direction Inverts the Zoom direction for *Dolly* and *Continue* zooming.

Invert Wheel Zoom Direction Inverts the direction of the mouse wheel zoom.

View Navigation

Navigation Mode The default navigation mode for `Shift-F` in the 3D View.

Walk

Reverse Mouse Inverts the mouse's Y movement.

Mouse Sensitivity Speed factor for when looking around, high values mean faster mouse movement.

Teleport Duration Interval of time warp when teleporting in navigation mode.

Walk Speed Base speed for walking and flying.

Speed Factor The multiplication factor for the speed boost.

Gravity Simulates the effect of gravity when walking.

View Height The distance from the ground floor to the camera when walking.

Jump Height The maximum height of a jump.

Fly

There no additional options for fly mode.

NDOF Device

Pan Sensitivity The overall sensitivity for panning in the 3D View.

Orbit Sensitivity The overall sensitivity for orbiting in the 3D View.

Deadzone The threshold for the amount of movement needed from the device's rest position for Blender to interrupt that movement.

Navigate Method Navigation style for the viewport.

Free Uses the full 6-degrees of freedom.

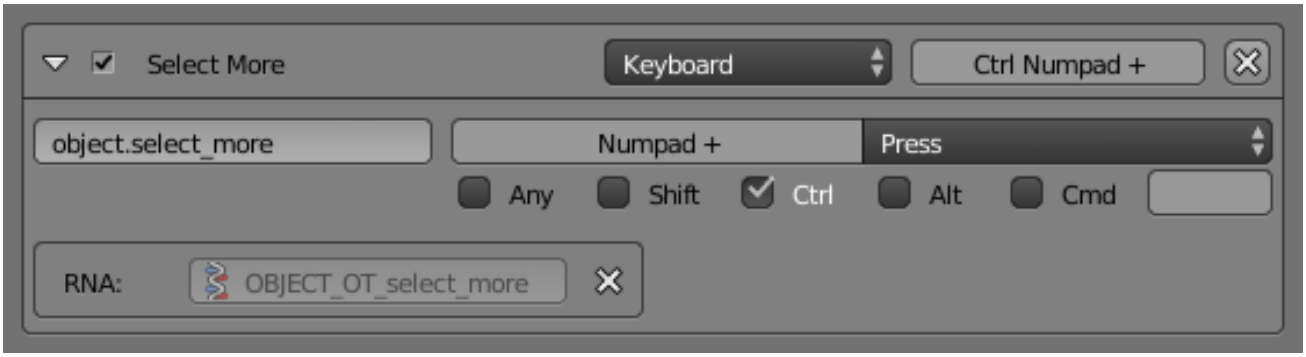
Orbit Orbit about the view center.

Rotate Method Rotation style for the viewport.

Turntable Rotates the view keeping the horizon horizontal.

Trackball Is less restrictive, allowing any orientation.

Keymap Editor



The Keymap editor lets you change the default Hotkeys. You can change keymaps for each of Blender's editors.

Keymap Presets A list of predefined keymaps.

1. Select the keymap you want to change and click on the white arrows to open up the keymap tree.
2. Select which Input will control the function.

- Keyboard: Only hotkey or combo hotkey `E`, `Shift-E`.
- Mouse: Left/middle/right click. Can be combined with `Alt`, `Shift`, `Ctrl`, `Cmd`.
- NDOF: ToDo.
- Tweak: Click and drag. Can also be combined with the four previous keys.
- Text input: Use this function by entering a text.
- Timer: Used to control actions based on a time period. e.g. By default, *Animation Step* uses "Timer 0", *Smooth View* uses "Timer 1".

3. Change hotkeys as you want. Just click on the shortcut input and enter the new shortcut.

   If you want to restore the default settings for a keymap, just click on the *Restore* button at the top right of this keymap.

Tip: Instead of deleting the default keymap to create yours, you can just add a new *Preset* for both the mouse and keyboard.

Export/Import Key Configuration

In some cases, you may need to save your configuration in an external file (e.g. if you need to install a new system or share your keymap configuration with the community). To do this, simply press the *Export Key Configuration* button found in the header. After doing so a the file browser will open to choose where to store the configuration. The *Import Key Configuration* button installs a keymap configuration that is on your computer but not in Blender.

The exported keymap will only contain keymaps and categories that have been modified by the user. In addition, add-ons may register keymaps to their respective functions, however, these keymaps are not exported unless changed by the user. This exported file may be thought of as a *"keymap delta"* instead of a full keymap export.

Add-ons

The Add-ons tab lets you manage secondary scripts, called "Add-ons" that extends Blender's functionality. In this tab you can search, install, enable and disable Add-ons.

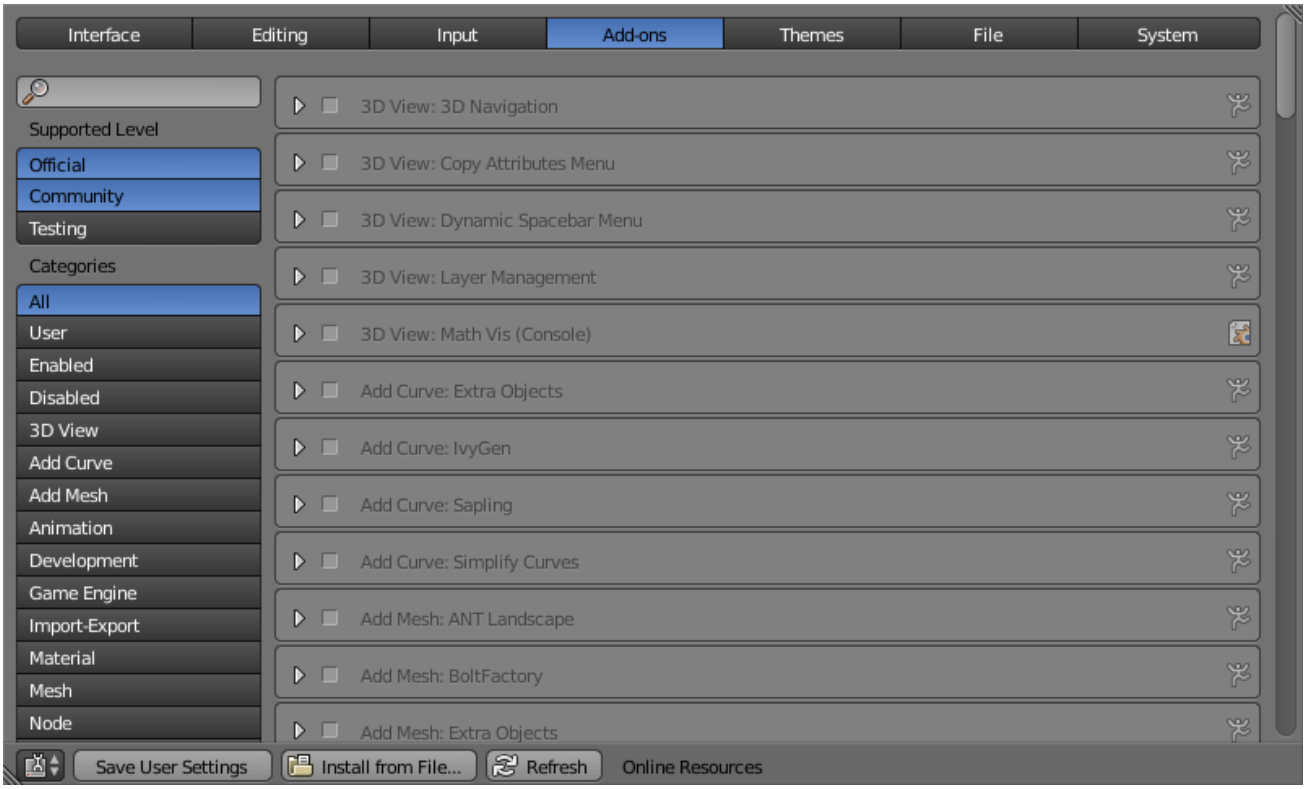


Fig. 2.344: Add-ons tab in the User Preferences.

Searching

Blender comes with some useful Add-ons already, ready to be enabled. But you can also add your own, or any interesting ones you find on the web.

Filtering

Support Level Blender's add-ons are split into two groups depending on who writes/supports them:

- Official: Add-ons that are written by Blender developers.
- Community: Add-ons that are written by people in the Blender community.

Categories Add-ons are divided into categories by what areas of Blender they affect.

Enabling and Disabling

Enable and disable an add-on by checking or unchecking the box on the right of the add-on you chose, as shown in the figure.

Fig. 2.345: Enabling an Add-on.

The add-on functionality should be immediately available. If the Add-on does not activate when enabled, check the *Console window* for any errors, that may have occurred.

Add-on Information

You can click the arrow at the left of the add-on box to see more information, such as its location, a description and a link to the documentation. Here you can also find a button to report a bug specific of this add-on.

Tip: Saving Add-on Preferences

If you want an Add-on to be enabled every time you start Blender, you will need to *Save User Settings*.

Add-on Preferences

Individual Activation

Addons that activate or change multiple hotkeys now have a special system of activation. For example, with the "UI: Pie Menu Offial" add-on for each menu there's a selection box to activate the menu & it's hotkey.

With Pie menus, First you activate the addon. This activates the "Addons Preferences Submodule Activation". You then need to expand the Addons Preferences, then you will see the list of Pie Menu types you can choose from. From here you can individually activate the menus you like to use. If the menu conflicts with another favorite, there's no need to activate it. You can activate any combination & save as user settings so your activation are available next time you start Blender.

Header

Install from File For add-ons that you found on the web or your own to show on the list, you have to install them first by clicking *Install from File...* and providing a `.zip` or `.py` file.

Now the add-on will be installed, not automatically enabled. The search field will be set to the add-on's name (to avoid having to look for it). Enable the add-on by turning on the check-box.

Refresh Scans the *Add-on Directory* for new add-ons.

Online Resources This menu contains a list of helpful links for both users and people who are interested in writing their own add-on.

Scripts Catalog Provides an index of Add-ons that are included with Blender as well as listing a number of external Add-ons.

How to share your add-on Information on how to get your add-on into Blender.

Add-ons development guidelines Guidelines on writing new add-on that you might want to get into Blender.

API Concepts A quick introduction to Blender's API.

Add-on Tutorial A quick tutorial on the essentials of writing an add-on.

Tip: User Defined Add-on Path

You can also create a personal directory containing new add-ons and configure your files path in the *File* tab of the *User Preferences*. To create a personal script directory:

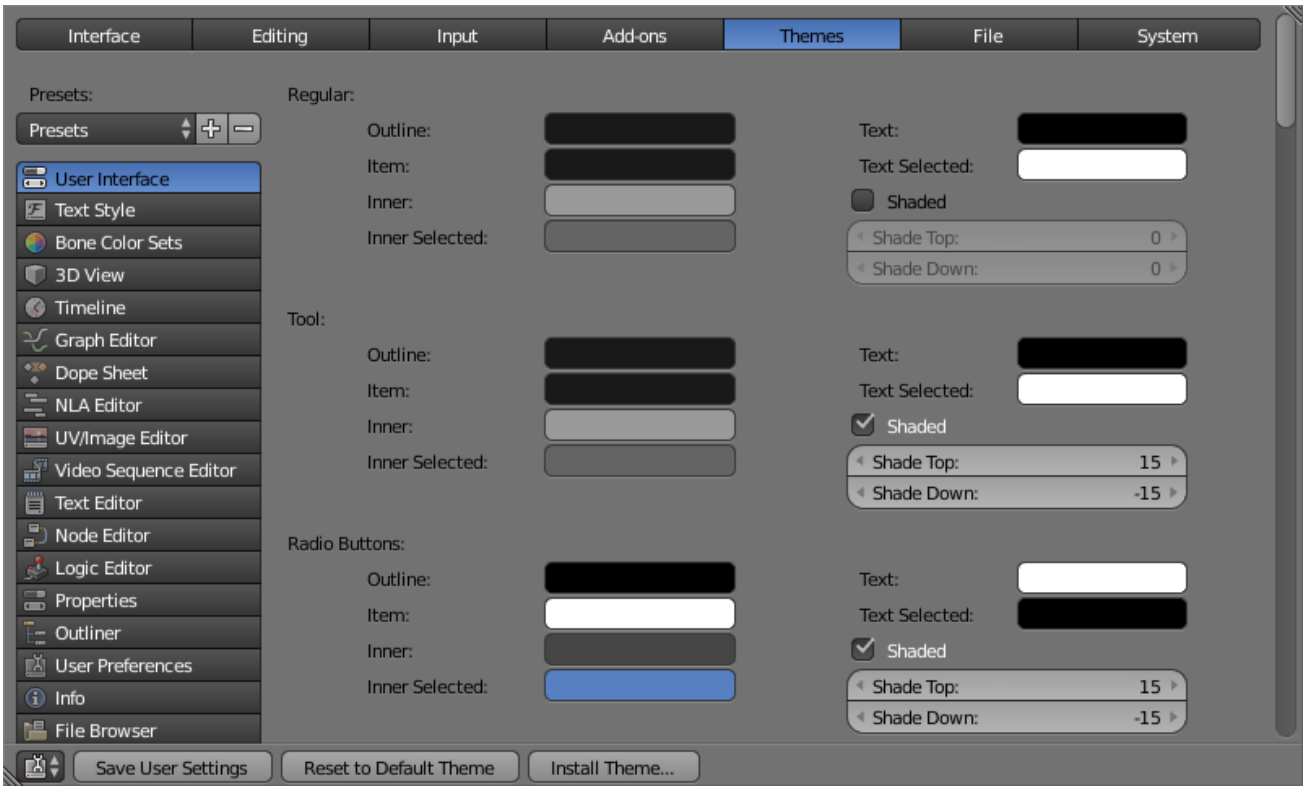
1. Create an empty directory in a location of your choice (i.e. `my_scripts`).
2. Add a subdirectory under `my_scripts` called `addons` (it *must* have this name for Blender to recognize it).
3. Open the *File* tab of the *User Preferences*.
4. Set the *Scripts* in the User Preferences to point to your script directory (i.e. `my_scripts`).
5. Save the User Preferences and restart Blender for it to recognize the new add-ons location.

Now when you install add-ons you can select the *Target Path* option to *User Pref* (from the *File* tab).

Blender will copy newly installed add-ons under the directory selected in your User Preferences.

Themes

The *Themes* tab allows you to customize interface appearance and colors.



The colors for each editor can be set separately by simply selecting the editor you wish to change in the multi-choice list at the left, and adjusting colors as required. Notice that changes appear in real-time on your screen. In addition, details such as the dot size in the *3D View* or the *Graph Editor* can also be changed.

Themes use Blender's preset system to save a theme. This will save the theme to an XML file in the `./scripts/presets/interface_theme/` subdirectory of one of the *configuration directories*.

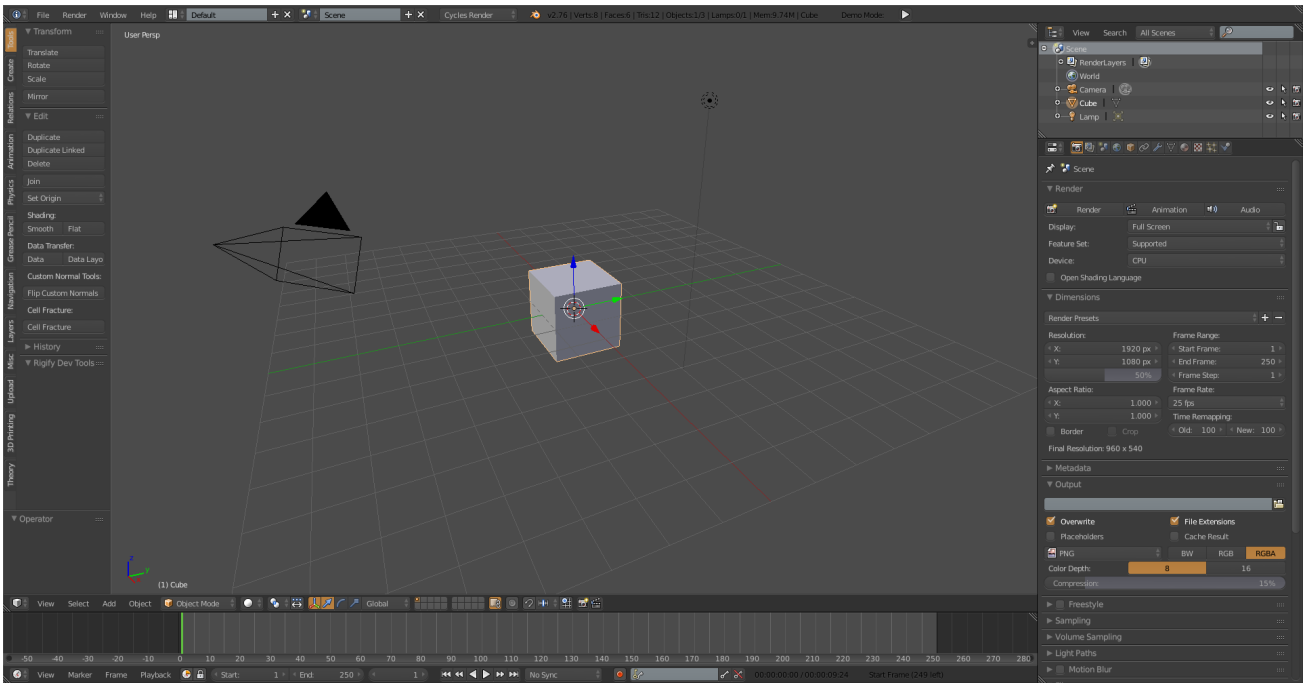


Fig. 2.346: Blender comes bundled with a small selection of themes.
This is an example of the theme *Elsyiun*.

File

The *File* tab in *User Preferences* allows you to configure auto-save preferences and set default file paths for blend-files, rendered images, and more.

File Paths

Locations for various external files can be set for the following options:

Fonts Default location when searching for font files.

Textures Default location when searching for image textures.

Render Output Where rendered images/videos are saved.

Scripts An additional location to search for Python scripts. See *Scripts Path* below.

Sounds Default location when searching for sound files.

Temp The location where temporary files are stored.

Render Cache The location where cached render images are stored.

I18n Branches The path to the `/branches` directory of your local svn-translation copy, to allow translating from the UI.

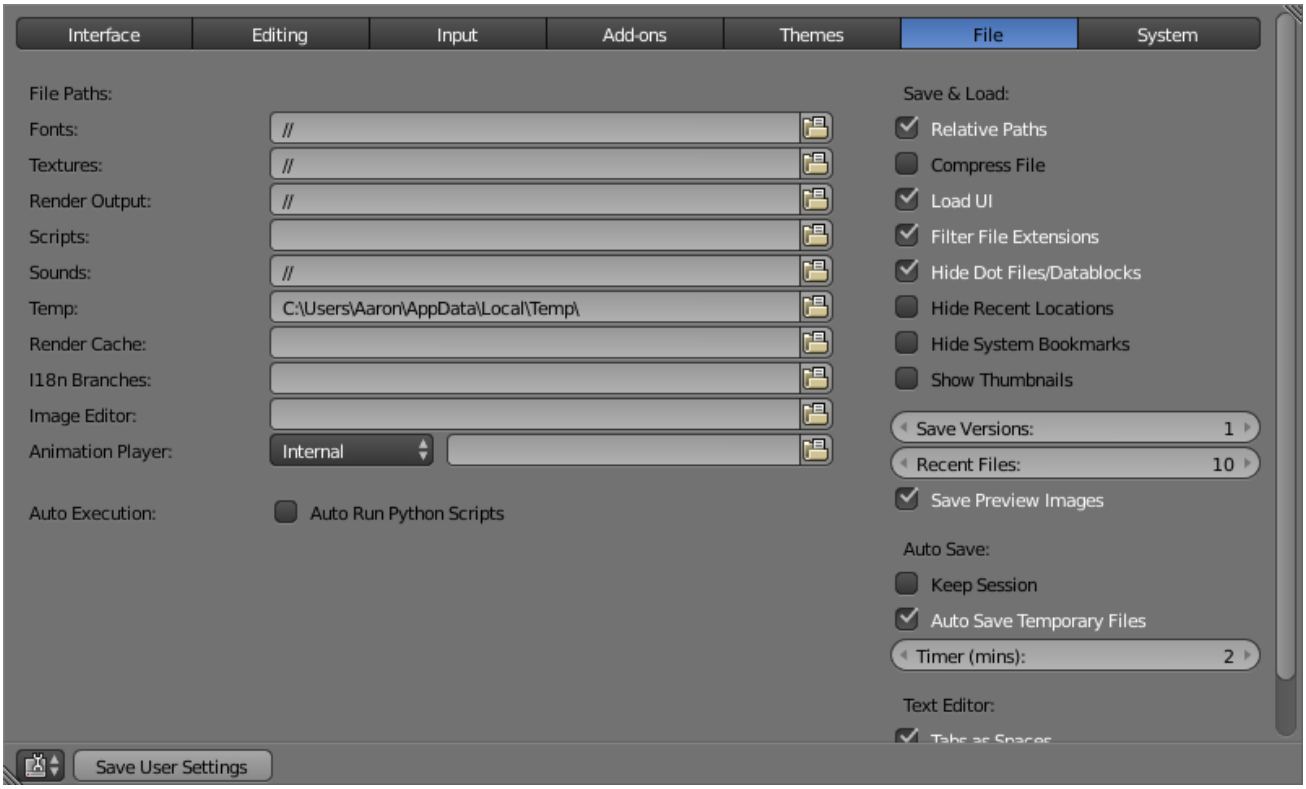


Image Editor The path to an external program to use for image editing.

Animation Player The path to an external program to use for playback of rendered animations.

Note: If these folders do not exist, they will *not* be created automatically.

Scripts Path

By default Blender looks in several directories (OS dependant) for scripts. By setting a user script path in the preferences an additional directory is looked in. This can be used to store certain scripts/templates/presets independently of the currently used Blender Version.

Inside the specified folder, specific subfolders have to be created to tell Blender what to look for where. This folder structure has to mirror the structure of the scripts folder found in the installation directory of Blender:

- scripts
- add-ons
- modules
- presets
- camera
- cloth
- interface_theme
- operator
- render
- ...

- startup
- templates Not all of the folders have to be present.

> **Warning:** Be sure that you have the right privileges for running the executable accessing the path defined. On MS-Windows for instance, if the option "Run this program as an administrator" is enabled for the executable, it will lead to a failure to open the editor due to a limitation within the OS User Account Control. Running a program with elevated privileges is potentially dangerous!

Auto Execution

Python scripts (including driver expressions) are not executed by default for security reasons.

Auto Run Python Scripts You may choose to ignore these security issues and allow scripts to be executed automatically.

Excluded Paths Blend files in these folders will *not* automatically run Python scripts. This can be used to define where blend-files from untrusted sources are kept.

See also:

Python Security

Save & Load

Relative Paths By default, external files use a *relative path*.

Compress File Compress blend-file when saving.

This option will compact your files whenever Blender is saving them. Dense meshes, large packed textures or lots of elements in your scene will result in a large blend being created.

This option may slow down Blender when you quit, or under normal operation when Blender is saving your backup files. Using this option traces processor time for file-size.

Load UI Default setting is to load the Window layout (the *Screens*) of the saved file. This can be changed individually when loading a file from the *Open blend-file* panel of the *File Browser*.

Filter File Extensions By activating this, the file region in the File Browser will only show appropriate files (i.e. blend-files when loading a complete Blender setting). The selection of file types may be changed in the file region.

Fig. 2.347: File extension filter.

Hide Dot File/Data-blocks Hide file which start with `.` on file browsers (in Linux and Apple systems, `.` files are hidden).

Hide Recent Locations Hides the *Recent* panel of the *File Browser* which displays recently accessed folders.

Hide System Bookmarks Hide System Bookmarks in the *File Browser*.

Show Thumbnails Displays a thumbnail of images and movies when using the *File Browser*.

Save Versions Number of versions created for the same file (for backup).

This option tells Blender to keep the indicated number of saved versions of your file in your current working directory when you manually save a file. These files will have the extension: `.blend1`, `.blend2`, etc., with the number increasing to the number of versions

you specify. Older files will be named with a higher number. e.g. With the default setting of 2, you will have three versions of your file: `*.blend` (your last save), `*.blend1` (your second last save) and `*.blend2` (your third last save).

Recent Files Number of files displayed in *File* → *Open Recent*.

Save Preview Images Previews of images and materials in the *File Browser* are created on demand. To save these previews into your blend-file, enable this option (at the cost of increasing the size of your blend-file).

Auto Save

Keep Session Always saves the blend-file after quiting Blender and reloads it after re-starting Blender.

Auto Save Temporary Files Enable Auto Save (create a temporary file).

Checking this box tells Blender to *automatically* save a backup copy of your work-in-progress to the Temp directory (refer to the *File* tab in the *User Preferences* for its location).

The Auto Saved files are named using a random number and have a blend extension.

Timer Time to wait between automatic saves.

This specifies the number of minutes between each Auto Save. The default value of the Blender installation is 5 (5 minutes). The minimum is 1, and the Maximum is 60 (Save at every one hour).

Read more about Auto Save options.

Text Editor

Tabs as Spaces When hitting `Tab` the tabs get written as keyboard spaces.

Author Name that will be used in exported files when the format supports such feature.

System

The *System* tab allows you to set resolution, scripting console preferences, sound, graphics cards, and internationalization.

General

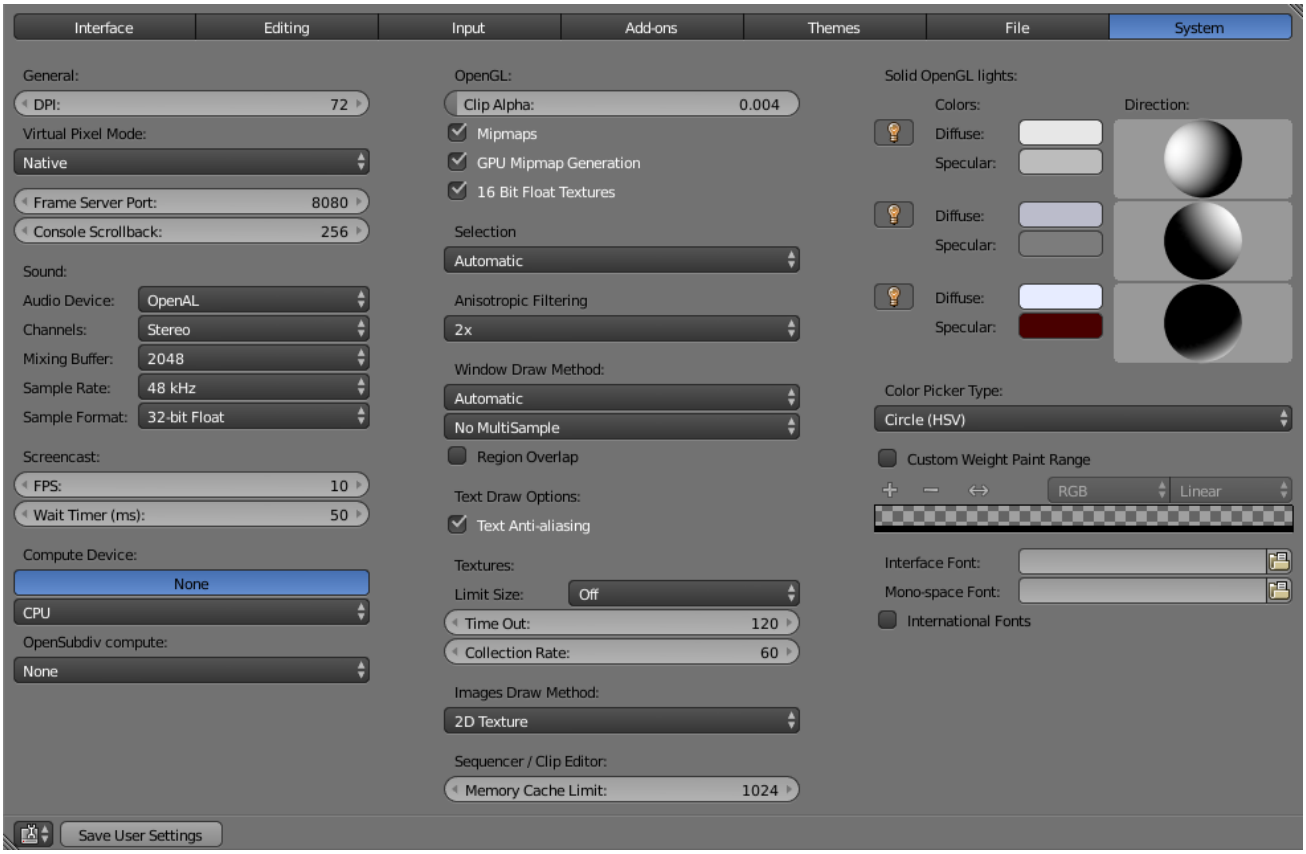
DPI Value of the screen resolution which controls the size of Blender's interface fonts and internal icons shown. Useful for taking screen shots for book printing and use of high resolution monitors. During typical usage, you may prefer to use zoom which is an available in many parts of Blender interface.

Virtual Pixel Mode Allows you to select global scaling. While the DPI only scales the interface, this will scale line width, vertex-size. This is intended for hi-dpi monitors, and is auto-detected on macOS.

Native The normal pixel size.

Double Double of the native pixel size.

Frame Server Port TCP/IP port used in conjunction with the IP Address of the machine for frameserver rendering. Used when working with distributed rendering. Avoid changing this port value unless it is conflicting with already existing service ports used by your Operating System and/or softwares. Always consult your operating system documentation and services or consult your system administrator before changing this value.



Console Scrollback The number of lines, buffered in memory of the console window. Useful for debugging purposes and command line rendering.

Sound

Audio Device Set the audio output device or no audio support:

None No Audio support (no audio output, audio strips can be loaded normally)

SDL Uses Simple Direct Media Layer API from libsdl.org to render sounds directly to the sound device output. Very useful for sequencer strips editing.

OpenAL Provides buffered sound rendering with 3D/spatial support. Used for 3D source support by *Speaker Objects* and the *Game Engine*.

Sound options

Specific to *SDL* or *OpenAL* enabled.

Channels Set the audio channel count. Available options are: *Stereo*, *4 Channels* , *5.1 Surround* , *7.1 Surround*

Mixing Buffer Set the number of samples used by the audio mixing buffer. Available options are: *512* , *1024* , *2048*, *4096* , *8192*, *16384*, and *32768*

Sample Rate Set the audio sample rate. Available options are: *44.1 Khz*, *48 Khs*, *96 Khz* and *192Khz*

Sample Format Set the audio sample format. Available options are: *32 bit float*, *8 bit Unsigned*, *16 Bits Signed*, *24 Bits Signed*, *32 Bits Signed*, *32 Bits Float*, and *64 Bits Float*.

Screencast

These settings are used to control the frame-rate for recording a *Screencast*.

FPS Frame-rate for screencast playback.

Wait Timer Time in milliseconds between each frame recorded for screencast.

Compute Device

The Options here will set the compute device used by the Cycles render engine.

None When set to *None* or the only option is *None*: your CPU will be used as a computing device for Cycles Render Engine

CUDA If the system has a compatible Nvidia CUDA enabled graphics card you will be able to use it to render with the *Cycles* render engine.

OpenCL If the system has a compatible OpenCL device, it will show up has an option for rendering cycles.

Note: that this currently has limited support, see: *Cycles Features* page for more information.

OpenSubdiv Compute

The Options here will set the compute device used by OpenSubdiv for the *Subdivision Surface Modifier*.

None Disables any OpenSubdiv compute devices, makes sure legacy subdivision method is used. Use this option when OpenSubdiv causes any bugs or regressions.

CPU Single threaded CPU implementation. It is mainly useful in cases when GPU compute is possible and threaded CPU option causes artifacts (it is unlikely to happen, but still possible).

OpenMP Multi-threaded CPU implementation. Use it for maximum performance in cases when GPU compute is not available.

GLSL Transform Feedback Uses GPU to perform calculations, has minimal requirements to video card and driver.

GLSL Compute Uses GPU to perform calculations, supposed to be more efficient than *Transform Feedback* but also has higher requirements to video card and driver.

OpenGL

Clip Alpha Clip alpha below this threshold in the 3D View. Note that the default is set to a low value to prevent issues on some GPU's.

Mipmaps Scale textures for 3D View using Mipmap filtering. This increases display quality, but uses more memory.

GPU MipMap Generation Generate MipMaps on the GPU. Offloads the CPU Mimpap generation to the GPU.

16 Bit Float Textures Enables the use of 16 Bit per component Texture Images (Floating point Images).

Selection Selection method to use for selecting.

Automatic Automatically choses the best setting depending on your OS, GPU, and drivers.

OpenGL Select Legacy OpenGL selection method for legacy hardware.

OpenGL Occlusion Queries More optimized OpenGL selection method. Use this method if you are using an *OpenSubdiv Compute* compute device.

Anisotropic Filtering Sets the level of anisotropic filtering. This improves the quality of how textures are drawn at the cost of performance. Available Options are: *Off* (No Filtering), *2x*, *4x*, *8x*, and *16x*.

Window Draw Method

Window Draw Method Specifies the Window Draw Method used to display Blender Window(s).

Automatic Automatically set based on graphics card and driver.

Triple Buffer Use a third buffer for minimal redraws at the cost of more memory. If you have a capable GPU, this is the best and faster method of redraw.

Overlap Redraw all overlapping regions. Minimal memory usage, but more redraws. Recommended for some graphics cards and drivers combinations.

Overlap Flip Redraw all overlapping regions. Minimal memory usage, but more redraws (for graphics drivers that do flipping). Recommended for some graphic cards and drivers combinations.

Full Do a full redraw each time. Only use for reference, or when all else fails. Useful for certain cards with bad to no OpenGL acceleration at all.

Multi-Sampling This enables *FSAA* for smoother drawing, at the expense of some performance.

Note: This is known to cause selection issues on some configurations, see: *Invalid Selection*.

Region Overlap This checkbox will enable Blender to draw regions overlapping the 3D View. It means that the *Tool Shelf* and *Properties regions*, will be drawn overlapping the 3D View editor.

If you have a capable graphics card and drivers with *Triple Buffer* support, clicking the checkbox will enable the overlapping regions to be drawn using the *Triple Buffer* method, which will also enable them to be drawn using Alpha, showing the 3D View contents trough the regions.

Text Draw Options Enable interface text anti-aliasing. When disabled, texts are drawn using text straight render (Filling only absolute Pixels).

Textures

Limit Size Limit the maximum resolution for pictures used in textured display to save memory. The limit options are specified in a square of pixels, (e.g.: the option 256 means a texture of 256×256 pixels) This is useful for game engineers, whereas the texture limit matches paging blocks of the textures in the target graphic card memory. Available Options are: *Off* (No limit), *128*, *256*, *512*, *1024*, *2048*, *4096*, and *8192*.

Time Out Time since last access of a GL texture in seconds, after which it is freed. Set to 0 to keep textures allocated. Minimum: *0*, Maximum: *3600*.

Collection Rate Number of seconds between each run of the GL texture garbage collector. Minimum: *0*, Maximum: *3600*.

Image Draw Method Method to draw images as the following options are supported:

2D Texture Uses CPU for display transform and draws images as a 2D texture.

GLSL Fastest method using GLSL for display transform and draws images as a 2D texture.

Draw Pixels Uses CPU for display transform and draws images as a 2D texture.

Sequencer/Clip Editor

Memory Cache Limit Upper limit of the sequencer's memory cache (megabytes). For optimum clip editor and sequencer performance, high values are recommended.

Solid OpenGL lights

Solid OpenGL Lights are used to light the 3D View, mostly during *Solid view*. Lighting is constant and position "world" based. There are three virtual light sources, also called OpenGL auxiliary lamps, used to illuminate 3D View scenes, which will not display in renders.

The Lamp Icons allows the user to enable or disable OpenGL Lamps. At least one of the three auxiliary OpenGL Lamps must remain enabled for the 3D View. The lamps are equal, their difference is their positioning and colors. You can control the direction of the lamps, as well as their diffuse and specular colors. Available Options are:

Use Toggles the specific lamp.

Diffuse This is the constant color of the lamp.

Specular This is the highlight color of the lamp.

Direction Clicking with LMB in the sphere and dragging the mouse cursor let us the user change the direction of the lamp by rotating the sphere. The direction of the lamp will be the same as shown at the sphere surface.

Color Picker Type

Choose which type of *color space* you prefer. It will show when clicking LMB on any color button.

See the different color picker types at the *Extended Controls* page.

Custom Weight Paint Range

Mesh skin weighting is used to control how much a bone deforms the mesh of a character. To visualize and paint these weights, Blender uses a color ramp (from blue to green, and from yellow to red). Enabling the checkbox will enable an alternate map using a ramp starting with an empty range. Now you can create your custom map using the common color ramp options. For detailed information about how to use color ramps, see: to the *Extended Controls* page.

Internationalization

Blender supports a wide range of languages, enabling this check box will enable Blender to support International Fonts. International fonts can be loaded for the User Interface and used instead of Blender default bundled font.

This will also enable options for translating the User Interface through a list of languages and Tips for Blender tools which appear whenever the user hovers a mouse over Blender tools.

Blender supports I18N for internationalization. For more Information on how to load International fonts, see: *Editing Texts* page.

Info Editor

Introduction

The Info Editor is found at the top of the Default Screen and has the following components.

Header

Fig. 2.348: Info Editor header.
Editor Type Selector (red), Menus (blue), Screen Data-block (green), Scene Data-block (orange), Engine Selector (purple), Resource Information (aqua).

Menus

Provides access to the Blender's main menu options.

File

See *File* menu.

Render

Render See *Render Panel*.

OpenGL Render See *OpenGL Render*

Show/Hide Render View `F11` Shows (or hides) the editor where the last render was performed.

Play Rendered Animation `Ctrl-F11` Plays the last rendered animation using the internal *Animation Player* or an external video player, which has to be defined in the File tab of the User Preferences.

Window

Duplicate Window `Ctrl-Alt-W` Duplicates the current window so that a new one is created with the same screen layout and size. Useful for multiple monitors.

Toggle Window Fullscreen `Alt-F11` Toggles full screen on or off.

Screenshot, Screencast See *Screen Capture*.

Toggle System Console Shows or hides the *System Console*.

Help

See *Help Menu*.

Controls

Back to Previous A button shown when an area is maximized to return to tiled areas.

Screen *Data-block menu* used to select and edit *Screens* (window layouts).

Scene *Data-block menu* to select different *Scenes*. Having multiple Scenes allows you to work with separate virtual environments, with completely separate data, or with object and/or mesh data linked between them.

Engine Gives a list of selectable render and game engines.

Render/Baking progress A progressbar and a cancel button are shown while rendering or baking. Hovering over them shows a time estimate.

Capture Stop A button shown while *screen casting* to stop the recording.

Report Message Label for an operator to display results or warnings. It disappears after a short time. By clicking with `LMB` on the icon on the left side, the full report is copied into a new text data-block, which you can be open in the Text Editor.

Blender Icon Clicking on the Blender logo opens the *Splash Screen*.

Blender version This label displays the Blender version.

Resource Information

Scene Displays information about the current loaded scene dependent on the mode and object type. When two numbers are shown, the first one means the selected, and the second one means the total count. This can be the number of vertices, faces, triangles or bones, as well as the selected objects and lamps.

Memory The "Mem" label shows the calculated memory consumption by Blender. This can help to identify, when you are reaching the limits of your hardware.

Active Object The object type of the current selected object.

Report Console

When the Info Editor's area is scaled up, it reveals the Report console, where a scripting trail is displayed. Whenever an operator has been executed, it leaves a report, creating a log.

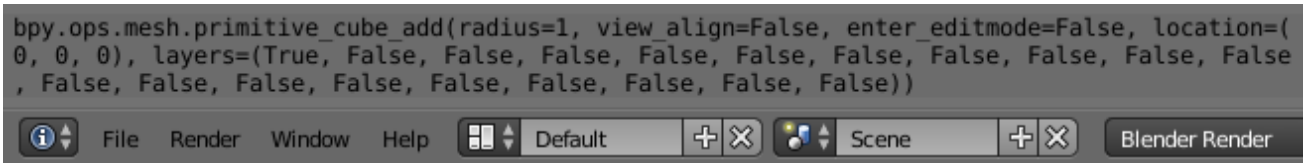


Fig. 2.349: The Report Console after adding a Cube.

File

The options to manage files are:

New **`Ctrl-N`** Clears the current scene and loads startup.blend.

Open **`Ctrl-O`** *Open* a blend-file.

Open Recent **`Shift-Ctrl-O`** Displays a list of *recently* saved blend-files to open.

Recover Last Session This will load the `quit.blend` file Blender automatically saves just before exiting. So this *option* enables you to *recover* your last work session, e.g. if you closed Blender by accident.

Recover Auto Save *This* will open an automatically saved file to *recover* it.

Save `Ctrl-S` *Save* the current blend-file.

Save As `Shift-Ctrl-S` Opens file browser to specify file name and location of *save*.

Save Copy `Shift-Alt-S` *Saves* a copy of the current file.

User Preferences `Ctrl-Alt-U` Opens the *User Preferences Editor* in new window.

Save User Settings `Ctrl-U` Saves the current scene and preferences to *startup.blend*.

Load Factory Settings Restores the default startup-file as *factory settings*.

Link `Ctrl-Alt-O` Links data from an external blend-file (library) to the current scene. The edition of that data is only possible in the external library. *Link* and *Append* is used to load in only selected parts from another file. See *Linked Libraries*.

Append `Shift-F1` Appends data from an external blend file to the current scene. The new data is copied from the external file, and completely unlinked from it.

Import Blender can use information stored in a variety of other format files which are created by other graphics programs. See *Import/Export*.

Export Normally you save your work in a blend-file, but you can export some or all of your work to a format that can be processed by other graphics programs. See *Import/Export*.

External Data External data, like texture images and other resources, can be stored inside the .blend file (packed) or as separate files (unpacked). Blender keeps track of all unpacked resources via a relative or absolute path. See *pack or unpack external Data*

Automatically Pack Into .blend This option activates the file packing. If enabled, every time the blend-file is saved, all external files will be saved (packed) in it.

Pack All Into .blend Pack all used external files into the blend-file.

Unpack Into Files Unpack all files packed into this blend-file to external ones.

Make All Paths Relative Make all paths to external files *relative* to current blend-file.

Make All Paths Absolute Make all paths to external files absolute. Absolute ones have full path from the system's root.

Report Missing Files This option is useful to check if there are links to unpacked files that no longer exist. After selecting this option a warning message will appear in the Info editors header. If no warning is shown, there are no missing external files.

Find Missing Files In case we have broken links in our blend file, this option will help us fix the problem. A File Browser will show up. Select the desired directory (or a file within that directory), and a search will be performed in it, recursively in all contained directories. Every missing file found in the search will be recovered. Those recoveries will be done as absolute paths, so if you want to have relative paths you will need to select *Make All Paths Relative*.

Note: Recovered files might need to be reloaded. You can do that one by one, or you can save the blend file and reload it again, so that all external files are reloaded at once.

Quit `Ctrl-Q` Closes Blender and the file is saved into `quit.blend`.

Screen Capture

Screenshots

Reference

Mode: All modes
Menu: *Window* → *Save Screenshot*
Hotkey: `Ctrl-F3`

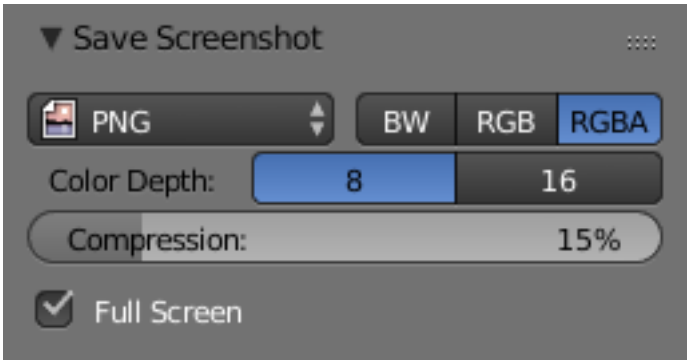


Fig. 2.350: Save Screenshot Option.

`Ctrl-F3` will take a screenshot of Blender and then open the *File Browser*, allowing you to specify the name and location of the screenshot. In the example image at the right, the `PNG` format will be the output of the screenshot taken (settings are the same as the ones available to save render results). When the *File Browser* opens, on the left, there is a tab called *Save Screenshot* where you can find format settings and a checkbox with the option *Full Screen*.

- Check the Option to save the entire Blender window (full width and height of the Blender window you are using when you call the command).
- Uncheck the box to save only your active area (where your mouse is located when you call the command).

Screencasts

Reference

Mode: All modes
Menu: *Window* → *Make Screencast*
Hotkey: `Alt-F3`

This is a quick way to make screen-casts from within Blender.

Note: This is limited to a single window and does *not* support audio.

For recording tutorials you may want to use more comprehensive, 3rd party solutions.

Screencasts will record your actions over time either as a video or sequence of image files. The type and location of the output are determined by the settings in the *Output panel* of the Properties *Render tab*. The default settings will generate a screencast consisting of a series of `PNG` images captured every 50 ms and stored in the `/tmp` folder. If you want to record a video, set the *Output* to one of the *Movie File Formats* supported by your system listed in the *Output panel* format menu. If you are unsure what video codecs your system supports, select `AVI JPEG`.

Note: You can change the frame-rate for a screencast in the *User Preferences*.

When you start Blender Screencasts, the header of the *Info Editor* will change, and it will show you a button for stopping your capture.

Fig. 2.351: Info Header with the Capture Stop Button.

Note: The only way to stop the Screencast

Pressing the Stop button in the header of the Info Editor is the only way to stop the Screencast capture. If you press `Esc`, the shortcut will only work for operations performed in the Blender *User Interface*, (it will stop animations, playbacks and so on...), but will not work to stop *Screencasts*.

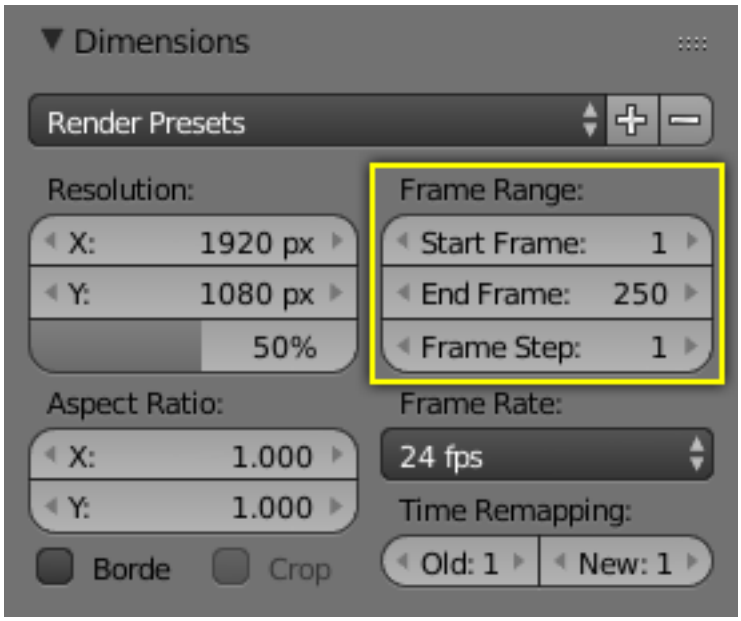


Fig. 2.352: *Render → Dimensions Panel → Frame Range*

The frames are stored using a suffix added to their file name, where the suffix is composed of the numbers present in the fields for *start* and *end frames*, defined in the Frame Range of the Dimensions panel, *Render tab*. (See Fig. *Render → Dimensions Panel → Frame Range* highlighted in yellow)

Note: The configuration of the End frame, present in the Frame Range of the Dimensions Panel, **will not** stop your capture automatically. You will always have to stop the Screencast manually, using the Stop button.

The Videos are generated internally in the same manner as the *Screenshots*, using the width and height of the Window you are working in. If you choose to capture to a Video file, Blender will have to pass those frames to a Video codec.

Warning: Some codecs limit the output width/height or the video quality:

- When you save your *Screencast* in an Image format, the Images will be saved using the entire Blender Window, with full width and height, and the quality of the Image will be defined by its type (e.g. JPG, PNG, and so on) and configuration (e.g. Slider *quality* of the .JPG format).
- When you save your *Screencast* in a Video format, it will be sent to a codec. Depending on the codec limitations, the resulting output Video could be scaled down. Furthermore, some combinations of Window width and height cannot be processed by certain codecs. In these cases, the *Screencast* will try to start, but will immediately stop. In order to solve this, choose another Window format and/or another codec.

Blender Window Dimension

There is a way to match the Blender Window dimensions with the Output Video File, achieving standard dimensions for the output of the Blender Screencast. (i.e. NTSC, HD, Full HD, etc). You can control the width and height of your Blender Window, starting Blender from a Command Line. To learn more about starting Blender from a command line, see the page about *Blender Console Window*.

2.2.6 Other

File Browser

Introduction

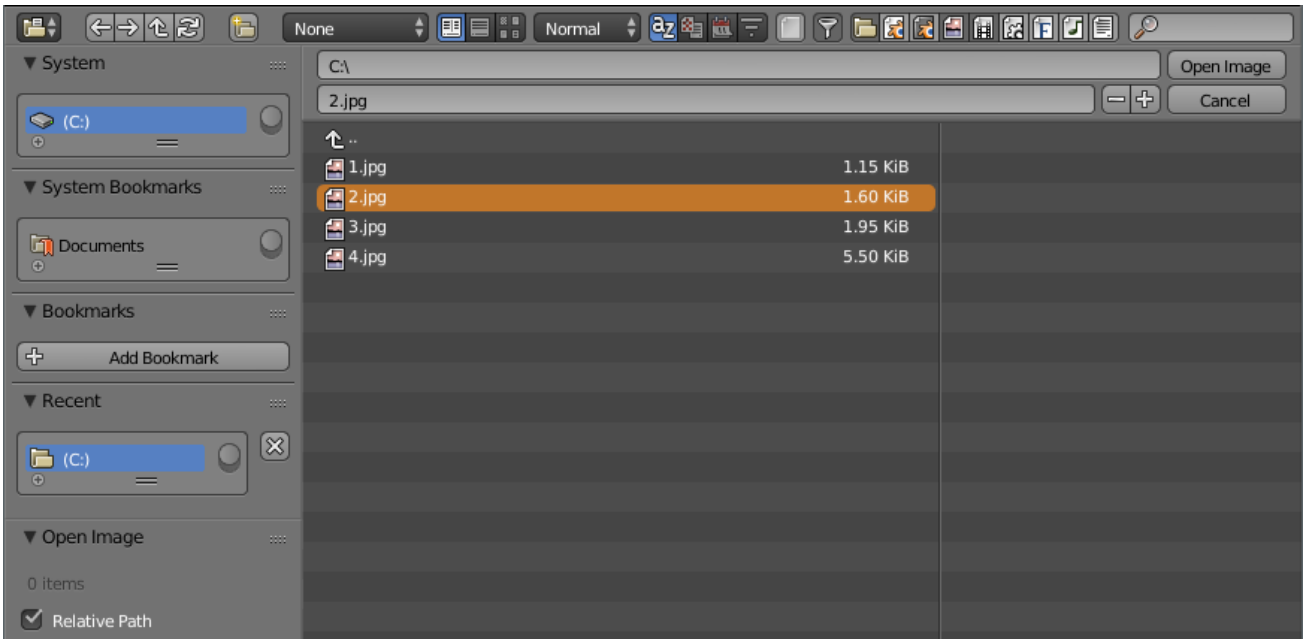


Fig. 2.353: The File Browser.

Usage

The File Browser is used in all the file-related operations. It has multiple use cases, while its often used for save/load.

These include:

- Opening and Saving Blend files.
- Import/Export other file formats.
- Picking new locations for existing file-paths (images, video, fonts...).
- Browsing inside other blend-files, when using *Linked Libraries*.

  You can also keep the File Browser open, as with any other editor type, to browse through the file system. In this case, confirm/cancel buttons will be absent.

  The main purpose of this is to be able to drag media files:

- Images into the *Video Sequence Editor* (to set background or apply as material textures).
- Media files into the *Video Sequence Editor*.

  On the other hand, if the File Browser is opened for a file action (opening, saving, importing, etc.), it will appear maximized and waiting for an operation to complete before returning to the former screen layout.

Opening an Image Sequence

The filename of the images must contain a digit, indicating the frame. The sequence could be opened by the selection of the images and by the confirmation with the *Open image* button or `Return`.

Header

Navigation icon buttons Tools for navigation of files.

Left Arrow **`Backspace`** Move to previous folder.

Right Arrow **`Shift-Backspace`** Move to next folder.

Up Arrow **`P`** Move up to parent directory.

Cycle Arrows **`Numpad .`** Refresh current folder.

Create Directory Prompts you to enter the name of a newly created directory inside the current one `I`.

Recursion The number of directory levels to show at once.

- None (only the current directory)
- One level
- Two Levels
- Three levels

Display type Controls how files are displayed.

- Short list
- Detailed list
- Thumbnails (show *previews*)

Display size The size of thumbnails or the width of columns.

Tiny, small, normal, large

Sorting Sorts files by on of the following methods:

- Alphabetically
- By file type

- By date of last edit
- By file size

Show hidden Shows hidden files (starting with `.`) `H`.

File filtering Filters files by type.

- Folders
- blend-files
- Backup blend-files
- Image files
- Movie files
- Script files
- Font files
- Sound files
- Text files

Search box Filter files by name.

File Region

File Path The text field for the current path. `Tab` will auto-complete an existing path. If you type a non existing directory path, you will be prompted to create that new directory.

File Name Text field to edit the file name and extension. If the background is red, a file with same name already exist in the folder. `Tab` will auto-complete to existing names in the current directory.

Increment Filename +, – Adds/increase or removes/decreases a trailing number to your file name (use to make *versions* of a file).

Confirm The main button to Open Directory/File or Save (As) `Return` or double click with :kbd‘LMB‘ on the entry confirms with that file or data-block.

- `Shift-LMB` – Open the file externally (selected in *File*).
- `Alt-LMB` – Open the directory externally (using the system's file manager).

Cancel Cancels the Open or Save file selection and closes the File browser `Esc` or by using the *Back to Previous* in the Info editor header.

Tool Shelf

The left region displays different ways to find files and several options. Clicking with `LMB` on one of the entries, the File Browser will navigate to that folder.

System

The system panel contains a list of drives that are available to navigate through to find files.

System Bookmarks

Bookmarks that are common for a particular operating system.

Bookmarks

A *List View* of shortcuts to folders, that you want to be able to access often without having to navigate to them in the file browser.

Add + This button adds the current directory to the list.

Recent

This is a list of recently accessed folders. You can control how many folders appear in this list by going to the *File* tab of the *User Preferences*, in the *Recent Files* number button.

Operator Panel

Link/Append from Library See *Linked libraries*.

Open, Save, Save As Blender File See *Opening Files* or *Saving Files*.

Open, Replace, Save As Image See *Supported Graphics Formats*.

For the common option:

Relative Path See *Relative paths*.

Main Region

Navigation

Entering a Directory A single `LMB` click on a directory enters that directory.

Parent Directory `Backspace`, `P` Takes you up one level of directory.

Arrow Keys

With `Alt` pressed. ToDo.

Selection

Select Both `LMB` and `RMB` works.

(De)select All `A` Toggles selecting all files.

Dragging Dragging with `LMB` starts a *border selection*.

Arrow Keys

It is also possible to select/deselect files by "walking" through them using the arrow keys:

- Just using an arrow key, the next file in the chosen direction will be selected and all others deselected.
- Holding down `Shift` while doing this does not deselect anything so it extends to the selection, plus it allows to deselect files by navigating into a block of already selected ones (minimum two files in sequence).
- Holding down `Ctrl-Shift` further selects/deselects all files in-between.

If no file is selected, the arrow key navigation selects the first or last file in the directory, depending on the arrow direction.

If you select a directory and hit enter, you will now go into that directory (and highlighting 'parent' entry will bring you up one level).

File Management

Delete Files `Delete, X` Delete the currently selected files.

Rename `Ctrl-LMB` Can be used on a file or directory to rename it.

Data Previews

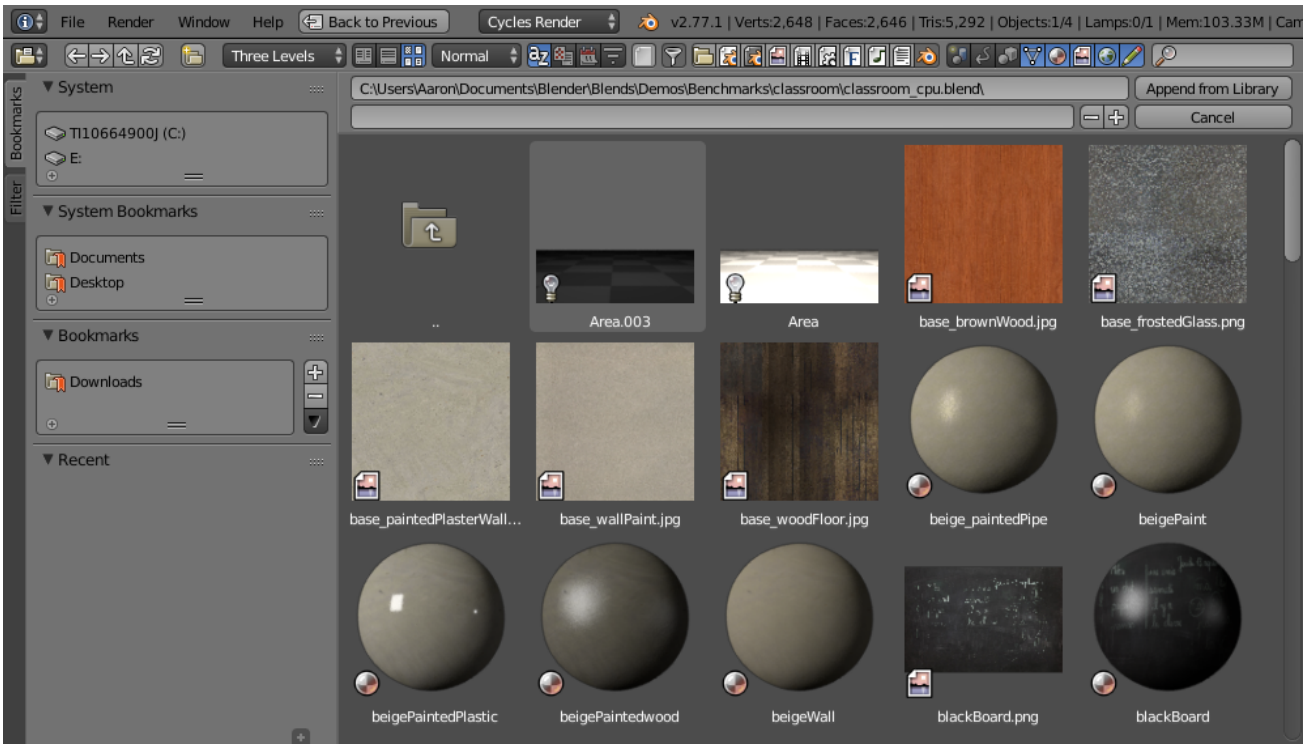


Fig. 2.354: File Browser Previews.

The File Browser supports many types of previews. These include:

- Image and video formats
- Blend-files
- Internal *Data-blocks*
- Fonts

Data-Blocks

Creating and Deleting Previews

Previews can be created and deleted in many ways from *Info Editor* → *File* → *Data Previews*

Refresh Data-Block Previews Ensures that data-block previews are available and up to date.

Batch-Generate Previews Generates previews for selected blend-files.

Scenes Generate previews for scenes.

Groups Generate previews for groups.

Objects Generate previews for objects.

Mat/Tex/.. Generate Previews for materials, textures.

Trusted Blend Files Enables Python evaluation for blend-files.

Save Backups Enables backups in case blend-files become corrupt while generating previews.

Note: If you are generating previews for large file make sure to watch the amount of disk space.

Clear Data-block Previews Clears data-block previews.

Scenes Clears previews for scenes.

Groups Clears previews for groups.

Objects Clears previews for objects.

Materials Clears previews for materials.

Lamps Clears previews for lamps.

Worlds Clears previews for worlds.

Textures Clears previews for textures.

Images Clears previews for images.

Batch-Clear Previews Clears previews for selected blend-files.

Scenes Generate previews for scenes.

Groups Generate previews for groups.

Objects Generate previews for objects.

Mat/Tex/.. Generate Previews for materials, textures.

Trusted Blend Files Enables Python evaluation for blend-files.

Save Backups Enables backups in case blend-files become corrupt while generating previews.

Note: If you are generating previews for large files make sure to watch the amount of disk space.

Python Console

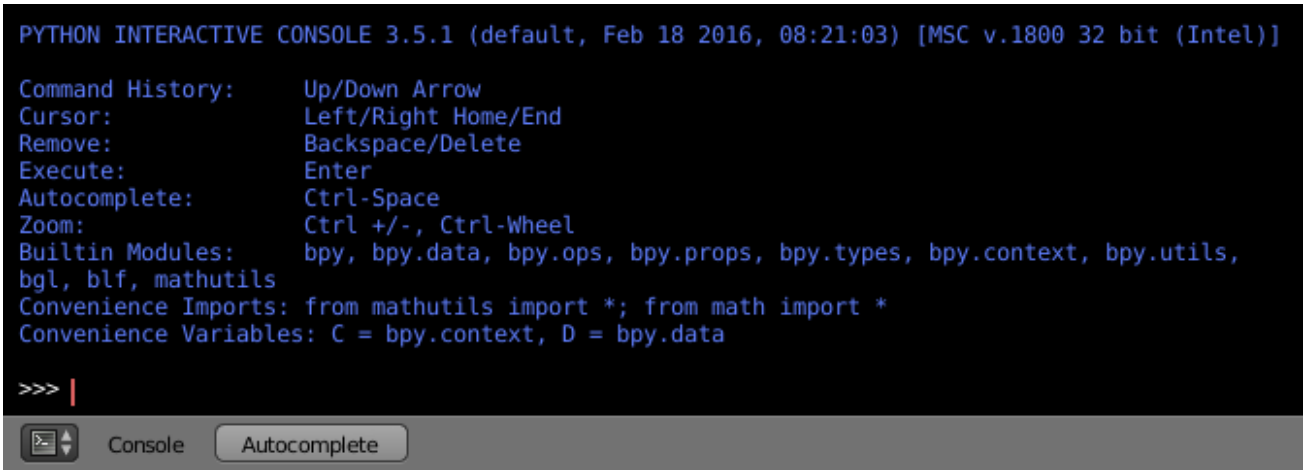
The Python console is a quick way to execute commands, with access to the entire Python API, command history and auto-complete.

Its a good way to explore possibilities, which can then be pasted into larger scripts.

Usage

Accessing Built-in Python Console

By pressing `Shift-F4` in any Blender Editor type (3D View, Timeline etc.,) you can change it to a Console Editor.

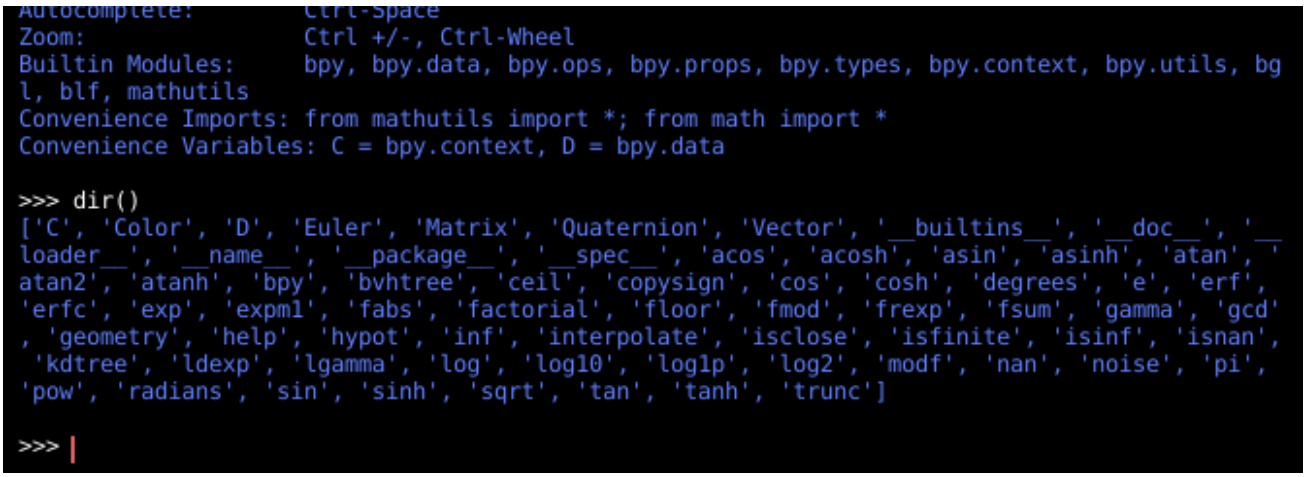


From the screenshot above, you will notice that apart from the usual hot keys that are used to navigate, by pressing `Ctrl-Spacebar` you can enable Auto-complete feature.

The command prompt is typical for Python 3.x, the interpreter is loaded and is ready to accept commands at the prompt >>>

First look at the Console Environment

To check what is loaded into the interpreter environment, type dir() at the prompt and execute it.



Following is a quick overview of the output:

- `C`: Quick access to `bpy.context`.
- `D`: Quick access to `bpy.data`.
- `bpy`: Top level Blender Python API module.

Auto Completion at work

Now, type `bpy.` and then press `Ctrl-Spacebar` and you will see the Console auto-complete feature in action.

You will notice that a list of sub-modules inside of bpy appear. These modules encapsulate all that we can do with Blender Python API and are very powerful tools.

Lets list all the contents of bpy.app module.

Notice the green output above the prompt where you enabled auto-completion. What you see is the result of auto completion listing. In the above listing all are module attribute names, but if you see any name end with `(`, then that is a function.

We will make use of this a lot to help our learning the API faster. Now that you got a hang of this, lets proceed to investigate some of modules in bpy.

Before tinkering with the modules..

If you look at the 3D View in the default Blender scene, you will notice three objects: Cube, Lamp and Camera.

- All objects exist in a context and there can be various modes under which they are operated upon.
- At any instance, only one object is active and there can be more than one selected object.
- All objects are data in the blend-file.
- There are operators/functions that create and modify these objects.

For all the scenarios listed above (not all were listed, mind you..) the bpy module provides functionality to access and modify data.

Examples

bpy.context

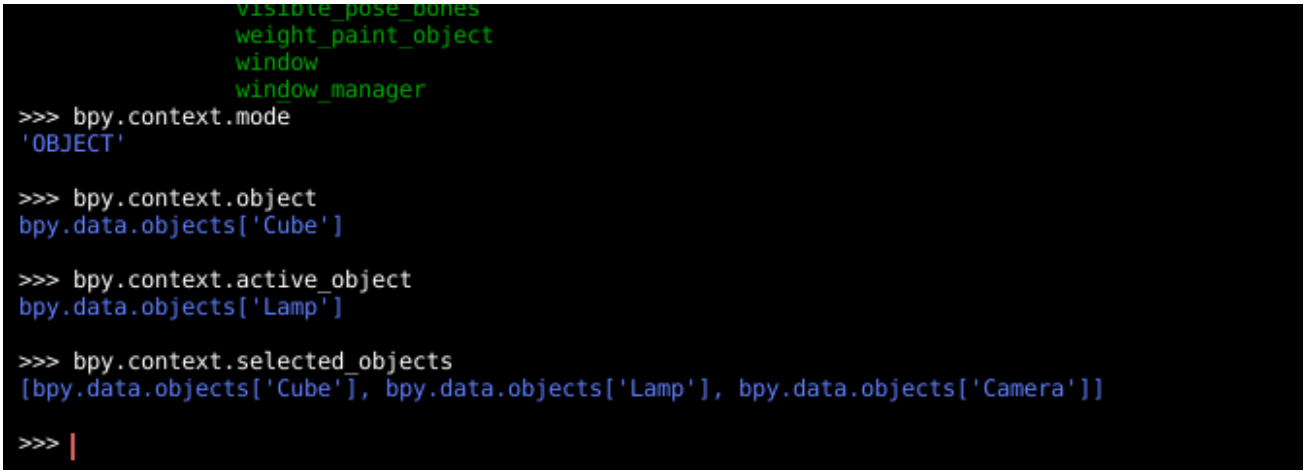
Note: For the commands below to show the proper output, make sure you have selected object(s) in the 3D View.

Try it out!

bpy.context.mode Will print the current 3D View mode (Object, Edit, Sculpt etc.,).

bpy.context.object or bpy.context.active_object Will give access to the active object in the 3D View.

Change X location to a value of 1:




```
bpy.context.object.location.x = 1
```

Move object from previous X location by 0.5 unit:

```
bpy.context.object.location.x += 0.5
```

Changes X, Y, Z location:

```
bpy.context.object.location = (1, 2, 3)
```

Same as above:

```
bpy.context.object.location.xyz = (1, 2, 3)
```

Data type of objects location:

```
type(bpy.context.object.location)
```

Now that is a lot of data that you have access to:

```
dir(bpy.context.object.location)
```

`bpy.context.selected_objects` Will give access to a list of all selected objects.

Type this and then press `Ctrl-Spacebar`:

```
bpy.context.selected_objects
```

To print out the name of first object in the list:

```
bpy.context.selected_objects[0]
```

The complex one... But this prints a list of objects not including the active object:

```
[obj for␣
↪obj in bpy.context.selected_objects if obj != bpy.context.object]
```

bpy.data

`bpy.data` has functions and attributes that give you access to all the data in the blend-file.

You can access following data in the current blend-file: objects, meshes, materials, textures, scenes, screens, sounds, scripts, etc.

That is a lot of data.

Try it out!

```
            use_autopack
            user_map(
            values(
            version
            window_managers
            worlds
>>> bpy.data.objects
<bpy_collection[3], BlendDataObjects>

>>> for object in bpy.data.objects:
...     print(object.name + " is at location " + str(object.location))
...
Camera is at location <Vector (7.4811, -6.5076, 5.3437)>
Cube is at location <Vector (0.0000, 0.0000, 0.0000)>
Lamp is at location <Vector (4.0762, 1.0055, 5.9039)>

>>>
```

Exercise

After `Return` twice it prints the names of all objects belonging to the Blender scene with name "Scene":

```
for obj in bpy.data.scenes['Scene'].objects: print(obj.name)
```

Unlink the active object from the Blender scene named 'Scene':

```
bpy.data.scenes['Scene'].objects.unlink(bpy.context.active_object)
```

```
bpy.data.materials['Material'].shadows

bpy.data.materials['Material'].shadows = False
```

bpy.ops

The tool system is built around the concept of operators. Operators are typically executed from buttons or menus but can be called directly from Python too.

See the bpy.ops API documentation for a list of all operators.

Lets create a set of five Cubes in the 3D View. First, delete the existing Cube object by selecting it and pressing `X`

Try it out!

The following commands are used to specify that the objects are created in layer 1. So first we define an array variable for later reference:

```
mylayers = [False] * 20
mylayers[0] = True
```

We create a reference to the operator that is used for creating a cube mesh primitive:

```
add_cube = bpy.ops.mesh.primitive_cube_add
```

Now in a *for loop*, we create the five objects like this (in the screenshot above, another method is used) : Press `Return` twice after entering the command at the shell prompt:

```
for index in range(5):
    add_cube(location=(index * 3, 0, 0), layers=mylayers)
```

```
>>> mylayers = [False] * 20
>>> mylayers[0] = True
>>>
>>> add_cube = bpy.ops.mesh.primitive_cube_add
>>>
>>> for index in range(5):
...     add_cube(location=(index * 3, 0, 0), layers=mylayers)
...
{'FINISHED'}
{'FINISHED'}
{'FINISHED'}
{'FINISHED'}
{'FINISHED'}

>>>
```

2.3 Data System

2.3.1 Introduction

Each blend-file contains a database. This database contains all scenes, objects, meshes, textures, etc. that are in the file.

A file can contain multiple scenes and each scene can contain multiple objects. Objects can contain multiple materials which can contain many textures. It is also possible to create links between different objects.

Outliner

You can easily inspect the contents of your file by using the *Outliner* editor, which displays all of the data in your blend-file.

The *Outliner* allows you to do simple operations on objects, such as selecting, renaming, deleting, linking and parenting.

Read more about the Outliner

Pack and Unpack Data

Blender has the ability to encapsulate (incorporate) various kinds of data within the blend-file that is normally saved outside of the blend-file. For example, an image texture that is an external image file can be put "inside" the blend-file via *File* → *External Data* → *Pack into blend-file*. When the blend-file is saved, a copy of that image file is put inside the blend-file. The blend-file can then be copied or emailed anywhere, and the image texture moves with it.

You know that an image texture is packed, because you will see a little "Christmas present gift box" displayed in the header.

Unpack Data

When you have received a packed file, you can *File* → *External Data* → *Unpack into Files...*. If files are packed, there is also track of their original path, which can be relative or absolute (this is needed in case of unpacking to original location).

Options

Use files in current directory (create when necessary) Unpacks all files in the same directory // as the blend file, grouping them in proper folders (like ''textures" for instance). However, if the final file exists already, it will use that file, instead of unpacking it.

Write files to current directory (overwrite existing files) Unpacks all files in the same directory as the blend file, grouping them in proper folders (like ''textures" for instance). If the final file exists already, it will overwrite it.

Use files in original location (create when necessary) Unpacks all files in their original location. However, if the final file exists already, it will use that file, instead of unpacking it.

Write files to original location (overwrite existing files) Unpacks all files in their original location. If the final file exists already, it will overwrite it.

Disable AutoPack, keep all packed files Cancels the operation and deactivates the *Automatically Pack Into .blend* option.

2.3.2 Data-Blocks

The base unit for any Blender project is the data-block. Examples of data-blocks include: meshes, objects, materials, textures, node-trees, scenes, texts, brushes and even screens.

For clarity, bones, sequence strips and vertex groups are **not** data-blocks, they belong to armature, scene and mesh types respectively.

Some common characteristics:

- They are the primary contents of the blend-file.
- They can link to each other, for reuse and instancing. (child/parent, object/object-data, with modifiers and constraints too).
- Their names are unique.
- They can be added/removed/edited/duplicated.
- They can be linked between files (only enabled for a limited set of data-blocks).
- They can have their own animation data.
- They can have *Custom Properties*.

When doing more complex projects, managing data-blocks becomes more important, especially when inter-linking blend-files.

Users (Garbage Collection)

It is good to be aware of how Blender, handles data-blocks lifetime, when they are freed and why.

Blender follows the general rule where unused data is eventually removed.

Since it is common to add and remove a lot of data while working, this has the advantage of not having to manually manage every single data-block.

This works by skipping zero user data-blocks when writing blend-files.

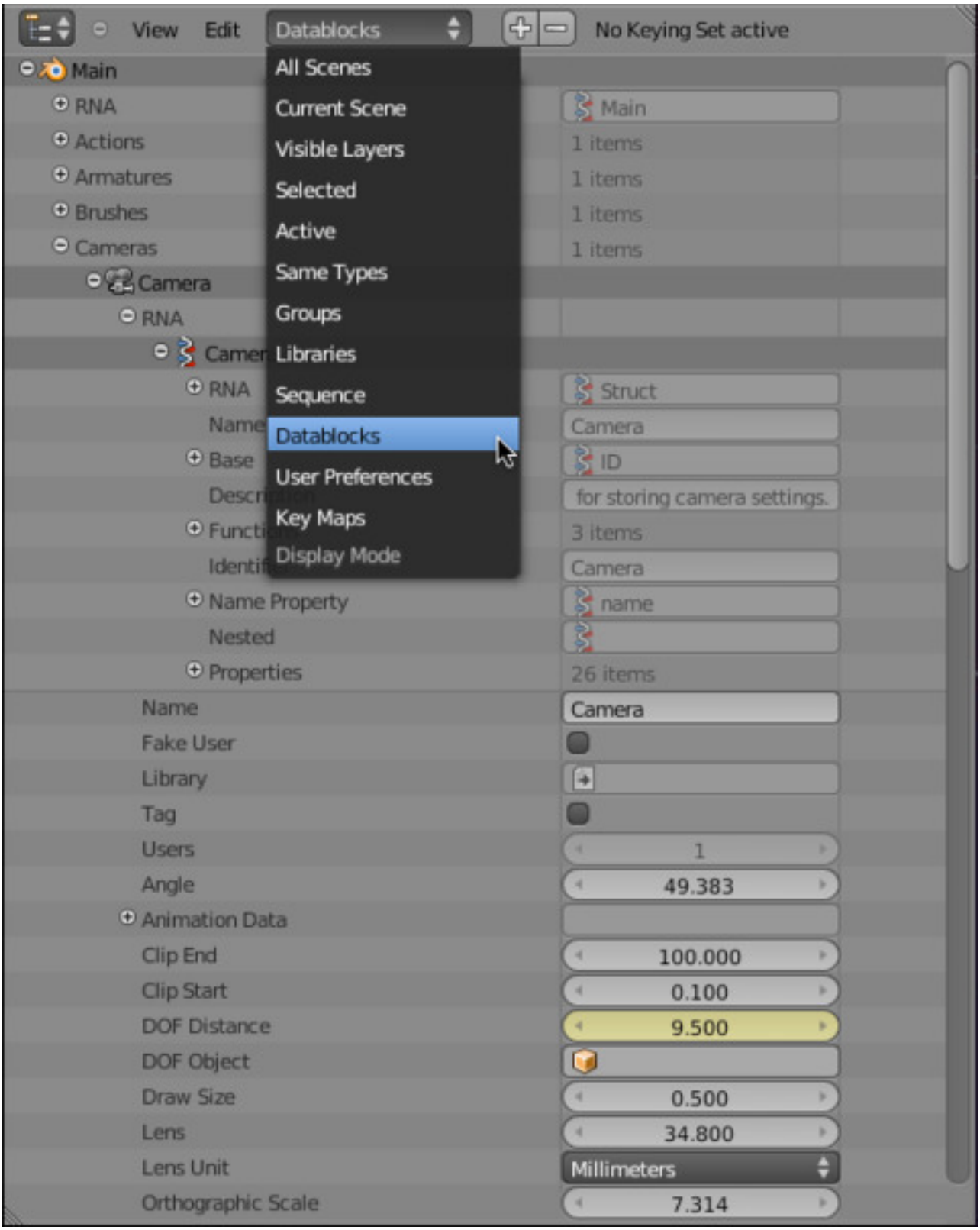


Fig. 2.355: Data-blocks view.

In some cases, you want to save a data-block even when it is unused (typically for re-usable asset libraries). see *Fake User*.

Fake User

Since zero user data-blocks are not saved, there are times when you want to force the data to be kept irrespective of its users.

If you are building a blend-file to serve as a library of things that you intend to link to and from other files, you will need to make sure that they do not accidentally get deleted from the library file.

Do this by giving the data-blocks a *Fake User*, by pressing the *F* button next to the name of the data-block. This prevents the user count from ever becoming zero: therefore, the data-block will not be deleted. (since Blender does not keep track of how many other files link to this one.)

Users (Sharing)

Many data-blocks can be shared among other data-blocks,

Examples where sharing data is common:

- Sharing textures among materials.
- Sharing meshes between objects (instances).
- Sharing animated actions between objects, for example to make all the lights dim together.

You can also share data-blocks between files, see:

- *linked libraries*.

Removing Data-Blocks

As covered in *Users (Garbage Collection)*, data-blocks are typically removed when they are no longer used.

There are some exceptions to this, however.

The following data-blocks can be removed directly: Scene, Text, Group and Screen.

Other data-blocks such as groups and actions can be *Unlinked* from the *Outliner* context menu.

Tip: Some data (images especially) is hard to keep track of, especially since image views are counted as users.

For data-blocks that can be unlinked hold `Shift`, while pressing on the *X* button. This force clears the user-count, so the data-block will be removed on reloading.

Data-Block Types

For reference, here is a table of data-blocks types stored in blend-files.

Link Library Linking, supports being linked into other blend-files.

Pack File Packing, supports file contents being packed into the blend-file.

Type	Link	Pack	Description
Action	✓		Stores animation F-Curves. Used as data-block animation data, and the Non-Linear-Editor.
Armature	✓		Skeleton used to deform meshes. Used as object data & by the Armature Modifier.
Brush	✓		Used by paint tools.
Camera	✓		Used as object data.
Curve	✓		Used by camera, font & surface objects.
Font	✓	✓	References font files. Used by Font object-data.
GreasePencil	✓		2D/3D sketch data. Used as overlay *helper* info, by the 3D View, Image, Sequencer & MovieClip editors.
Group	✓		Reference object's. Used by dupli-groups & often library-linking.
Image	✓	✓	Image files. Used by textures & shader nodes.
Lamp	✓		Used as object-data.
Lattice			Grid based lattice deformation. Used as object data and by the Lattice Modifier.

Continued on next page

Table 2.9 – continued from previous page

Type	Link	Pack	Description
Library		✓	References to external blend-files. Access from the outliner's blend-file view.
LineStyle	✓		Used by the FreeStyle render-engine.
Mask	✓		2D animated mask curves. Used by compositing nodes & sequencer strip.
Material	✓		Set shading and texturing render properties. Used by objects, meshes & curves.
Mesh	✓		Geometry verts/edges/faces. Used as object-data.
MetaBall	✓		An isosurface in 3D space. Used as object-data.
MovieClip	✓		Reference to an image sequence or video file. Used in the motion-tracking editor.
NodeGroup	✓		Collections of re-usable nodes. Used in the Node Editor.
Object	✓		An entity in the scene with location, scale, rotation. Used by scenes & groups.
Particle	✓		Particle settings. Used by particle systems.

Continued on next page

Table 2.9 – continued from previous page

Type	Link	Pack	Description
Palette	✓		Store color presets. Access from the paint tools.
Scene	✓		Primary store of all data displayed and animated. Used as top-level storage for objects & animation.
Screen			Screen layout. Used by each window, which has its own screen.
ShapeKeys			Geometry shape storage, which can be animated. Used by mesh, curve, and lattice objects.
Sounds	✓	✓	References to sound files. Used by speaker objects and the Game Engine.
Speaker	✓		Sound sources for a 3D scene. Used as object-data.
Text	✓		Text data. Used by Python scripts and OSL shaders.
Texture	✓		2D/3D textures. Used by materials, world and brushes.
World	✓		Used by scenes for render environment settings.
WindowManager			TODO.

2.3.3 Custom Properties

Custom properties are a way to store your own meta-data in Blender's data-blocks which can be used for rigging (where

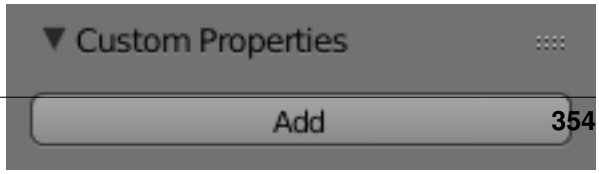


Fig. 2.356: Custom Properties Panel.

bones and objects can have custom properties driving other properties), and Python scripts, where it's common to define new settings not available in Blender.

Only certain data supports custom properties:

- All *data-blocks types*.
- Bones and Pose-Bones.
- Sequence strips.

To add a custom property, find the *Custom Properties* panel, found at the bottom of most *Properties Editor*, and hit *Add*.

Editing Properties

User Interface

Custom properties can be edited using the panel available for data types that support it.

Property Name The name of the custom property

Property Value Todo.

Min The minimum value the custom property can take.

Max The maximum value the custom property can take.

Use Soft Limits Enables limits that the *Property Value* slider can be adjusted to without having to input the value numerically.

Soft Min The minimum value for the soft limit.

Soft Max The maximum value for the soft limit.

Tooltip Allows you to write a custom *Tooltip* for your property.

Fig. 2.357: Custom Properties Edit Region.

Python Access

Custom properties can be accessed in a similar way to dictionaries, with the constraints that keys can only be strings, and values can only be strings, numbers, arrays and nested properties.

See the API documentation for details.

2.3.4 Scenes

Introduction

Scenes are a way to organize your work. Each blend-file can contain multiple scenes, which share other data such as objects and materials.

Scene management and library appending/linking are based on Blender's *Library and Data System*, so it is a good idea to read that manual page first, if you are not familiar with the basics of that system.

You can select and create scenes with the *Scene data-block* menu in the *Info Editor* header.

Fig. 2.358: Scene data-block menu.

Controls

Scenes A list of available scenes.

Add +

New Creates an empty scene with default values.

Copy Settings Creates an empty scene, but also copies the settings from the active scene into the new one.

Link Objects This option creates a new scene with the same settings and contents as the active scene. However, instead of copying the objects, the new scene contains links to the objects in the old scene. Therefore, changes to objects in the new scene will result in the same changes to the original scene, because the objects used are literally the same. The reverse is also true.

Link Object Data Creates new, duplicate copies of all of the objects in the currently selected scene, but each one of those duplicate objects will have links to the object-data (meshes, materials and so on) of the corresponding objects in the original scene.

This means that you can change the position, orientation and size of the objects in the new scene without affecting other scenes, but any modifications to the object-data (meshes, materials, etc.) will also affect other scenes. This is because a single instance of the "object-data" is now being shared by all of the objects in all of the scenes, that link to it. This has the effect of making a new independent copy of the object-data.

Full Copy Using this option, nothing is shared. This option creates a fully independent scene with copies of the active scene's contents. Every object in the original scene is duplicated, and a duplicate, private copy of its object-data is made as well.

Note: To choose between these options, it is useful to understand the difference between *Objects* and *Object Data*. See *Duplication*.

The choices for adding a scene, therefore, determine just how much of this information will be *copied from* the active scene to

the new one, and how much will be *shared* (linked).

Delete X You can delete the current scene by clicking the *X* next to the name in the Info Editor.

Linking to a Scene

You can link any object from one scene to another. Just open the scene where these objects are, from the 3D View header access *Object* → *Make Links...* and choose the scene where you want your objects to appear. The selected objects will be added to that scene, but remain linked to the original objects.

To make them single user (independent and unlinked) in a given scene, go to that scene, select them, then from the 3D View header access *Object* → *Make Single User*. You will be presented with a few options that allow you to free up the data-blocks (Object, Material, Texture...) that you want.

Scene Properties

Scene

Camera Used to select which camera is used as the active camera.

Background Allows you to use a scene as a background, this is typically useful when you want to focus on animating the foreground for example, without background elements getting in the way.

This scene can have its own animation, physics-simulations etc, but you will have to select it from the *Scene* data-block menu, if you want to edit any of its contents.

Tip: This can also be used in combination with *Linking to a Scene*, where one blend-file contains the environment, which can be re-used in many places.

Active Clip Active movie clip for constraints and viewport drawing.

Units

Length Presets Common unit scales to use.

Length

None Uses *Blender Units*.

Metric, Imperial Standard unit of measurement for lengths.

Angle Standard unit for angular measurement.

Degrees, Radians

Tip: When you are using *Degrees*, the radian value is also displayed in the tooltip.

Unit Scale Scale factor to use when converting between *Blender Units* and *Metric*/*Imperial*.

Tip: Usually you will want to use the *Length* presets to change to scale factor, as this does not require looking up values to use for conversion.

Separate Units When *Metric* or *Imperial* display units as multiple multiple values, for example, “2.285m” will become “2m 28.5cm”.

Table 2.10: Imperial Units

Full Name	Short Name(s)	Scale of a Meter
thou	mil	0.0000254
inch	", in	0.0254
foot, feet	', ft	0.3048
yard	yd	0.9144
chain	ch	20.1168
furlong	fur	201.168
mile	mi, m	1609.344

Table 2.11: Metric Units

Full Name	Short Name(s)	Scale of a Meter
micrometer	um	0.000001
millimeter	mm	0.001
centimeter	cm	0.01
decimeter	dm	0.1
meter	m	1.0
dekameter	dam	10.0
hectometer	hm	100.0
kilometer	km	1000.0

Keying Sets

See *Keying Sets*.

Color Management

Options to control how images appear on the screen.

For *Color Management settings* for more information.

Audio

Options to control global audio settings.

Volume Volume for the scene.

Update Animation Cache Updates the audio animation cache. This is useful if you start noticing artifact in the audio.

Distance Model

Distance Model TODO.

Speed Speed of the sound for the Doppler effect calculations.

Doppler Pitch factor for Doppler effect calculation.

Format

Channels TODO.

Mix Rate TODO.

Gravity

Options to control global gravity used for physic effects.

See the *Physics Introduction* for more information.

Rigid Body World

The *Rigid Body World* is a group of Rigid Body objects, which holds settings that apply to all rigid bodies in this simulation.

See *Rigid Body World* for more information.

Simplify

Subdivision Maximum number of *Viewport*/*Render* subdivisions to use for the *Subdivision Modifier*

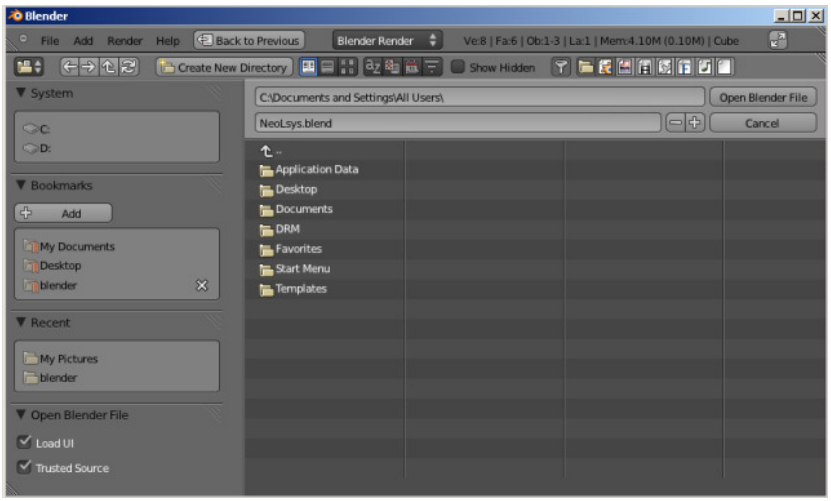
Child Particles Percentage of *Child Particles* to see in the *Viewport*/*Render*.

Use Camera Cull Automatically culls objects based on the camera fulcrum.

Margin Margin for the camera space culling.

2.3.5 Files

Opening Files



Usage

Reference

Menu: *File* → *Open*
Hotkey: `Ctrl-O` or `F1`

The upper text field displays the current directory path, and the lower text field contains the selected filename.

Warning: For Linux and macOS users:

When exiting you are **not** asked to save unsaved changes to the scene you were previously working on. So take care to save your work.

On MS-Windows, there is a *Save & Load* option to warn on exit.

Options

Load UI When *Load UI* is checked, it loads the screen layout saved inside each blend-file, replacing the current layout. Otherwise the file screen layout is ignored.

> **Tip:** If you want to work on the blend-file using your own defaults, start a fresh Blender, then open the file browser and turn off the *Load UI* button, and then open the file.

Trusted Source When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file yourself, or you trust that the person who gave it to you did not include any malicious code with it. See *Python Security* to configure default trust options.

Other File Open Options

From the *File* menu, you can also open files with the following tools:

Open Recent Lists recently used files. Click on one to load it in.

Recover Last Session This will load the `quit.blend` file Blender automatically saved just before exiting. This option enables you to recover your last work session if, for example, you closed Blender by accident.

Recover Auto Save This will allow you to open an automatically saved file to recover it.

See also:

Auto Saves

Saving Files

Reference

Editor: Info
Menu: File

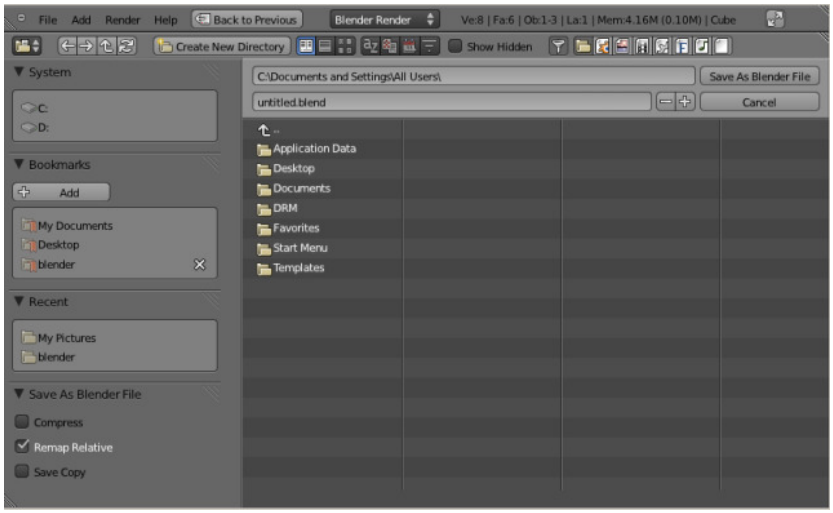
There are a number of slightly different methods you can use to save your blend-file to your hard drive:

Save **Ctrl-S, Ctrl-W** Save an existing blend-file over itself.

Save As **Ctrl-Shift-S, F2** Choose a file to save the blend-file to.

Save Copy **Ctrl-Alt-S** Choose a file to save the blend-file to, but return to editing the original file upon completion. This can be used to save backups of the current working state without modifying the original file.

If the file name does not end with `.blend`, the extension is automatically appended. If a file with the same given name already exists, the text field will turn red as a warning that the file will be overwritten.



Tip: Use the *plus* or *minus* buttons to the right of the file name, or `NumpadPlus`, `NumpadMinus` to increase/decrease a number at the end of the file name (e.g. changing `file_01.blend` to `file_02.blend`).

Options

The save options appear in the operator panel.

Compress File When enabled, the saved file will be smaller, but take longer to save and load.

Remap Relative This option remaps *relative paths* (such as linked libraries and images) when saving a file in a new location.

Save Copy This option saves a copy of the actual working state but does not make the saved file active.

Legacy Mesh Format Save the blend-file, but ignore faces with more than four vertices ("n-gons") so that older versions of Blender (before 2.63) can open it.

See also:

Auto Save

Importing and Exporting Files

Sometimes you may want to utilize files that either came from other 2D or 3D software, or you may want to use the things you have made in Blender and edit them in other software. Luckily, Blender offers a wide range of file formats (e.g. OBJ, FBX, 3DS, PLY, STL... etc) that can be used to import and export.

These formats can be accessed from the menus: *File* → *Import* and *File* → *Export*.

Popular formats are enabled by default, other formats are also supported and distributed with Blender, these can be enabled in the User Preferences through the use of *Add-ons*.

Hint: If you are not interested in technical details, a good rule of thumb for selecting import/export formates for your project is:

Use STL (STereoLithography) if you intend to import/export the files for CAD software. This format is also commonly used for loading into 3D printing software.

Use FBX (Filmbox) if you intend to export objects with rigs and/or animation to be used in other 3D creation suites or game development.

Use ABC (Alembic) if you want to import/export a large amount of scene data.

See also:

A list of these add-ons can also be found on the Add-ons Catalog.

Relative Paths

Many blend-files reference external images or other linked blend-files. A path tells Blender where to look for these files. If the external files are moved, the blend-file that references them will not look right.

When you specify one of these external files, the default option is to make the path relative. Blender stores a partial path evaluated relative to the directory location of the referencing blend-file. This choice helps when you need to reorganize folders or move your files.

With a relative path, you can move the blend-file to a new location provided the externally linked files are moved along with it. For example, you could send someone a folder that contains a blend-file and a sub-folder of external images that it references.

When relative paths are supported, the File Browser provides a *Relative Path* check box, when entering the path into a text field, use a double slash prefix (//) to make it so.

Relative paths are the default but this can be changed in the *File* Tab of the User Preferences Editor.

Note: You cannot enter relative paths into a new *untitled* blend-file. Save it before linking to external files.

Hint: If it is necessary to relocate a blend-file relative to its linked resources, use Blender's File *Save As* function which has an option to *Remap Relative* file links.

Media Formats

Introduction

TODO.

Color Spaces

TODO.

Supported Graphics Formats

Image Formats

This is the list of image file formats supported internally by Blender:

Format	*Channel Depth*	Alpha	*Metadata*	DPI	Extensions
BMP	8bit			✓	`.bmp`
Iris	8bit	✓			`.sgi .rgb .bw`
PNG	8, 16bit	✓	✓	✓	`.png`
JPEG	8bit		✓	✓	`.jpg .jpeg`
JPEG 2000	8, 12, 16bit	✓			`.jp2 .jp2 .j2c`
Targa	8bit	✓			`.tga`
Cineon & DPX	8, 10, 12, 16bit	✓			`.cin .dpx`
OpenEXR	float 16, 32bit	✓	✓	✓	`.exr`
Radiance HDR	float	✓			`.hdr`
TIFF	8, 16bit	✓		✓	`.tif .tiff`

Hint: If you are not interested in technical details, a good rule of thumb for selecting output formats for your project is:

Use OpenEXR if you intend to do compositing or color-grading on these images.

Use PNG if you intend on-screen output or encoding into multiple video formats.

Use JPEG for on-screen output where file size is a concern and quality loss is acceptable.

All these formats support compression which can be important when rendering out animations.

Note: Quicktime

On macOS, Quicktime can be used to access file formats not natively supported (such as GIF).

Channel Depth

Image file formats support a varying number of bits per pixel. This affects the color quality and file-size.

Commonly used depths:

8 bit (256 levels) Most common for on-screen graphics and video

10, 12, 16 bit (1024, 4096, 65536 levels) Used for some formats focusing on photography and digital films (such as DPX and JPEG 2000).

16 bit half float Since full 32bit float is often more than enough precision, half float can save on disk space while providing a high dynamic range.

32 bit float Highest quality color depth.

Internally Blender's image system supports either:

- 8 bit per channel (4 x 8 bits).
- 32 bit float per channel (4 x 32 bits) - *using 4x as much memory.*

Images higher than 8 bits per channel will be converted into a float on loading into Blender.

Note: Floating point is often used for *HDRI*,

When an image has float colors, all imaging functions in Blender default to use that. This includes the Video Sequence Editor, texture mapping, background images, and the Compositor.

Export

Save As Render ToDo.

Copy The Copy checkbox will define if the data-block will reference the newly created file or the reference will be unchanged, maintaining it with the original one.

Format Details

Cineon & DPX

Cineon is Kodak's standard for film scanning, 10 bits/channel and logarithmic. DPX has been derived from Cineon as the ANSI/SMPTE industry standard. DPX supports 16 bits color/channel, linear as well as logarithmic. DPX is currently a widely adopted standard used in the film hardware/software industry.

DPX as well as Cineon only stores and converts the "visible" color range of values between 0.0 and 1.0 (as a result of rendering or composite).

OpenEXR

ILM's OpenEXR has become a software industry standard for HDR image files, especially because of its flexible and expandable structure.

An OpenEXR file can store multiple layers and passes. This means OpenEXR images can be loaded into a compositor keeping render layers, passes intact.

Output Options

Available options for OpenEXR render output are:

Color Depth Saves images in a custom 16 bits per channel floating point format. This reduces the actual "bit depth" to 10 bits, with a 5 bits power value and 1 bit sign.

Float (Half), Float (Full)

Codec

`PIZ` Lossless wavelet compression. Compresses images with grain well.

`ZIP` Standard lossless compression using Zlib.

`RLE` Run-length encoded, lossless, works well when scanlines have same values.

`PXR24` Lossy algorithm from Pixar, converting 32 bits floats to 24 bits floats.

Z Buffer Save the depth information. In Blender, this now is written in floats too, denoting the exact distance from the camera in "Blender unit" values.

Preview On rendering animations (or single frames via command line), Blender saves the same image also as a JPEG, for quick preview or download.

Radiance HDR

Radiance is a suite of tools for lighting simulation. Since Radiance had the first (and for a long time the only) HDR image format, this format is supported by many other software packages.

Radiance (.hdr) files store colors still in 8 bits per component, but with an additional (shared) 8 bits exponent value, making it 32 bits per pixel.

Supported Video Formats

Video Formats

These formats are primarily used for compressing rendered sequences into a playable movie (they can also be used to make plain audio files).

A codec is a little routine that compresses the video so that it will fit on a DVD, or be able to be streamed out over the Internet, over a cable, or just be a reasonable file size. Codecs compress the channels of a video down to save space and enable continuous playback. *Lossy* codecs make smaller files at the expense of image quality. Some codecs, like H.264, are great for larger images. Codecs are used to encode and decode the movie, and so must be present on both the encoding machine (Blender) and the target machine. The results of the encoding are stored in a container file.

There are dozens, if not hundreds, of codecs, including XviD, H.264, DivX, Microsoft, and so on. Each has advantages and disadvantages and compatibility with different players on different operating systems.

Most codecs can only compress the RGB or YUV color space, but some support the Alpha channel as well. Codecs that support RGBA include:

- Quicktime
- PNG TIFF Pixlet is not loss-less, and may be only available on macOS.
- Lagarith Lossless Video Codec

AVI Codec AVI codec compression. Available codecs are operating-system dependent. When an AVI codec is initially chosen, the codec dialog is automatically launched. The codec can be changed directly using the *Set Codec* button which appears (*AVI Codec settings.*).

AVI Jpeg AVI but with Jpeg compression. Lossy, smaller files but not as small as you can get with a Codec compression algorithm. Jpeg compression is also the one used in the DV format used in digital camcorders.

AVI Raw Audio-Video Interlaced (AVI) uncompressed frames.

Frameserver Blender puts out frames upon request as part of a render farm. The port number is specified in the User Preferences.

H.264 Encodes movies with the H.264 codec.

MPEG Encodes movies with the MPEG codec.

Ogg Theora Encodes movies with the Theora codec as Ogg files.

QuickTime Apple's Quicktime `.mov` file. The Quicktime codec dialog is available when this codec is installed on macOS. See *Quicktime* in *Video Containers*.

Xvid Encodes movies with the Xvid codec.

Video Containers

MPEG-1: `.mpg`, `.mpeg` A standard for lossy compression of video and audio. It is designed to compress VHS-quality raw digital video and CD audio down to 1.5 Mbit/s.

MPEG-2: `.dvd`, `.vob`, `.mpg`, `.mpeg` A standard for "the generic coding of moving pictures and associated audio information". It describes a combination of lossy video compression and lossy audio data compression methods which permit storage and transmission of movies using currently available storage media and transmission bandwidth.

MPEG-4(DivX): `.mp4`, `.mpg`, `.mpeg` Absorbs many of the features of MPEG-1 and MPEG-2 and other related standards, and adds new features.

AVI: `.avi` A derivative of the Resource Interchange File Format (RIFF), which divides a file's data into blocks, or "chunks".

Quicktime: `.mov` A multi-tracked format. QuickTime and MP4 container formats can use the same MPEG-4 formats; they are mostly interchangeable in a QuickTime-only environment. MP4, being an international standard, has more support.

DV: `.dv` An intraframe video compression scheme, which uses the discrete cosine transform (DCT) to compress video on a frame-by-frame basis. Audio is stored uncompressed.

H.264: `.avi` ***for now.*** A standard for video compression, and is currently one of the most commonly used formats for the recording, compression, and distribution of high definition video.

Xvid: `.avi` ***for now*** A video codec library following the MPEG-4 standard. It uses ASP features such as b-frames, global and quarter pixel motion compensation, lumi masking, trellis quantization, and H.263, MPEG and custom quantization matrices. Xvid is a primary competitor of the DivX Pro Codec.

Ogg: `.ogg`, `.ogv` A free lossy video compression format. It is developed by the Xiph.Org Foundation and distributed without licensing fees.

Matroska: `.mkv` An open standard free container format, a file format that can hold an unlimited number of video, audio, picture or subtitle tracks in one file.

Flash: `.flv` A container file format used to deliver video over the Internet using Adobe Flash Player.

Wav: `.wav` An uncompressed (or lightly compressed) Microsoft and IBM audio file format.

Mp3: `.mp3` A highly-compressed, patented digital audio encoding format using a form of lossy data compression. It is a common audio format for consumer audio storage, as well as a de facto standard of digital audio compression for the transfer and playback of music on digital audio players.

Video Codecs

None For audio-only encoding.

MPEG-1 See *Video Formats*.

MPEG-2 See *Video Formats*.

MPEG-4(DivX) See *Video Formats*.

HuffYUV Lossless video codec created by Ben Rudiak-Gould which is meant to replace uncompressed YCbCr as a video capture format.

DV See *Video Formats*.

H.264 See *Video Formats*.

Xvid See *Video Formats*.

Theora See Ogg in *Video Formats*.

Flash Video See *Video Formats*.

FFmpeg video codec #1 A.K.A. FFV1, a loss-less intra-frame video codec. It can use either variable length coding or arithmetic coding for entropy coding. The encoder and decoder are part of the free, open-source library libavcodec in FFmpeg.

Audio Containers

MP2 A lossy audio compression format defined by ISO/IEC 11172-3.

MP3 See MP3 in *Video Formats* (above).

AC3 Audio Codec 3, an audio compression technology developed by Dolby Laboratories.

AAC Advanced Audio Codec, a standardized, lossy compression and encoding scheme for digital audio. – AAC generally achieves better sound quality than MP3 at similar bit rates.

Vorbis An open-standard, highly-compressed format comparable to MP3 or AAC. – Vorbis generally achieves better sound quality than MP3 at similar bit rates.

FLAC Free Lossless Audio Codec. Digital audio compressed by FLAC's algorithm can typically be reduced to 50-60% of its original size, and decompressed into an identical copy of the original audio data.

PCM Pulse Code Modulation, a method used to digitally represent sampled analog signals. It is the standard form for digital audio in computers and various Blu-ray, Compact Disc and DVD formats, as well as other uses such as digital telephone systems.

Known Limitations

Video Output Size

Some codecs impose limitations on output size, `H.264`, for example requires both the height and width to be divisible by 2.

2.3.6 Append and Link

These functions help you reuse materials, objects and other *data-blocks* loaded from an external source blend-file. You can build libraries of common content and share them across multiple referencing files.

Link creates a reference to the data in the source file such that changes made there will be reflected in the referencing file the next time it is reloaded.

Whereas *Append* makes a full copy of the data into your blend. You can make further edits to your local copy of the data, but changes in the external source file will not be reflected in the referencing file.

Reference

Mode: All Modes
Menu: *File* → *Append or Link*
Hotkey: `Shift-F1` or `Ctrl-Alt-O`

In the *File Browser* navigate to the external source blend-file and select the data-block you want to reuse.

Options:

Relative Path Available only when linking, see *relative paths*.

Select Makes the object *Active* after it is loaded.

Active Layer The object will be assigned to the visible layers in your scene. Otherwise, it is assigned to the same layers it resides on in the source file.

Instance Groups This option links the Group to an object, adding it to the active scene.

Fake User Sets a *Fake User* for the append items.

Localize All ToDo.

When you select an Object type, it will be placed in your scene at the cursor. Many other data types, cameras, curves, and materials for example, must be linked to an object before they become visible.

Newly added Group types are available in *Add* → *Group Instances* in 3D View, or for NodeTree groups, the same menu in the Node Editor.

Look in the Outliner, with display mode set to *blend-file*, to see all your linked and appended data-blocks. `Ctrl-LMB` on a file name allows you to redirect a link to another file.

Hint: You cannot move a linked object. Its position is defined in its source file.

If you want to modify the object locally you can either:

Use *Dupli-Groups* Instead of linking to *Objects* directly, it is often more useful to link in *Groups*, which can be assigned to empties and moved, while maintaining the link to the original file.

It is also useful to be able to add/remove objects from the group without having to manage linking in multiple objects.

Make Objects Local Use *Object* → *Make Local* → *Selected Objects* to make the position editable.

This means that object data (animation, constraints, modifiers...) will be local to your blend-file. But the object-data will still be linked and remain immutable.

Note: Appending data you already have linked will add objects/groups to the scene, but will keep them linked (and un-editable).

This is done so existing relationships with linked data remain intact.

Proxy Objects

Used with rigged models, proxy objects, allow specified bone layers to be linked back to the source file while the remainder of the object and its skeleton are edited locally.

`Ctrl-Alt-P` makes the active linked object into a local proxy, appending "_proxy" to its name.

Set the *Protected Layers* in the source file using the Skeleton panel of the Armatures tab. See *Armature Layers*. The bones in protected layers will have their position restored from the source file when the referencing file is reloaded.

Known Limitations

For the most part linking data will work as expected, however, there are some corner-cases which are not supported.

Circular Dependencies

In general, dependencies should not go in both directions.

Attempting to link or append data which links back to the current file will likely result in missing links.

Object Rigid-Body Constraints

When linking objects *directly* into a blend-file, the *Rigid Body* settings **will not** be linked in since they are associated with their scene's world.

As an alternative, you could link in the entire scene and set it as a *Background Set*.

2.4 Modeling

2.4.1 Introduction

The creation of a 3D scene needs at least three key components: Models, Materials and Lights. In this part, the first of these is covered, that being modeling. Modeling is simply the art and science of creating a surface that either mimics the shape of a real-world object or expresses your imagination of abstract objects.

Modeling can take many forms in Blender depending on the type of *object* you are trying to model. Some objects are not able to be modeled, these being:

- Speakers
- Cameras

- Lamps

Modes

Depending on the type of object you are trying to model, there are different types of modeling *mode*. Because modes are not specific to modeling they are covered in different parts of the manual.

Edit Mode

Edit mode is the main mode that modeling takes place. Edit mode is used to edit the following types of objects:

- Meshes
- Curves
- Surfaces
- Metaballs
- Text objects
- Lattice

Because each of these are different types of object they have different types of transforms and therefore have different set of tool. Because of this each has its own section described below.

Mesh Modeling Typically begins with a *Mesh Primitive* shape (e.g. circle, cube, cylinder...).

Curve Modeling Uses control points to define the shape of the curve.

Surface Modeling Similar to curve modeling, but instead of being limited to simple linear paths, they allow the creation of three dimensional surfaces, potentially with volume.

Metaball Modeling Begins similarly to mesh modeling (see above), with a base shape like a cube or sphere, but instead of extruding these base shapes, these objects are clumped together to form a larger object. In order to accomplish this, the metaballs have a liquid-like quality, when two or more are brought together, they merge by smoothly rounding out the point of connection, appearing as one unified object.

This can also be a quick way to get started with a rough shape which can be converted to a mesh later.

Text Modeling Text modeling is an easy way to create logos and to simply add text to a scene.

Modifiers Modifiers are automatic operations that affect an object in a non-destructive way. With modifiers, you can perform many effects automatically that would otherwise be tedious to do manually.

2.4.2 Meshes

Introduction

Mesh Modeling typically begins with a *Mesh Primitive* shape (e.g. circle, cube, cylinder...). This mesh primitive is defined by an array of points in 3D space called vertices (singular form is *Vertex*). From there you might begin extruding faces and moving vertices to create a larger, more complex shape.

Modeling Modes

The 3D View has three principal modes that allow for the creation of, editing and manipulation of the mesh models. Each of the three modes have a variety of tools. Some tools may be found in one or more of the modes.

Modes that used for modeling:

- Object Mode
- Edit Mode

- Sculpt Mode

Creation of a mesh primitive typically starts by adding a mesh object in *Object Mode*. Limited types of editing such as size, location, and orientation can be accomplished in *Object Mode*. *Object Mode* also provides the means to Join and Group multiple mesh primitives.

More detailed editing of the mesh model shape is done in *Edit Mode*, and *Sculpt Mode*. The nature of these three modes determines the tools that are available within the various panels of the 3D View. Switching between modes while modeling is common. Some tools may be available in more than one mode while others may be unique to a particular mode.

You can work with geometric objects in two modes.

Object Mode

Object Mode Operations in *Object Mode* affect the whole object. *Object Mode* has the following header in the 3D View:

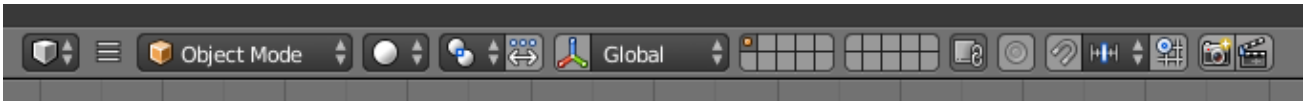


Fig. 2.359: Object Mode Header.

Edit Mode

Operations in *Edit Mode* affect only the geometry of an object, but not global properties such as location or rotation. *Edit Mode* has the following header in the 3D View:

Fig. 2.360: Edit Mode Header.

Tools and modes in the 3D View header are (left to right):

- View, Select, and Mesh menus
- Blender Mode
- Display method for 3D View
- Pivot center
- 3D manipulator widget
- Selection mode
- Depth buffer clipping (hide)
- Proportional editing
- Snap
- OpenGL render

You can switch between the Object and Edit Modes with `Tab`. You can change to any mode by selecting the desired *Mode* in the menu in the 3D View header.

Visualization

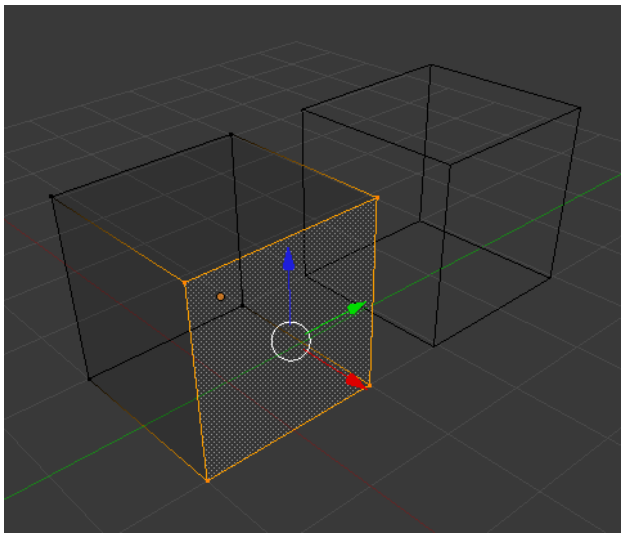


Fig. 2.361: One cube selected.

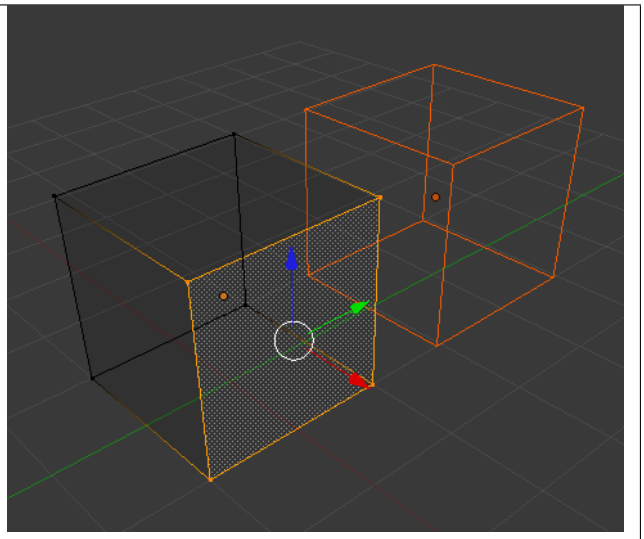


Fig. 2.362: Two cubes selected before entering Edit Mode.

By default, Blender highlights selected geometry in orange in both *Object Mode* and *Edit Mode*.

In *Object Mode* with *Wireframe* shading enabled `Z`, objects are displayed in black when unselected and in orange when selected. If more than one object is selected, all selected objects except the active object, typically the object last selected, are displayed in a darker orange color. Similarly, in *Edit Mode*, unselected geometry is drawn in black while selected faces, edges, or vertices are drawn in orange. The active face is highlighted in white.

In *Edit Mode*, only one mesh can be edited at the time. However, several objects can be joined into a single mesh (`Ctrl-J` in *Object Mode*) and then separated again (`P` in *Edit Mode*). If multiple objects are selected before entering *Edit Mode*, all the selected objects remain highlighted in orange indicating that they are part of the active selection set.

If two vertices joined by an edge are selected in *Vertex selection mode*, the edge between them is highlighted too. Similarly, if enough vertices or edges are selected to define a face, that face is also highlighted.

Tool Shelf

Open/close the *Mesh Tools* panel using `T`. When entering *Edit Mode*, several mesh tools become available.

Most of these tools are also available as shortcuts (displayed in the *Tooltips* for each tool) and/or in the *Specials* menu `W`, the *Edge* menu `Ctrl-E`, and *Face* menu `Ctrl-F`. The properties of each tool are displayed in the operator panel at the bottom of the *Tool Shelf*.

Even more mesh editing tools can be enabled in the *User Preferences* → *Add-ons*.

Properties Region

Open/close the *Properties region* using `N`.

In the *Properties region*, panels directly related to mesh editing are the *Transform* panel, where numeric values can be entered, and the *Mesh Display* panel, where for example normals and numeric values for distances, angles, and areas can be turned on.

Other useful tools are found in the *Properties Editor* under the *Object* and *Object Data* tabs, including display options and *Vertex groups*.

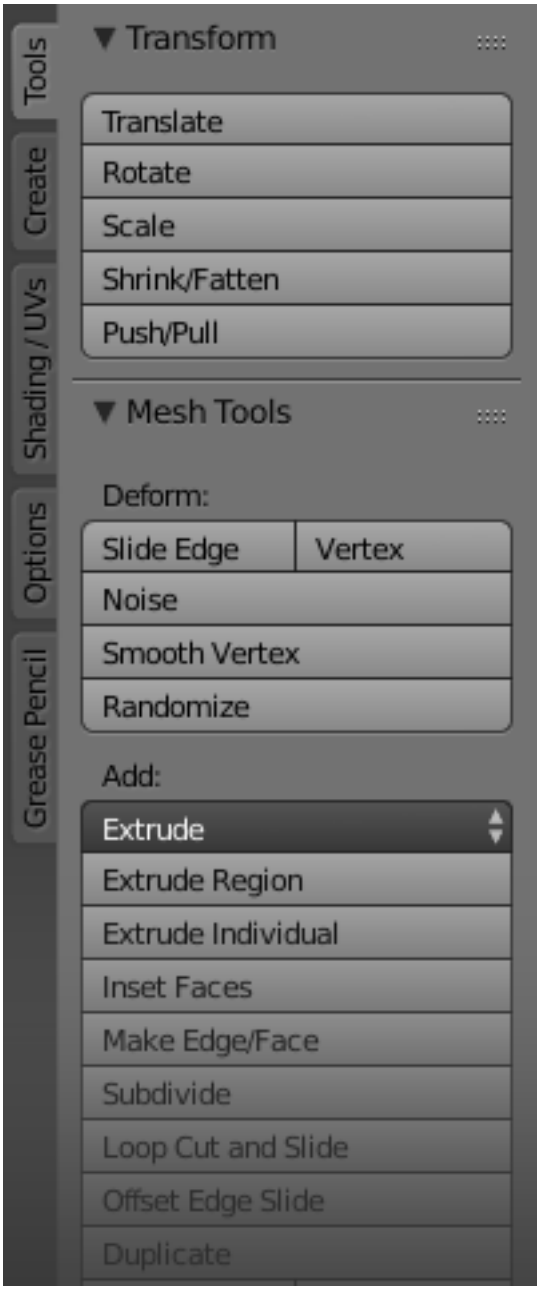


Fig. 2.363: The Tool Shelf panel in edit mode.

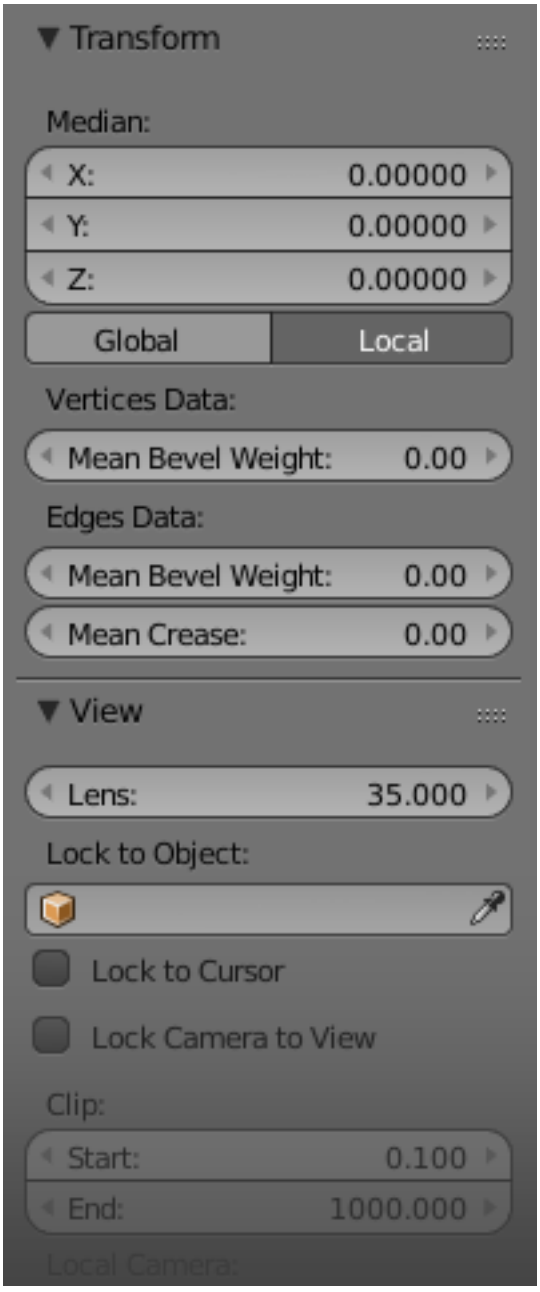


Fig. 2.364: The Properties region in edit mode.

Structure

With meshes, everything is built from three basic structures: *Vertices*, *Edges* and *Faces*.

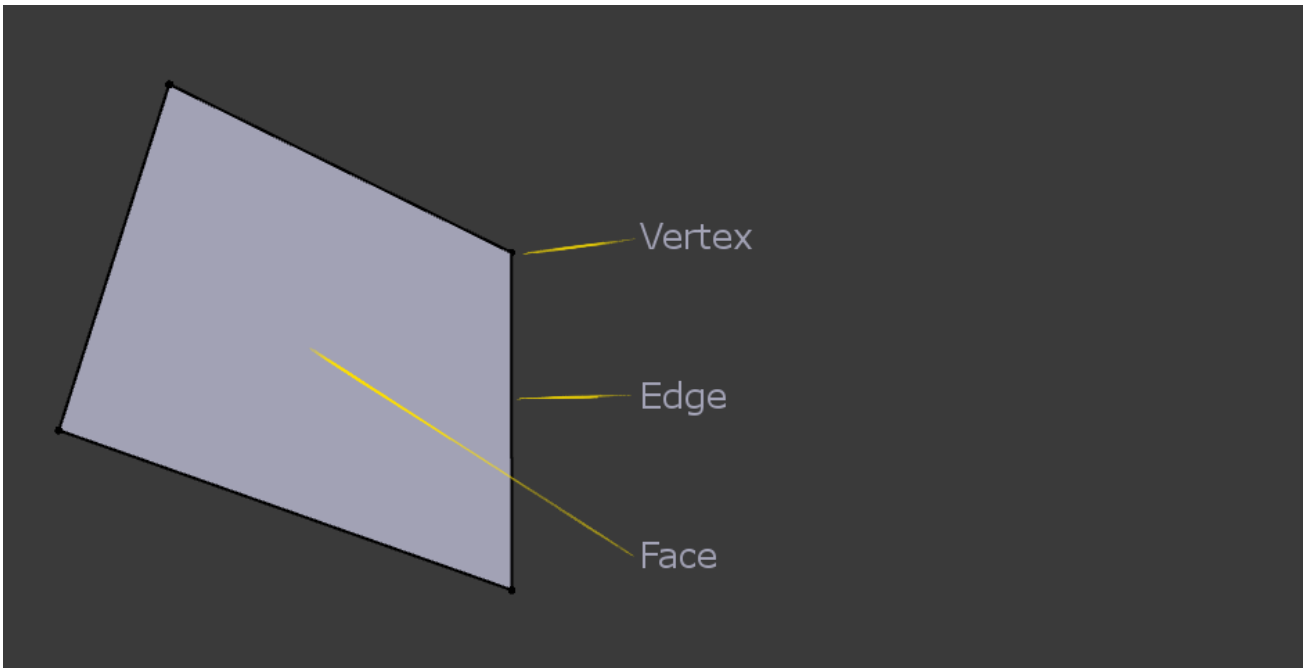


Fig. 2.365: Example of mesh structure.

Vertices

A vertex is primarily a single point or position in 3D space. It is usually invisible in rendering and in *Object Mode*. Do not mistake the center point of an object for a vertex. It looks similar, but it is bigger and you cannot select it. Fig. *Vertex example.* shows the center point labeled as "A"; "B" and "C" are vertices.

A simple way to create a new vertex is to click `Ctrl-LMB` in *Edit Mode*. Of course, as a computer screen is two-dimensional, Blender cannot determine all three vertex coordinates from a single mouse click, so the new vertex is placed at the depth of the 3D cursor. Using the method described above, any vertices selected previously are automatically connected to the new ones by an edge. In the image above, the vertex labeled "C" is a new vertex added to the cube with a new edge added between "B" and "C".

Edges

An edge always connects two vertices by a straight line. The edges are the "wires" you see when you look at a mesh in wireframe view. They are usually invisible on the rendered image. They are used to construct faces. Create an edge by selecting two vertices and pressing `F`.

Faces

Faces are used to build the actual surface of the object. They are what you see when you render the mesh. If this area does not contain a face, it will simply be transparent or non-existent in the rendered image. To create a face, select three or more suitable vertices and press `F`.

A face is defined as the area between either three (triangles), four (quadrangles) or more (ngons) vertices, with an edge on every side. These are often abbreviated to *tris, quads & ngons*.

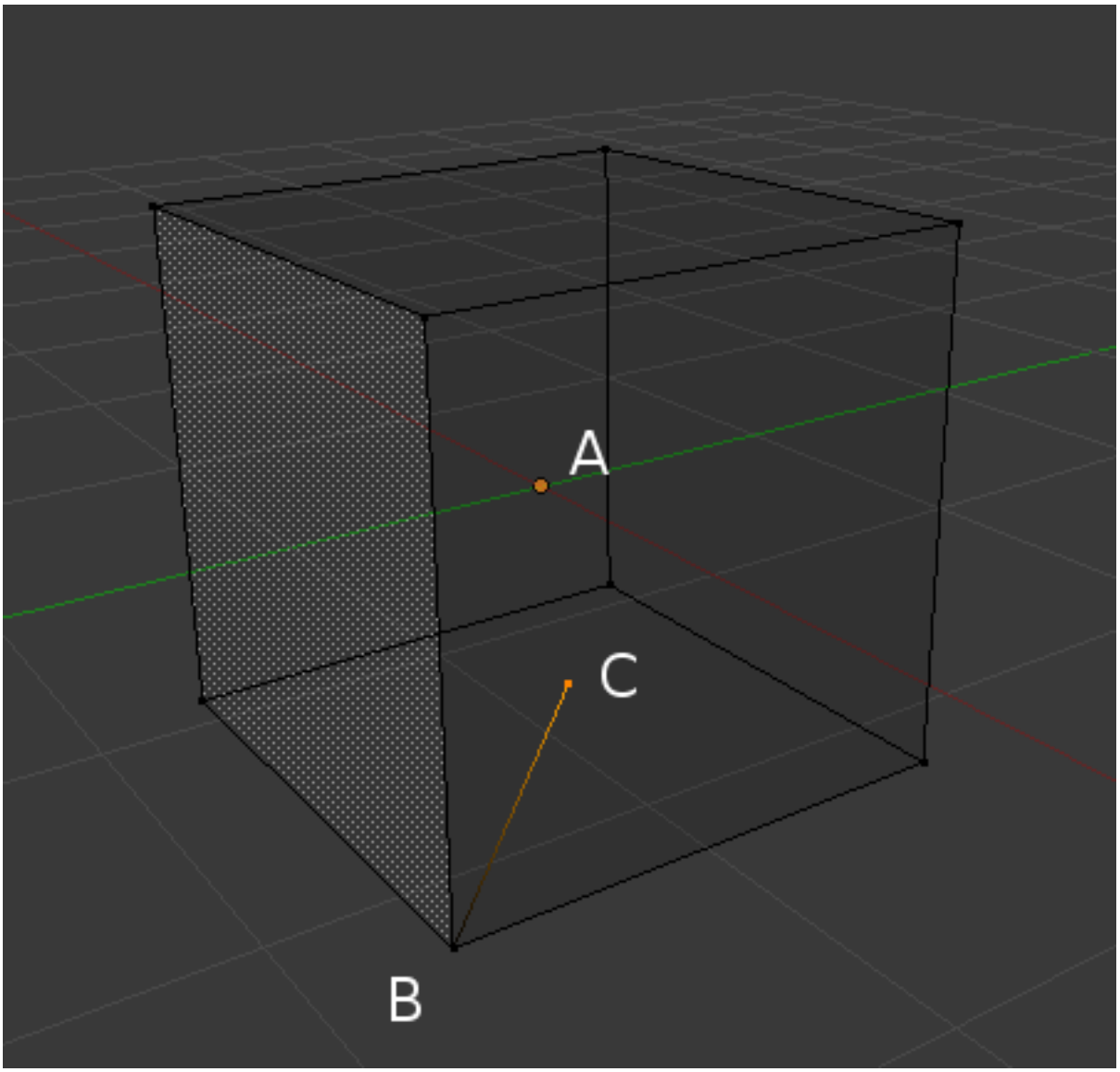


Fig. 2.366: Vertex example.

Triangles are always flat and therefore easy to calculate. On the other hand, quadrangles "deform well" and are therefore preferred for subdivision modeling.

While you could build a cube with triangular faces, it would just look more confusing in *Edit Mode*.

Loops

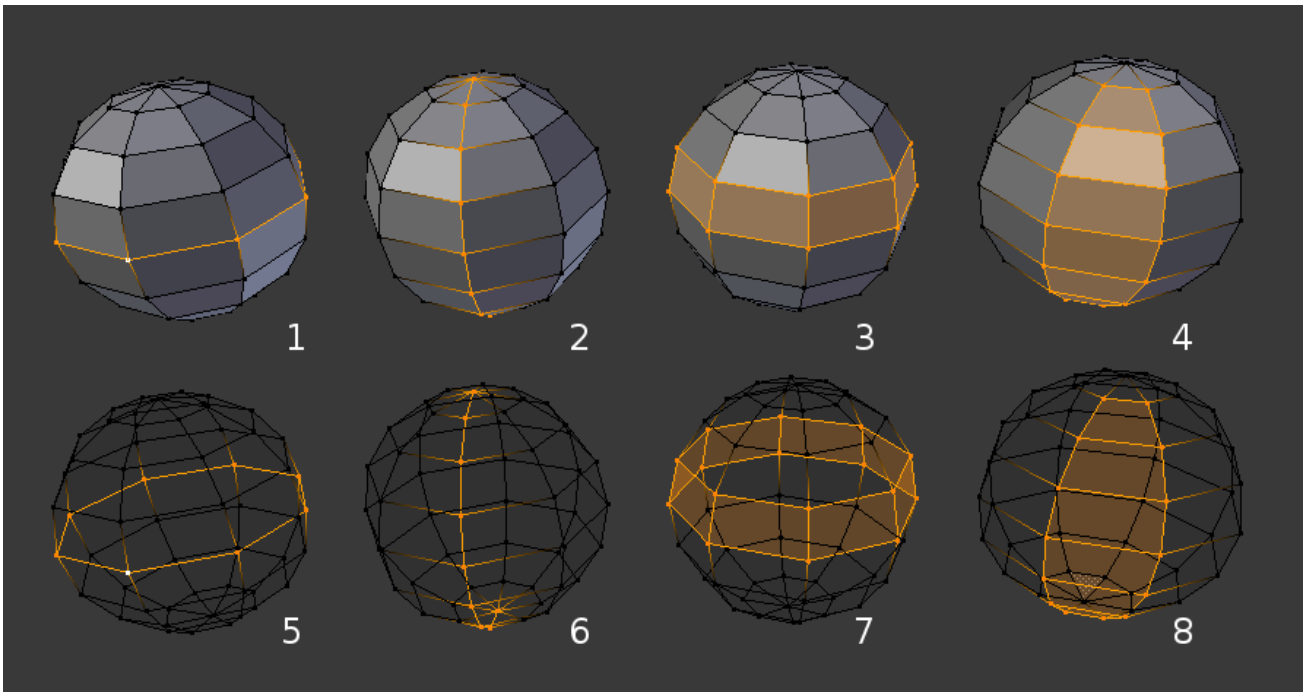


Fig. 2.367: Edge and Face Loops.

Edge and *Face Loops* are sets of faces or edges that form continuous "loops" as shown in Fig. *Edge and Face Loops.*. The top row (1 - 4) shows a solid view, the bottom row (5 - 8) a wireframe view of the same loops.

Note: Note that loops (2 and 4) do not go around the whole model. Loops stop at so called poles because there is no unique way to continue a loop from a pole. Poles are vertices that are connected to either three, five, or more edges. Accordingly, vertices connected to exactly one, two or four edges are not poles.

In the image above, loops that do not end in poles are cyclic (1 and 3). They start and end at the same vertex and divide the model into two partitions. Loops can be a quick and powerful tool to work with specific, continuous regions of a mesh and are a prerequisite for organic character animation. For a detailed description of how to work with loops in Blender, see: *Advanced Selection*.

Edge Loops

Loops (1 and 2) in Fig Edge and Face Loops are edge Loops. They connect vertices so that each one on the loop has exactly two neighbors that are not on the loop and placed on both sides of the loop (except the start and end vertex in case of poles).

Edge Loops are an important concept especially in organic (subsurface) modeling and character animation. When used correctly, they allow you to build models with relatively few vertices that look very natural when used as subdivision surfaces and deform very well in animation.

Take Fig. *Edge and Face Loops.* in organic modeling as an example: the edge loops follow the natural contours and deformation lines of the skin and the underlying muscles and are more dense in areas that deform more when the character moves, for example at the shoulders or knees.

Further details on working with Edge Loops can be found in *Edge Loop Selection*.

Face Loops

These are a logical extension of Edge Loops in that they consist of the faces between two Edge Loops, as shown in loops (3 and 4) in Fig. *Edge and Face Loops.*. Note that for non-circular loops (4) the faces containing the poles are not included in a Face Loop.

Further details on working with Face Loops can be found in *Face Loop Selection*.

Primitives

Reference

Mode: Object Mode
Menu: *Add* → *Mesh*
Hotkey: `Shift-A`

A common object type used in a 3D scene is a mesh. Blender comes with a number of "primitive" mesh shapes that you can start modeling from.

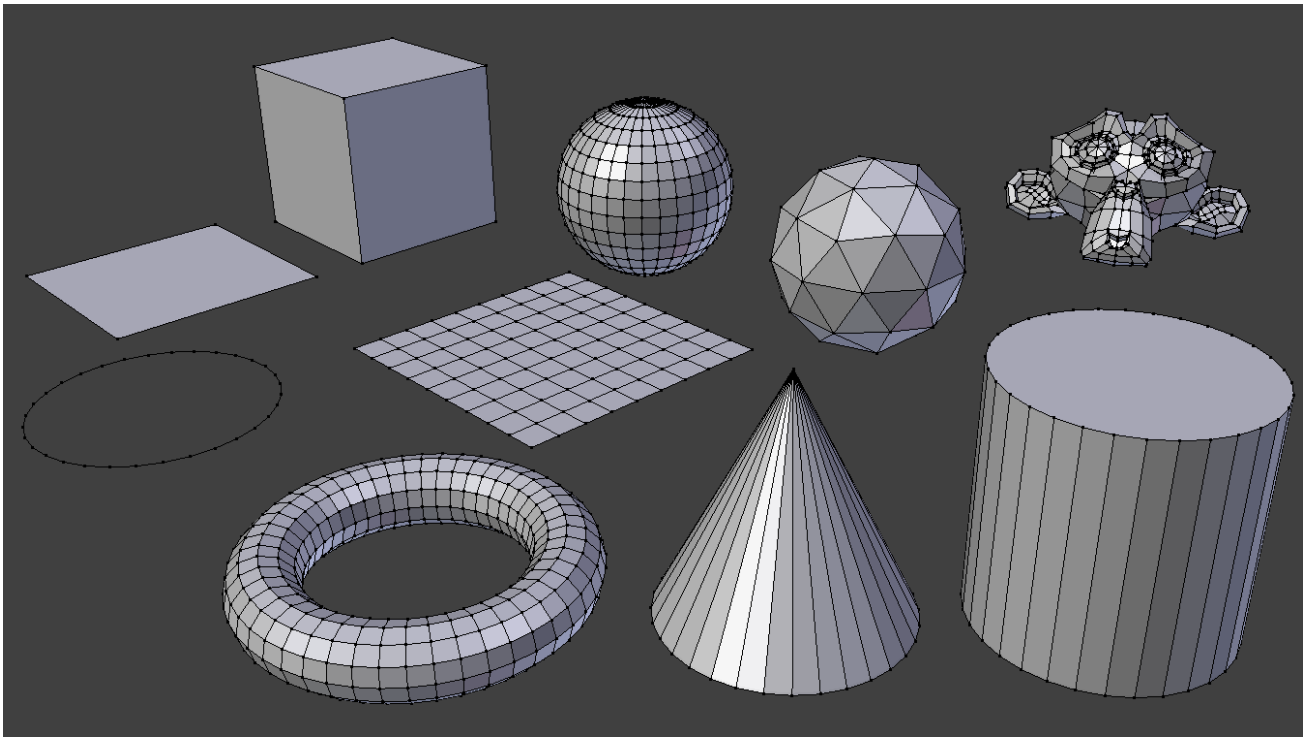


Fig. 2.368: Blender's standard primitives.

Note: Note about planar primitives

You can make a planar mesh three-dimensional by moving one or more of the vertices out of its plane (applies to *Plane*, *Circle* and *Grid*). A simple circle is actually often used as a starting point to create even the most complex of meshes.

Common Options

The Option can be specified in the Operator panel in the *Tool Shelf*, which appears when the object is created. Options included in more than one primitive are:

Vertices, Segments, Subdivisions Since the edges of a mesh are straight specifying a number of vertices produces polygonal shapes. i.e. six vertices for an hexagon. The higher the vertex count the closer a circular/spherical shape will be approximated and the shape will appear smoother.

Radius, Size Sets the starting size.

Generate UVs ToDo.

Plane

The standard plane is a single quad face, which is composed out of four vertices, four edges, and one face. It is like a piece of paper lying on a table; it is not a three-dimensional object because it is flat and has no thickness. Objects that can be created with planes include floors, tabletops, or mirrors.

Cube

A standard cube contains eight vertices, twelve edges, and six faces, and is a three-dimensional object. Objects that can be created out of cubes include dice, boxes, or crates.

Circle

Vertices The number of vertices that define the circle or polygon.

Fill Type Set how the circle will be filled.

Triangle Fan Fill with triangular faces which share a vertex in the middle.

N-gon Fill with a single n-gon.

Nothing Do not fill. Creates only the outer ring of vertices.

UV Sphere

A standard UV sphere is made out of quad faces and a triangle fan at the top and bottom. It can be used for texturing.

Segments Number of vertical segments. Like the Earth's meridians, going pole to pole.

Rings Number of horizontal segments. These are like the Earth's parallels.

> **Note:** Rings are face loops and not edge loops, which would be minus one.

Icosphere

An icosphere is a polyhedra sphere made up of triangles. Icospheres are normally used to achieve a more isotropical layout of vertices than a UV sphere.

Subdivisions How many recursions are used to define the sphere. At level 1 the Icosphere is an icosahedron, a solid with 20 equilateral triangular faces. Any increasing level of subdivision splits each triangular face into four triangles.

Note: Subdividing an icosphere rises the vertex count very high even with few iterations (10 times creates 5,242,880 triangles), Adding such a dense mesh is a sure way to cause the program to crash.

Cylinder

Objects that can be created out of cylinders include handles or rods.

Vertices The number of vertical edges between the circles used to define the cylinder or prism.

Depth Sets the starting height for the cylinder.

Cap Fill Type Similar to circle (see above). When set to none, the created object will be a tube. Objects that can be created out of tubes include pipes or drinking glasses (the basic difference between a cylinder and a tube is that the former has closed ends).

Cone

Objects that can be created out of cones include spikes or pointed hats.

Vertices The number of vertical edges between the circles or tip, used to define the cone or pyramid.

Radius 1 Sets the radius of the circular base of the cone.

Radius 2 Sets the radius of the tip of the cone. which will creates a frustum. A value of 0 will produce a standard cone shape.

Depth Sets the starting height for the cone.

Base Fill Type Similar to circle (see above).

Torus

A doughnut-shaped primitive created by rotating a circle around an axis. The overall dimensions can be defined by two methods.

Operator Presets Torus preset settings for reuse. These presets are stored as scripts in the proper presets directory.

Major Segments Number of segments for the main ring of the torus. If you think of a torus as a "spin" operation around an axis, this is how many steps in the spin.

Minor segments Number of segments for the minor ring of the torus. This is the number of vertices of each circular segment.

Torus Dimensions

Add Mode Change the way the torus is defined.

Major/Minor, Exterior/Interior

Major Radius Radius from the origin to the center of the cross sections.

Minor Radius Radius of the torus's cross section.

Exterior Radius If viewed along the major axis, this is the radius from the center to the outer edge.

Interior Radius If viewed along the major axis, this is the radius of the hole in the center.

Grid

A regular quadratic grid which is a subdivided plane. Example objects that can be created out of grids include landscapes and other organic surfaces.

X Subdivisions The number of spans in the X axis.

Y Subdivisions The number of spans in the Y axis.

Monkey

This is a gift from old NaN to the community and is seen as a programmer's joke or "Easter Egg". It creates a monkey's head once you press the *Monkey* button. The Monkey's name is "Suzanne" and is Blender's mascot. Suzanne is very useful as a standard test mesh, much like the Utah Tea Pot or the Stanford Bunny.

Note: Add-ons

In addition to the basic geometric primitives, Blender has a number of script generated meshes to offer as pre-installed add-ons. These become available when enabled in the *User Preferences* (filter by *Add Mesh*).

Selecting

Introduction

There are many ways to select elements, and it depends on what *Mesh Select Mode* you are in as to what selection tools are available. First we will go through these modes and after that a look is taken at basic selection tools.

Selection Mode

Select Mode Header Widgets

Reference

Mode: Edit Mode
Menu: *3D View Header* → *Select Mode*
Hotkey: `Ctrl-Tab`

In *Edit Mode* there are three different selection modes. You can enter the different modes by selecting one of the three buttons in the header.

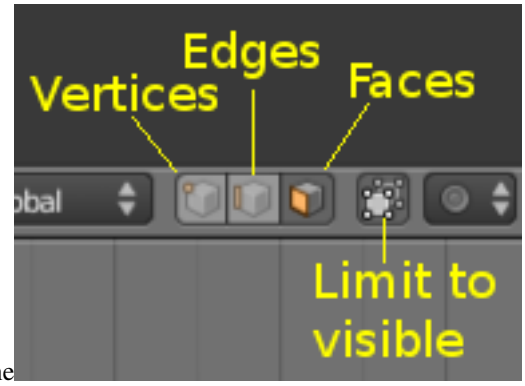


Fig. 2.369: Edit Mode selection buttons.

Vertices In this mode vertices are drawn as points.

Selected vertices are drawn in orange, unselected vertices in black, and the active or last selected vertex in white.

Edges In this mode the vertices are not drawn.

Instead the selected edges are drawn in orange, unselected edges black, and the active or last selected edge in white.

None

Faces In this mode the faces are drawn with a selection point in the middle which is used for selecting a face.

Selected faces and their selection point are drawn in orange, unselected faces are drawn in black, and the active or last selected face is highlighted in white.

When using these buttons, you can make use of modifier keys, see: *Switching Select Mode*.

Almost all tools are available in all three mesh selection modes. So you can *Rotate*, *Scale*, *Extrude*, etc. in all modes. Of course rotating and scaling a *single* vertex will not do anything useful *(without setting the pivot point to another location)*, so some tools are more or less applicable in some modes.

Switching Select Mode

When switching modes in an "ascendant" way (i.e. from simpler to more complex), from *Vertices* to *Edges* and from *Edges* to *Faces*, the selected parts will still be selected if they form a complete element in the new mode.

For example, if all four edges in a face are selected, switching from *Edges* mode to *Faces* mode will keep the face selected. All selected parts that do not form a complete set in the new mode will be unselected.

Hence, switching in a "descendant" way (i.e. from more complex to simpler), all elements defining the "high-level" element (like a face) will be selected (the four vertices or edges of a quadrangle, for example).

Multiple Selection Modes

By holding `Shift-LMB` when selecting a selection mode, you can enable multiple *Selection Modes* at once.

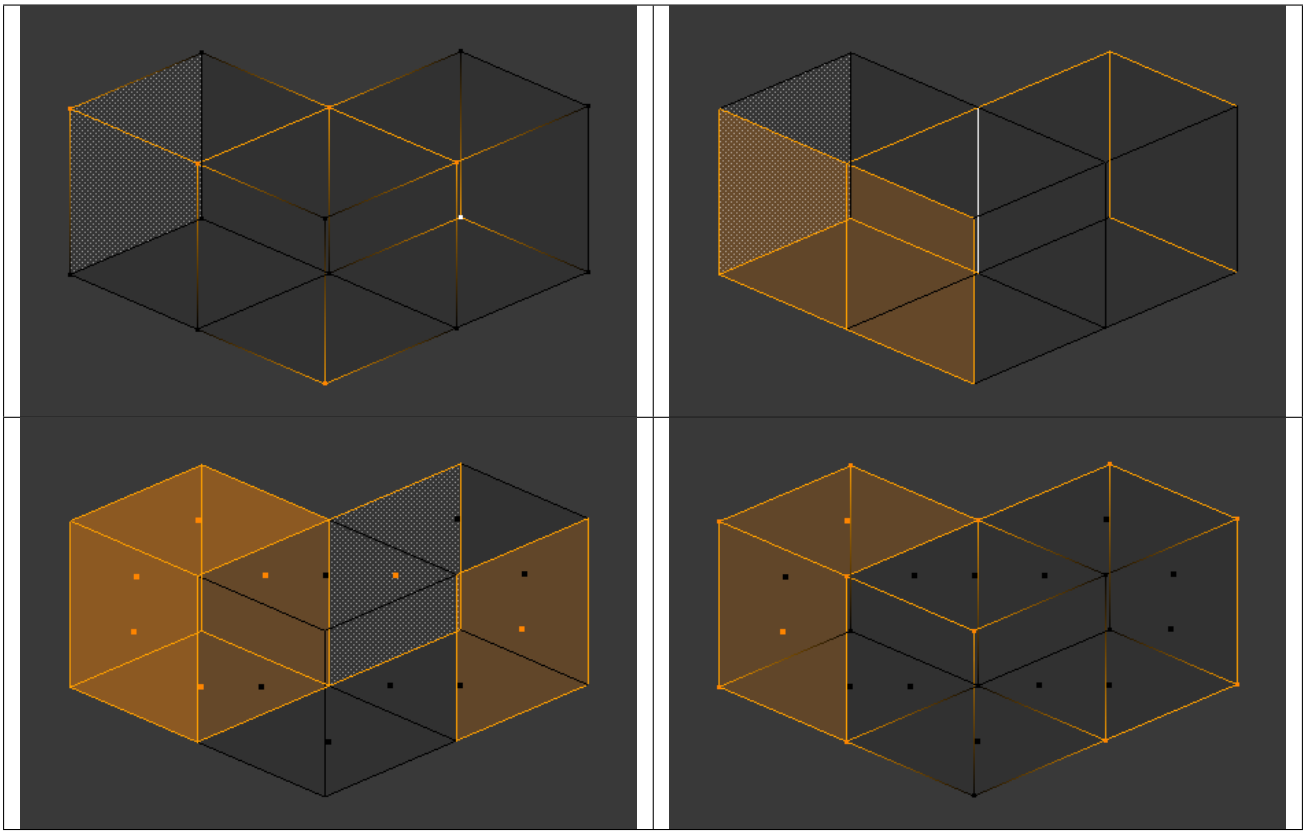
This allows you to quickly select Vertices/Edges/Faces, without first having to switch modes.

Expanding Selection Modes

By holding `Ctrl` when selecting a higher selection mode, all elements touching the current selection will be added, even if the selection does not form a complete higher element.

See Fig. *Selection Modes.* for examples of the different modes.

Table 2.12: Mixed mode example.



Selection Tools

The select menu in edit mode contains tools for selecting components. These are described in more detail in the following pages.

Border Select `B` Enables a rectangular region for selection

Circle Select `C` Enables a circular shaped region for selection

(De)select All `A` Select all or none of the mesh components.

Invert Selection `Ctrl-I` Selects all geometries that are not selected, and deselect currently selected components.

Select Random Selects a random group of vertices, edges, or faces, based on a percentage value.

Checker Deselect Deselect alternating faces, to create a checker like pattern.

Select Sharp Edges This option will select all edges that are between two faces forming an angle less than a given value, which is asked you *via* a small pop-up menu. The lower is this angle limit, the sharper will be the selected edges. At 180, **all** *manifold* edges will be selected.

Linked Flat Faces `Ctrl-Shift-Alt-F` Select connected faces based on a threshold of the angle between them. This is useful for selecting faces that are planar.

Interior Faces Select faces where all edges have more than two faces.

Side of Active Selects all data on the mesh in a single axis

Select Faces by Sides Selects all faces that have a specified number of edges.

Non Manifold `Ctrl-Shift-Alt-M` Selects *non-manifold* geometry. See *Mesh Advanced Selection*.

Loose Select all vertices or edges that do not form part of a face.

Similar **Shift-G** Select geometry based on how similar certain properties are to it.

> **Note:** The items shown in the menu depend on the *Selection Mode*.

More **Ctrl-NumpadPlus** Propagates selection by adding geometry that are adjacent to selected elements.

Less **Ctrl-NumpadMinus** Deselects geometry that form the bounds of the current selection

Mirror Select mesh items at the mirrored location.

Pick Linked **L** Selects all geometries connected to the geometry under the cursor.

Linked **Ctrl-L** Selects all geometries connected to the current selection.

Vertex Path Selects a vertex path between two selected vertices

Edge Loop Selects a loop of edges from a selected edge

Edge Ring Selects edges parallel to a selected edge in the same ring of faces

Loop Inner-Region Converts a closed selection of edges to the region of faces it encloses

Boundary Loop Converts a selection of faces to the ring of edges enclosing it

Basic Selection

Reference

Mode: Edit Mode
Hotkey: RMB and Shift-RMB

The most common way to select an element is to RMB on that item; this will replace the existing selection with the new item.

Adding to a Selection

To add to the existing selection, hold down Shift while right clicking. Clicking again on a selected item will deselect it.

As in *Object Mode*, there is a unique *active* element, displayed in a lighter shade (in general, the last element selected). Depending on the tools used, this element might be very important!

Note that there is no option to choose what element to select between overlapping ones (like the Alt-RMB click in *Object Mode*). However, if you are in solid, shaded, or textured viewport shading mode (not bounding box or wireframe), you will have a fourth button in the header that looks like a cube, just right of the select mode ones.

When enabled, this limits your ability to select based on visible elements (as if the object was solid), and prevents you from accidentally selecting, moving, deleting or otherwise working on backside or hidden items.

Selecting Elements in a Region

Reference

Mode: Edit Mode
Hotkey: B, C, and Ctrl-LMB click and drag

Region selection allows you to select groups of elements within a 2D region in your 3D View. The region can be either a circle or rectangle. The circular region is only available in *Edit Mode*. The rectangular region, or *Border Select, is available in both *Edit Mode* and *Object Mode*.

Note: What is selected using both these tools is affected by the *Limit Selection to visible* feature (available under the 3D View) in *Solid Viewport Shading Mode*.

For example,

- in solid shading mode and face selection mode, all faces *within* the selection area will be selected;
- while in the wireframe shading mode and face selection mode, only faces whose handle are within the selection area will be selected.

Rectangular region (Border select)

Border Select is available in either *Edit Mode* or *Object Mode*. To activate the tool use the `B`. Use *Border Select* to select a group of objects by drawing a rectangle while holding down `LMB`. In doing this you will select all objects that lie within or touch this rectangle. If any object that was last active appears in the group it will become selected *and* active.

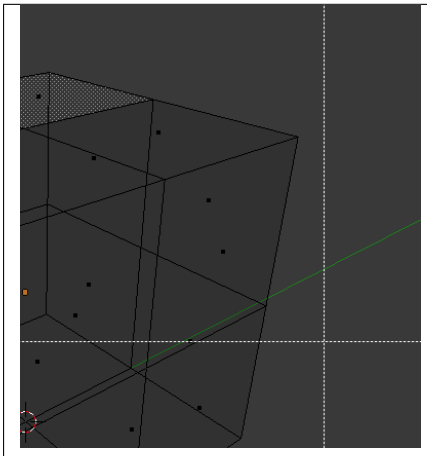


Fig. 2.374: Start.

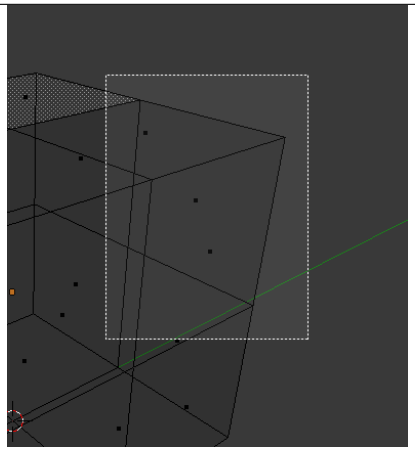


Fig. 2.375: Selecting.

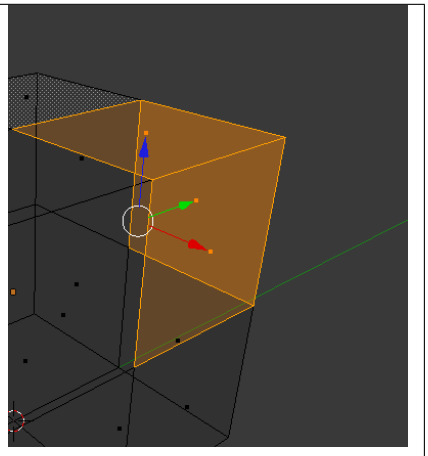


Fig. 2.376: Complete.

In Fig. *Start.*, *Border Select* has been activated and is indicated by showing a dotted cross-hair cursor. In Fig. *Selecting.* the *selection region* is being chosen by drawing a rectangle with the `LMB`. The selection area is only covering the selection handles of three faces. Finally, by releasing `LMB` the selection is complete; see Fig. *Complete.*.

Note: Border select adds to the previous selection, so in order to select only the contents of the rectangle, deselect all with `A` first. In addition, you can use `MMB` while you draw the border to deselect all objects within the rectangle.

Circular region

This selection tool is only available in *Edit Mode* and can be activated with `C`. Once in this mode the cursor changes to a dashed cross-hair with a 2D circle surrounding it. The tool will operate on whatever the current select mode is. Clicking or dragging with the `LMB`, causing elements to be inside the circle will cause those elements to be selected.

You can enlarge or shrink the circle region using `NumpadPlus` and `NumpadMinus`, or the `Wheel`.

Table 2.13: After.

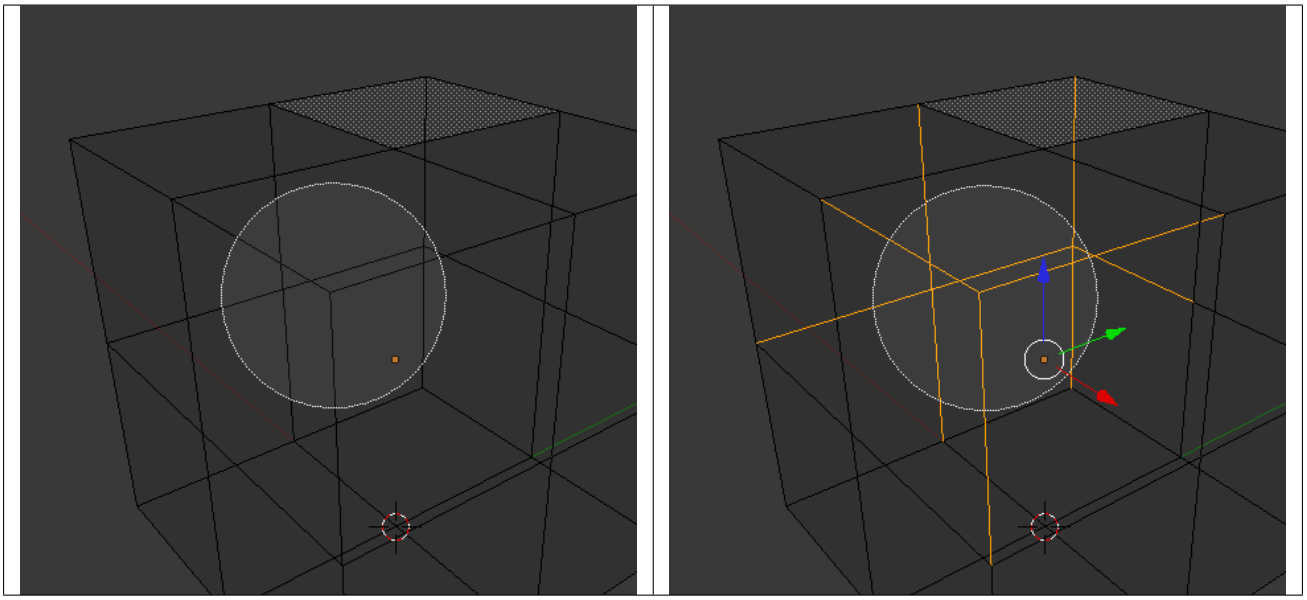


Fig. *Circle Region Select.* is an example of selecting edges while in *Edge Select Mode*. As soon as an edge intersects the circle the edge becomes selected. The tool is interactive such that edges are selected while the circle region is being dragged with the LMB.

If you want to deselect elements, hold MMB and begin clicking or dragging again.

For *Faces* select mode, the circle must intersect the face indicators usually represented by small pixel squares; one at the center of each face.

To exit from this tool, click RMB, or press Esc.

Lasso region

Lasso select is similar to *Border* select in that you select objects based on a region, except *Lasso* is a hand-drawn region that generally forms a circular/round-shaped form; kind of like a lasso.

Lasso is available in either *Edit Mode* or *Object Mode*. To activate the tool use the Ctrl-LMB while dragging. The one difference between *Lasso* and *Border* select is that in *Object Mode*, *Lasso* only selects objects where the lasso region intersects the objects' center.

To deselect, use Ctrl-Shift-LMB while dragging.

Table 2.14: Complete.

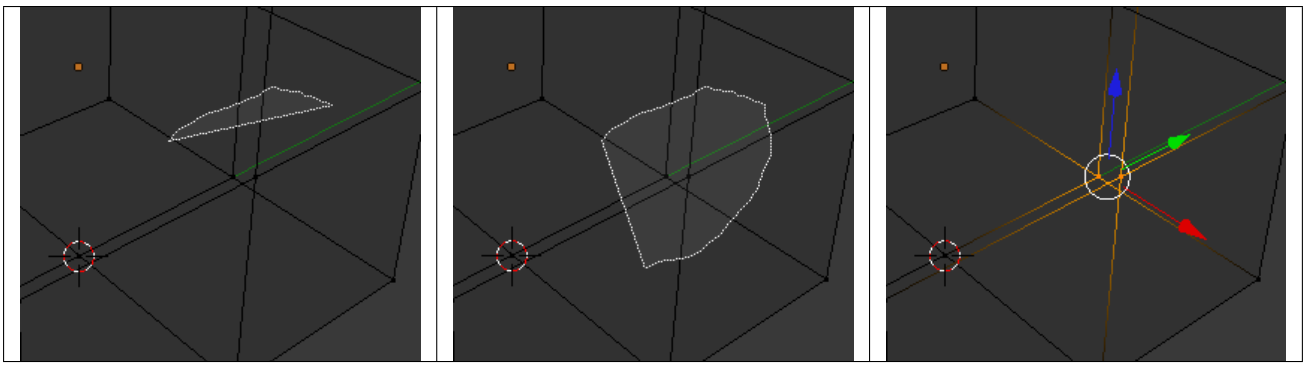


Fig. *Lasso selection.* is an example of using the *Lasso select tool* in *Vertex Select Mode*.

Additional Selection Tools

The select menu in edit mode contains additional tool for selecting components:

(De)select All `A` Select all or none of the mesh components.

Invert Selection `Ctrl-I` Selects all components that are not selected, and deselect currently selected components.

More `Ctrl-NumpadPlus` Propagates selection by adding components that are adjacent to selected elements.

Less `Ctrl-NumpadMinus` Deselects components that form the bounds of the current selection

Advanced Selection

The select menu in edit mode contains additional tool for selecting components:

Mirror Select mesh items at the mirrored location.

Linked Selects all components that are connected to the current selection. (see *Select Linked*)

Random Selects a random group of vertices, edges, or faces, based on a percentage value.

Checker Deselect Deselect alternating faces, to create a checker like pattern.

Select Every N Number of Vertices Selects vertices that are multiples of N.

Sharp Edges This tool selects all edges between two faces forming an angle greater than the angle option, Where an increasing angle selects sharper edges.

Linked Flat Faces `Ctrl-Shift-Alt-F` Select connected faces based on a threshold of the angle between them. This is useful for selecting faces that are planar.

Non Manifold `Ctrl-Shift-Alt-M` Selects the *non-manifold* geometry of a mesh. This entry is available when editing a mesh, in Vertex and Edge selection modes only. The *redo* panel provides several selection options:

Extend Lets you extend the current selection.

Wire Selects all the edges that do not belong to any face.

Boundaries Selects edges in boundaries and holes.

Multiple Faces Selects edges that belong to three or more faces.

Non Contiguous Selects edges that belong to exactly two faces with opposite normals.

Vertices Selects vertices that belong to *wire* and *multiple face* edges, isolated vertices, and vertices that belong to non adjoining faces.

Interior Faces Select faces where all edges have more than two faces.

Side of Active Selects all data on the mesh in a single axis

Select Faces by Sides Selects all faces that have a specified number of edges.

Loose Geometry Select all vertices or edges that do not form part of a face.

Select Linked

Reference

Mode: Edit Mode
Menu: *Select* → *Linked*
Hotkey: `Ctrl-L`

Select parts of a mesh connected to already selected elements. This is often useful when a mesh has disconnected, overlapping parts, where isolating it any other way would be tedious.

To give more control, you can also enable delimiters so the selection is constrained by seams, sharp-edges, materials or UV islands.

Hint: You can also select linked data directly under the cursor, using the `L` shortcut to select or `Shift-L` to deselect linked.

This works differently in that it uses the geometry under the cursor instead of the existing selection.

Select Similar

Reference

Mode: Edit Mode
Menu: *Select* → *Similar...*
Hotkey: `Shift-G`

Select components that have similar attributes to the ones selected, based on a threshold that can be set in tool properties after activating the tool. Tool options change depending on the selection mode:

Vertex Selection Mode:

Normal Selects all vertices that have normals pointing in similar directions to those currently selected.

Amount of Adjacent Faces Selects all vertices that have the same number of faces connected to them.

Vertex Groups Selects all vertices in the same *vertex group*.

Amount of connecting edges Selects all vertices that have the same number of edges connected to them.

Edge Selection Mode:

Length Selects all edges that have a similar length as those already selected.

Direction Selects all edges that have a similar direction (angle) as those already selected.

Amount of Faces Around an Edge Selects all edges that belong to the same number of faces.

Face Angles Selects all edges that are between two faces forming a similar angle, as with those already selected.

Crease Selects all edges that have a similar *Crease* value as those already selected.

Bevel Selects all edges that have the same *Bevel Weight* as those already selected.

Seam Selects all edges that have the same *Seam* state as those already selected. *Seam* is a true/false setting used in *UV-texturing*.

Sharpness Selects all edges that have the same *Sharp* state as those already selected. *Sharp* is a true/false setting (a flag) used by the *Edge Split Modifier*.

Face Selection Mode:

Material Selects all faces that use the same material as those already selected.

Image Selects all faces that use the same UV-texture as those already selected (see *UV-texturing* pages).

Area Selects all faces that have a similar area as those already selected.

Polygon Sides Selects all faces that have the same number of edges.

Perimeter Selects all faces that have a similar perimeter as those already selected.

Normal Selects all faces that have a similar normal as those selected. This is a way to select faces that have the same orientation (angle).

Co-planar Selects all faces that are (nearly) in the same plane as those selected.

Selecting Loops

You can easily select loops of components:

Edge Loops and Vertex Loops

Reference

Mode: Edit Mode –> Vertex or Edge select mode
Menu: *Select* → *Edge Loop* or *Mesh* → *Edges* → *Edge Loop*
Hotkey: `Alt-RMB` or `Ctrl-E` → *Edge Loop*

Holding `Alt` while selecting an edge selects a loop of edges that are connected in a line end to end, passing through the edge under the mouse pointer. Holding `Alt-Shift` while clicking adds to the current selection.

Edge loops can also be selected based on an existing edge selection, using either *Select* → *Edge Loop*, or the *Edge Loop Select* option of the *Edge Specials* menu `Ctrl-E`.

Note: *Vertex* mode

In *Vertex* select mode, you can also select edge loops, by using the same hotkeys, and clicking on the *edges* (not on the vertices).

The left sphere shows an edge that was selected longitudinally. Notice how the loop is open. This is because the algorithm hit the vertices at the poles and terminated because the vertices at the pole connect to more than four edges. However, the right sphere shows an edge that was selected latitudinally and has formed a closed loop. This is because the algorithm hit the first edge that it started with.

Face Loops

Reference

Mode: Edit Mode –> Face or Vertex select modes
Hotkey: `Alt-RMB`

In face select mode, holding `Alt` while selecting an *edge* selects a loop of faces that are connected in a line end to end, along their opposite edges.

In vertex select mode, the same can be accomplished by using `Ctrl-Alt` to select an edge, which selects the face loop implicitly.

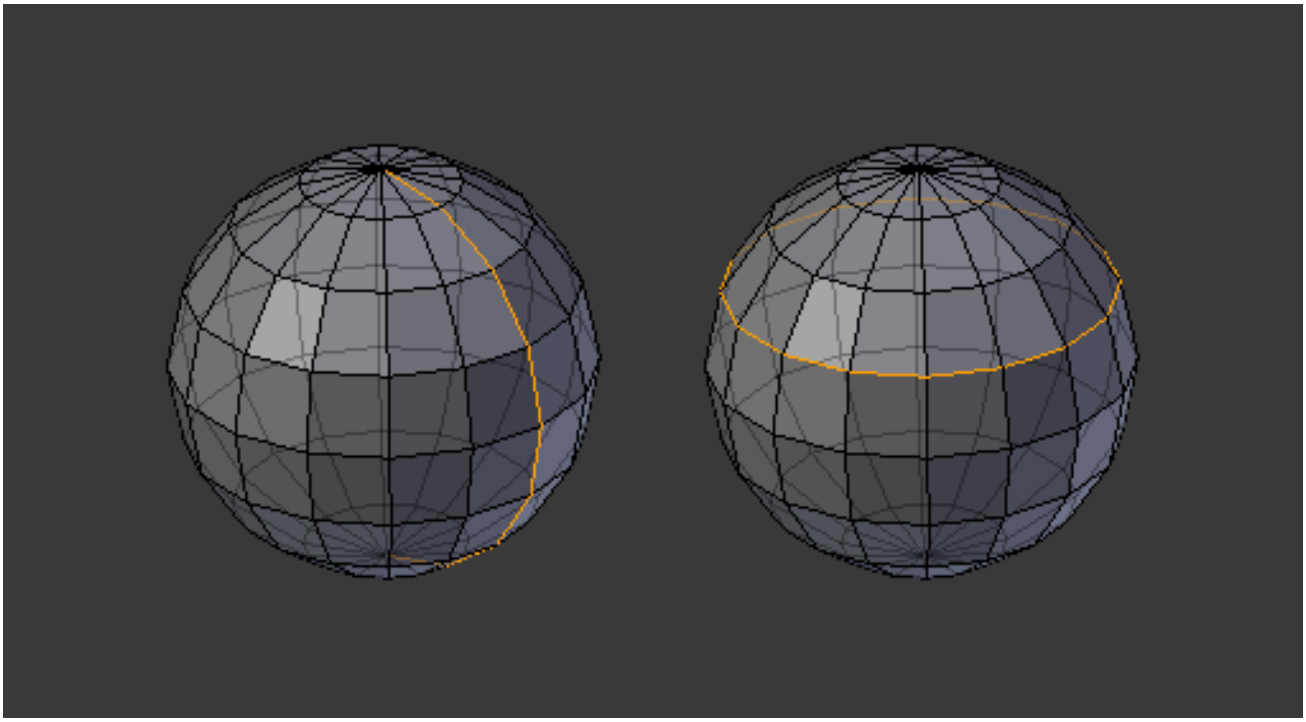


Fig. 2.382: Longitudinal and latitudinal edge loops.

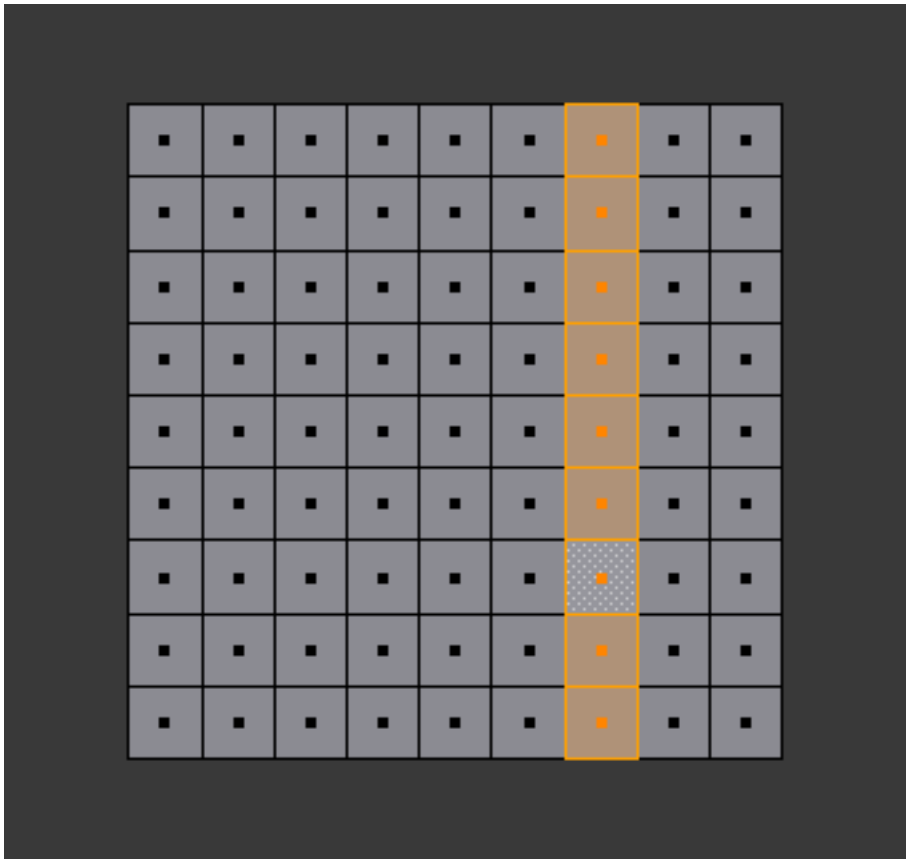


Fig. 2.383: Face loop selection.

This face loop was selected by clicking with Alt-RMB on an edge, in *face* select mode. The loop extends perpendicular from the edge that was selected.

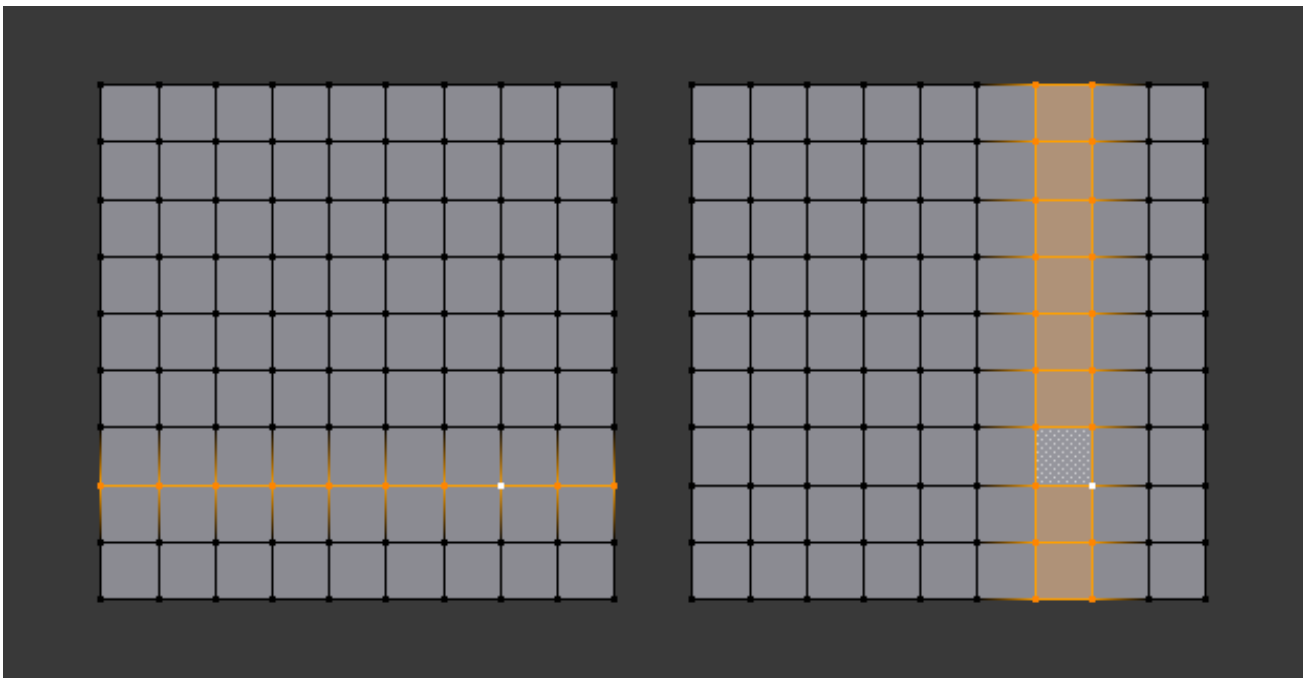


Fig. 2.384: Alt versus Ctrl-Alt in vertex select mode.

A face loop can also be selected in *Vertex* select mode. Technically Ctrl-Alt-RMB will select an *Edge Ring*, however, in *Vertex* select mode, selecting an *Edge Ring* implicitly selects a *Face Loop* since selecting opposite edges of a face implicitly selects the entire face.

Edge Ring

Reference

Mode: Edit Mode –> Edge select mode
Menu: *Select* → *Edge Ring* or *Mesh* → *Edges* → *Edge Ring*
Hotkey: Ctrl-Alt-RMB or Ctrl-E → *Select* → *Edge Ring*

In *Edge* select mode, holding Ctrl-Alt while selecting an edge selects a sequence of edges that are not connected, but on opposite sides to each other continuing along a *face loop*.

As with edge loops, you can also select edge rings based on current selection, using either *Select* → *Edge Ring*, or the *Edge Ring Select* option of the *Edge Specials* menu Ctrl-E.

Note: *Vertex* mode

In *Vertex* select mode, you can use the same hotkeys when *clicking on the edges* (not on the vertices), but this will directly select the corresponding face loop...

In Fig. *A selected edge loop, and a selected edge ring.* the same edge was clicked on, but two different "groups of edges" were selected, based on the different commands. One is based on edges during computation and the other is based on faces.

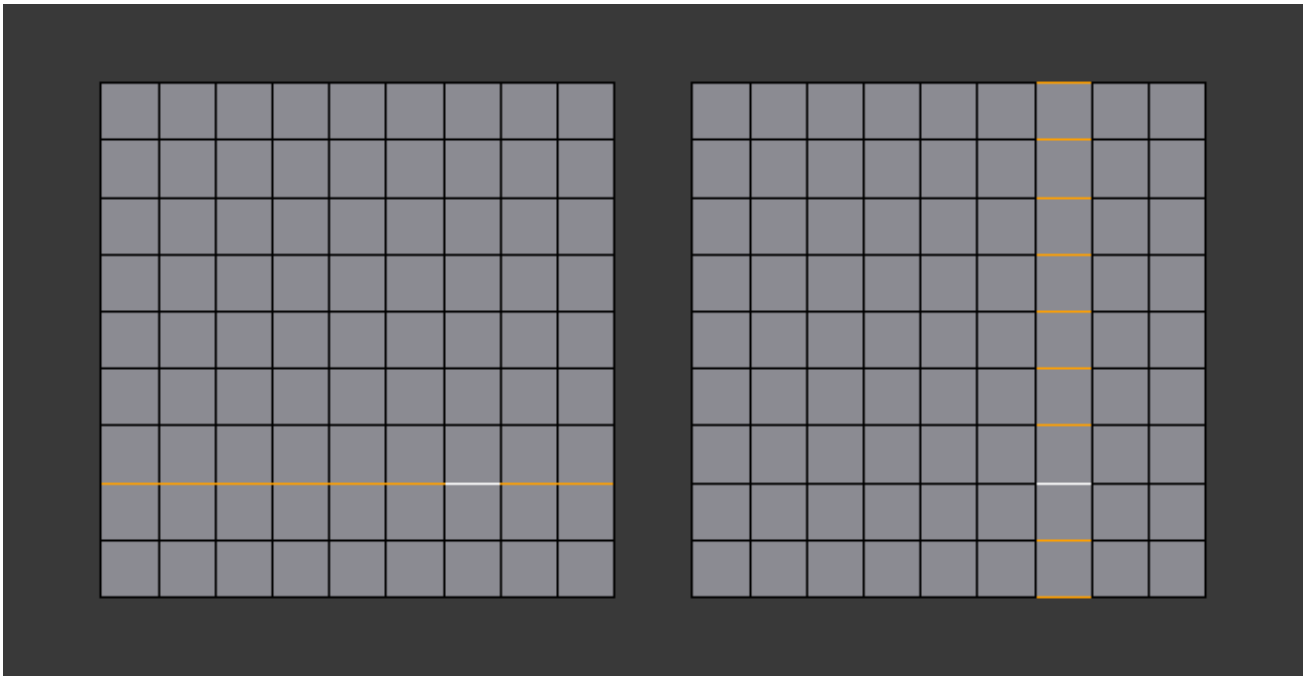


Fig. 2.385: A selected edge loop, and a selected edge ring.

Path Selection

Reference

Mode: Edit Mode
Hotkey: `Ctrl-RMB` and the menu item *Select* → *Shortest Path*

Selects all geometry along the shortest path from the active vertex/edge/face to the one which was selected.

Loop Inner-Region

Reference

Mode: Edit Mode –> Edge select mode
Menu: *Select* → *Select Loop Inner-Region* or *Mesh* → *Edges* → *Select Loop Inner-Region*
Hotkey: `Ctrl-E` → *Select Loop Inner-Region*

Select Loop Inner-Region selects all edges that are inside a closed loop of edges. While it is possible to use this operator in *Vertex* and *Face* selection modes, results may be unexpected. Note that if the selected loop of edges is not closed, then all connected edges on the mesh will be considered inside the loop.

Boundary Loop

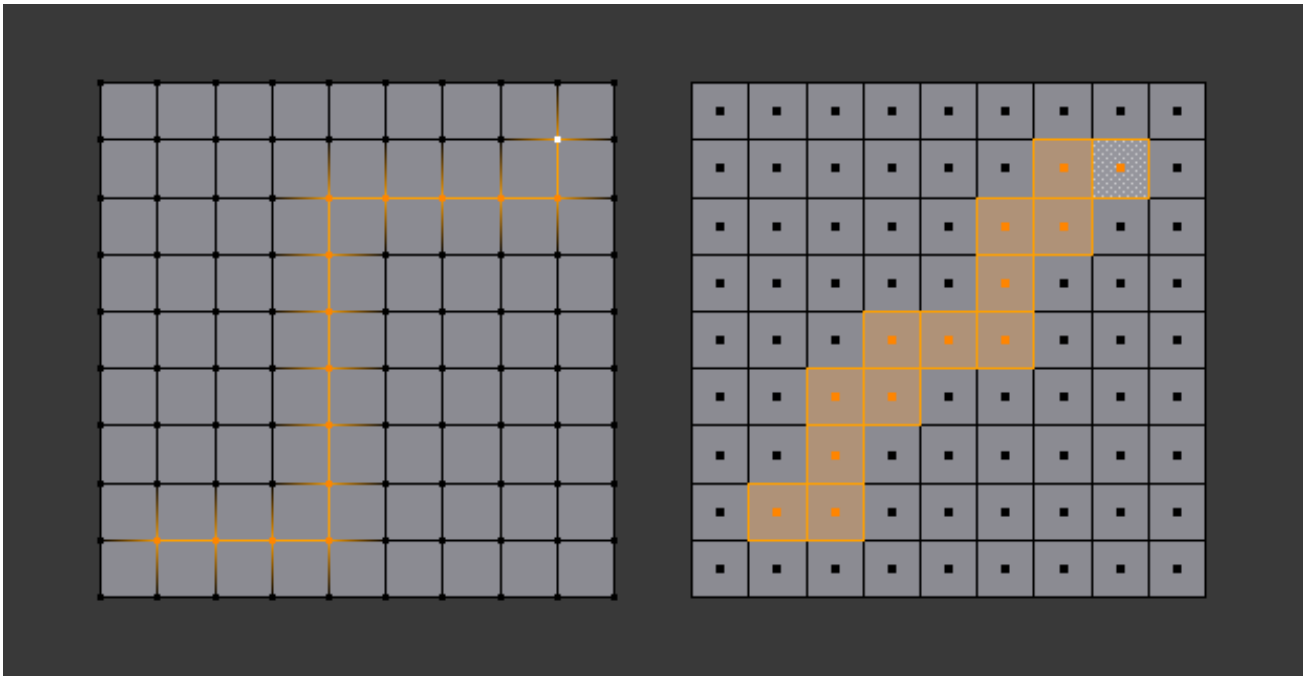


Fig. 2.386: Select a face or vertex path with Ctrl-RMB.

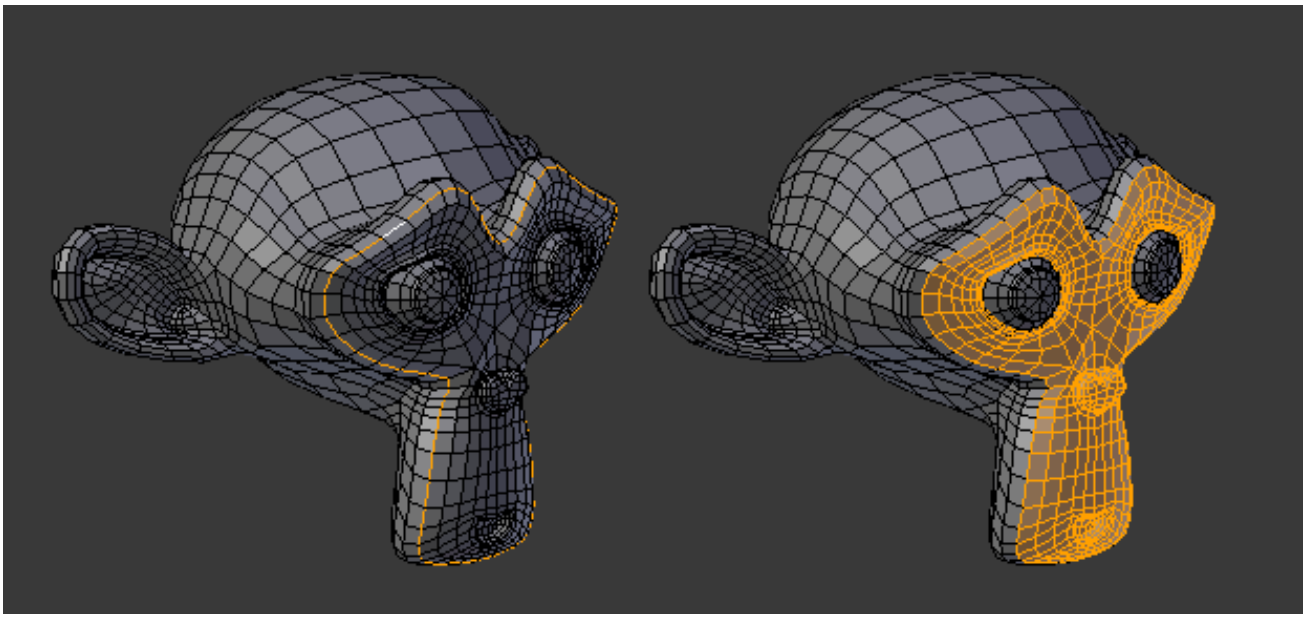


Fig. 2.387: Loop to Region.

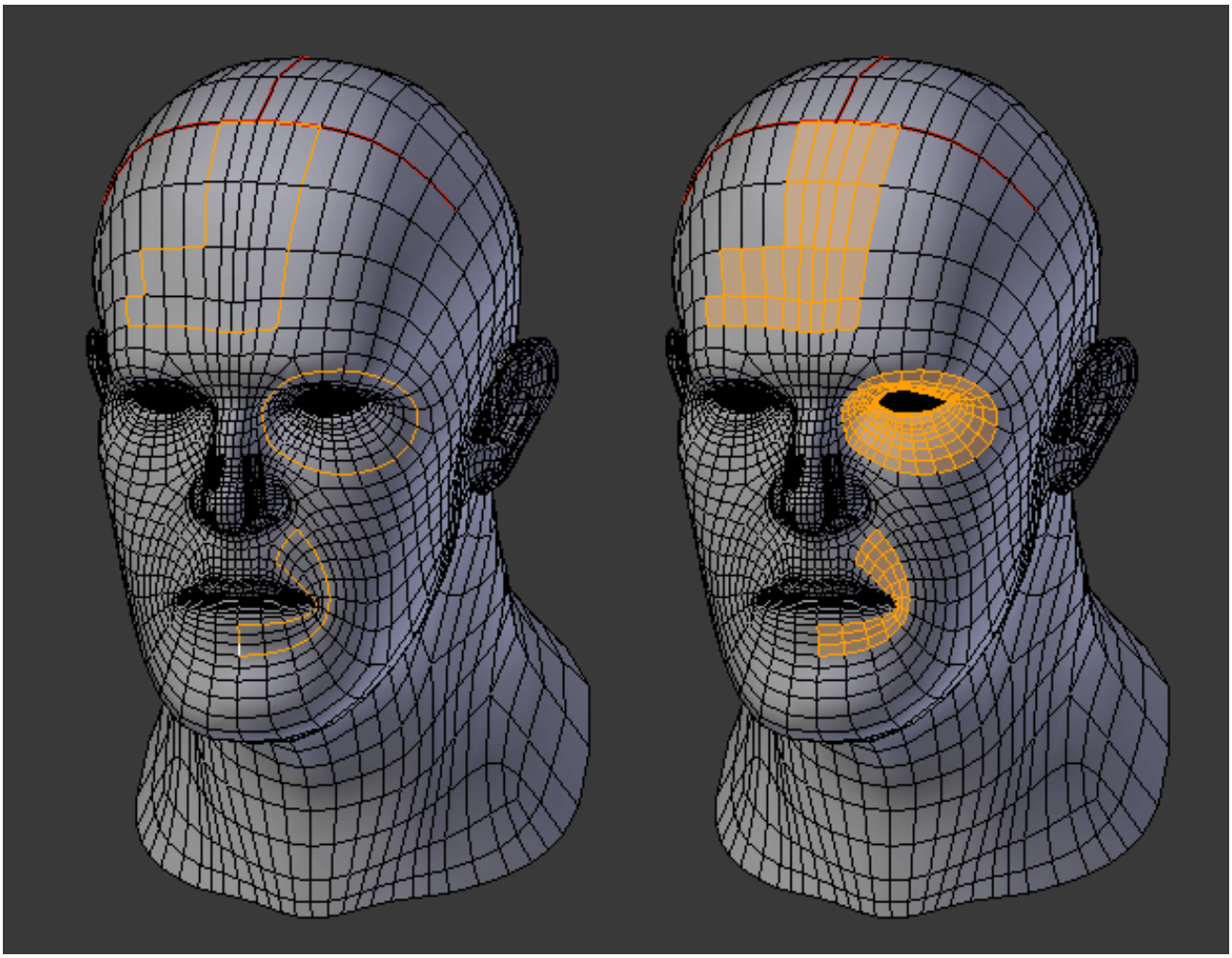


Fig. 2.388: This tool handles multiple loops fine, as you can see.

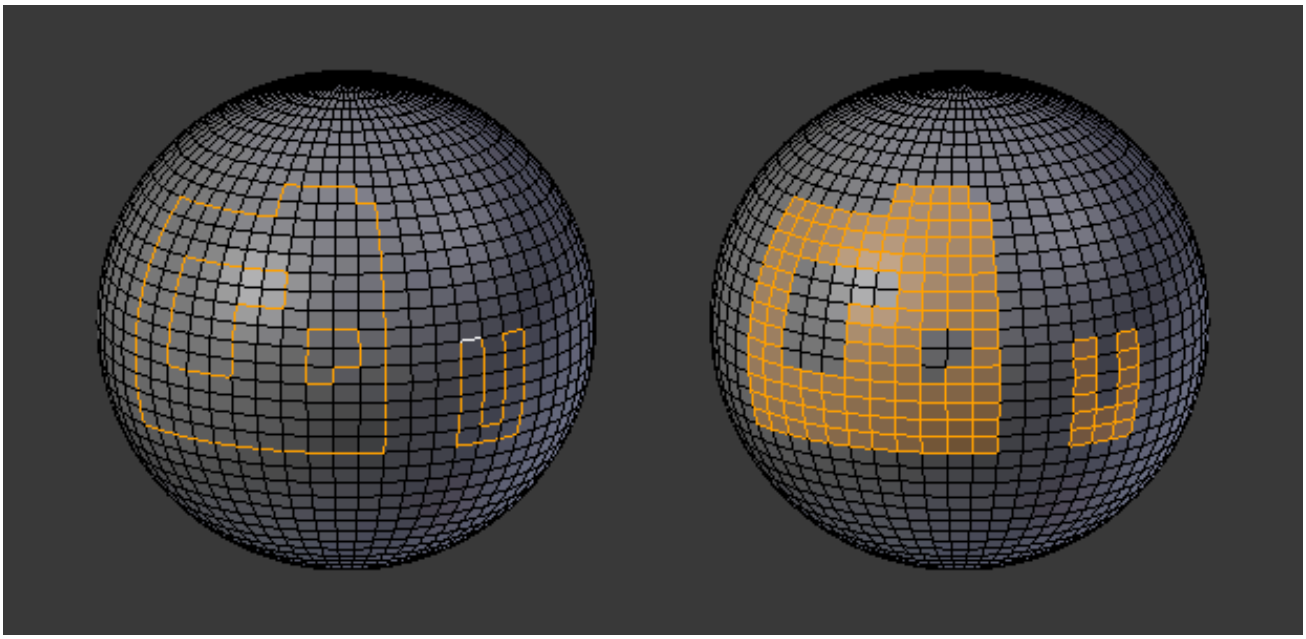


Fig. 2.389: This tool handles “holes” just fine as well.

Reference

Mode: Edit Mode –> Edge select mode
Menu: *Select* → *Select Boundary Loop* or *Mesh* → *Edges* → *Select Boundary Loop*
Hotkey: `Ctrl-E` → *Select Boundary Loop*

Select Boundary Loop does the opposite of *Select Loop Inner-Region*, based on all regions currently selected, it selects only the edges at the border of these regions. It can operate in any select mode, but will always switch to *Edge* select mode when run.

All this is much more simple to illustrates with examples:

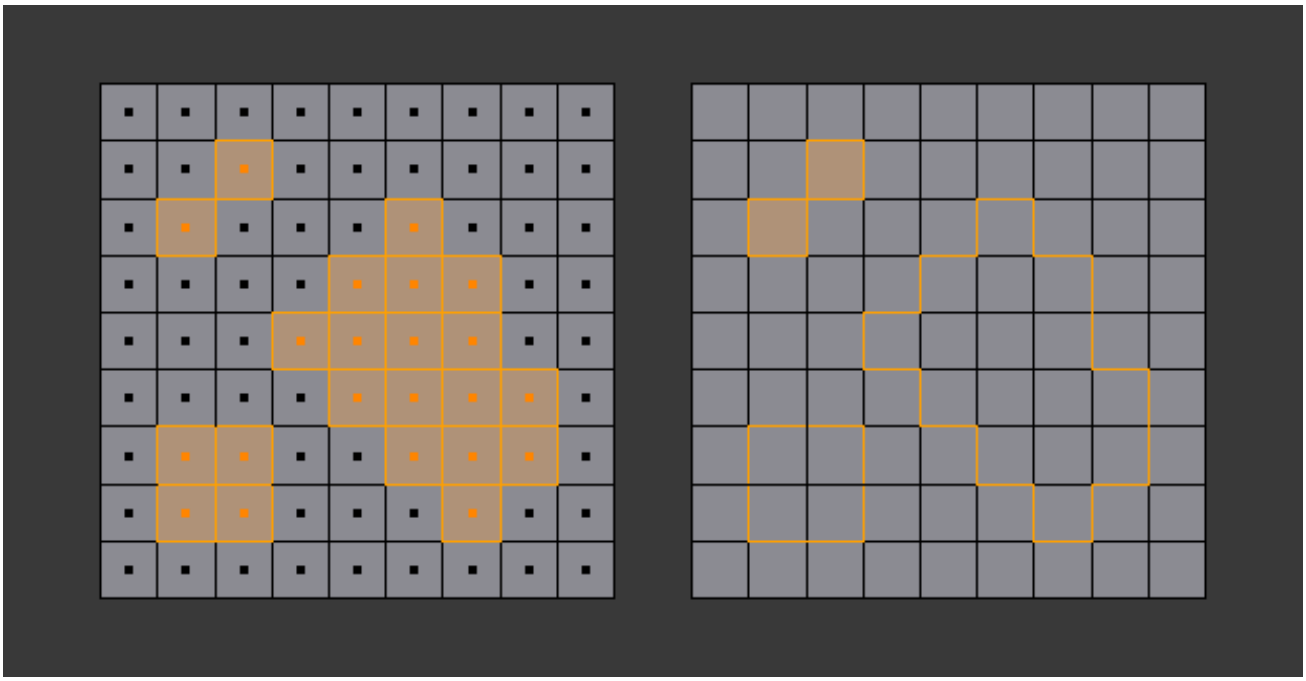


Fig. 2.390: Select Boundary Loop does the opposite and forces into Edge Select Mode.

Selecting Edges

Fig. 2.391: Buttons for the selection modes.

Edges can be selected in much the same way as vertices and faces by right-clicking them while Edge Select Mode is activated. Pressing `Shift` while clicking will add/subtract to the existing selection.

Edge Loops

Reference

Mode: Edit Mode (Mesh)
Menu: *Select* → *Edge Loop*
Hotkey: `Alt-RMB`, or `Shift-Alt-RMB` for modifying existing selection

Edge loops can be selected by first selecting an edge (vertex or edge selection mode), and then going to *Select* → *Edge Loop*. The shortcut `Alt-RMB` on an edge (either vertex or edge select mode) is a quicker and more powerful way of doing so. More powerful, because you can add/remove loops from an existing selection if you press `Shift` too.

Note, that if you want to select a loop while being in vertex select mode, you still have to perform the shortcut on an edge – while you, for just selecting vertices, would `RMB` on a vertex.

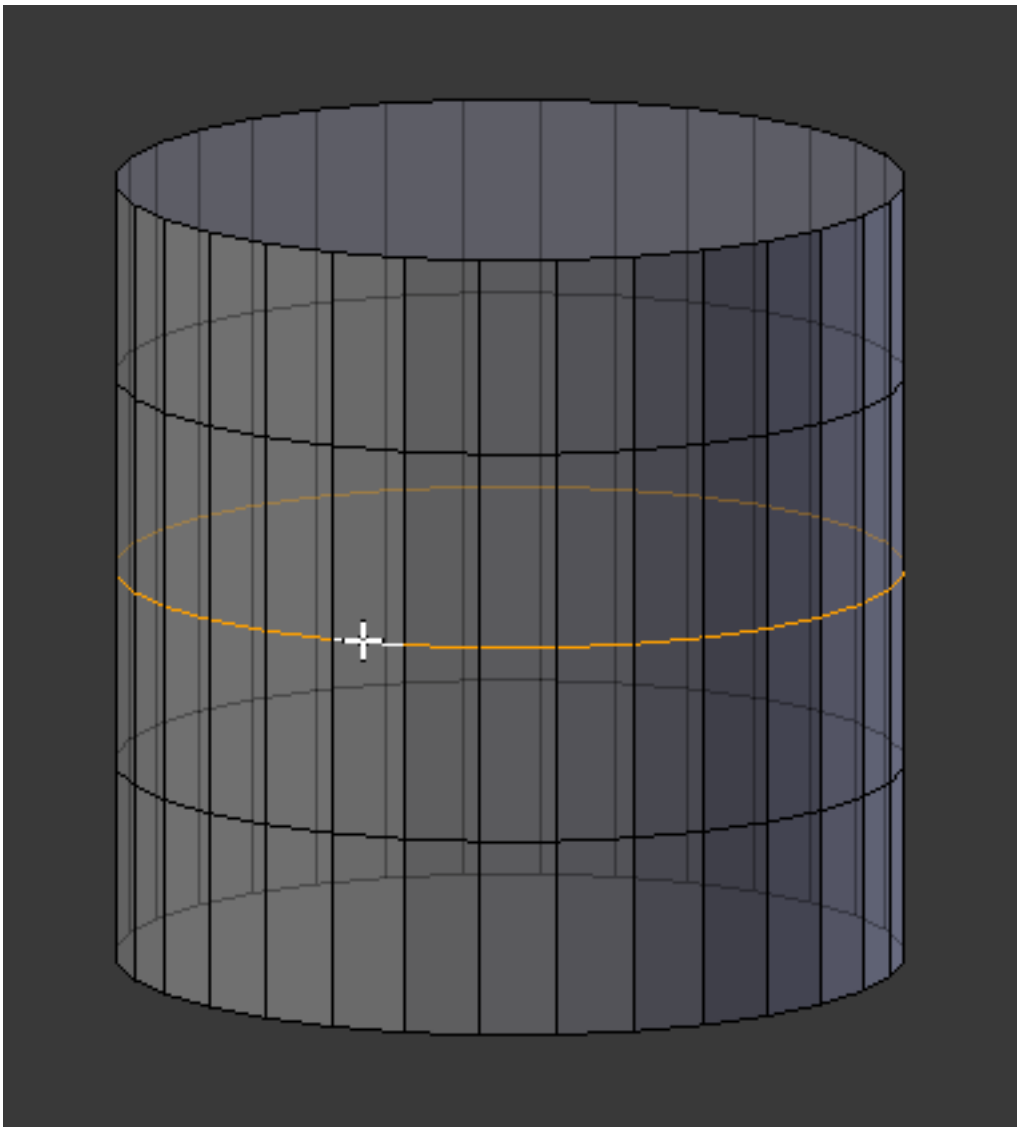


Fig. 2.392: An edge loop.

Edge Rings

Reference

Mode: Edit Mode (Mesh)
Menu: *Select* → *Edge Ring*
Hotkey: `Alt-Ctrl-RMB`, or `Shift-Alt-Ctrl-RMB` for modifying existing selection

Edge Rings are selected similarly. Based on the selection of an edge go to *Select* → *Edge Ring*. Or use `Alt-Ctrl-RMB` on an edge.

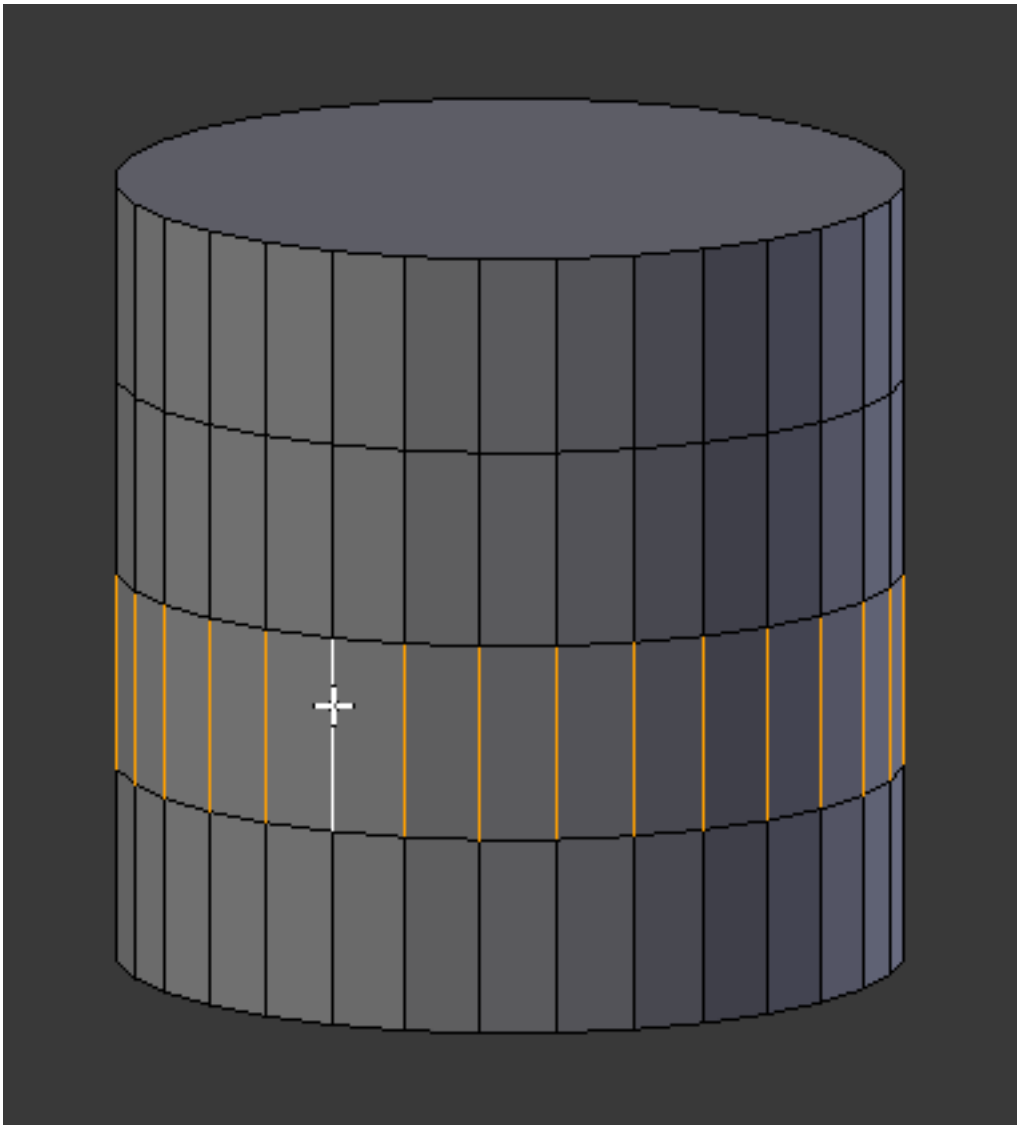


Fig. 2.393: An Edge Ring.

Note: Convert selection to whole faces

If the edge ring selection happened in Edge Select Mode, switching to Face Select Mode will erase the selection.

This is because none of those faces had all its (four) edges selected, just two of them.

Instead of selecting the missing edges manually or by using `Shift-Alt-RMB` twice, it is easier to first switch to Vertex Select Mode, which will kind of "flood" the selection. A subsequent switch to Face Select Mode will then properly select the faces.

Selecting Faces

Fig. 2.394: Activated the Face Select Mode.

To select parts of a mesh face-wise, you have to switch to Face Select Mode. Do this by clicking the button shown above, or press `Ctrl-Tab` to spawn a menu. The selection works as usual with `RMB` ; to add/remove to an existing selection, additionally press `Shift`.

Face Loops

Reference

Mode: Edit Mode (Mesh)
Hotkey: `Alt-RMB` - or `Shift-Alt-RMB` for modifying existing selection

Face Loops are pretty much the same as Edge Rings. If you want to select a Face Loop, there is no menu entry that works based on a selected face. Using *Select* → *Edge Ring* would select a "cross" with the prior selected face as the middle. If you want to avoid switching to Edge Select Mode to select a Face Loop, use the `Alt-RMB` shortcut.

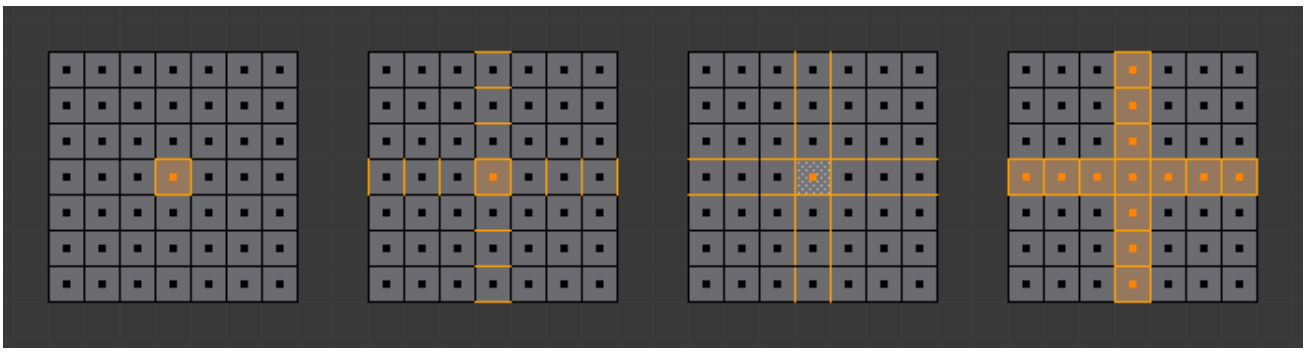


Fig. 2.395: Different Loopselect Operations on a grid in Face Select Mode.

- Just the selected face.
- Select the face, then *Select* → *Edge Ring*. See, how Blender selects edges, even if being in Face Select Mode. If these edges are desired and you want to work on them, switch in Edge Select Mode. Switching to Vertex Select Mode would flood the selection and leave you with the 4th image as result, after going back to Face Select Mode.
- Select the face, the *Select* → *Edge Loop*. As in the example above, Blender pretends to be in Edge Select Mode and takes the four edges of the selected face as base for the selection operation.
- This selection was created by `Alt-RMB` on the left edge of the center face, followed by twice `Shift-Alt-RMB` on the top edge of the center face. Two times, because the first click will remove the selected face loop (in this case, just the original selected face), while the second click will add the whole vertical running loop to the selection, creating the cross.

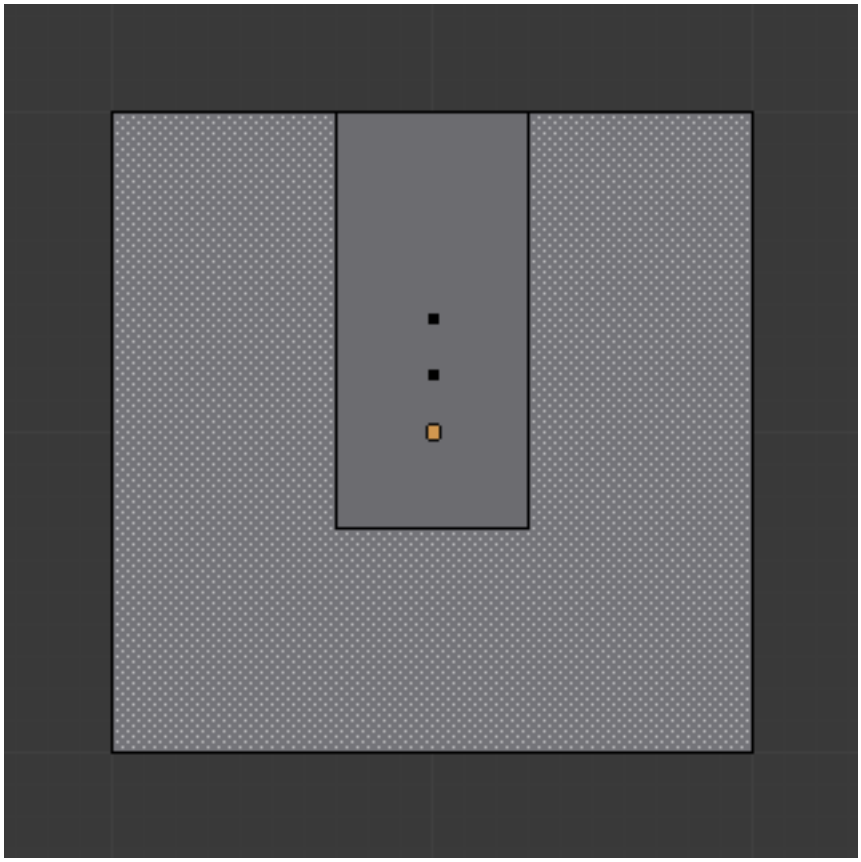


Fig. 2.396: N-gon-Face having its center dot inside another face.

N-gons in Face Select Mode

As already known, faces are marked with a little square dot in the middle of the face. With n-gons that can lead in certain cases to a confusing display. The example shows the center dot of the U-shaped ngon being inside of the oblong face inside the "U". It is not easy to say which dot belongs to which face (the orange dot in the image is the object center). Luckily, you do not need to care much, because to select a face, you do not have to click the center dot, but the face itself.

Tip: Face selection

To select a face: Click the face, not the dot!

Editing

Introduction

Blender provides a variety of tools for editing meshes. These are available through the *Mesh Tools* palette, the Mesh menu in the 3D View header, and context menus in the 3D View, as well as individual shortcut keys.

Note: All the "transform precision/snap" keys `Ctrl` and/ or `Shift` work also for all these advanced operations... However, most of them do not have *axis locking* possibilities, and some of them do not take into account *pivot point* and/or *transform orientation* either.

These transform tools are available in the *Transform* section of the *Mesh* menu in the header. Note that some of these can also be used on other editable objects, like curves, surfaces, and lattices.

Types of Tools

The mesh tools are found in various places, and available through shortcuts as well.

Transform and Deform tools:
- Translate
- Rotate
- Scale
- Mirror
- Shrink/Flatten/Along Normal
- Push/Pull
- To Sphere
- Shear
- Warp
- Edge Slide
- Vertex Slide
- Noise
- Smooth Vertex
- Rotate Edge

Merge and Remove tools:
- Delete
- Dissolve
- Merge
- Auto-Merge
- Remove Doubles
- Tris to Quads
- Unsubdivide

Add and Divide tools:
- Make Edge/Face
- Fill
- Beauty Fill
- Solidify
- Quads to Tris
- Extrude Region
- Extrude Individual
- Subdivide
- Loop Cut/Slide
- Knife tool
- Vertex connect
- Duplicate
- Spin
- Screw
- Symmetrize
- Inset
- Bevel
- Wireframe

Separate tools:
- Rip
- Rip fill
- Split
- Separate
- Edge Split

Accessing Mesh Tools

Mesh Tools Palette

When you select a mesh and `Tab` into edit mode, the *Tool Shelf* changes from *Object Tools* to *Mesh Tools*. These are only some of the mesh editing tools.

Menus

The *Mesh* menu is located in the header. Some of the menus can be accessed with shortcuts: `Ctrl-F` brings up the Face tool menu `Ctrl-E` brings up the Edge tool menu `Ctrl-V` brings up the Vertex tool menu

Basics

Translation, Rotation, Scale

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Transform* → *Grab/Move, Rotate, Scale, ...*

Hotkey: G, R, S

Once you have a selection of one or more elements, you can grab/move G, rotate R or scale S them, like many other things in Blender, as described in the *Manipulation in 3D Space* section.

To move, rotate and scale selected components, either use the *Translate*, *Rotate*, and *Scale* buttons, the *transform manipulators*, or the shortcuts:

G, R, and S respectively. After moving a selection, the options in the Tool Shelf allow you to fine-tune your changes, limit the effect to certain axes, turn proportional editing on and off, etc.

Of course, when you move an element of a given type (e.g. an edge), you also modify the implicitly related elements of other kinds (e.g. vertices and faces).

You also have in *Edit Mode* an extra option when using these basic manipulations: the *proportional editing*.

Adding Geometry

In Blender, for modeling, you have several ways of adding mesh elements. Some of them are basic objects that adds a starting block of data (called data-block in Blender) when adding their basic geometry to the scene. We have ten available mesh Objects, and those starting meshes are also called mesh primitives. In Blender, we have a set of basic primitives so you can add a starting mesh to modify and model to suit your specific needs. Also, you have specific tools to add, duplicate, move and delete elements, which will be explained in other pages of the modeling section present in this manual.

This page explains how to add basic geometry creating objects from primitives and how to add more elements to your primitives, including the addition of other primitives and basic elements when you are modeling.

To enter Edit you can select Edit from the modes menu as explained in the Interface overview, or use Tab with a mesh object selected.

To select and add one of the primitives to work with press Shift-A in Edit mode. Blender automatically detects the appropriate context for the object type you are editing, and will show a list of compatible, combining elements. If you are editing Mesh types, Blender will show a list of primitive meshes to add to your object. Other contexts are also automatically detected for the correct element additions. (See Fig. *Blender's mesh primitives.*, you can add primitives to already existing objects, in Edit Mode)

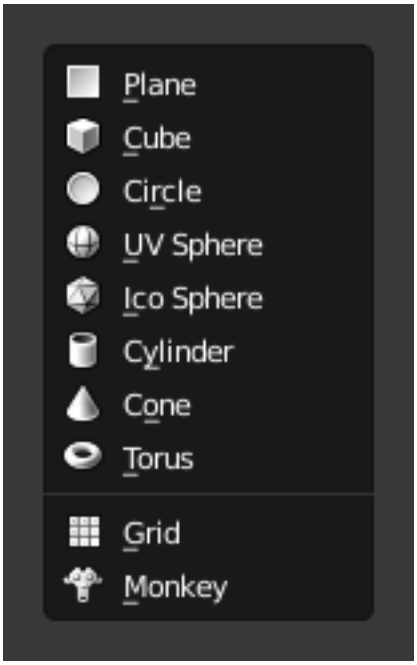


Fig. 2.397: Blender's mesh primitives.

A menu opens from which you can select the primitive you wish to add to the object.

There are many cases when it is useful to directly add a mesh to an object. Maybe you want to model a teapot. It would be useful to model the cup and the handle as separate meshes and only combine them when you are done.

Adding elementary parts to meshes

As explained before in *Mesh Structures*, meshes are objects formed from basic elements such as vertices, edges and faces.

The most elementary part of a mesh is the vertex, a point in 3D space; the line between two or more interconnected vertices is called an edge, and three or more edges can be connected to form a face. The geometry of the faces performing the model is called topology.

Creating vertices

The most basic element, a vertex, can be added with a left button mouse click while pressing `Ctrl` when no other vertices are selected, or `Ctrl-LMB`.

To create interconnected vertices, you can add a vertex and continuously make subsequent `Ctrl-LMB` operations with the last one vertex selected. This will link the last selected vertex with the vertex created at the mouse position with an edge (See Fig. *Adding vertices one by one.*), and will continuously connect them creating vertices if you continue repeating this operation. (see Fig. 3 Creating simple connected vertices with `Ctrl-LMB`).

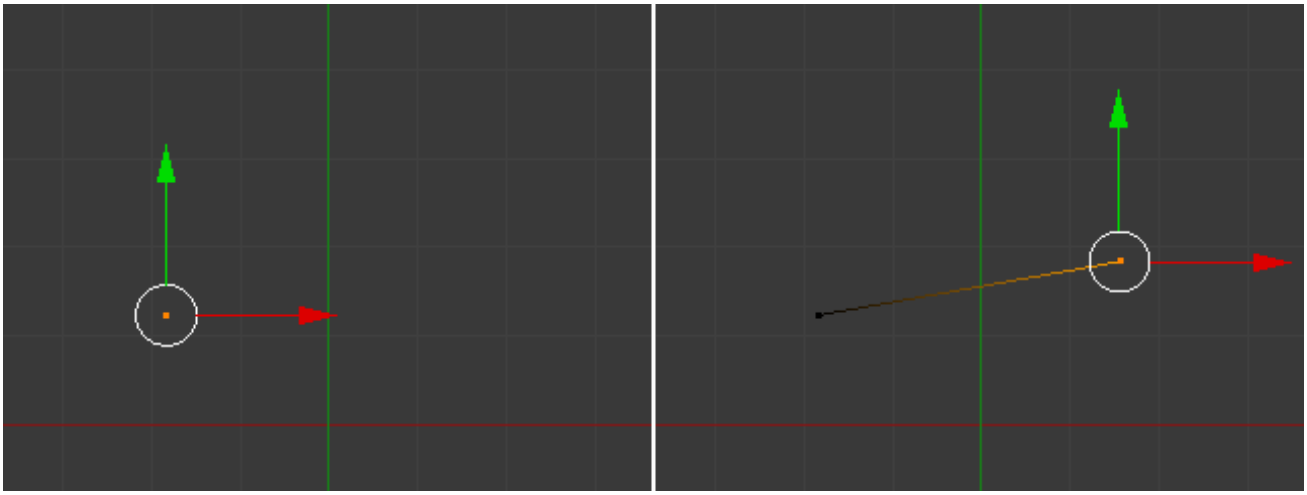


Fig. 2.398: Adding vertices one by one.

Creating Edges

In addition to automatically creating edges from vertices, if you have two vertices selected, you can connect them with and edge using the shortcut `F` (Fill). If you have more than two vertices selected, this will automatically create face(s).

Creating Faces

Creating Faces with the Mouse

If you have two vertices selected and already connected with an edge, left-click while pressing `Ctrl-LMB` will create a planar face, also known as a quad. Blender will follow your mouse cursor and will use the planar view from your viewport to create those quads.

For `Ctrl-LMB`, Blender will automatically rotate the last selected Edge (the source) for the subsequent operations if you have at least one face created, dividing the angles created between the newly-created edge and the last two edges, performing a smooth angle between them. Blender will calculate this angle using the last positive and negative position of the last X and Y coordinates and the last connected unselected edge. If this angle exceeds a negative limit (following a quadrant rule) between the recently created edge and the last two, Blender will wrap the faces. But if you do not want Blender rotating and smoothing edges automatically when extruding from `Ctrl-LMB`, you can also inhibit Blender from rotating sources using the shortcut `Ctrl-Shift-LMB`. In this case, Blender will not rotate the source dividing the angle between those edges when creating a face.

For both cases, Blender will inform the user about the source rotation during the creation process. If you look at the Bottom of the Mesh Tools Panel, if you press `Ctrl-LMB`, you will see that the Rotate Source is automatically checked and if `Ctrl-Shift-LMB` is used, it will be automatically unchecked. Examples:

- Creating Faces with shortcut `Ctrl-LMB`, (see Fig. - Faces created with source automatically rotated)

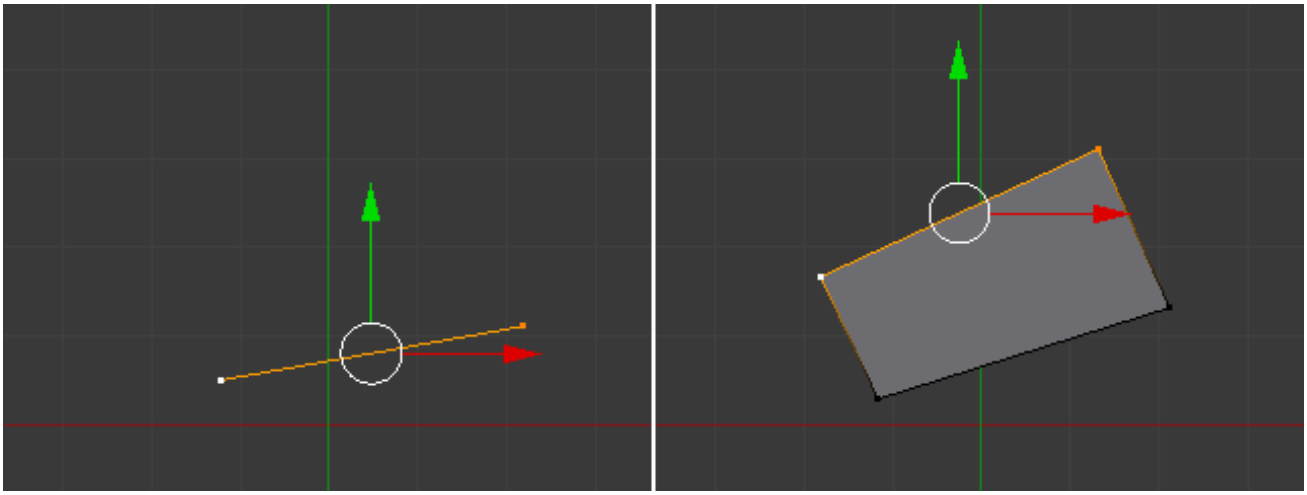


Fig. 2.399: Quad from an Edge with source automatically rotated.

- Creating Faces with shortcut `Ctrl-Shift-LMB`, (see Fig. Faces created with no source rotation)

If you have three or more vertices selected, and left click with mouse while pressing `Ctrl-LMB`, you will also create planar faces, but along the vertices selected, following the direction of the cursor. This operation is similar to an extrude operation, which is explained in the *Extrude* page.

Tip: When adding Objects with `Ctrl-LMB`, The extrusions of the selected elements, being vertices, edges and faces with the `Ctrl-LMB`, is viewport dependent. This means, once you change your viewport, for example, from top to left, bottom or right, the extrusion direction will also follow your viewport and align your extrusions with your planar view.

Filling Faces

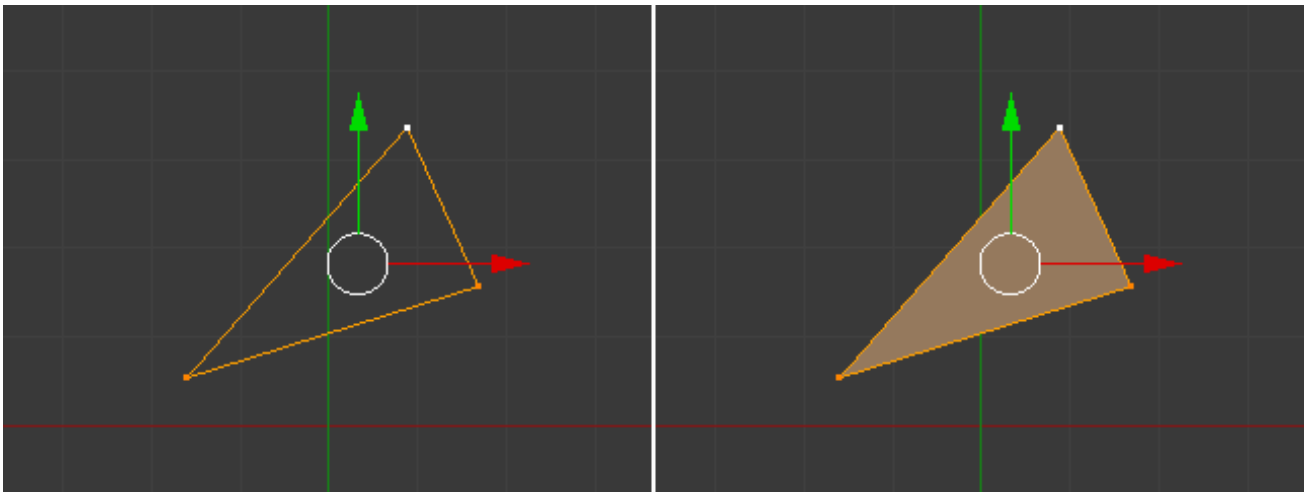


Fig. 2.400: Filling a triangle.

You can also create faces with at least three vertices selected, using `F` to fill them with edges and faces, or only fill edges with faces if they are already connected (Fill) (See Fig. *Filling a triangle.*). For four or more vertices, it is mandatory that you have coplanar vertices. four coplanar vertices will create a quad when filled, and more than four coplanar vertices will create a Ngon face.

Note: Note that you can only modify the mesh of the object you are editing. To modify other objects you need to leave,

select them and re-enter Edit Mode.

Hint: When you are modeling, that, in order to facilitate the modeling, the best solution is to imagine what primitive type suits better for your model. If you will model a cuboid, the best solution is to start with a primitive cube, and so on.

Deleting and Merging

These tools can be used to remove components.

Delete

Delete `X`, `Delete` Deletes selected vertices, edges, or faces. This operation can also be limited to:

Vertices Delete all vertices in current selection, removing any faces or edges they are connected to.

Edges Deletes any edges in the current selection. Removes any faces that the edge shares with it.

Faces Removes any faces in current selection.

Only Edges & Faces Limits the operation to only selected edges and adjacent faces.

Only faces Removes faces, but edges within face selection are retained.

Edge Collapse Collapses edges into single vertices. This can be used to remove a loop of faces.

Edge Loop Deletes an edge loop. If the current selection is not an edge loop, this operation does nothing.

Dissolve

Dissolve operations are also accessed from the delete menu. Instead of removing the geometry, which may leave holes that you have to fill in again, dissolve will remove the geometry and fill in the surrounding geometry.

Dissolve Removes selected geometry, but keeps surface closed, effectively turning the selection into a single n-gon. Dissolve works slightly different based on if you have edges, faces or vertices selected. You can add detail where you need it, or quickly remove it where you do not.

Limited Dissolve Limited Dissolve reduces detail on planar faces and linear edges with an adjustable angle threshold.

Face Split When dissolving vertices into surrounding faces, you can often end up with very large, uneven ngons. The face split option limits dissolve to only use the corners of the faces connected to the vertex.

Convert Triangles to Quads

Tris to Quads `Alt-J` This takes adjacent tris and removes the shared edge to create a quad. This tool can be performed on a selection of multiple triangles.

This same action can be done on a selection of just two tris, by selecting them and using the shortcut `F`, to create a face.

Unsubdivide

Reference

Mode: Edit Mode
Menu: *Mesh* → *Edges* → *Unsubdivide*

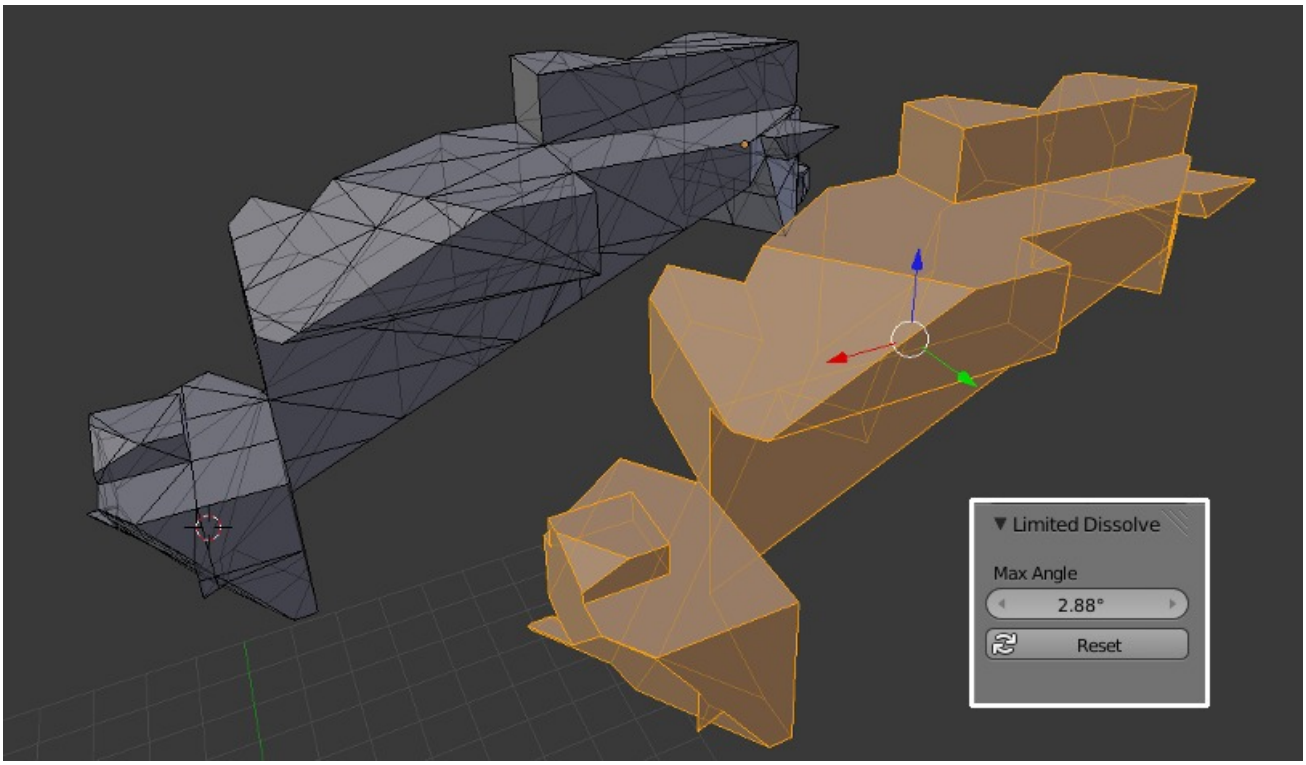


Fig. 2.401: Example showing the how Limited Dissolve can be used.

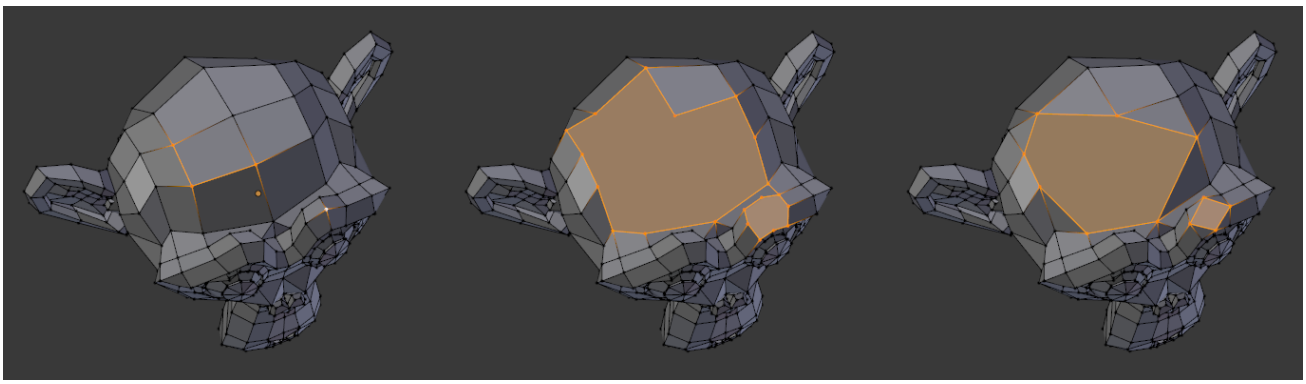


Fig. 2.402: Dissolve Face Split option.
Left: the input, middle: regular dissolve, right: Face Split enabled.

Unsubdivide functions as the reverse of subdivide by attempting to remove edges that were the result of a subdivide operation. If additional editing has been done after the subdivide operation, unexpected results may occur.

Iterations How many subdivisions to remove.

Merging

Merging Vertices

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Merge...*, *Specials* → *Merge* or *Vertex Specials* → *Merge*
Hotkey: `Alt-M`

This tool allows you to merge all selected vertices into an unique one, deleting all others. You can choose the location of the surviving vertex in the menu this tool pops up before executing:

At First Only available in *Vertex* select mode, it will place the remaining vertex at the location of the first one selected.

At Last Only available in *Vertex* select mode, it will place the remaining vertex at the location of the last one selected (the active one).

At Center Available in all select modes, it will place the remaining vertex at the center of the selection.

At Cursor Available in all select modes, it will place the remaining vertex at the 3D Cursor.

Collapse This is a special option, as it might let "live" more than one vertex. In fact, you will have as many remaining vertices as you had "islands" of selection (i.e. groups of linked selected vertices). The remaining vertices will be positioned at the center of their respective "islands". It is also available *via* the *Mesh* → *Edges* → *Collapse* menu option...

Merging vertices of course also deletes some edges and faces. But Blender will do everything it can to preserve edges and faces only partly involved in the reunion.

AutoMerge Editing

Reference

Mode: Edit Mode
Menu: *Mesh* → *AutoMerge Editing*

The *Mesh* menu as a related toggle option: *AutoMerge Editing*. When enabled, as soon as a vertex moves closer to another one than the *Limit* setting (*Mesh Tools* panel, see below), they are automatically merged.

Remove Doubles

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Vertices* → *Remove Doubles*, *Specials* → *Remove Doubles* or *Vertex Specials* → *Remove Doubles*
Hotkey: `W`, *Remove Doubles*

Remove Doubles is a useful tool to simplify a mesh by merging vertices that are closer than a specified distance to each other. An alternate way to simplify a mesh is to use the *Decimate modifier*.

Merge Distance Sets the distance threshold for merging vertices, in Blender units.

Unselected Allows vertices in a selection to be merged with unselected vertices. When disabled, selected vertices will only be merged with other selected ones.

Make Edge/Face

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Make Face/Edge*
Hotkey: `F`

This is a context-sensitive tool which creates geometry by filling in the selection. When only two vertices are selected it will create an edge, otherwise it will create faces.

The typical use case is to select vertices and press `F`, however, Blender also supports creating faces from different selections to help quickly build up geometry.

Methods

The following methods are used automatically depending on the context.

Isolated vertices

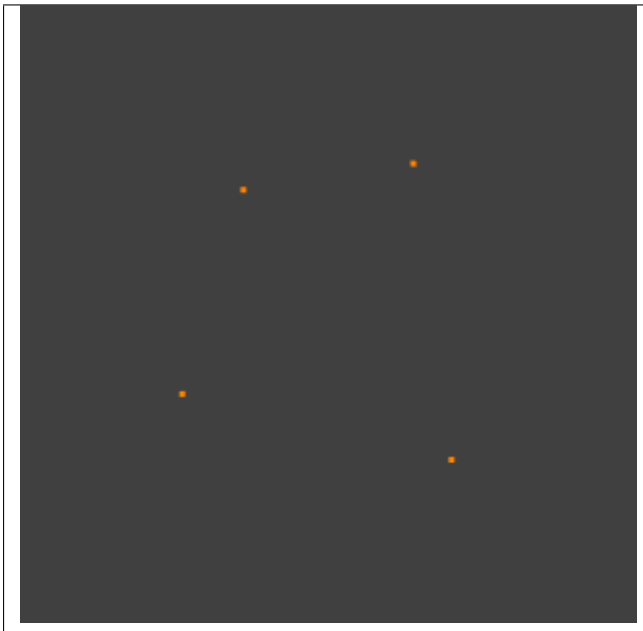


Fig. 2.403: Before.

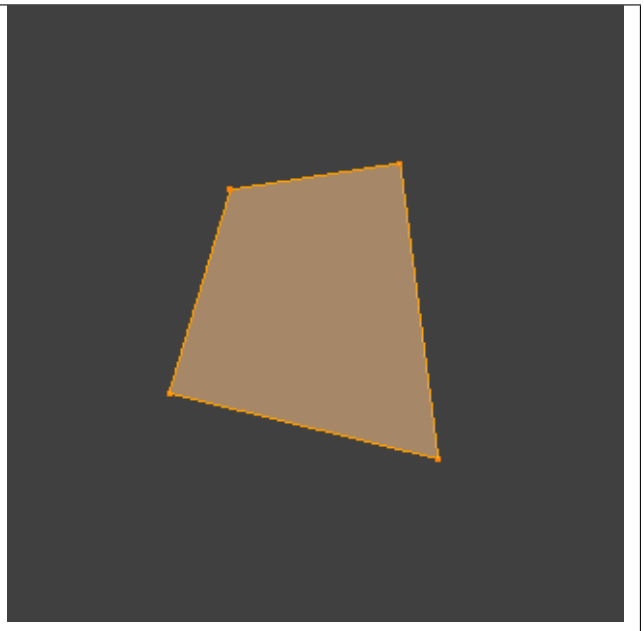


Fig. 2.404: After.

Isolated edges

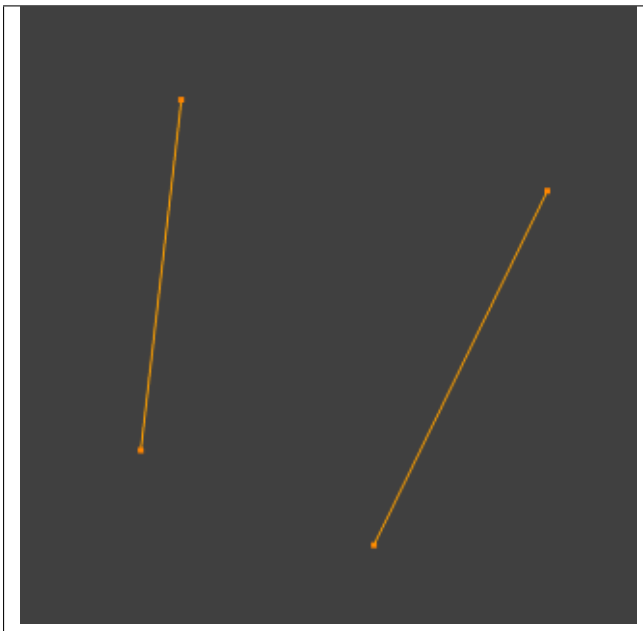


Fig. 2.405: Before.

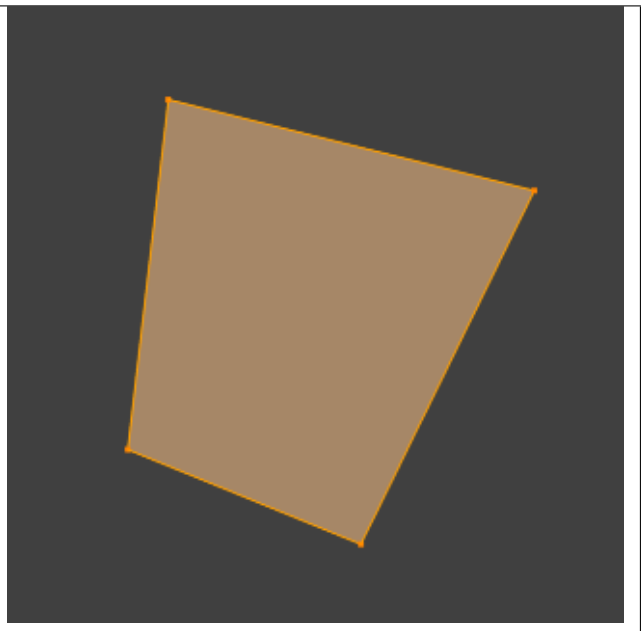


Fig. 2.406: After.

N-gon from edges

When there are many edges Blender will make an ngon, note that this does not support holes, to support holes you need to use the *Fill* Faces tool.

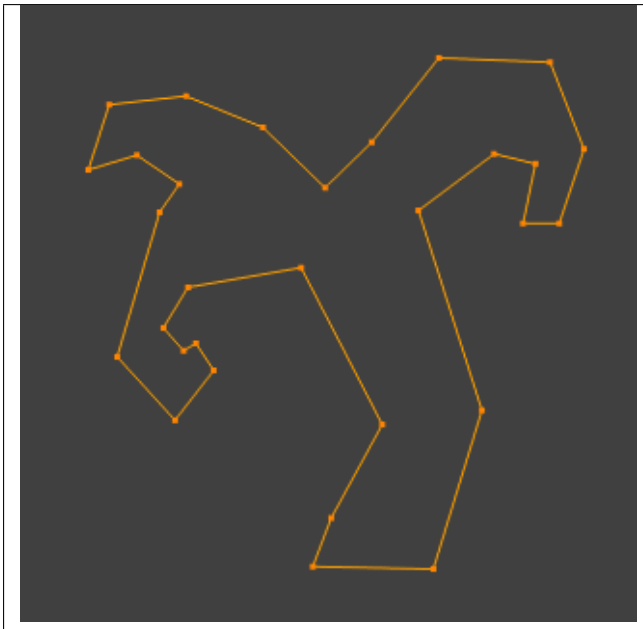


Fig. 2.407: Before.

Fig. 2.408: After.

Mixed vertices/edges

Existing edges are used to make the face as well as an extra vertex.

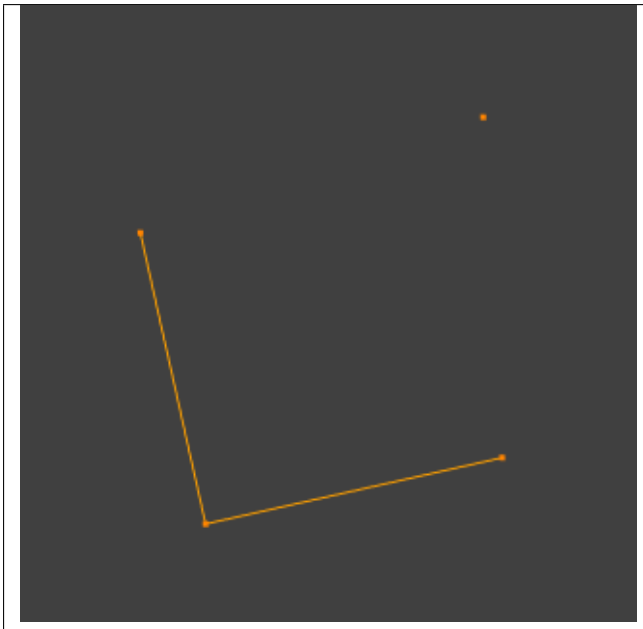


Fig. 2.409: Before.

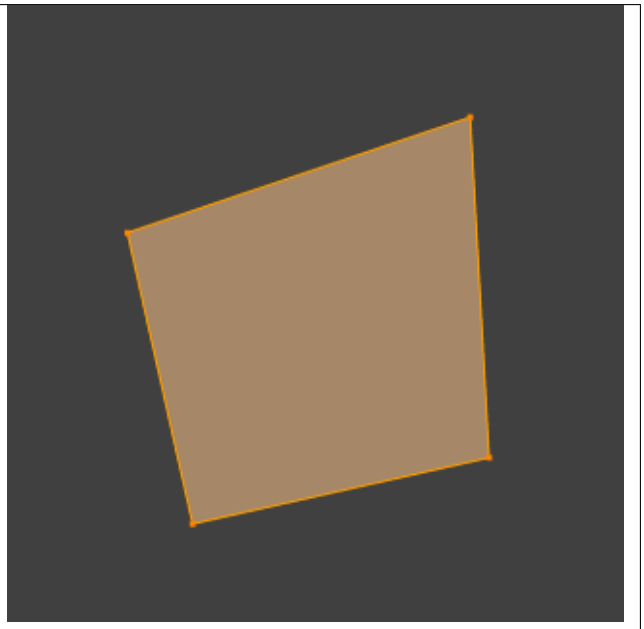


Fig. 2.410: After.

Edge-Net

Sometimes you may have many connected edges without interior faces.

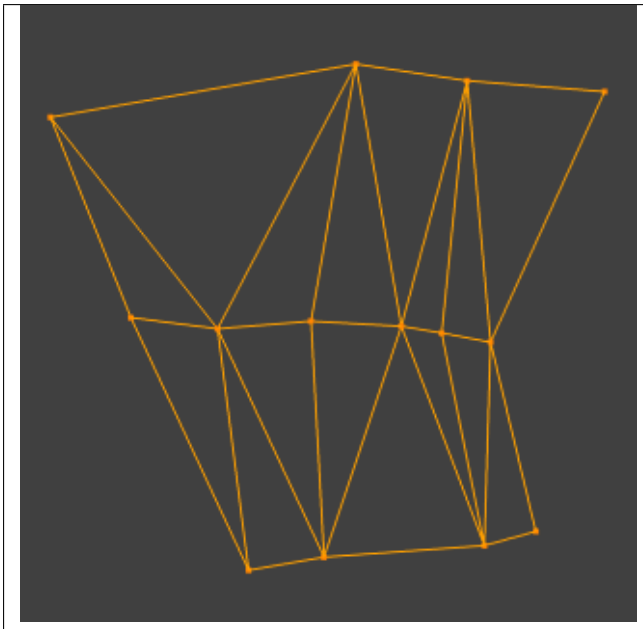


Fig. 2.411: Before.

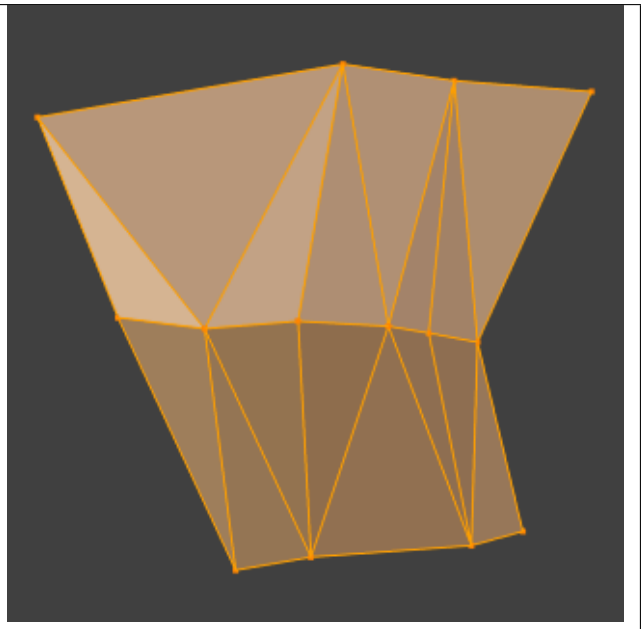


Fig. 2.412: After.

Point Cloud

When there are many isolated vertices, Blender will calculate the edges for an n-gon.

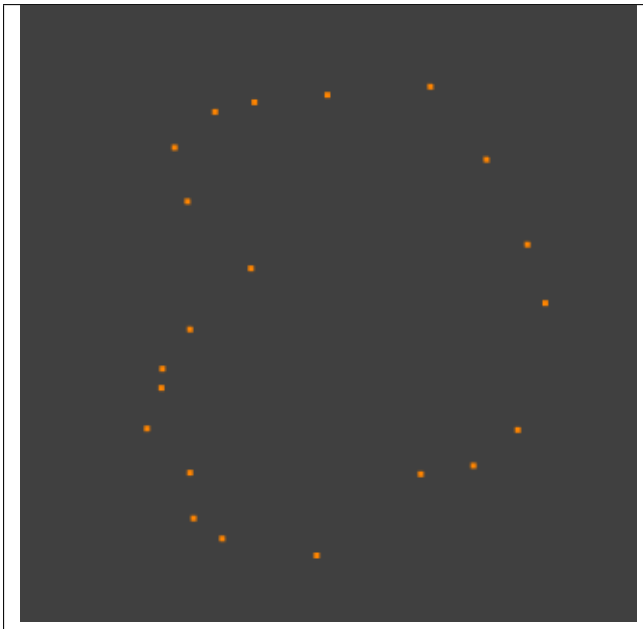


Fig. 2.413: Before.

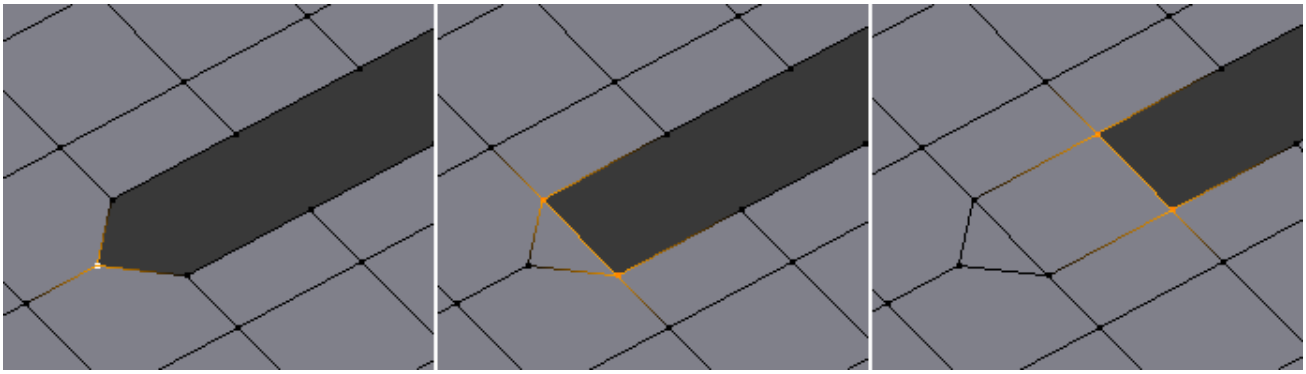
Fig. 2.414: After.

Single Vertex Selection

With a single vertex selected on a boundary, the face will be created along the boundary, this saves manually selecting the other two vertices. Notice this tool can run multiple times to continue creating faces.

See also:

For other ways to create faces see:



- *Fill*
- *Grid Fill*
- *Bridge Edge Loops*

Mirror Editing

Snap to Symmetry

Reference

Mode: Edit Mode
Menu: *Mesh* → *Snap to Symmetry*

The *Snap to Symmetry* tool works on meshes which are mostly symmetrical but have vertices which have been moved enough that Blender does not detect then as mirrored (when x-mirror option is enable for example).

This can be caused by accident when editing without x-mirror enabled. Sometimes models imported from other applications are asymmetrical enough that mirror fails too.

Direction Specify the axis and direction to snap. Can be any of the three axes, and either positive to negative, or negative to positive.

Threshold Specify the search radius to use when finding matching vertices.

Factor Support for blending mirrored locations from one side to the other (0.5 is an equal mix of both).

Center Snap vertices in the center axis to zero.

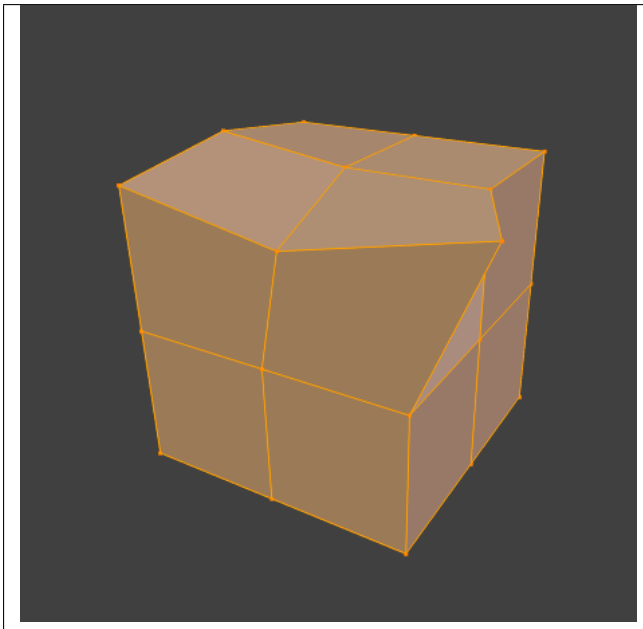


Fig. 2.415: Before Snap to Symmetry.

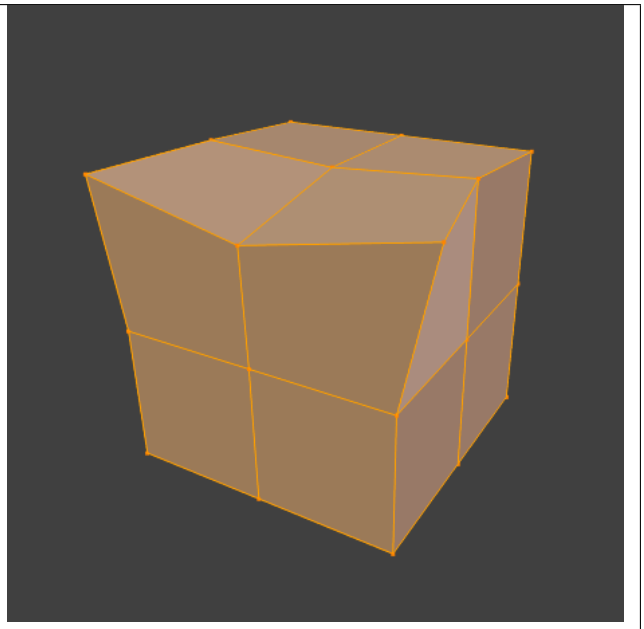


Fig. 2.416: After Snap to Symmetry.

Symmetrize Mesh

Reference

Mode: Edit Mode
Menu: *Mesh* → *Symmetrize*

The *Symmetrize* tool is a quick way to make a mesh symmetrical. *Symmetrize* works by cutting the mesh at the pivot point of the object, and mirroring over the geometry in the specified axis, and merges the two halves together (if they are connected).

Direction Specify the axis and direction of the effect. Can be any of the three axes, and either positive to negative, or negative to positive.

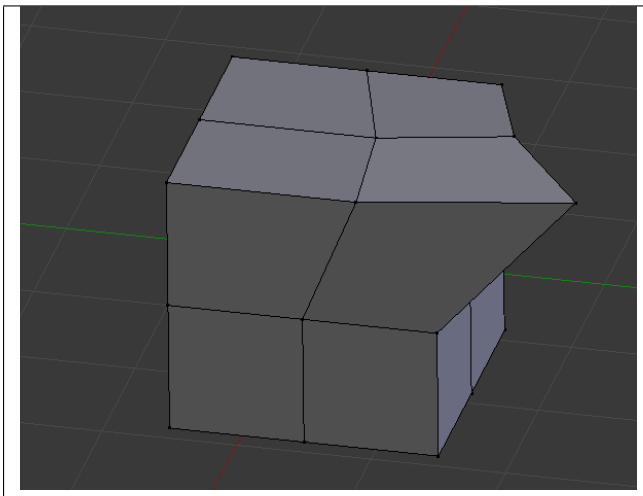


Fig. 2.417: Mesh before Symmetrize.

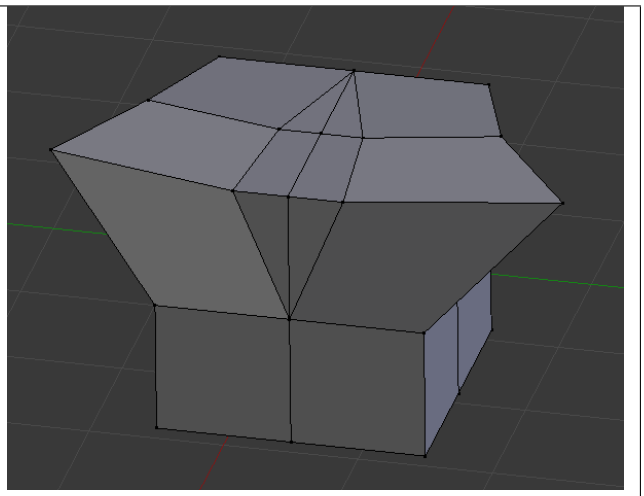


Fig. 2.418: Mesh after Symmetrize.

See also:

See *Mirror* for information on mirroring, which allows you to flip geometry across an axis.

Mesh Options

X-Mirror

Reference

Mode: Edit Mode
Panel: *Mesh Options* → *X-mirror*

The *X-mirror* option of the *Mesh Options* panel allows you edit both "sides" of your mesh in a single action. When you transform an element (vertex, edge or face), if there is its exact X-mirrored counterpart (in local space), it will be transformed accordingly, through a symmetry along the local X axis.

Note: The conditions for *X-Mirror* to work are quite strict, which can make it difficult to use. To have an exact mirrored version of a (half) mesh, its easier and simpler to use the *Mirror modifier*

Topology Mirror

Reference

Mode: Edit Mode
Panel: *Mesh Options* → *Topology Mirror*

Note: For *Topology Mirror* to work the *X Mirror* option must be enabled.

When using the *X Mirror* option to work on mirrored Mesh Geometry the vertices that are mirrored must be perfectly placed. If they are not exactly positioned in their mirror locations then *X Mirror* will not treat those vertices as mirrored. This can be annoying because often the out of position vertices are only very slightly out of position.

Topology Mirror tries to solve this problem by determining which vertices are mirrored vertices not only by using their positions but also by looking at how those vertices are related to others in the Mesh Geometry. It looks at the overall Mesh Geometry topology to determine if particular vertices will be treated as mirrored. The effect of this is that mirrored vertices can be non-symetrical and yet still be treated as mirrored when *X Mirror* and *Topology Mirror* are both active.

Note: The *Topology Mirror* functionality will work more reliably on mesh geometry which is more detailed. If you use very simple geometry for example, a *Cube* or *UV Sphere* the *Topology Mirror* option will often not work.

Example

For an example of how to use *Topology Mirror* open up a new Blender scene, then delete the default cube and add a Monkey object to the 3D View.

1. Press `Tab` to put the Monkey object into *Edit Mode*.
2. With the *X Mirror* option disabled move one of the Monkey object's vertices slightly.
3. Then Turn *X Mirror* option on again but leave *Topology Mirror* disabled
4. If you now move that vertice again *X Mirror* will not work and the mirrored vertices will not be altered.
5. If you then enable *Topology Mirror* and move the same vertices again, then *X Mirror* should still mirror the other vertice, even though they are not perfectly positioned.

Vertex Tools

This page covers many of the tools in the *Mesh* → *Vertices* menu. These are tools that work primarily on vertex selections, however, some also work with edge or face selections.

Merging

Merging Vertices

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Merge...*, *Specials* → *Merge* or *Vertex Specials* → *Merge*
Hotkey: `Alt-M`

This tool allows you to merge all selected vertices to a unique one, deleting all others. You can choose the location of the surviving vertex in the menu this tool pops up before executing:

At First Only available in *Vertex* select mode, it will place the remaining vertex at the location of the first one selected.

At Last Only available in *Vertex* select mode, it will place the remaining vertex at the location of the last one selected (the active one).

At Center Available in all select modes, it will place the remaining vertex at the center of the selection.

At Cursor Available in all select modes, it will place the remaining vertex at the 3D Cursor.

Collapse This is a special option, as it might let "live" more than one vertex. In fact, you will have as much remaining vertices as you had "islands" of selection (i.e. groups of linked selected vertices). The remaining vertices will be positioned at the center of their respective "islands". It is also available *via* the *Mesh* → *Edges* → *Collapse* menu option...

Merging vertices of course also deletes some edges and faces. But Blender will do everything it can to preserve edges and faces only partly involved in the reunion.

AutoMerge Editing

Reference

Mode: Edit Mode
Menu: *Mesh* → *AutoMerge Editing*

The *Mesh* menu as a related toggle option: *AutoMerge Editing*. When enabled, as soon as a vertex moves closer to another one than the *Limit* setting (*Mesh Tools* panel, see below), they are automatically merged.

Remove Doubles

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Vertices* → *Remove Doubles*, *Specials* → *Remove Doubles* or *Vertex Specials* → *Remove Doubles*
Hotkey: W → *Remove Doubles* or *Mesh* → *Vertices* → *Remove doubles*

Remove Doubles is a useful tool to simplify a mesh by merging vertices that are closer than a specified distance to each other. An alternate way to simplify a mesh is to use the *Decimate modifier*.

Merge Distance Sets the distance threshold for merging vertices, in Blender units.

Unselected Allows vertices in selection to be merged with unselected vertices. When disabled, selected vertices will only be merged with other selected ones.

Separating

Rip

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Rip*
Hotkey: V

Rip creates a "hole" into a mesh by making a copy of selected vertices and edges, still linked to the neighbor non-selected vertices, so that the new edges are borders of the faces on one side, and the old ones, borders of the faces of the other side of the rip.

Examples

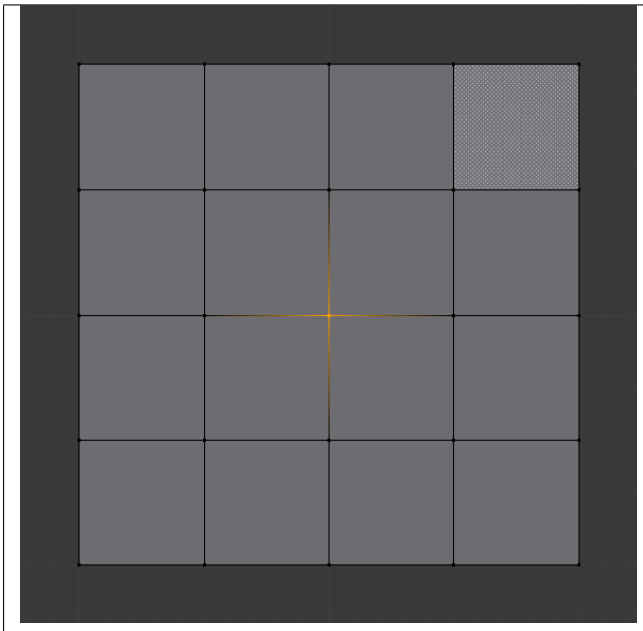


Fig. 2.419: Selected vertex.

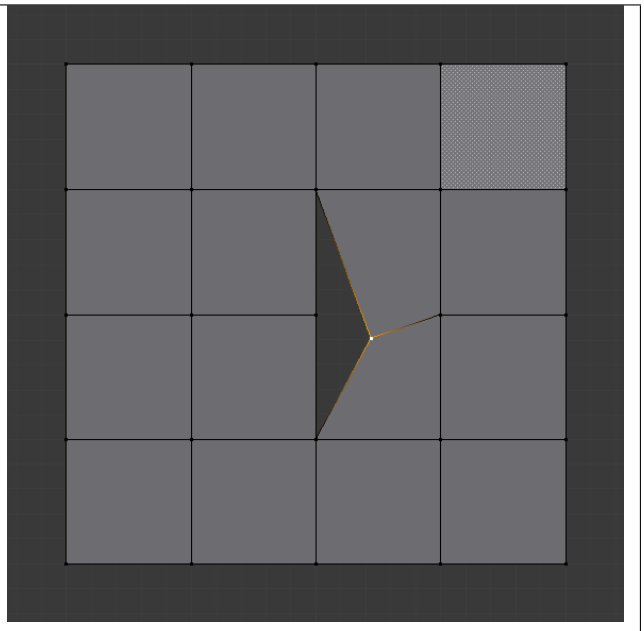


Fig. 2.420: Hole created after using rip on vertex.

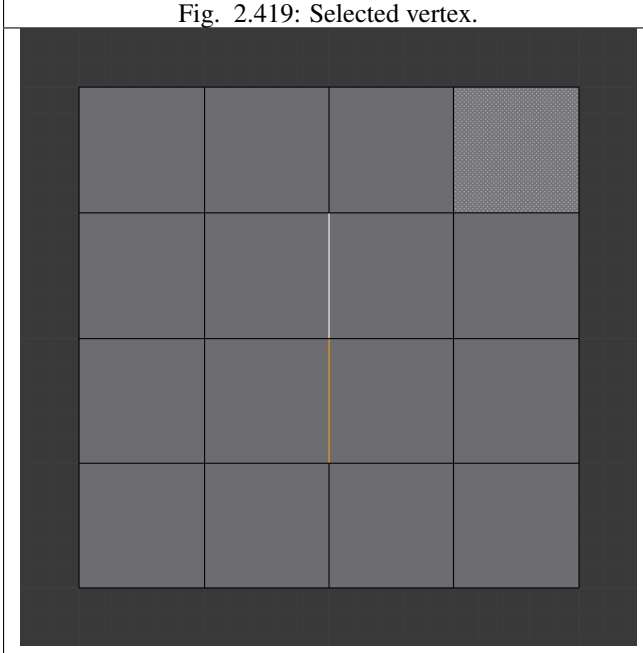


Fig. 2.421: Edges selected.

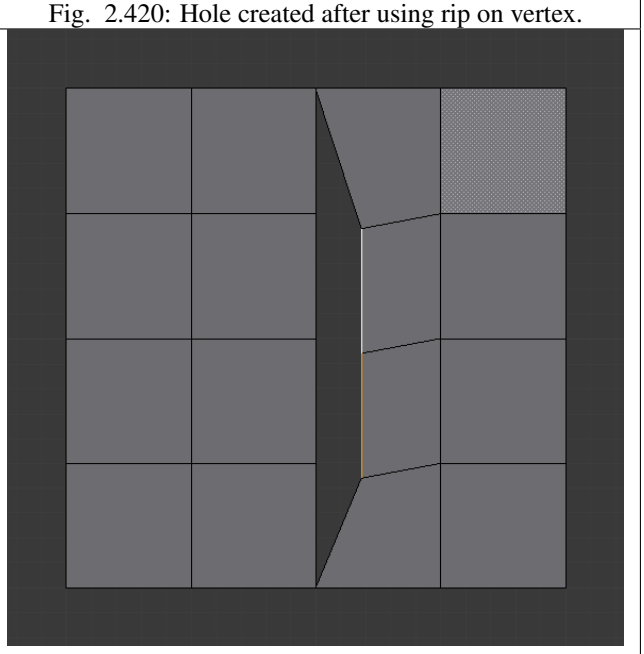
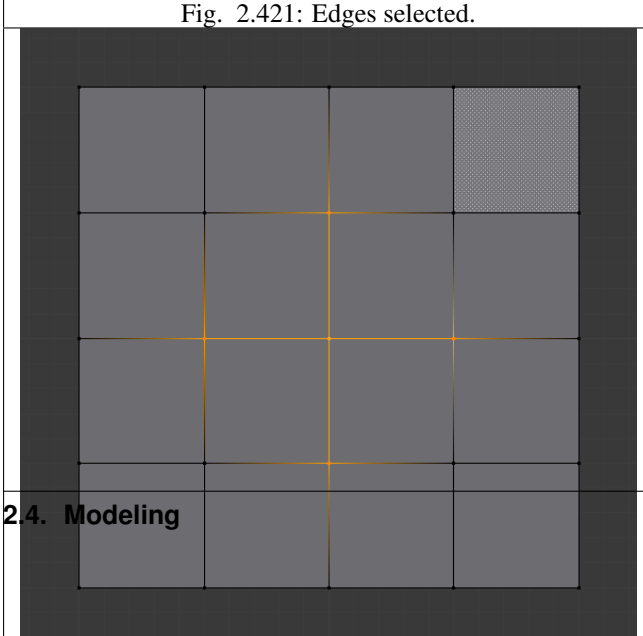
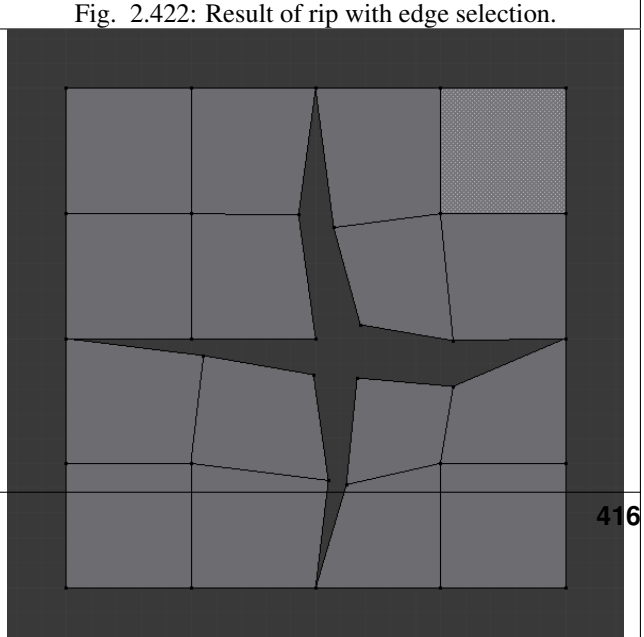


Fig. 2.422: Result of rip with edge selection.





Limitations

Rip will only work when edges and/or vertices are selected. Using the tool when a face is selected (explicitly or implicitly), will return an error message *"Cannot perform ripping with faces selected this way"* If your selection includes some edges or vertices that are not "between" two faces *manifold*, it will also fail with message *"No proper selection or faces include"*.

Rip Fill

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Rip Fill*
Hotkey: `Alt-V`

Rip fill works the same as the Rip tool above, but instead of leaving a hole, it fills in the gap with geometry.

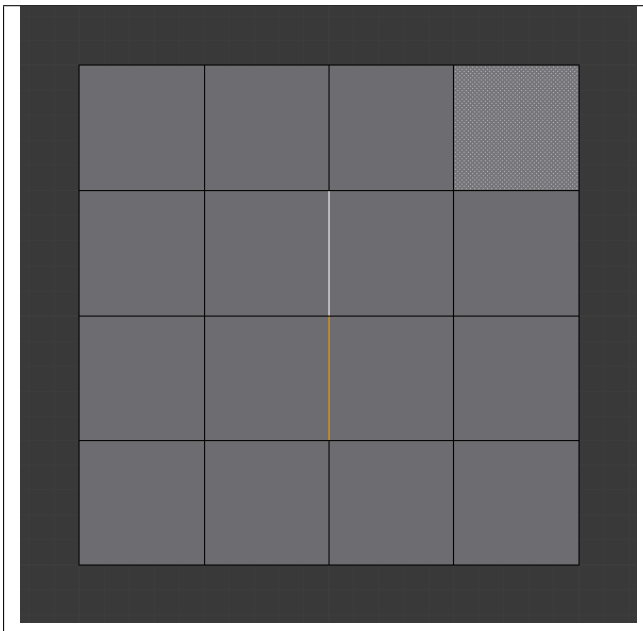


Fig. 2.425: Edges selected.

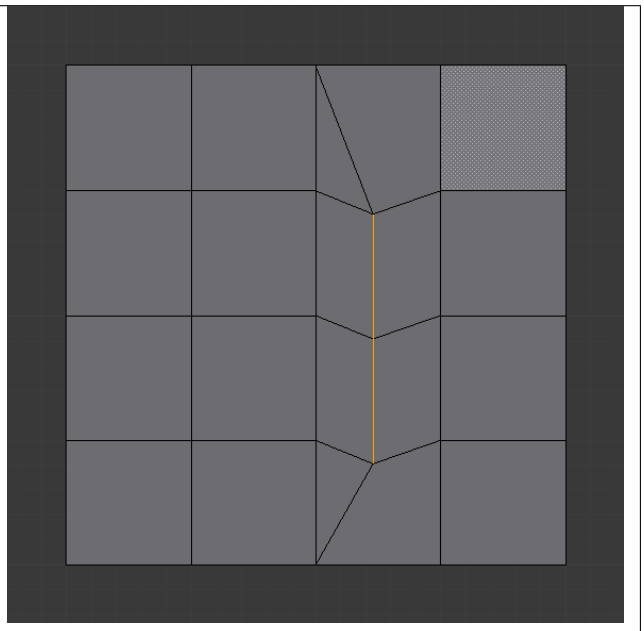


Fig. 2.426: Result of rip fill.

Split

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Split*
Hotkey: `Y`

A quite specific tool, it makes a sort of copy of the selection, removing the original data *if it is not used by any non-selected element*. This means that if you split an edge from a mesh, the original edge will still remain unless it is not linked to anything else. If you split a face, the original face itself will be deleted, but its edges and vertices remain unchanged. And so on.

Note that the "copy" is left exactly at the same position as the original, so you must move it G to see it clearly...

Separate

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Separate*
Hotkey: P

This will separate the selection in another mesh object, as described *here*.

Connect Vertex Path

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Connect Vertex Path*
Hotkey: J

This tool connects vertices in the order they are selected, splitting the faces between them.

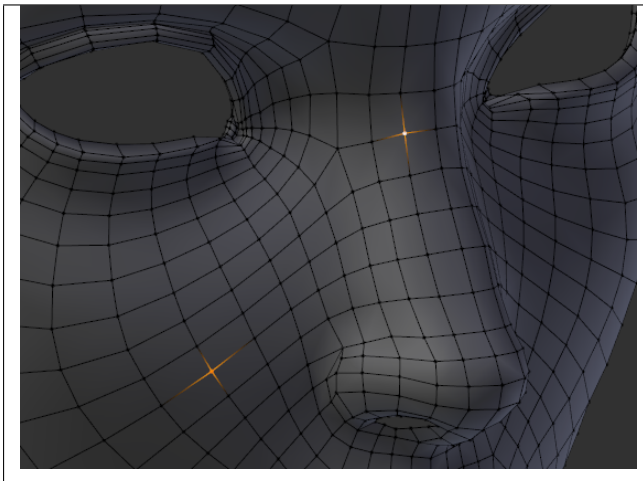


Fig. 2.427: Two disconnected vertices.

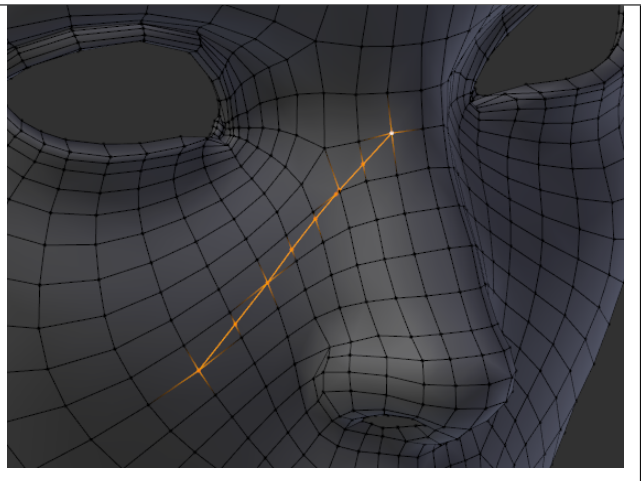


Fig. 2.428: Result of connecting.

Running a second time will connect the first/last endpoints.

Vertices not connected to any faces will create edges, so this can be used as a way to quickly connect isolated vertices too.

Connect Vertices

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Connect Vertices*

This tool connects selected vertices by creating edges between them and splitting the face.

This tool can be used on many faces at once.

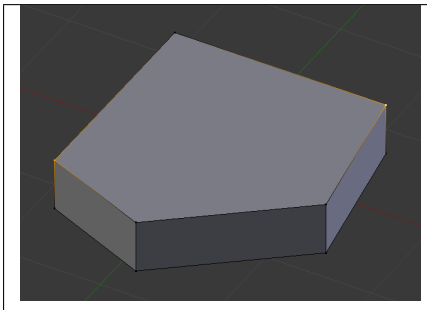


Fig. 2.429: Vertices before connecting.

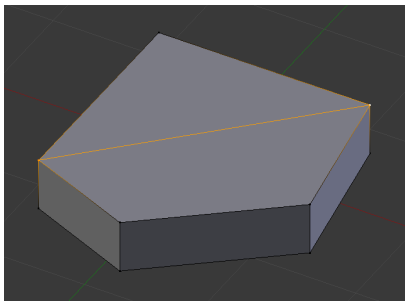


Fig. 2.430: After connecting vertices.

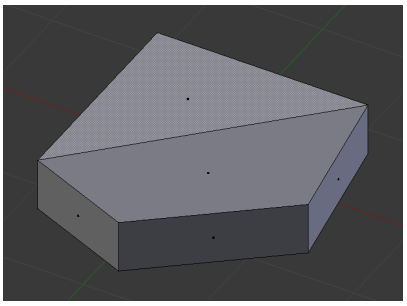


Fig. 2.431: Resulting face pair.

The main difference between this tool and *Connect Vertex Path*, is this tool ignores selection order and connects all selected vertices that share a face.

Vertex Slide

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Vertices* → *Vertex Slide*
Hotkey: `Shift-V`

Vertex Slide will transform a vertex along one of its adjacent edges. Use `Shift-V` to enter tool. Highlight the desired edge by moving the mouse, then confirm with `LMB`. Drag the cursor to specify the position along the line formed by the edge, then `LMB` again to move the vertex.

`Shift` Higher precision control.

`Ctrl` Snap to value (useful to combine with auto merge)

`LMB` confirms the tool

`RMB` or `Esc` Cancels.

`Alt` or `C` Toggle clamping the slide within the edge extents.

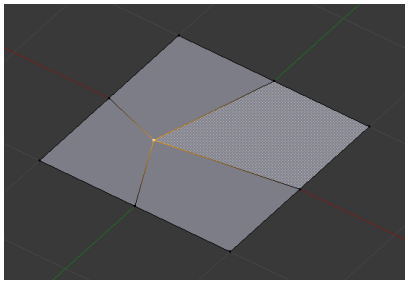


Fig. 2.432: Selected vertex.

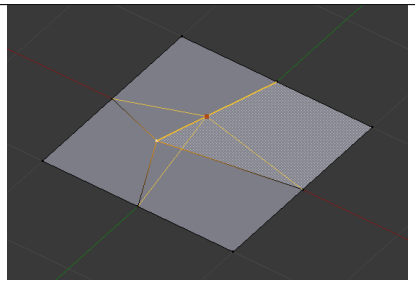


Fig. 2.433: Positioning vertex interactively.

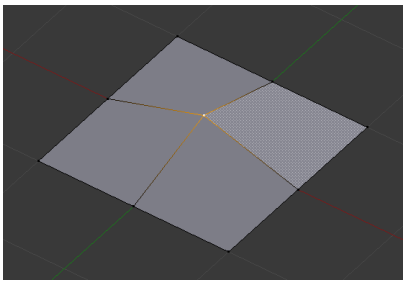


Fig. 2.434: Repositioned vertex.

Smooth

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Vertices* → *Smooth*, *Specials* → *Smooth* or *Vertex Specials* → *Smooth*
Hotkey: *Mesh* → *Vertices* → *Smooth vertex*

This will apply once the *Smooth Tool*.

Make Vertex Parent

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Make Vertex Parent*
Hotkey: `Ctrl-P`

This will parent the other selected object(s) to the vertices/edges/faces selected, as described *here*.

Add Hook

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Add Hook*
Hotkey: `Ctrl-H`

Adds a *Hook Modifier* (using either a new empty, or the current selected object) linked to the selection. Note that even if it appears in the history menu, this action cannot be undone in *Edit Mode* - probably because it involves other objects...

Blend From Shape, Propagate Shapes

Reference

Mode: Edit Mode
Menu: *(Vertex) Specials* → *Blend From Shape* and *Mesh* → *Vertices* → *Shape Propagate*

These are options regarding *shape keys*.

Shape Propagate Apply selected vertex locations to all other shape keys.

Blend From Shape Blend in shape from a shape key.

Edge Tools

Make Edge/Face

Reference

Mode: Edit Mode
Menu: *Mesh* → *Edges* → *Make Edge/Face*
Hotkey: F

It will create an edge or some faces, depending on your selection.

See also *Creating Geometry*.

Set Edge Attributes

Edges can have several different attributes that affect how certain other tools affect the mesh.

Mark Seam and Clear Seam

Reference

Mode: Edit Mode (Vertex or Edge select modes)
Menu: *Mesh* → *Edges* → *Mark Seam/Clear Seam* (or the same options in *Edge Specials* menu)

Seams are a way to create separations, "islands", in UV maps. See the *UVTexturing section* for more details. These commands set or unset this flag for selected edges.

Mark Sharp and Clear Sharp

Reference

Mode: Edit Mode (Vertex or Edge select modes)
Menu: *Mesh* → *Edges* → *Mark Seam/Clear Seam* (or the same options in *Edge Specials* menu)

The *Sharp* flag is used by the *Edge Split Modifier*, which is part of the smoothing technics. As seams, it is a property of edges, and these commands set or unset it for selected ones.

Adjust Bevel Weight

Reference

Mode: Edit Mode (Vertex or Edge select modes)
Menu: *Mesh* → *Edges* → *Edge Bevel Weight*

This edge property, a value between (0.0 to 1.0), is used by the *Bevel modifier* to control the bevel intensity of the edges. This command enters an interactive mode (a bit like transform tools), where by moving the mouse (or typing a value with the keyboard) you can set the (average) bevel weight of selected edges.

Crease Subdivision

Reference

Mode: Edit Mode (Vertex or Edge select modes)
Menu: *Mesh* → *Edges* → *Edge Crease*
Hotkey: `Shift-E`

This edge property, a value between (0.0 to 1.0), is used by the *Subdivision Surface modifier* to control the sharpness of the edges in the subdivided mesh. This command enters an interactive mode (a bit like transform tools), where by moving the mouse (or typing a value with the keyboard) you can set the (average) crease value of selected edges. To clear the crease edge property, enter a value of -1.

Edge Slide

Reference

Mode: Edit Mode (Vertex or Edge select modes)
Menu: *Mesh* → *Edges* → *Slide Edge* (or the same option in *Edge Specials* menu)

Hotkey: G, G

Slides one or more edges across adjacent faces with a few restrictions involving the selection of edges (i.e. the selection *must* define a valid loop, see below.)

Shift Higher precision control.

Ctrl Snap to value (useful to combine with auto merge).

LMB Confirms the tool.

RMB or Esc Cancels.

Even E Forces the edge loop to match the shape of the adjacent edge loop. You can flip to the opposite vertex using F. Use Alt-Wheel to change the control edge.

Flip F When Even mode is active, this flips between the two adjacent edge loops the active edge loop will match.

Alt or C Toggle clamping the slide within the edge extents.

This tool has a factor, which is displayed in the 3D View footer and in the *Tool Shelf* (after confirmation). A numerical value between (-1 to 1) can be entered for precision.

In *Proportional* mode, Wheel, or Left and Right changes the selected edge for calculating a proportion. Unlike *Percentage* mode, *Proportional*

Holding Ctrl or Shift control the precision of the sliding. Ctrl snaps movement to 10% steps per move and Shift snaps movement to 1% steps. The default is 5% steps per move.

Usage

By default, the position of vertices on the edge loop move as a percentage of the distance between their original position and the adjacent edge loop, regardless of the edges' lengths.

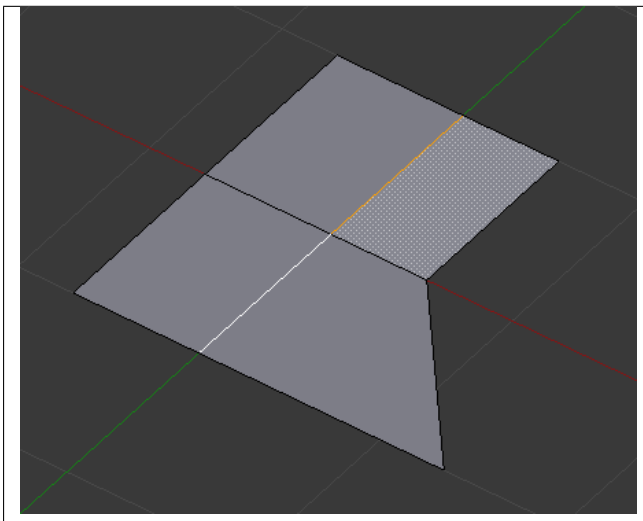


Fig. 2.435: Selected Edge Loop.

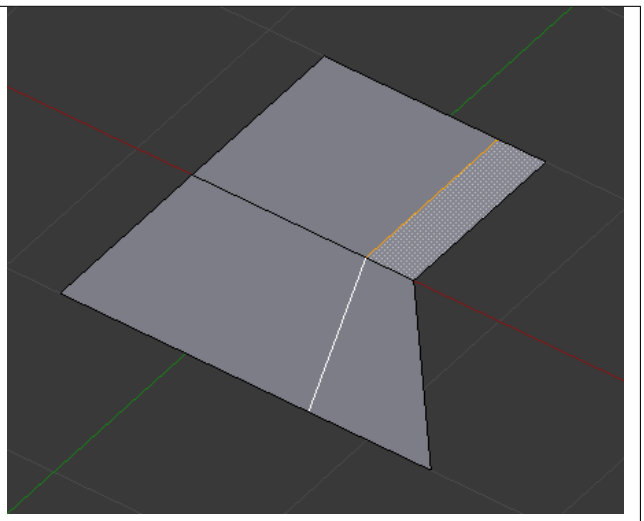


Fig. 2.436: Repositioned Edge Loop.

Even mode

Even mode keeps the shape of the selected edge loop the same as one of the edge loops adjacent to it, rather than sliding a percentage along each perpendicular edge.

In *Even* mode, the tool shows the position along the length of the currently selected edge which is marked in yellow, from the vertex that as an enlarged red marker. Movement of the sliding edge loop is restricted to this length. As you move the mouse the length indicator in the header changes showing where along the length of the edge you are.

To change the control edge that determines the position of the edge loop, use the `Alt-Wheel` to scroll to a different edge.

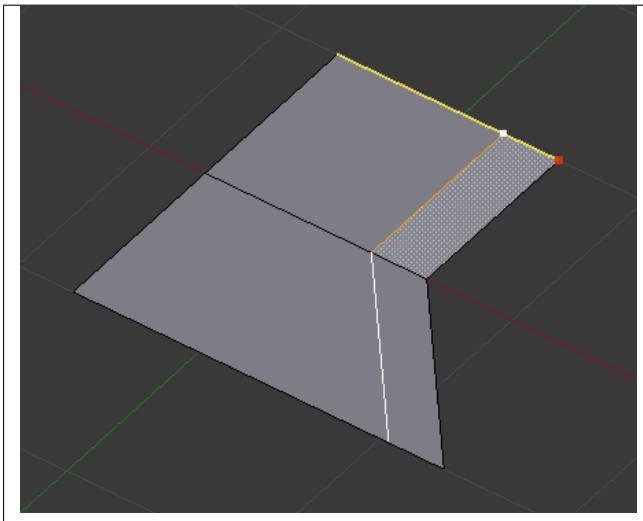


Fig. 2.437: Even Mode Enabled.

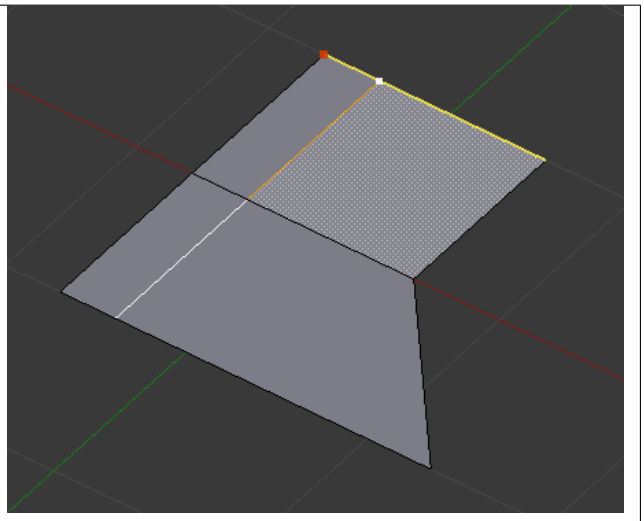


Fig. 2.438: Even Mode with Flip Enabled.

Moving the mouse moves the selected edge loop towards or away from the start vertex, but the loop line will only move as far as the length of the currently selected edge, conforming to the shape of one of the bounding edge loops.

Limitations & Workarounds

There are restrictions on the type of edge selections that can be operated upon. Invalid selections are:

Loop crosses itself This means that the tool could not find any suitable faces that were adjacent to the selected edge(s). Fig. Loop crosses is an example that shows this by selecting two edges that share the same face. A face cannot be adjacent to itself.

Multiple edge loops The selected edges are not in the same edge loop, which means they do not have a common edge. You can minimize this error by always selecting edges end to end or in a "Chain". If you select multiple edges just make sure they are connected. This will decrease the possibility of getting looping errors.

Border Edge When a single edge was selected in a single sided object. An edge loop cannot be found because there is only one face. Remember, edge loops are loops that span two or more faces.

A general rule of thumb is that if multiple edges are selected they should be connected end to end such that they form a continuous chain. This is *literally* a general rule because you can still select edges in a chain that are invalid because some of the edges in the chain are in different edge loops.

Rotate Edge

Reference

Mode: Edit Mode (Vertex or Edge select modes)
Menu: *Mesh* → *Edges* → *Rotate Edge CW* / *Rotate Edge CCW*

Rotating an edge clockwise or counter-clockwise spins an edge between two faces around their vertices. This is very useful for restructuring a mesh's topology. The tool can operate on one explicitly selected edge, or on two selected vertices or two selected faces that implicitly share an edge between them.

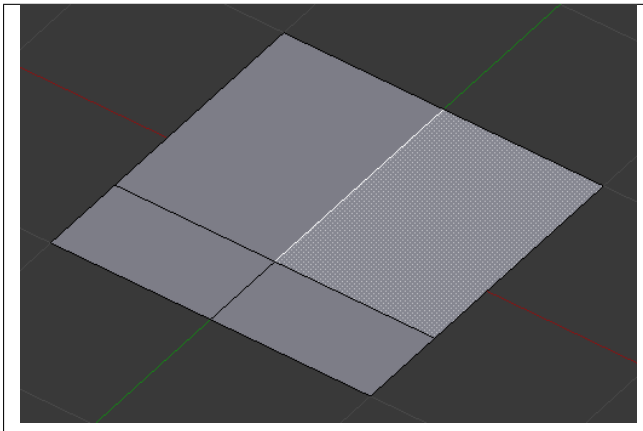


Fig. 2.439: Selected Edge.

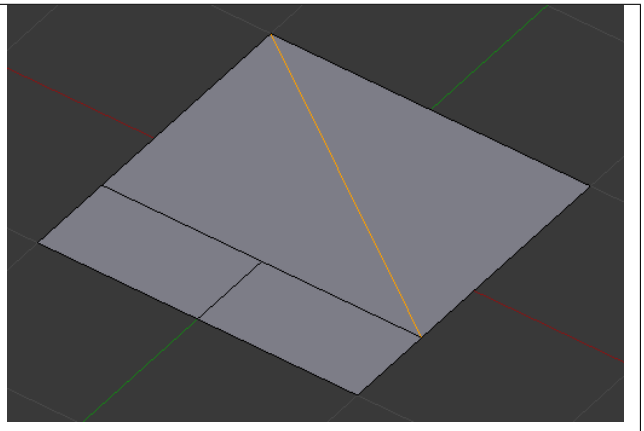


Fig. 2.440: Edge, rotated CW.

Using Face Selection

To rotate an edge based on faces you must select two faces, Fig. Adjacent selected faces, otherwise Blender notifies you with an error message, "`ERROR: Could not find any select edges that can be rotated`". Using either *Rotate Edge CW* or *Rotate Edge CCW* will produce exactly the same results as if you had selected the common edge shown in Fig. Selected edge rotated CW and CCW.

Delete Edge Loop

Reference

Mode: Edit Mode (Vertex or Edge select modes)
Menu: *Mesh* → *Delete* → *Edge Loop*
Hotkey: `X` or `Delete`, *Edge Loop*

Delete Edge Loop allows you to delete a selected edge loop if it is between two other edge loops. This will create one face-loop where two previously existed.

Note: The *Edge Loop* option is very different to the *Edges* option, even if you use it on edges that look like an edge loop. Deleting an edge loop merges the surrounding faces together to preserve the surface of the mesh. By deleting a chain of edges, the edges are removed, deleting the surrounding faces as well. This will leave holes in the mesh where the faces once were.

Example

The selected edge loop on the UV Sphere has been deleted and the faces have been merged with the surrounding edges. If the edges had been deleted by choosing *Edges* from the (*Erase* menu) there would be an empty band of deleted faces all the way around the sphere instead.

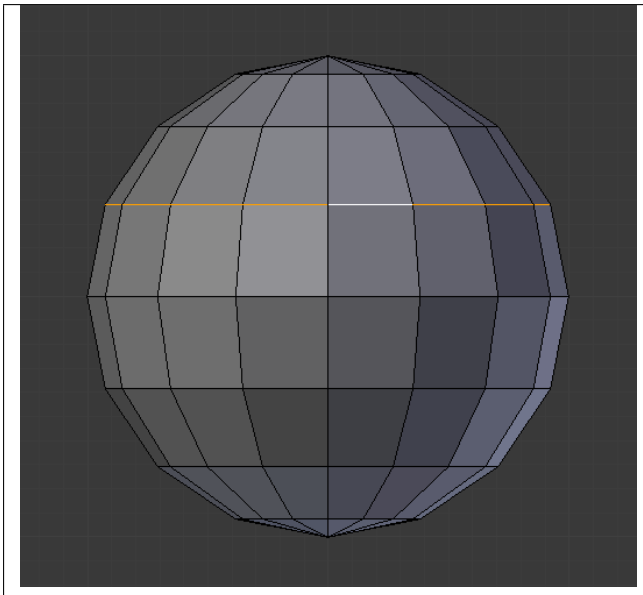


Fig. 2.441: Selected Edge Loop.

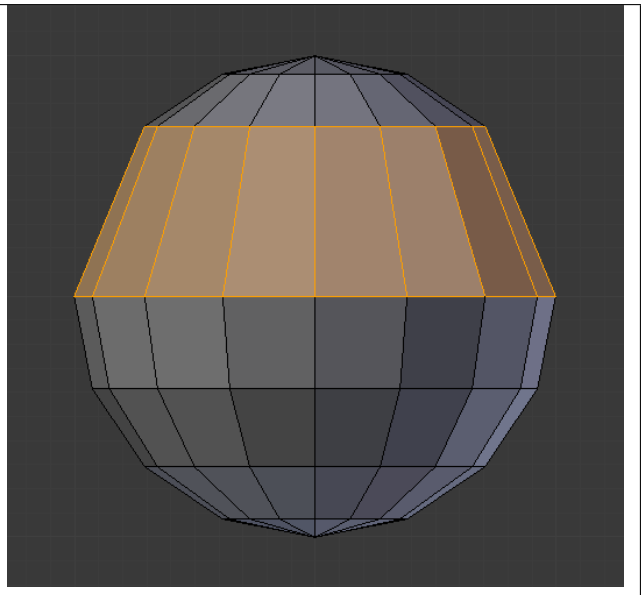


Fig. 2.442: Edge Loop Deleted.

Collapse

Reference

Mode: Edit Mode
Menu: *Mesh* → *Delete* → *Edge Collapse*
Hotkey: `Alt-M`, *Collapse*

This takes a selection of edges and for each edge, merges its two vertices together. This is useful for taking a ring of edges and collapsing it, removing the face loop it ran through.

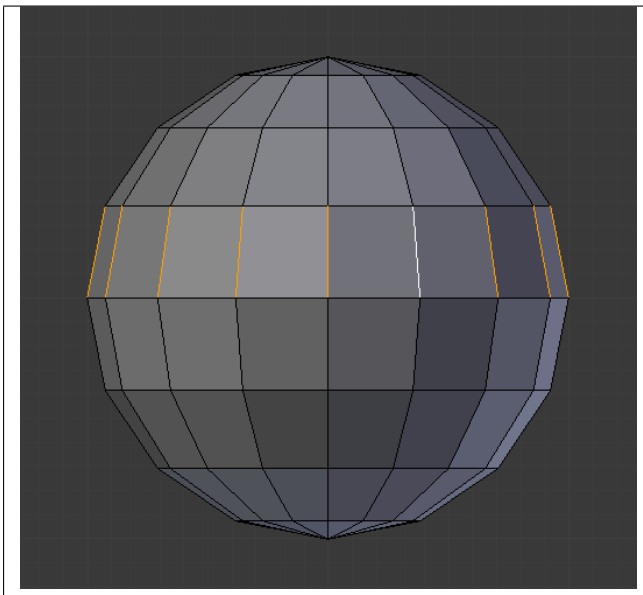


Fig. 2.443: Selected Edge Ring.

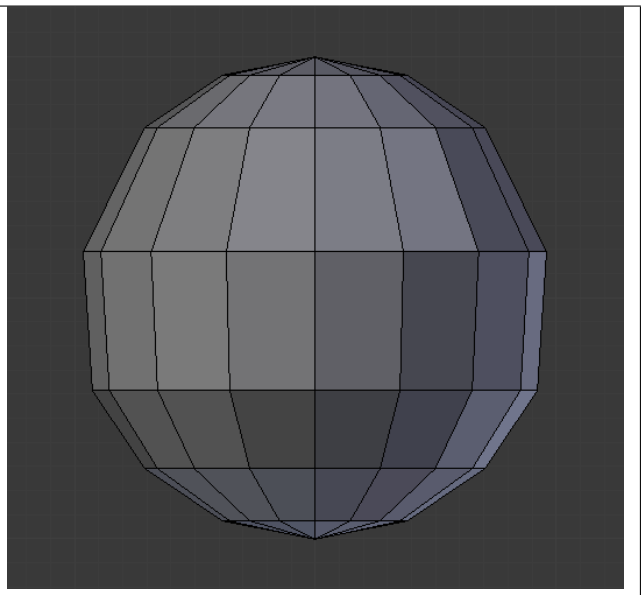


Fig. 2.444: Edge Ring Collapsed.

Edge Split

Reference

Mode: Edit Mode
Menu: *Mesh* → *Edges* → *Edge Split*

Edge split is similar to the rip tool. When two or more touching interior edges, or a border edge is selected when using *Edge split*, a hole will be created, and the selected edges are duplicated to form the border of the hole.

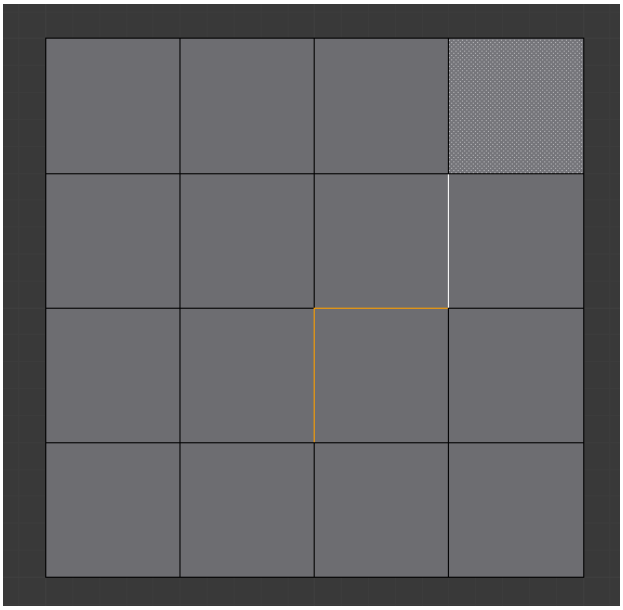


Fig. 2.445: Selected Edges.

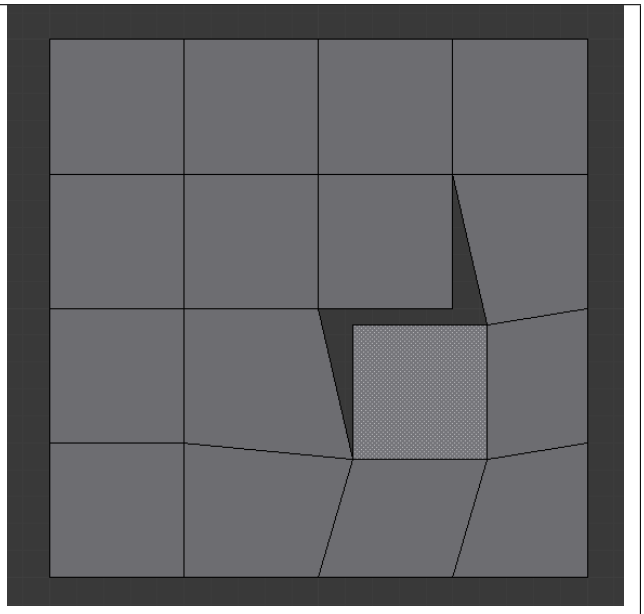


Fig. 2.446: Adjacent face moved to reveal hole left by split.

Bridge Edge Loops

Reference

Mode: Edit Mode
Menu: *Mesh* → *Edges* → *Bridge Edge Loops*

Bridge Edge Loops connects multiple edge loops with faces.

Simple example showing two closed edge loops.

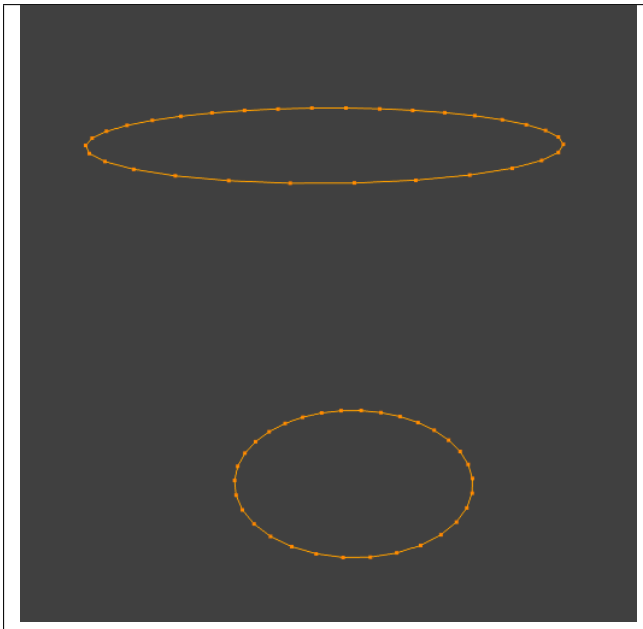


Fig. 2.447: Input.

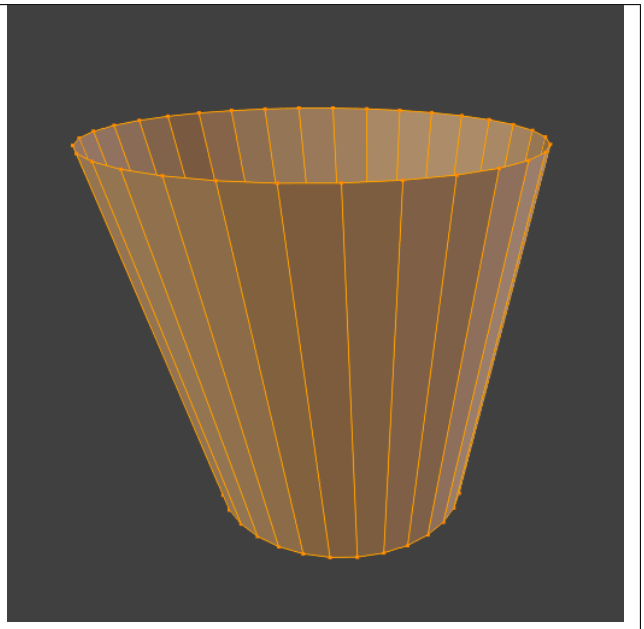


Fig. 2.448: Bridge Result.

Example of bridge tool between edge loops with different numbers of vertices.

Fig. 2.449: Input.

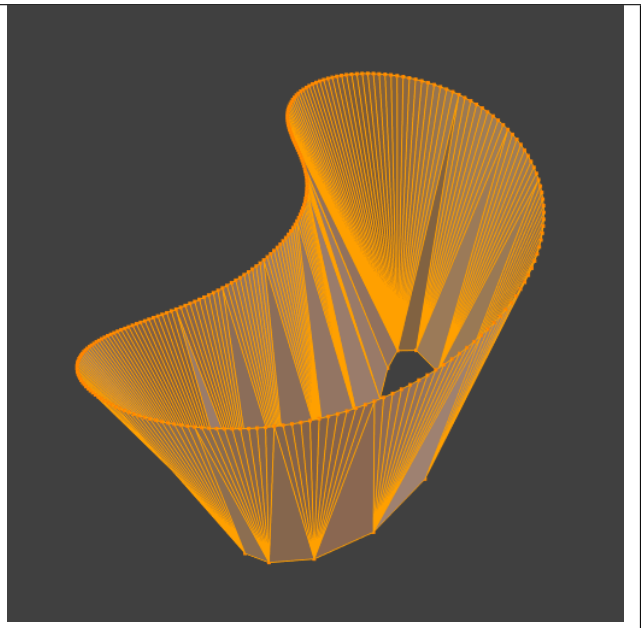


Fig. 2.450: Bridge Result.

Example using the bridge tool to punch holes in face selections and connect them.

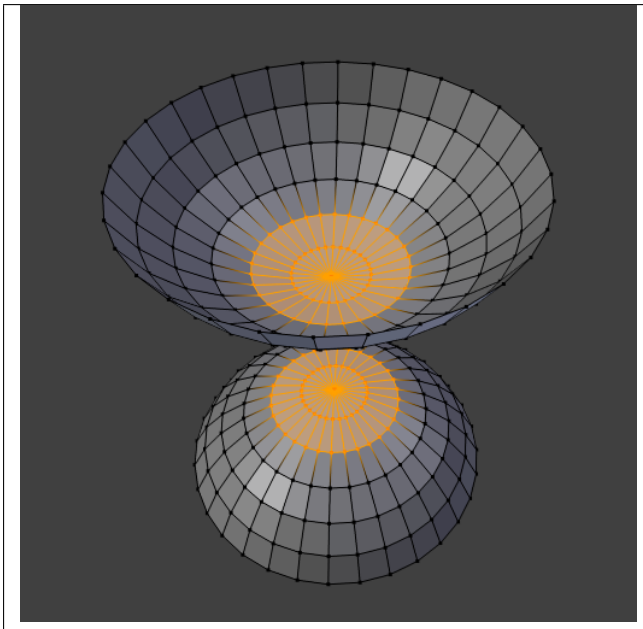


Fig. 2.451: Input.

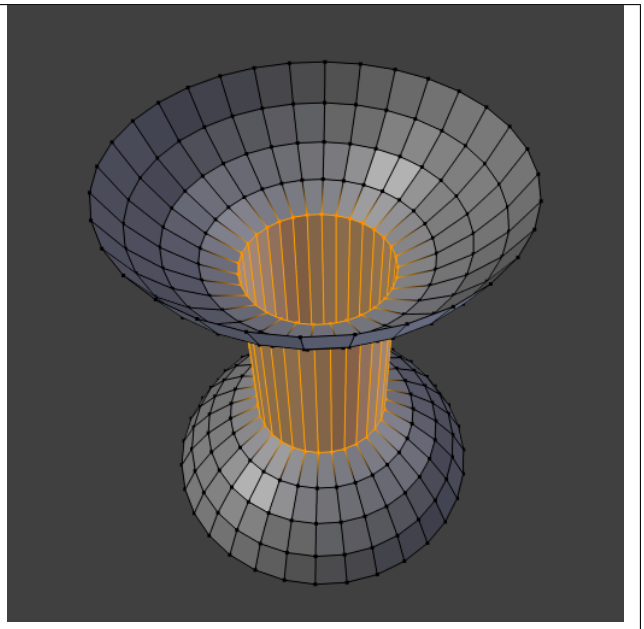


Fig. 2.452: Bridge Result.

Example showing how bridge tool can detect multiple loops and loft them in one step.

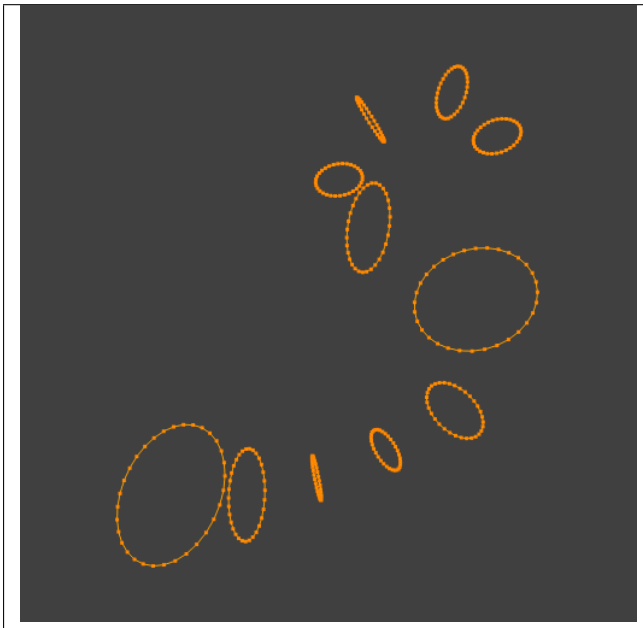


Fig. 2.453: Input.

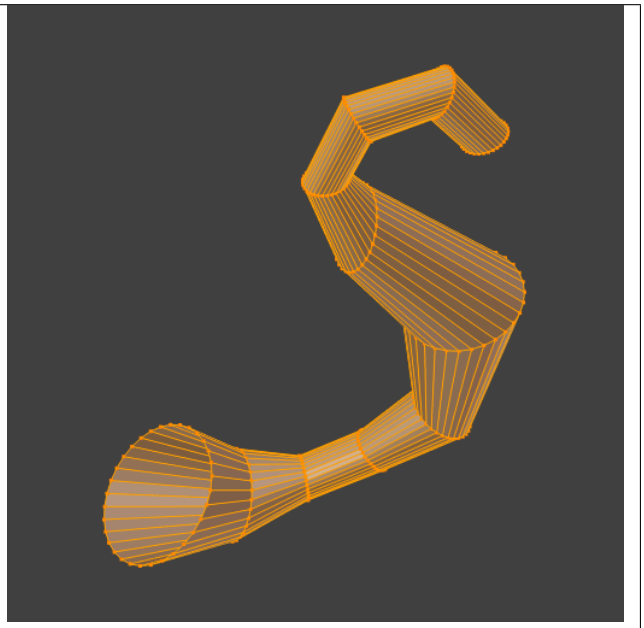


Fig. 2.454: Bridge Result.

Example of the subdivision option and surface blending with UV's.

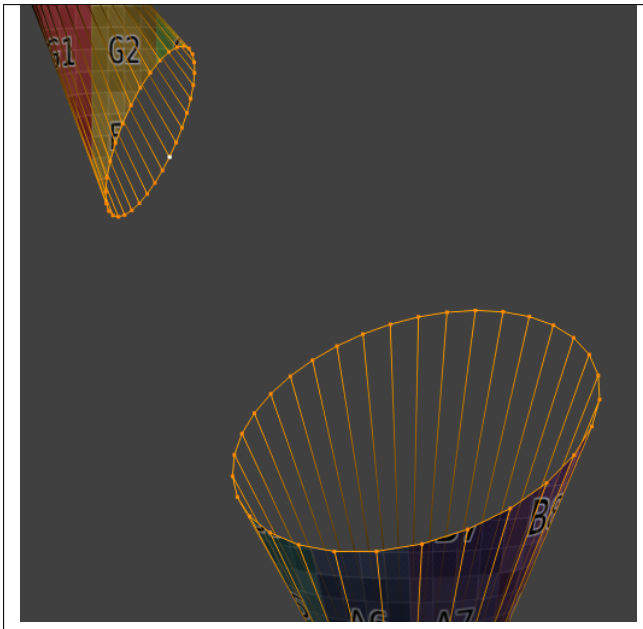


Fig. 2.455: Input.

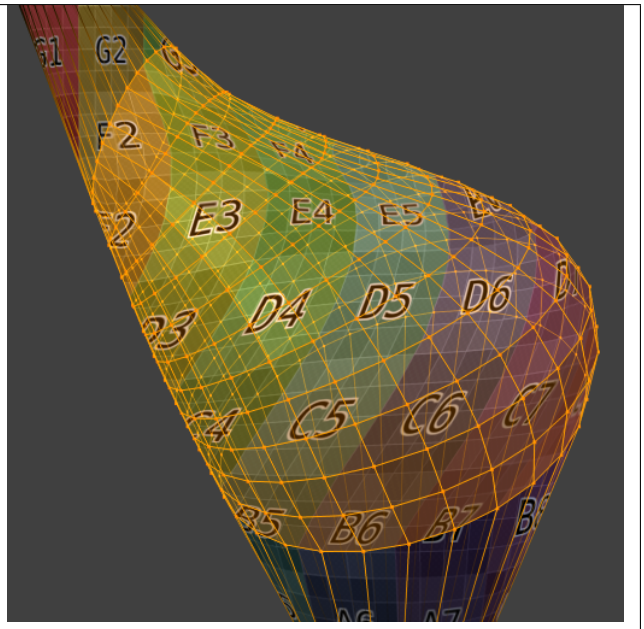


Fig. 2.456: Bridge Result.

Face Tools

These are tools that manipulate faces.

Creating Faces

Make Edge/Face

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Make Edge/Face*
Hotkey: `F`

This will create an edge or some faces, depending on your selection. Also see *Creating Geometry*.

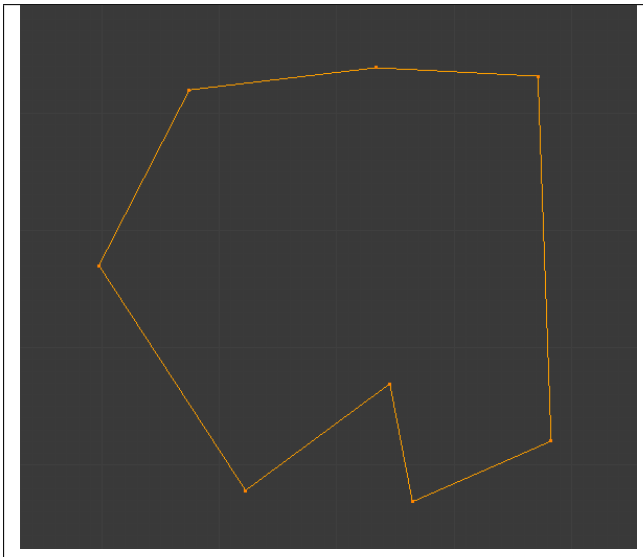


Fig. 2.457: A closed perimeter of edges.

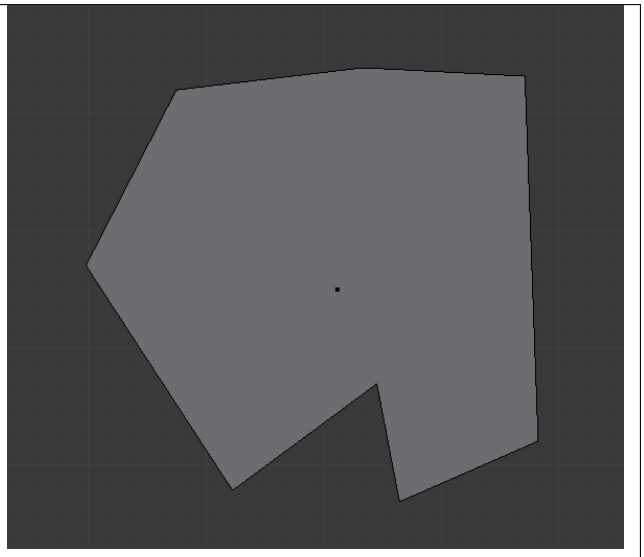


Fig. 2.458: Filled using fill.

Fill

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Fill/Beautify Fill*
Hotkey: `Alt-F`

The *Fill* option will create *triangular* faces from any group of selected edges or vertices, as long as they form one or more complete perimeters.

note, unlike creating n-gons, fill supports holes.

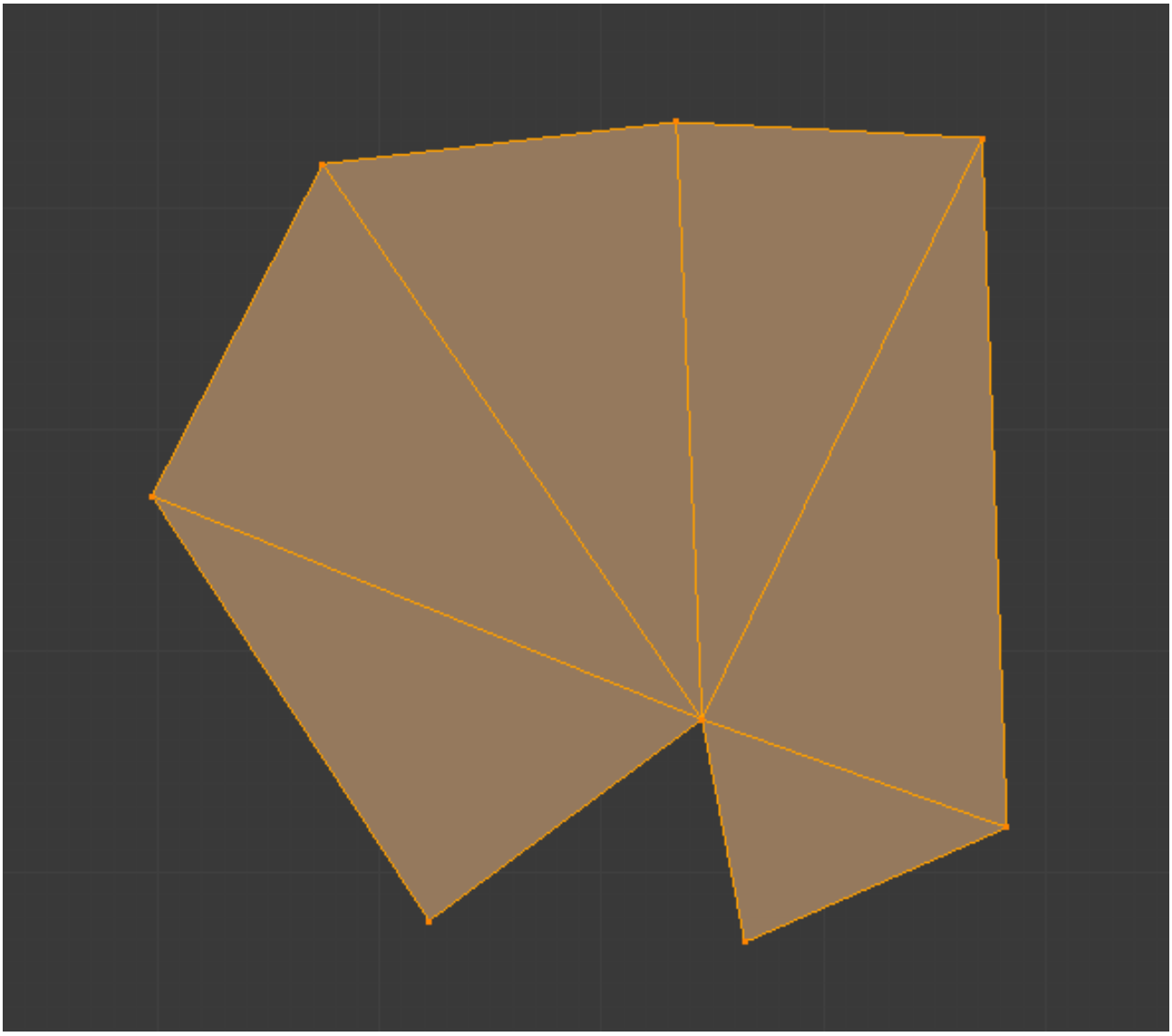


Fig. 2.459: Filled using fill.

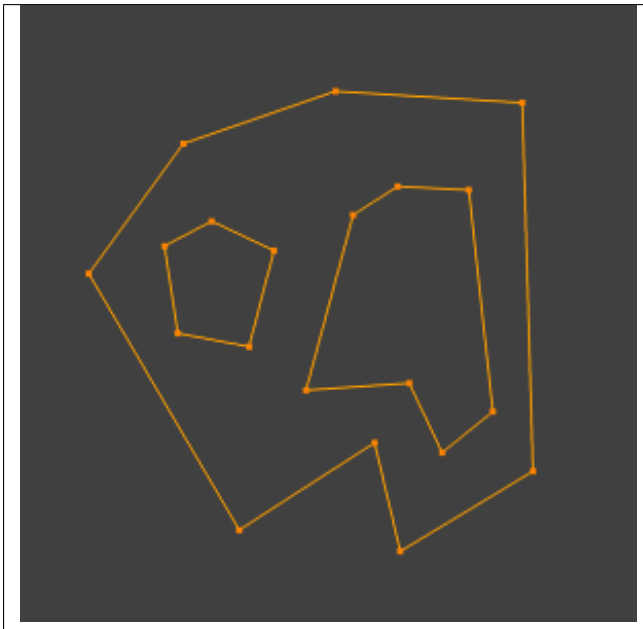


Fig. 2.460: A closed perimeter of edges with holes.

Fig. 2.461: Filled using fill.

Beauty Fill

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Fill/Beautify Fill*
Hotkey: `Alt-Shift-F`

Beautify Fill works only on selected existing faces. It rearrange selected triangles to obtain more "balanced" ones (i.e. less long thin triangles).

Fig. 2.462: Text converted to a mesh.

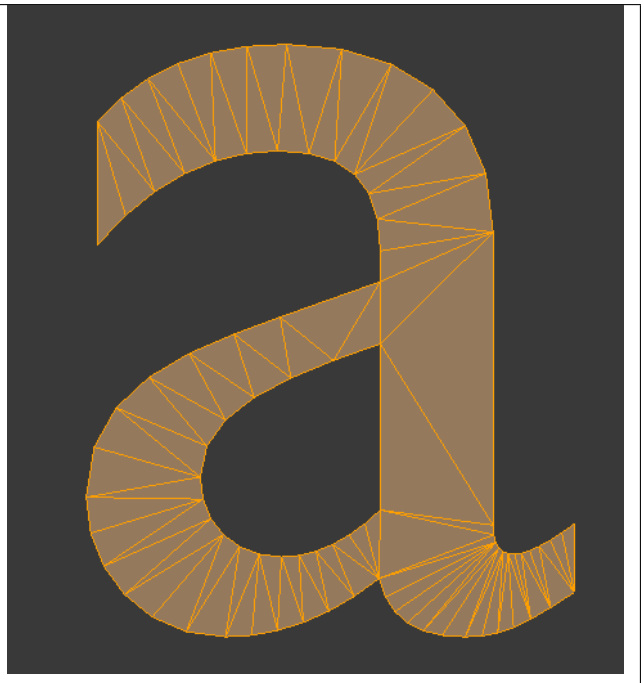


Fig. 2.463: Result of Beauty Fill, `Alt-Shift-F`.

Grid Fill

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Fill/Grid Fill*

Grid Fill uses a pair of connected edge-loops to fill in a grid that follows the surrounding geometry.

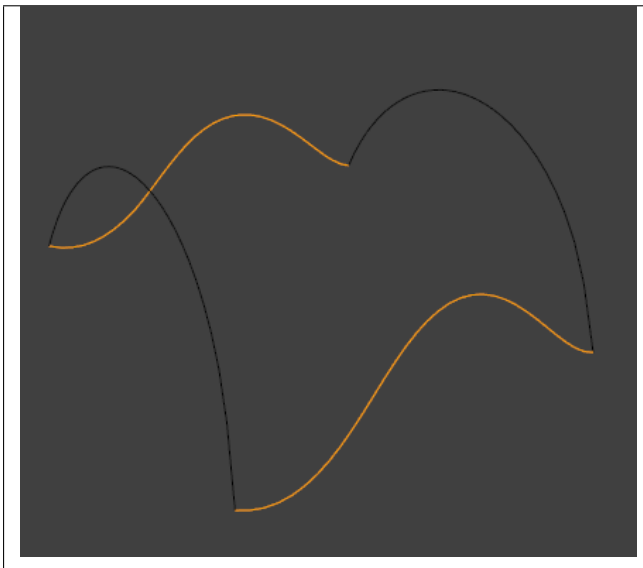


Fig. 2.464: Input.

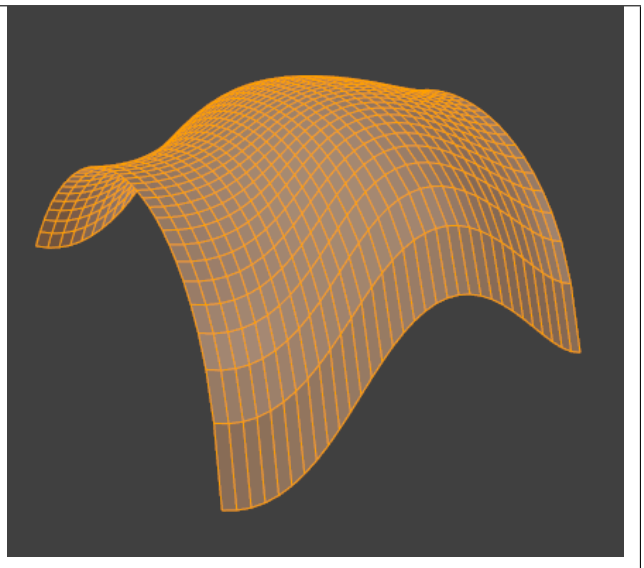


Fig. 2.465: Grid Fill Result.

Convert Quads to Triangles

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Convert Quads to Triangles* or *Face Specials* → *Triangulate*
Hotkey: `Ctrl-T`

As its name intimates, this tool converts each selected quadrangle into two triangles. Remember that quads are just a set of two triangles.

Convert Triangles to Quads

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Faces* → *Convert Triangles to Quads*
Hotkey: `Alt-J`

This tool converts the selected triangles into quads by taking adjacent tris and removes the shared edge to create a quad, based on a threshold. This tool can be performed on a selection of multiple triangles.

This same action can be done on a selection of two tris, by selecting them and using the shortcut `F`, to create a face, or by selecting the shared edge and dissolving it with the shortcut `X` → *Dissolve*.

To create a quad, this tool needs at least two adjacent triangles. If you have an even number of selected triangles, it is also possible not to obtain only quads. In fact, this tool tries to create "squarishest" quads as possible from the given triangles, which means some triangles could remain.

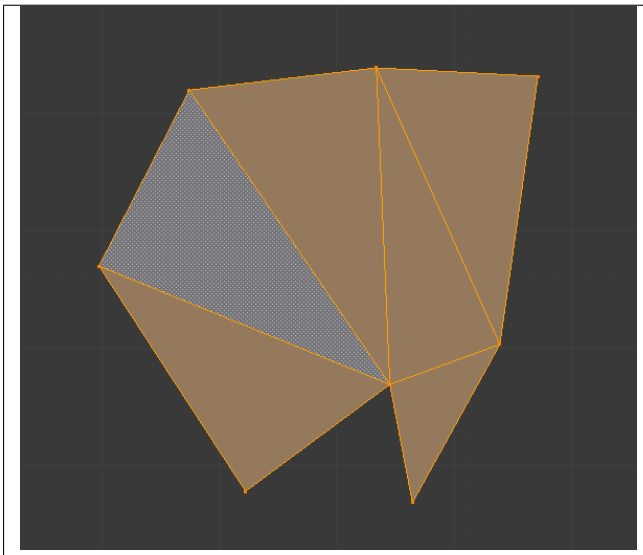


Fig. 2.466: Before converting tris to quads.

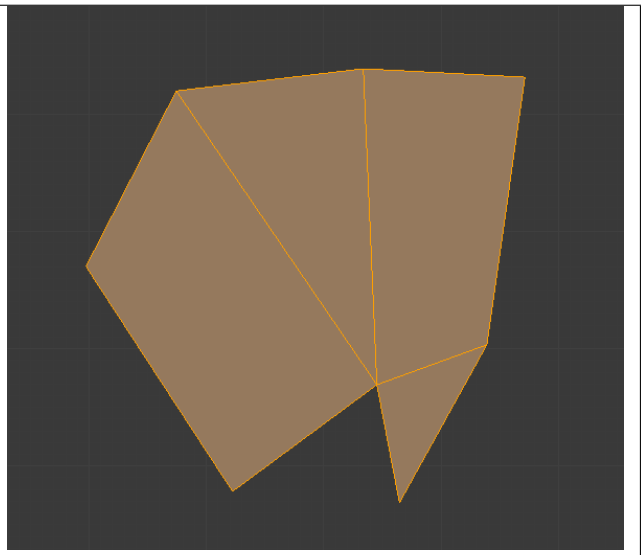


Fig. 2.467: After converting tris to quads.

All the menu entries and hotkey use the settings defined in the *Mesh Tools* panel:

Max Angle This values, between (0 to 180), controls the threshold for this tool to work on adjacent triangles. With a threshold of 0.0, it will only join adjacent triangles that form a perfect rectangle (i.e. right-angled triangles sharing their hypotenuses). Larger values are required for triangles with a shared edge that is small, relative to the size of the other edges of the triangles.

Compare UVs When enabled, it will prevent union of triangles that are not also adjacent in the active UV map.

Compare Vcol When enabled, it will prevent union of triangles that have no matching vertex color.

Compare Sharp When enabled, it will prevent union of triangles that share a "sharp" edge.

Compare Materials When enabled, it will prevent union of triangles that do not use the same material index.

Solidify

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Solidify*
Hotkey: `Ctrl-F` → *Solidify*

This takes a selection of faces and solidifies them by extruding them uniformly to give volume to a *non-manifold* surface. This is also available as a *Modifier*. After using the tool, you can set the offset distance in the Tool Palette.

Thickness Amount to offset the newly created surface. Positive values offset the surface inward relative to the normals. Negative values offset outward.

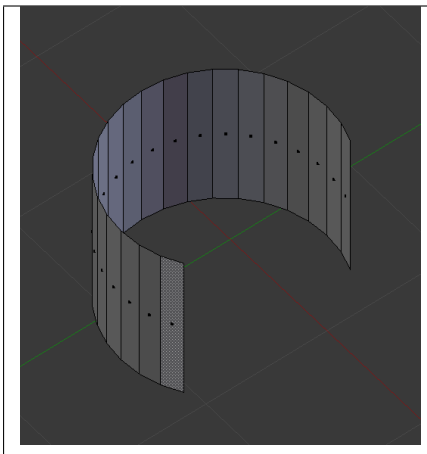


Fig. 2.468: Mesh before solidify operation.

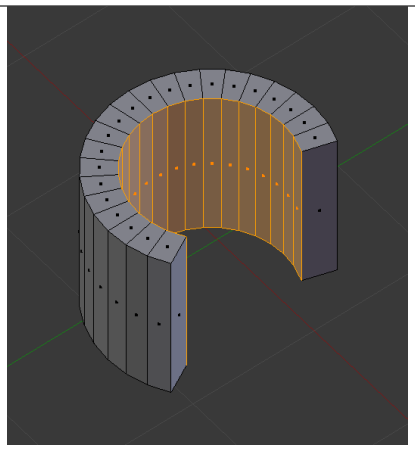


Fig. 2.469: Solidify with a positive thickness.

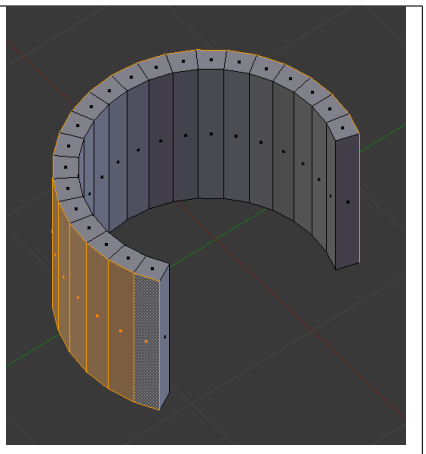


Fig. 2.470: Solidify with a negative thickness.

Rotate Edges

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Rotate Edge CW*

This command functions the same edge rotation in edge mode.

It works on the shared edge between two faces and rotates that edge if the edge was selected.

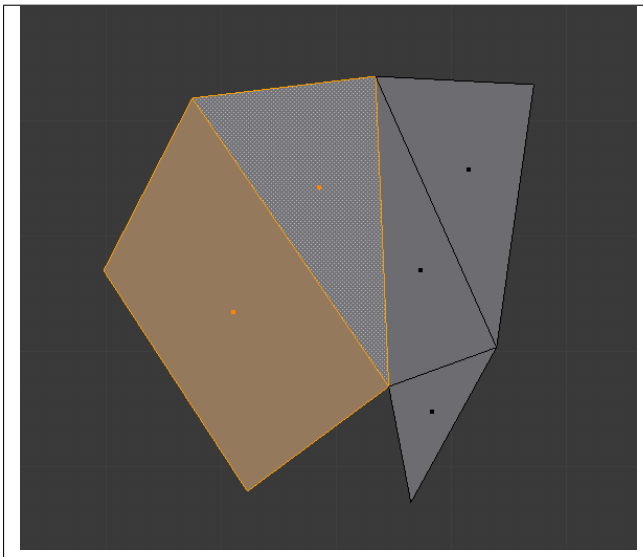


Fig. 2.471: Two Faces Selected.

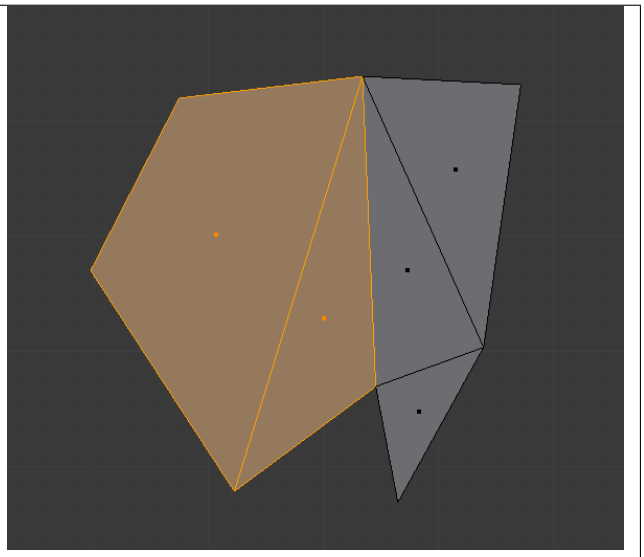


Fig. 2.472: Full Render.

See *Rotate Edge* for more information.

Normals

See *Editing Normals* for more information.

Normals

Introduction

Todo.

Editing

Flip Direction

Reference

Mode: Edit Mode
Menu: *Mesh* → *Normals* → *Flip* or *Specials* → *Flip Normals*
Hotkey: `W`, *Flip Normals*

Well, it will just reverse the normals direction of all selected faces. Note that this allows you to precisely control the direction (**not** the orientation, which is always perpendicular to the face) of your normals, as only selected ones are flipped.

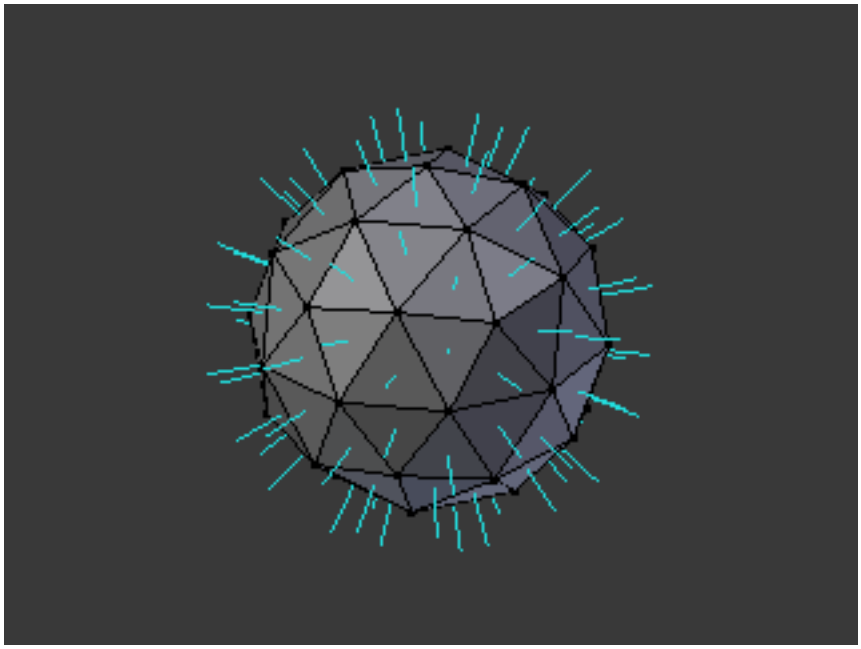


Fig. 2.473: Normals visualization.

Recalculate Normals

Reference

Mode: Edit Mode
Menu: *Mesh* → *Normals* → *Recalculate Outside* and *Mesh* → *Normals* → *Recalculate Inside*
Hotkey: `Ctrl-N` and `Ctrl-Shift-N`

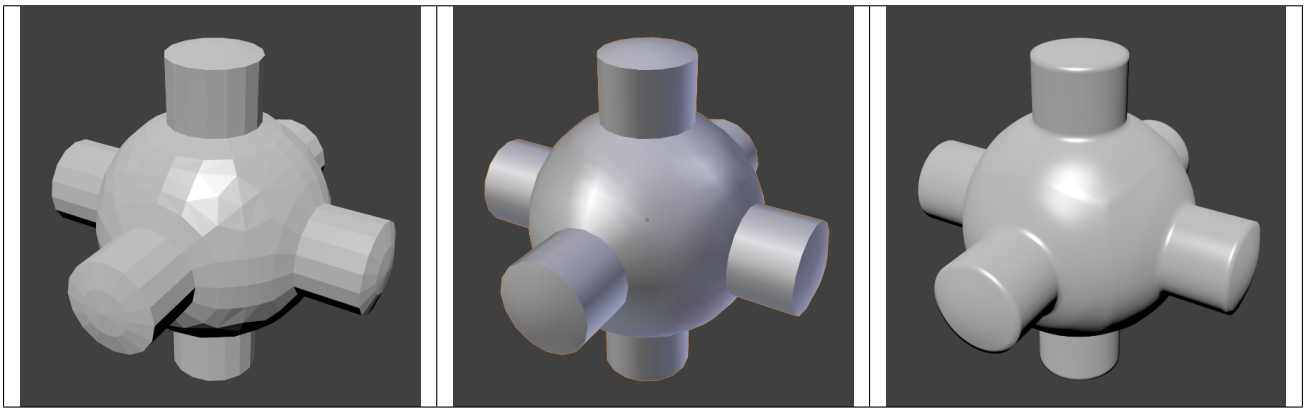
These commands will recalculate the normals of selected faces so that they point outside (respectively inside) the volume that the face belongs to. This volume do not need to be closed. In fact, this means that the face of interest must be adjacent with at least one non-coplanar other face. For example, with a *Grid* primitive, recalculating normals does not have a meaningful result.

Tip: For Visualization in *Edit Mode* see *Normals*.

Smoothing

Mesh Shading

Table 2.15: Example mesh rendered flat, smoothed using edge split, and using Subdivision Surface. Note how edges are rendered differently. Sample blend-file.



As seen in the previous sections, polygons are central to Blender. Most objects are represented by polygons and truly curved objects are often approximated by polygon meshes. When rendering images, you may notice that these polygons appear as a series of small, flat faces.

Sometimes this is a desirable effect, but usually we want our objects to look nice and smooth. This section shows you how to visually smooth an object, and how to apply the *Auto Smooth* filter to quickly and easily combine smooth and faceted polygons in the same object.

The last section on this page shows possibilities for smoothing a mesh's geometry, not only its appearance.

Smooth Shading

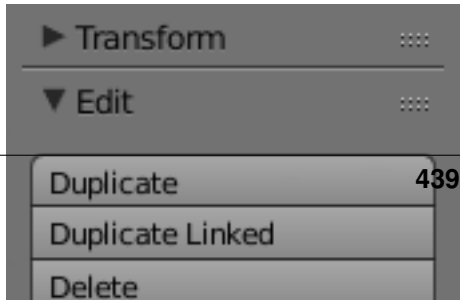
Reference

Mode: Object Mode
Panel: *Tool Shelf* → *Tools* → *Edit* → *Shading*

Reference

Mode: Edit Mode
Panel: *Tool Shelf* → *Shading/ UVs* → *Shading*
Menu: *Mesh* → *Faces* → *Shade Smooth / Shade Flat*
Hotkey: `Ctrl-F` → *Shade Smooth / Shade Flat*

The easiest way is to set an entire object as smooth or faceted by selecting a mesh object, and in *Object Mode*, click *Smooth* in the *Tool Shelf*. This button does not stay pressed; it forces the assignment of the "smoothing" attribute to each face in the mesh, including when you add or delete geometry.



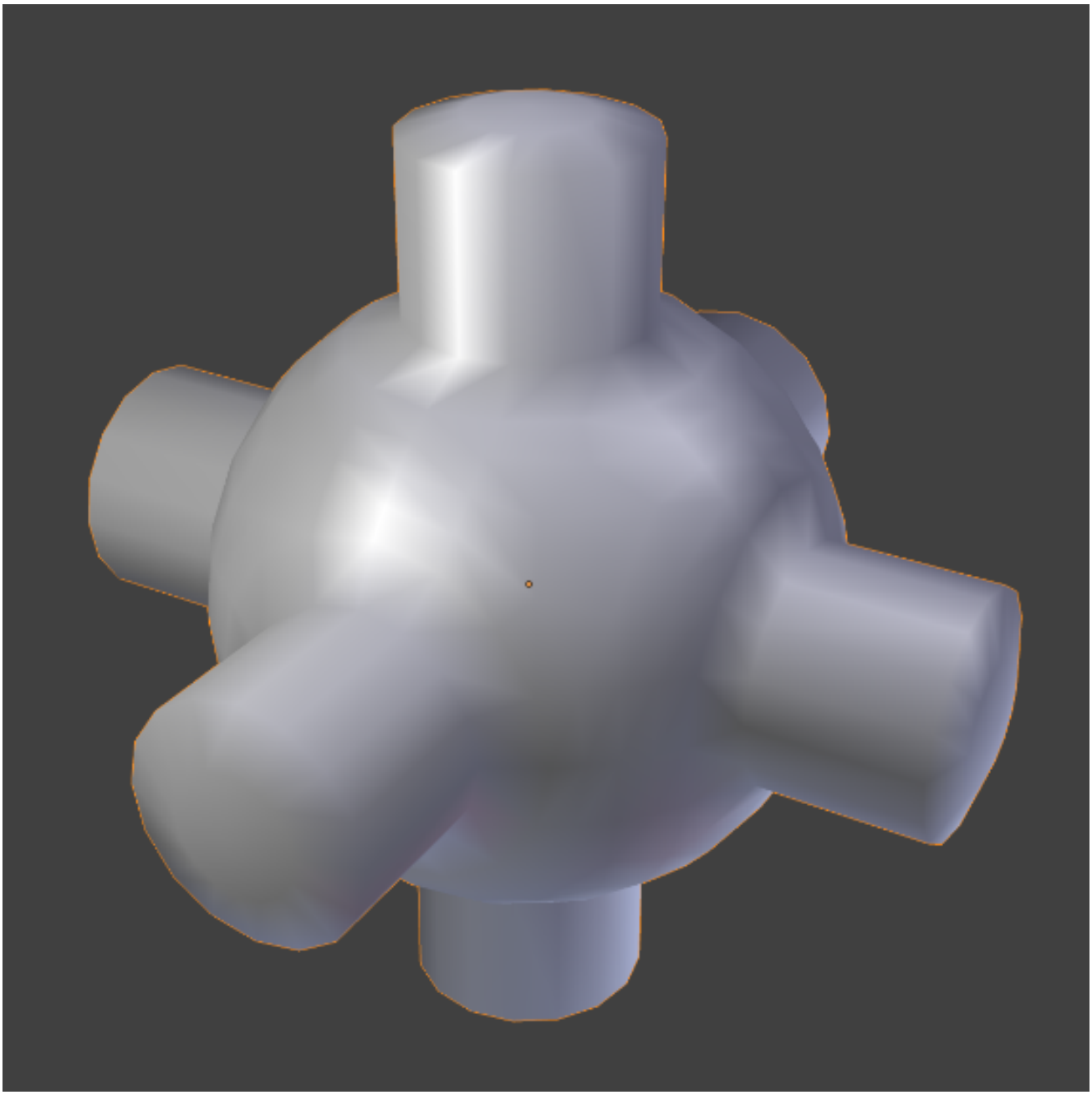


Fig. 2.475: Same mesh smooth shaded.

Notice that the outline of the object is still strongly faceted. Activating the smoothing features does not actually modify the object's geometry; it changes the way the shading is calculated across the surfaces, giving the illusion of a smooth surface. Click the *Flat* button in the *Tool Shelf* 's *Shading panel* to revert the shading back to that shown in the first image above.

Smoothing parts of a mesh

Alternatively, you can choose which edges to smooth by entering *Edit Mode*, then selecting some faces and clicking the *Smooth* button. The selected edges are marked in yellow.

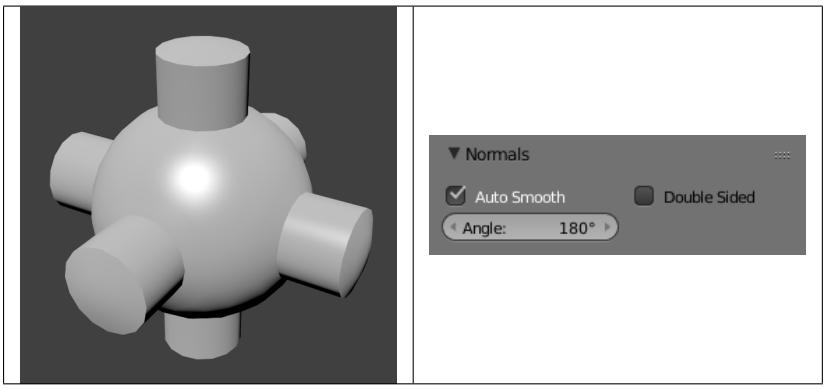
When the mesh is in *Edit Mode*, only the selected edges will receive the "smoothing" attribute. You can set edges as flat (removing the "smoothing" attribute) in the same way by selecting edges and clicking the *Flat* button.

Auto Smooth

Reference

Panel: *Properties editor* → *Object Data*

Table 2.16: Normals panel with *Auto Smooth* enabled.



It can be difficult to create certain combinations of smooth and solid faces using the above techniques alone. Though there are workarounds (such as splitting off sets of faces by selecting them and pressing `Y`), there is an easier way to combine smooth and solid faces, by using *Auto Smooth*. Auto smoothing can be enabled in the mesh tab in the Properties Editor in the *Normals* panel.

Edge Split Modifier

With the *Edge Split Modifier* a result similar to *Auto Smooth* can be achieved with the ability to choose which edges should be split, based on angle. Those Angles are marked as sharp.

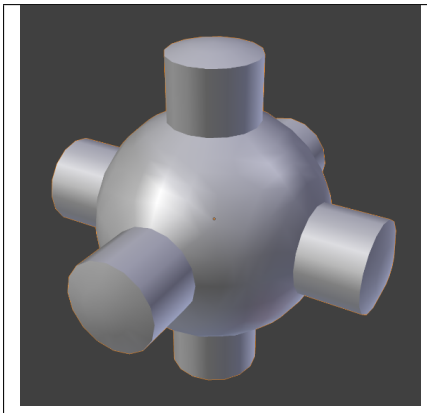


Fig. 2.477: Edge Split modifier enabled, based on angle.

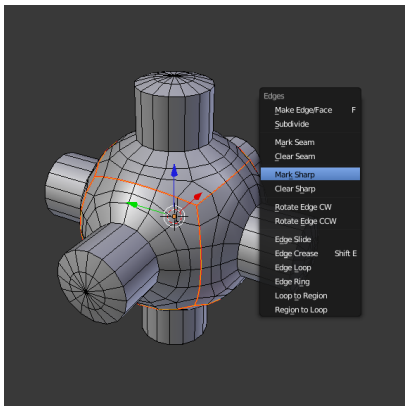


Fig. 2.478: Edges marked as sharp.

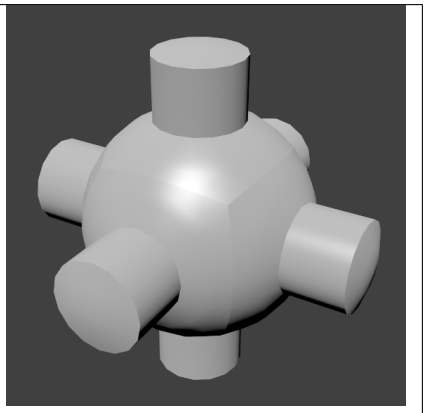


Fig. 2.479: Resulting render with sharp edge weighting.

Smoothing the mesh geometry

The above techniques do not alter the mesh itself, only the way it is displayed and rendered. Instead of just making the mesh look like a smooth surface, you can also physically smooth the geometry of the mesh with these tools:

Mesh editing tools

You can apply one of the following in *Edit Mode*:

Smooth This relaxes selected components, resulting in a smoother mesh.

Laplacian Smooth Smooths geometry by offers controls for better preserving larger details.

Subdivide Smooth Adjusting the *smooth* parameter after using the *subdivide* tool results in a more organic shape. This is similar to using the subdivide modifier.

Bevel This Bevels selected edged, causing sharp edges to be flattened.

Modifiers

Alternatively, you can smooth the mesh non-destructively with one or several of the following modifiers:

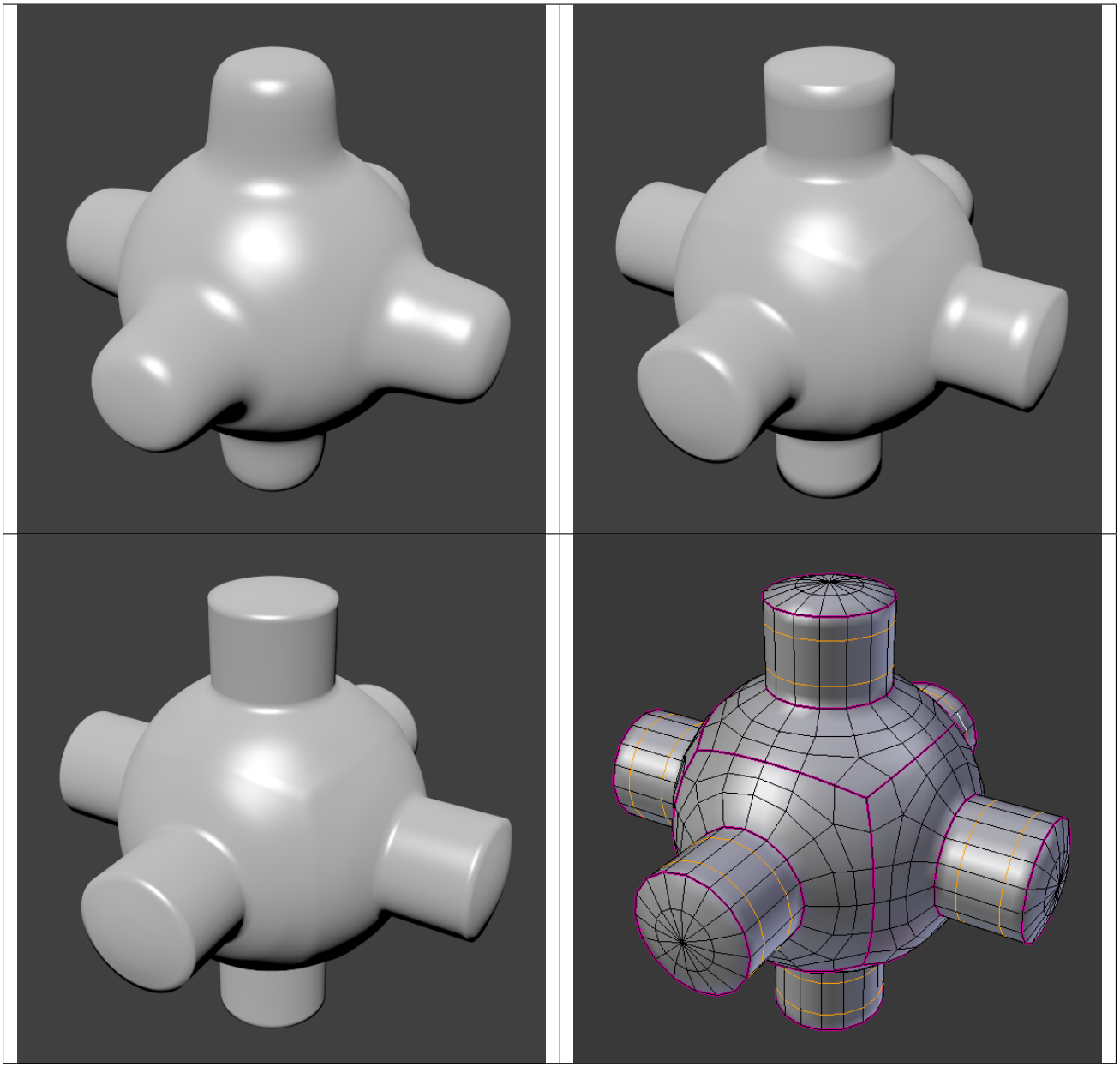
Smooth Modifier Works like the *Smooth* tool in *Edit Mode*; can be applied to specific parts of the mesh using vertex groups.

Laplactian Smooth Modifier Works like the *Laplacian Smooth* tool in *Edit Mode*; can be applied to specific parts of the mesh using vertex groups.

Bevel Modifier Works like the *Bevel* tool in *Edit Mode*; Bevel can be set to work on an angle threshold, or on edge weight values.

Subdivision Surface Modifier Catmull-Clark subdivision produces smooth results. Sharp edges can be defined with *subdivision creases* or by setting certain edges to "sharp" and adding an *Edge Split Modifier* (set to *From Marked As Sharp*) before the *Subdivision Surface* modifier.

Table 2.17: 3D View showing creased edges (pink) and added edges loops (yellow).



Transformation

Mirror

Reference

Mode: Edit Mode
Menu: *Mesh* → *Mirror* → *Desired Axis*
Hotkey: `Ctrl-M`

The mirror tool mirrors a selection across a selected axis.

The mirror tool in *Edit Mode* is similar to *Mirroring in Object Mode*. It is exactly equivalent to scaling by -1 vertices, edges or faces around one chosen pivot point and in the direction of one chosen axis, only it is faster/handier.

After this tool becomes active, select an axis to mirror the selection on entering `X`, `Y`, or `Z`.

You can also interactively mirror the geometry by holding the `MMB` and dragging in the direction of the desired mirror direction.

Axis of symmetry

For each transformation orientation, you can choose one of its axes along which the mirroring will occur.

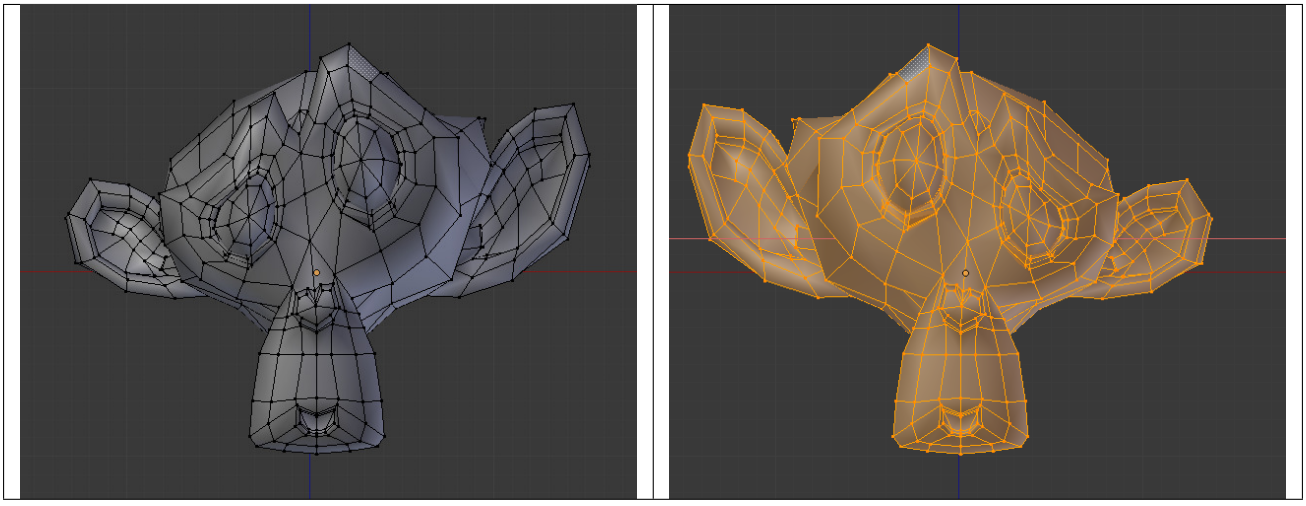
As you can see, the possibilities are infinite and the freedom complete: you can position the pivot point at any location around which we want the mirroring to occur, choose one transformation orientation and then one axis on it.

Pivot Point

Pivot points must be set first. Pivot points will become the center of symmetry. If the widget is turned on it will always show where the pivot point is.

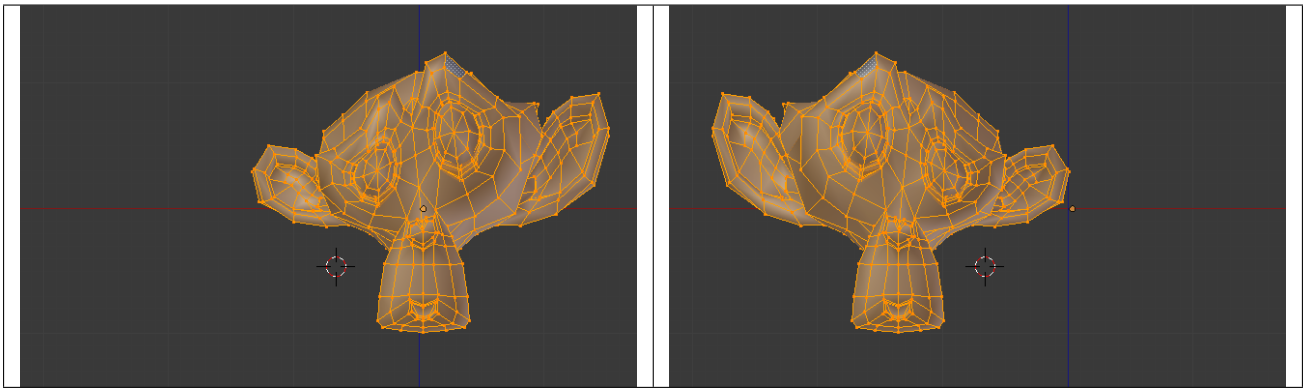
In Fig. *Mirror around the Individual Centers.* the pivot point default to median point of the selection of vertices in *Edit Mode*. This is a special case of the *Edit Mode* as explained on the *pivot point page*.

Table 2.18: Mesh after mirrored along X axis.



In Fig. *Mirror around the 3D Cursor.* the pivot point is the *3D Cursor*, the transformation orientation is *Local*, a.k.a. the Object space, and the axis of transformation is X.

Table 2.19: Mesh after mirrored along X axis using the 3D cursor as a pivot point.



Transformation Orientations

Transformation Orientations are found on the 3D View header, next to the *Widget* buttons. They decide which coordinate system will rule the mirroring.

Shrink Fatten

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Transform* → *Shrink/Fatten Along Normals*
Hotkey: `Alt-S`

This tool translates selected vertices/edges/faces along their own normal (perpendicular to the face), which, on "standard normal meshes", will shrink/fatten them.

This transform tool does not take into account the pivot point or transform orientation.

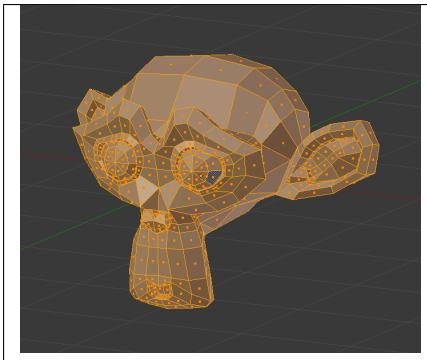


Fig. 2.488: Mesh before shrink/flatten.

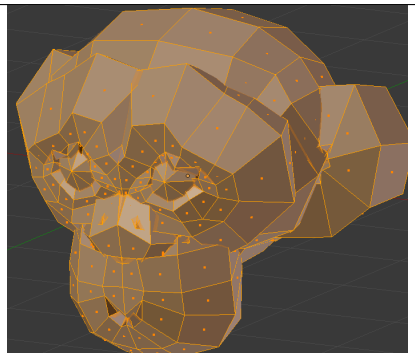


Fig. 2.489: Inflated using a positive value.

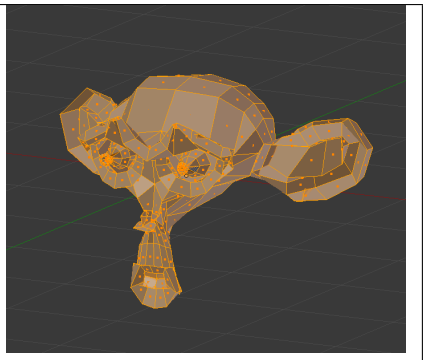


Fig. 2.490: Shrunk using a negative value.

Smooth

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Vertices* → *Smooth vertex*
Hotkey: `Ctrl-V` → *Smooth vertex*

This tool smooths the selected components by averaging the angles between faces. After using the tool, options appear in the *Tool Shelf*:

Number of times to smooth The number of smoothing iterations

Axes Limit the effect to certain axes.

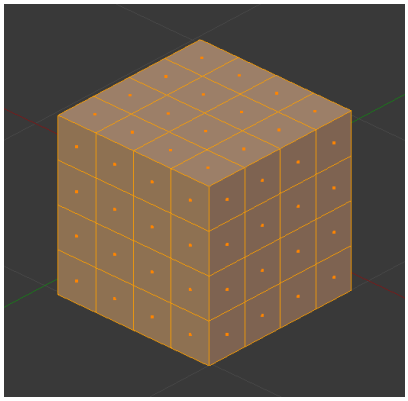


Fig. 2.491: Mesh before smoothing.

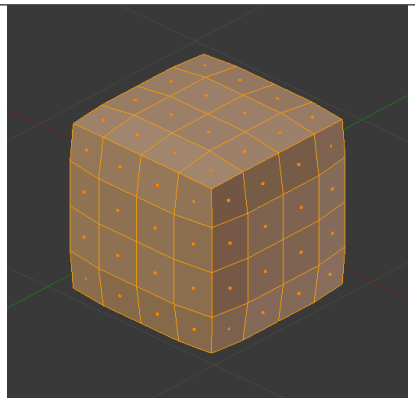


Fig. 2.492: Mesh after one smoothing iteration.

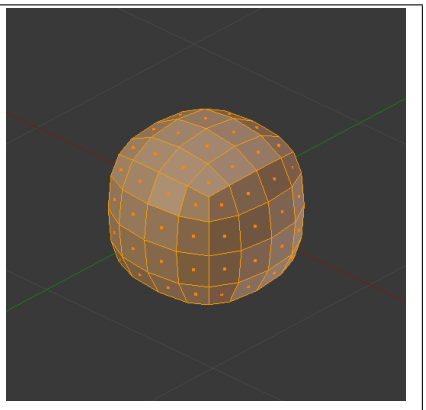


Fig. 2.493: Mesh after ten smoothing iterations.

Laplacian Smooth

Reference

Mode: Edit Mode
Hotkey: `W` → *Laplacian Smooth*

See the *Laplacian Smooth Modifier* for details.

Laplacian smooth is uses an alternative smoothing algorithm that better preserves the overall mesh shape. Laplacian smooth exists as a mesh operation and as a non-destructive modifier.

Note: The *Smooth modifier*, which can be limited to a *Vertex Group*, is a non-destructive alternative to the smooth tool.

Note: Real Smoothing versus Shading Smoothing

Do not mistake this tool with the shading smoothing options described at *this page*, they do not work the same! This tool modifies the mesh itself, to reduce its sharpness, whereas *Set Smooth* / *AutoSmooth* and co. only control the way the mesh

is shaded, creating an *illusion* of softness, but without modifying the mesh at all...

Noise

Reference

Mode: Edit Mode
Panel: Mesh Tools

Note: *Noise* is an old feature. The *Displace Modifier* is a non-destructive alternative to the Noise tool and is a more flexible way to realize these sort of effects. The key advantages of the modifier are that it can be canceled at any moment, you can precisely control how much and in which direction the displacement is applied, and much more...

The *Noise* function allows you to displace vertices in a mesh based on the gray values of the first texture slot of the material applied to the mesh.

The mesh must have a material and a texture assigned to it for this tool to work. To avoid having the texture affect the material's properties, it can be disabled in the texture menu.

The *Noise* function displaces vertices along the object's ±Z-Axis only.

Noise permanently modifies your mesh according to the material texture. Each click adds onto the current mesh. For a temporary effect, map the texture to Displacement for a render-time effect. In *Object Mode* or *Edit Mode*, your object will appear normal, but will render deformed.

The deformation can be controlled by modifying the *Mapping* panel and/or the texture's own panel (e.g. *Clouds*, *Marble*, etc.).

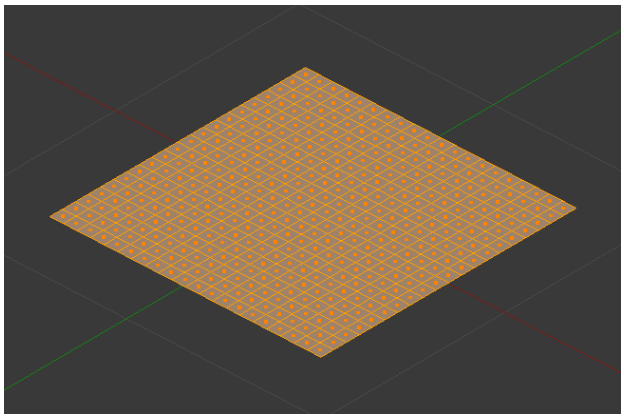


Fig. 2.494: Mesh before noise is added.

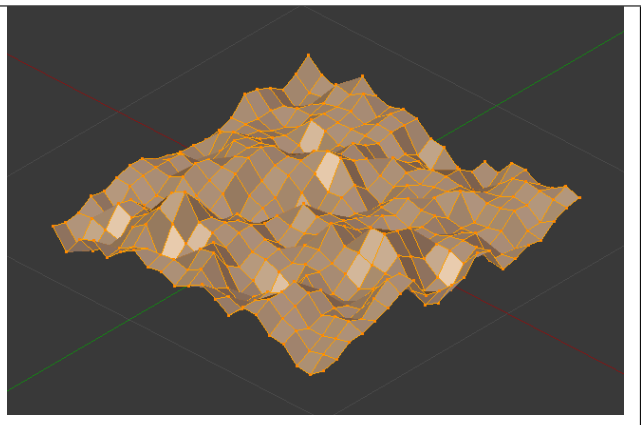


Fig. 2.495: Mesh after noise is added, using basic cloud texture.

Push/Pull

Reference

Mode: Object and Edit Modes

Menu: *Object/Mesh* → *Transform* → *Push Pull*

Push/Pull will move the selected elements (Objects, vertices, edges or faces) closer together (Push) or further apart (Pull). Specifically, each element is moved towards or away from the center by the same distance. This distance is controlled by moving the mouse up (Push) or down (Pull), numeric input or through slider control.

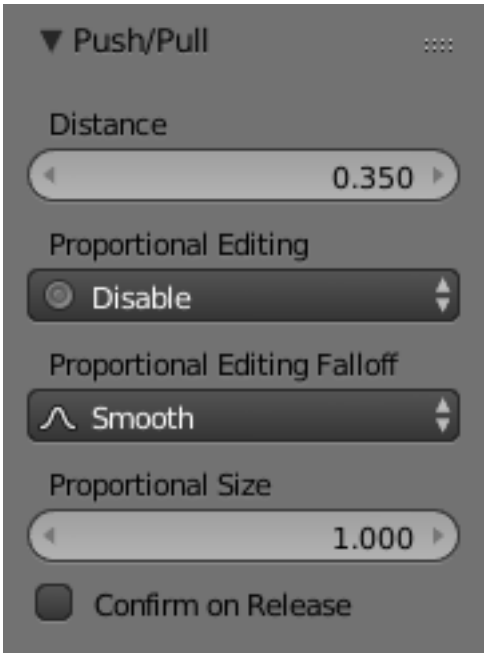


Fig. 2.496: Push/Pull distance.

Usage

Select the elements you want to operate on and activate the Push/Pull transform function. The Push/Pull option can be invoked from the *Object/Mesh* → *Transform* → *Push/Pull* menu option or by pressing `Spacebar` and using the search menu to search for *Push* or *Pull*. The amount of movement given to the selection can be determined interactively by moving the mouse or by typing a number. Pressing `Return` will confirm the transformation. The confirmed transformation can be further edited by pressing `F6` or by going into the Tool Shelf `T` and altering the Distance slider provided that no other actions take place between the *Push/Pull* transform confirmation and accessing the slider.

Note that the result of the *Push/Pull* transform is also dependant on the number and type of selected elements (Objects, vertices, faces etc). See below for the result of using *Push/Pull* on a number of different elements.

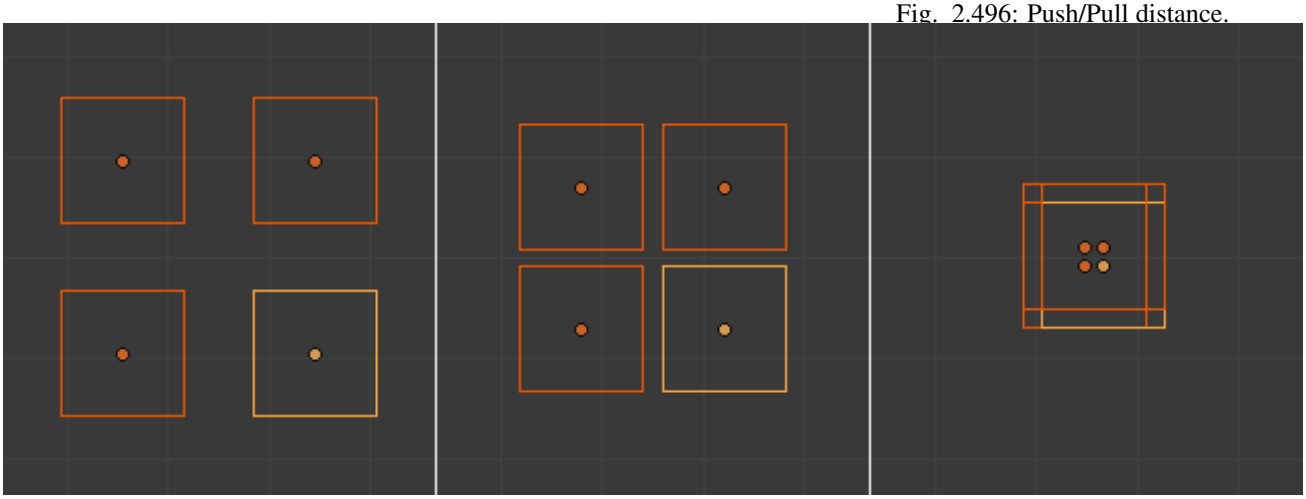


Fig. 2.497: Equidistant Objects being pushed together.

Shear

Reference

Mode: Object and Edit Modes
Menu: *Object/Mesh/Curve/Surface* → *Transform* → *Shear*
Hotkey: `Shift-Ctrl-Alt-S`

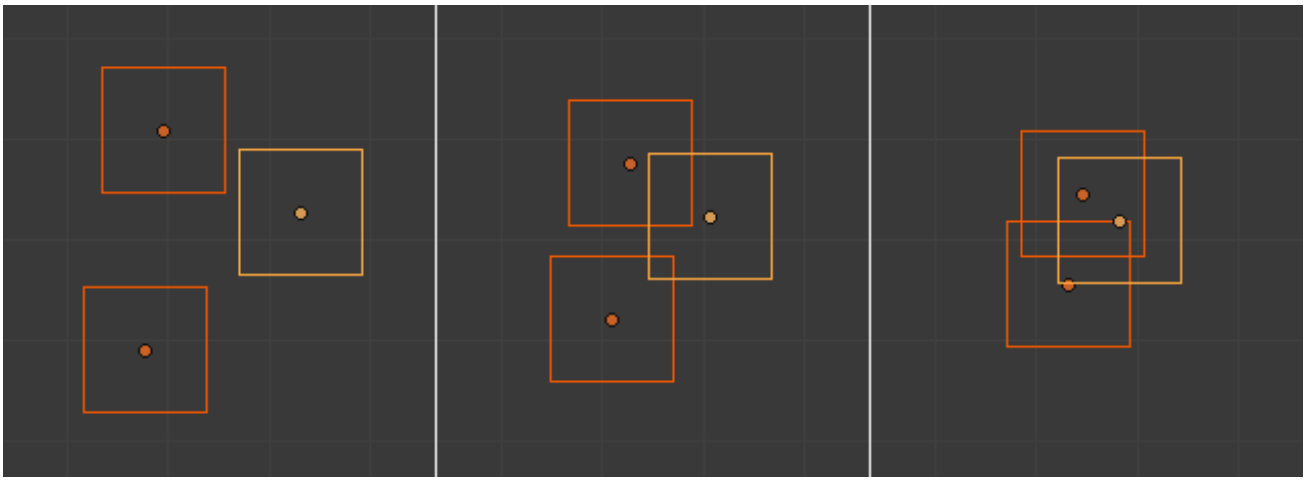


Fig. 2.498: Random Objects being pushed together.

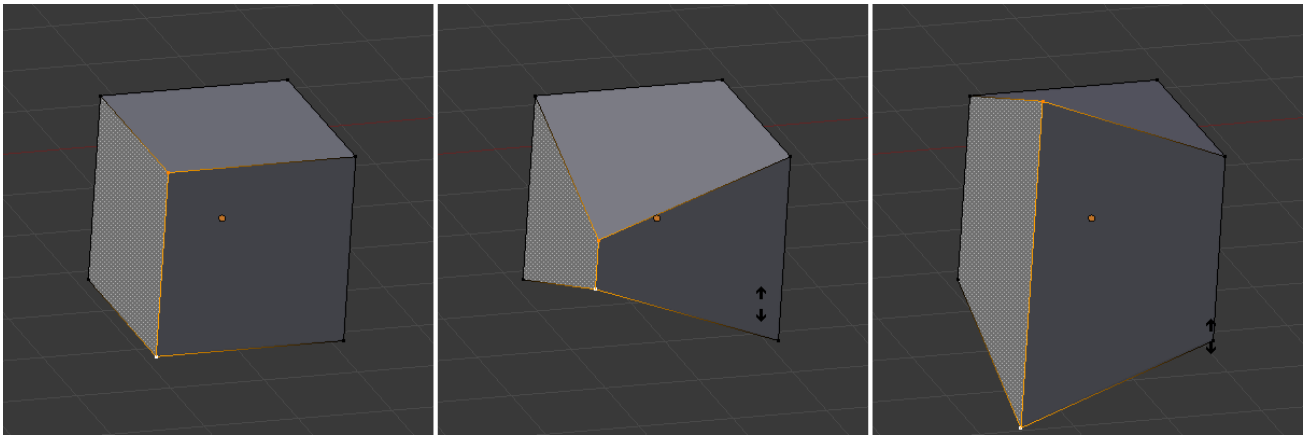


Fig. 2.499: Vertices being pushed together, then pulled apart.

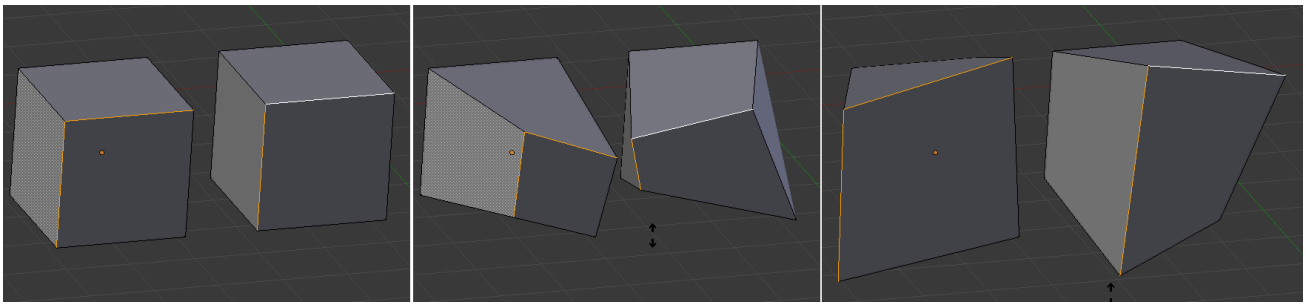


Fig. 2.500: Edges on separate meshes being pushed together, then pulled apart.

Shearing is a form of movement where parallel surfaces move past one another. During this transform, movement of the selected elements will occur along the horizontal axis of the current view. The axis location will be defined by the *Pivot Point*. Everything that is "above" this axis will move (Shear) in the same direction as your mouse pointer (but always parallel to the horizontal axis). Everything that is "below" the horizontal axis will move in the opposite direction.

Fig. 2.501: Shear Offset Factor.

Usage

Select the elements you want to operate on and activate the *Shear* transform function. The *Shear* option can be invoked from the *Object/Mesh/Curve/Surface* → *Transform* → *Shear* menu option or by pressing `Shift-Ctrl-Alt-S`. The amount of movement given to the selection can be determined interactively by moving the mouse or by typing a number. Pressing `Return` will confirm the transformation. The confirmed transformation can be further edited by pressing `F6` or by going into the Tool Shelf and altering the Offset slider provided that no other actions take place between the *Shear* transform confirmation and accessing the slider.

Note that the result of the *Shear* transform is also dependant on the number and type of selected elements (Objects, vertices, faces etc). See below for the result of using *Shear* on a number of different elements.

The three frames of the image above show the effects of shearing on the selected vertices when the pivot point is altered. In frame B, the *Pivot Point* is set to *Median Point* (indicated by the yellow line) and the mouse was moved to the left during the transform. In frame C, the *Pivot Point* is set to the 3D cursor which is located above the mesh (indicated again by the yellow line). When the mouse is moved to the left during a *Shear* transform the selected vertices are moved to the right as they are below the horizontal axis.

Tip: Shear transform magnitude

The magnitude of the *Shear* transform applied to the selected elements is directly proportional to the distance from the horizontal axis. i.e. the further from the axis, the greater the movement.

The three frames of the image above show the effects of shearing on the selected Objects when the *Pivot Point* is altered. In frame B, the *Pivot Point* is set to *Median Point* (indicated by the yellow line) and the mouse was moved to the left during

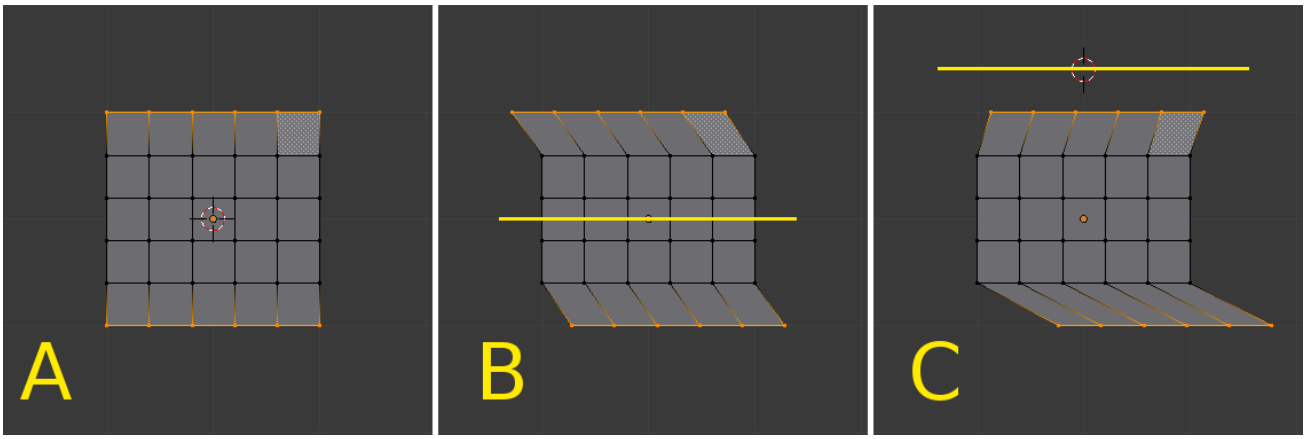


Fig. 2.502: The effects of a Shear transform with different Pivot Points. See the text below for additional information.

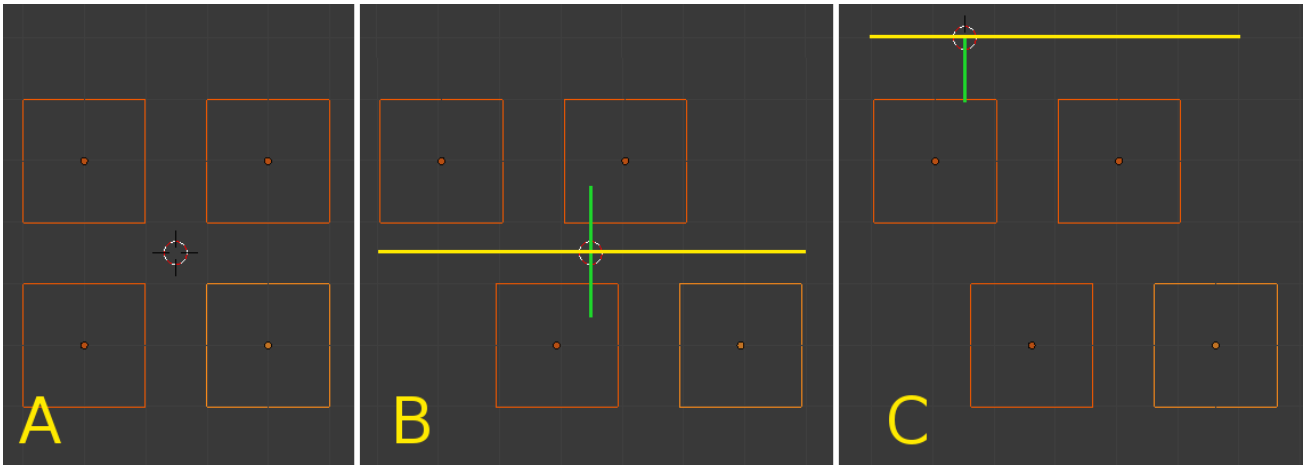


Fig. 2.503: The effects of a Shear transform on Objects with different Pivot Points. See the text below for additional information.

the transform. In frame C, the *Pivot Point* is set to the 3D cursor which is located above the Objects (indicated again by the yellow line). When the mouse is moved to the left during a *Shear* transform all of the selected Objects are moved to the right as they are below the horizontal axis. Again, note that the magnitude of the transform is proportional to the distance from the horizontal axis. In this case, the lower Objects move further than the upper ones.

To Sphere

Reference

Mode: Edit Mode
Menu: *Mesh* → *Transform* → *To Sphere*
Hotkey: `Shift-Alt-S`

The *To Sphere* transformation will give the selection spherical qualities. The Fig. *Monkey with increasing sphericity.* below shows the results of applying the *To Sphere* transformation to the monkey mesh.

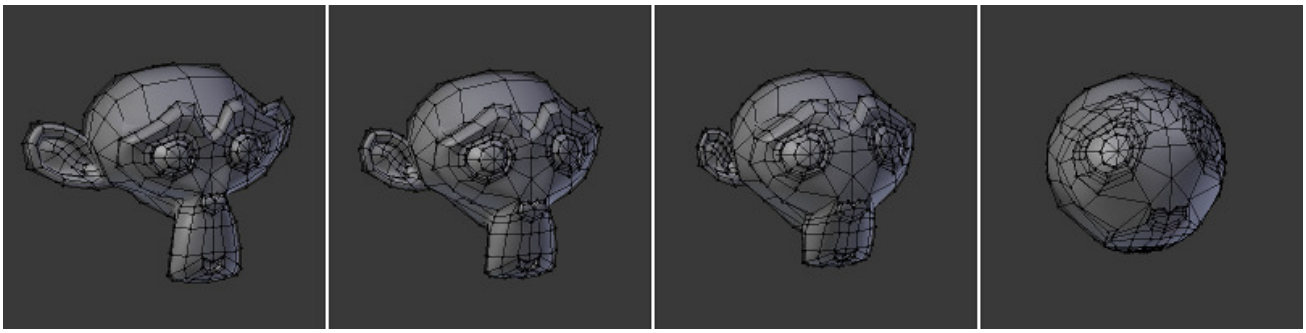


Fig. 2.504: Monkey with increasing sphericity.
The sequence above shows a monkey mesh with a 0, 0.25 (25%), 0.5 (50%) and 1 (100%) To Sphere transform applied.

Usage

Select the elements you want to operate on and activate the *To Sphere* transform function. The *To Sphere* option can be invoked from the *Mesh* → *Transform* → *To Sphere* menu option or by pressing `Shift-Alt-S`. The amount of sphericity given to the selection can be determined interactively by moving the mouse or by typing a number between 0 and 1. Pressing `Return` will confirm the transformation. The confirmed transformation can be further edited by pressing `F6` or by going into the *Tool Shelf* and altering the *Factor* slider provided that no other actions take place between the *To Sphere* transform confirmation and accessing the slider.

Note that the result of the *To Sphere* transform is also dependant on the number of selected mesh elements (vertices, faces etc). As can be seen in the below image, the result will be smoother and more spherical when there are more mesh elements available to work with.

The *To Sphere* transform will generate different results depending on the number and arrangement of elements that were selected (as shown by the below image).

Warp

Reference

Fig. 2.505: To Sphere Factor.

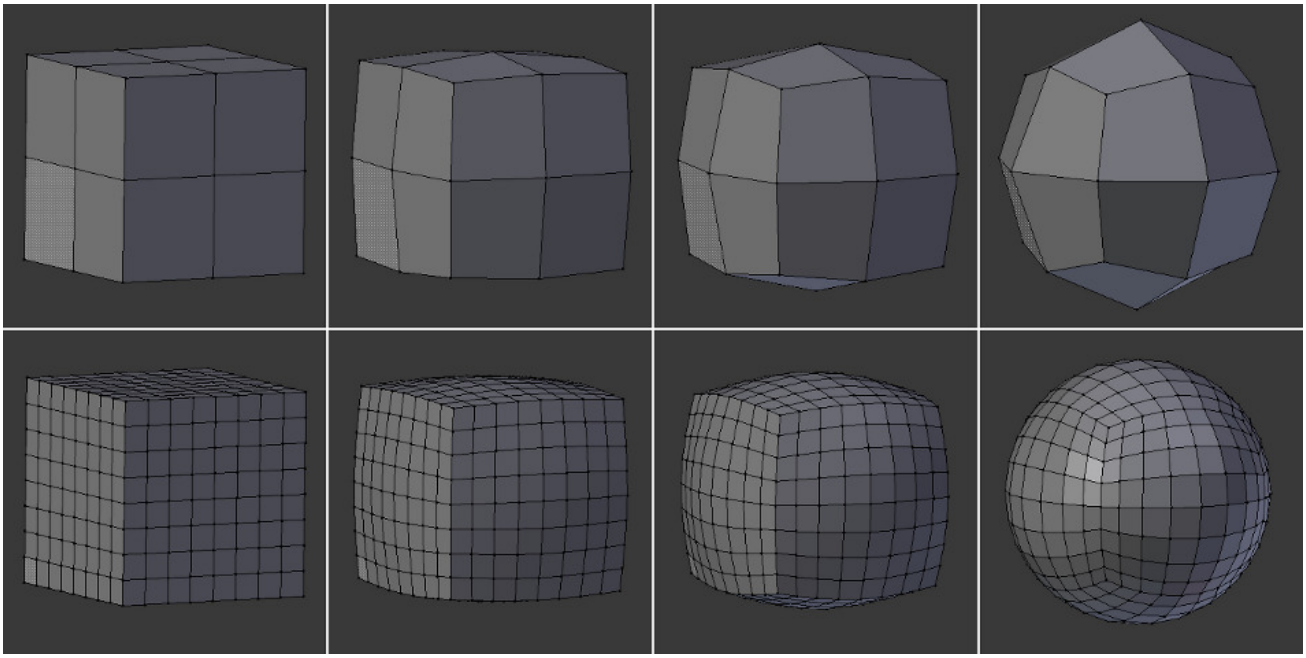


Fig. 2.506: To Sphere applied to cubes with different subdivision levels.
In this image sequence, To Sphere was applied to the entire cube at levels of 0, 0.25 (25%), 0.5 (50%) and 1 (100%) respectively.

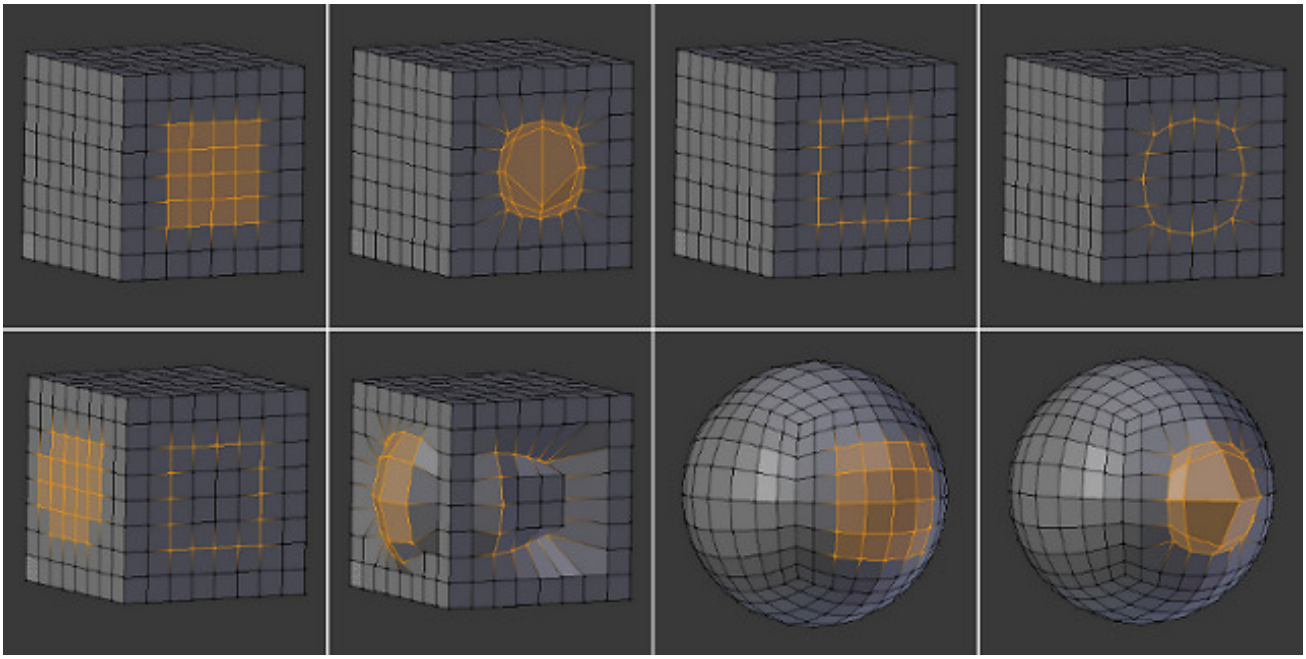


Fig. 2.507: To Sphere applied to different selections.

Mode: Object and Edit Modes
Menu: *Object/Mesh/Curve/Surface* → *Transform* → *Warp*

In *Edit Mode*, the *Warp* transformation takes selected elements and warps them around the 3D cursor by a certain angle. Note that this transformation is always dependent on the location of the 3D cursor. The Pivot Point is not taken into account. The results of the *Warp* transformation are also view dependent.

In *Object Mode*, the *Warp* transformation takes the selected Objects and causes them to move in an orbit-like fashion around the 3D cursor. Similar to *Edit Mode*, the Pivot Point is not taken into account and the results are view dependent.

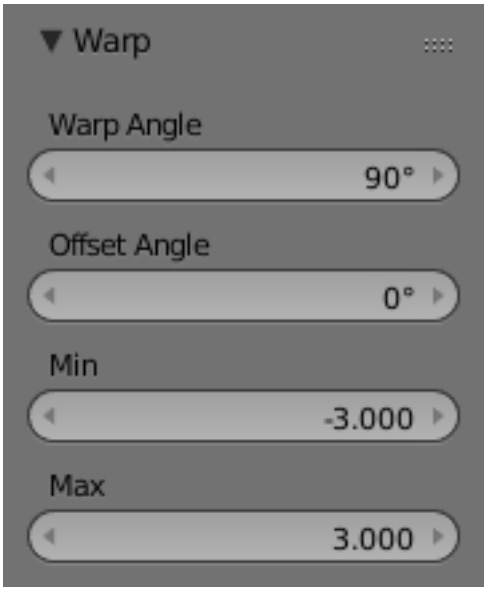


Fig. 2.508: Warp tool options.

Usage

Select the elements you want to operate on and activate the *Warp* transform function. The *Warp* option can be invoked from the *Object/Mesh/Curve/Surface* → *Transform* → *Warp* menu option. The amount of warping given to the selection can be determined interactively by moving the mouse or by typing a number. Pressing `Return` will confirm the transformation. The confirmed transformation can be further edited by pressing `F6` or by going into the Tool Shelf and altering the Angle slider provided that no other actions take place between the *Warp* transform confirmation and accessing the slider.

Cursor position and view

The location of the 3D cursor can be used to alter the results of the *Warp* transformation. As can be seen from the example in this section, the *Warp*

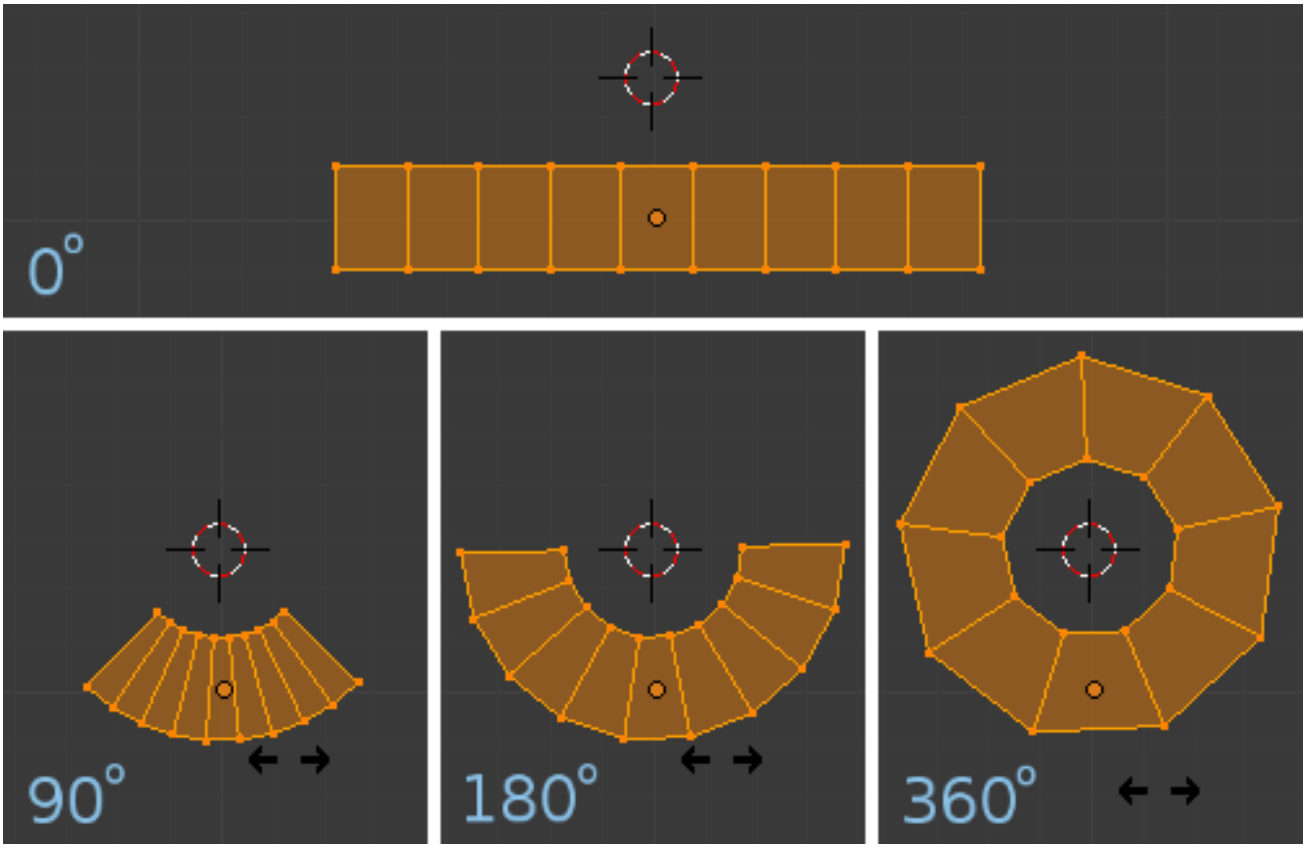


Fig. 2.509: In this example, a plane is warped around the 3D cursor by the indicated number of degrees.

radius is dependent on the distance of the cursor from the selected elements.
The greater the distance, the greater the radius.

The result of the *Warp* transform is also influenced by your current view. The example in this section shows the results of a 180 degree *Warp* transform applied to the same Suzanne mesh when in different views. A 3D render is also provided for comparison.

Note: Warping text

If you want to warp text, you will need to convert it from a Text Object to Mesh by pressing `Alt-C` and selecting the *Mesh from Curve/Meta/Surf/Text* option.

Example

This was made by creating the Blender logo and text as separate Objects. The text was converted to a mesh and then warped around the Blender logo.

Bend

Reference

Mode: Object and Edit Modes
Menu: *Object/Mesh/Curve/Surface* → *Transform* → *Bend*

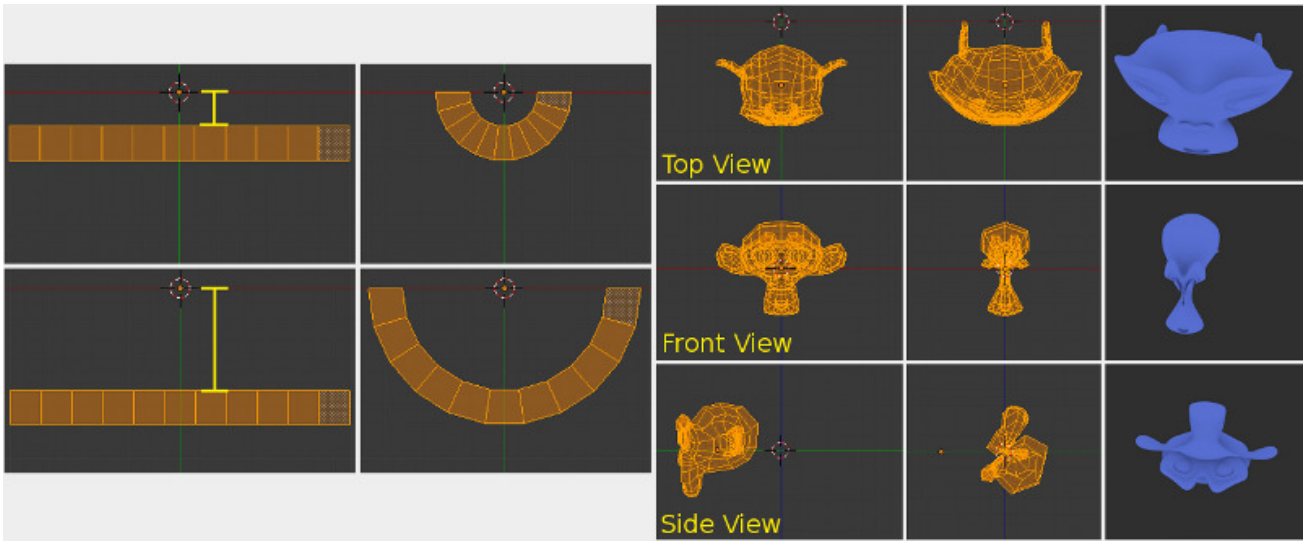


Fig. 2.510: The left side of this image shows how the Warp transform is influenced by the location of the cursor. The right hand side shows the influence of the current view.

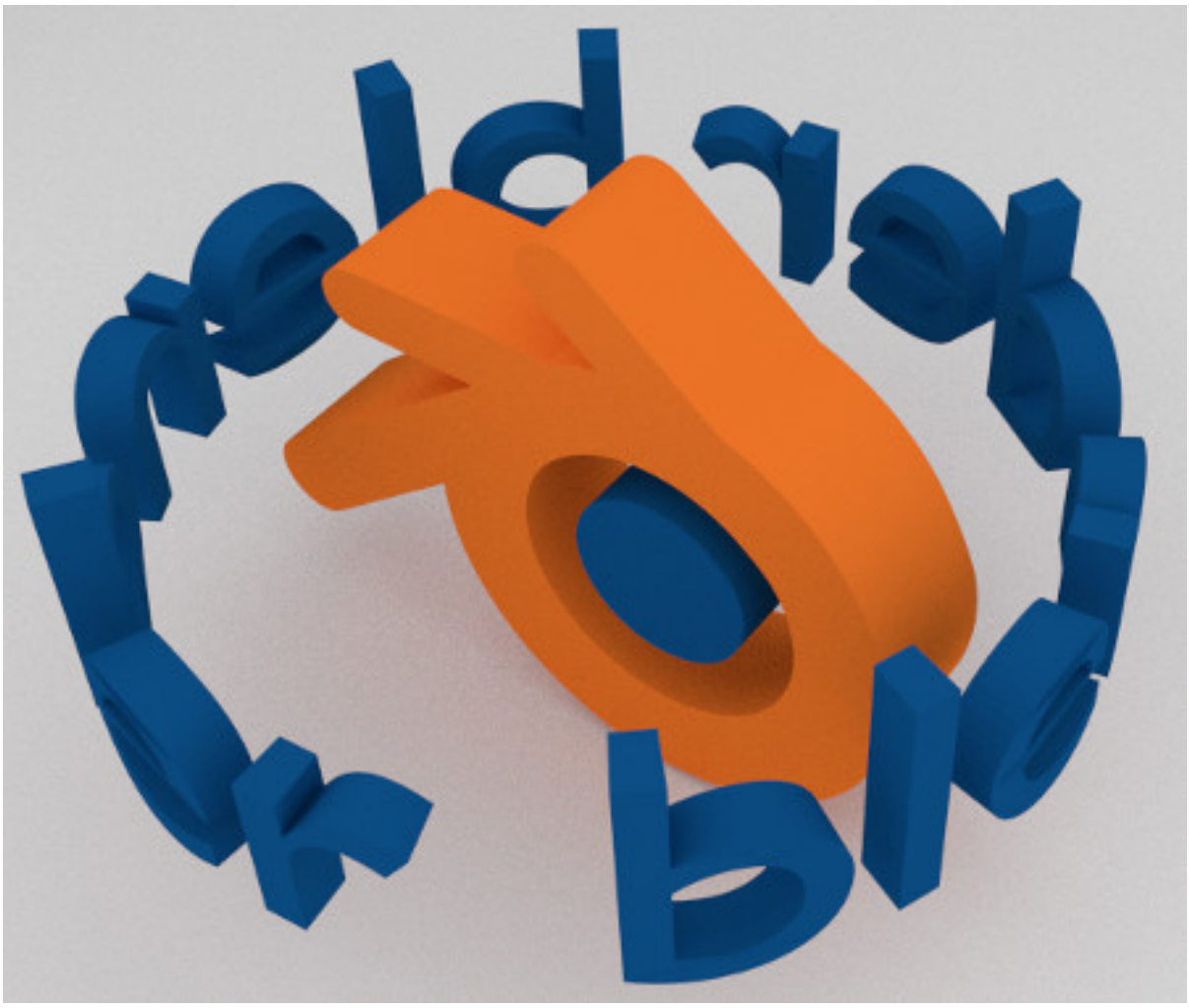


Fig. 2.511: Text wrapped around logo.

Hotkey: `Shift-W`

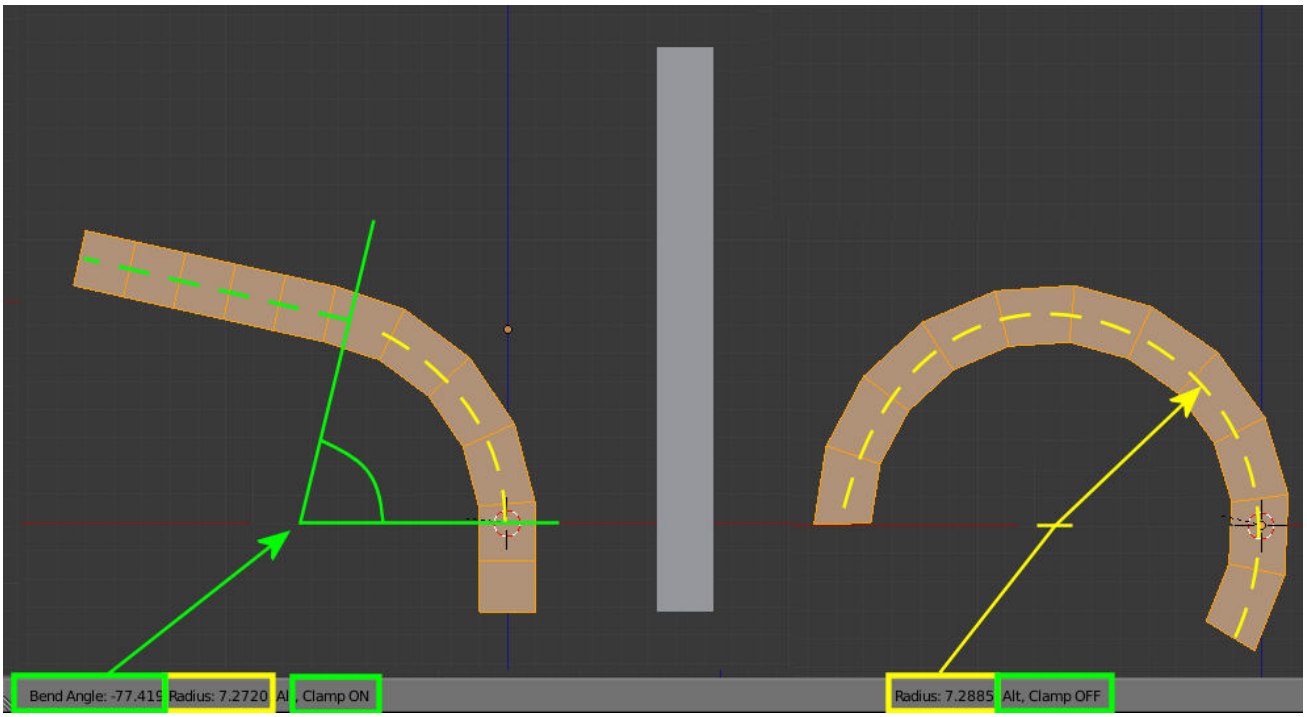


Fig. 2.512: Bend Transform with Clamp on and off.

This tool rotates a line of selected elements forming an arc between the mouse-cursor and the 3D-cursor.

Usage

The bend tool can be used in any case where you might want to bend a shape in two with a gradual transition between both sides.

This may take a little getting used to, the basics are listed below controls are noted here:

- The initial position of the cursors define the axis to bend on.
- The distance of the mouse-cursor to the 3D-cursor controls how sharp the bend will be.
- The relative angle of the mouse-cursor to the initial axis defines the bend angle.

If this seems overly complicated, its probably best to try the tool where it becomes quickly apparent how the tool reacts to your input.

Bend Angle The amount of rotation.

Radius The sharpness of the bend.

Clamp Normally the arc turns through a clamped rotation angle with the selected elements extended along a tangent line beyond that (see above left). When the clamp is deactivated, the arc continues around aligning the selected elements into a circle (right).

When off `Alt` all selected elements follow a circle, even when outside the segment between the 3D cursor and the mouse.

Note: Unlike most other transform modes *Bend* is not effected by *Pivot Point* or *Transform Orientation*, always using the View Plane instead.

Hint: You can turn the bend angle through multiple rotations potentially forming a spiral shape.

Duplicating

Introduction

This section covers mesh editing tools that add additional geometry by duplicating existing geometry in some way.

- *Duplicate Geometry*.
- *Extrusion*.
- *Spin*.
- *Screw*.

Note: Multiple Viewports

When you use one of the duplication tools in the *Mesh Tools* panel, Blender cannot guess which view you want to work in (if you have more than one opened, of course...) As the view is often important for these tools, once you have activated one, your cursor turns into a sort of question mark. Click with it inside the area you want to use.

Duplicate

Reference

Mode: Edit Mode
Menu: *Mesh* → *Duplicate*
Hotkey: `Shift-D`

This tool simply duplicates the selected elements, without creating any links with the rest of the mesh (unlike extrude, for example), and places the duplicate at the location of the original. Once the duplication is done, only the *new* duplicated elements are selected, and you are automatically placed in grab/move mode, so you can translate your copy elsewhere...

In the *Tool Shelf* are settings for *Vector* offset, *Proportional Editing*, *Duplication Mode* (non-functional?), and *Axis Constraints*.

Note that duplicated elements belong to the same *vertex groups* as the "original" ones. The same goes for the *material indices*, the edge's *Sharp* and *Seam* flags, and probably for the other vertex/edge/face properties...

Extrude

Extrude Region

Reference

Mode: Edit Mode
Panel: *Mesh Tools* → *Extrude*

Menu: *Mesh* → *Extrude Region*
Hotkey: `E` or `Alt-E`

One tool of paramount importance for working with meshes is the *Extrude* tool. It allows you to create parallelepipeds from rectangles and cylinders from circles, as well as easily create such things as tree limbs. *Extrude* is one of the most frequently used modeling tools in Blender. It is simple, straightforward, and easy to use, yet very powerful.

The selection is extruded along the common normal of selected faces. In every other case the extrusion can be limited to a single axis by specifying an axis (e.g. `X` to limit to the X axis or `Shift-X` to the YZ plane. When extruding along the face normal, limiting movement to the global Z axis requires pressing `Z` twice, once to disable the face normal Z axis limit, and once to enable the global Z axis limit.

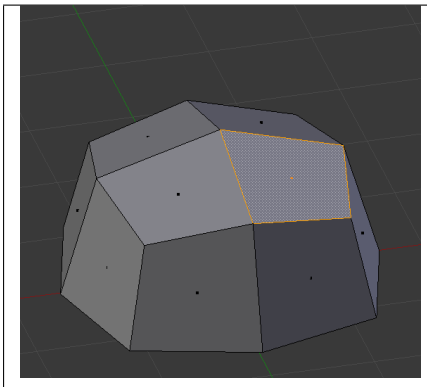


Fig. 2.513: Selected face.

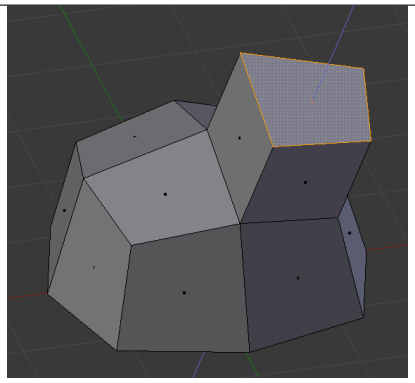


Fig. 2.514: During extrude.

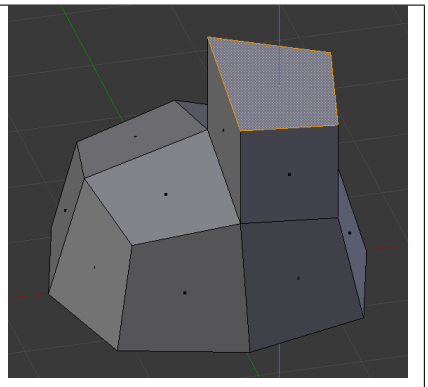


Fig. 2.515: Set to Z axis.

Although the process is quite intuitive, the principles behind *Extrude* are fairly elaborate as discussed below:

- First, the algorithm determines the outside edge-loop of the extrude; that is, which among the selected edges will be changed into faces. By default (see below), the algorithm considers edges belonging to two or more selected faces as internal, and hence not part of the loop.
- The edges in the edge-loop are then changed into faces.
- If the edges in the edge-loop belong to only one face in the complete mesh, then all of the selected faces are duplicated and linked to the newly created faces. For example, rectangles will result in parallelepipeds during this stage.
- In other cases, the selected faces are linked to the newly created faces but not duplicated. This prevents undesired faces from being retained "inside" the resulting mesh. This distinction is extremely important since it ensures the construction of consistently coherent, closed volumes at all times when using *Extrude*.
- When extruding completely closed volumes (like e.g. a cube with all its six faces), extrusion results merely in a duplication, as the volume is duplicated, without any link to the original one.
- Edges not belonging to selected faces, which form an "open" edge-loop, are duplicated and a new face is created between the new edge and the original one.
- Single selected vertices which do not belong to selected edges are duplicated and a new edge is created between the two.

Extrude Individual

Reference

Mode: Edit Mode
Panel: *Mesh Tools* → *Extrude Individual*
Menu: *Mesh* → *Extrude Individual*

Hotkey: `Alt-E`

Extrude Individual allows you to extrude a selection of multiple faces as individuals, instead of as a region. The faces are extruded along their own normals, rather than their average. This has several consequences: first, "internal" edges (i.e. edges between two selected faces) are no longer deleted (the original faces are).

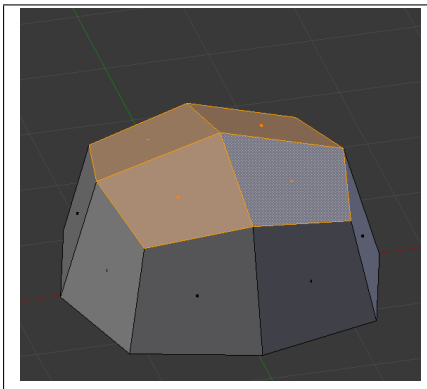


Fig. 2.516: Selection of multiple faces.

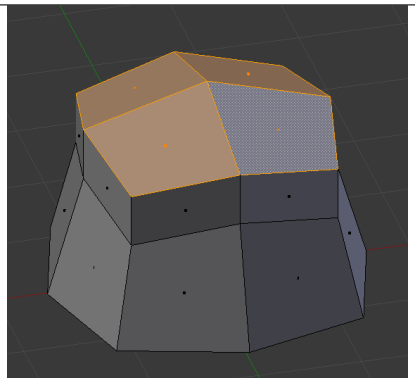


Fig. 2.517: Extruded using extrude region.

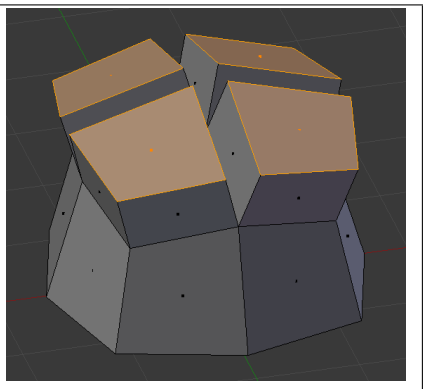


Fig. 2.518: Extruded using Extrude Individual.

Extrude Edges and Vertices Only

Reference

Mode: Edit Mode, Vertex and Edge
Hotkey: `Alt-E`

If vertices are selected while doing an extrude, but they do not form an edge or face, they will extrude as expected, forming a *non-manifold* edge. Similarly, if edges are selected that do not form a face, they will extrude to form a face.

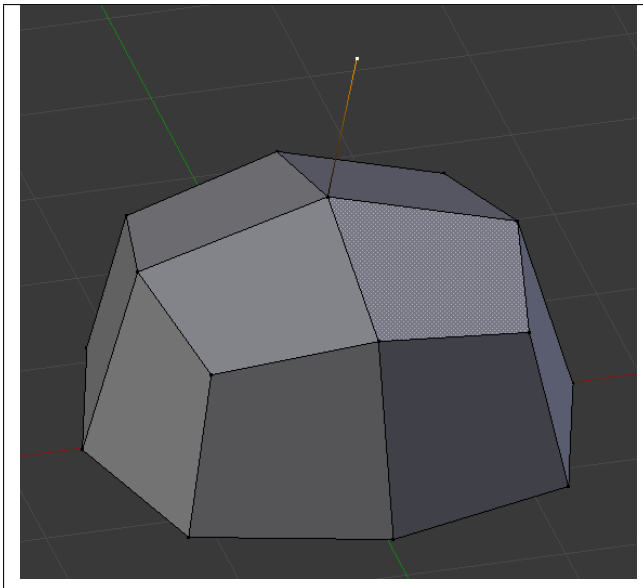


Fig. 2.519: Single vertex extruded.

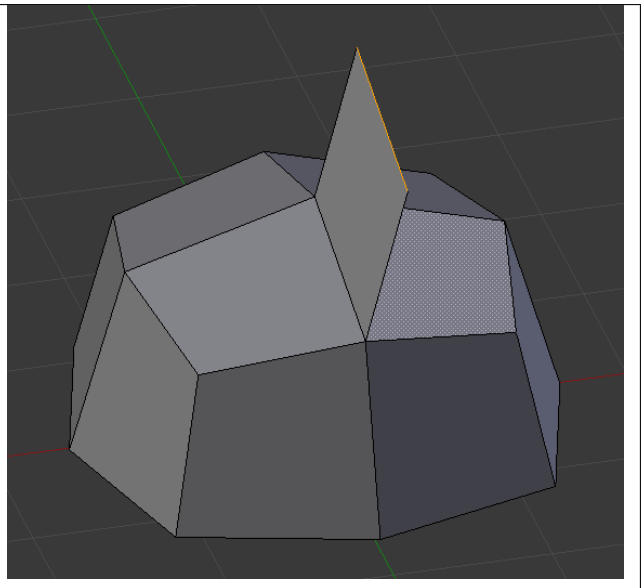


Fig. 2.520: Single edge extruded.

When a selection of vertices forms an edge or face, it will extrude as if the edge was selected. Likewise for edges that form a face.

To force a vertex or edge selection to extrude as a vertex or edge, respectively, use `Alt-E` to access the Extrude *Edges Only* and *Vertices Only*.

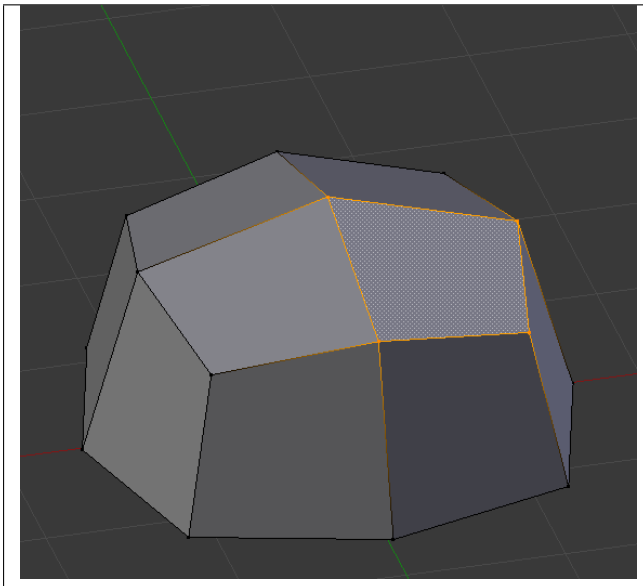


Fig. 2.521: Vertex selected.

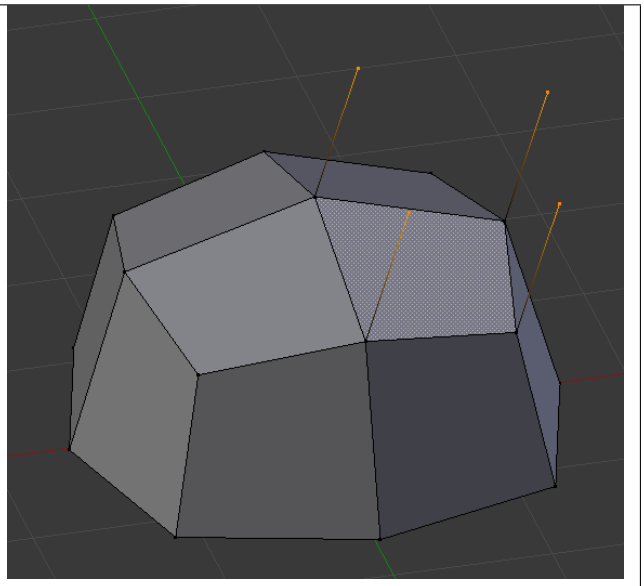


Fig. 2.522: Vertices Only extrude.

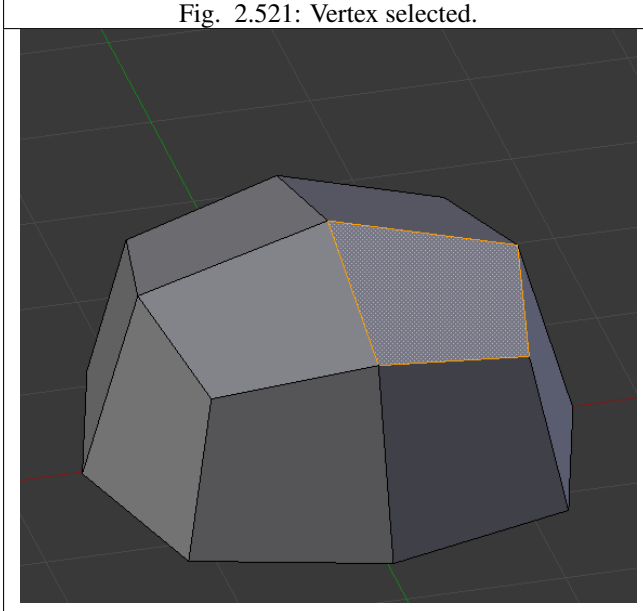


Fig. 2.523: Edge selected.

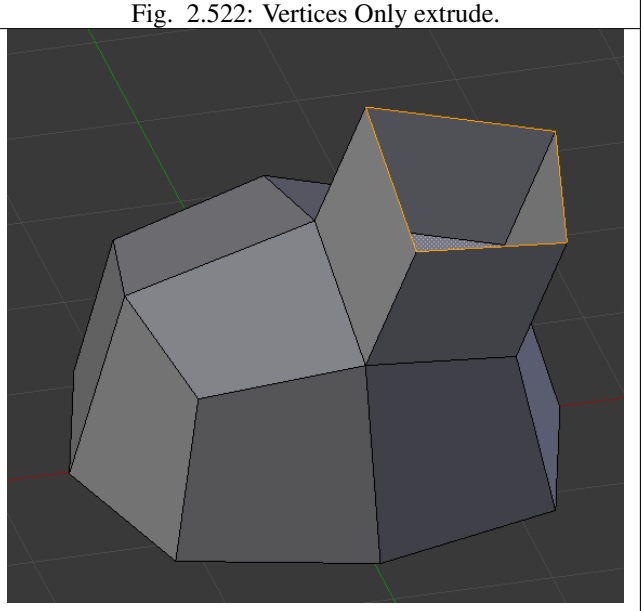


Fig. 2.524: Edge Only extrude.

Inset

Reference

Mode: Edit Mode
Menu: *Mesh* → *Faces* → *Inset*
Hotkey: `I`

This tool takes the currently selected faces and creates an inset of them, with adjustable thickness and depth. The tool is modal, such that when you activate it, you may adjust the thickness with your mouse position. You may also adjust the depth of the inset during the modal operation by holding `Ctrl`.

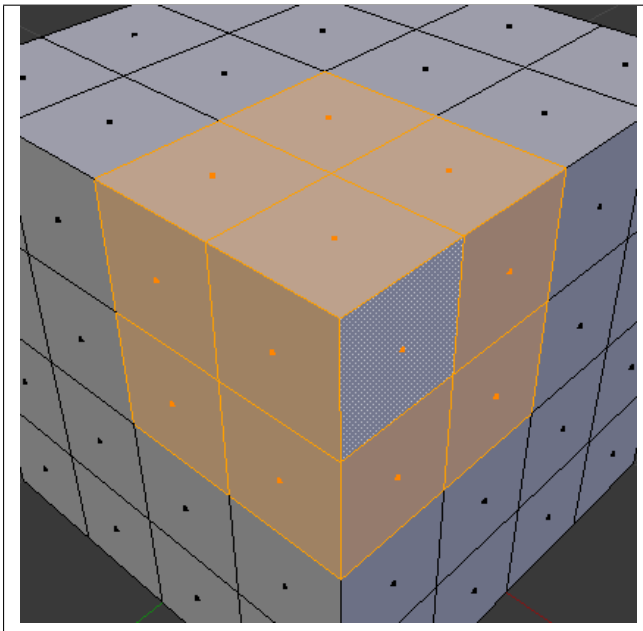


Fig. 2.525: Selection to inset.

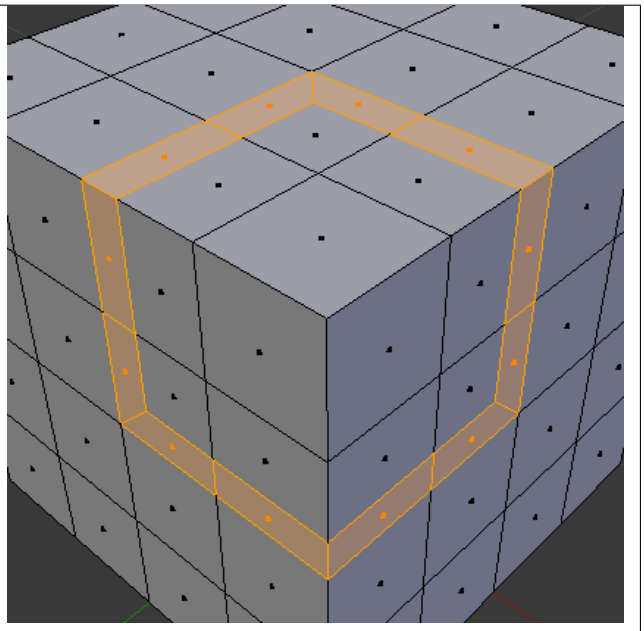


Fig. 2.526: Selection with inset.

Options

Boundary Determines whether open edges will be inset or not.

Offset Even Scale the offset to give more even thickness.

Offset Relative Scale the offset by surrounding geometry.

Thickness Set the size of the inset.

Depth Raise or lower the newly inset faces to add depth.

Outset Create an outset rather than an inset.

Select Outer Toggle which side of the inset is selected after operation.

Spin

Reference

Mode: Edit Mode
Panel: Mesh Tools

Use the *Spin* tool to create the sort of objects that you would produce on a lathe (this tool is often called a "lathe"-tool or a "sweep"-tool in the literature, for this reason). In fact, it does a sort of circular extrusion of your selected elements, centered on the 3D cursor, and around the axis perpendicular to the working view...

- The point of view will determine around which axis the extrusion spins...

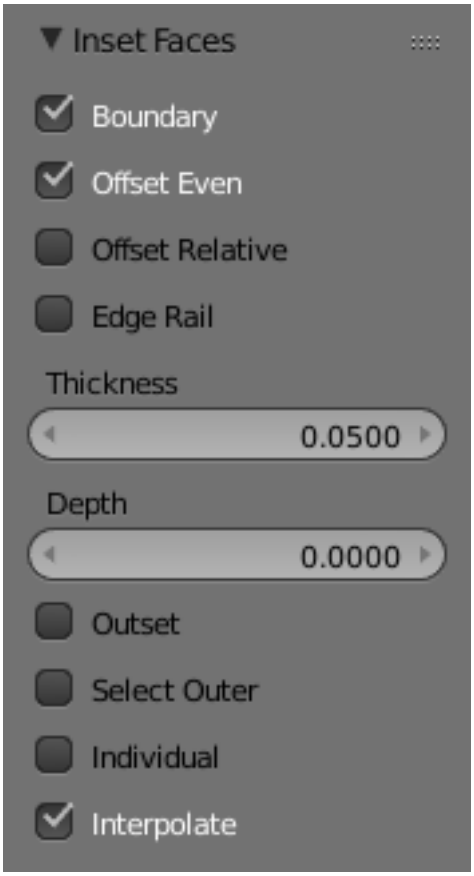


Fig. 2.527: Inset Operator Settings.

- The position of the 3D cursor will be the center of the rotation.

Here are its settings:

Steps Specifies how many copies will be extruded along the "sweep".

Dupli When enabled, will keep the original selected elements as separated islands in the mesh (i.e. unlinked to the result of the spin extrusion).

Angle specifies the angle "swept" by this tool, in degrees (e.g. set it to 180 for half a turn).

Center Specifies the center of the spin. By default it uses the cursor position.

Axis Specify the spin axis as a vector. By default it uses the view axis.

Example

First, create a mesh representing the profile of your object. If you are modeling a hollow object, it is a good idea to thicken the outline. Fig. *Glass profile.* shows the profile for a wine glass we will model as a demonstration.

Go to the *Edit Mode* and select all the vertices of the Profile with `A`.

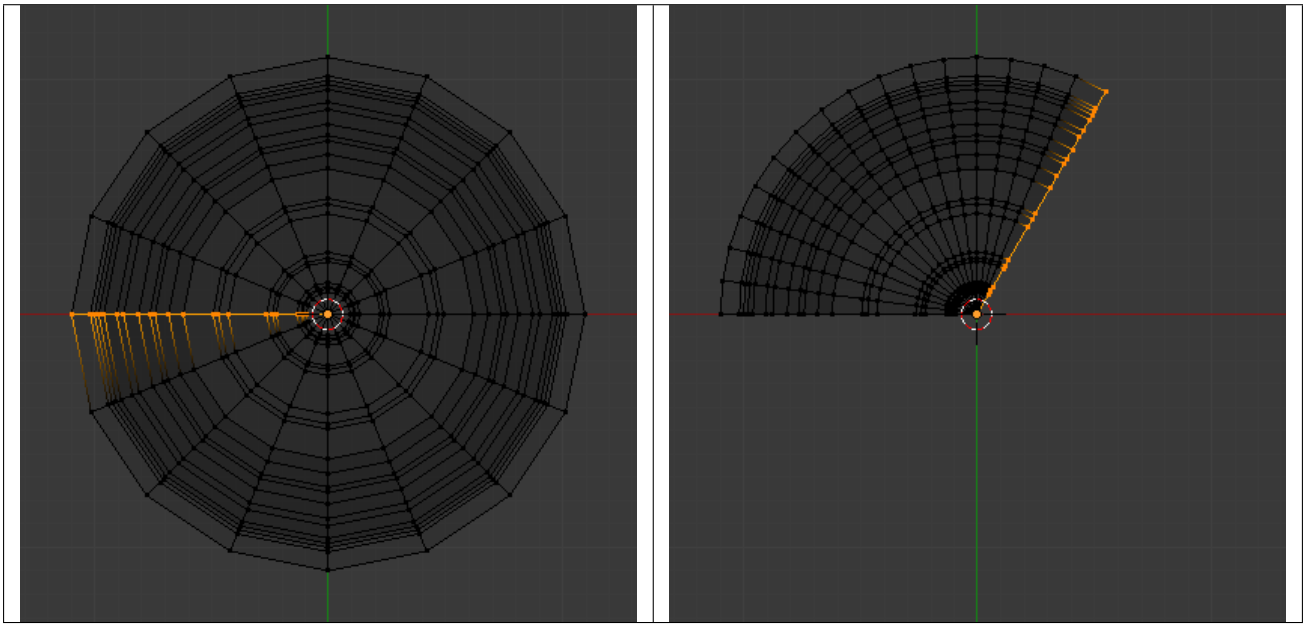
We will be rotating the object around the cursor in the top view, so switch to the top view with `Numpad7`.

Place the cursor along the center of the profile by selecting one of the vertices along the center, and snapping the 3D cursor to that location with *Mesh* → *Cursor* → *Selection*. (Fig. *Glass profile, top view in Edit Mode, just before spinning.*) shows the wine glass profile from top view, with the cursor correctly positioned.

Click the *Spin* button. If you have more than one 3D View open, the cursor will change to an arrow with a question mark and you will have to click in the area containing the top view before continuing. If you have only one 3D View open, the spin will happen immediately. Fig. *Spun profile.* shows the result of a successful spin.

Angle

Table 2.20: Spun profile using an angle of 120.



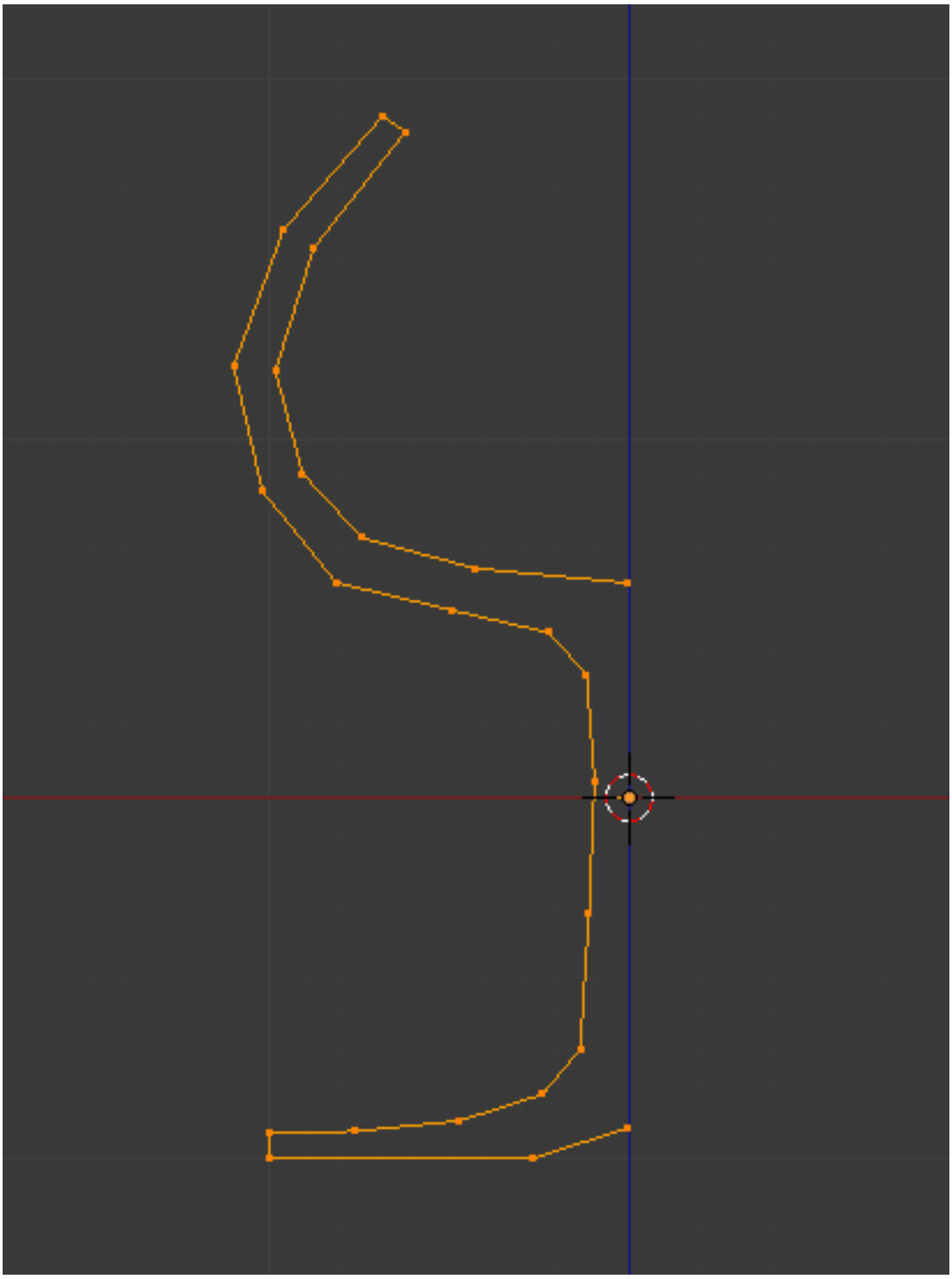


Fig. 2.528: Glass profile.

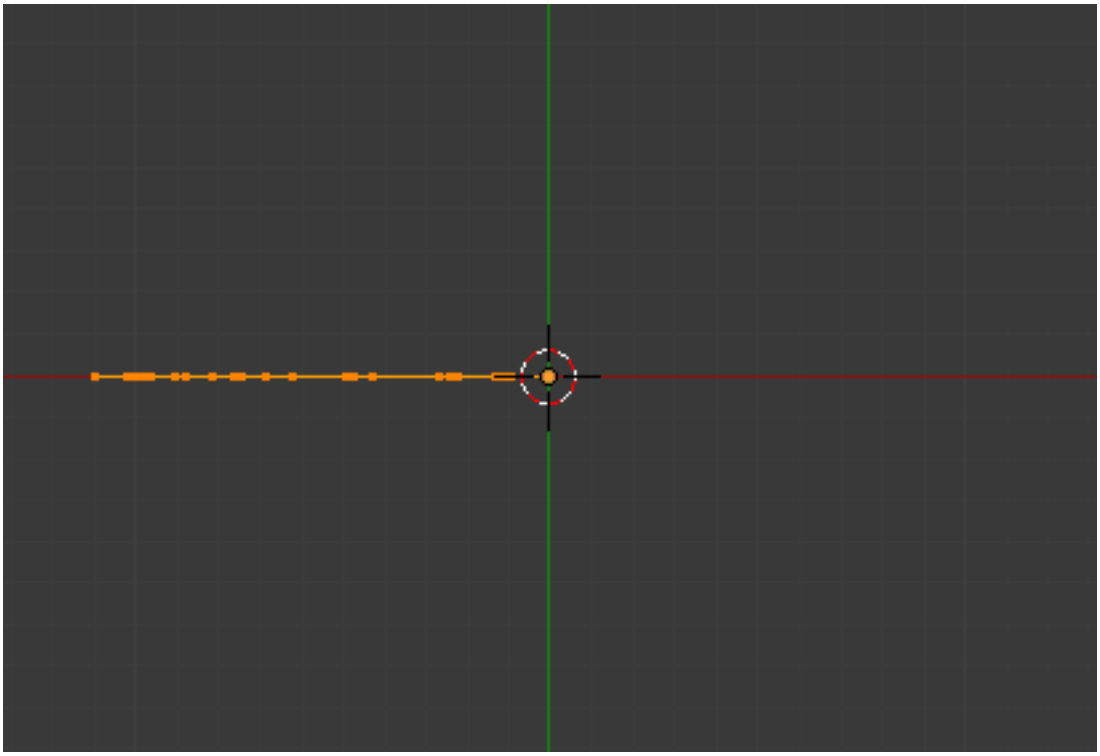


Fig. 2.529: Glass profile, top view in Edit Mode, just before spinning.

Dupli

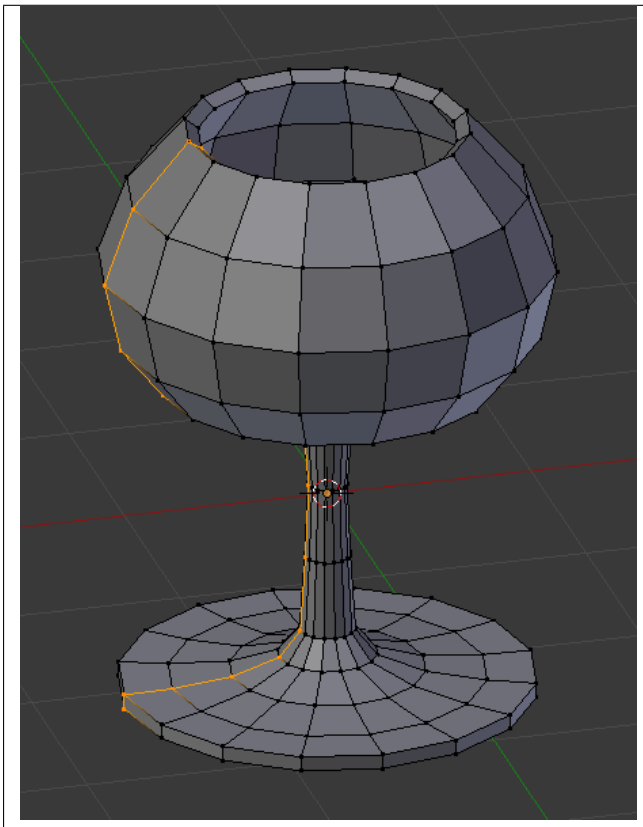


Fig. 2.532: Result of spin operation.

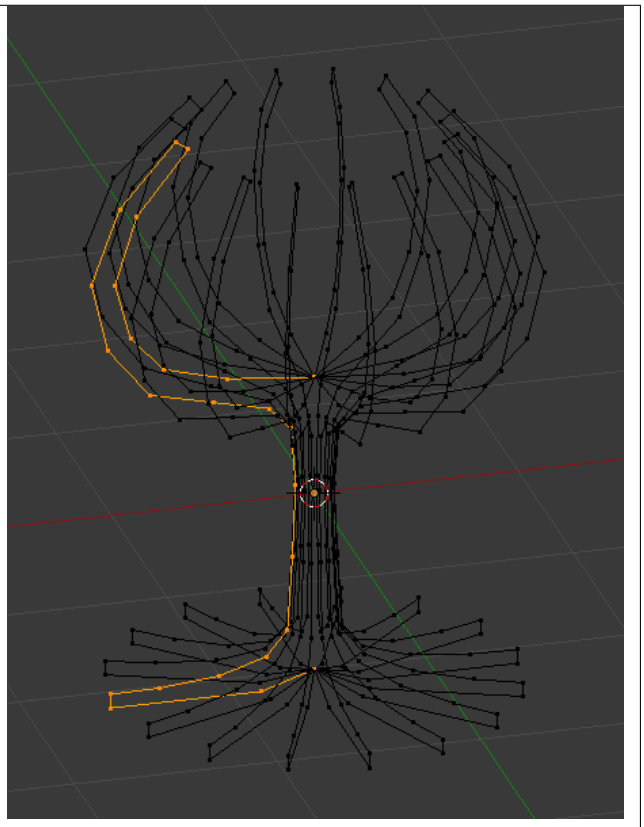


Fig. 2.533: Result of Dupli enabled.

Merge Duplicates

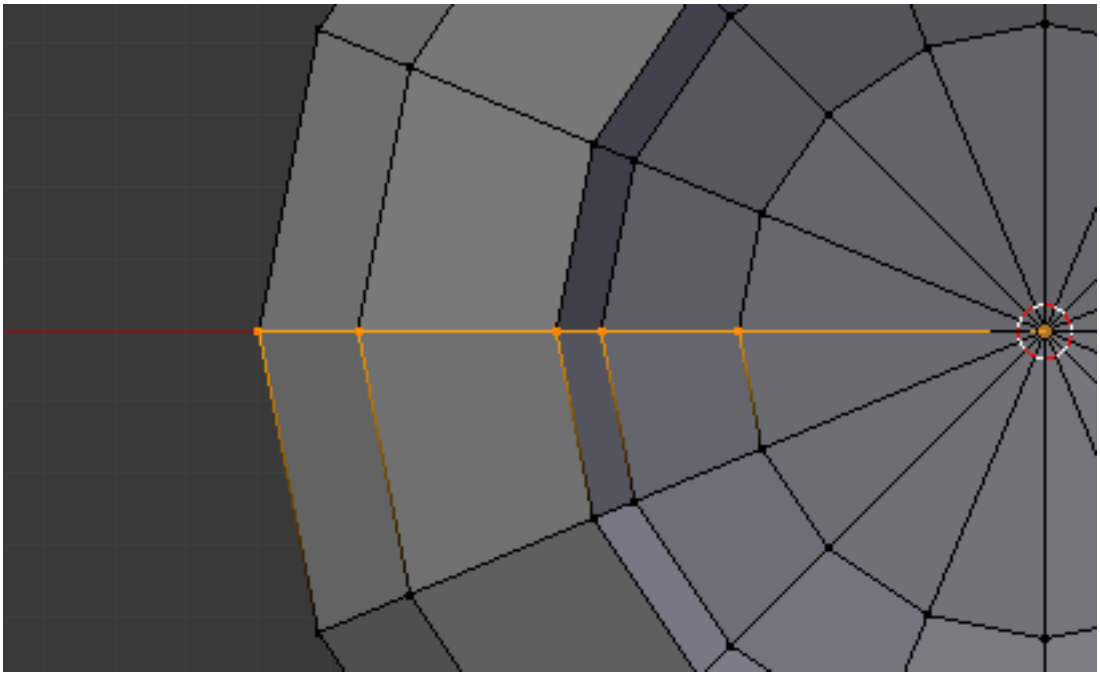


Fig. 2.534: Duplicate vertices.

The spin operation leaves duplicate vertices along the profile. You can select all vertices at the seam with Box select `B` shown in Fig. *Duplicate vertices.* Seam vertex selection and perform a *Remove Doubles* operation.

Notice the selected vertex count before and after the *Remove Doubles* operation `Vertex count after removing doubles`. If all goes well, the final vertex count (38 in this example) should match the number of the original profile noted in *Mesh data* → *Vertex and face numbers*. If not, some vertices were missed and you will need to weld them manually. Or, worse, too many vertices will have been merged.

Note: Merging two vertices in one

To merge (weld) two vertices together, select both of them by `Shift-RMB` clicking on them. Press `S` to start scaling and hold down `Ctrl` while scaling to scale the points down to 0 units in the X, Y and Z axis. `LMB` to complete the scaling operation and click the *Remove Doubles* button in the Tool shelf in *Edit Mode* (also available with `W` → *Remove Doubles*).

Alternatively, you can use `W` → *Merge* from the same *Specials* menu (or `Alt-M`). Then, in the new pop-up menu, choose whether the merged vertex will be at the center of the selected vertices or at the 3D cursor. The first choice is better in our case!

Recalculate Normals

All that remains now is to recalculate the normals to the outside by selecting all vertices, pressing `Ctrl-N` and validating *Recalc Normals Outside* in the pop-up menu.

Screw Tool

Reference

Mode: Edit Mode

Panel: *Edit Mode* → *Mesh Tools* or T → *Add* → *Screw Button*

---

he *Screw* tool combines a repetitive *Spin* with a translation, to generate a screw-like, or spiral-shaped, object. Use this tool to create screws, springs, or shell-shaped structures (Sea shells, Wood Screw Tips, Special profiles, etc).

The main difference between the Screw Tool and the *Screw Modifier* is that the Screw Tool can calculate the angular progressions using the basic profile angle automatically. Or it can adjusting the Axis angular vector without using a second modifier (for example, using the Screw Modifier with a Bevel Modifier, Curve Modifier, etc...), resulting in a much cleaner approach for vertex distribution and usage.

This tool works using open or closed profiles, as well as profiles closed with faces. You can use profiles like an open-edge part that is a part of a complete piece, as well as a closed circle or a half-cut sphere, which will also close the profile end.

You can see some examples of Meshes generated with the *Screw* tool in Fig. *Wood Screw tip done with the screw tool.* and Fig. *Spring done with the screw tool.*.

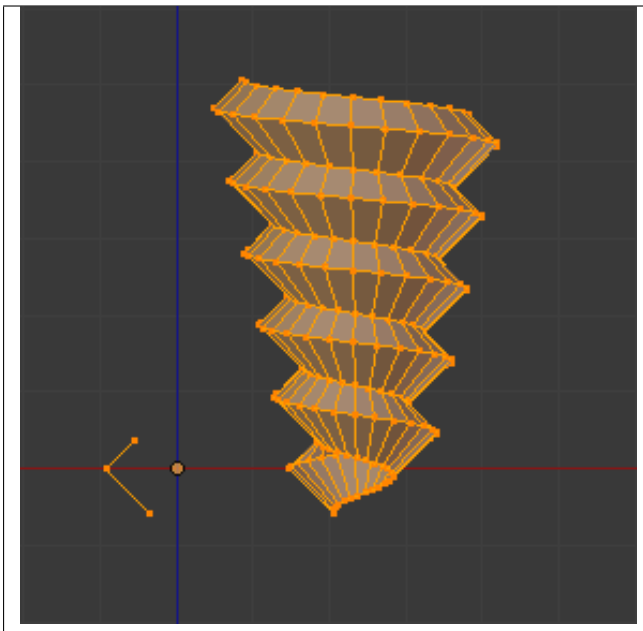


Fig. 2.535: Wood Screw tip done with the screw tool.

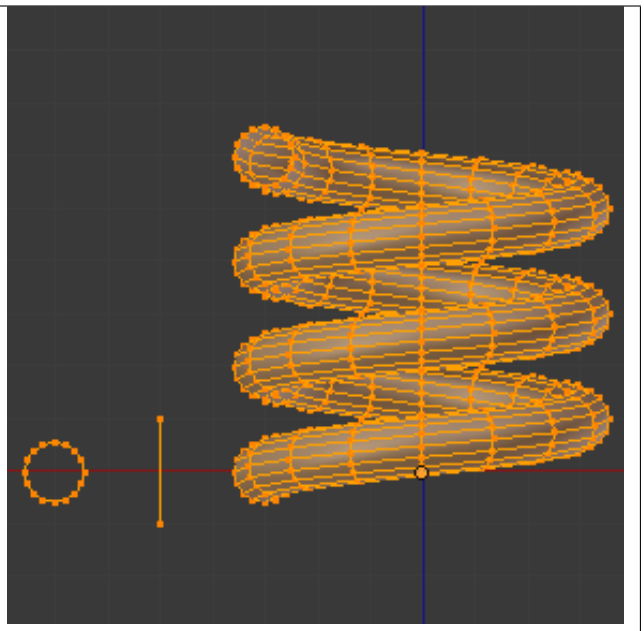


Fig. 2.536: Spring done with the screw tool.

Usage

This tool works only with Meshes. In *Edit Mode*, the button for the *Screw* tool operation is located in the *Mesh Tools* Panel, T → *Add* → *Screw Button*. To use this tool, you need to create at least one open profile or line to be used as a vector for the height, angular vector and to give Blender a direction.

The *Screw* function uses two points given by the open line to create an initial vector to calculate the height and basic angle of the translation vector that is added to the "Spin" for each full rotation (see examples below). If the vector is created with only two vertices at the *same* (X, Y, Z) location (which will not give Blender a vector value for height), this will create a normal "Spin".

Having at least one vector line, you can add other closed support profiles that will follow this vector during the extrusions (See limitations). The direction of the extrusions is calculated by two determinant factors, your point of view in Global Space and the position of your cursor in the 3D View using Global coordinates. The profile and the vector must be fully selected in *Edit Mode* before you click the *Screw Button* (See Limitations.) When you have the vector for the open profile and the other closed profiles selected, click the *Screw* Button.

Limitations

There are strict conditions about your profile selection when you want to use this tool. You must have at least one open line or open profile, giving Blender the starting Vector for extrusion, angular vector and height. (e.g. a simple edge, a half circle, etc...). You need only to ensure that at least one reference line has two "free" ends. If two open Lines are given, Blender will not determine which of them is the vector, and will then show you an error message, "`You have to select a string of connected vertices too`". You need to select all of the profile vertices that will participate in the *Screw* Tool operation; if they are not properly selected, Blender will also show you the same message.

Note that the open line is always extruded, so if you only use it to "guide" the screw, you will have to delete it after the tool completion (use linked-selection, `Ctrl-L`, to select the whole extrusion of the open line).

If there is any problem with the selection or profiles, the tool will warn you with the error message: "`You have to select a string of connected vertices too`" as seen in Fig. *Screw Error message in the Header of the Info editor.* and Fig. *Error message when clicking in the Screw Tool with an incorrect or bad selection.*, both in the Info Editor and at the place where you clicked to start performing the operation (when you click the Screw Button).

Fig. 2.537: Screw Error message in the Header of the Info editor.

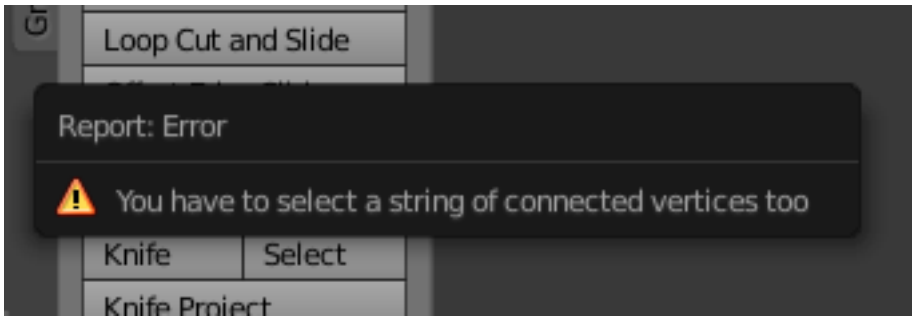


Fig. 2.538: Error message when clicking in the Screw Tool with an incorrect or bad selection.

You may have as many profiles as you like (like circles, squares, and so on) – Note that not all vertices in a profile need to be in the same plane, even if this is the most common case. You may also have other, more complex, selected closed islands, but they have to be closed profiles because Blender will seek for only one open profile for the translation, height and angular vector. Some closed meshes that overlap themselves may not screw correctly (for example: Half UV-sphere works fine, but more than half could cause the Screw Tool to have wrong behavior or errors), and profiles that are closed with faces (like a cone or half sphere) will be closed automatically at their ends, like if you were extruding a region.

Tip: Simple way to not result in error

Only one open Profile, all of the others can be closed, avoid volumes and some profiles closed with faces...

Options

This tool is an interactive and modal tool, and only works in the *Edit Mode*.

Once you click in the *Screw* tool in the Mesh Tools Panel, Blender will enter in the *Screw* interactive mode, and the Operator Panel at the end of the Mesh Tools Panel will be replaced so you can adjust the values explained below. To show the Mesh Tools Panel, use the shortcut `T` in the Edit Mode of the 3D View editor.

Once you perform any other operation, Blender leaves the interactive mode and accepts all of the values. Because it is modal, you cannot return to the interactive mode after completing/leaving the operation or changing from *Edit Mode* to *Object Mode*. If you want to restart the operation from its beginning, you can press `Ctrl-Z` at any time in *Edit Mode*.

The basic location of the cursor at the point of view (using Global coordinates) will determine around which axis the selection is extruded and spun at first (See Fig. *Properties region → Cursor.*). Blender will copy your cursor location coordinates to the values present in the *Center* values of the *Screw* interactive Panel. Depending on the Global View position, Blender will automatically add a value of 1 to one of the Axis Vectors, giving the profiles a starting direction for the Screw Operation and also giving a direction for the extrusions. (See examples below.)

The position of the 3D cursor will be the starting center of the rotation. Subsequent operations (e.g. pressing the Screw button again), will start from the last selected element. Continuous operations without changing the selection will repeat the operation continuously from the last point.

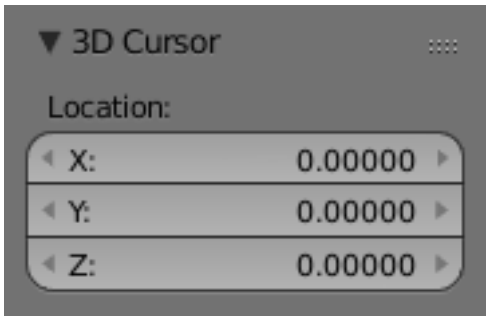


Fig. 2.539: *Properties region → Cursor*.

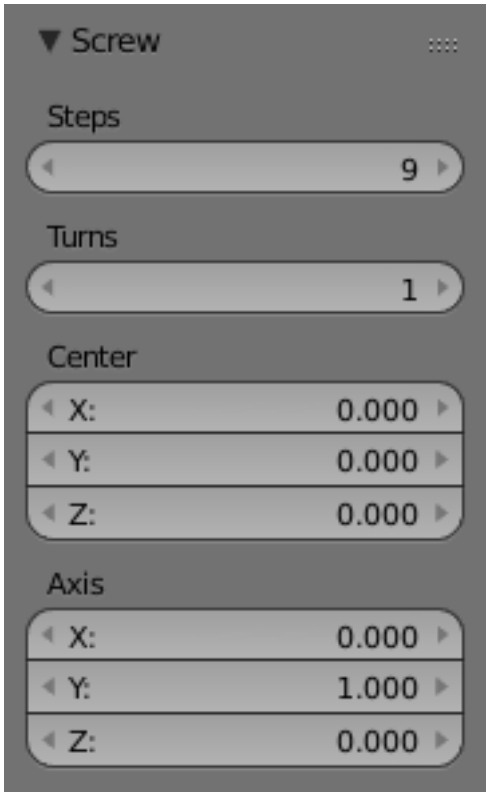


Fig. 2.540: Screw Tools Operator Panel (Edit Mode).

Center These numeric fields specify the center of the spin. When the tool is called for the first time, it will copy the (X, Y, Z) location (Global Coordinates) of the cursor presently in the 3D View to start the operation. You can specify the cursor coordinates using the Transform Panel in 3D View, using shortcut `T` to toggle the Panel, and typing in the 3D Cursor Location coordinates. You can adjust these coordinates interactively and specify another place for the spin center during the interactive session. (See Fig. *Screw Tools Operator Panel (Edit Mode).*)

Steps This number button specifies how many extrusion(s) will be done for each 360 degree turn. The steps are evenly distributed by dividing 360 degree by the number of steps given. The minimum value is 3; the maximum is 256 (See

Fig. *Screw Tools Operator Panel (Edit Mode).*)

Turns This number button specifies how many turns will be executed. Blender will add a new full 360 degree turn for each incremental number specified here. The minimum value is 1; the maximum is 256. (See Fig. *Screw Tools Operator Panel (Edit Mode).*)

Axis These three numeric fields vary from (-1.0 to 1.0) and are clamped above those limits. These values correspond to angular vectors from (-90 to 90) degrees. Depending on the position where you started your cursor location and Object operation in the viewport and its axis positions in Global View space and coordinates, Blender will give the proper Axis vector a value of 1, giving the angular vector of the profile a starting direction and giving the extrusions a starting direction based on your view. Blender will let you adjust your axis angular vectors and you can tweak your object such that you can revert the direction of the screw operation (by reverting the angular vector of the height), meaning you can revert the clockwise and counterclockwise direction of some operations, and also adjust the angular vectors of your profile, bending it accordingly. (See Fig. *Screw Tools Operator Panel (Edit Mode).*)

Examples

The Spring example

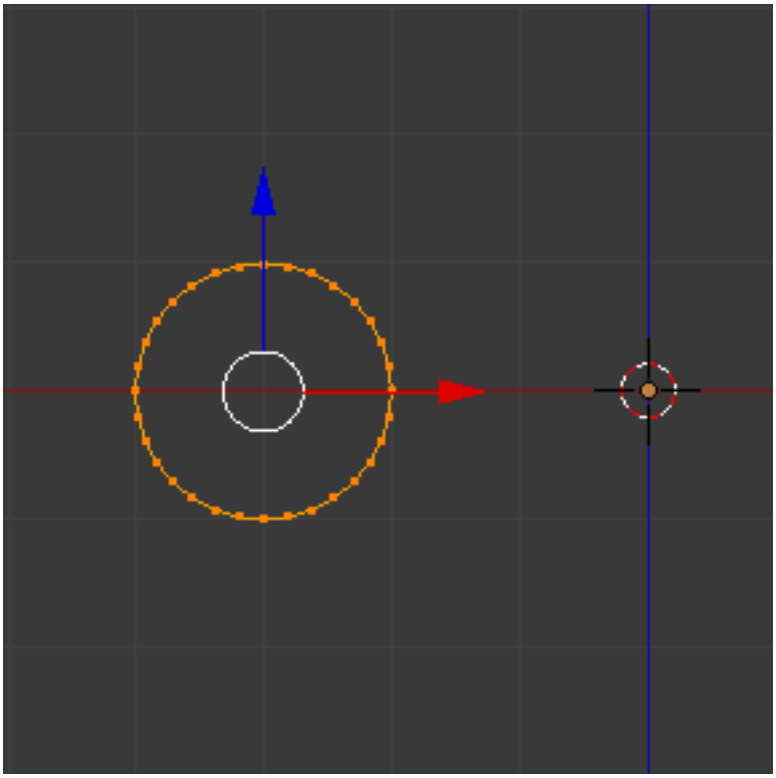


Fig. 2.541: Circle placed at X (-3, 0, 0).

1. Open Blender and delete the default Cube.
2. Change from perspective to orthographic view using shortcut `Numpad5`.
3. Change your view from *User Ortho* to *Front Ortho*, using the shortcut `Numpad1`. You will see the X (red) and Z (blue) coordinate lines.
4. In case you have moved your cursor by clicking anywhere in the screen, again place your cursor at the Center, using the shortcut `Shift-S` choosing *Cursor to Center* or the Transform Panel, placing your cursor at (0, 0, 0) typing directly into the Cursor 3D Location.
5. Add a circle using shortcut `Shift-A` → *Mesh* → *Circle*.
6. Rotate this circle using the shortcut `R X 9 0` and `Return`.

7. Apply the Rotation using Ctrl-A and choosing *Rotation*
8. Grab and move this circle three Blender Units on the *X-Axis* to the left; you can use the shortcut Ctrl while grabbing with the mouse using the standard transform widgets (clicking on the red arrow shown with the object and grabbing while using shortcut Ctrl until the down left info in the 3D View marks D. -3.0000 (3.0000) Global), or press the shortcut G X Minus 3 and Return. You can use the Transform Panel (toggled with the shortcut T , and type Minus 3 and Return in the Location too. (See the Fig. *Circle placed at X (-3, 0, 0).*).
9. You will have to scale your circle using the shortcut S . 5, then Return.
10. Now enter *Edit Mode* using shortcut Tab.
11. De-select all vertices using the shortcut A.

Now we will create a height vector for Blender:

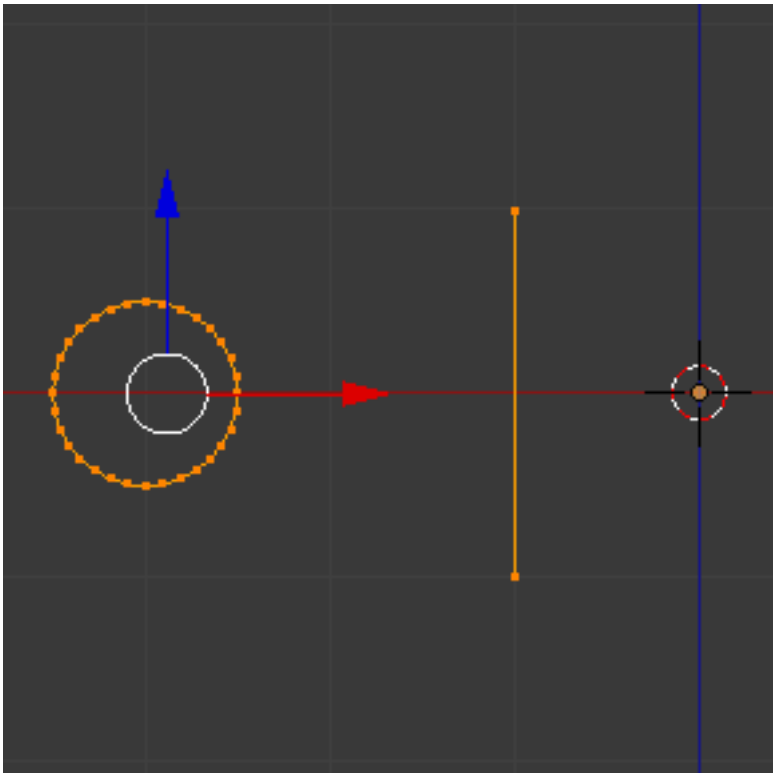


Fig. 2.542: Profile and vector created.

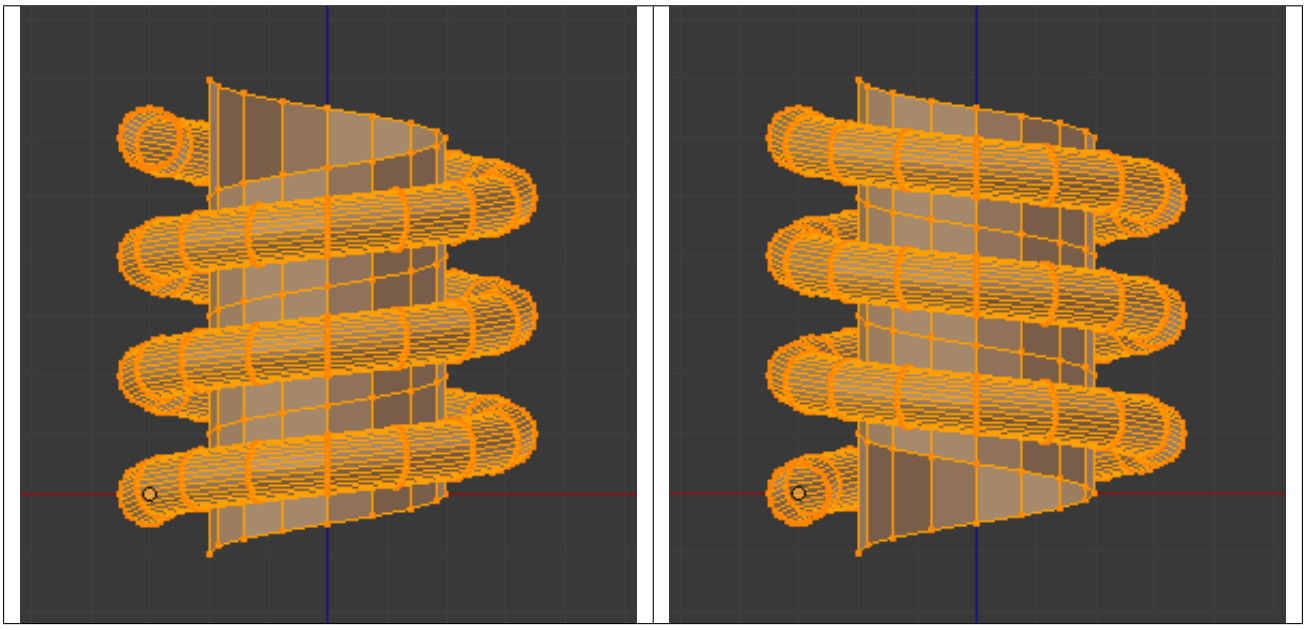
1. Press Ctrl and Left click LMB near the circle, in more or less at the light gray line of the square above the circle, and, while still pressing Ctrl, Left Click LMB again in the gray line below the circle. You have created two vertices and an Edge, which Blender will use as the first height and angle vector.
2. Now, in the Transform Panel, in the median, clicking in the Global coordinates, for the (X, Y, Z) coordinates, put (-2, 0, -1).
3. Right Click RMB in the other vertex, and again, type its coordinates for (X, Y, Z) to (-2, 0, 1). This will create a straight vertical line with 2 Blender units of Height.
4. De-select and select everything again with the shortcut A. (See Fig. *Profile and vector created.*)
5. Place again your cursor at the center. (Repeat step 2)
6. At this point, we will save this blend-file to recycle the Spring for another exercise; click with LMB in *File*, it is placed at the header of the Info editor, (At the top left side), and choose *Save as*. Our suggestion is to name it *Screw Spring Example.blend* and click in *Save as blend-file*. You can also use the shortcut Shift-Ctrl-S to open the File Browser in order to save your blend-file.

7. Click Screw and adjust the Steps and Turns as you like and we have a nice spring, but now here comes the interesting part!

Clockwise and Counterclockwise using the Spring Example

Still in the interactive session of the *Screw Tool*, you will see that the *Z-Axis* Value of the *Screw* Panel is set to 1.000. Left click `LMB` in the middle of the Value and set this value to -1.000. At first, the Spring was being constructed in a Counterclockwise direction, and you reverted the operation 180 degrees in the *Z-Axis*. This is because you have changed the angular vector of the height you have given to Blender to the opposite direction (remember, -90 to 90 = 180 degrees ?). See Fig. *Spring direction.*.

Table 2.21: Flipped to Clockwise direction.



It is also important to note that this vector is related to the same height vector axis used for the extrusion and we have created a parallel line with the *Z-Axis*, so, the sensibility of this vector is in practical sense reactive only to negative and positive values because it is aligned with the extrusion axis. Blender will clamp the positive and negative to its maximum values to make the extrusion follow a direction, even if the profile starts reverted. The same rule applies to other Global axes when creating the Object for the *Screw* Tool; this means if you create your Object using the Top View (Shortcut `Numpad7` with a straight parallel line following another axis (for the Top View, the *Y-Axis*), the vector that gives the height for extrusion will also change abruptly from negative to positive and vice versa to give the extrusion a direction, and you will have to tweak the corresponding Axis accordingly to achieve the Clockwise and Counterclockwise effect.

Note: Vectors that are not parallel with Blender Axis

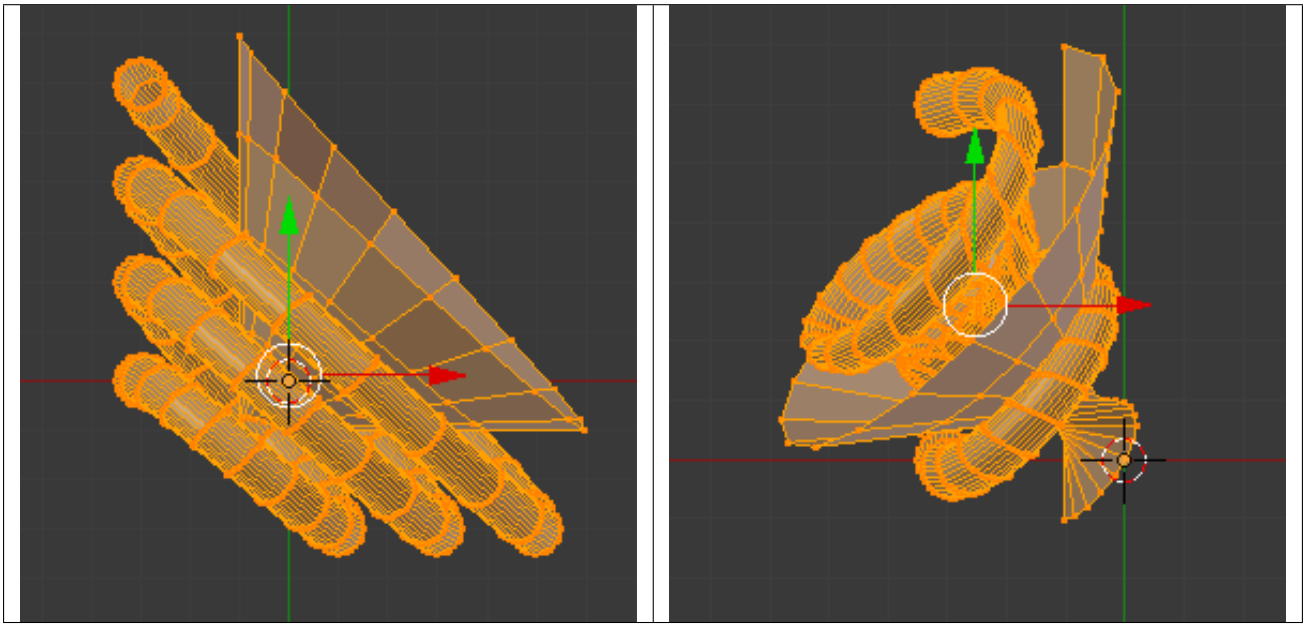
The high sensibility for the vector does not apply to vectors that give the Screw Tool a starting angle (Ex: any non-parallel vector), meaning Blender will not need to clamp the values to stabilize a direction for the extrusion, as the inclination of the vector will be clear for Blender and you will have the full degree of freedom to change the vectors. Our example is important because it only changes the direction of the profile without the tilt and/or bending effect, as there is only one direction for the extrusion, parallel to one of the Blender Axes.

Bending the Profiles using the Spring Example

Still using the Spring Example, we can change the remaining vector for the angles that are not related to the extrusion Axis of our Spring, thus bending our spring with the remaining vectors and creating a profile that will also open and/or close because of the change in starting angular vector values. What we are really doing is changing the starting angle of

the profile prior to the extrusions. It means that Blender will connect each of the circles inclined with the vector you have given. Below we show two bent Meshes using the Axis vectors and the Spring example. See Fig. *Bended Mesh.*. These two Meshes generated with the *Screw* tool were created using the Top Ortho View.

Table 2.22: The vector angle is maintained along the extrusions.



Creating perfect Screw Spindles

Using the Spring Example, it is easy to create perfect Screw Spindles (like the ones present in normal screws that we can buy in hardware stores). Perfect Screw Spindles use a profile with the same height as its vector, and the beginning and ending vertex of the profile are placed at a straight parallel line with the axis of extrusion. The easiest way of achieving this effect is to create a simple profile where the beginning and ending vertices create a straight parallel line. Blender will not take into account any of the vertices present in the middle but those two to take its angular vector, so the spindles of the screw (which are defined by the turns value) will assembly perfectly with each other.

1. Open Blender and click in *File* located at the header of the Info editor again, choose *Open Recent* and the file we saved for this exercise. All of the things will be placed exactly the way you saved before. Choose the last saved blend-file; in the last exercise, we gave it the name *Screw Spring Example.blend*.
2. Press the shortcut `A` to de-select all vertices.
3. Press the shortcut `B`, and Blender will change the cursor; you are now in border selection mode.
4. Open a box that selects all of the circle vertices except the two vertices we used to create the height of the extrusions in the last example.
5. Use the shortcut `X` to delete them.
6. Press the shortcut `A` to select the remaining vertices.
7. Press the shortcut `W` for the *Specials Menu*, and select *Subdivide*
8. Now, click with the Right Mouse button at the middle vertex.
9. Grab this vertex using the shortcut `G X Minus 1` and `Return`. See Fig. *Profile for a perfect screw spindle.*.
10. At this point, we will save this blend-file to recycle the generated Screw for another exercise; click with `LMB` in *File* – it is in the header of the Info editor (at the top left side), and choose *Save as*. Our suggestion is to name it *Screw Hardware Example.blend* and click in *Save as blend-file*. You can also use the shortcut `Shift-Ctrl-S` to open the File Browser in order to save your blend-file.
11. Press shortcut `A` twice to de-select and select all vertices again.

12. Now press Screw.
13. Change Steps and Turns as you like. Fig. *Generated Mesh.* - Shows you an example of the results.

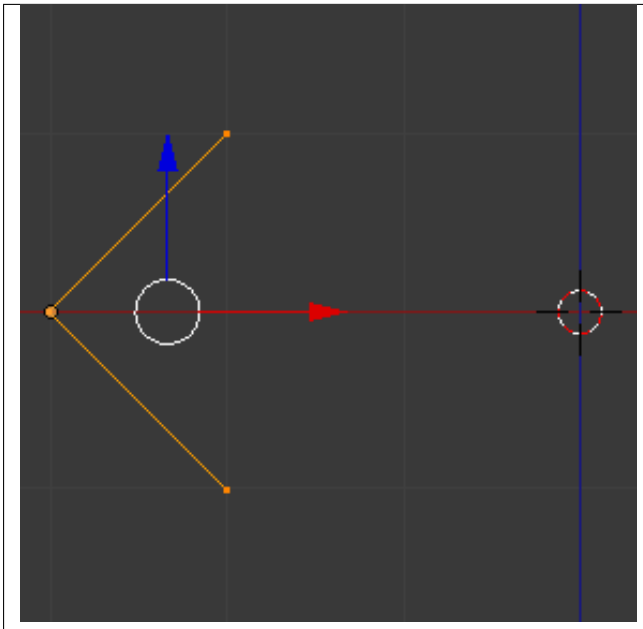


Fig. 2.547: Profile for a perfect screw spindle.
The starting and ending vertices are forming a parallel line with the Blender Axis.

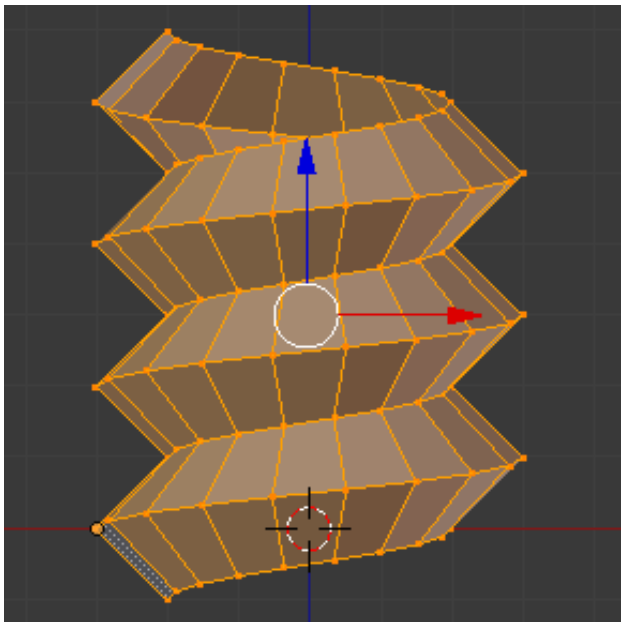
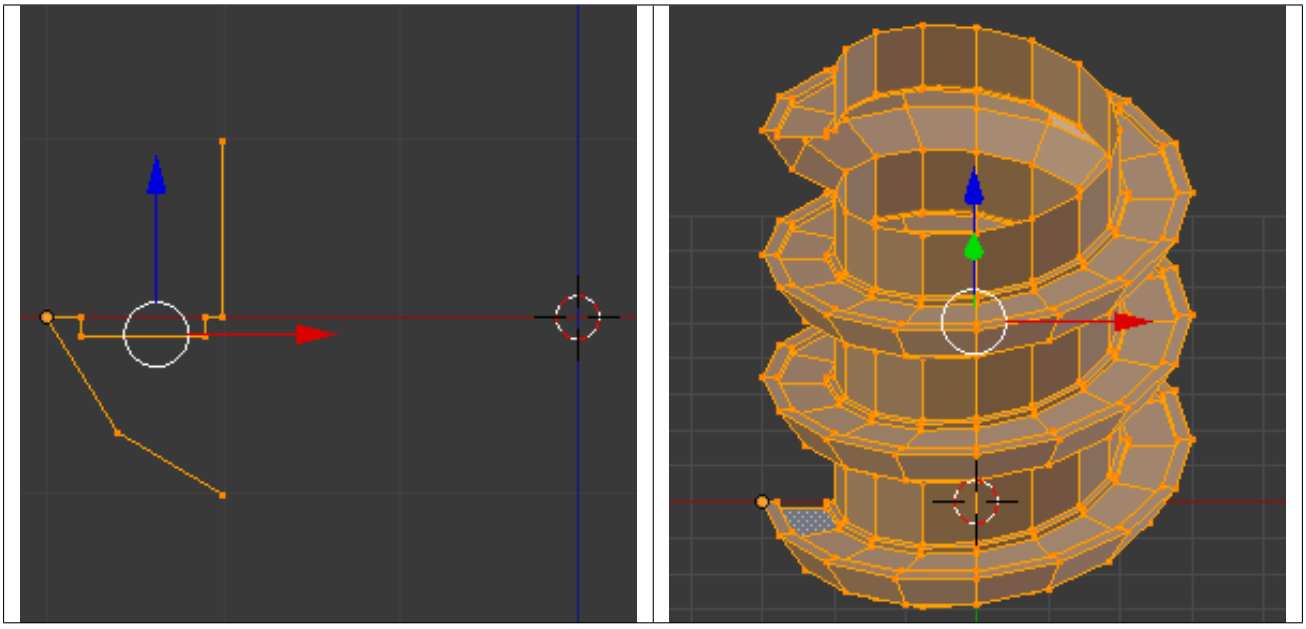


Fig. 2.548: Generated Mesh.
You can use this technique to perform normal screw modeling.

Here, in Fig. *Ramp.*, we show you an example using a different profile, but maintaining the beginning and ending vertices at the same position. The generated mesh looks like a medieval ramp!

Table 2.23: Generated Mesh with the profile at the left. We have inclined the visualization a bit.



As you can see, the Screw spindles are perfectly assembled with each other, and they follow a straight line from top to bottom. You can also change the Clockwise and Counterclockwise direction using this example, to create right and left screw spindles. At this point, you can give the screw another dimension, changing the Center of the Spin Extrusion, making it more suitable to your needs or calculating a perfect screw and merging its vertices with a cylinder, modeling its head, etc.

A Screw Tip

As we have explained before, the *Screw* tool generates clean and simple meshes to deal with; they are light, well-connected and are created with very predictable results. This is due to the Blender calculations taking into account not only the height of the vector, but also its starting angle. It means that Blender will connect the vertices with each other in a way that they follow a continuous cycle along the extruded generated profile.

In this example, you will learn how to create a simple Screw Tip (like the ones we use for wood; we have shown an example at the beginning of this page). To make this new example as short as possible, we will recycle our last example (again).

1. Open Blender and click in *File* located in the header of the Info editor again; choose *Open Recent* and the file we saved for this exercise. All of the things will be placed exactly the way you saved before. Choose the last saved blend-file; in the last exercise, we gave it the name *Screw Hardware Example.blend*.
2. Grab the upper vertex and move a bit to the left, but no more than you have moved your last vertex. (See Fig. *Profile With Starting Vector Angle.*)
3. Press the shortcut `A` twice to de-select and select all.
4. Press the shortcut `Shift-S` and select *Cursor to Center*
5. Press Screw.

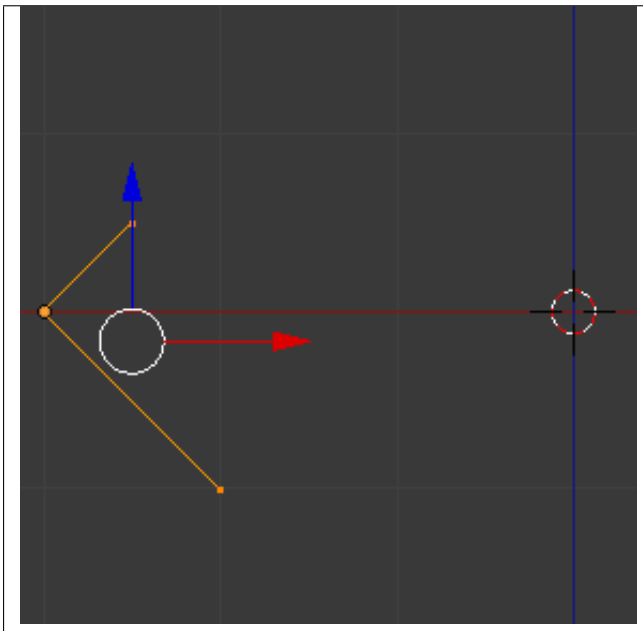


Fig. 2.551: Profile With Starting Vector Angle.

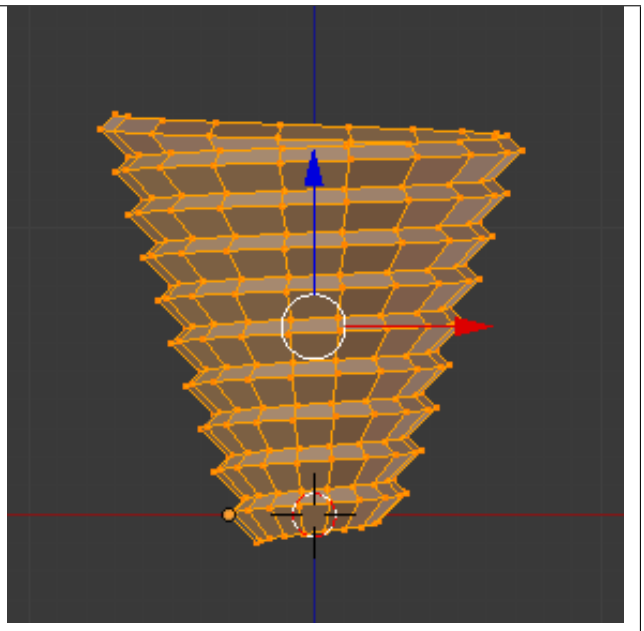


Fig. 2.552: Generated Mesh with the Profile.

As you can see in Fig. *Generated Mesh with the Profile.*, Blender follows the basic angular vector of the profile, and the profile basic angle determines whether the extruded subsequent configured turns will open or close the resulting mesh following this angle. The vector of the extrusion angle is determined by the starting and ending Vertex of the profile.

Subdividing

Introduction

Subdividing adds resolution by cutting existing faces and edges into smaller pieces. There are several tools that allow you to do this:

Subdivide Divide a face or edge into smaller units, adding resolution.

Loop Subdivide Insert a loop of edges between existing ones.

Vertex Connect Connects selected vertices with edges that split faces.

Knife Subdivide Cut edges and faces interactively.

Bevel Subdivides edges or vertices, making them faceted or rounded.

Subdivide

Reference

Mode: Edit Mode
Panel: Mesh Tools
Menu: *Mesh* → *Edges* → *Subdivide*, *Specials* → *Subdivide/Subdivide Smooth*

Subdividing splits selected edges and faces by cutting them in half or more, adding necessary vertices, and subdividing accordingly the faces involved, following a few rules, depending on the settings:

- When only one edge of a face is selected (Triangle mode), triangles are subdivided into two triangles, and quads, into three triangles.
- When two edges of a face are selected:
  - If the face is a triangle, a new edge is created between the two new vertices, subdividing the triangle in a triangle and a quad.
  - If the face is a quad, and the edges are neighbors, we have *three* possible behaviors, depending on the setting of *Corner Cut Type* (the select menu next to the *Subdivide* button, in *Mesh Tools* panel) See below for details.
  - If the face is a quad, and the edges are opposite, the quad is just subdivided in two quads by the edge linking the two new vertices.
- When three edges of a face are selected:
  - If the face is a triangle, this means the whole face is selected and it is then sub-divided in four smaller triangles.
  - If the face is a quad, first the two opposite edges are subdivided as described above. Then, the "middle" edge is subdivided, affecting its new "sub-quad" as described above for only one edge.
- When four edges of a face (a quad) are selected, the face is subdivided into four smaller quads.

Options

These options are available in the *Tool Panel* after running the tool;

Number of Cuts Specifies the number of cuts per edge to make. By default this is 1, cutting edges in half. A value of 2 will cut it into thirds, and so on.

Smoothness Displaces subdivisions to maintain approximate curvature, The effect is similar to the way the subdivision modifier might deform the mesh.

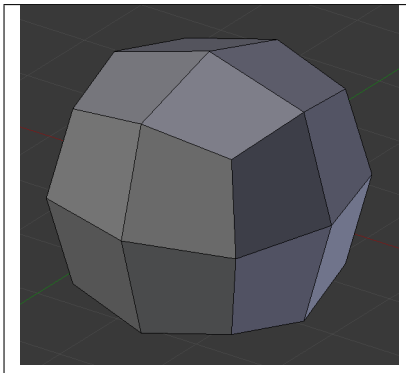


Fig. 2.553: Mesh before subdividing.

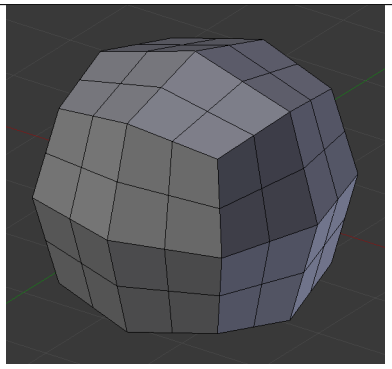


Fig. 2.554: Subdivided with no smoothing.

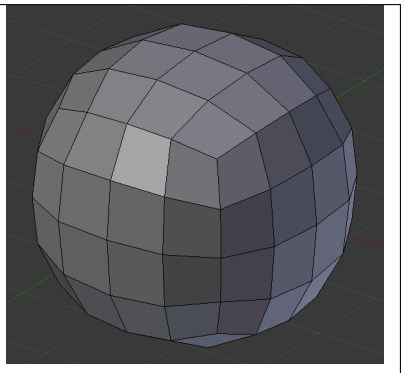


Fig. 2.555: Subdivided with smoothing of 1.

Quad/Tri Mode Forces subdivide to create triangles instead of n-gons, simulating old behavior (see examples below).

Corner Cut Type This select menu controls the way quads with only two adjacent selected edges are subdivided.

Fan the quad is sub-divided in a fan of four triangles, the common vertex being the one opposite to the selected edges.

Inner vertices The selected edges are sub-divided, then an edge is created between the two new vertices, creating a small triangle. This edge is also sub-divided, and the "inner vertex" thus created is linked by another edge to the one opposite to the original selected edges. All this results in a quad sub-divided in a triangle and two quad.

Path First an edge is created between the two opposite ends of the selected edges, dividing the quad in two triangles. Then, the same goes for the involved triangle as described above.

Straight Cut

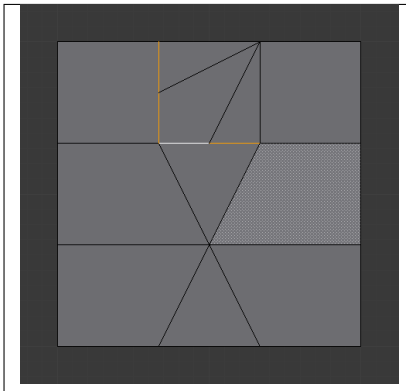


Fig. 2.556: Fan cut type.

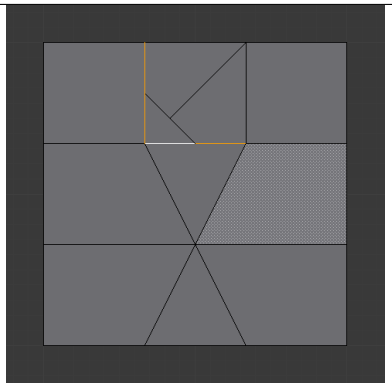


Fig. 2.557: Inner vertices cut type.

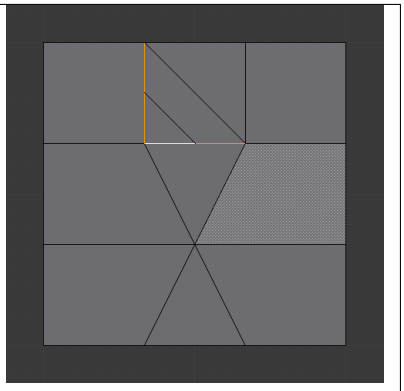


Fig. 2.558: Path cut type.

Fractal Displaces the vertices in random directions after the mesh is subdivided.

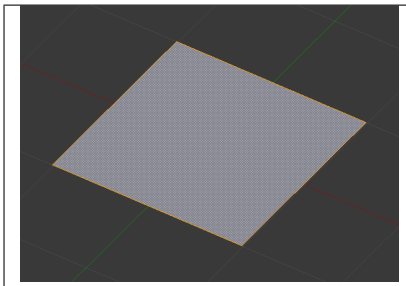


Fig. 2.559: Plane before subdivision.

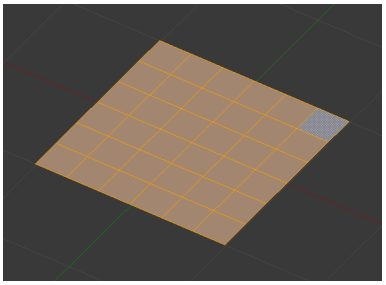


Fig. 2.560: Regular subdivision.

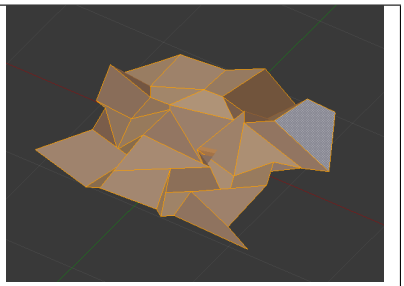


Fig. 2.561: Same mesh with fractal added.

Along Normal Causes the vertices to move along the their normals, instead of random directions.

Random Seed Changes the random seed of the noise function, producing a different result for each seed value.

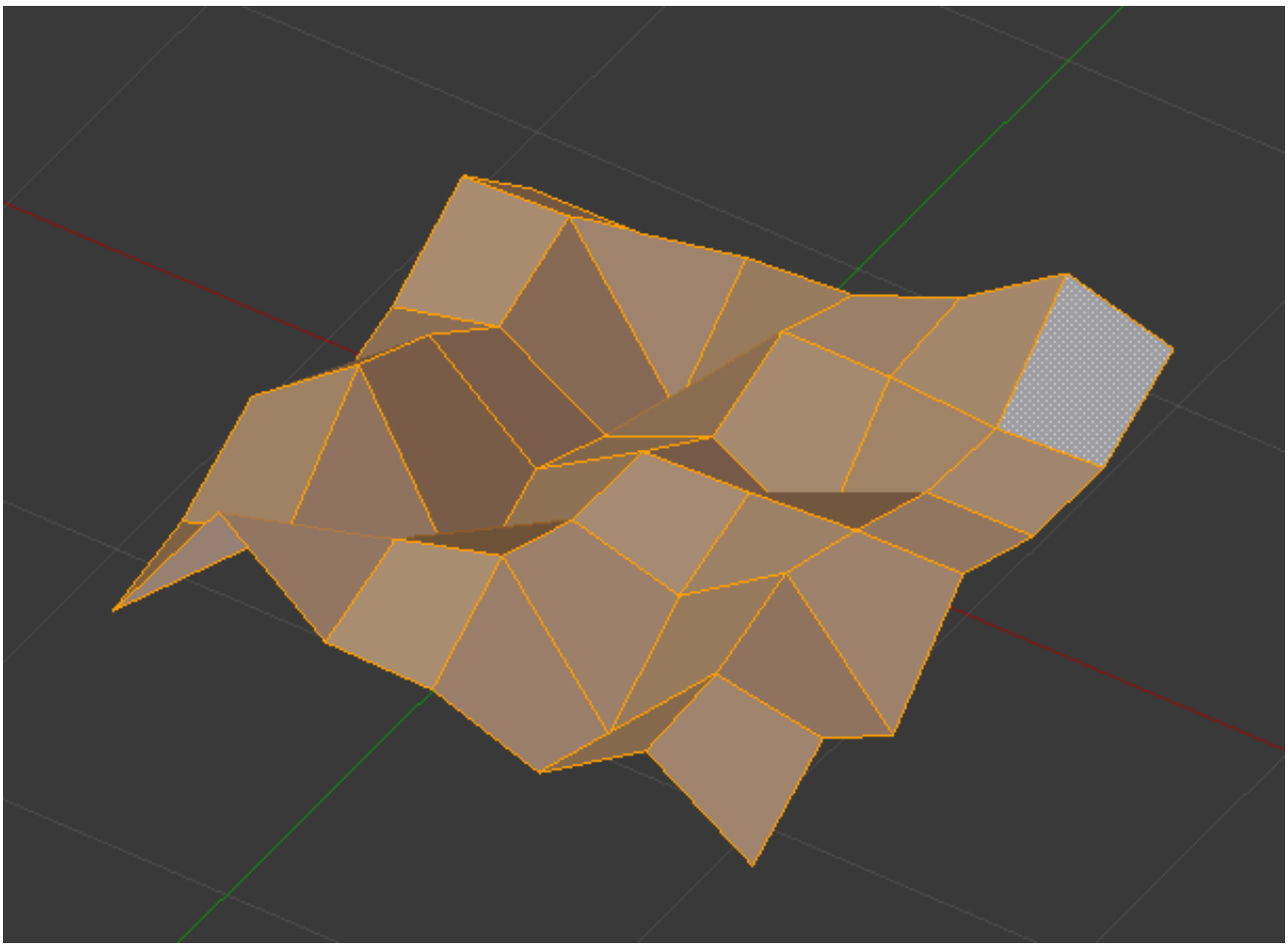


Fig. 2.562: Along normal set to 1.

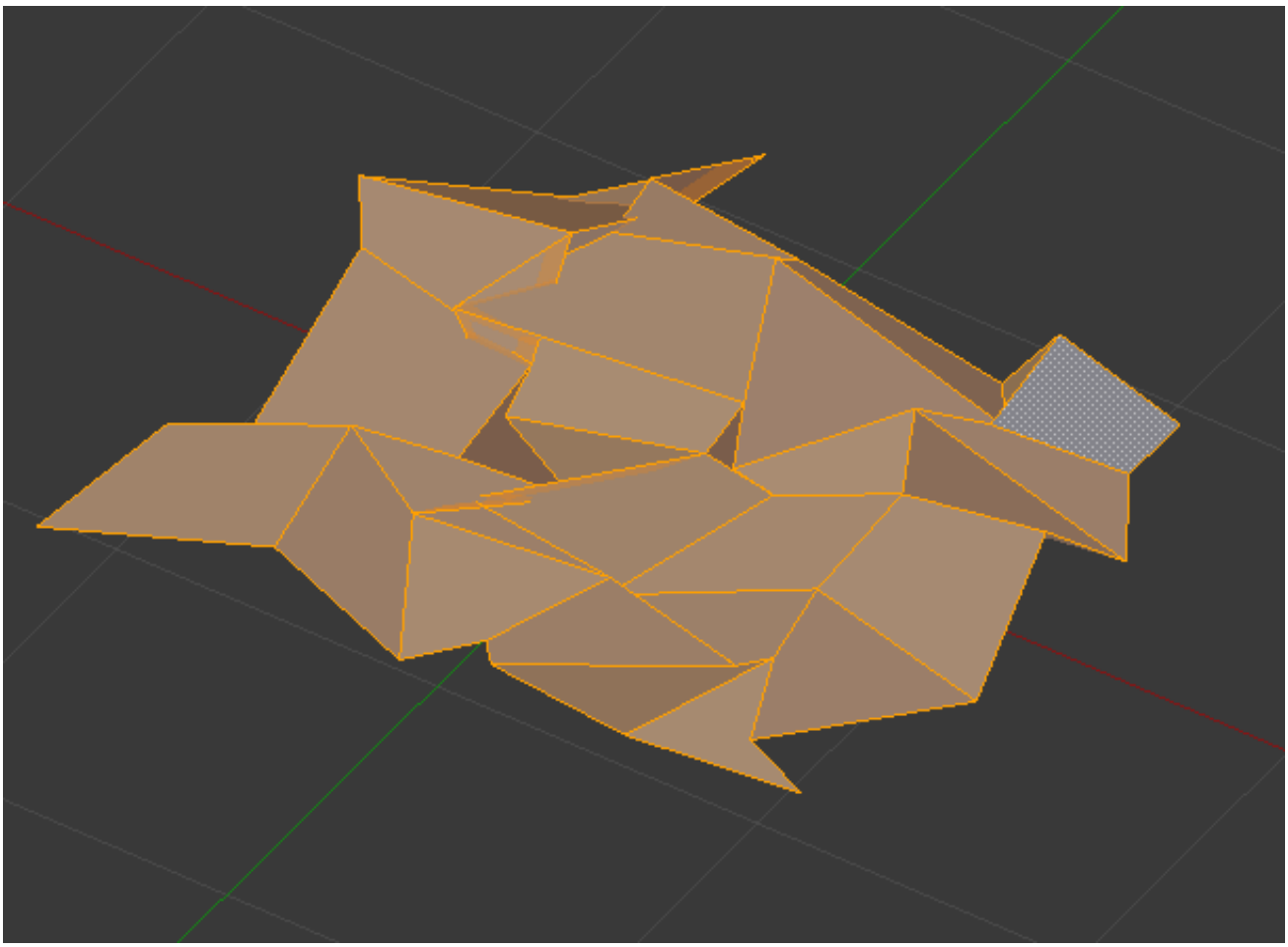


Fig. 2.563: Same mesh with a different seed value.

Examples

Below are several examples illustrating the various possibilities of the *Subdivide* and *Subdivide Multi* tools. Note the selection after subdivision.

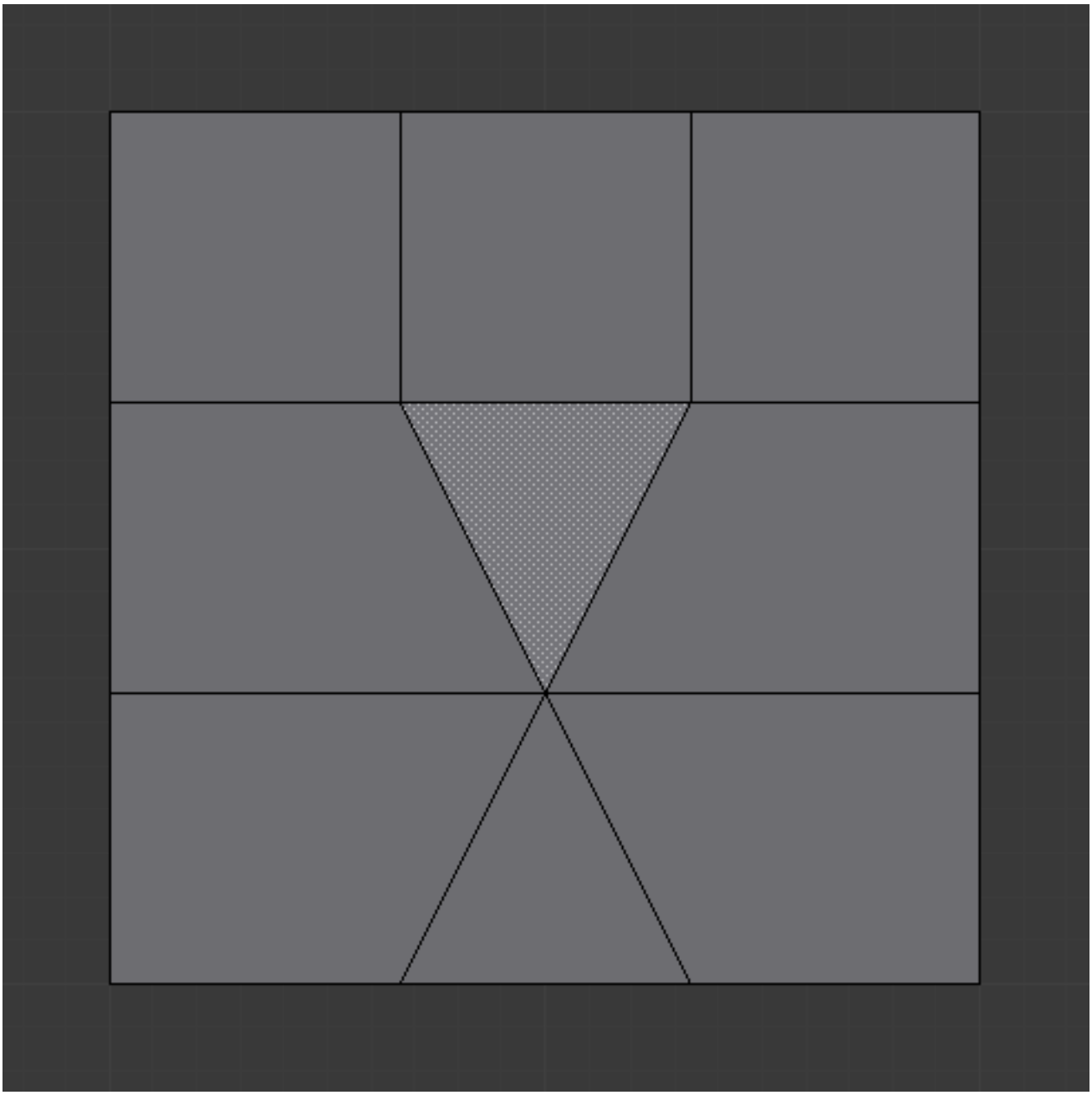


Fig. 2.564: The sample mesh.

One Edge

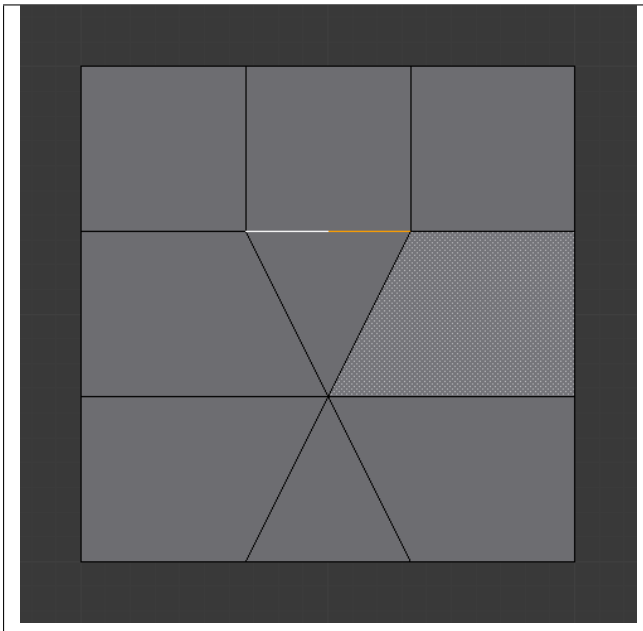


Fig. 2.565: One Edges.

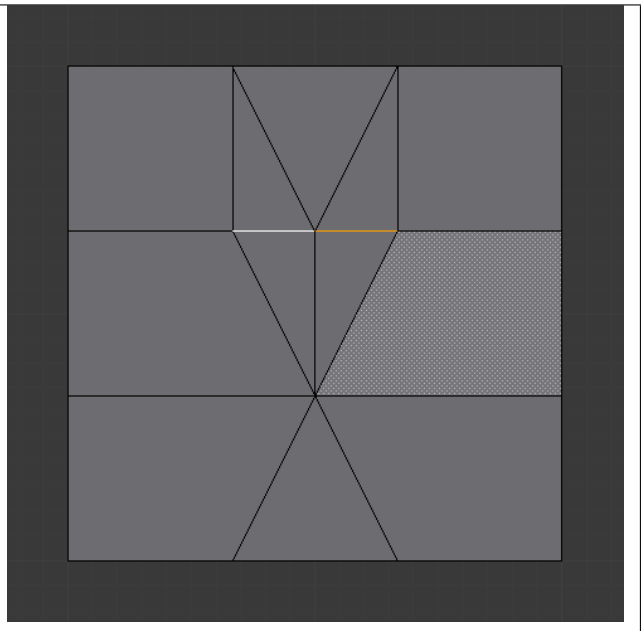
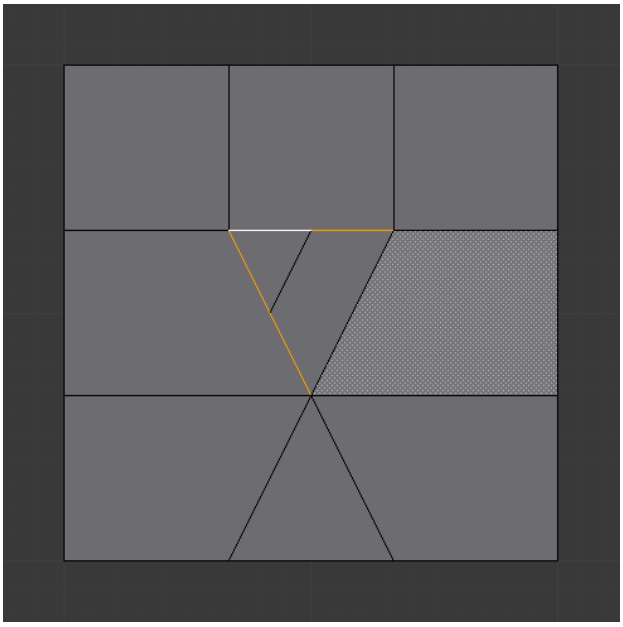


Fig. 2.566: Quad/Tri Mode.

Two Tri Edges



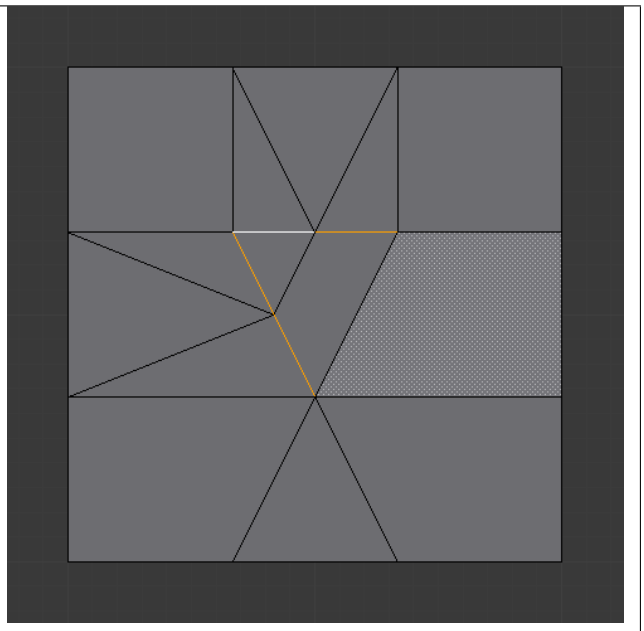
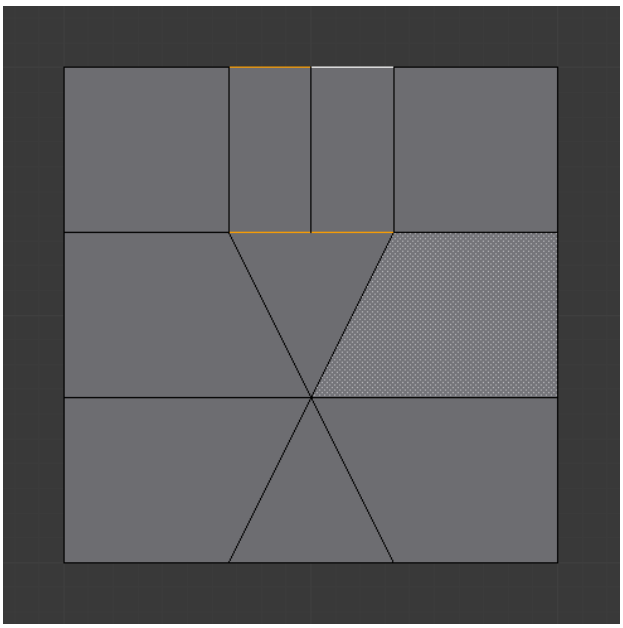


Fig. 2.567: Quad/Tri Mode.

Two Opposite Quad Edges



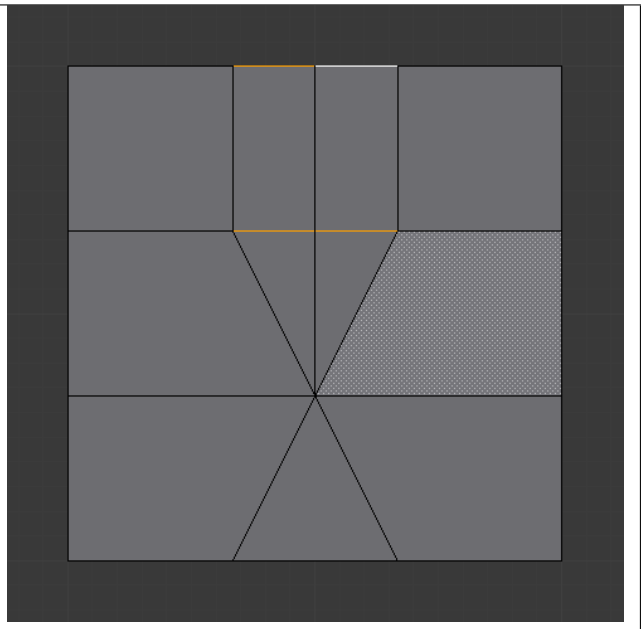


Fig. 2.568: Quad/Tri Mode.

Two Adjacent Quad Edges

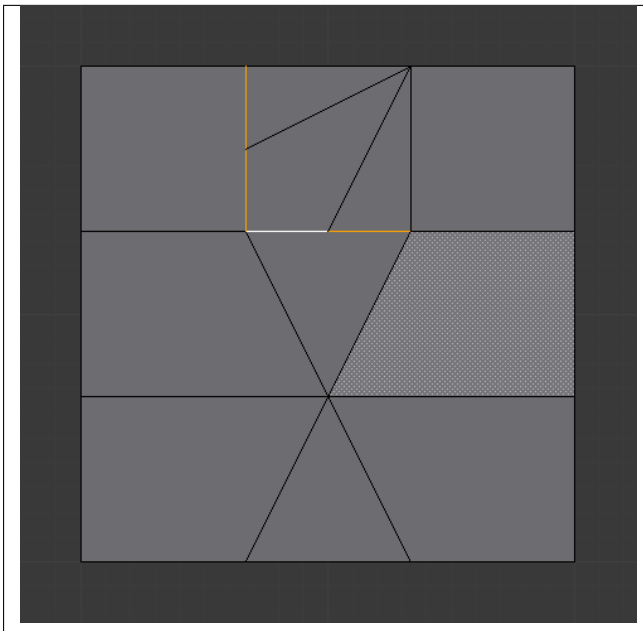


Fig. 2.569: Fan cut type.

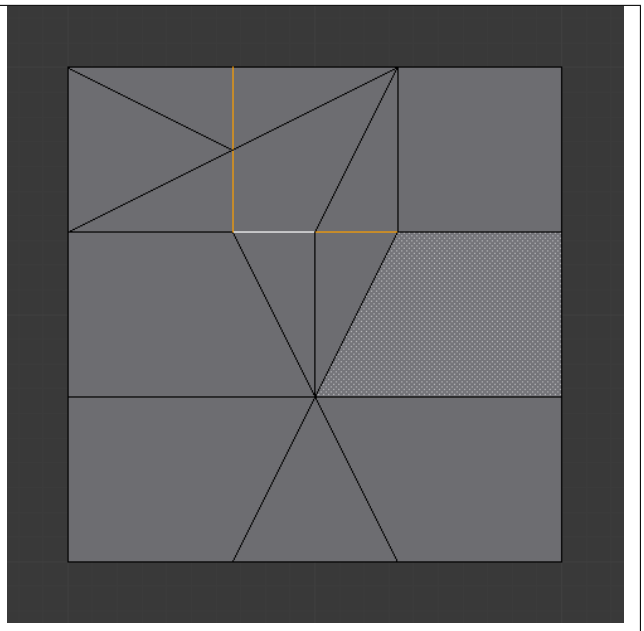


Fig. 2.570: Quad/Tri Mode.

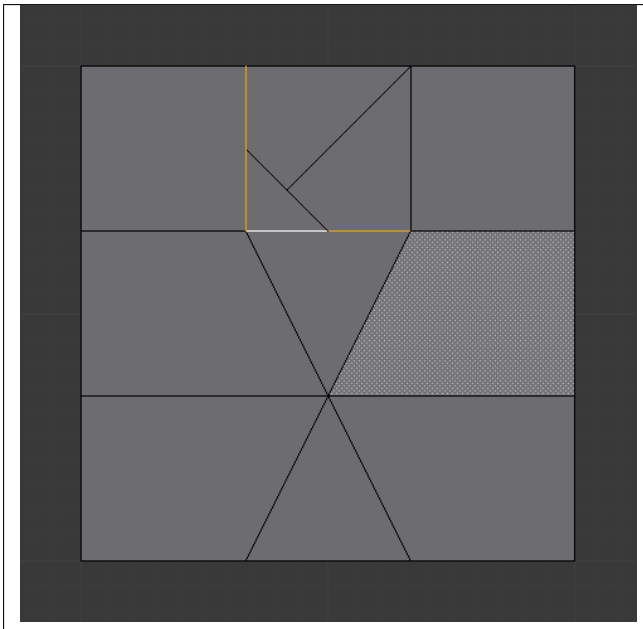


Fig. 2.571: Innervert cut type.

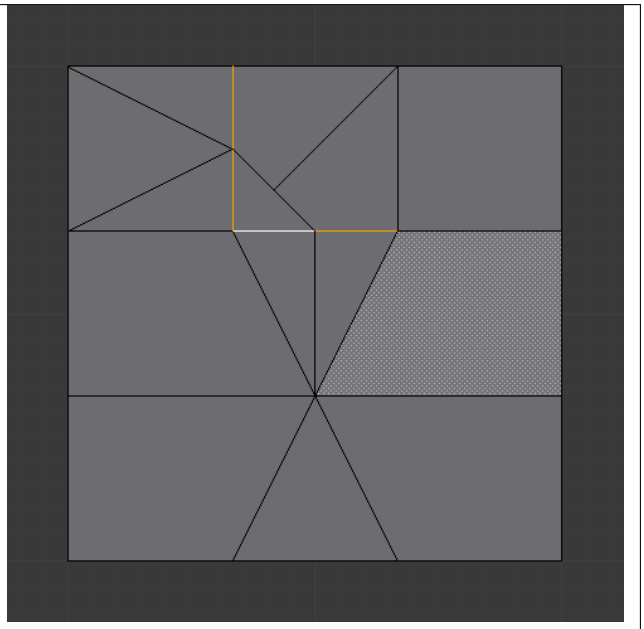


Fig. 2.572: Quad/Tri Mode.

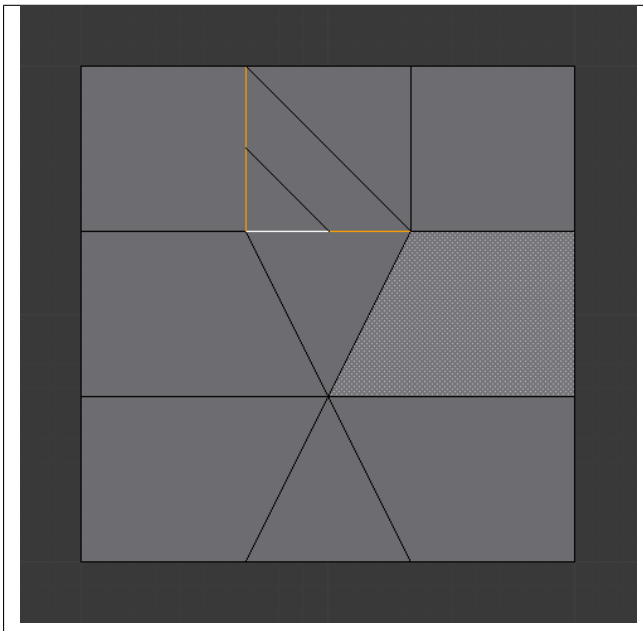


Fig. 2.573: Path cut type.

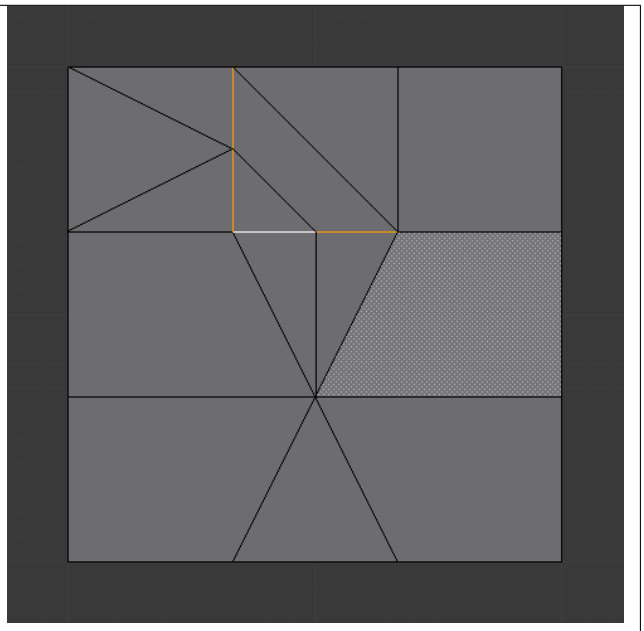
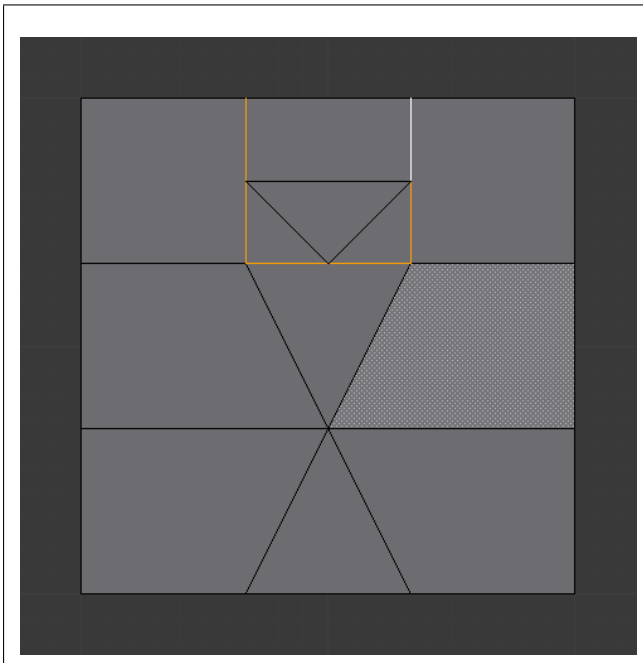


Fig. 2.574: Quad/Tri Mode.

Three Edges



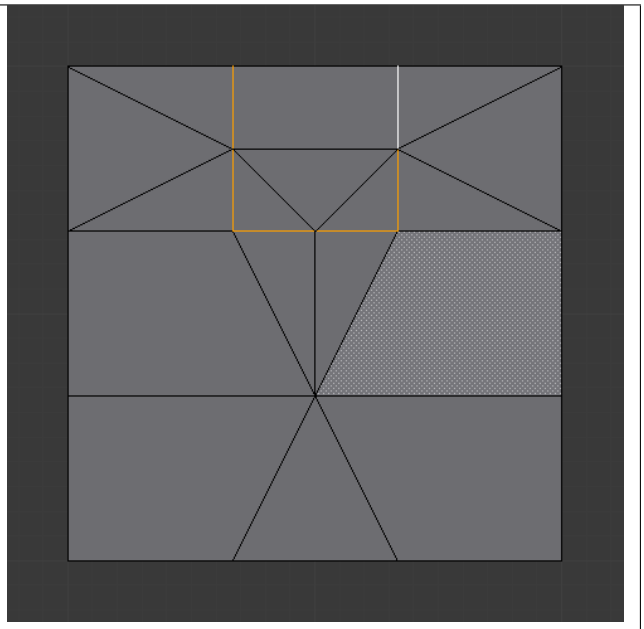
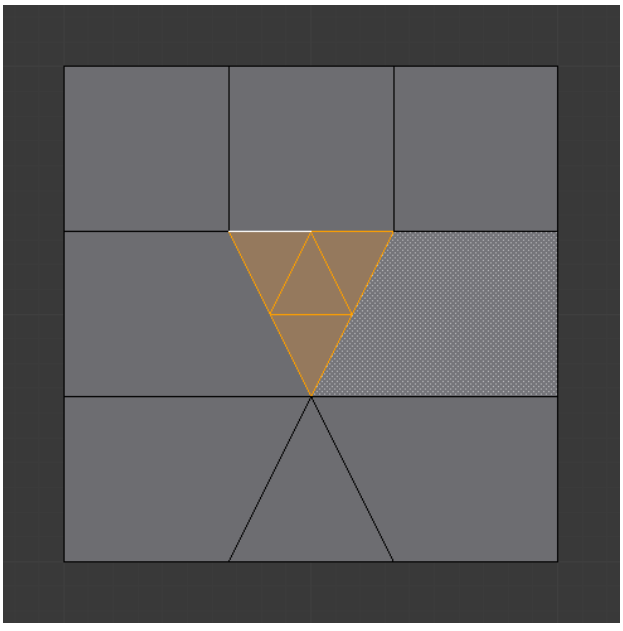


Fig. 2.575: Quad/Tri Mode.

Tri



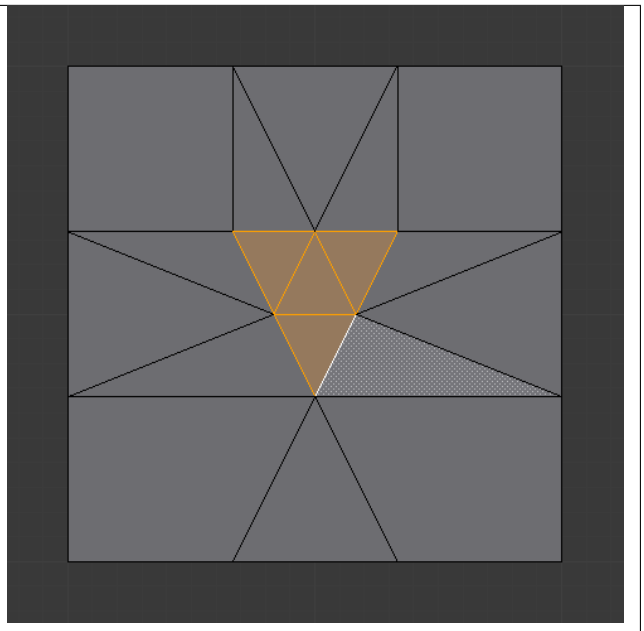
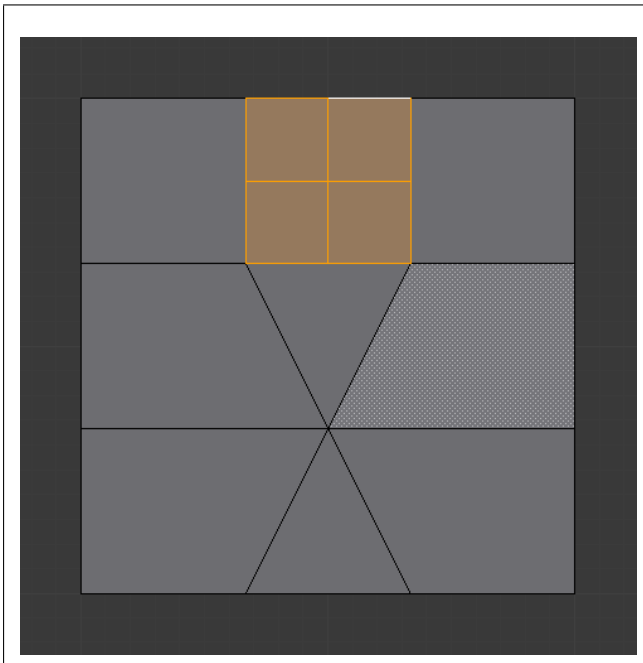


Fig. 2.576: Quad/Tri Mode.

Quad/Four Edges



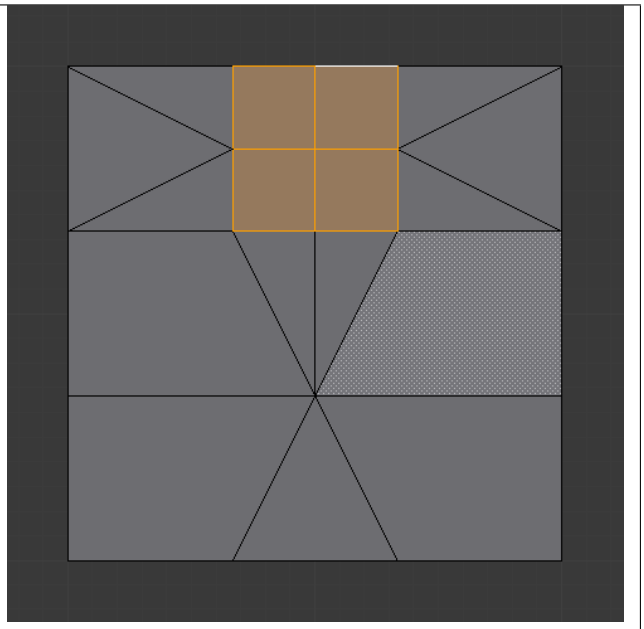


Fig. 2.577: Quad/Tri Mode.

Multicut

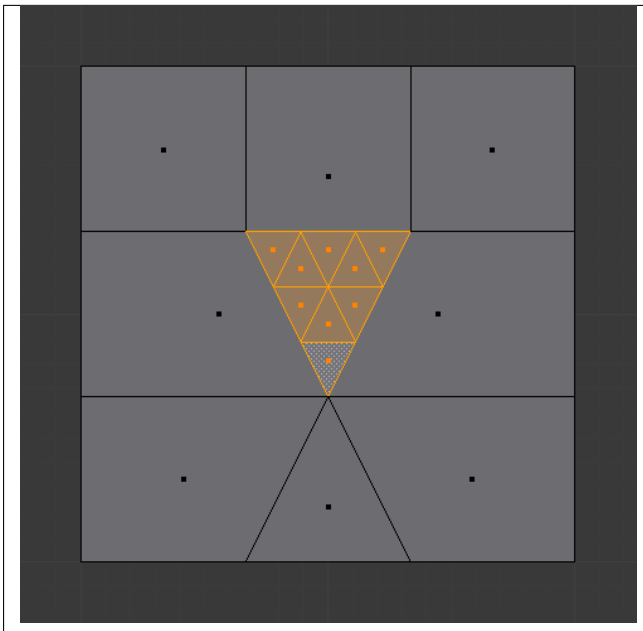


Fig. 2.578: Tri with two cuts.

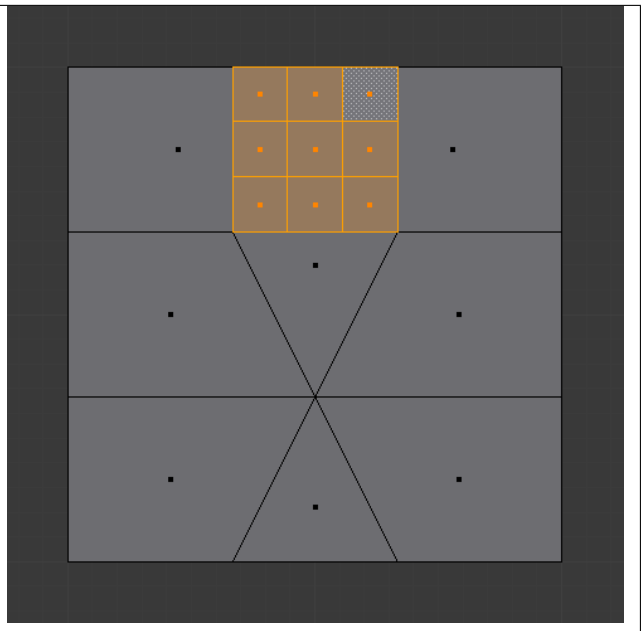


Fig. 2.579: Quad with two cuts

Loop Subdivide

Reference

Mode: Edit Mode

Panel: Mesh Tools
Hotkey: Ctrl-R

Loop Cut splits a loop of faces by inserting a new edge loop intersecting the chosen edge. The tool is interactive and has two steps:

Usage

Pre-visualizing the cut After the tool is activated, move the cursor over a desired edge. The cut to be made is marked with a magenta colored line as you move the mouse over the various edges. The to-be-created edge loop stops at the poles (tris and ngons) where the existing face loop terminates.

Sliding the new edge loop Once an edge is chosen via LMB, you can move the mouse along the edge to determine where the new edge loop will be placed. This is identical to the *Edge Slide tool*. Clicking LMB again confirms and makes the cut at the pre-visualized location, or clicking RMB forces the cut to exactly 50%. This step is skipped when using multiple edge loops (see below)

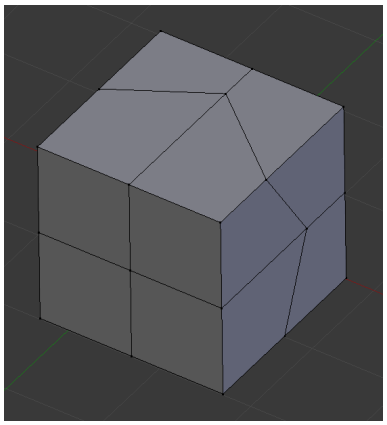


Fig. 2.580: Mesh before inserting edge loop.

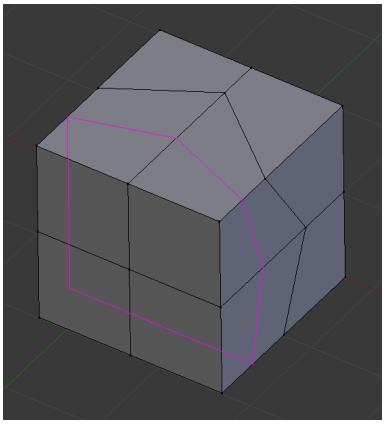


Fig. 2.581: Preview of edge loop location.

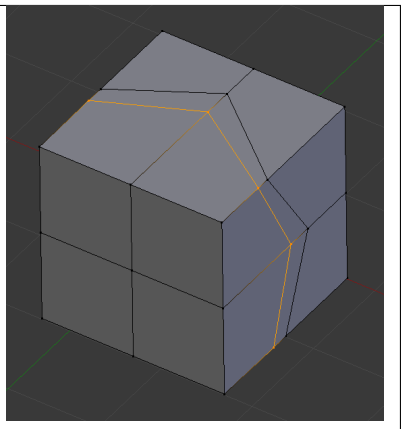


Fig. 2.582: Interactive placement of edge loop between adjacent loops.

Options

Options are only available while the tool is in use, and are displayed in the 3D View header.

Even E Only available for single edge loops. This matches the shape of the edge loop to one of the adjacent edge loops. (See *Edge Slide tool* for details).

Flip F When Even is enabled, this flips the target edge loop to match. (See *Edge Slide tool* for details).

Number of Cuts Wheel or NumpadPlus / NumpadMinus After activating the tool, but before confirming initial loop location, you can increase and decrease the number of cuts to create, by entering a number with the keyboard, scrolling Wheel or using NumpadPlus and NumpadMinus.

Note: When creating multiple loops, these cuts are uniformly distributed in the original face loop, and you will *not* be able to control their positions.

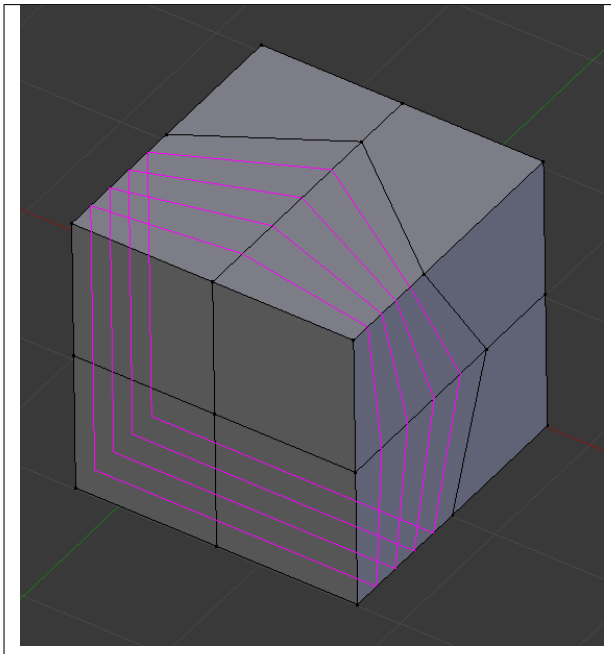


Fig. 2.583: Preview of multiple edge loops.

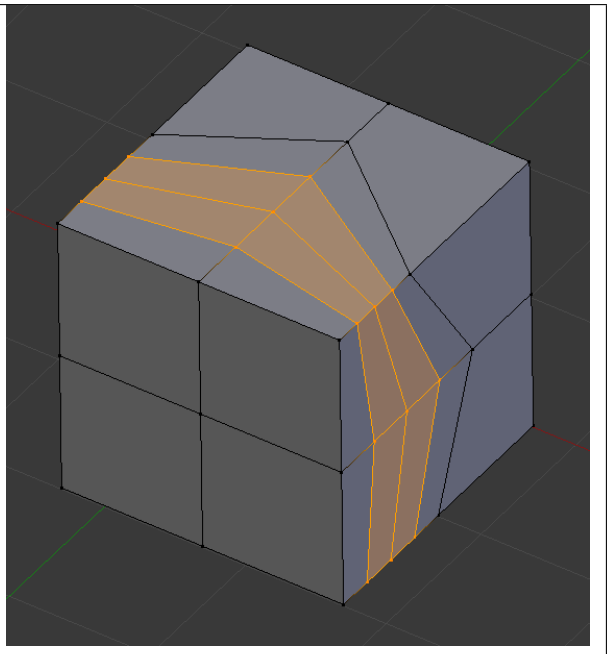


Fig. 2.584: Result of using multiple cuts.

Smoothing `Alt-Wheel` Smoothing causes edge loops to be placed in an interpolated position, relative to the face it is added to, causing them to be shifted outwards or inwards by a given percentage, similar to the *Subdivide Smooth* command. When not using smoothing, new vertices for the new edge loop are placed exactly on the pre-existing edges. This keeps subdivided faces flat, but can distort geometry, particularly when using *Subdivision Surfaces*. Smoothing can help maintain the curvature of a surface once it is subdivided.

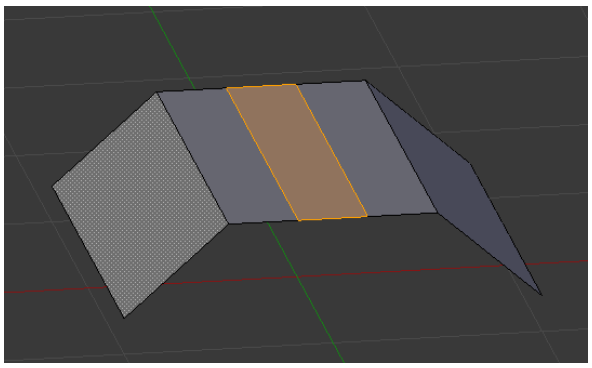


Fig. 2.585: Added edge loops without smoothing.

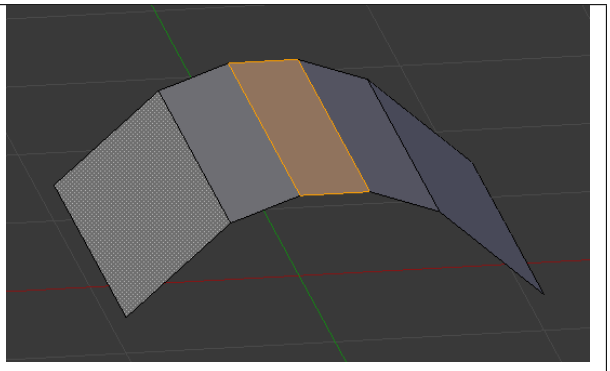


Fig. 2.586: Same edge loops, but with smoothing value.

Knife Tool

Reference

Mode: Edit Mode
Panel: Mesh Tools
Hotkey: `K` or `Shift-K`

The knife tool can be used to interactively cut up geometry by drawing lines or closed loops to create holes.

Usage

When you press K (or Shift-K), the Knife tool becomes active.

Drawing the cut line When using *Knife*, the cursor changes to an icon of a scalpel and the header changes to display options for the tool. You can draw connected straight lines by clicking LMB.

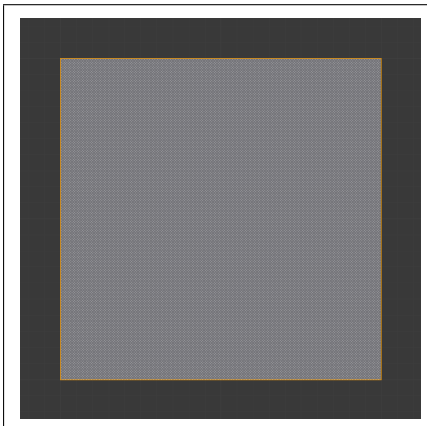


Fig. 2.587: Mesh before knife cut.

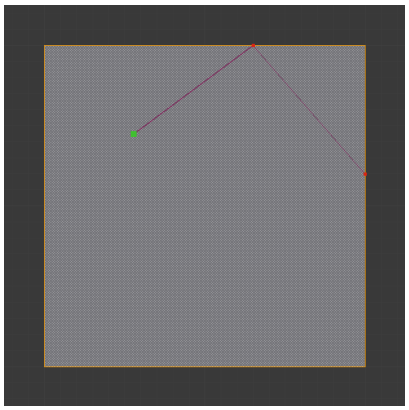


Fig. 2.588: Knife cut active.

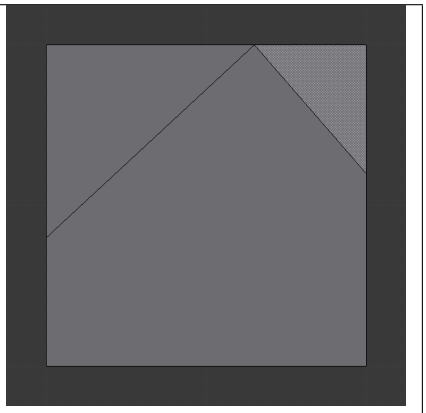


Fig. 2.589: After confirming knife cut.

Options

Knife selection Shift-K Activates the knife so only selected faces are cut.

New cut E Begins a new cut. This allows you to define multiple distinct cut lines. If multiple cuts have been defined, they are recognized as new snapping points.

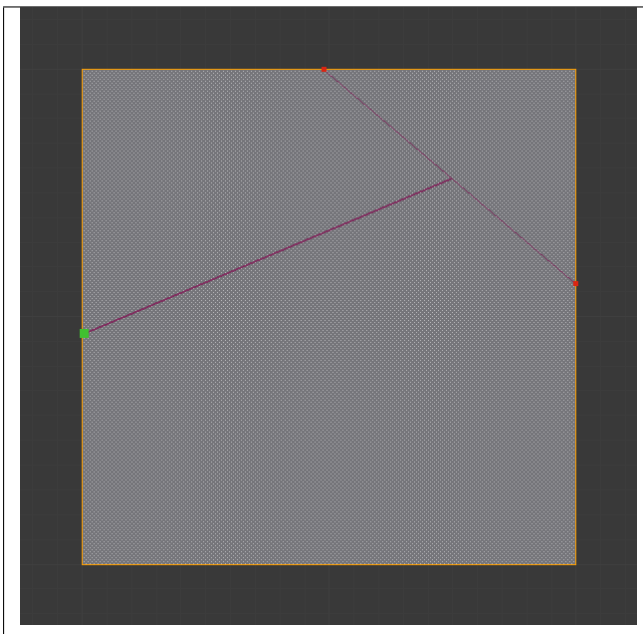


Fig. 2.590: Creating multiple cuts.

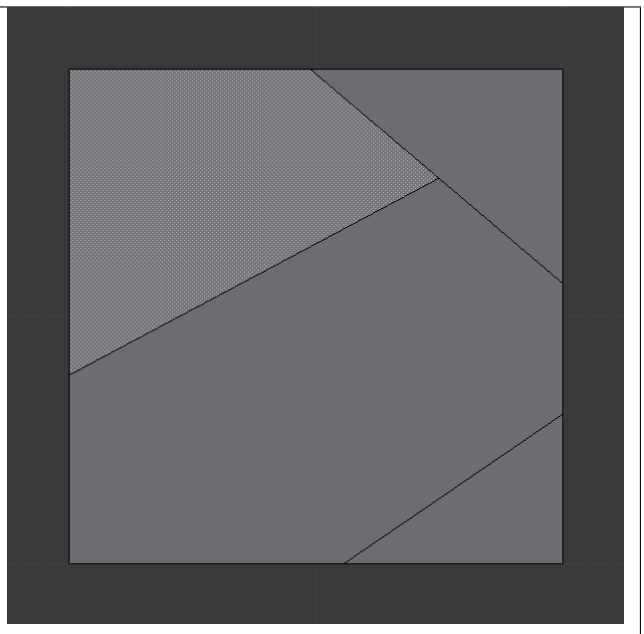


Fig. 2.591: Result of starting new cuts while in the tool.

Midpoint snap Ctrl Hold to snap the cursor to the midpoint of edges

Ignore snap Shift Hold to make the tool ignore snapping.

Cut through: Z Allow the cut tool to cut through to obscured faces, instead of only the visible ones.

Angle constrain C Constrains the cut to 45 degree increments.

Close loop: Double click LMB This is a quick way to close the loop you are currently cutting.

Draw a continuous line: LMB drag. So you can draw a freehand line over a surface, points will be created at edge intersections.

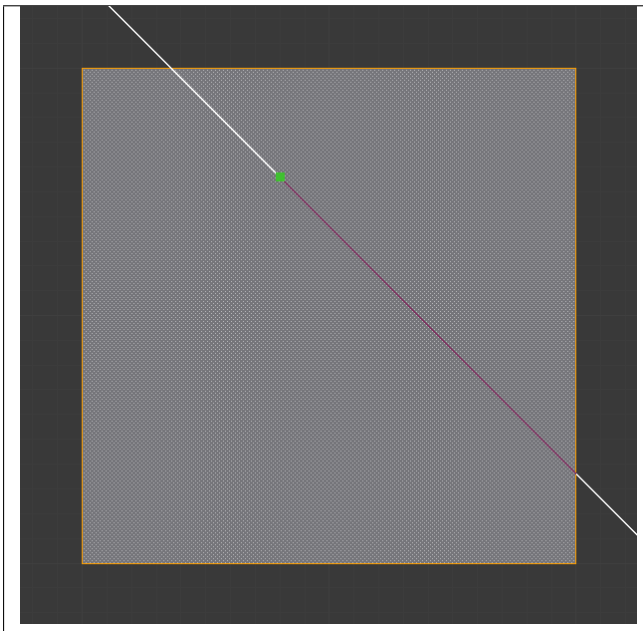


Fig. 2.592: Constraining cut angle.

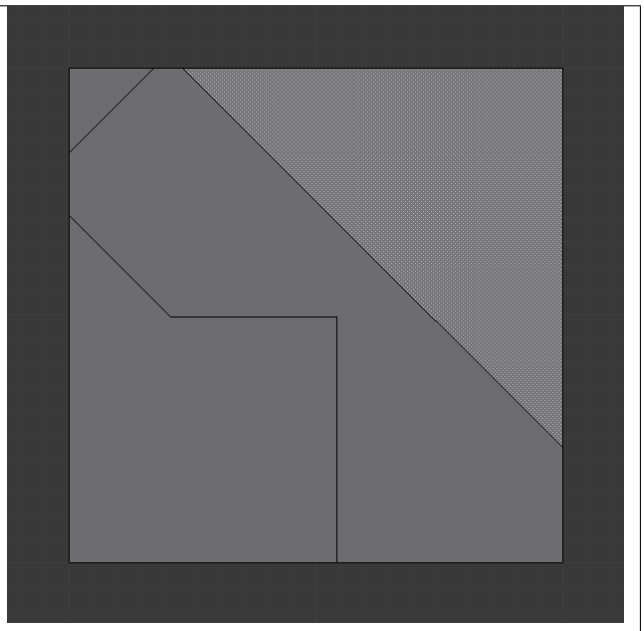


Fig. 2.593: Result of constraining cut angle.

Confirming and selection

Pressing `Esc` or `RMB` at any time cancels the tool, and pressing `LMB` or `Return` confirms the cut, with the following options:

`Return` will leave selected every edge except the new edges created from the cut.

Limitations

Cuts that begin or end in the middle of a face, will be ignored. This is a limitation of the current geometry that can be modeled in Blender.

Knife Project

Knife projection is a non-interactive tool where you can use objects to cookie-cut into the mesh rather than hand drawing the line.

This works by using the outlines of other selected objects in edit-mode to cut into the mesh, resulting geometry inside the cutters outline will be selected.

Outlines can be wire or boundary edges.

To use Knife Project, in 'object' mode select the "cutting object" first then shift select the "object to be cut". Now tab into edit mode and press "knife project".

Examples

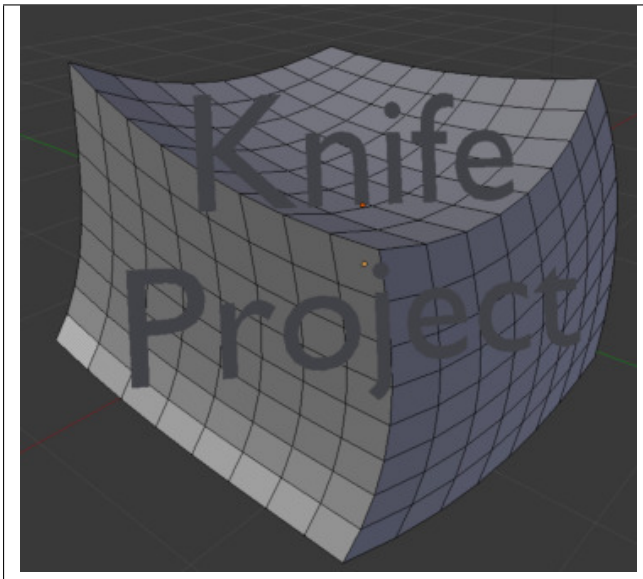


Fig. 2.594: Before projecting from a text object.

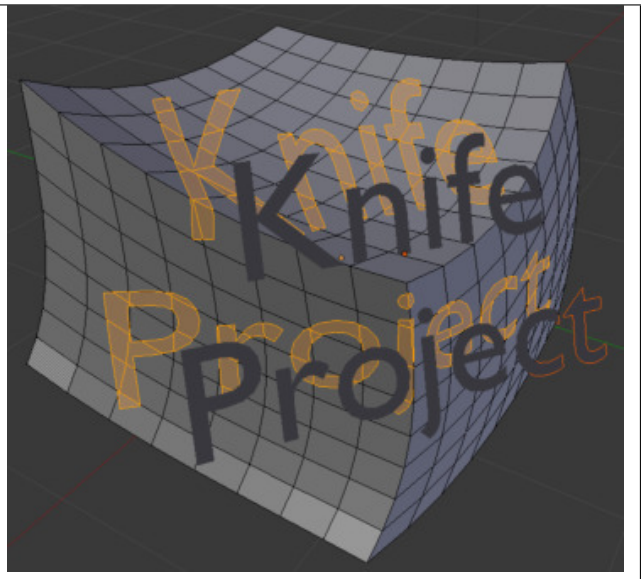


Fig. 2.595: Resulting knife projection.

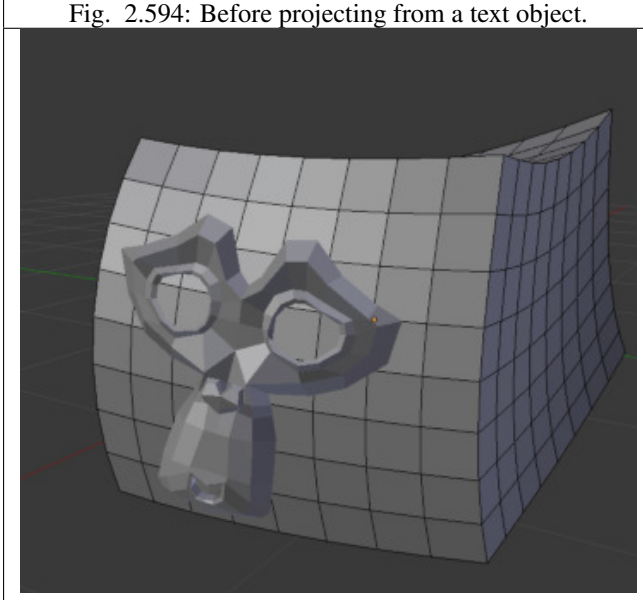


Fig. 2.596: Before projecting from a mesh object.

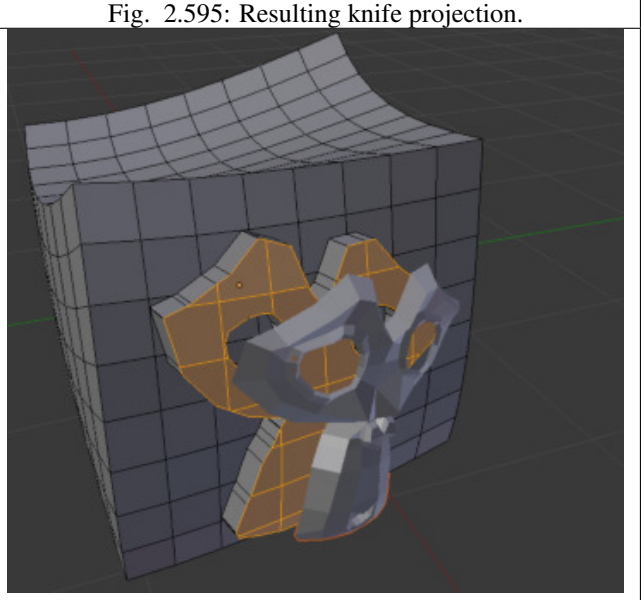


Fig. 2.597: Resulting knife projection (extruded after).

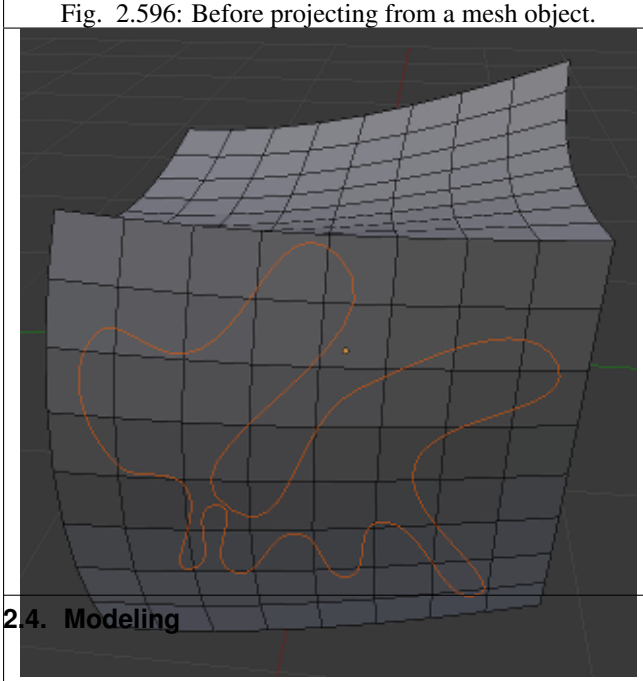


Fig. 2.598: Before projecting from a 3D curve object.

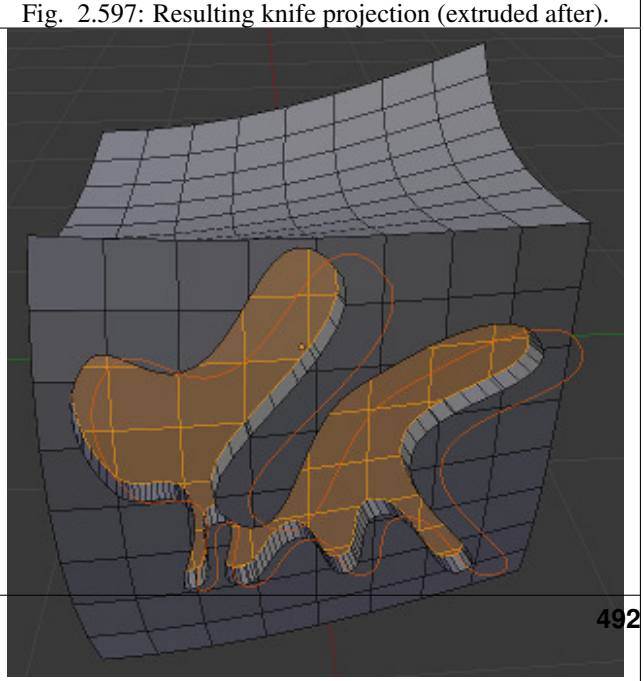


Fig. 2.599: Resulting knife projection (extruded after).

Known Issues

Cutting holes into single faces may fail, this is the same limitation as with the regular knife tool but more noticeable for text, this can be avoided by projecting onto more highly subdivided geometry.

Mesh Bisect

Reference

Mode: Edit Mode
Menu: *Mesh* → *Bisect*

The bisect tool is a quick way to cut a mesh in-two along a custom plane.

There are three important differences between this and the knife tool.

- The plane can be numerically adjusted in the operator panel for precise values.
- Cuts may remove geometry on one side.
- Cuts can optionally fill in the holes created, with materials and UVs & vertex-colors based on the surrounding geometry.

This means the bisect tool can cut off parts of a mesh without creating any holes.

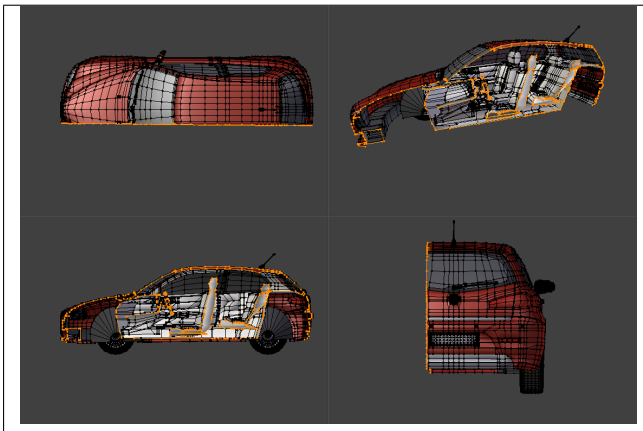


Fig. 2.600: Example of a common use of bisect.

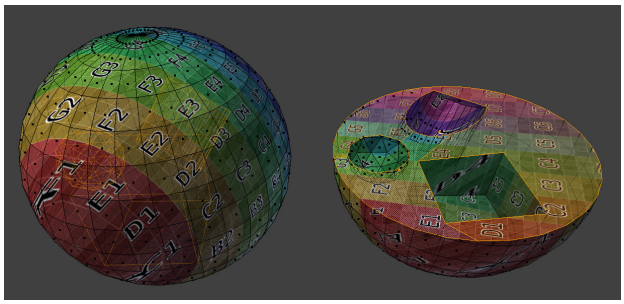


Fig. 2.601: Example of bisect with fill option

Vertex Connect

Reference

Mode: Edit Mode
Menu: Mesh –> Vertices –> Connect
Hotkey: J

This tool joins selected vertices by edges, The main difference between this and creating edges is that faces are split by the newly joined vertices.

When many vertices are selected, faces will be split by their selected vertices.

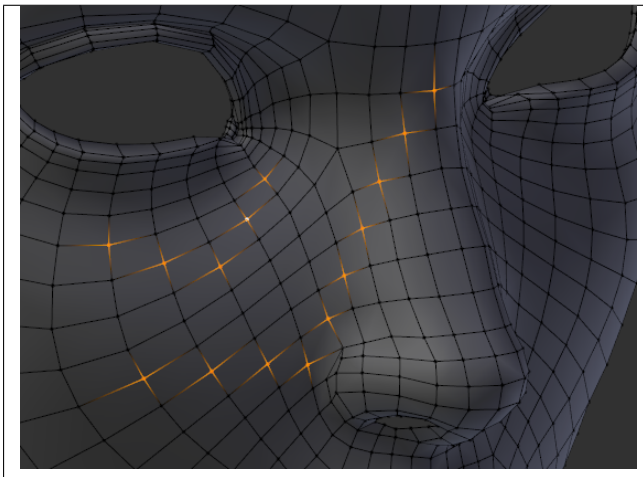


Fig. 2.602: Before.

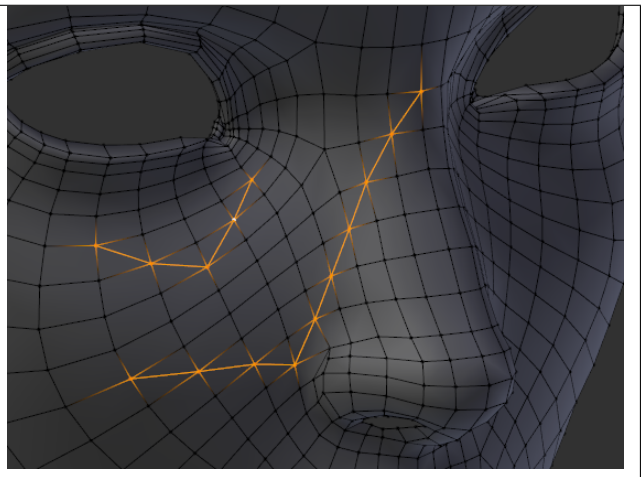


Fig. 2.603: After.

When there are only two vertices selected, a cut will be made across unselected faces, a little like the knife tool; however, this is limited to straight cuts across connected faces.

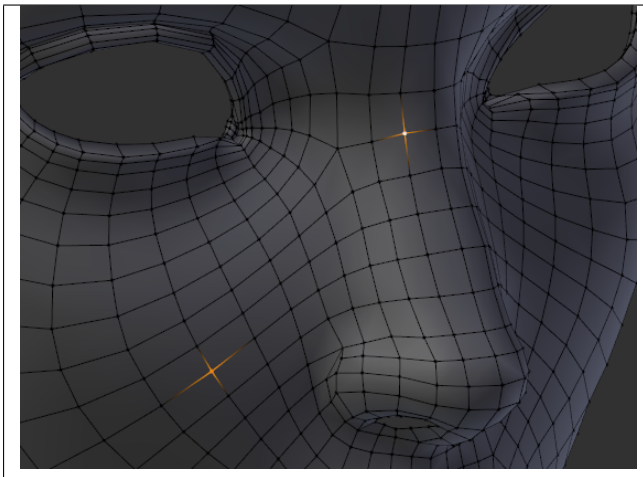


Fig. 2.604: Before.

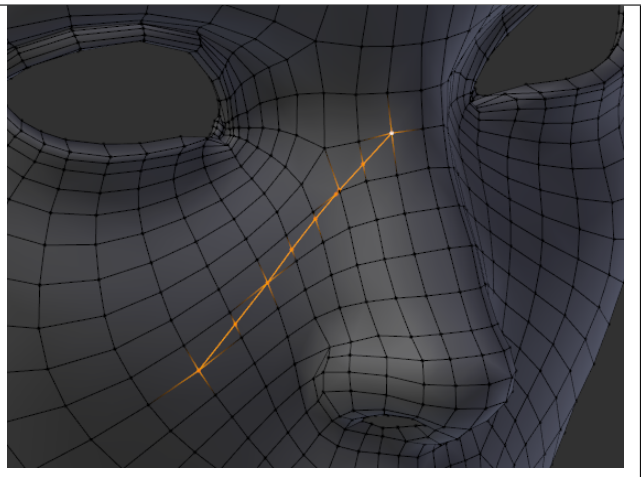


Fig. 2.605: After.

Bevel

Reference

Mode: Edit Mode
Menu: *Mesh* → *Edges* → *Bevel* or `Ctrl-E` → *Bevel*
Hotkey: `Ctrl-B` or `W` → *Bevel*
Menu (vertex-only): *Mesh* → *Vertices* → *Bevel* or `Ctrl-V` → *Bevel*
Hotkey (vertex-only): `Shift-Ctrl-B`

The bevel tool allows you to create chamfered or rounded corners to geometry. A bevel is an effect that smooths out edges

and corners. True world edges are very seldom exactly sharp. Not even a knife blade edge can be considered perfectly sharp. Most edges are intentionally beveled for mechanical and practical reasons.

Bevels are also useful for giving realism to non-organic models. In the real world, the blunt edges on objects catch the light and change the shading around the edges. This gives a solid, realistic look, as opposed to un-beveled objects which can look too perfect.

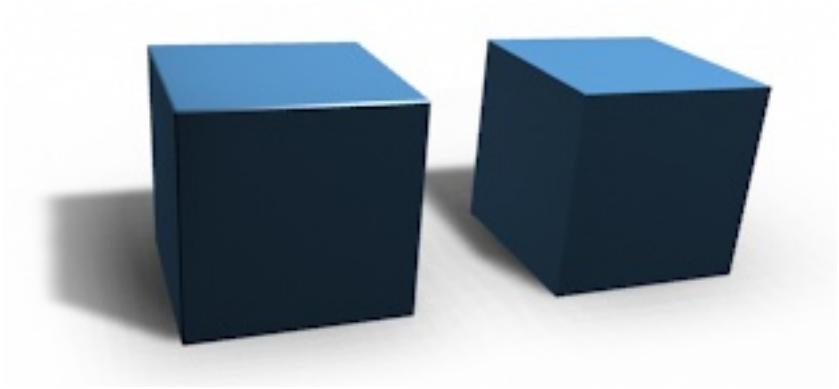


Fig. 2.606: Cubes with and without bevel.

Usage

The *Bevel* tool works only on selected edges. It will recognize any edges included in a vertex or face selection as well, and perform the bevel the same as if those edges were explicitly selected. In "vertex only" mode, the *Bevel* tool works on selected vertices instead of edges. The *Bevel* tool smooths the edges and/or "corners" (vertices) by replacing them with faces making smooth profiles with a specified number of *segments* (see the options below for details about the bevel algorithm).

Use `Ctrl-B` or a method listed above to run the tool. Move the mouse to interactively specify the bevel offset, and scroll the `Wheel` to increase or decrease the number of segments. (see below)

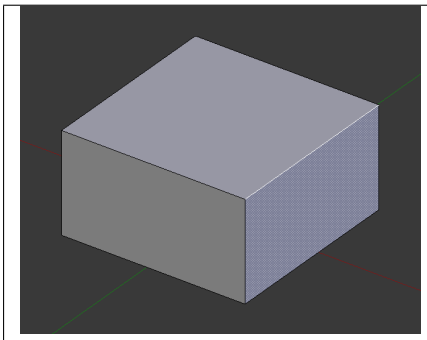


Fig. 2.607: Selected edge before beveling.

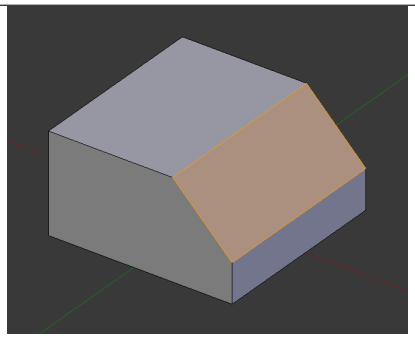


Fig. 2.608: Result of bevel (one segment).

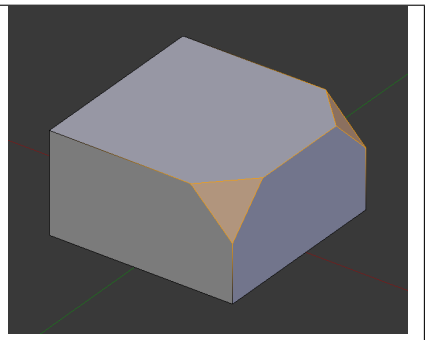


Fig. 2.609: Result of bevel (vertex only).

Note: Normal (edge) beveling only works on edges that have exactly two faces attached to them. Vertex bevel has no such restriction.

Options

Amount You can change the bevel amount by moving the mouse towards and away from the object, a bit like with transform tools. The exact meaning of the value depends on the *Amount Type* option (see below). As usual, the scaling can be controlled to a finer degree by holding `Shift` to scale in 0.001 steps. `LMB` finalizes the operation, `RMB` or `Esc` aborts the action.

Amount Type Selects how the *Amount* value controls the size of the bevel. According to the selection, the amount is:

- *Offset* – The distance of a new edge from the original.
- *Width* – The width of the bevel face.
- *Depth* – The perpendicular distance from the original edge to the bevel face.
- *Percent* – The percentage of the length of adjacent edges that the new edges slide.

Segments The number of segments in the bevel can be defined by scrolling the mouse `Wheel` to increase or decrease this value. The greater the number of segments, the smoother the bevel.

Alternatively, you can manually enter a segment number value while using the tool, or in the Mesh Tool options panel after using the tool.

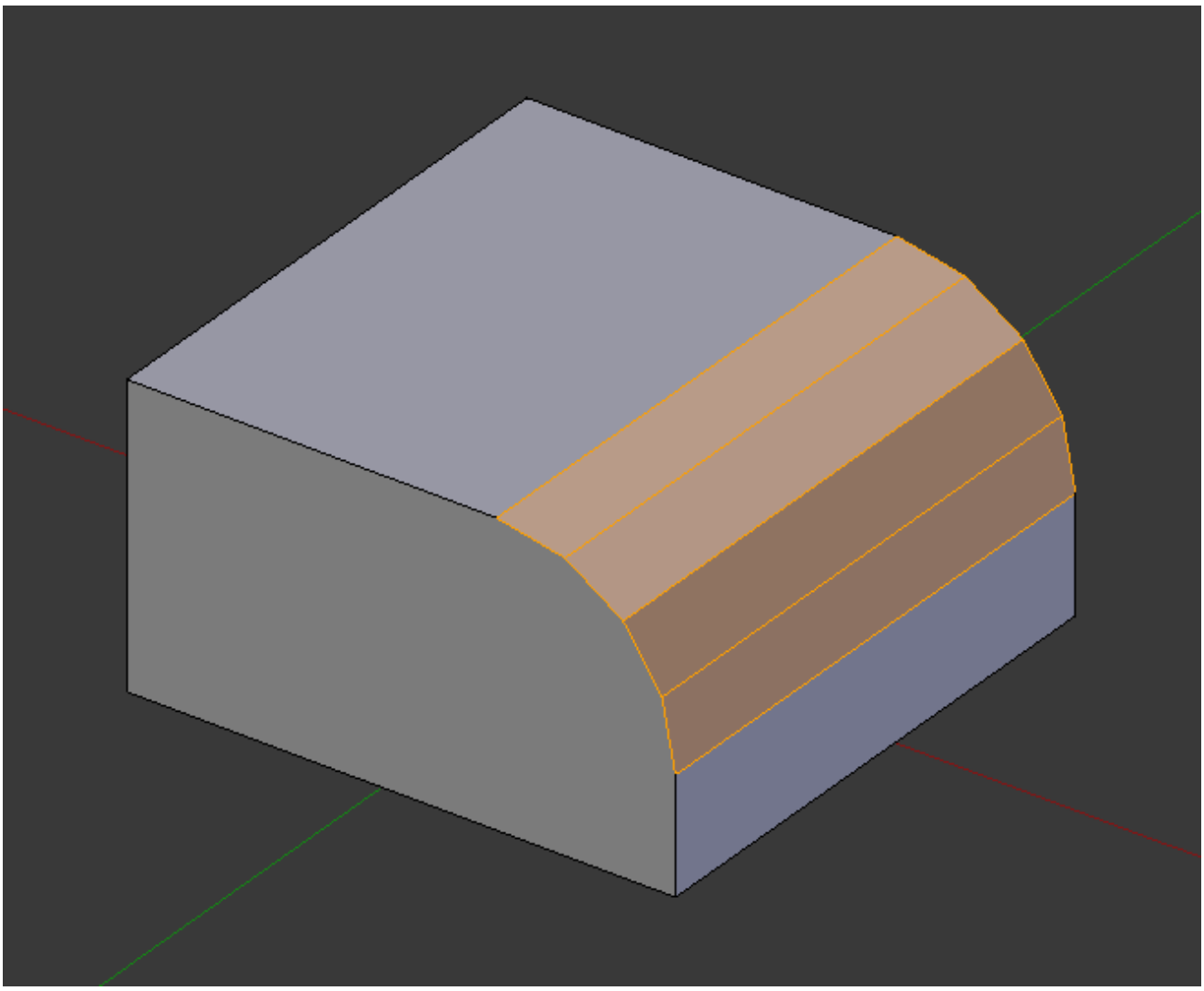


Fig. 2.610: Bevel with four segments.

Profile This is a number between 0 and 1 that controls the shape of the profile (side view of a beveled edge). The default value, 0.5, gives a circular arc (if the faces meet at right angles). Values less than that give a flatter profile, with 0.25 being exactly flat, and values less than that giving a concave bevel. Values more than 0.5 give a more "bulged-out" profile.

Vertex Only When selected, the tool is in "vertex only" mode, and only vertices will be beveled.

Clamp Overlap When selected, the bevel amount is not allowed to go larger than an amount that causes overlapping collisions with other geometry.

Material The *Material* number specifies which material should be assigned to the new faces created by the *Bevel* tool. With the default, -1, the material is inherited from the closest existing face ("closest" can be a bit ambiguous). Otherwise, the number is the slot index of the material to use for all newly created faces.

Examples

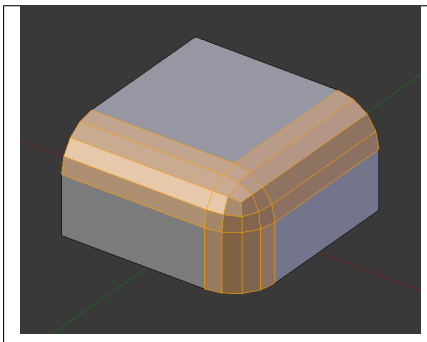


Fig. 2.611: Result of beveling multiple edges.

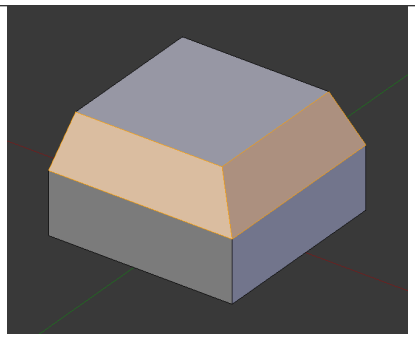


Fig. 2.612: Another example of beveling multiple edges.

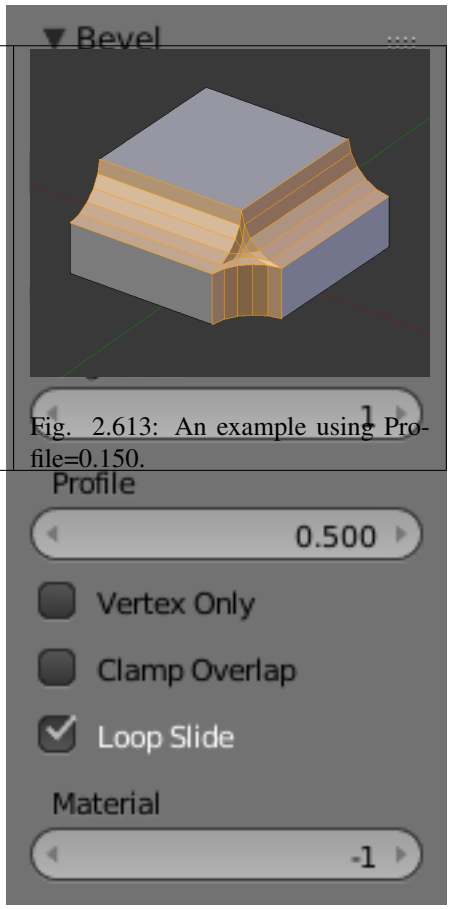


Fig. 2.613: An example using Profile=0.150.

See also:

Bevel Modifier

The *Bevel Modifier* is a non destructive alternative to the bevel tool.

Mesh Clean-up

These tools are to help cleanup degenerate geometry and fill in missing areas of a mesh.

Decimate Geometry

Reference

Mode: Edit Mode
Menu: *Mesh* → *Clean up* → *Decimate Geometry*

The Decimate modifier allows you to reduce the vertex/face count of a mesh with minimal shape changes.

Ratio Ratio of triangles to reduce to.

Vetex Group Use the active vertex group as an influence.

Weight Strength of the vertex group.

Invert Inverts the vertex group.

Symmetry Maintain symmetry on either the *X*, *Y*, or *Z* axis.

See also:

This tool works similar to the *Decimate Modifier*.

Fill Holes

Reference

Mode: Edit Mode
Menu: *Mesh* → *Clean up* → *Fill Holes*

This tool is can take a large selection and detect the holes in the mesh, filling them in.

This is different from the face creation operator in three important respects:

- Holes are detected, so there is no need to manually find and select the edges around the holes.
- Holes can have a limit for the number of sides (so only quads or tris are filled in for example).
- Mesh data is copied from surrounding geometry (UVs, vertex-colors, multi-res, all layers), since manually creating this data is very time consuming.

Split Non-Planar Faces

Reference

Mode: Edit Mode
Menu: *Mesh* → *Clean up* → *Split Non-Planar Faces*

This tool avoids ambiguous areas of geometry by splitting non-flat faces when they are bent beyond a given limit.

Delete Loose Geometry

Reference

Mode: Edit Mode
Menu: *Mesh* → *Clean up* → *Delete Loose*

This tool removes disconnected vertices and edges (optionally faces).

Degenerate Dissolve

Reference

Mode: Edit Mode
Menu: *Mesh* → *Clean up* → *Degenerate Dissolve*

This tool collapses / removes geometry which you typically will not want.

- Edges with no length.
- Faces with no areas (faces on a point or thin faces).
- Face corners with no area.

Miscellaneous Editing Tools

Sort Mesh Elements

This tool (available from the *Specials*, *Vertices*, *Edges* and *Faces* menus) allows you to reorder the matching selected mesh elements, following various methods. Note that when called from the *Specials* menu, the affected element types are the same as the active select modes.

View Z Axis Sort along the active view's Z axis, from farthest to nearest by default (use *Reverse* if you want it the other way).

View X Axis Sort along the active view's X axis, from left to right by default (again, there is the *Reverse* option).

Cursor Distance Sort from nearest to farthest away from the 3D cursor position (*Reverse* also available).

Material Sort faces, and faces only, from those having the lowest material's index to those having the highest. Order of faces inside each of those "material groups" remains unchanged. Note that the *Reverse* option only reverses the order of the materials, *not* the order of the faces inside them.

Selected Move all selected elements to the beginning (or end, if *Reverse* enabled), without affecting their relative orders. Warning: This option will also affect **unselected** elements' indices!

Randomize Randomizes indices of selected elements (*without* affecting those of unselected ones). The seed option allows you to get another randomization – the same seed over the same mesh/set of selected elements will always give the same result!

Reverse Simply reverse the order of the selected elements.

Note: To enable viewing indeces:

Type "bpy.app.debug = True" into the Python Console and a checkbox will appear in the properties region under *Mesh Display* → *Edge Info* → *Indices*

Join

Reference

Mode: Object Mode

Menu: *Object* → *Join*
Hotkey: `Ctrl-J`

Joining merges all objects into the last selected *Active* object.

This tools works for meshes, curves, surfaces, metas and armature object types.

Note: Object data has many attributes which may be handled when joining.

Materials, vertex-groups, UV and Vertex layers will be merged.

Modifiers, constraints, groups and parent relationships are ignored when joining and will not be applied to the active object.

Separate

Reference

Mode: Edit Mode
Menu: *Mesh* → *Vertices* → *Separate*
Hotkey: `P`

At some point, you will come to a time when you need to cut parts away from a mesh to be separate.

To separate an object, the vertices (or faces) must be selected and then separated, though there are several different ways to do this.

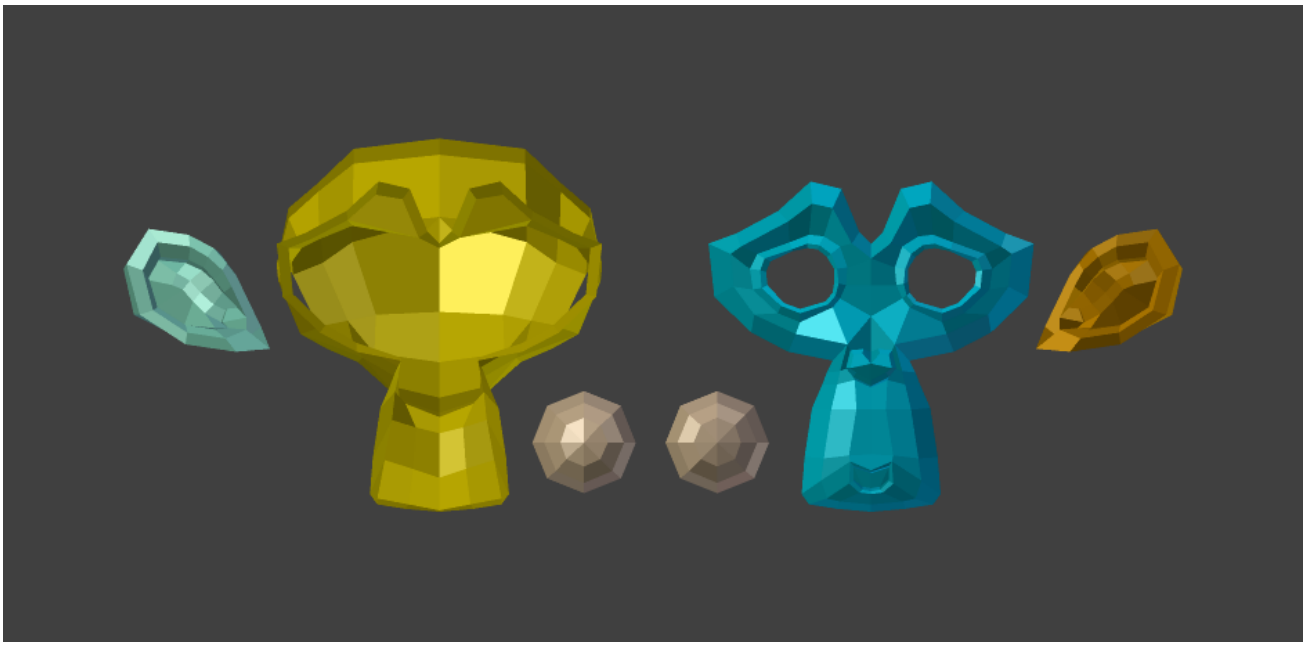


Fig. 2.614: Suzanne dissected neatly.

Selected This option separates the selection to a new object.

All Loose Parts Separates the mesh in its unconnected parts.

By Material Creates separate mesh objects for each material.

Data Transfer

The *Data Transfer* tool transfers several types of data from one mesh to another. Data types include vertex groups, UV layers, vertex colors, custom normals...

Transfer works by generating a mapping between source mesh's items (vertices, edges, etc.) and destination ones, either on an one-to-one basis, or mapping several source items to a single destination one by interpolated mapping.

Data

Reference

Mode: Object Mode
Panel: *Object Tools* → *Data*
Hotkey: `Shift-Ctrl-T`

Transfers layout of data layer(s) from active to selected meshes.

Freeze Operator Prevent changes to settings to re-run the operator. This is useful if you are editing several settings at once with heavy geometry

Data Type Which data to transfer.

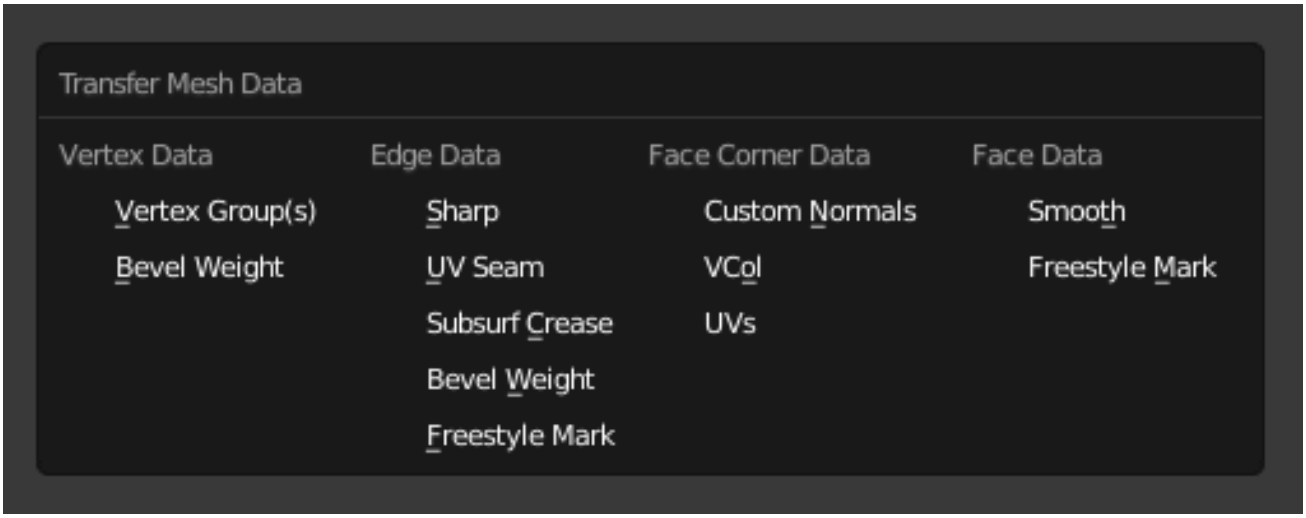


Fig. 2.615: Data types.

Create Data Add data layers on destination meshes if needed.

Vertex Mapping Method used to map source vertices to destination ones. Because the options change depending on the *Data Type* options are explained in *Vertex Mapping* below.

Vertex Mapping

Topology The simplest option, expects both meshes to have identical number of items, and match them by order (indices). Useful e.g. between meshes that were identical copies, and got deformed differently.

One-To-One Mappings Those always select only one source item for each destination one, often based on shortest distance.

Vertices

Nearest Vertex Uses source's nearest vertex.

Nearest Edge Vertex Uses source's nearest vertex of source's nearest edge.

Nearest Face Vertex Uses source's nearest vertex of source's nearest face.

Edges

Nearest Vertices Uses source's edge which vertices are nearest from destination edge's vertices.

Nearest Edge Uses source's nearest edge (using edge's midpoints).

Nearest Face Edge Uses source's nearest edge of source's nearest face (using edge's midpoints).

Face Corners A face corner is not a real item by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, ...) aspects are used to match them together.

Nearest Corner and Best Matching Normal Uses source's corner having the most similar *split* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner and Best Matching Face Normal Uses source's corner having the most similar *face* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner of Nearest Face Uses source's nearest corner of source's nearest face.

Faces

Nearest Face Uses source's nearest face.

Best Normal-Matching: Uses source's face which normal is most similar with destination one.

Interpolated Mappings Those use several source items for each destination one, interpolating their data during the transfer.

Vertices

Nearest Edge Interpolated Uses nearest point on nearest source's edge, interpolates data from both source edge's vertices.

Nearest Face Interpolated Uses nearest point on nearest source's face, interpolates data from all that source face's vertices.

Projected Face Interpolated Uses point of face on source hit by projection of destination vertex along its own normal, interpolates data from all that source face's vertices.

Edges

Projected Edge Interpolated This is a sampling process. Several rays are cast from along the destination's edge (interpolating both edge's vertex normals),

and if enough of them hit a source's edge, all hit source edges' data are interpolated into destination one.

Face Corners A face corner is not a real item by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, ...) aspects are used to match them together.

Nearest Face Interpolated Uses nearest point of nearest source's face, interpolates data from all that source face's corners.

Projected Face Interpolated Uses point of face on source hit by projection of destination corner along its own normal, interpolates data from all that source face's corners.

Faces

Projected Face Interpolated This is a sampling process. Several rays are cast from the whole destination's face (along its own normal), and if enough of them hit a source's face, all hit source faces' data are interpolated into destination one.

Auto Transform Automatically computes the transformation to get the best possible match between source and destination meshes.

Object Transform Evaluate source and destination meshes in global space.

Only Neighbor Geometry Source elements must be closer than given distance from destination one.

Max Distance Maximum allowed distance between source and destination element (for non-topology mappings).

Ray Radius Width of rays. Useful when raycasting against vertices or edges.

Mix Mode How to affect destination elements with source values.

All Replaces everything in destination (note that *Mix Factor* is still used).

Above Threshold Only replaces destination value if it is above given threshold *Mix Factor*. How that threshold is interpreted depends on data type, note that for boolean values this option fakes a logical AND.

Below Threshold Only replaces destination value if it is below given threshold *Mix Factor*. How that threshold is interpreted depends on data type, note that for boolean values this option fakes a logical OR.

Mix, Add, Subtract, Multiply Apply that operation, using mix factor to control how much of source or destination value to use. Only available for a few types (vertex groups, vertex colors).

Mix Factor How much of the transfered data gets mixed into existing one (not supported by all data types).

Data Layout

Reference

Mode: Object Mode
Panel: *Object Tools* → *Data layout*

Transfers layout of data layer(s) from active to selected meshes.

Data Type Which data to transfer.

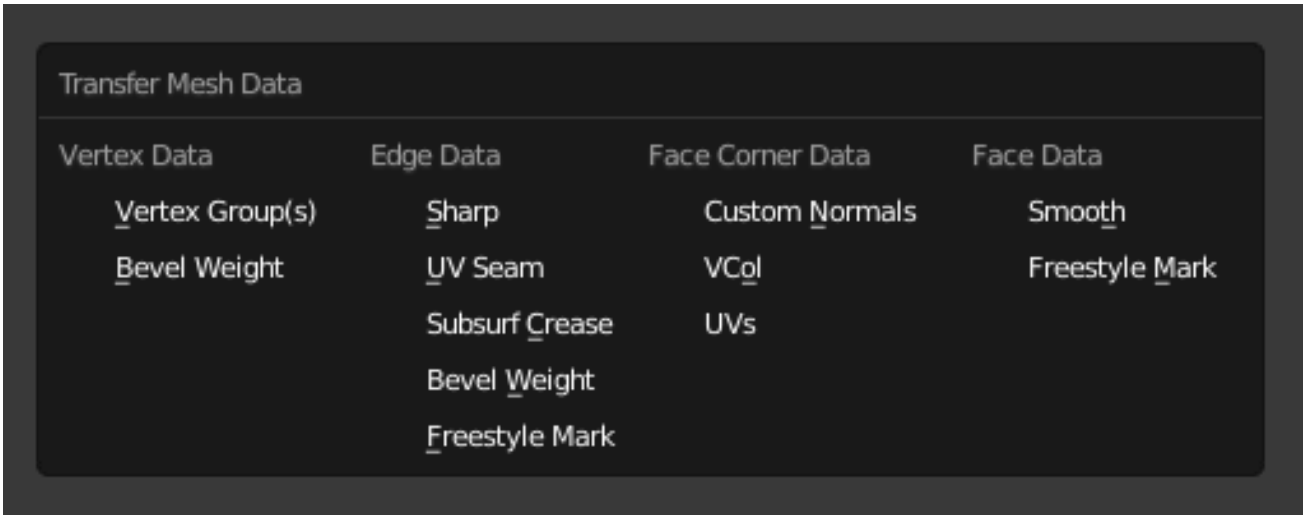


Fig. 2.616: Data types.

Exact Match Also Delete some data layers from destination if necessary, so that it matches the source exactly.

Source Layers Selection Which layers to transfer, in case of multi-layer types.

Active Layer Only transfer the active data layer.

All Layers Transfer all data layers.

Destination Layers Matching How to match source and destination layers.

By Name Match target data layers to affect by name.

By Order Match target data layers to affect by order (indices).

Properties

Object Data

Normals

Fig. 2.617: Normals panel

Auto Smooth Edges where an angle between the faces is smaller than specified in the *Angle* button will be smoothed, when shading of these parts of the mesh is set to smooth.

Angle Angle number button.

Double Sided Lighting with positive normals on the backside of the mesh in the viewport (OpenGL).

Example

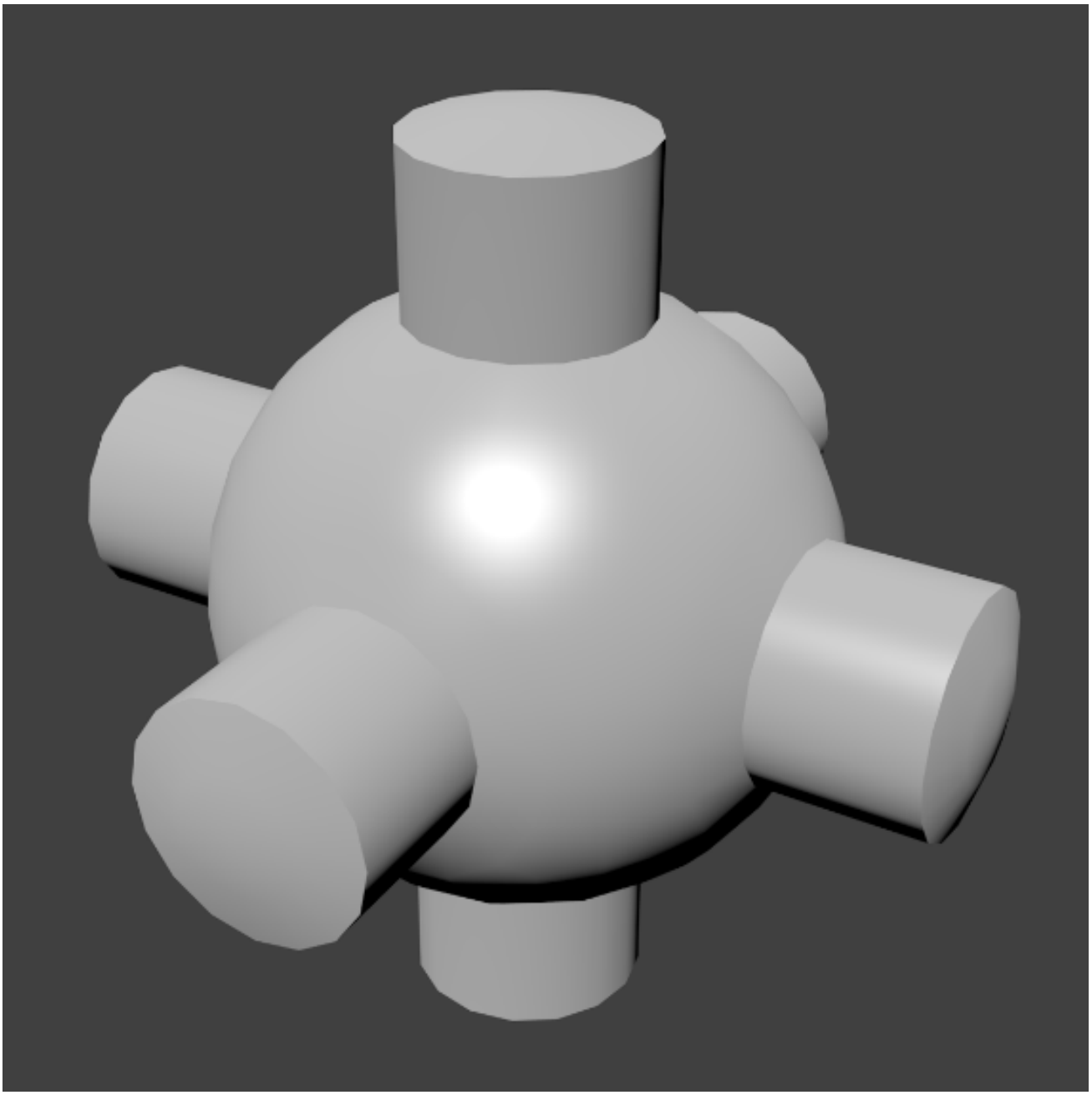


Fig. 2.618: Example mesh with *Auto Smooth* enabled.

Texture Space

This are settings of the texture space used by the generated texture mapping.
The visualization of the texture space can be activated in the *Display*.

Texture Mesh

Auto Texture Space Location, Size

Vertex Groups

See *Vertex Groups Panel*.

TODO.

Shape Keys

See *Shape Keys Panel*.

TODO.

UV Maps

See *UV Maps Panel*.

TODO.

Vertex Colors

TODO.

Geometry Data

TODO.

Vertex Groups

Introduction

Vertex Groups are mainly used to tag the vertices belonging to parts of a Mesh Object or *Lattice*. Think of the legs of a chair or the hinges of a door, or hands, arms, limbs, head, feet, etc. of a character. In addition you can assign different *weight values* (in the range [0.0, 1.0]) to the vertices within a Vertex Group. Hence Vertex Groups are sometimes also named *Weight Groups*.

Fig. 2.619: The Vertex Group Panel.

Vertex Groups are most commonly used for Armatures (See also *Skinning Mesh Objects*). But they are also used in many other areas of Blender, like for example:

- Shape keys
- Modifiers
- Particle Generators
- Physics Simulations

Many more usage scenarios are possible. Actually you can use Vertex Groups for whatever makes sense to you. In some contexts Vertex Groups can also be automatically generated (i.e. for rigged objects). However, in this section we will focus on manually created (user-defined) Vertex Groups.

Note: Vertex groups only apply to Mesh and Lattice Objects

Any other Object type has no vertices, hence it cannot have Vertex Groups.

Typical Usage Scenarios for Vertex groups

Skinning an armature If you want to animate your mesh and make it move, you will define an armature which consists of a bunch of bones. Vertex Groups are used to associate parts of the Mesh to Bones of the Armature, where you can specify an influence *weight* in the range (0.0 - 1.0) for each vertex in the Vertex Group.

Working with Modifiers Many modifiers contain the ability to control the modifier influence on each vertex separately. This is also done via Vertex Groups and the weight values associated to the vertices.

Quickly select/edit/hide parts of a mesh By defining mesh regions with Vertex Groups you can easily select entire parts of your mesh with three clicks and work on them in isolation without having to create separate objects. With the hide function you can even remove a vertex group from the view (for later unhide).

Cull out and duplicate parts of a mesh Consider modeling a Lego block. The most simple brick consists of a base and a stud (the bump to connect the bricks together). To create a four-stud block, you would want to be able to easily select the stud vertices, and, still in *Edit Mode*, duplicate them and position them where you want them.

Vertex Groups Panel

Vertex Groups are maintained within the *Object Data* Properties Editor, and there in the *Vertex Groups* panel.

Fig. 2.620: The Vertex Group Panel.

Active Vertex Group A *List Views & Presets*.

Lock Locks the group from being edible. You can only rename or delete the group.

Add + Create a empty vertex group.

Specials

Sort Vertex Groups Sorts Vertex Groups alphabetically.

Copy Vertex Group Add a Copy of the active Vertex Group as a new Group. The new group will be named like the original group with "_copy" appended at the end of its name. And it will contain associations to exactly the same verts with the exact same weights as in the source vertex group.

Copy Vertex Groups to Linked Copy Vertex Groups of this Mesh to all linked Objects which use the same mesh data (all users of the data).

Copy Vertex Group to Selected Copy all Vertex Groups to other Selected Objects provided they have matching indices (typically this is true for copies of the mesh which are only deformed and not otherwise edited).

Mirror Vertex Group Mirror all Vertex Groups, flip weights and/or names, editing only selected vertices, flipping when both sides are selected; otherwise copy from

unselected. Note this function will be reworked (and fully documented) in a future release.

Remove from All Groups (not available for locked groups) Unassigns the selected Vertices from all groups. After this operation has been performed, the verts will no longer be contained in any vertex group.

Clear Active Group Remove all assigned vertices from the active Group. The group is made empty. Note that the vertices may still be assigned to other Vertex Groups of the Object. (not available for locked groups).

Delete All Groups Remove all Vertex Groups from the Object.

Lock All Lock all groups.

Unlock All Unlock all groups.

Lock_Invert All Invert Group Locks.

Editing Vertex Groups

When you switch either to *Edit Mode* or to *Weight Paint Mode* Vertex weights can be edited. The same operations are available in the 3D Views menu *Mesh* → *Vertices* → *Vertex Groups* or `Ctrl-G`.

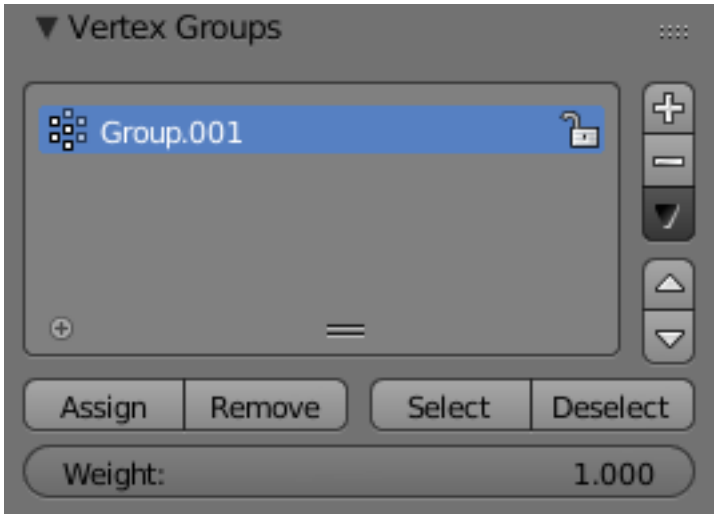


Fig. 2.621 Vertex Group Panel in Edit or Weight Paint Mode.

Assign To assign the Selected vertices to the Group with the weight as defined in the *Weight* (see below).

Remove To Remove the selected vertices from the Group (and thus also delete their weight values).

Select To Select all vertices contained in the Group.

Deselect To deselect all verts contained in the group.

Weight The weight value that gets assigned to the selected vertices.

Hint: Multiple objects sharing the same mesh data have the peculiar property that the group names are stored on the object, but the weights in the mesh. This allows you to name groups differently on each object, but take care because removing a vertex group will remove the group from all objects sharing this mesh.

Assigning a Vertex Group

Creating Vertex Groups

Vertex Groups are maintained within the *Mesh* tab (1)
in the Properties Editor. As long as no Vertex groups are defined (the default for new Mesh Objects), the Panel is empty (2).

You create a vertex group by LMB on the *Add* button (+) on the right Panel border (3). Initially the group is named "Group" (or "Group.nnn" when the name already exists) and gets displayed in the Panel (2) (see next image).

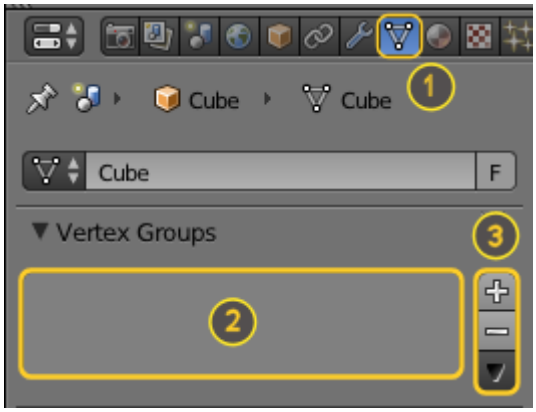


Fig. 2.622: Empty Vertex Group Panel.

Vertex Groups Panel Controls

Once a new Vertex Group has been added, the new Group appears in the vertex Groups panel. There you find three clickable elements:

Group Name The Groupname can be changed by double clicking LMB on the name itself. Then you can edit the name as you like.

Plus Icon When the little icon in the left lower corner can be clicked, a new row opens up where you can enter a search term. This becomes handy when the number of vertex groups gets big.

Drag Handle If you have a large number of vertex groups and you want to see more then a few Groups, you can LMB on the small drag handle to tear the vertex groups list larger or smaller.

Active Group When a Vertex Group is created, then it is also automatically marked as the *Active Group*. This is indicated by setting the background of the panel entry to a light blue color. If you have two or more groups in the list, then you can change the active group by LMB on the corresponding entry in the Vertex Group panel.

Deleting Vertex Groups

You delete a Vertex Group by first making it the active group (select it in the panel) and then LMB the *Remove* button (-) at the right Panel border.

Deleting a Vertex Group only deletes the vertex assignments to the Group. The vertices themselves are not deleted.

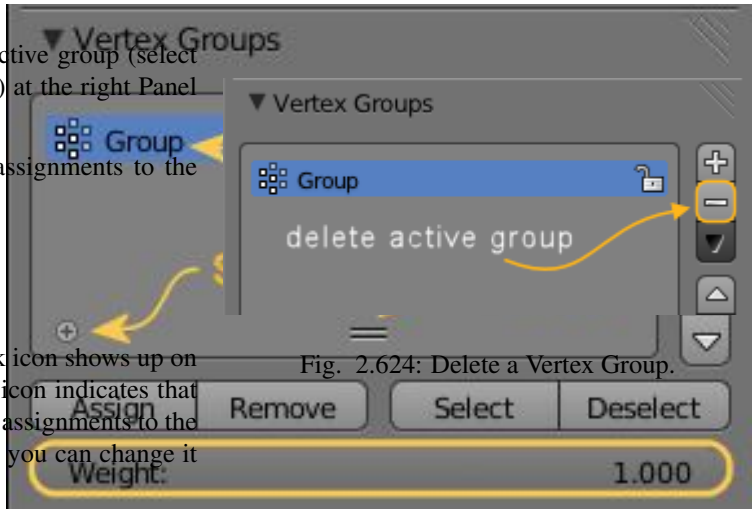


Fig. 2.624: Delete a Vertex Group.

Locking Vertex Groups

Right after creation of a Vertex Group, an open lock icon shows up on the right side of the Vertex Group List entry. This icon indicates that the Vertex Group can be edited. You can add vertex assignments to the group or remove assignments from the group. And you can change it with the weight paint brushes, etc.

When you click on the icon, it changes to a closed lock icon and all vertex group modifications get disabled. You can only rename or delete the group, and unlock it again. No other operations are allowed on locked Vertex Groups, thus all corresponding function buttons become disabled for locked Vertex Groups.

Fig. 2.623: One Vertex Group.

Working with Content of Vertex Groups

Assigning Verts to a Group

Fig. 2.625: Lock a Vertex Group.

You add vertices to a group as follows:

- Select the group from the group list, thus make it the Active Group (1).
- From the 3D View select `Shift-RMB` all vertices that you want to add to the group.
- Set the weight value that shall be assigned to all selected verts (2).
- `LMB` the *Assign* button to assign the selected verts to the active group using the given weight (3).

Note that weight Assignment is not available for locked Vertex Groups. The Assign button is grayed out in that case.

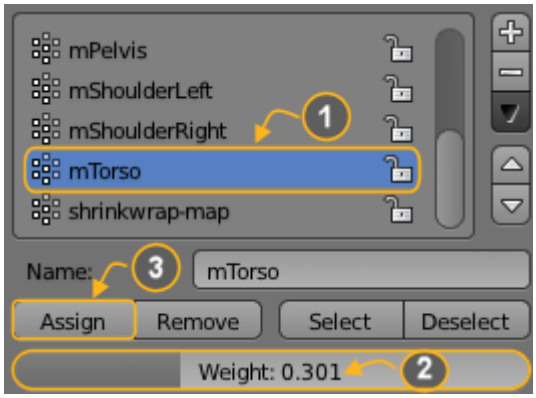


Fig. 2.626: Assign weights to active group.

Note: Assign is additive

The *Assign* button only adds the currently selected vertices to the active group. Vertices already assigned to the group are not removed from the group.

Also keep in mind that a vertex can be assigned to multiple groups.

Checking Assignments

To be sure the selected verts are in the desired Vertex Group, you can try press the deselect button. If the vertices remain selected then they are not yet in the current Vertex Group.

At this point you may assign then, but take care since all selected vertices will have their weight set to the value in the *Weight:* field.

Removing assignments from a Group

You remove vertices from a group as follows:

- Select the group from the group list (make it the active group).
- Select all vertices that you want to remove from the group.
- Press the *Remove* button.

Note that Removing weight Assignments is not available for locked Vertex Groups. The Remove button is grayed out in that case.

Using Groups for Selecting/Deselecting

You can quickly select all assigned vertices of a group:

- (optionally) press `A` once or twice to unselect all vertices.
- Select the group from the group list (make it the active group).
- When you now `LMB` click the *Select* button, then the vertices assigned to the active group will be selected and highlighted in the 3D View.
- When you `LMB` click the *Deselect* button instead, then the vertices assigned to the active group will be deselected in the 3D View.

Note: Selecting/Deselecting is additive

If you already have verts selected in the 3D View, then selecting the verts of a group will add the verts but also keep the already-selected verts selected. Vice versa, deselecting the verts of a vertex group will only deselect the verts assigned to the group and keep all other verts selected.

Finding ungrouped Verts

You can find ungrouped vertices as follows:

- Press A once or twice to unselect all vertices.
- In the header of the 3D View: Navigate to *Select* → *Ungrouped verts*

Weight Editing

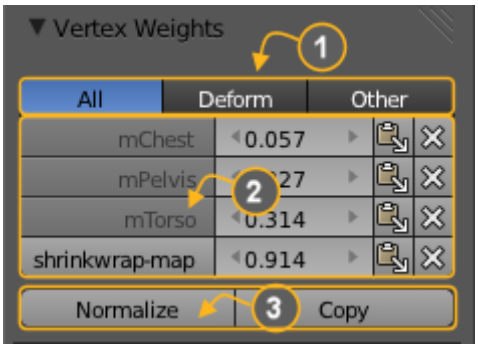


Fig. 2.627: Vertex Weights Panel.
Vertex Group Categories (1), Weight Table (2), Function bar (3).

As mentioned before in *Vertex Groups* each entry in a Vertex Group also contains a weight value in the range of (0.0 to 1.0). Blender provides a *Vertex Weights* panel from where you can get (and edit) information about the weight values of each Vertex of a mesh. That is: to which Vertex Groups the vertex is assigned with which weight value.

The Vertex Weights panel can be found in the right Properties region of the 3D View. It is available in Edit Mode and in Weight Paint Mode (when Vertex Selection masking is enabled as well). The panel is separated into the sections:

- *Vertex Group Categories*
- *Weight Table*
- *Function bar*

Vertex Group Categories

Actually we do not have any strict categories of Vertex Groups in Blender. Technically they all behave the same way. However, we can identify two implicit categories of Vertex Groups:

Deform Groups

These Vertex Groups are sometimes also named *Weight Groups* or *Weight Maps*. They are used for defining the weight tables of Armature bones. All Deform Groups of an Object are strictly related to each other via their weight values.

Strictly speaking, the sum of all deform weights for any vertex of a mesh should be exactly 1. 0. In Blender this constraint is a bit relaxed (see below). Nevertheless, Deform Groups should always be seen as related to each other. Hence we have provided a filter that allows restricting the Vertex Weight panel to display only the Deform bones of an Object.

Other Groups

All other usages of Vertex Groups are summarized into the *Other* category. These vertex groups can be found within Shape keys, Modifiers, etc... There is really no good name for this category, so we kept it simple and named it *Other*.

Weight Table

The Weight Table shows all weights associated to the *active vertex*. Note that a vertex does not necessarily have to be associated to any vertex groups. In that case the Vertex Weights Panel is not displayed.

Tip: The active Vertex

That is the most recently selected vertex. This vertex is always highlighted so that you can see it easily in the mesh. If the active Vertex does not have weights, or there is no active vertex selected at the moment, then the Vertex Weights Panel disappears.

Each row in the Weight table contains four active elements:

Fig. 2.628: Change Active Group.

Set the Active Group

As soon as you select any of the Vertex Group Names in the Weight table, the referenced Vertex Group becomes the new Active group.

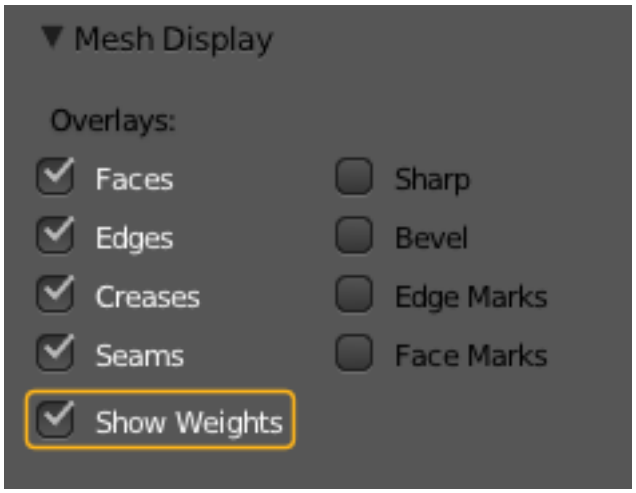


Fig. 2.629: Enable display of Weights in Edit Mode.

Display Weights in Edit Mode

When you are in edit mode, you can make the Weights of the active Group visible on the mesh:

Search the *Mesh Display* panel in the Properties region. And there enable the *Show Weights* option. Now you can see the weights of the active Vertex Group displayed on the mesh surface.

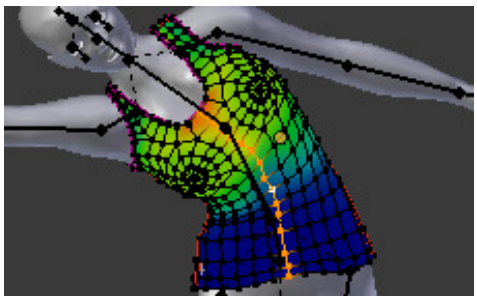


Fig. 2.630: Weights in Edit Mode.

Edit Weights in Edit Mode

It is now very easy to work with Vertex Groups in Edit Mode. All edit options of the mesh are available and you have direct visual control over how your Weights change when you edit the weight values.

Fig. 2.631: Change Weight Value.

Change a weight

You can either enter a new weight value manually (click on the number and edit the value), or you can change the weight by LMB and while holding down the mouse button, drag right or left to increase/decrease the weight value. You also can use the right/left arrows displayed around the weight value to change the weight in steps.

Fig. 2.632: Paste weights.

Paste a weight to other verts

LMB the Paste Icon allows you to forward a single weight of the active Vertex to all selected vertices. But note that weights are only pasted to verts which already have a weight value in the affected Vertex Group.

Fig. 2.633: Delete weights.

Delete a weight from a Group

LMB the Delete Icon will instantly remove the weight from the active vertex. Thus the entire row disappears when you click on the delete icon.

Function bar

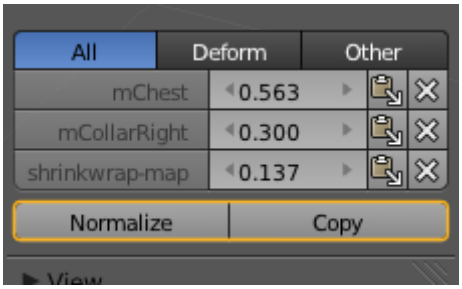


Fig. 2.634: Vertex Weights panel.

The function bar contains two functions:

Normalize Normalizes the weights of the active Vertex. That is all weights of the active vertex are recalculated such that their relative weight is maintained and the weight sum is 1.0.

Copy Copies all weights defined for the active Vertex to all selected Verts. Thus all previously defined weights are overwritten.

Tip: The filter setting is respected

Note that both functions only work on the Vertex Groups currently displayed in the Weights Table. So if for example only the *Deform weights* are displayed, then Normalize and Copy only affect the Deform bones.

About locked Vertex Groups

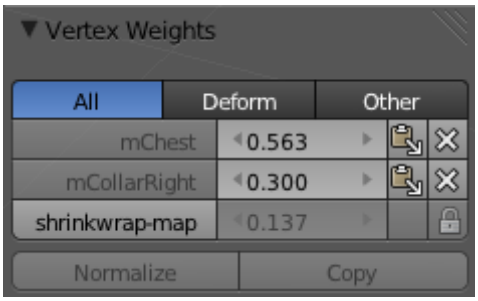


Fig. 2.635: Vertex Weights panel Locked.

Whenever a Weight Group is locked, all data changing functions get disabled:

- Normalize the vertex Weights.
- Copy the Vertex weights.
- Change the Weight of the active vert.
- Paste to selected verts.

Tip: The filter setting is respected

If you have for example all deform weight groups unlocked and all other vertex groups locked, then you can safely select *Deform* from the Filter row and use all available functions from the Weight table again.

Mesh Display

Reference

Mode: Edit Mode
Panel: *Properties region* → *Mesh Display*

This panel is available only in edit mode, when the object being edited is a mesh.

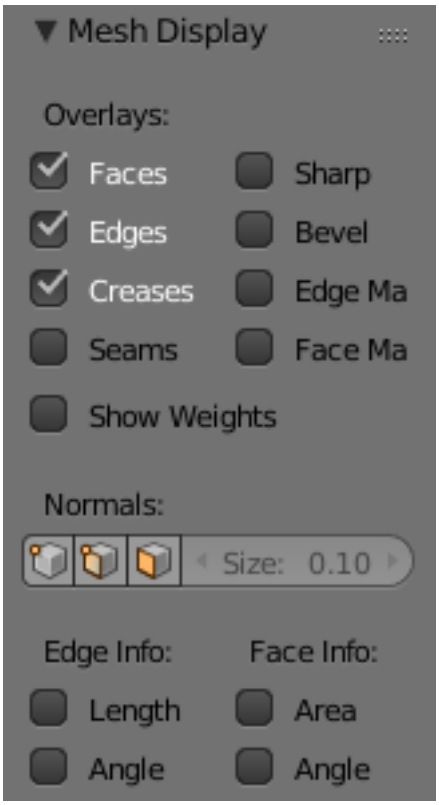


Fig. 2.636: Mesh Display Panel.

Overlays

The Overlays section provides controls for highlighting parts of the mesh.

Edges Toggles the option to see the selected edges highlighted. If enabled the edges that have both vertices selected will be highlighted. This only affects in vertex selection mode and when *UV Unwrapping*.

Faces Defines if the selected faces will be highlighted in the *3D View*. This affects all selection modes.

Creases and Bevel Weight Highlights edges marked with a crease weight for the *Subdivision Surface Modifier* and/or a bevel weight for the *Bevel Modifier*, respectively. In both cases, the higher the weight, the brighter the highlight.

Seams and Sharp Highlights edges marked as a UV seam for unwrapping and/or sharp edges for the *Edge Split Modifier*

Edge Marks and Face Marks Used by Freestyle.

Show Weight Displays the vertex weights as a texture.

Normals

Show Displays the normals of faces and/or vertices using the Face and Vertex checkboxes.

Vertex, Loop, Face

Size You can also change the length of the axis that points the direction of the normal.

Show Extra

Numerical measures of the selected elements on screen. The *Units* can be set in the Scene tab.

Edge Length and Edge Angle Shows the length and angle of the selected edges.

Face Area and Face Angle Show the area and angles of the selected faces.

Mesh Analysis

Reference

Mode: Edit Mode
Panel: *Properties region* → *Mesh Analysis*

Mesh analysis is useful for displaying attributes of the mesh, that may impact certain use cases.

The mesh analysis works in *Edit Mode* and *Solid* Viewport shading. It shows areas with a high value in red, and areas with a low value in blue. Geometry outside the range is displayed gray.

Currently the different modes target 3D-printing as their primary use.

Warning: There are some known limitations with mesh analysis:

- Currently only displayed with deform modifiers.
- For high-poly meshes is slow to use while editing the mesh.

Overhang

Extrusion 3D printers have a physical limit to the overhang that can be printed, this display mode shows the overhang with angle range and axis selection.

Thickness

Printers have a limited *wall-thickness* where very thin areas cannot be printed, this test uses ray casting and a distance range to the thickness of the geometry.

Intersections

Another common cause of problems for printing are intersections between surfaces, where the inside/outside of a model cannot be reliably detected.

Unlike other display modes, intersections have no variance and are either on or off.

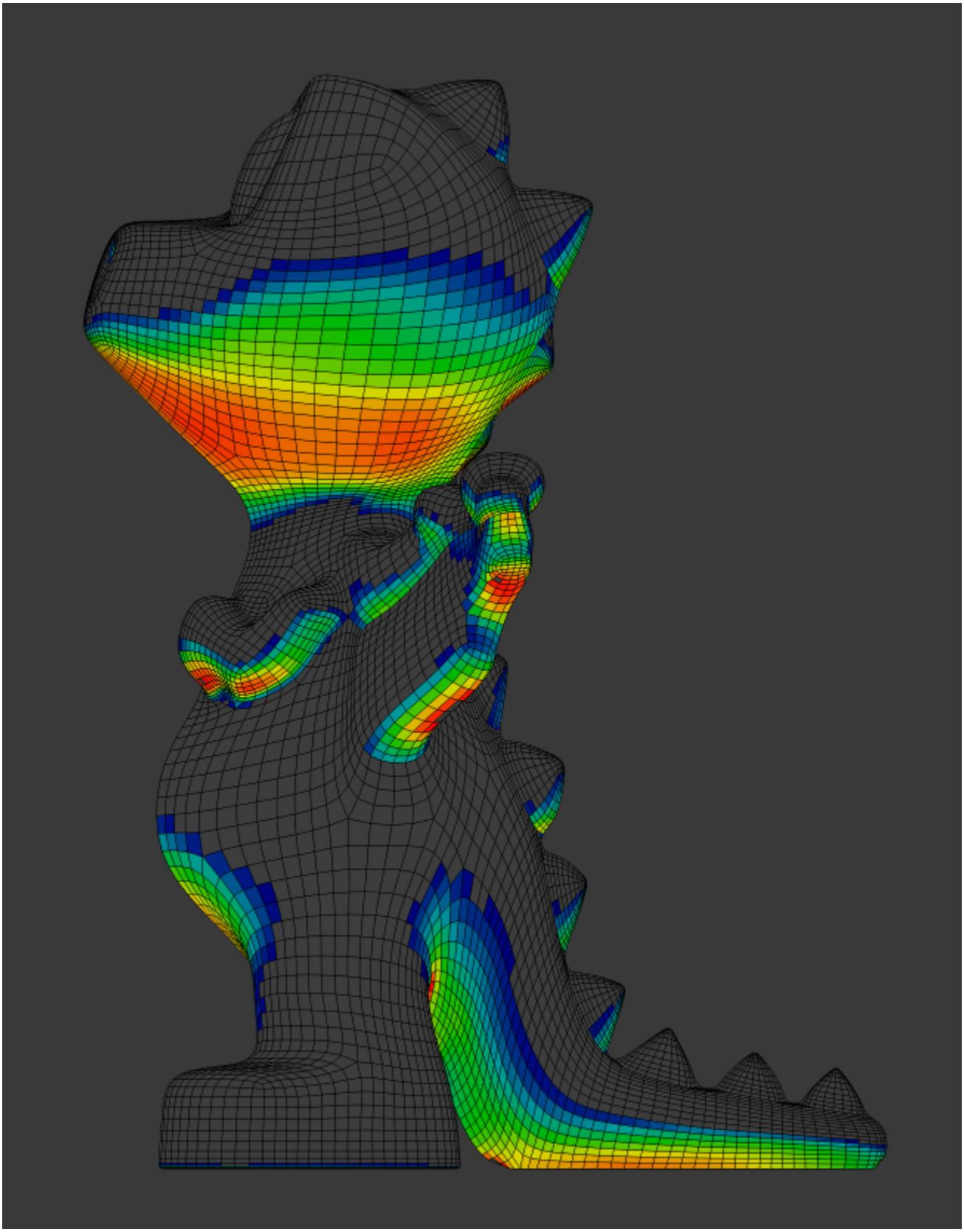


Fig. 2.637: Overhang.

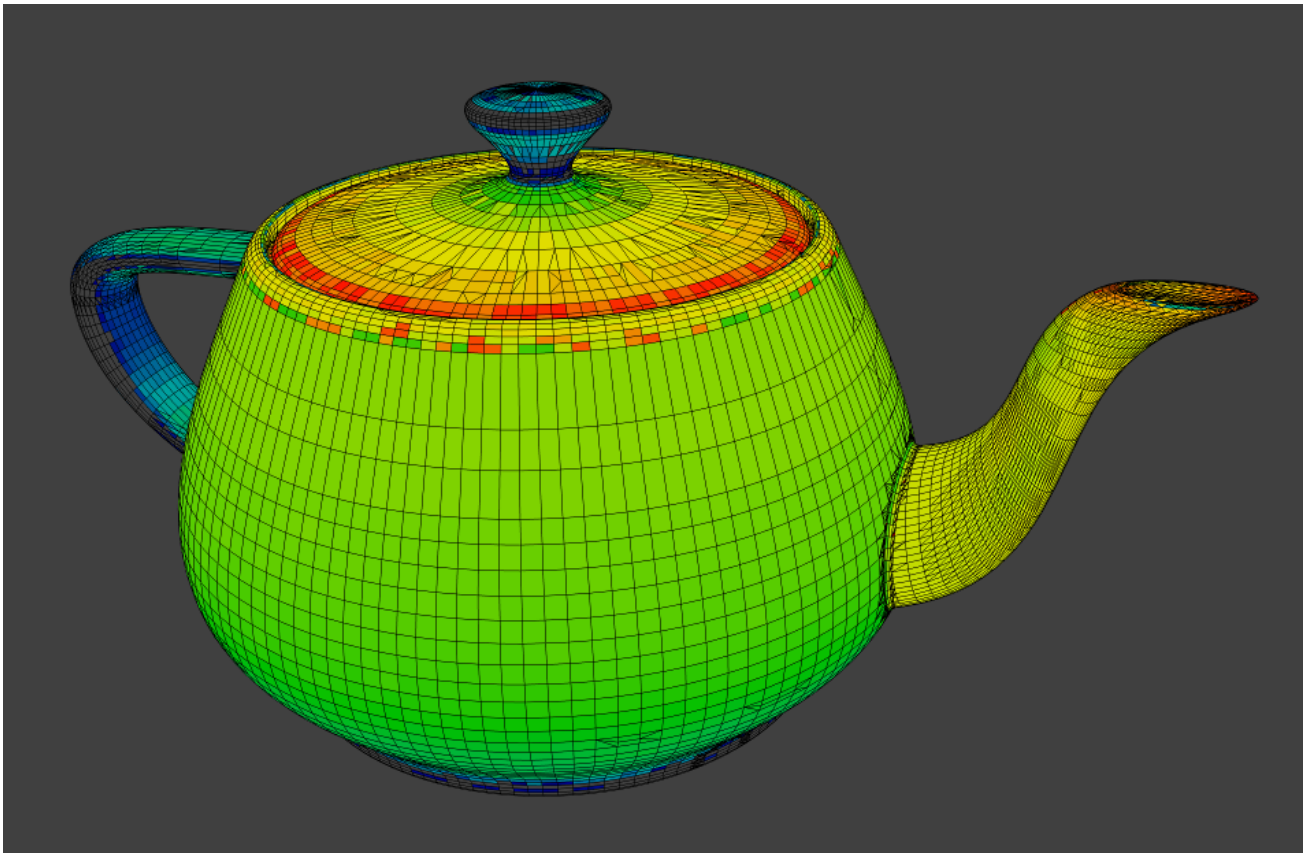


Fig. 2.638: Thickness.

Distortion

Distorted geometry can cause problems since the triangulation of a distorted ngon is undefined.

Distortion is measured by faces which are not flat, with parts of the face pointing in different directions.

Sharp Edges

Similar to wall-thickness, sharp edges can form shapes that are too thin to be able to print.

2.4.3 Curves

Introduction

Curves and *Surfaces* are particular types of Blender Objects. They are expressed by mathematical functions rather than a series of points.

Blender offers both *Bézier Curves* and *Non-Uniform Rational B-Splines (NURBS)*. Both Bézier curves and NURBS curves and surfaces are defined in terms of a set of "control points" (or "control vertices") which define a "control polygon".

Both Bézier and NURBs curves are named after their mathematical definitions, and choosing between them is often more a matter of how they are computed behind the scenes than how they appear from a modeler's perspective. Bézier curves are generally more intuitive because they start and end at the control points that you set, but NURBs curves are more efficient for the computer to calculate when there are many twists and turns in a curve.

The main advantage to using curves instead of polygonal meshes is that curves are defined by less data and so can produce results using less memory and storage space at modeling time. However, this procedural approach to surfaces can increase demands at render time.

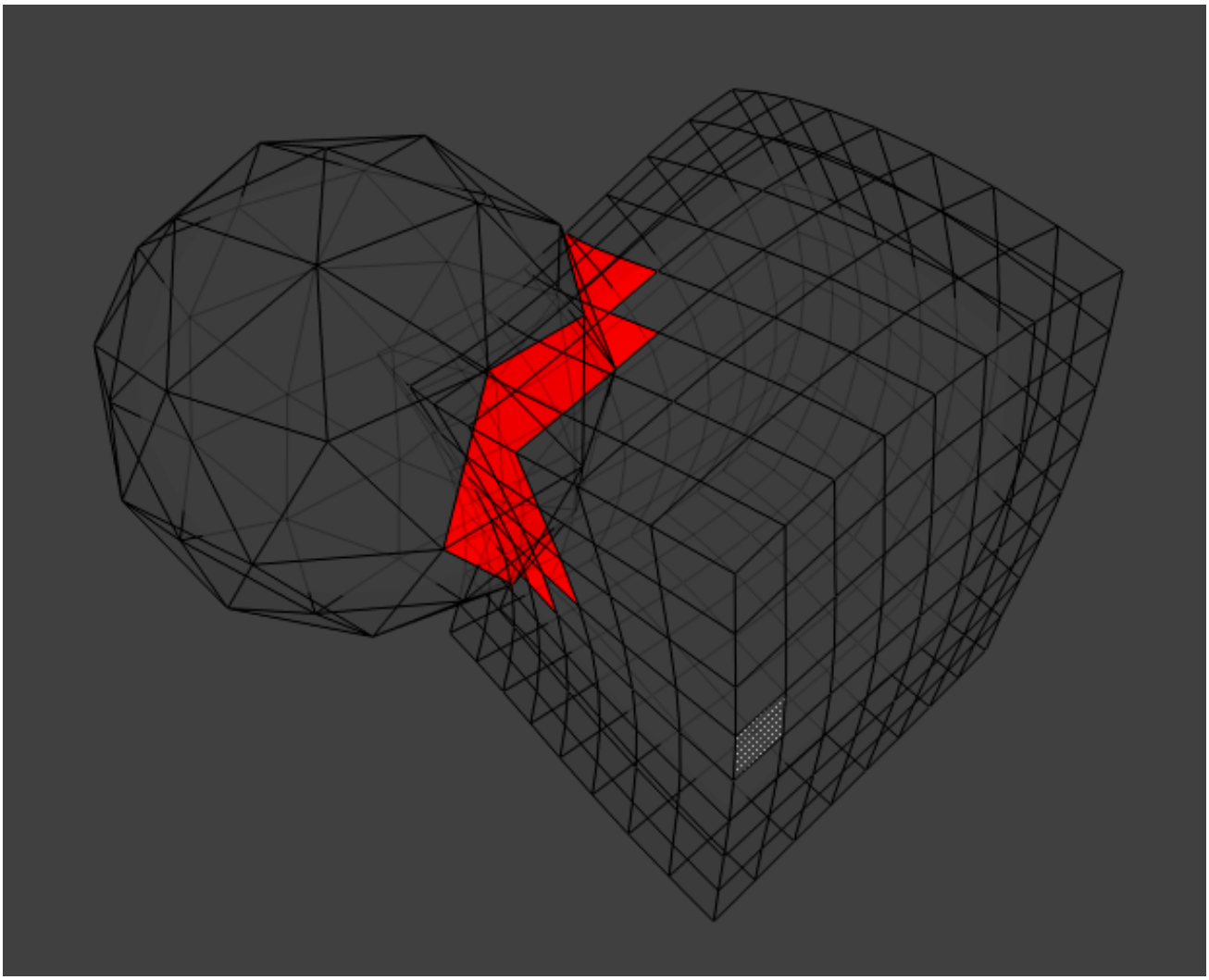
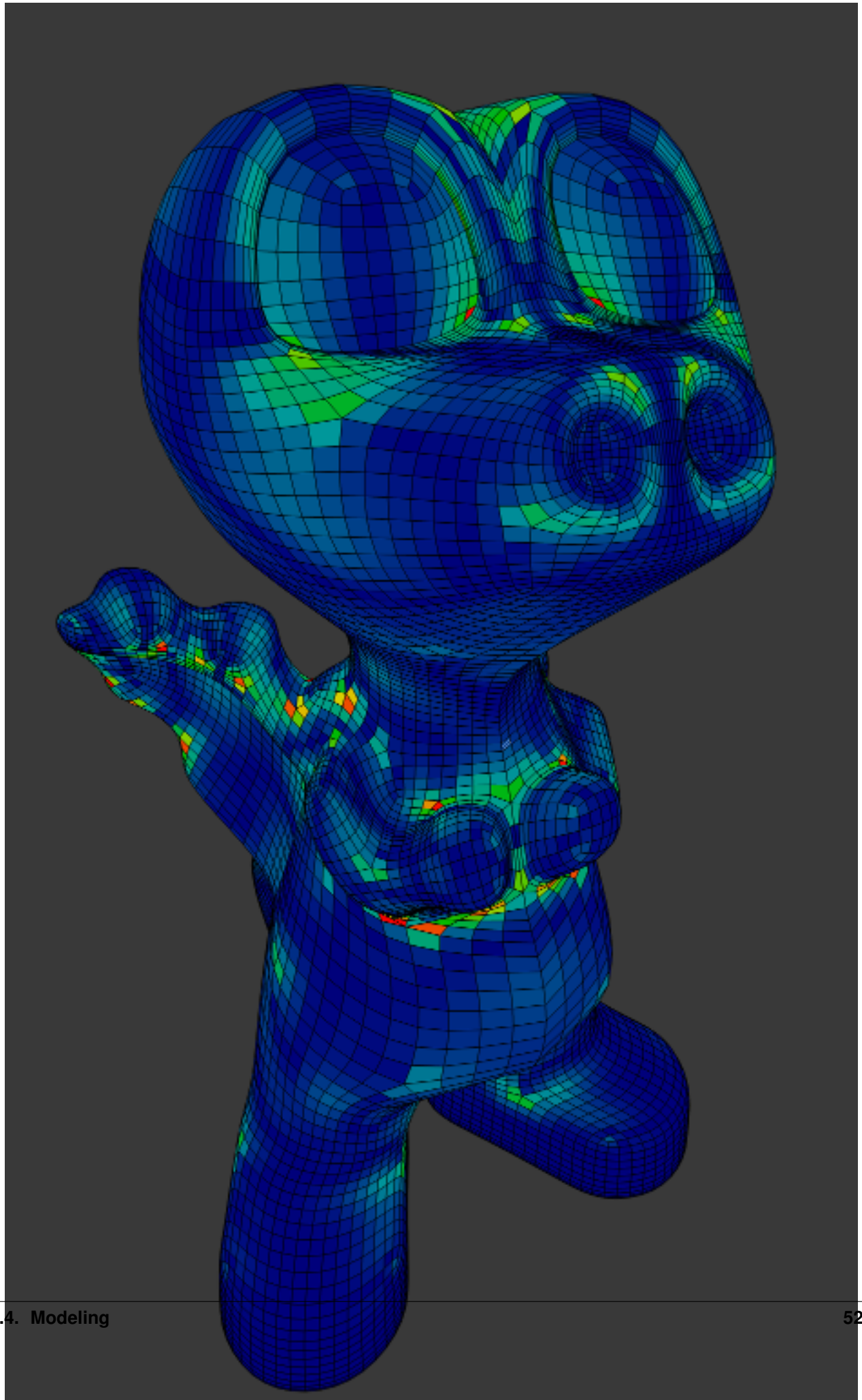


Fig. 2.639: Intersecting faces.



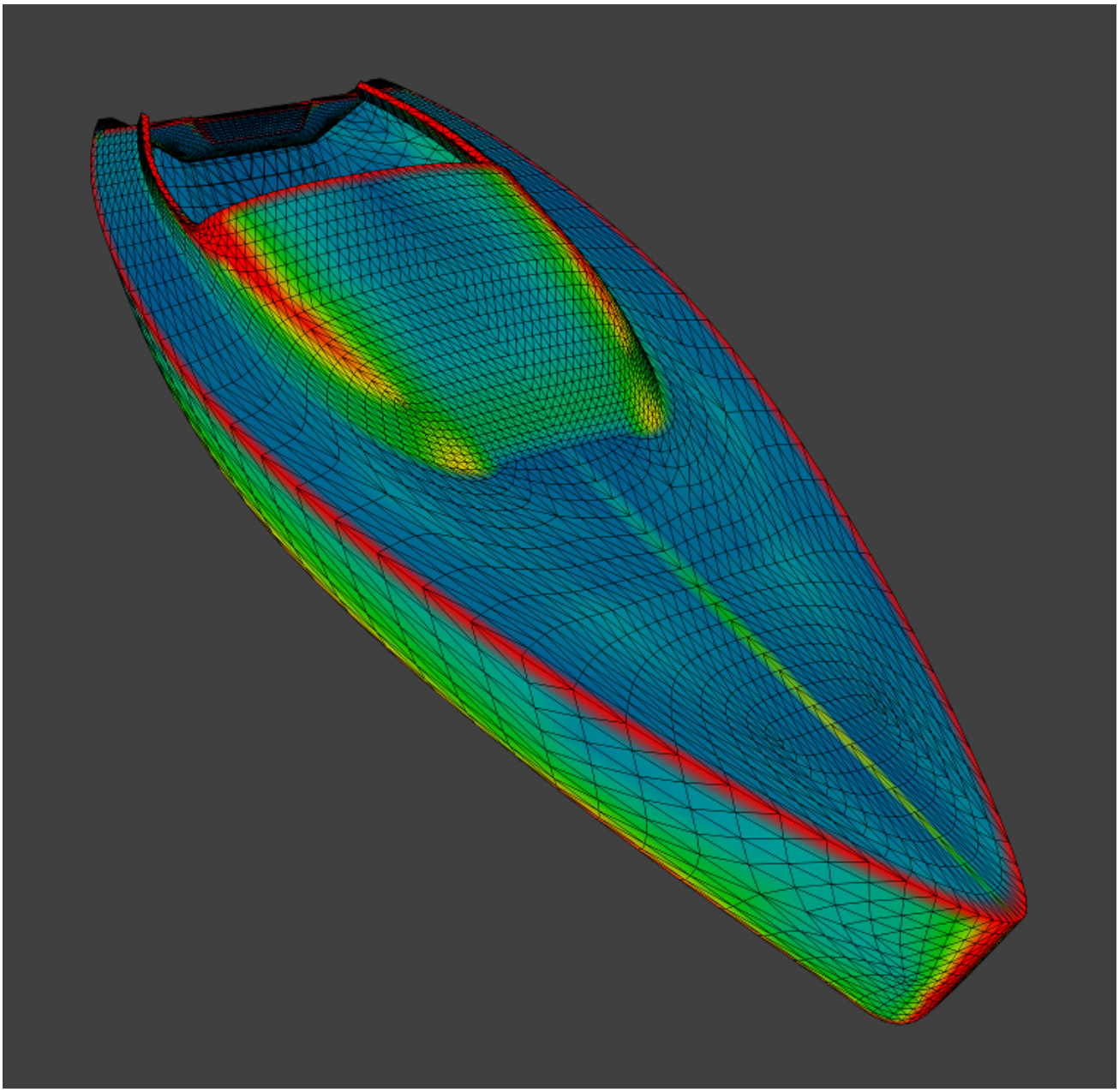


Fig. 2.641: Sharp edges.

Fig. 2.642: Blender logo made from Bézier curves.

Certain modeling techniques, such as *extruding a profile along a path*, are possible only using curves. On the other hand, when using curves, vertex-level control is more difficult and if fine control is necessary, *mesh editing* may be a better modeling option.

Bézier curves are the most commonly used curves for designing letters or logos. They are also widely used in animation, both as *paths* for objects to move along and as *F-Curves* to change the properties of objects as a function of time.

Primitives

In Object Mode, the *Add Curve* menu, Blender provides five different curve primitives:

Fig. 2.643: Add Curve menu.

Bézier Curve

Adds an open 2D Bézier curve with two control points.

Bézier Circle

Adds a closed, circle-shaped 2D Bézier curve (made of four control points).

NURBS Curve

Adds an open 2D NURBS curve, with four control points, with *Uniform* knots.

NURBS Circle

Adds a closed, circle-shaped 2D NURBS curve (made of eight control points).

Path

Adds a NURBS open 3D curve made of five aligned control points, with *Endpoint* knots and the *Curve Path* setting enabled.

Bézier Curves

The main elements used in editing Bézier Curves are the Control Points and Handles. A Segment (the actual Curve) is found between two Control Points. In the image below, the Control Points can be found in the middle of the pink line while the Handles comprise the extensions from the Control Point. By default the arrows on the Segment represents the direction and *relative* speed and direction of movement Objects will have when moving along the curve. This can be altered by defining a custom F-Curve.

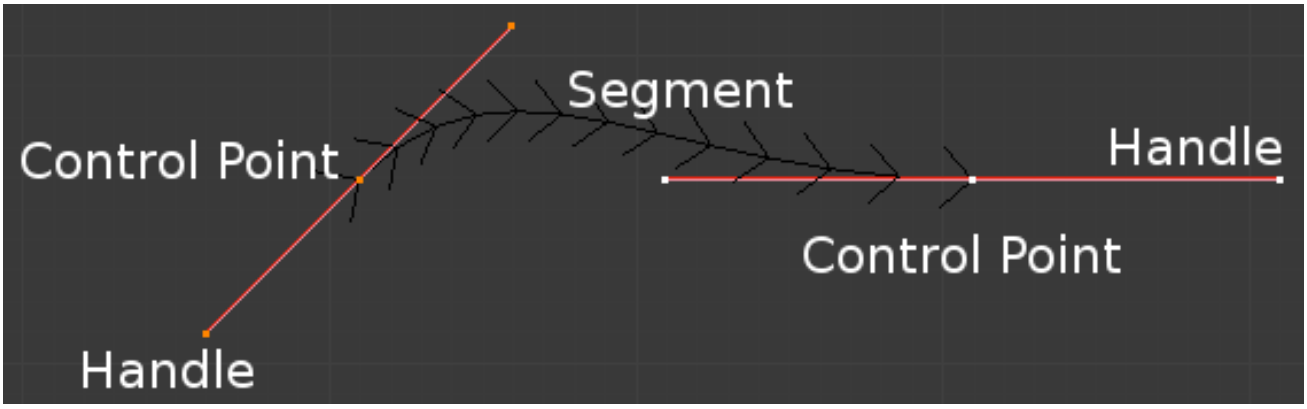


Fig. 2.644: Bézier Curve in Edit Mode.

Editing Bézier Curves

A Bézier curve can be edited by moving the locations of the Control Points and Handles:

1. Add a Curve by `Shift-A` to bring up the *Add* menu, followed by *Curve* → *Bézier*.
2. Press `Tab` to enter *Edit Mode*.
3. Select one of the Control Points and move it around. Use `LMB` to confirm the new location of the Control Point, or use `RMB` to cancel.
4. Now select one of the Handles and move it around. Notice how this changes the curvature of the curve.

To add more Control Points:

1. Select at least two adjacent Control Points.
2. Press `W` and select *Subdivide*.
3. Optionally, you can press `F6` immediately after the subdivision to modify the number of subdivisions.

Note that while in *Edit Mode* you cannot directly select a Segment. To do so, select all of the Control Points that make up the Segment you want to move.

There are four Bézier curve handle types. They can be accessed by pressing `V` and selecting from the list that appears, or by pressing the appropriate hotkey combination. Handles can be rotated, moved, scaled and shrunk/fattened like any vertex in a mesh.

Bézier Curve Handle Types

Automatic `V-A` This handle has a completely automatic length and direction which is set by Blender to ensure the smoothest result. These handles convert to *Aligned* handles when moved. (Yellow handles.)

Vector `V-V` Both parts of a handle always point to the previous handle or the next handle which allows you to create curves or sections thereof made of straight lines or with sharp corners. Vector handles convert to *Free* handles when moved. (Green handles.)

Aligned `V-L` These handles always lie in a straight line, and give a continuous curve without sharp angles. (Purple handles.)

Free `V-F` The handles are independent of each other. (Black handles.)

Additionally, the `V-T` shortcut can be used to toggle between Free and Aligned handle types.

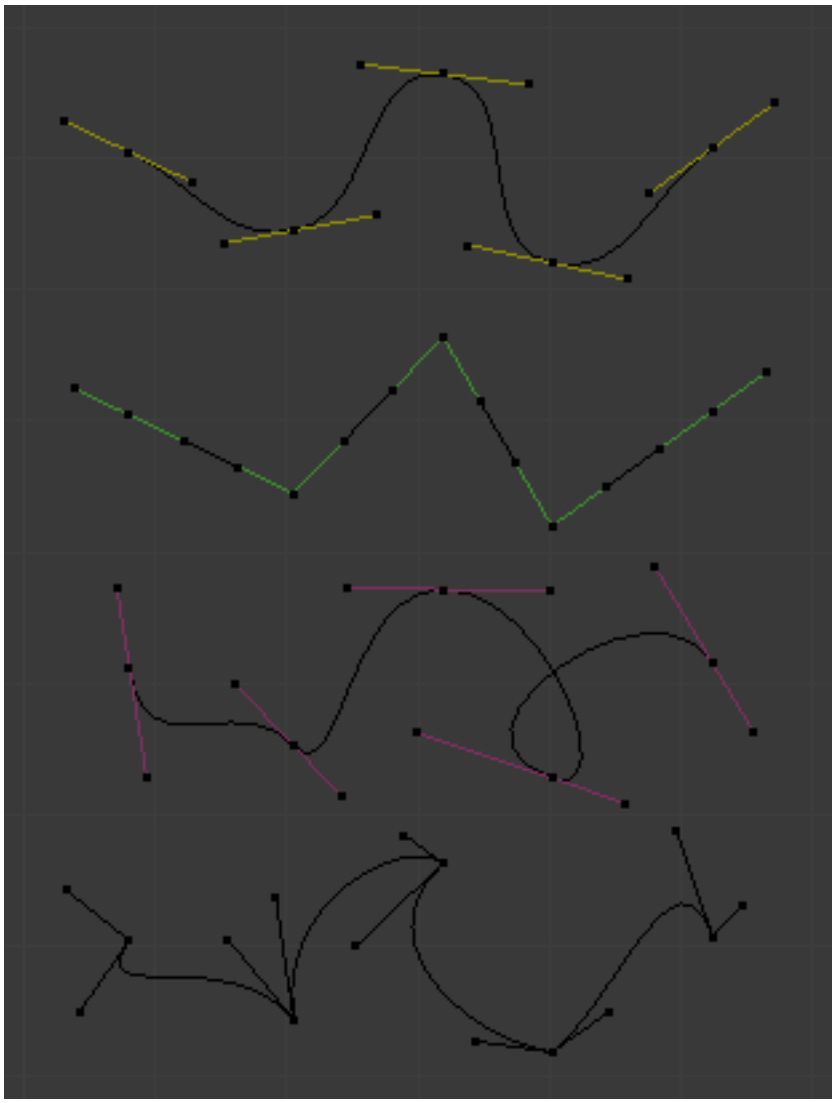


Fig. 2.645: Bézier Curve Handle Types.

Non-Uniform Rational B-Splines (NURBS)

One of the major differences between Bézier Objects and NURBS Objects is that Bézier Curves are approximations. For example, a Bézier circle is an *approximation* of a circle, whereas a NURBS circle is an *exact* circle. NURBS theory can be a *very* complicated topic. For an introduction, please consult the Wikipedia page. In practice, many of the Bézier curve operations discussed above apply to NURBS curves in the same manner. The following text will concentrate only on those aspects that are unique to NURBS curves.

Editing NURBS Curves

A NURBS Curve is edited by moving the location of the Control Points:

- Place a Curve by `Shift-A` to bring up the Add menu, followed by *Curve* → *NURBS curve*.
- Press `Tab` to enter *Edit Mode*.
- Select one of the Control Points and move it around. Use `LMB` to confirm the new location of the Control Point, or use `RMB` to cancel.
- If you want to add additional Control Points, select both of them, press `W` and select *Subdivide*. Press `F6` immediately after to determine how many subdivisions to make.

Properties

Curve Properties can be set from the *Object Data* option in the *Properties Header* (shown below in blue).

Shape

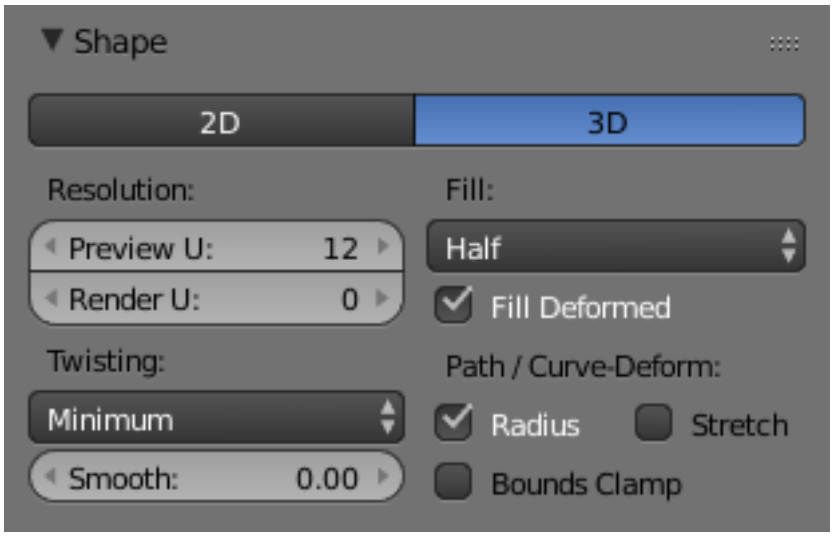


Fig. 2.646: Curves Shape panel.

2D and 3D Curves By default, new curves are set to be 3D, which means that Control Points can be placed anywhere in 3D space. Curves can also be set to 2D which constrain the Control Points to the Curve's local XY axis.

Resolution The *resolution* property defines the number of points that are computed between every pair of Control Points. Curves can be made more or less smooth by increasing and decreasing the resolution respectively. The *Preview U* setting determines the resolution in the 3D View while the *Render U* setting determines the Curve's render resolution. If *Render U* is set to zero (0), then the *Preview U* setting is used for both the 3D View and render resolution.

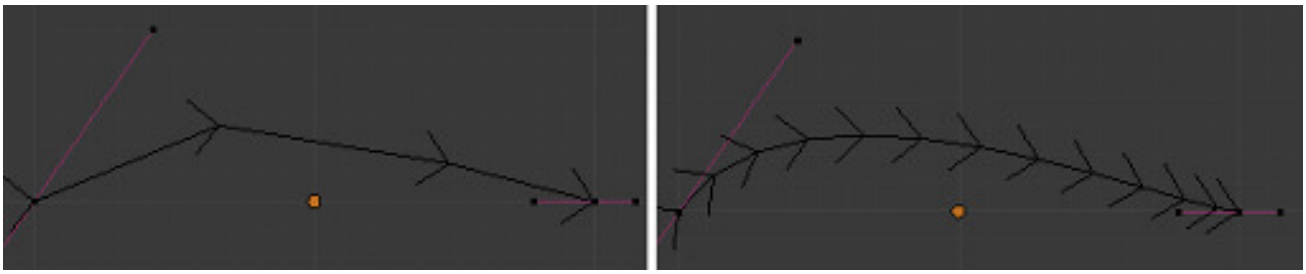


Fig. 2.647: Curves with a resolution of 3 (left) and 12 (right).

Twisting A 3D Curve has Control Points that are not located on the Curve's local XY plane. This gives the Curve a twist which can affect the Curve normals. You can alter how the twist of the Curve is calculated by choosing from *Minimum, Tangent* and *Z-Up* options from the select menu.

Fill Fill determines the way a Curve is displayed when it is Beveled (see below for details on Beveling). When set to *Half* (the default) the Curve is displayed as half a cylinder.

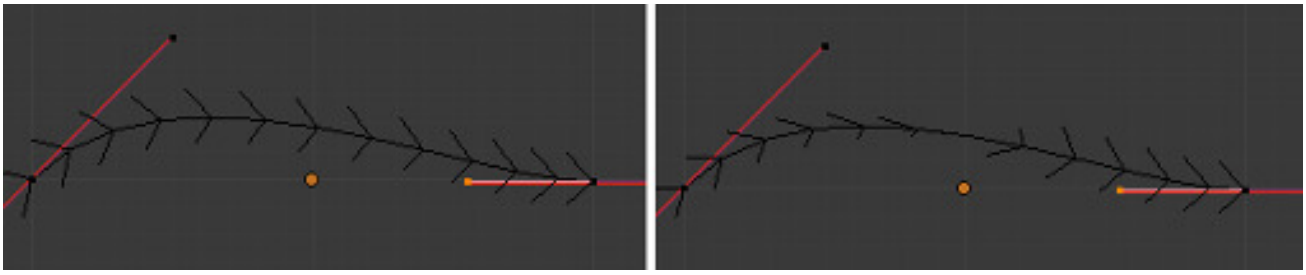


Fig. 2.648: Curves with a twist of minimum (left) and tangent (right).

Fill Deformed Fills the curve after applying all modification that might deform the curve (i.e. shape keys and modifiers).

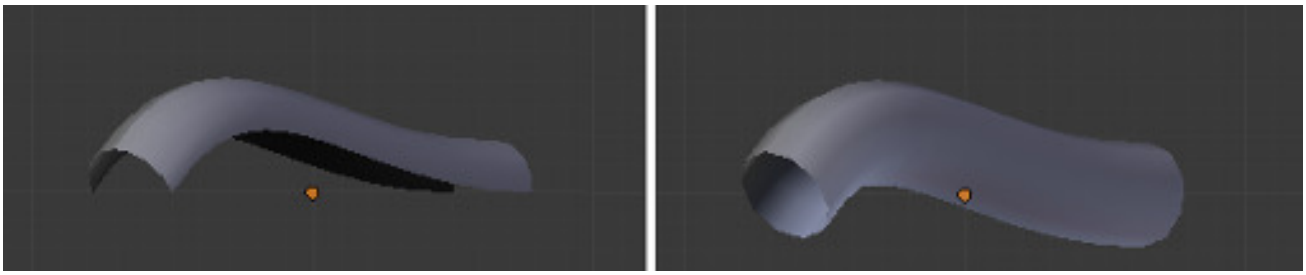


Fig. 2.649: Curves with a fill of half (left) and full (right).

Path/Curve-Deform These options are primarily utilized when using a Curve as a Path or when using the Curve Deform property. The *Radius, Stretch* and *Bounds Clamp* options control how Objects use the Curve and are dealt with in more detail in the appropriate links below.

See also:

- *Basic Curve Editing*
- *Animation Paths*

Geometry

Modification

Offset By default, text Objects are treated as curves. The Offset option will alter the space between letters.

Extrude Will extrude the curve along both the positive and negative local Z axes.

Bevel

Depth Changes the size of the bevel.

Resolution Alters the smoothness of the bevel.

Taper Object Tapering a Curve causes it to get thinner towards one end. You can also alter the proportions of the Taper throughout the tapered object by moving/scaling/rotating the Control Points of the Taper Object. The Taper Object can only be another Curve. Editing the Handles and Control Points of the Taper Object will cause the original Object to change shape.

Bevel Object Beveling a Bézier Curve with a Bézier Curve as the Bevel Object generally gives it the appearance of a plane, while using a Bézier Circle as the Bevel Object will give it the appearance of a cylinder. The Bevel Object can only be another Curve. Editing the Handles and Control Points of the Bevel Object will cause the original Object to change shape. Given the options available, it is best to experiment and see the results of this operation.

Fill Caps Seals the ends of a beveled Curve.

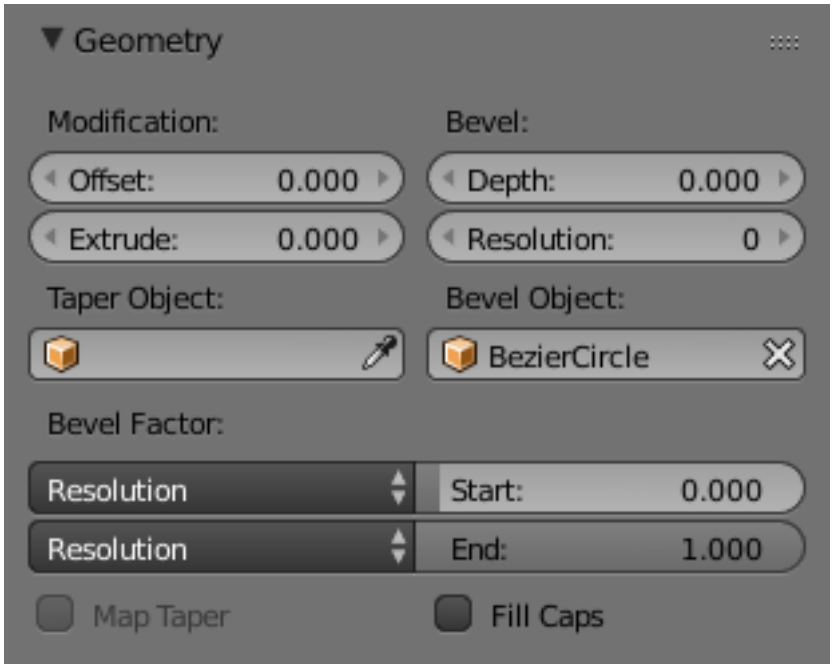


Fig. 2.650: Curves Geometry panel.

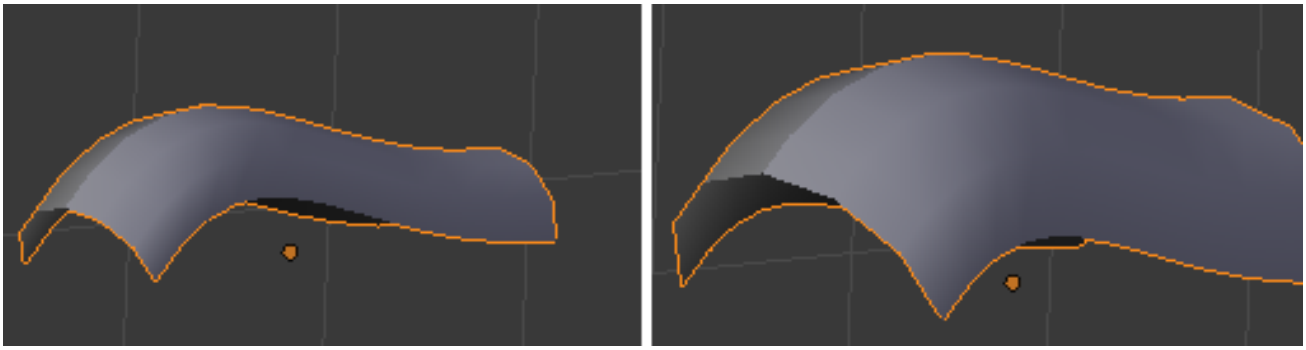


Fig. 2.651: A Curve with different Bevel depths applied.

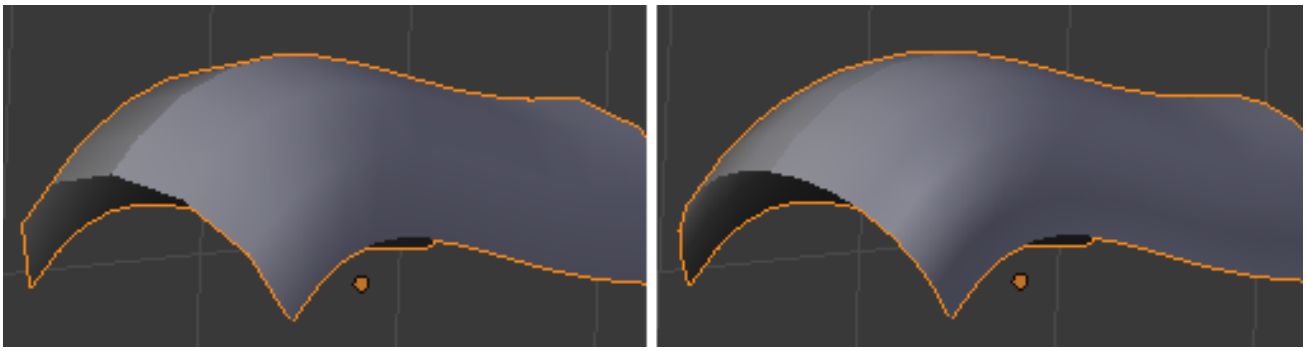


Fig. 2.652: A Curve with different resolutions applied.

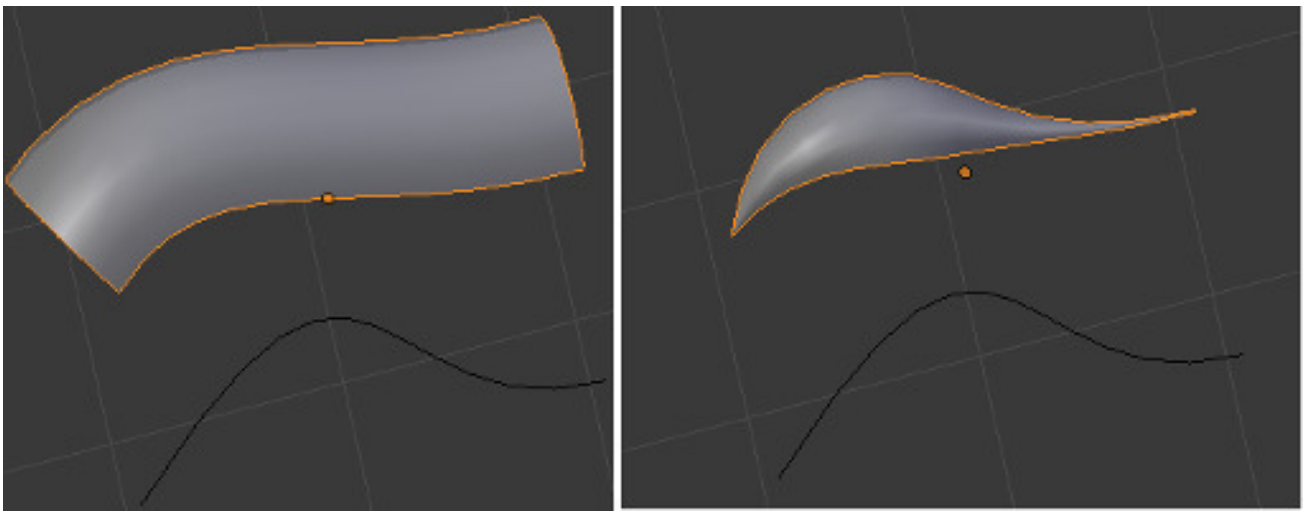


Fig. 2.653: A Curve before (left) and after (right) a Bézier Curve Taper Object was applied.

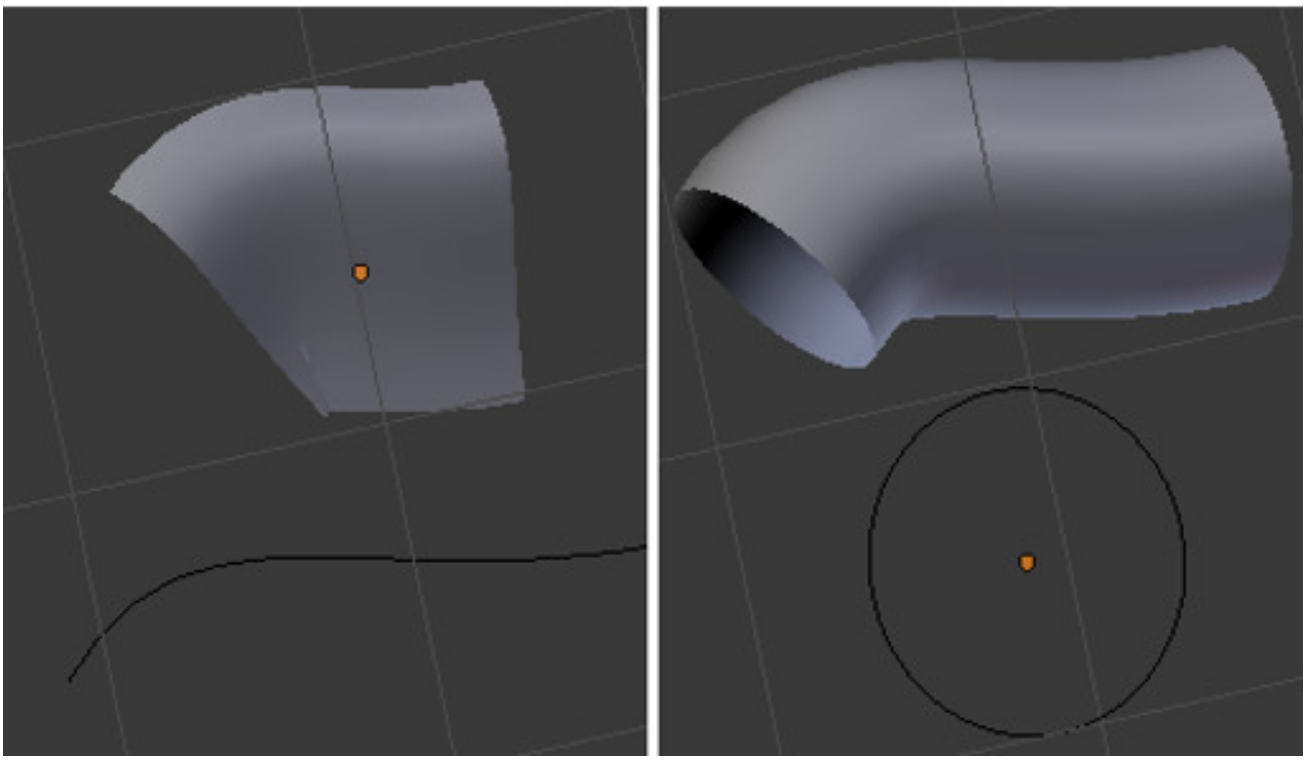


Fig. 2.654: A Curve with the Bevel Object as a Bézier Curve (left) and as a Bézier Circle (right).

Map Taper For Curves using a Taper Object and with modifications to the *Start/End Bevel Factor* the *Map Taper* option will apply the taper to the beveled part of the Curve (not the whole Curve).

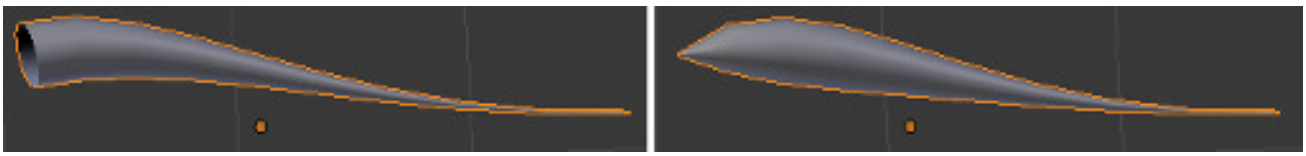


Fig. 2.655: A Curve without (left) and with (right) Map Taper applied.

Start Bevel Factor and End Bevel Factor These options determine where to start the Bevel operation on the Curve being beveled. Increasing the *Start Bevel Factor* to 0.5 will start beveling the Curve 50% of the distance from the start of the Curve (in effect shortening the Curve). Decreasing the *End Bevel Factor* by 0.25 will start beveling the Curve 25% of the distance from the end of the Curve (again, shortening the Curve).

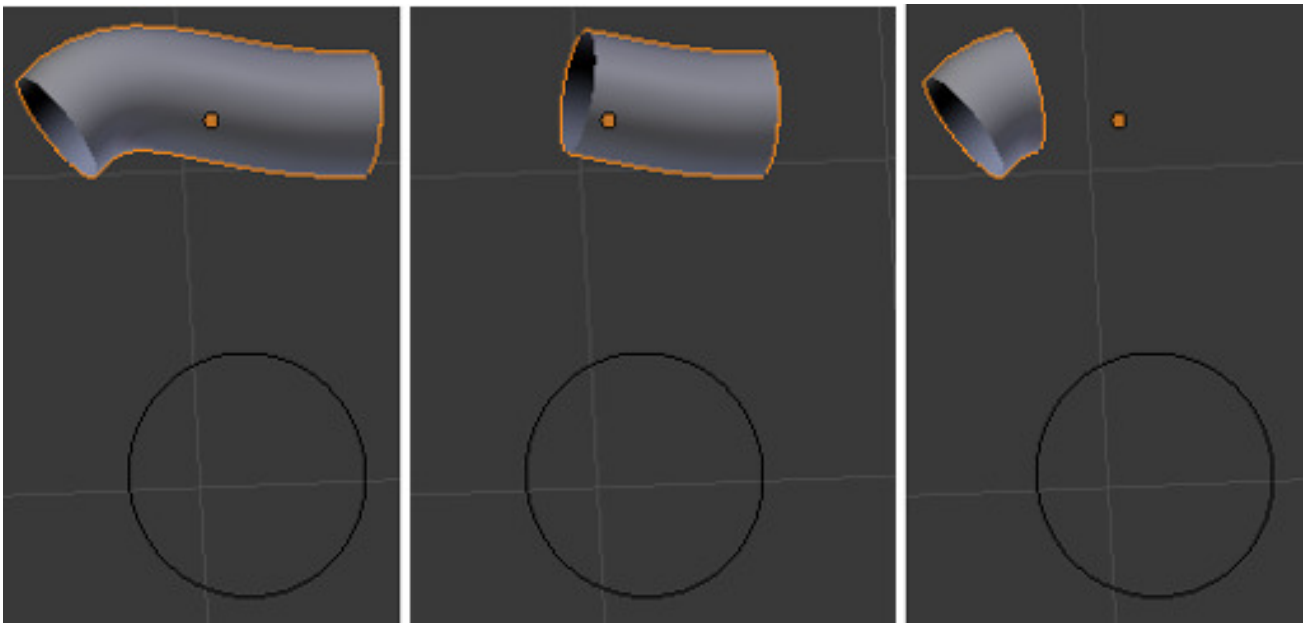


Fig. 2.656: A Curve with no Bevel factor applied (left), with a 50% Start Bevel Factor (middle) and with a 25% End Bevel Factor (right).

Path Animation

The *Path Animation* settings can be used to determine how Objects move along a certain path.

Frames The number of frames that are needed to traverse the path, defining the maximum value for the *Evaluation Time* setting.

Evaluation Time Parametric position along the length of the curve that object following it should be at (the position is evaluated by by dividing by the *Path Length* value).

Follow Make the curve path children rotate along the path.

Read more about utilizing Curves for paths during animation

Active Spline

The *Active Spline* panel becomes available during *Edit Mode*.

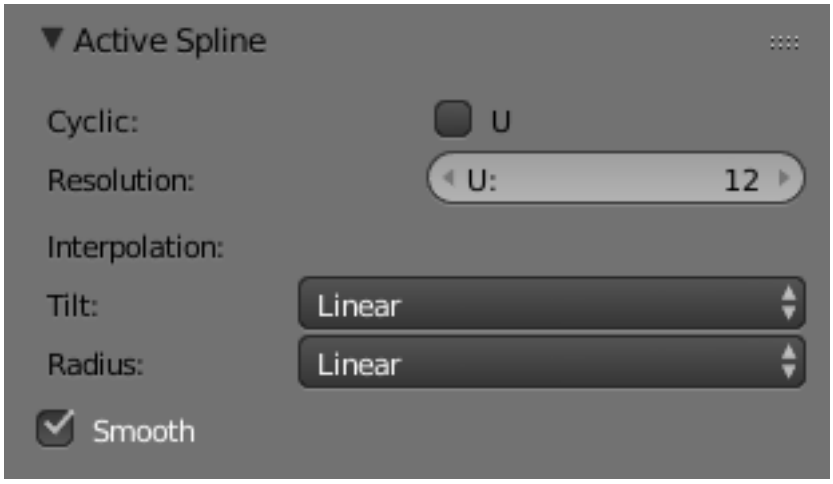


Fig. 2.657: Curves Active Spline panel.

Cyclic Closes the Curve.

Resolution Alters the smoothness of each segment by changing the number of subdivisions.

Interpolation

Tilt Alters how the tilt of a segment is calculated.

Radius Alters how the radius of a Beveled Curve is calculated. The effects are easier to see after Shrinking/Fattening a control point `Alt-S`.

Smooth Smooths the normals of the Curve.

NURBS Curves

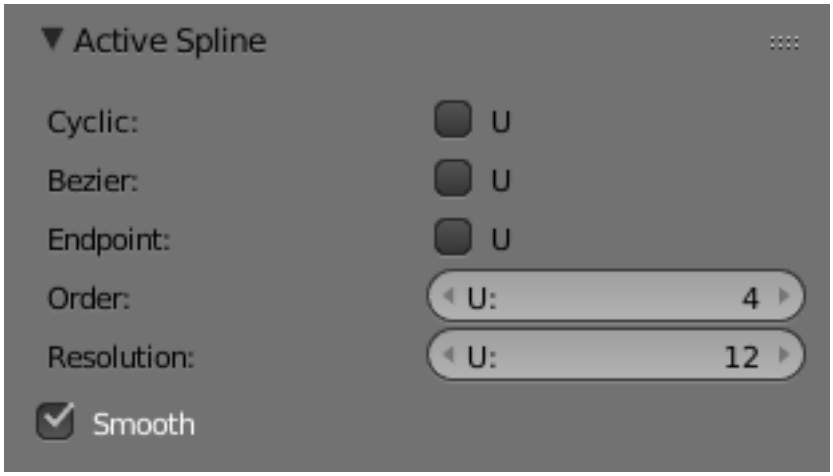


Fig. 2.658: NURBS Active Spline panel.

Knots One of the characteristics of a NURBS object is the *knot vector*. This is a sequence of numbers used to determine the influence of the control points on the curve. While you cannot edit the knot vectors directly, you can influence them through the *Endpoint* and *Bézier* options in the Active Spline panel. Note that the *Endpoint* and *Bézier* settings only apply to open NURBS curves.

Cyclic Makes the NURBS curve cyclic.

Bézier Makes the NURBS curve act like a Bézier curve.

Endpoint Makes the curve contact the end control points. Cyclic must be disabled for this option to work.

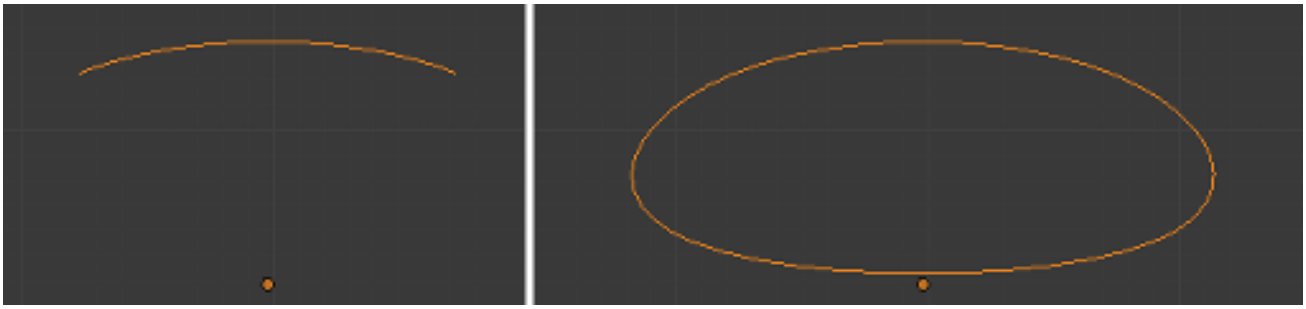


Fig. 2.659: A NURBS curve with Cyclic applied.

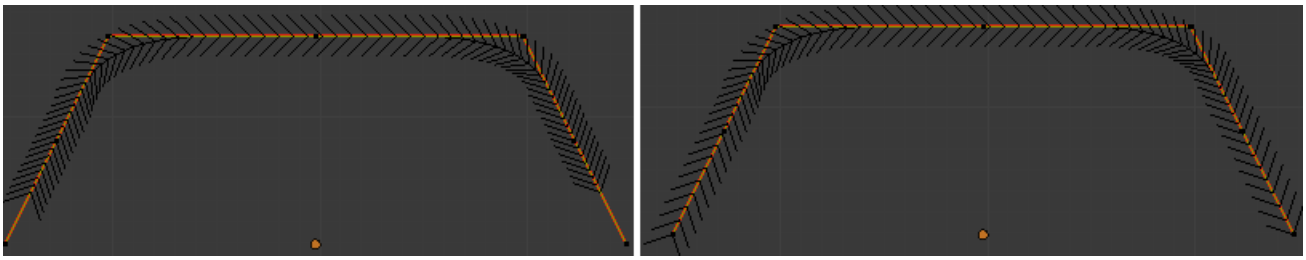


Fig. 2.660: A NURBS curve with Endpoint enabled.

Order The order of the NURBS curve determines the area of influence of the control points over the curve. Higher order values means that a single control point has a greater influence over a greater relative proportion of the curve. The valid range of *Order* values is 2-6 depending on the number of control points present in the curve.

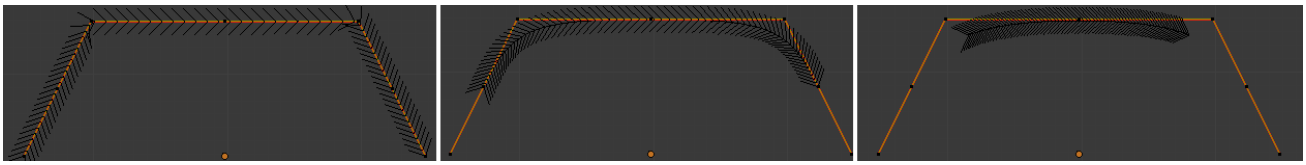


Fig. 2.661: NURBS curves with orders of 2 (left), 4 (middle) and 6 (right).

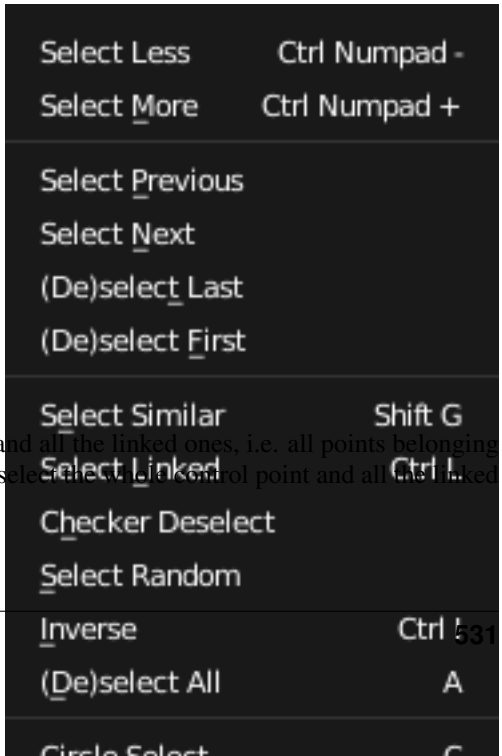
Selecting

Curve selection in *Edit Mode* is much less complex than with meshes! Mainly this is because you have only one selectable element type, the control points (no select mode needed here...). These points are a bit more complex than simple vertices, however, especially for Béziers, as there is the central vertex, and its two handles...

The basic tools are the same as with *meshes*, so you can select a simple control point with a `RMB`, add to current selection with `Shift-RMB`, `B` border-select, and so on.

One word about the Bézier control points: when you select the main central vertex, the two handles are automatically selected too, so you can grab it as a whole, without creating an angle in the curve. However, when you select a handle, only this vertex is selected, allowing you to modify this control vector...

`L` (or `Ctrl-L`) will add to the selection the cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same curve. Note that for Bézier, using `L` with a handle selected will select the whole control point and all the linked ones.



Select Menu

With curves, all "advanced" selection options are regrouped in the *Select* menu of the 3D Views header. Let us detail them:

- Random...
- Inverse
- Select/Deselect All

Border Select All these options have the same meaning and behavior as in *Object Mode* (and the specifics of *Border Select* in *Edit Mode* have already been discussed *here*).

Checker Deselect

Reference

Mode: Edit Mode
Menu: *Select* → *Checker Deselect*
Hotkey: None

This only works if you already have at least one control point selected. Using the current selection, it will add to it every nth control point, before and after the initial selection. The "selection step" is specified in the *N* pop-up number button shown during the tool start.

Nth Selection Number of points to select.

Skip Number of points to skip.

Offset Offsets at what point to start at.

Select/Deselect First/Last

Reference

Mode: Edit Mode
Menu: *Select* → *Select/Deselect First*, *Select* → *Select/Deselect Last*
Hotkey: None

These commands will toggle the selection of the first or last control point(s) of the curve(s) in the object. This is useful to quickly find the start of a curve (e.g. when using it as path...).

Select Next/Previous

Reference

Mode: Edit Mode
Menu: *Select* → *Select Next*, *Select* → *Select Previous*
Hotkey: None

These commands will select the next or previous control point(s), based on the current selection (i.e. the control points following or preceding the selected ones along the curve).

Select More/Less

Reference

Mode: Edit Mode
Menu: *Select* → *More/Less*
Hotkey: `Ctrl-NumpadPlus/Ctrl-NumpadMinus`

Their purpose, based on the currently selected control points, is to reduce or enlarge this selection.

More for each selected control point, select *all* its linked points (i.e. one or two...).

Less for each selected control point, if *all* points linked to this point are selected, keep this one selected. Otherwise, de-select it.

This implies two points:

- First, when *all* control points of a curve are selected, nothing will happen (as for *Less*, all linked points are always selected, and of course, *More* cannot add any). Conversely, the same goes when no control points are selected.
- Second, these tools will never "go outside" of a curve (they will never "jump" to another curve in the same object).

Editing

Introduction

This page covers the basics of curve editing. Curve basics, selecting and advanced editing are covered in the following pages:

- *Curve basics*
- *Curve Selecting*

Translation, Rotation, Scale

Reference

Mode: Edit Mode
Menu: *Curve* → *Transform* → *Grab/Move, Rotate, Scale, ...*
Hotkey: `G`, `R`, `S`

Like other elements in Blender, Curve control points can be grabbed/moved G, rotated R or scaled S as described in the *Basic Transformations* section. When in *Edit Mode*, *proportional editing* is also available for transformation actions.

Snapping

Reference

Mode: Edit Mode
Panel: Curve Tools

Mesh snapping also works with curve components. Both control points and their handles will be affected by snapping, except for within itself (other components of the active curve). Snapping works with 2D curves but points will be constrained to the local XY axes.

Deforming Tools

Reference

Mode: Edit Mode
Menu: *Curve* → *Transform*

The *To Sphere*, *Shear*, *Warp* and *Push/Pull* transform tools are described in the *Transformations* sections. The two other tools, *Tilt* and *Shrink/Fatten Radius* are related to *Curve Extrusion*.

Smoothing

Reference

Mode: Edit Mode
Hotkey: W → *smooth*

Curve smoothing is available through the specials menu. For Bézier curves, this smoothing operation reduces the distance between the selected control point/s and their neighbors, while keeping the neighbors anchored. Does not effect control point tangents.

Mirror

Reference

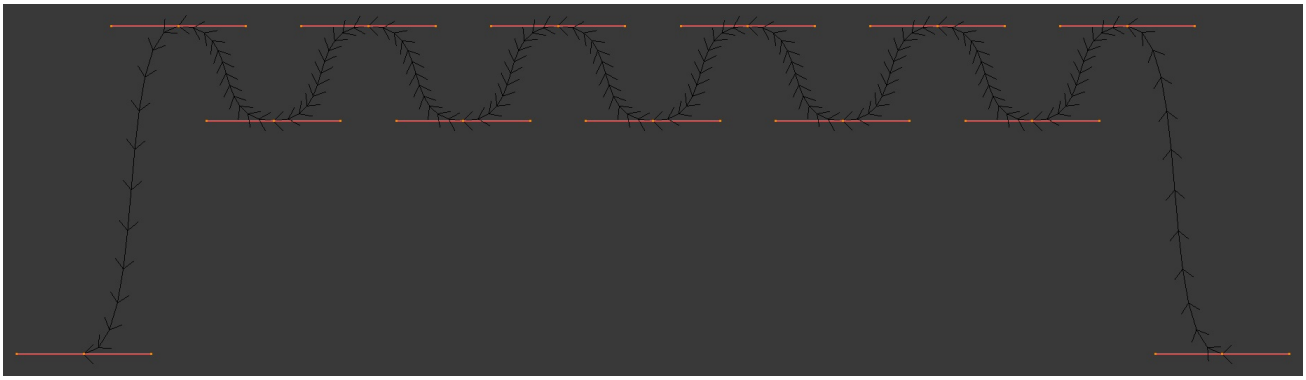


Fig. 2.663: Original, unsmoothed Curve.

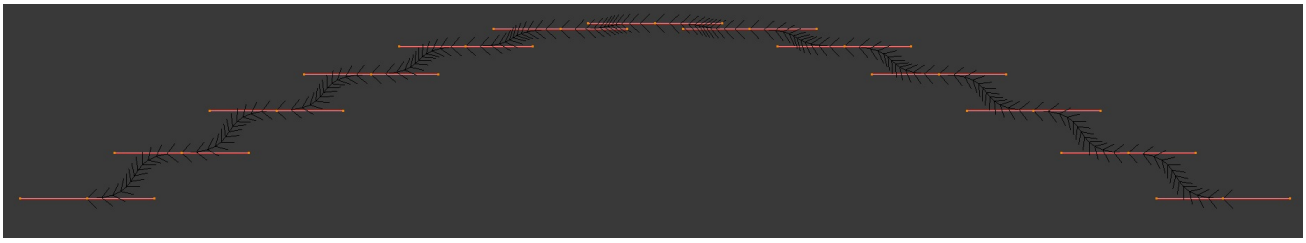


Fig. 2.664: Entire curve smoothed over 200 times by holding `Shift-R` to repeat last step.

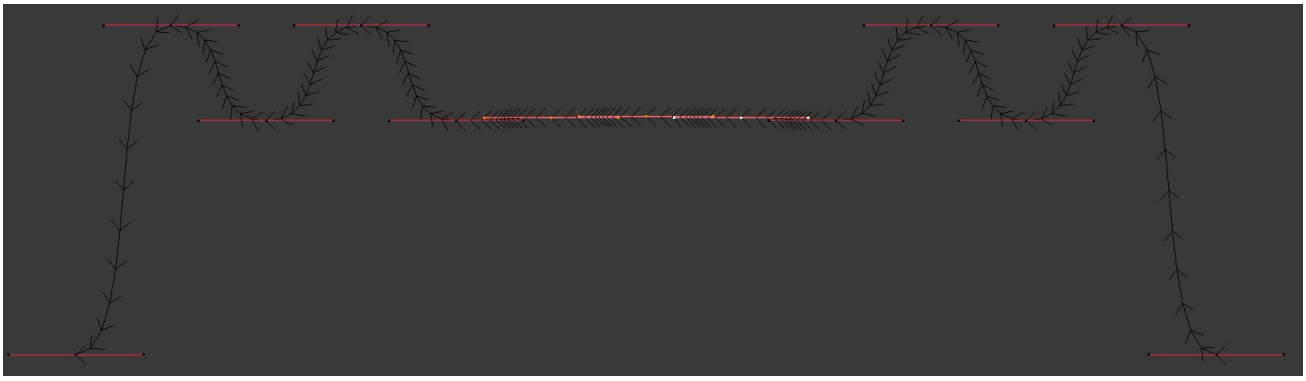


Fig. 2.665: Only three control points in the center smoothed over 200 times.

Mode: Edit Mode
Menu: *Curve* → *Mirror*
Hotkey: `Ctrl-M`

The *Mirror* tool is also available, behaving exactly as with *mesh vertices*,

Set Bézier Handle Type

Reference

Mode: Edit Mode
Panel: *Curve Tools* → *Handles*
Menu: *Curve* → *Control Points* → *Set Handle Type*
Hotkey: `V`

Handle types are a property of *Bézier curves.* and can be used to alter features of the curve. For example, switching to *Vector handles* can be used to create curves with sharp corners. Read the *Bézier curves* page for more details.

Extending Curves

Reference

Mode: Edit Mode
Menu: *Curve* → *Extrude*
Hotkey: `Ctrl-LMB`, `E`

Once a curve is created you can add new segments (in fact, new control points defining new segments), either by extruding, or placing new handles with `Ctrl-LMB`. Each new segment is added to one end of the curve. The Bézier curve can only be extend at the endpoints. `Ctrl-LMB` on inner control points will make unconnected duplicates.

Subdivision

Reference

Mode: Edit Mode
Panel: Curve Tools
Menu: *Surface tools* → *Modeling* → *Subdivide*
Hotkey: `W`

Curve subdivision simply subdivides all selected segments by adding one or more control points between the selected segments. To control the number of cuts, press `W` to make a single subdivision. Then press `F6` to bring up the *Number of Cuts* menu.

Duplication

Reference

Mode: Edit Mode
Menu: *Curve* → *Duplicate*
Hotkey: `Shift-D`

This command duplicates the selected control points, along with the curve segments implicitly selected (if any). The copy is selected and placed in *Grab* mode, so you can move it to another place.

Joining Curve Segments

Reference

Mode: Edit Mode
Menu: *Curve* → *Make Segment*
Hotkey: `F`

Two open curves can be combined into one by creating a segment between the two curves. To join two separated curves, select one end control point from each curve then press `F`. The two curves are joined by a segment to become a single curve.

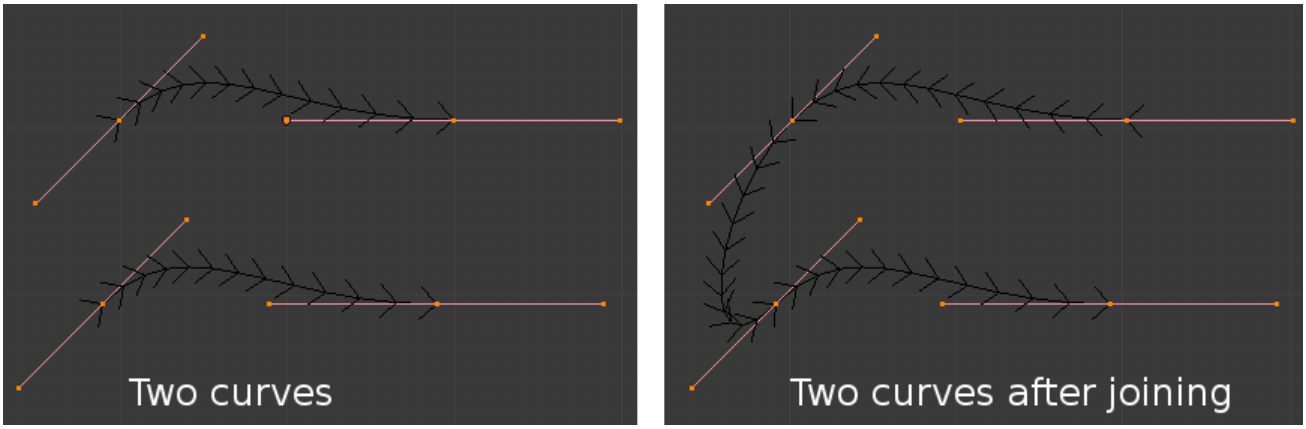


Fig. 2.666: Curves before and after joining.

Additionally, you can close a curve by joining the endpoints but note that you can only join curves of the same type (i.e. Bézier with Bézier, NURBS with NURBS)

Separating Curves

Reference

Mode: Edit Mode
Menu: *Curve* → *Separate*
Hotkey: `P`

Curve objects that are made of multiple distinct curves can be separated into their own objects by selecting the desired segments and pressing `P`. Note, if there is only one curve in a Curve object, pressing `P` will create a new Curve object with no control points.

Deleting Elements

Reference

Mode: Edit Mode
Menu: *Curve* → *Delete...*
Hotkey: `X`, `Delete`

Options for the *Erase* pop-up menu:

Selected This will delete the selected control points, *without* breaking the curve (i.e. the adjacent points will be directly linked, joined, once the intermediary ones are deleted). Remember that NURBS order cannot be higher than its number of control points, so it might decrease when you delete some control point. Of course, when only one point remains, there is no more visible curve, and when all points are deleted, the curve itself is deleted.

Segment This option is somewhat the opposite to the preceding one, as it will cut the curve, without removing any control points, by erasing one selected segment. This option always removes *only one* segment (the last "selected" one), even when several are in the selection. So to delete all segments in your selection, you will have to repetitively use the same erase option...

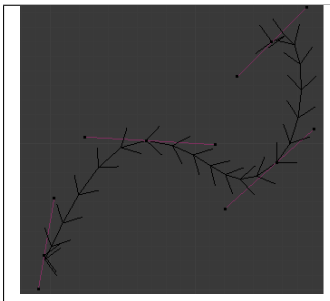


Fig. 2.667: Deleting Curve Selected.

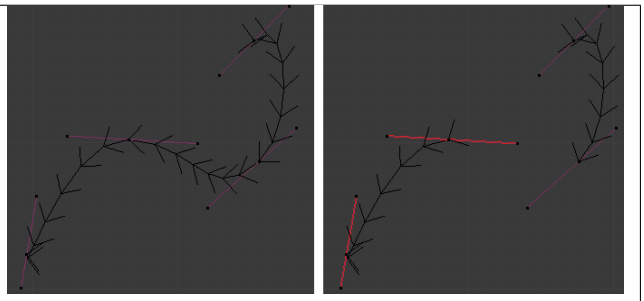


Fig. 2.668: Deleting Curve segments.

Opening and Closing a Curve

Reference

Mode: Edit Mode
Menu: *Curve* → *Toggle Cyclic*
Hotkey: `Alt-C`

This toggles between an open curve and closed curve (Cyclic). Only curves with at least one selected control point will be closed/open. The shape of the closing segment is based on the start and end handles for Bézier curves, and as usual on adjacent control points for NURBS. The only time a handle is adjusted after closing is if the handle is an *Auto* one. Fig. *Open and Closed curves.* is the same Bézier curve open and closed.

This action only works on the original starting control-point or the last control-point added. Deleting a segment(s) does not change how the action applies; it still operates only on the starting and last control-points. This means that `Alt-C` may actually join two curves instead of closing a single curve! Remember that when a 2D curve is closed, it creates a renderable flat face.

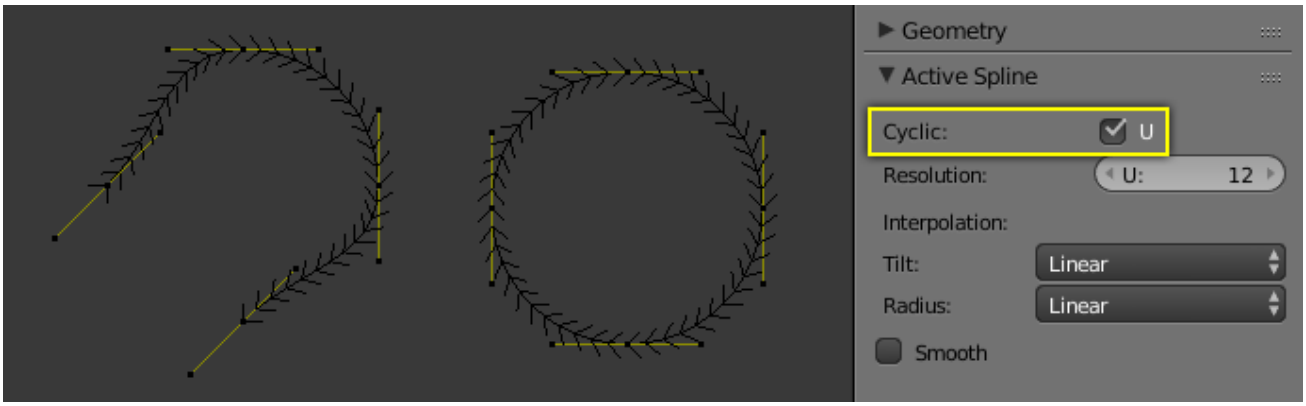


Fig. 2.669: Open and Closed curves.

Switch Direction

Reference

Mode: Edit Mode
Menu: *Curve* → *Segments* → *Switch Direction*, *Specials* → *Switch Direction*
Hotkey: `W-Numpad2`

This command will "reverse" the direction of any curve with at least one selected element (i.e. the start point will become the end one, and *vice versa*). This is mainly useful when using a curve as path, or using the bevel and taper options.

Converting Tools

Converting Curve Type

Reference

Mode: Edit Mode

Panel: Curve Tools –> Set Spline type

You can convert splines in a curve object between Bézier, NURBS, and Poly curves. Press `T` to bring up the Tool Shelf. Clicking on the *Set Spline Type* button will allow you to select the Spline type (Poly, Bézier or NURBS).

Note, this is not a "smart" conversion, i.e. Blender does not try to keep the same shape, nor the same number of control points. For example, when converting a NURBS to a Bézier, each group of three NURBS control points become a unique Bézier one (center point and two handles).

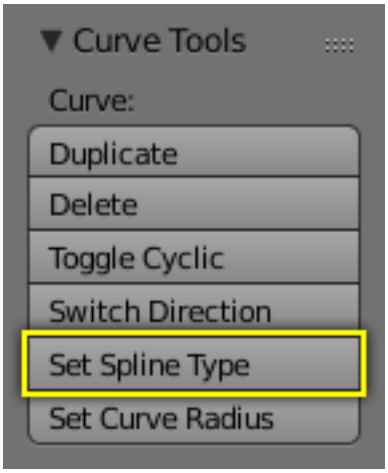


Fig. 2.670: Set Spline Type button.

Convert Curve to Mesh

Reference

Mode: Object Mode
Menu: *Object* → *Convert to* → *Mesh From Curve/Meta/Surface/Text*
Hotkey: `Alt-C`

There is also an "external" conversion, from curve to mesh, that only works in *Object Mode*. It transforms a *Curve* object into a *Mesh* object, using the curve resolution to create edges and vertices. Note that it also keeps the faces and volumes created by closed and extruded curves.

Convert Mesh to Curve

Reference

Mode: Object Mode
Menu: *Object* → *Convert to* → *Curve From Mesh/Text*
Hotkey: `Alt-C`

Mesh objects that consist of a series of connected vertices can be converted into curve objects. The resulting curve will be a Poly curve type, but can be converted to have smooth segments as described above.

Curve Parenting

Reference

Mode: Edit Mode
Hotkey: `Ctrl-P`

You can make other selected objects *children* of one or three control points `Ctrl-P`, as with mesh objects.

To select a mesh (that is in view) while editing a curve, `Ctrl-P` click on it. Select either one or three control points, then `Ctrl-RMB` the object and use `Ctrl-P` to make a vertex parent. Selecting three control points will make the child follow the median point between the three vertices. An alternative would be to use a *Child of Constraint*

Hooks

Reference

Mode: Edit Mode
Menu: *Curve* → *control points* → *hooks*
Hotkey: `Ctrl-H`

Hooks can be added to control one or more points with other objects.

Set Goal Weight

Reference

Mode: Edit Mode
Menu: `W` → *Set Goal Weight*

This sets the "goal weight" of selected control points, which is used when a curve has *Soft Body* physics, forcing the curve to "stick" to their original positions, based on the weight.

Hiding Elements

When in *Edit Mode*, you can hide and reveal elements from the display. This can be useful in complex models with many elements on the Screen.

Hide Selected elements Use `H`, or the *Curve* → *Show/Hide* → *Hide Selected* menu option from the 3D View header.

Show Hidden elements Use `Alt-H`, or the *Curve* → *Show/Hide* → *Show Hidden* menu option from the 3D View header.

Hide Unselected elements Use `Shift-H`, or the *Curve* → *Show/Hide* → *Hide Unselected* menu option from the 3D View header.

Curve Deform

Curve Deform provides a simple but efficient method of defining a deformation on a mesh. By parenting a mesh object to a curve, you can deform the mesh up or down the curve by moving the mesh along, or orthogonal to, the dominant axis. This is a most useful tool to make an object follow a complex path, like e.g. a sheet of paper inside a printer, a film inside a camera, the water of a canal...

The *Curve Deform* works on a (global) dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction, the mesh will traverse along the curve. Moving the mesh in an orthogonal direction will move the

mesh object closer or further away from the curve. The default settings in Blender map the Y axis to the dominant axis. When you move the object beyond the curve endings the object will continue to deform based on the direction vector of the curve endings.

If the "curve path" is *3D*, the *Tilt* value of its control points will be used (see the *Extrusion* section above) to twist the "curved" object around it. Unfortunately, the other *Radius* property is not used (it would have been possible, for example, to make it control the size of the "curved" object...).

Tip: Try to position your object over the curve immediately after you have added it, before adding the curve deform. This gives the best control over how the deformation works.

Note: Use modifiers!

The *Curve Deform* relationship is now also a modifier, called *Curve*. The *Curve* modifier function acts the same as its counterpart, except that when the modifier is used, the "dominant axis" is set inside its properties – and the *Track* X, Y, Z buttons no longer have an effect on it. And you have some goodies, like the possibility, if "curving" a mesh, to only curve one of its vertex groups...

Interface

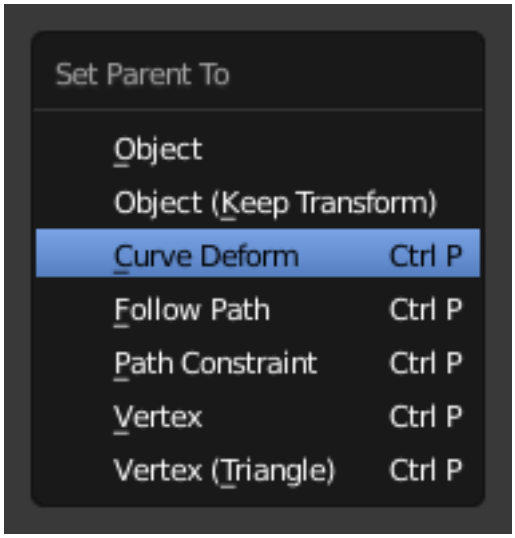


Fig. 2.671: Make Parent menu.

When parenting an object (mesh, curve, meta, ...) to a curve `Ctrl-P`, you will be presented with a menu (*Make Parent* menu).

By selecting *Curve Deform*, you enable the curve deform function on the mesh object.

The dominant axis setting is set on the mesh object. By default the dominant axis in Blender is the Y Axis. This can be changed by selecting one of the *Track* X, Y or Z buttons in the *Animation* Panel, (*Animation settings panel*), in Object tab.

Cyclic (or closed) curves work as expected where the object deformations traverse along the path in cycles. Note however that when you have more than one curve in the "parent" object, its "children" will only follow the first one.

The *Stretch* curve option allows you to let the mesh object stretch, or squeeze, over the entire curve. This option is in *Object Data* properties.

Fig. 2.672: Animation settings panel.

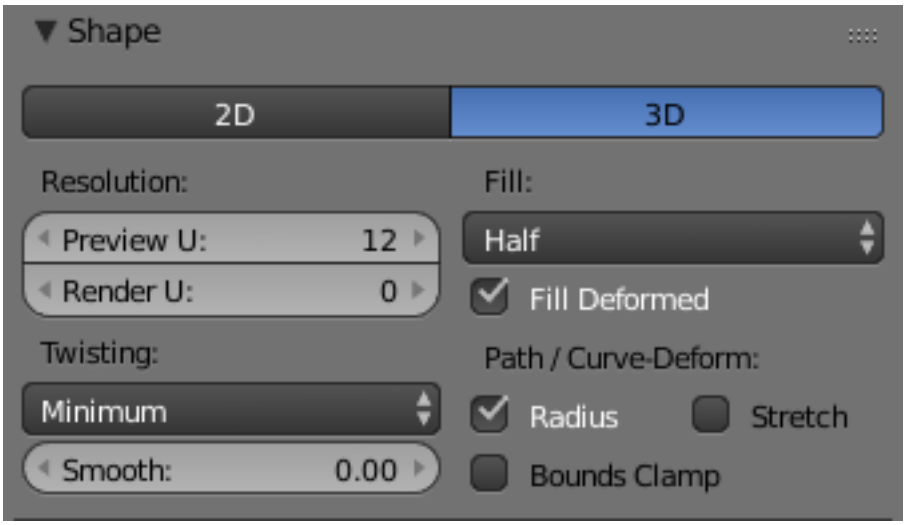


Fig. 2.673: Curve Shape Panel.

Example

Let us make a simple example:

- Remove default cube object from scene and add a Monkey with *Add* → *Mesh* → *Monkey*
- Press `Tab` to exit *Edit Mode*.
- Now add a curve with *Add* → *Curve* → *Bézier Curve*.

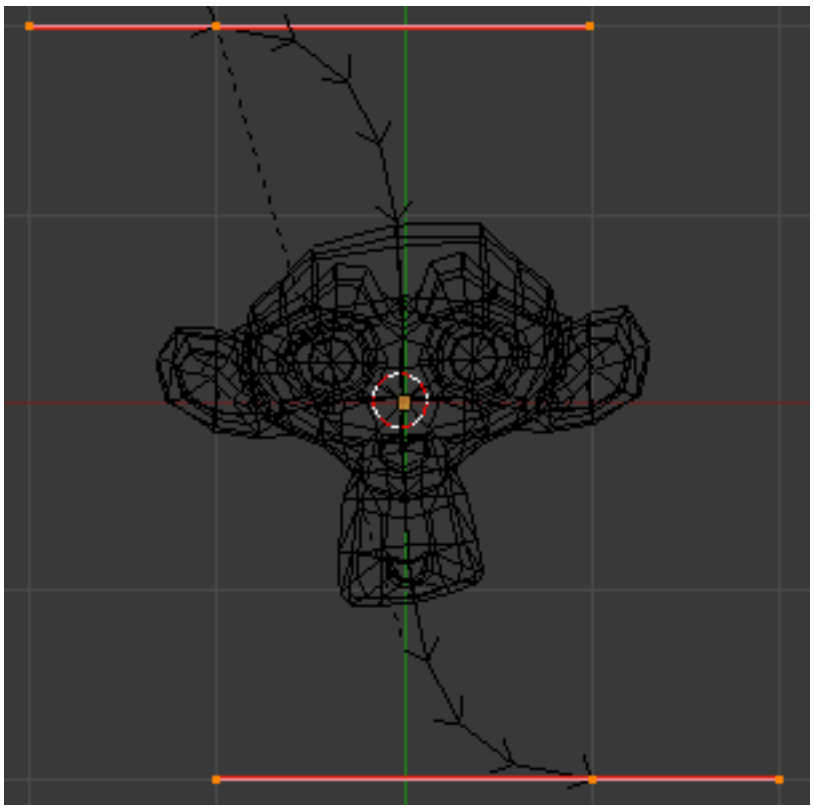


Fig. 2.674: Edit Curve.

- While in *Edit Mode*, move the control points of the curve as shown in Fig. *Edit Curve.*, then exit *Edit Mode* `Tab`.
- Now, you can use the new, modern, modifier way of "curving" the Monkey:
  - Select the Monkey `RMB`.
  - In the *Object Modifiers* properties, *Modifiers* panel, add a *Curve* modifier.
  - Type the name of the curve (should be "Curve") in the *Ob* field of the modifier, and optionally change the dominant axis to Y.
- Or you can choose the old, deprecated method (note that it creates a "virtual" modifier...):
  - Select the Monkey `RMB`, and then shift select the curve `Shift-RMB`.
  - Press `Ctrl-P` to open up the *Make Parent* menu.
  - Select *Make Parent* → *Curve Deform*.
- The Monkey should be positioned on the curve, as in Fig. *Monkey on a Curve.*.
- Now if you select the Monkey `RMB`, and move it `G`, in the Y-direction (the dominant axis by default), the monkey will deform nicely along the curve.

Tip: If you press `MMB` (or one of `X`, `Y`, `Z`) while moving the Monkey you will constrain the movement to one axis only.

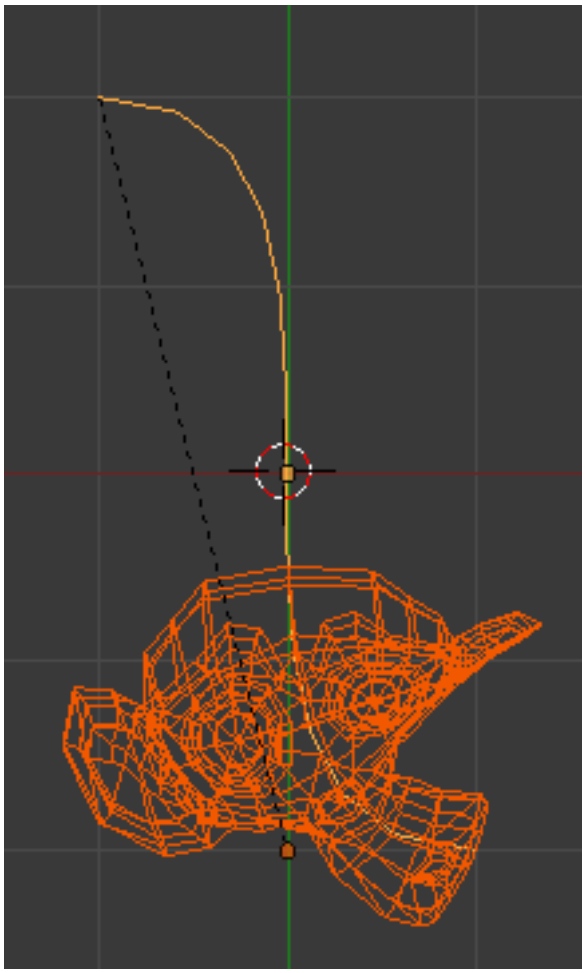


Fig. 2.675: Monkey on a Curve.

- In Fig. *Monkey deformations.*, you can see the Monkey at different positions along the curve.

Tip: Moving the Monkey in directions other than the dominant axis will create some odd deformations. Sometimes this is what you want to achieve, so you will need to experiment and try it out!

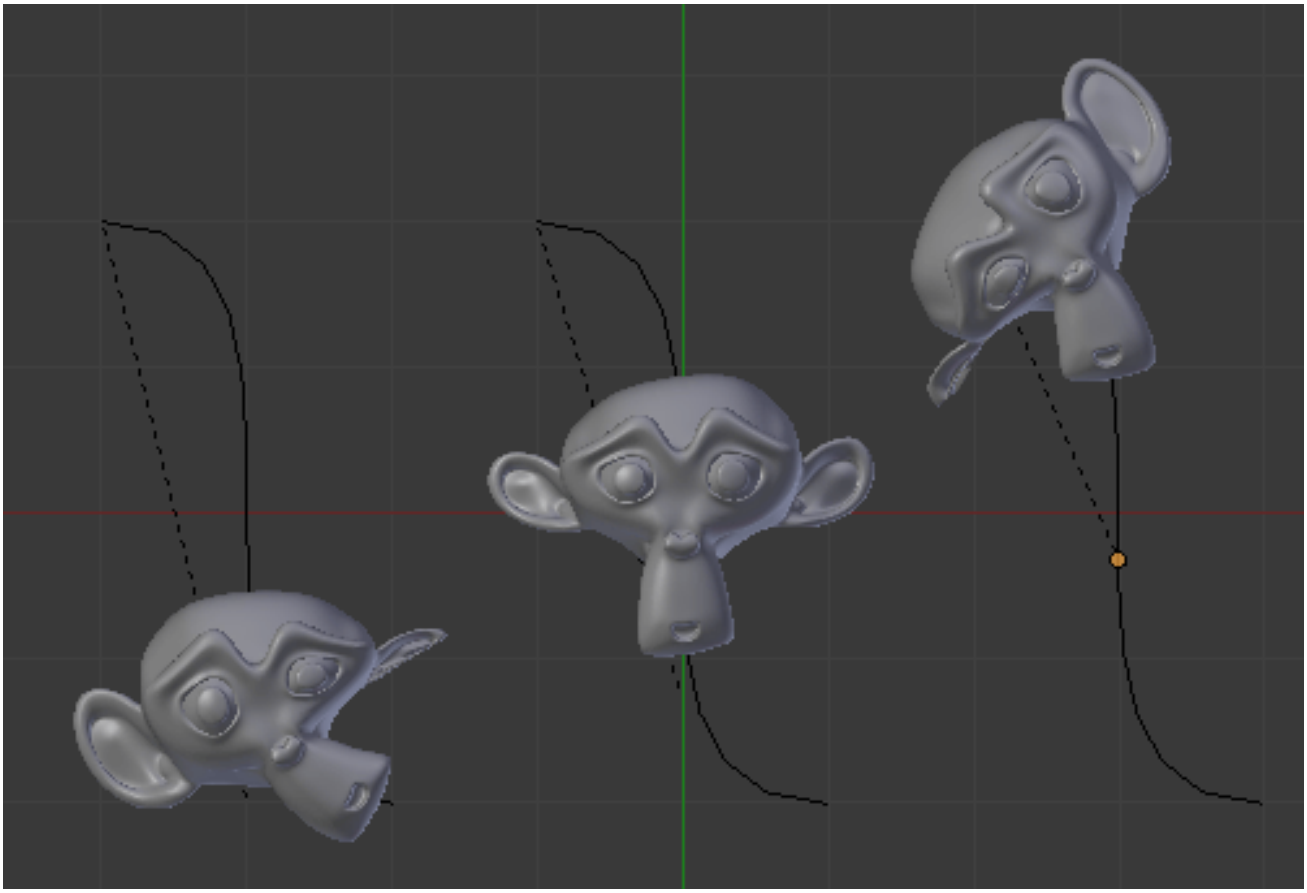


Fig. 2.676: Monkey deformations.

Curve Extrusion

Reference

Mode: Object or Edit Mode
Panel: *Curve and Surface*

Extrude Turns an one dimensional curve into a two dimensional curve by giving it height. Note that this is not related to *Extrude* used in mesh edit-mode. With a scale of one, an *Extrusion* of 0.5 will extrude the curve 0.5 BU in both directions, perpendicular to the curves normals.

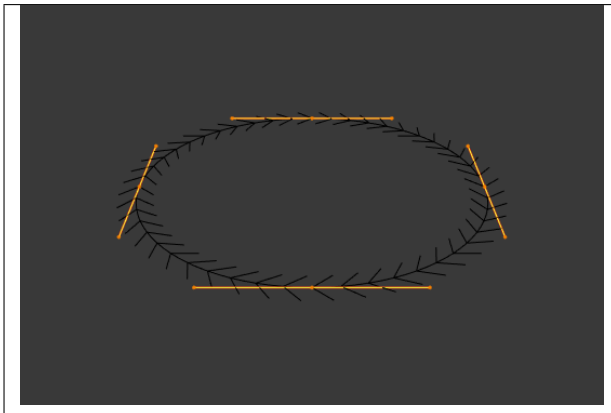


Fig. 2.677: Bézier Circle 0.0 extrude (Edit Mode).

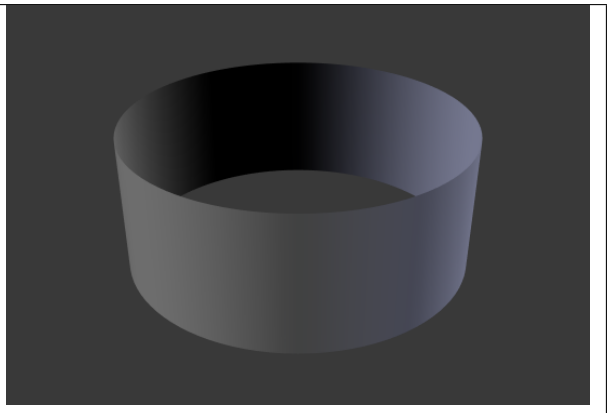


Fig. 2.678: Extruded by 0.5 (Object Mode).

Tilt This setting controls how the normals (visualization: arrows) twist around each control point – so it is only relevant with 3D curves! You set it using the *Tilt* transform tool in the `T` tool shelf, the `N` → *transform* → *Mean tilt*, *Curve* → *Transform* → *Tilt*.

You can also reset it to its default value (i.e. perpendicular to the original curve plane) with `Alt-T`, *Curve* → *Control Points* → *Clear Tilt*. With NURBS, the tilt is always smoothly interpolated. However, with Bézier, you can choose the interpolation algorithm between Linear, Ease, B-Spline, and Cardinal, in the *Properties Editor* → *Object Data* → *Active Spline* → *Tilt*.

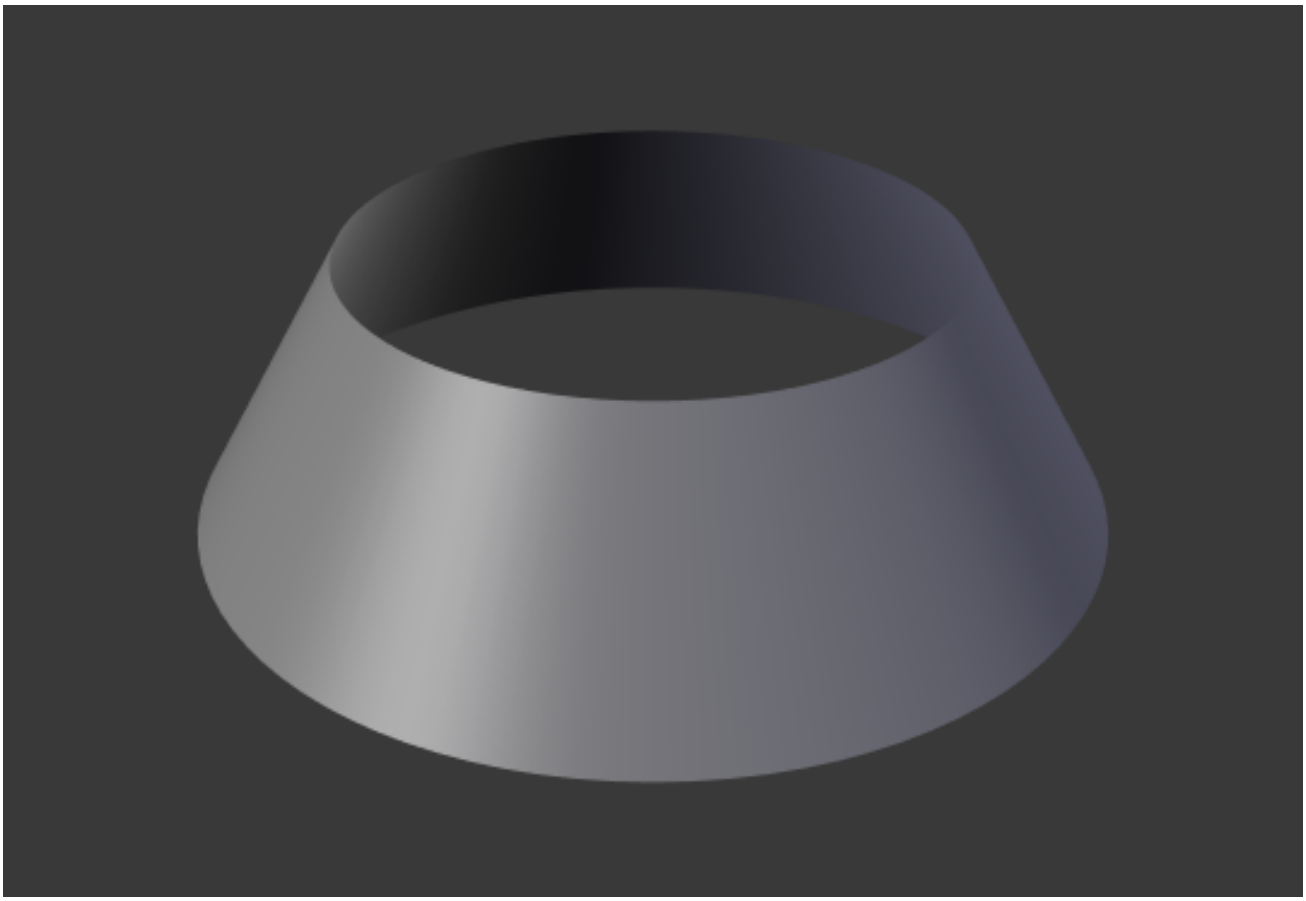


Fig. 2.679: 30 degree Mean Tilt of all control points.

Bevel Depth This will add a bevel to the extrusion. See below for its effects... Note that the bevel makes the extrusion wider and higher. If set to 0.0, there is no bevel.

Bevel Resolution Controls the resolution of the bevel created by a *Bevel Depth* higher than zero. If set the to 0 (the

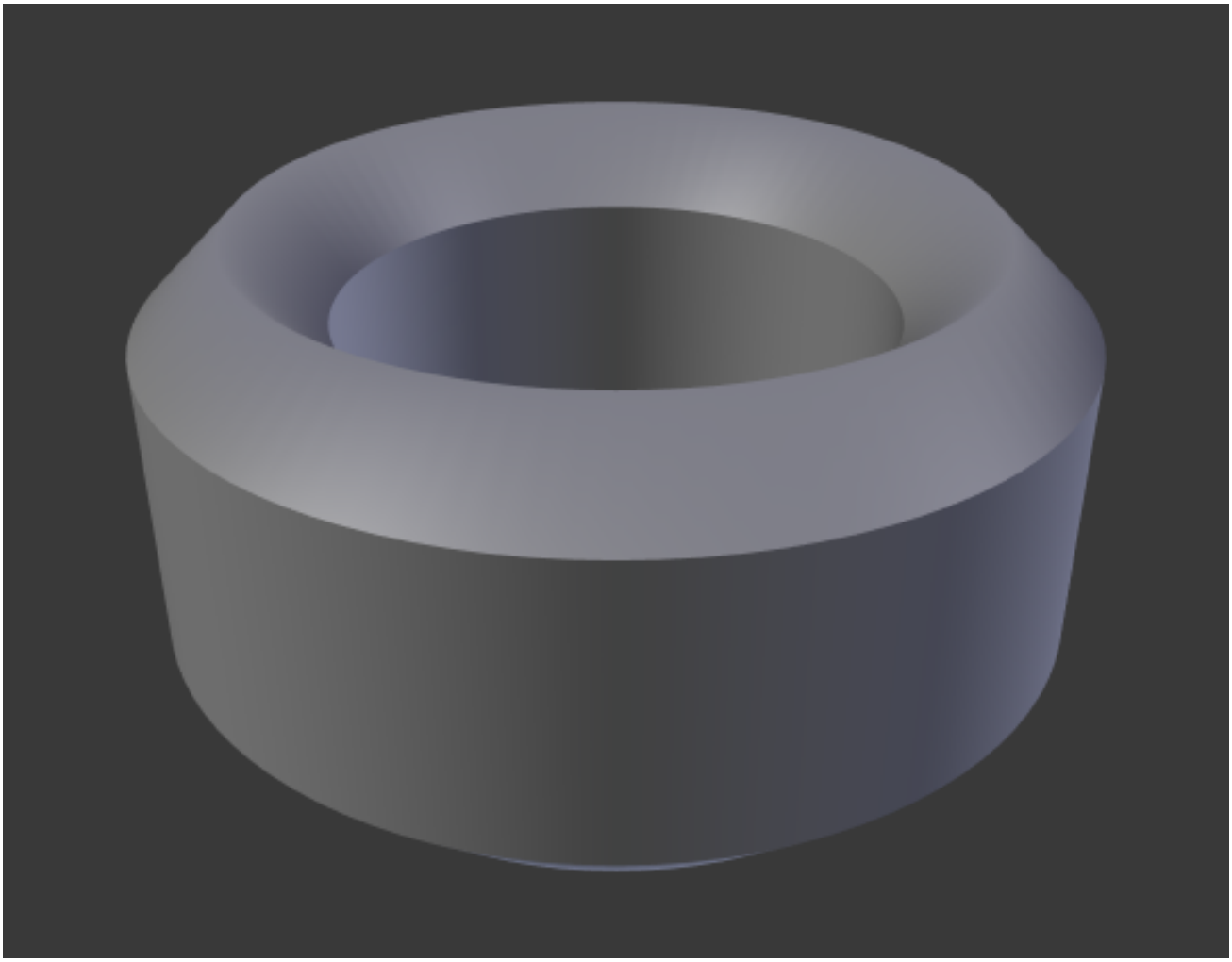


Fig. 2.680: Bevel depth of 0.25, fill set to full, zero Mean Tilt.

default), the bevel is a simple "flat" surface. Higher values will smooth, round off the bevel, similar to the resolution settings of the curve itself...

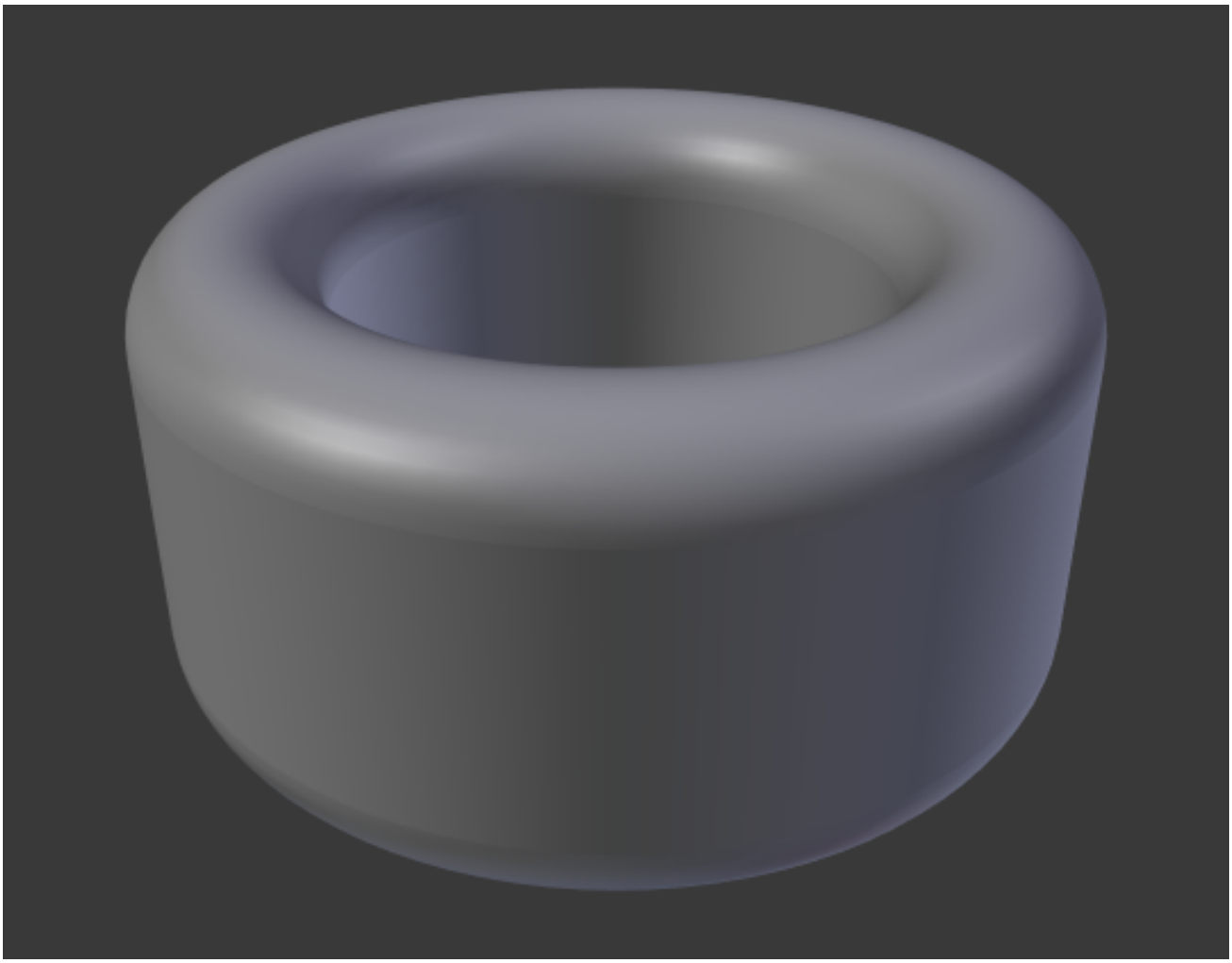


Fig. 2.681: Bevel resolution set to 10.

Offset Moves the extrusion parallel to the curve normals. *Almost like scaling*

Radius The Radius allows you to directly control the width of the extrusion along the "spinal" curve. The *Radius* of the points is set using the *Shrink/Fatten Radius* transform tool `Alt-S`, the *Curve* → *Transform* → *Shrink/Fatten Radius*, or the `N` → *transform* → *Radius*.

Tip: Remember, these curves can be converted into meshes with `Alt-C` in Object Mode

We have three sub-classes of results, depending on whether the curve is open or closed or 3D:

Open 2D Curve The extrusion will create a "wall" or "ribbon" following the curve shape. If using a *Bevel Depth*, the wall becomes a sort of slide or gutter. If your normals are facing the wrong way you can switch their direction as shown *here*

Closed 2D Curve This is probably the most useful situation, as it will quickly create a volume, with (by default) two flat and parallel surfaces filling the two sides of the extruded "wall". You can remove one or both of these faces by choosing the fill mode: both, front, back, or none.

The optional bevel depth will always create a 90 degree bevels here.

3D Curve Here the fact that the curve is closed or not has no importance – you will never get a volume with an extruded 3D curve, only a wall or ribbon, like with open 2D curves.

Fig. 2.682: -1 offset, 0.5 extrusion, 0.25 Bevel Depth, 10 Bevel resolution.

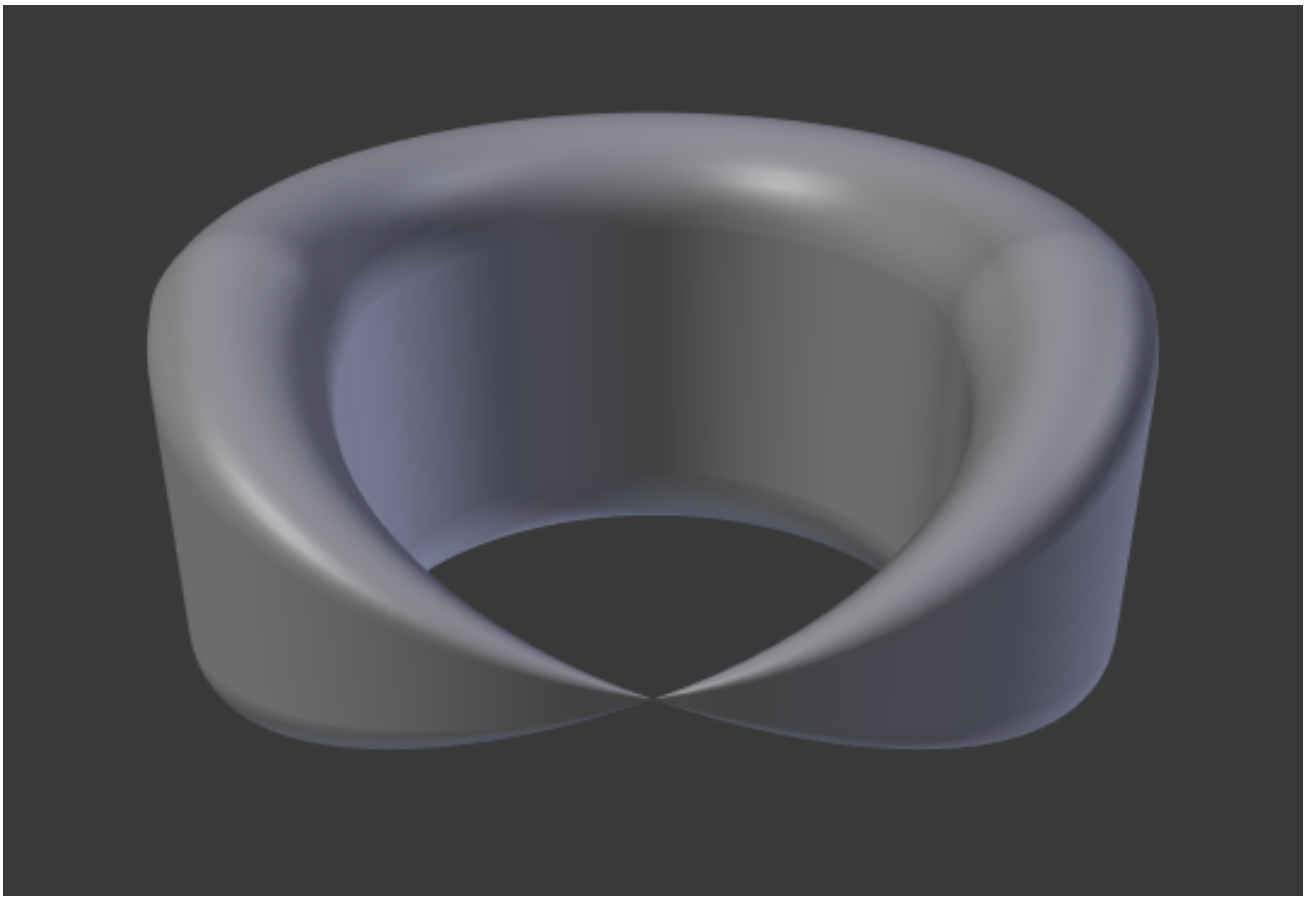


Fig. 2.683: One control point radius set to zero.

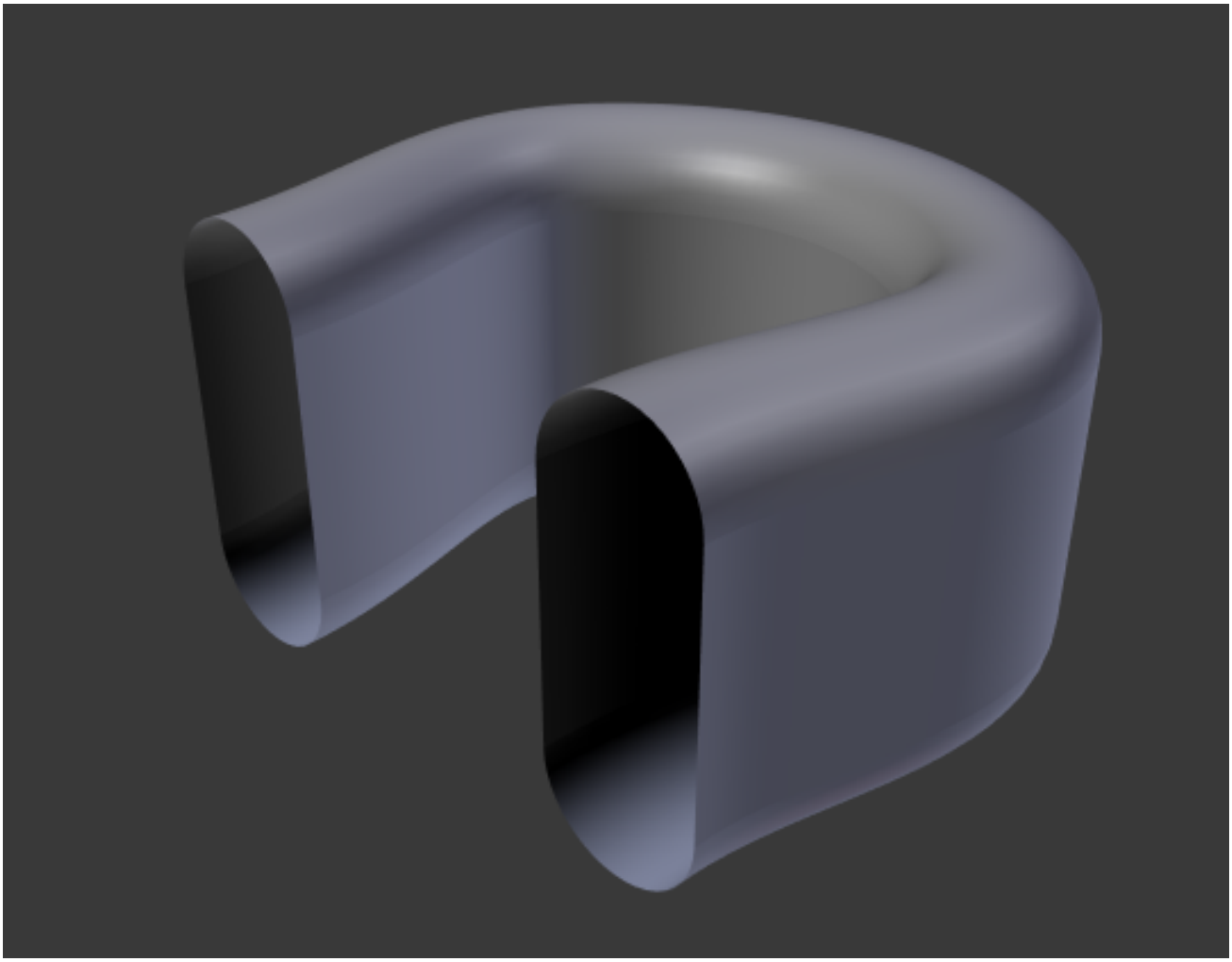


Fig. 2.684: Open 2D Curve with Alt-C, fill set to none, zero offset, 0.5 extrusion, 0.25 Bevel Depth, 10 Bevel resolution.

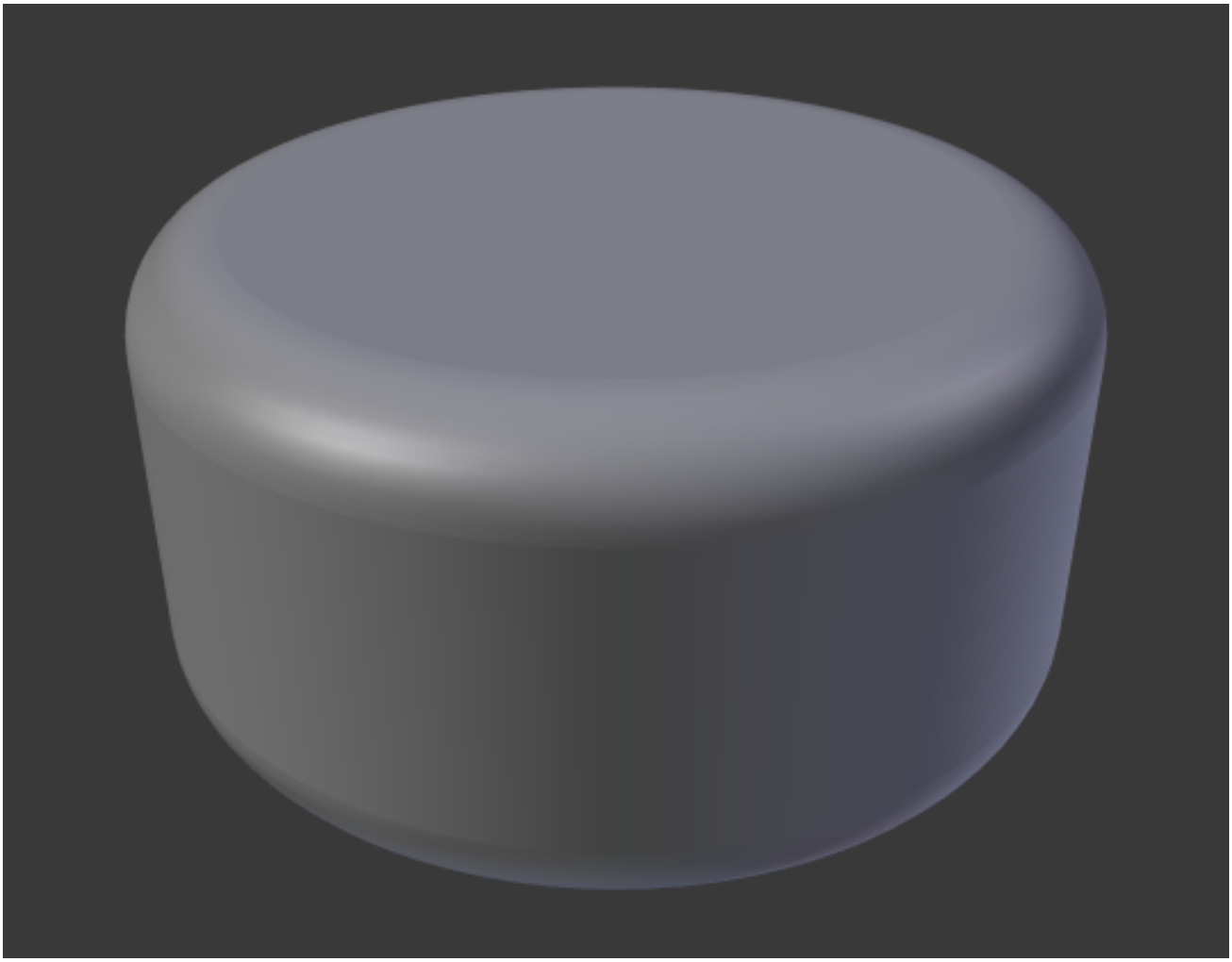


Fig. 2.685: Closed 2D Curve, 0.5 extrude, 0.25 Bevel Depth, 10 Bevel resolution, Fill: Both.

However, there is one more feature with 3D curves: the *Tilt* of the control points (see above). It will make the ribbon twist around the curve to create a mobius strip, for example.

Advanced Extrusion

These extrusions use one or two additional curve objects, to create very complex organic shapes.

To enable this type of extrusion, you have to type a valid curve object name in the *Bevel Object* field of the curve you are going to use as the "spinal column" of your extrusion. The "bevel" curve will control the cross section of the extruded object. Whether the *Bevel Object* curve is 2D or 3D has no importance, but if it is closed, it will create a "tube-like" extrusion; otherwise you will get a sort of gutter or slide object...

The object is extruded along the whole length of all internal curves. By default, the width of the extrusion is constant, but you have two ways to control it, the *Radius* property of control points (see above), and the "taper" object.

The taper curve is evaluated along *the local X axis*, using *the local Y axis* for width control. Note also that:

- It must be an *open curve*.
- The taper is applied independently to all curves of the extruded object.
- Only the first curve in a *Taper Object* is evaluated, even if you have several separated segments.
- The scaling starts at the first control-point on the left and moves along the curve to the last control-point on the right.
- Negative scaling, (negative local Y on the taper curve) is possible as well. However, rendering artifacts may appear.
- Might need to increase the curve resolution to see more detail of the taper
- With closed curves, the taper curve in *Taper Object* acts along the whole curve (perimeter of the object), not just the length of the object, and varies the extrusion depth. In these cases, you want the relative height of the *Taper Object* Taper curve at both ends to be the same, so that the cyclic point (the place where the endpoint of the curve connects to the beginning) is a smooth transition.

Examples

Let us taper a simple curve circle extruded object using a taper curve. Add a curve, then exit *Edit Mode*. Add another one (a closed one, like a circle); call it "BevelCurve", and enter its name in the *Bevel Object* field of the first curve (*Curve and Surface* tab). We now have a pipe. Add a third curve while in *Object Mode* and call it "TaperCurve". Adjust the left control-point by raising it up about 5 units.

Now return to the Object tab, and edit the first curve's *Taper Object* field in the Curve and Surface panel to reference the new taper curve which we called "TaperCurve". When you hit enter the taper curve is applied immediately, with the results shown in Fig. *Circle curve set as Bevel Object.*.

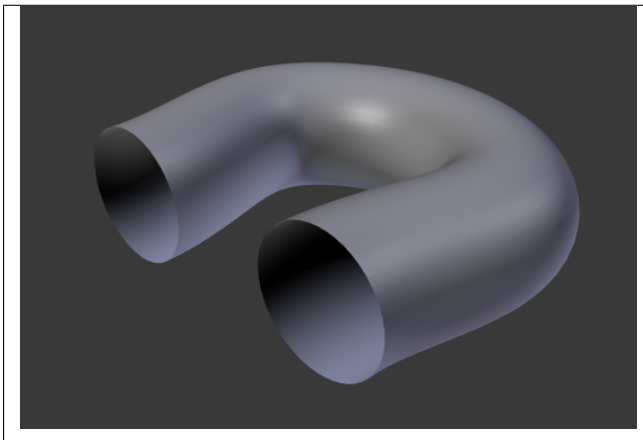


Fig. 2.686: Circle curve set as Bevel Object.

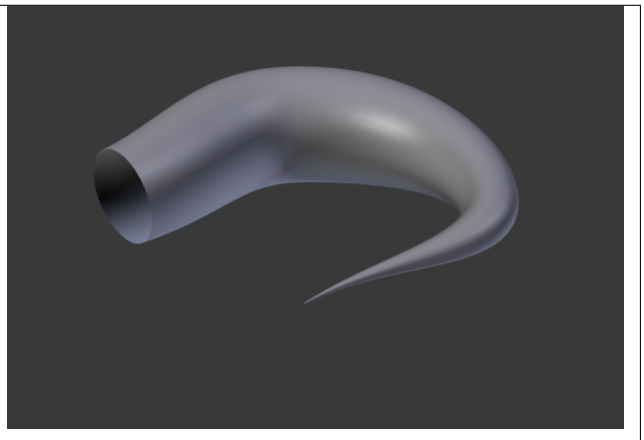


Fig. 2.687: Taper extruded curve.

You can see the *taper curve* being applied to the *extruded object*. Notice how the pipe's volume shrinks to nothing as the taper curve goes from left to right. If the taper curve went below the local Y axis the pipe's inside would become the outside, which would lead to rendering artifacts. Of course as an artist that may be what you are looking for!

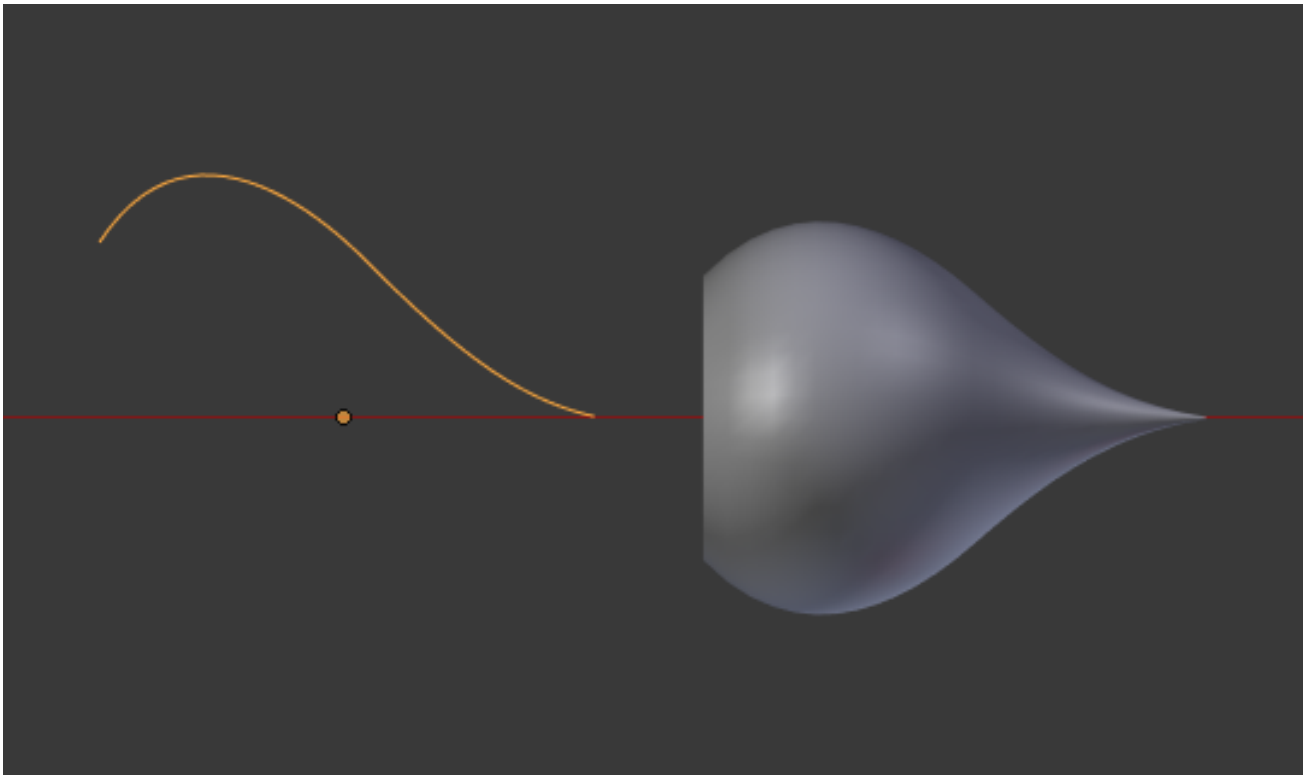


Fig. 2.688: Taper example 1.

In Fig. *Taper example 1.* you can clearly see the effect the left taper curve has on the right curve object. Here the left taper curve is closer to the object center and that results in a smaller curve object to the right.

In Fig. *Taper example 2.* a control point in the taper curve to the left is moved away from the center and that gives a wider result to the curve object on the right.

In Fig. *Taper example 3.* we see the use of a more irregular taper curve applied to a curve circle.

Draw Curve

Reference

Mode: Edit Mode
Panel: *Tool Shelf* → *Create* → *Draw Curve*
Menu: *Add* → *Draw Curve*
Hotkey: `Shift-LMB`

The Curve draw tool allows you to freehand draw curves.

Stroke Options

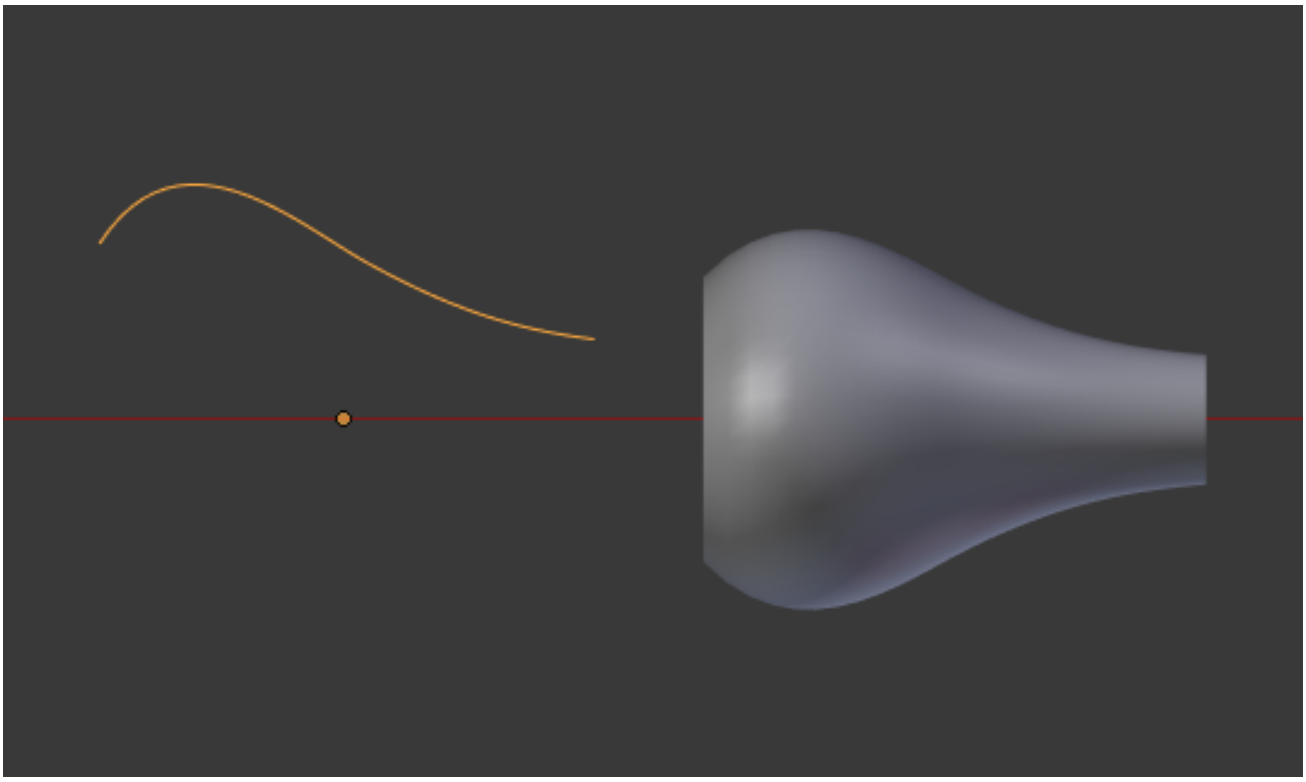


Fig. 2.689: Taper example 2.

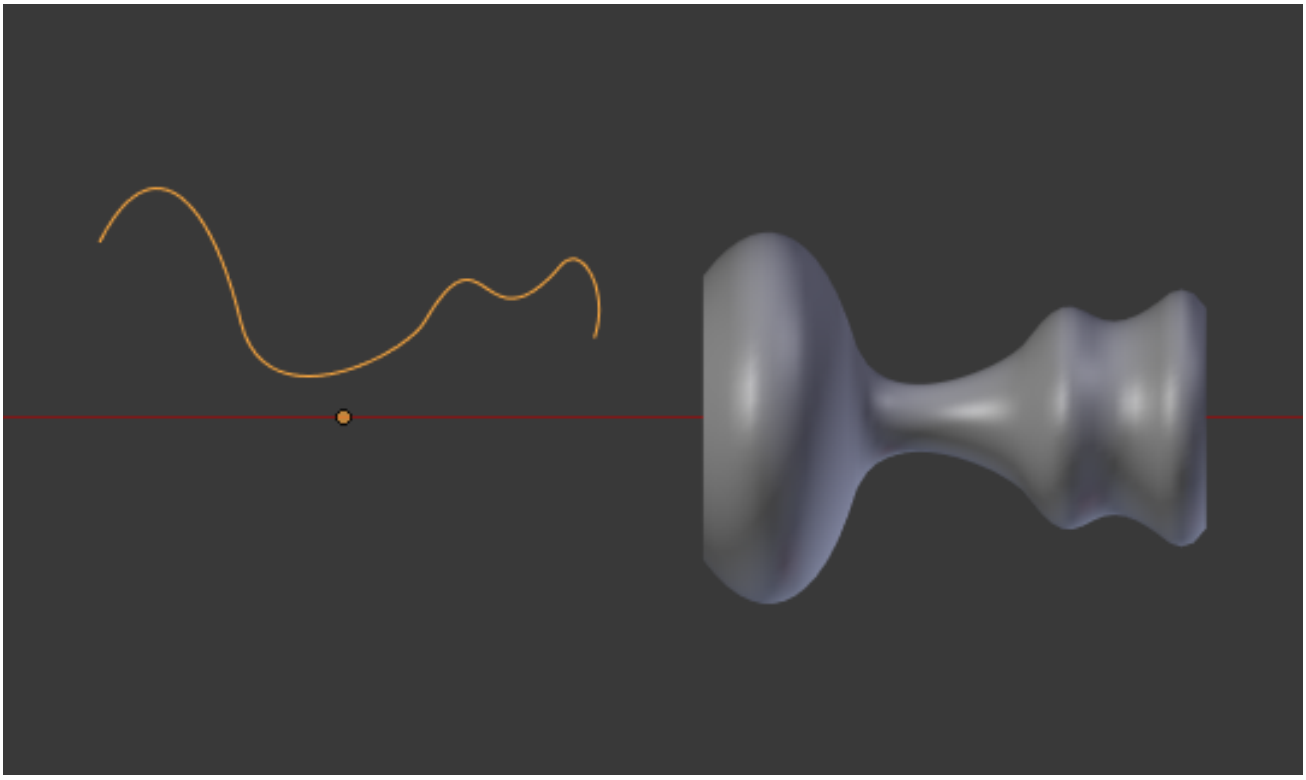


Fig. 2.690: Taper example 3.

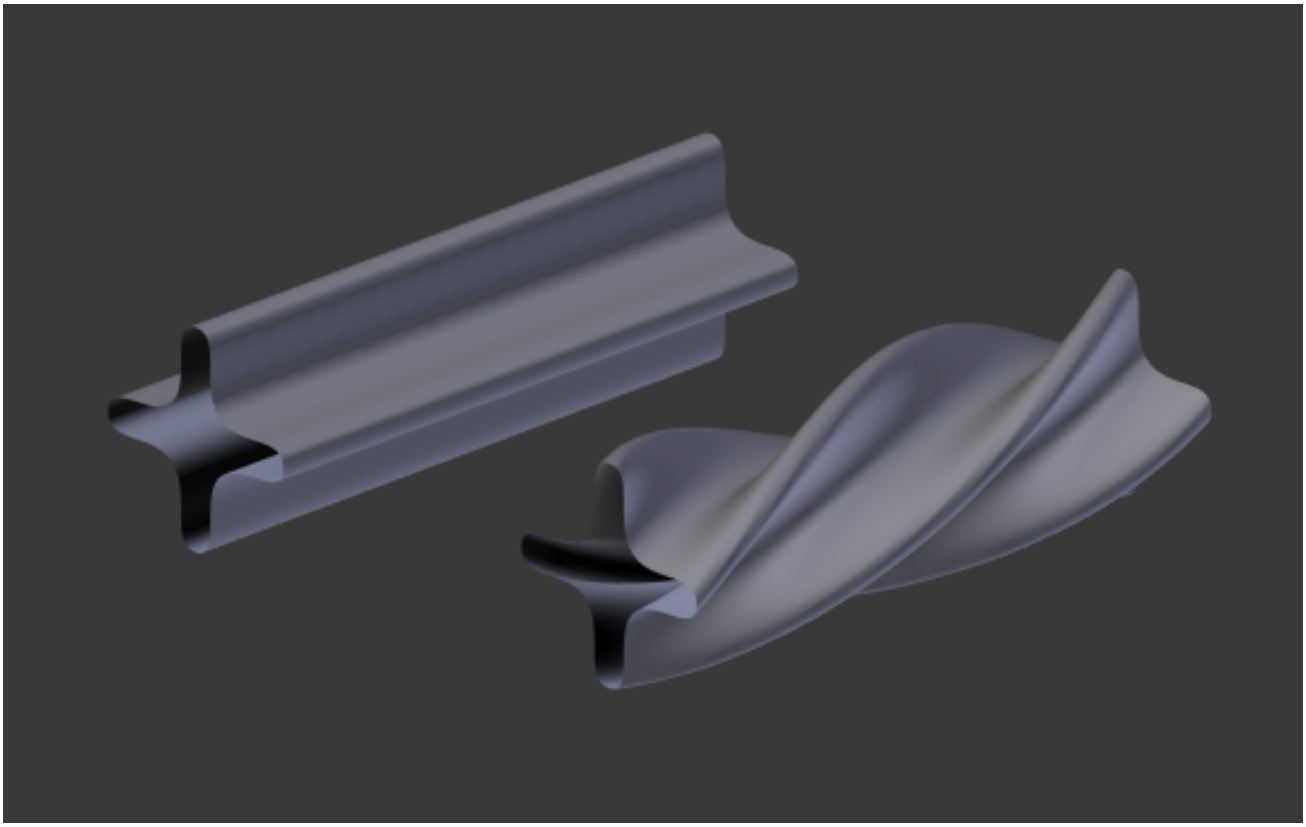


Fig. 2.691: Bevel extrusion with *Tilt* example.

These options can be found in *Tool Shelf* → *Options* → *Curve Stroke*.

Type Type of curve to use for drawing.

Poly Bézier Curve with strait line segments (auto handles).

Bézier

Tolerance Lower values give a result that is closer to the drawing stroke, while higher values give more smoothed results.

Method

Refit Incrementally re-fits the curve (gives best results).

Split Splits the curve until the tolerance is met (gives better drawing performance).

Detect Corners Detects corners and uses non-aligned handles for them.

Corner Angle Any angles above this are considered corners.

Pressure Radius

Min Minimum radius when the minimum pressure is applied (also the minimum when tapering)

Max Radius to use when the maximum pressure is applied (or when a tablet is not used).

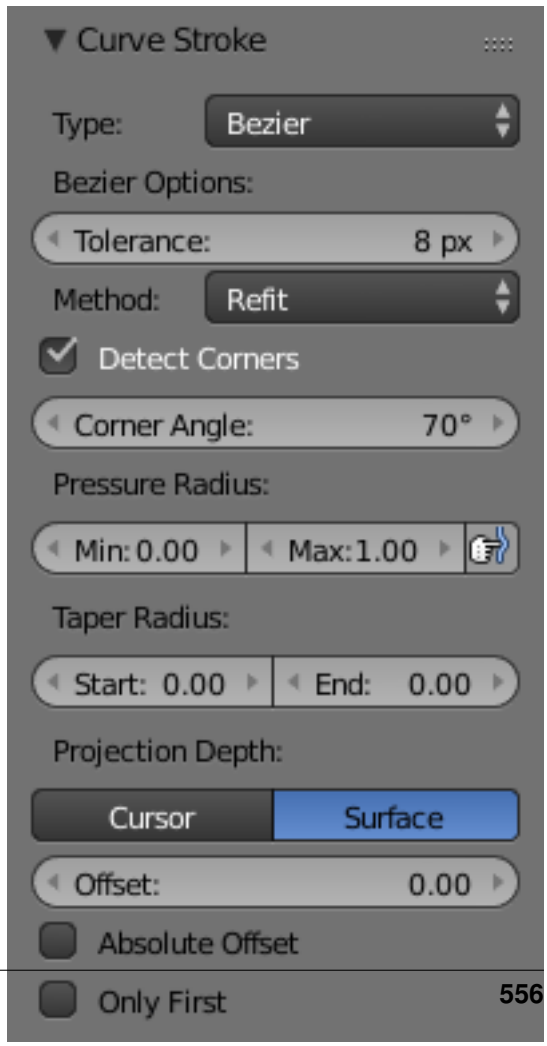


Fig. 2.692: Curve Stroke panel.

Projection Depth Options to control how where/how the curves are drawn.

Cursor Uses the depth under the cursor to draw curves.

Surface Used to draw on top of other objects.

Offset Distance to offset the curve from the surface.

Absolute Offset Applies a fixed offset (does not scale by the curve radius).

Only First Only uses the start of the stroke for the depth.

Normal/View Draws perpendicular to the surface.

Normal/Surface Draws aligned to the surface.

View Draws aligned to the viewport.

Draw Options

These option can be found in the *Redo Last Panel*.

Error Error distance in object units. This can be seen similar to a subdivision rate for the curve. Lower values give a result that is closer to the drawing stroke while higher values give more smoothed results.

Fit Method

Refit Incrementally re-fits the curve (gives best results).

Split Splits the curve until the tolerance is met (gives better drawing performance).

Corner Angle Any angles above this are considered corners.

Cyclic Toggles whether or not the curve is *Cyclic*.

Curve Display

Reference

Mode: Edit Mode
Panel: *Properties region* → *Curve Display*

When in Edit Mode, the Properties region contains options in the *Curve Display* panel for how curves are displayed in the 3D View.

Handles Toggles the option to draw the Bézier handles.

Normals Toggles the display of the curve normals.

Normal Size Length of the axis that points the direction of the normal.

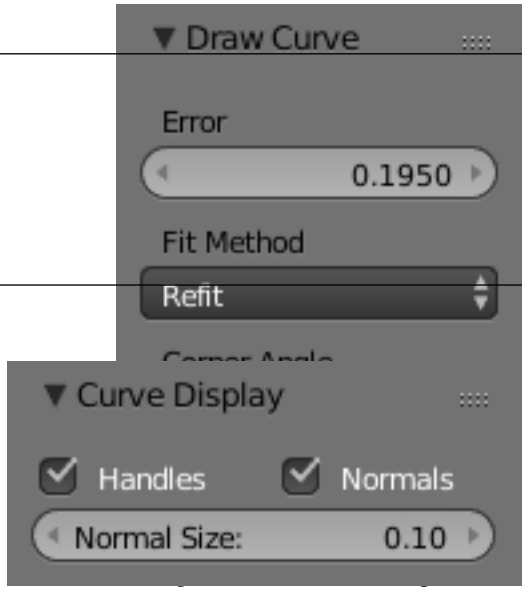


Fig. 2.694: Curve Display Panel.

2.4.4 Surfaces

Introduction

Curves are 2D objects, and surfaces are their 3D extension. Note however, that in Blender, you only have NURBS surfaces, no Bézier (you have the *Bézier* knot type, though; see below), nor polygonal (but for these, you have meshes!). Even though curves and surfaces share the same object type (with texts also...), they are not the same thing; for example, you cannot have in the same object both curves and surfaces.

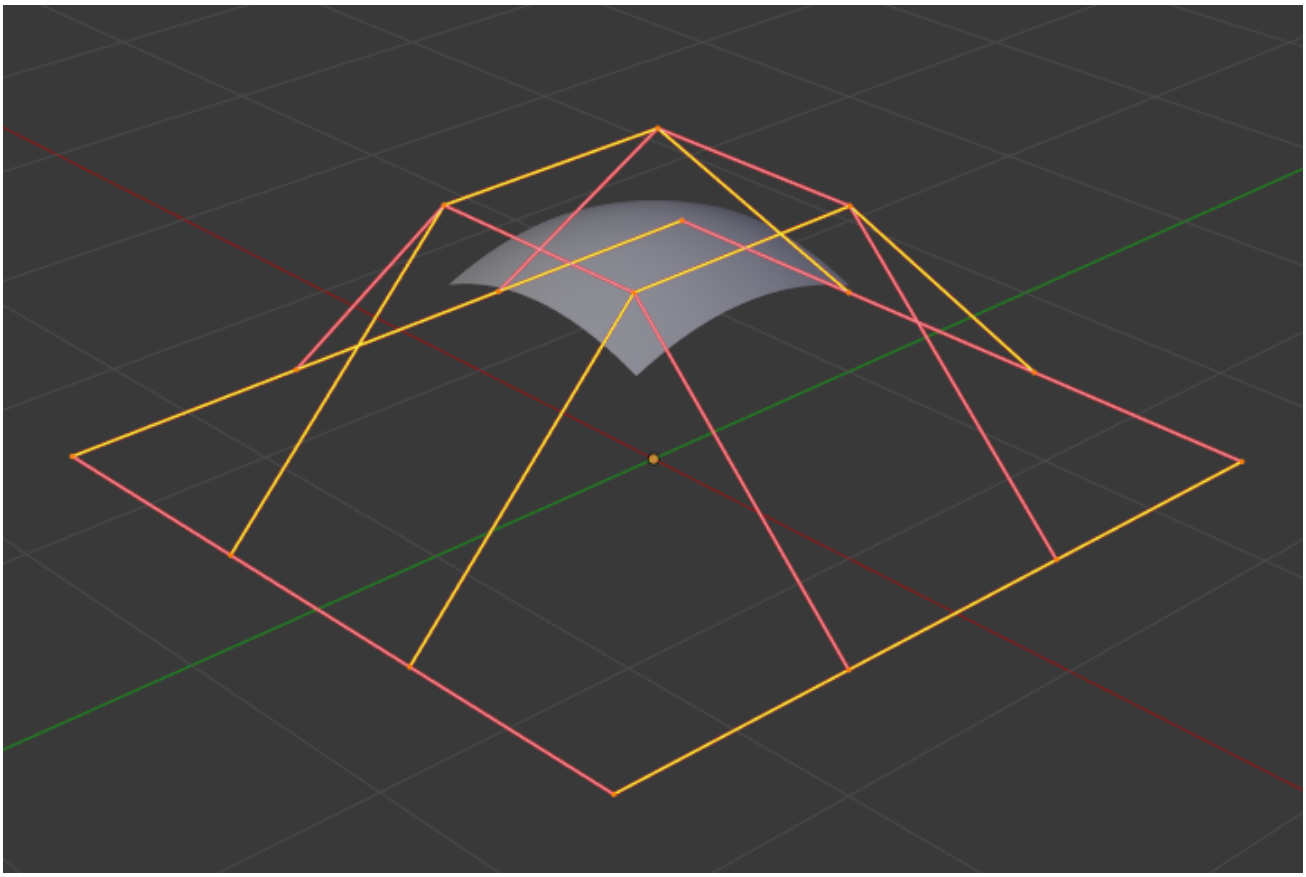


Fig. 2.695: Nurbs surface in Edit Mode.

As surfaces are 2D, they have two interpolation axes, U (as for curves) and V. It is important to understand that you can control the interpolation rules (knot, order, resolution) *independently* for each of these two dimensions (the U and V fields for all these settings, of course).

You may ask yourself "but the surface appears to be 3D, why is it only 2D?". In order to be 3D, the object needs to have "Volume", and a surface, even when it is closed, does not have volume; it is infinitely thin. If it had a volume the surface would have a thickness (its third dimension). Hence, it is only a 2D object, and has only two interpolation dimensions or axes or coordinates (if you know a bit of math, think of non-euclidean geometry – well, surfaces are just non-euclidean 2D planes...). To take a more "real life" example, you can roll a sheet of paper to create a cylinder; well, even if it "draws" a volume, the sheet itself will remain a (nearly...) 2D object!

In fact, surfaces are very similar to the results you get when *extruding a curve*.

Finding Surface Tools

As usual, you have the *Select* and *Surface* menus in the 3D View headers, and the *Specials* `W` pop-up one.

Visualization

There is nearly no difference from NURBS curves, except that the U direction is indicated by yellow grid lines, and the V one is materialized by pink grid lines, as you can see in Fig. *Nurbs surface in Edit Mode.*.

You can *hide and reveal* control points just as with curves.

Structure

Many of the concepts from *curves*, especially *NURBS* ones, carry directly over to NURBS surfaces, such as control points, *Order*, *Weight*, *Resolution*, etc. Here we will just talk about the differences.

It is very important to understand the difference between NURBS curves and NURBS surfaces: the first one has one dimension, the latter has two. Blender internally treats NURBS surfaces and NURBS curves completely differently. There are several attributes that separate them but the most important is that a NURBS curve has a single interpolation axis (U) and a NURBS surface has two interpolation axes (U and V).

However, you can have "2D" surfaces made of curves (using the *extrusion tools*, or, to a lesser extent, the filling of closed 2D curves). And you can have "1D" curves made of surfaces, like a NURBS surface with only one row (either in U or V direction) of control points produces only a curve...

Visually you can tell which is which by entering *Edit Mode* and looking at the 3D View header: either the header shows *Surface* or *Curve* as one of the menu choices. Also, you can *extrude* a whole NURBS surface curve to create a surface, but you cannot with a simple NURBS curve.

Control Points, Rows and Grid

Control points for NURBS surfaces are the same as for NURBS curves. However, their layout is quite constraining. The concept of "segment" disappears, replaced by "rows" and the overall "grid".

A "row" is a set of control points forming one "line" in one interpolation direction (a bit similar to *edge loops* for meshes). So you have "U-rows" and "V-rows" in a NURBS surface. The key point is that *all* rows of a given type (U or V) have the *same* number of control points. Each control point belongs to exactly one U-row and one V-row.

All this forms a "grid", or "cage", the shape of which controls the shape of the NURBS surface. A bit like a *lattice*...

This is very important to grasp: you cannot add a single control point to a NURBS surface; you have to add a whole U- or V-row at once (in practice, you will usually use the Extrude tool, or perhaps the Duplicate one, to add those...), containing exactly the same number of points as the others. This also means that you will only be able to "merge" different pieces of surfaces if at least one of their rows match together.

Weight

Guess what? Yes, it works exactly like *NURBS Curves*! *Weight* specifies how much each control point "pulls" on the curve.

In Fig. *One control point with a weight of 5.* a single control point, labeled "C", has had its *Weight* set to 5.0 while all others are at their default of 1.0. As you can see, that control point *pulls* the surface towards it.

If all the control points have the same *Weight* then each effectively cancels each other out. It is the difference in the weights that cause the surface to move towards or away from a control point.

The *Weight* of any particular control point is visible in the *Transform Properties* `N`, in the *W* field (and not the *Weight* field...).

Preset Weights

NURBS can create pure shapes such as circles, cylinders, and spheres (note that a Bézier circle is not a pure circle). To create pure circles, globes, or cylinders, you must set to specific values the weights of the control points. Some of which

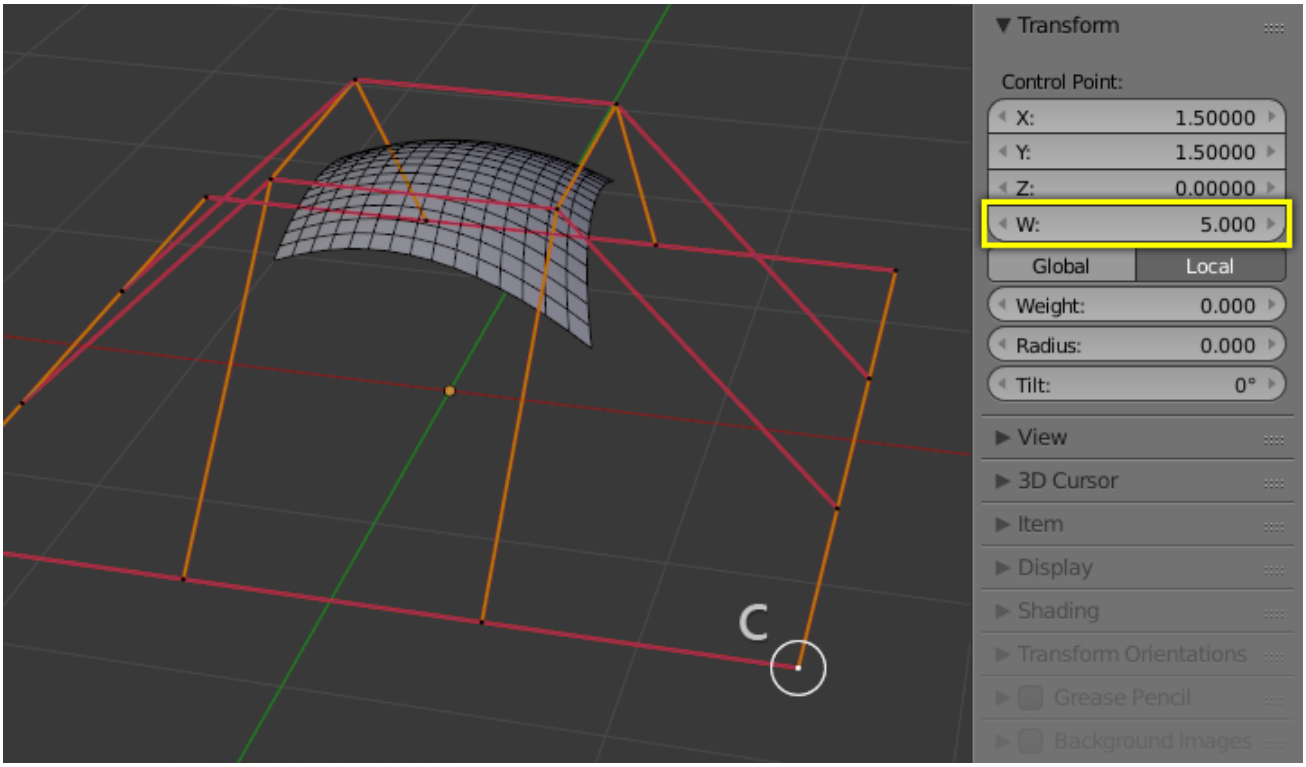


Fig. 2.696: One control point with a weight of 5.

are provided as presets in the *Curve Tools* panel (lower right corner). This is not intuitive, and you should read more on NURBS before trying this.

To create a sphere with 2D surfaces, its the same principle as with a 2D circle. You will note that the four different weights needed for creating a sphere (1.0, 0.707 = sqrt(0.5), 0.354 = sqrt(2)/4, and 0.25).

Primitives

To help get started in creating surfaces there are four preset NURBS surfaces, found in the *Add* → *Surface* → *NURBS Surface*, *NURBS Tube*, *NURBS Sphere* and *NURBS Torus*.

There are also two preset NURBS surface curves (with only one control point on each V-row): *NURBS Curve* and *NURBS Circle*.

Note how a circle NURBS surface is never filled, unlike its "real" curve counterpart...

Properties

The panels of the *Curve and Surface* tab are the same as for *curves*, just with fewer options...

Shape

You can adjust the resolution separately for both preview and render, to not slow things down in the viewport, but still get good render results.

Preview U, V

Render U, V

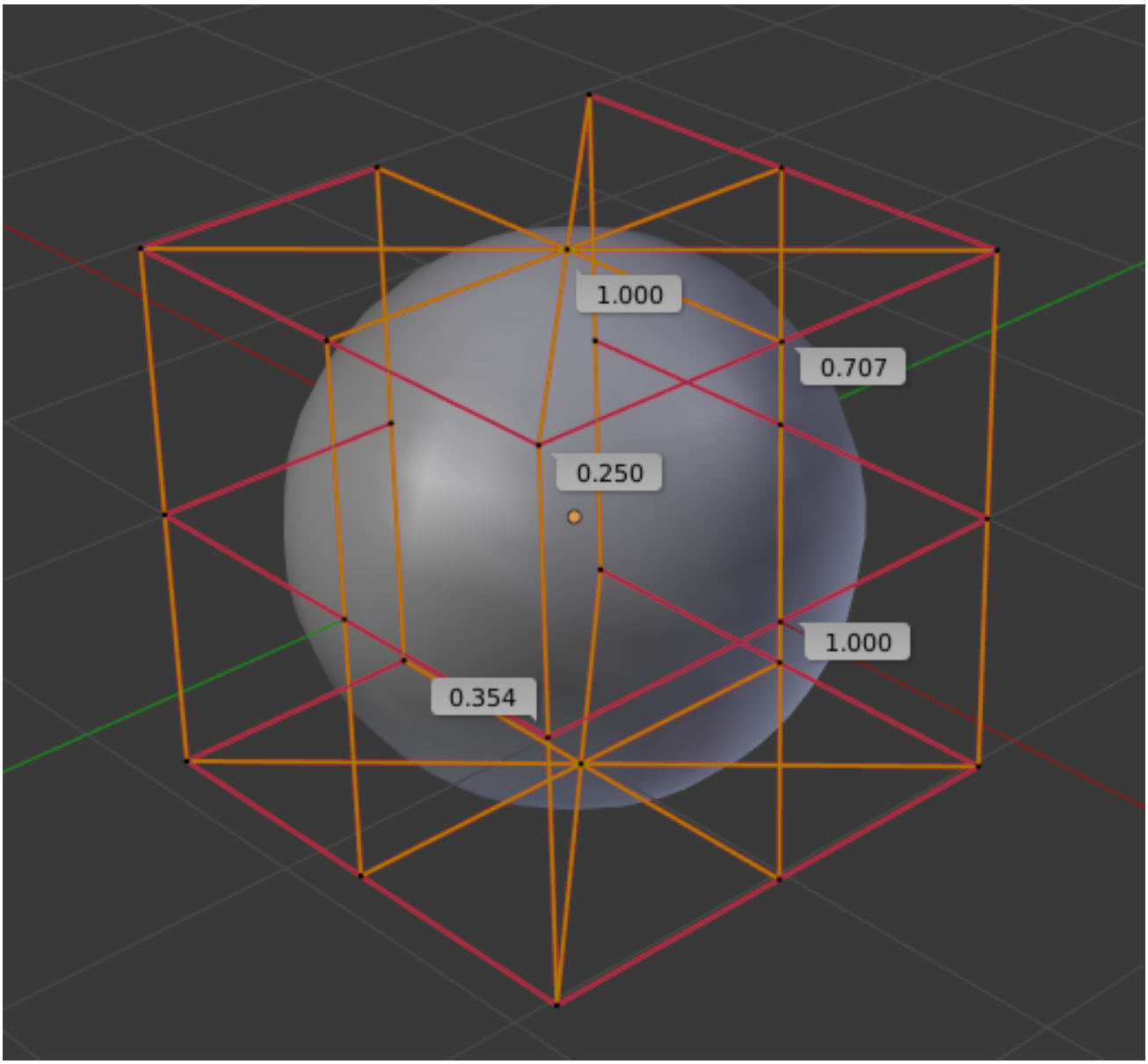


Fig. 2.697: A sphere surface.

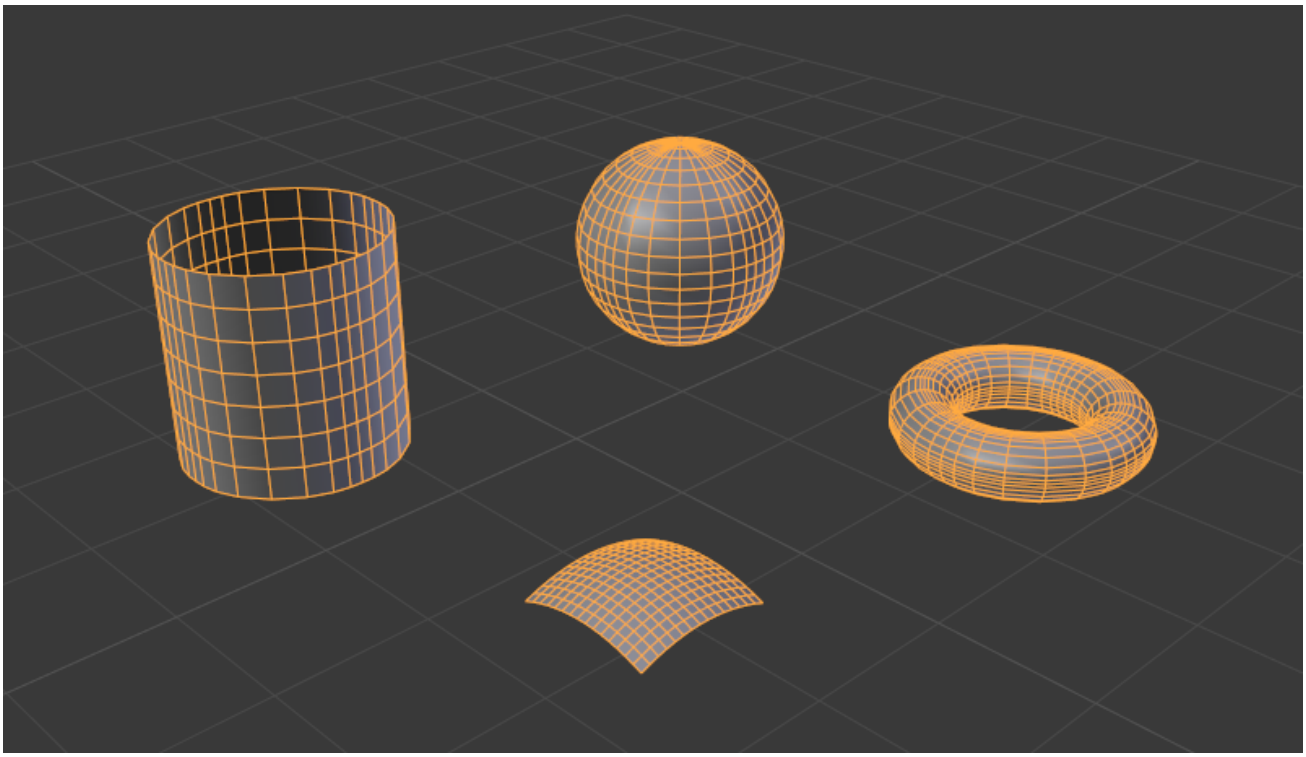


Fig. 2.698: NURBS surface primitives.

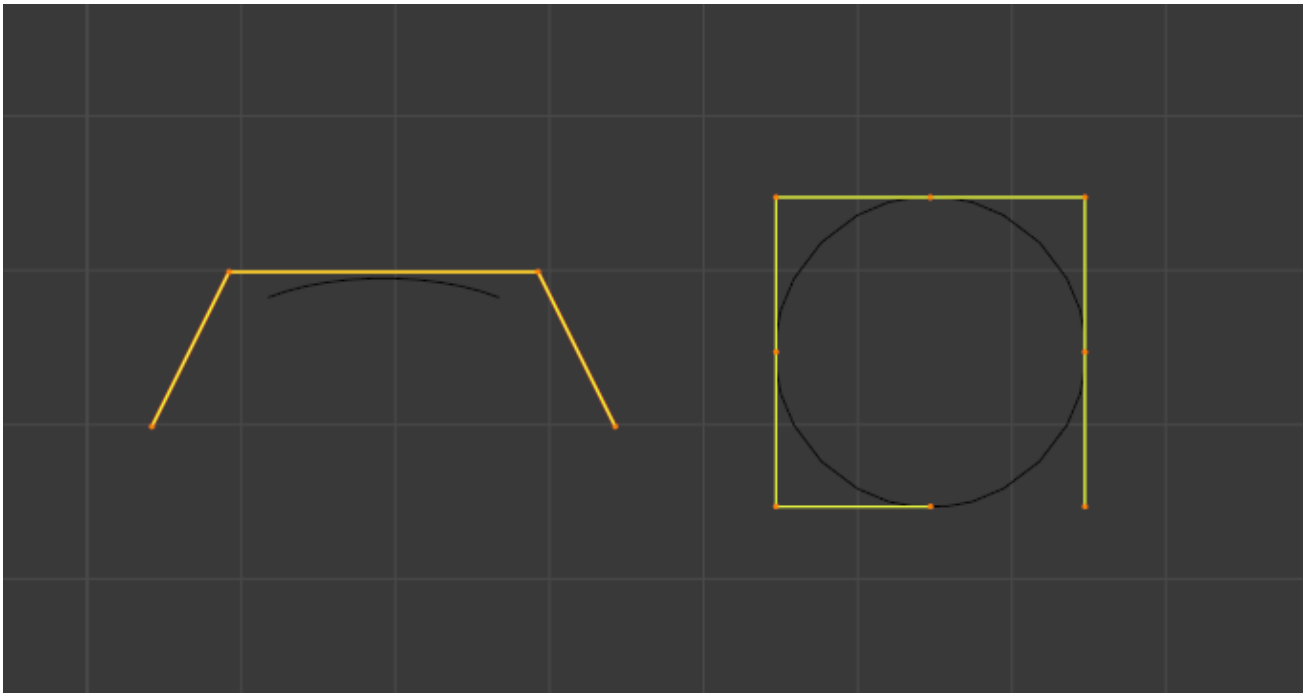


Fig. 2.699: NURBS curve primitives.

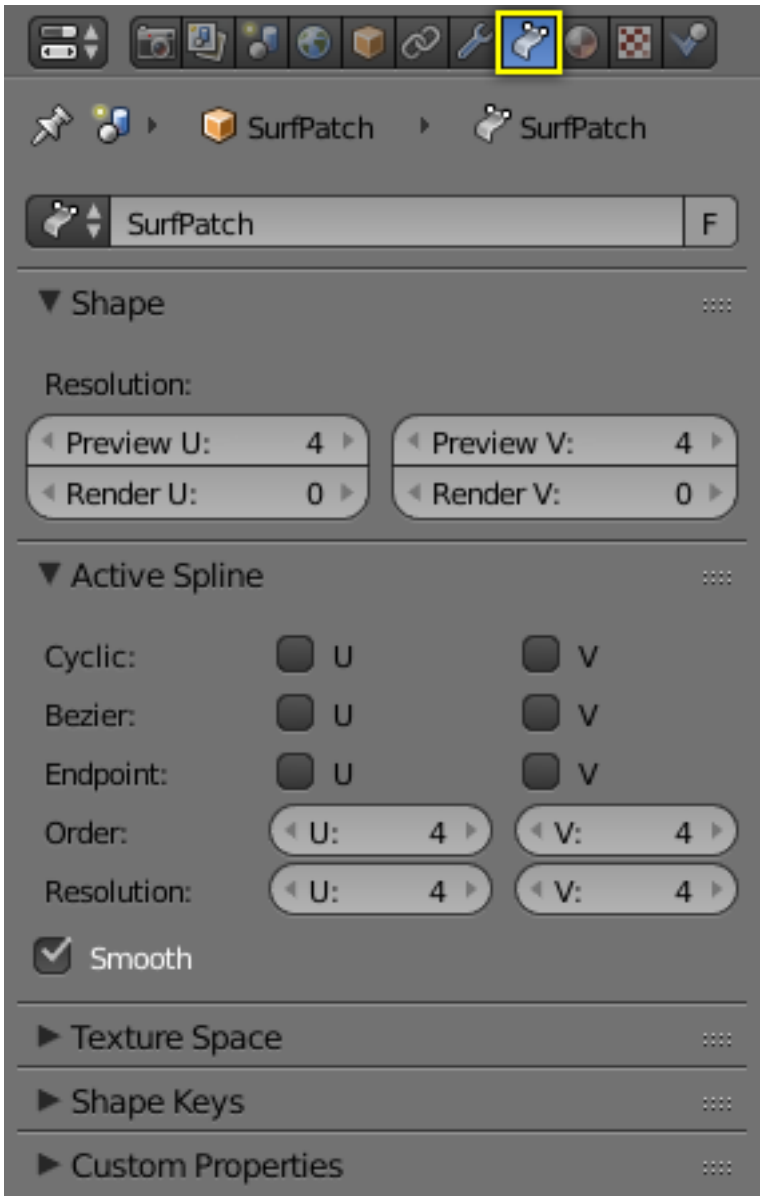


Fig. 2.700: Surface Properties.

Fig. 2.701: Shape panel.

Active Spline

Closed and Open Surfaces

Like curves, surfaces can be closed (cyclical) or open, independently in both directions, allowing you to easily create a tube, donut or sphere shape, and they can be drawn as "solids" in *Edit Mode*. This makes working with surfaces quite easy.

Bézier

Endpoint

Just like with *NURBS curves*, NURBS surfaces have two knot vectors, one for each U and V axis. Here again, they can be one of *Cyclic*, *Endpoint*, or *Bézier*, with the same properties as for curves. And as with curves, only open surfaces (in the relevant direction) are affected by this setting...

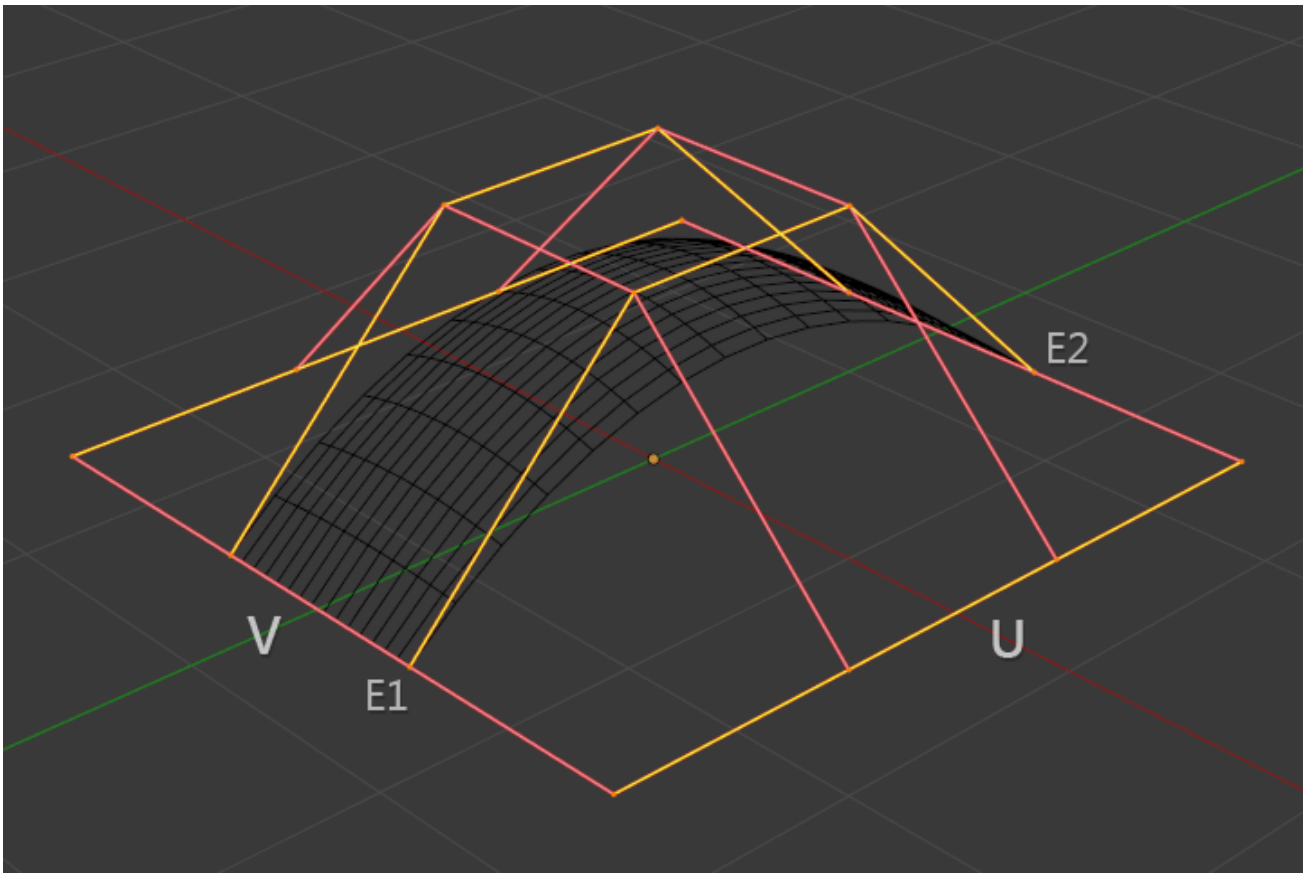


Fig. 2.702: Endpoint U.

In Fig. *Endpoint U.* the U interpolation axis is labeled as "U" and the V interpolation axis is labeled as "V". The U's interpolation axis has been set to *Endpoint* and as such the surface now extends to the outer edges from E1 to E2 along the U interpolation axis.

To cause the surface to extend to all edges you would set the V's axis to *Endpoint* as well.

Order

One more time, this property is the same as with *NURBS Curves*; it specifies how much the control points are taken into account for calculating the curve of the surface shape. For high Orders 1 the surface pulls away from the control points,

creating a smoother surface by assuming that the *Resolution* is high enough. For lowest Orders 2 the surface follows the control points, creating a surface that tends to follow the grid cage.

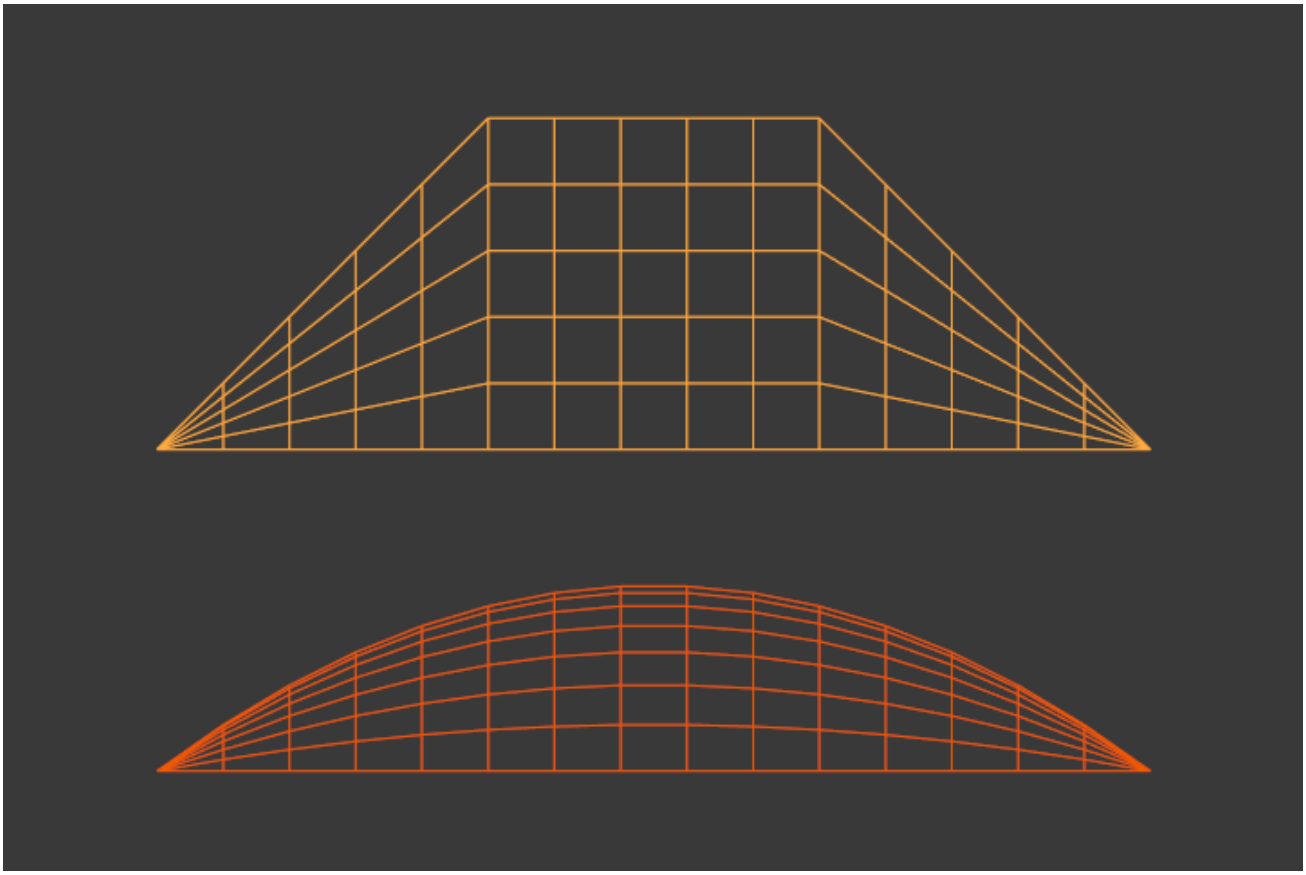


Fig. 2.703: Order 2 and order 4 surface.

For illustration purposes, in both Fig. *Order 2 and order 4 surface.*, the knot vectors were set to *Endpoint*, causing the surface to extend to all edges.

You can set independently the order for each interpolation axis, and like curves, it **cannot** be lower than 2, and higher than 6 or the number of control points on the relevant axis.

Resolution

Just like *NURBS curves*, *Resolution* controls the detail of the surface. The higher the *Resolution* the more detailed and smoother the surface is. The lower the *Resolution* the rougher the surface. However, here you have two resolution settings, one for each interpolation axis (U and V). Note that unlike with curves, you have only one resolution (the *Resolution* U and V fields, in the *Curve Tools* panel)...

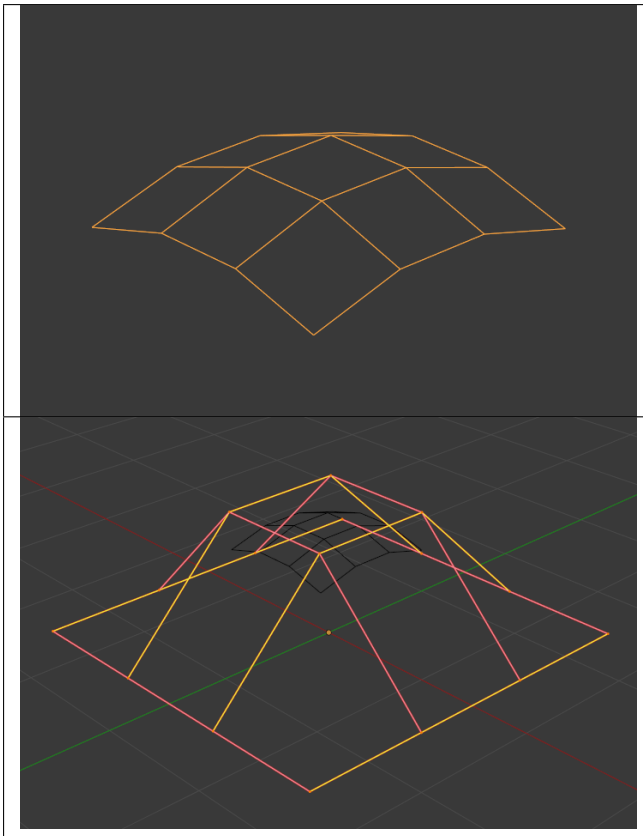


Fig. 2.704: Resolution 1×1.

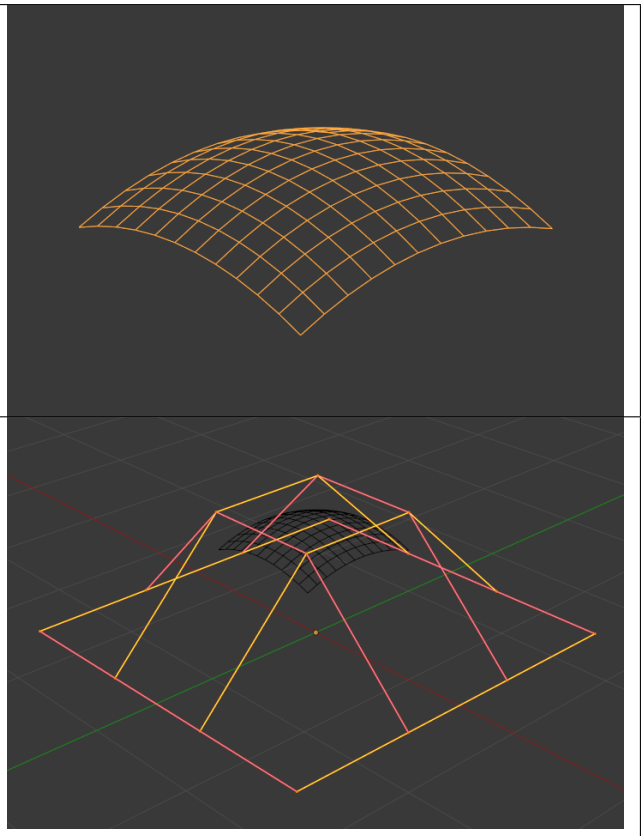


Fig. 2.705: Resolution 3×3.

Fig. *Resolution 1×1.* is an example of a surface resolution of 1 for both U and V. Fig. *Resolution 3×3.* surface is an example of a surface resolution of 3 for both U and V.

Smooth

Selecting

Surface selection in *Edit Mode* is very similar to *NURBS curve selection*. The basic tools are the same as with *meshes*, so you can select a simple control point with a `LMB` -click, add to current selection with `Shift-LMB` -clicks, `B` order-select, and so on.

`L`, `Ctrl-L` will add to the selection the mouse cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same surface.

Select Menu

The *Select* menu (3D View headers) is even simpler than for curves...

All these options have the same meaning and behavior as in *Object Mode* (and the specificities of *Border Select* in *Edit Mode* have already been discussed *here*).

Control Point Row

Reference

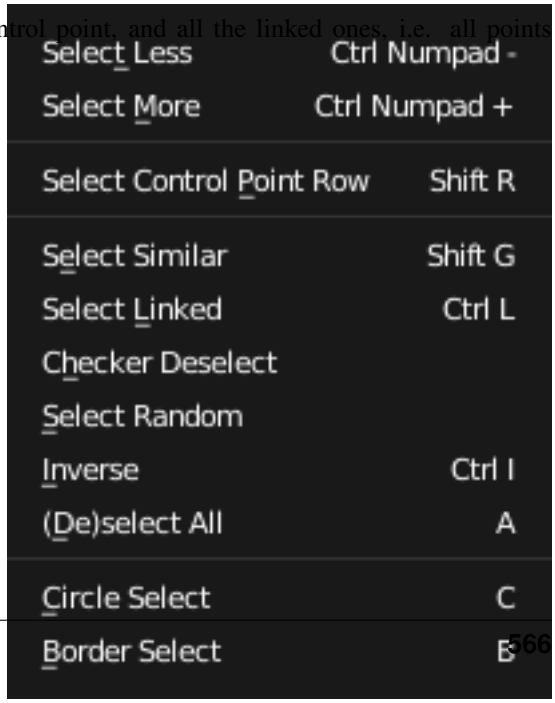


Fig. 2.706: Select Menu.

Mode: Edit Mode
Menu: *Select* → *Control Point Row*
Hotkey: `Shift-R`

This option works a bit like *edge loop selection* for meshes, inasmuch it selects a whole *row* of control points, based on the active (the last selected) one. The first time you press `Shift-R`, the V-row passing through (containing) the active point will be added to the *current* selection. If you use again this shortcut, you will toggle between the U- and V-row of this point, removing *everything else* from the selection.

More and Less

Reference

Mode: Edit Mode
Menu: *Select* → *More/Less*
Hotkey: `Ctrl-NumpadPlus` / `Ctrl-NumpadMinus`

These two options are complementary and very similar to *those for meshes*. Their purpose, based on current selected control points, is to reduce or enlarge this selection.

The algorithm is the same as with meshes:

More for each selected control point, select **all** its linked points (i.e. two, three or four).

Less for each selected control point, if **all** points linked to this point are selected, keep it selected. For all other selected control points, de-select them.

This implies two points:

- First, when **all** control points of a surface are selected, nothing will happen (as for *Less*, all linked points are always selected, and of course, *More* cannot add any). Conversely, the same goes when no control point is selected.
- Second, these tools will never "go outside" of a surface (they will never "jump" to another surface in the same object).

Editing

Surface editing has even fewer tools and options than its curve counterpart, but has many common points with it... So this page covers (or tries to cover) all the subjects, from the basics of surface editing to more advanced topics, like retopology.

Translation, Rotation, Scale

Reference

Mode: Edit Mode
Menu: *Surface* → *Transform* → *Grab/Move, Rotate, Scale, ...*
Hotkey: `G`, `R`, `S`

Once you have a selection of one or more control points, you can grab/move G, rotate R or scale S them, like many other things in Blender, as described in the *Manipulation in 3D Space* section.

You also have in *Edit Mode* an extra option when using these basic manipulations: the *proportional editing*.

Advanced Transform Tools

Reference

Mode: Edit Mode
Menu: *Surface* → *Transform*

The *To Sphere*, *Shear*, *Warp* and *Push/Pull* transform tools are described in the *Mesh Transformation* section. Surfaces have no specific transform tools.

NURBS Control Points Settings

Reference

Mode: Edit Mode
Panel: Curve Tools (Edit Mode) and Transform

We saw in a *previous page* that NURBS control points have a weight, which is the influence of this point on the surface. You set it either using the big *Set Weight* button in the *Curve Tools* panel (after having defined the weight in the number button to the right), or by directly typing a value in the *W* number button of the *Transform* panel.

Adding or Extruding

Reference

Mode: Edit Mode
Menu: *Surface* → *Extrude*
Hotkey: E, Ctrl-LMB

Unlike meshes or curves, you cannot generally directly add new control points to a surface (with Ctrl-LMB clicks), as you can only extend a surface by adding a whole U- or V-row at once. The only exception is when working on a NURBS surface curve, i.e. a surface with only one control point on each U- or V-row. In this special case, all works exactly as with *curves*.

Most of the time, only extrusion is available. As usual, once the tool is activated the extrusion happens immediately and you are placed into *Grab mode*, ready to drag the new extruded surface to its destination.

There are two things very important to understand:

- Surfaces are *2D* objects. So you cannot extrude anything *inside* a surface (e.g. "inner" row); it would not make any sense!
- The control "grid" *must* remain "squarish", which means that you can only extrude a whole row, not parts of rows here and there...

To summarize, the *Extrude* tool will only work, when one and only one whole border row is selected, otherwise nothing happens.

As for curves, you cannot create a new surface in your object out of nowhere, by just `Ctrl-LMB` - clicking with nothing selected. However, unlike for curves, there is no "cut" option allowing you to separate a surface into several parts, so you only can create a new surface by copying (*Duplication*) an existing one `Shift-D`, or adding a new one with the *Add* menu.

Examples

Images Fig. *Selecting control-point.* to Fig. *Extruding.* show a typical extrusion along the side of a surface.

In Fig. *Selecting control-point.* and *Shift-R*, a border row of control points were highlighted by selecting a single control point, and then using the handy row select tool `Shift-R` to select the rest of the control points.

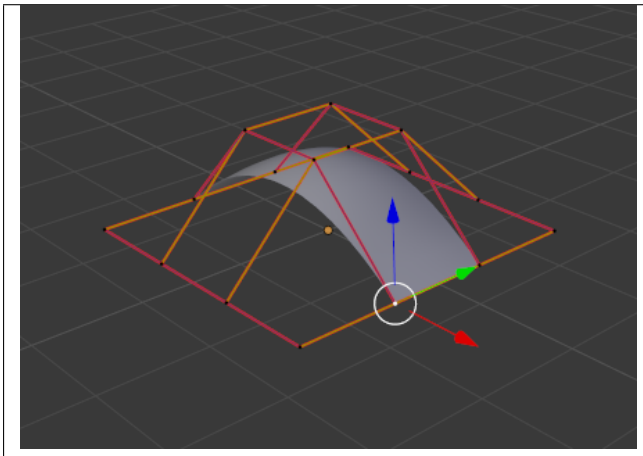


Fig. 2.707: Selecting control-point.

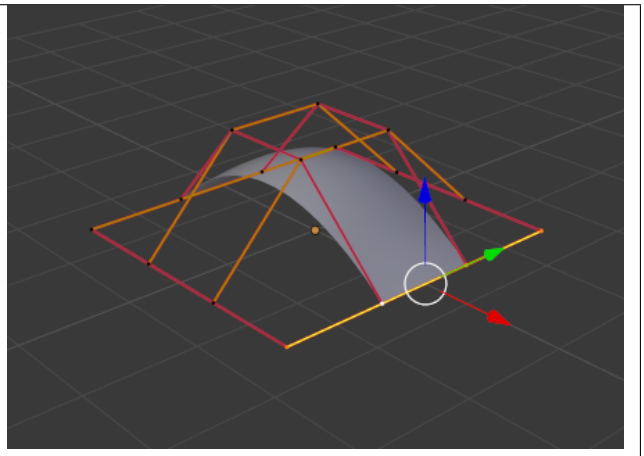


Fig. 2.708: Shift-R

The edge is then extruded using `E` as shown in Fig. *Extruding.*. Notice how the mesh has bunched up next to the highlighted edge. That is because the *new* extruded surface section is bunched up there as well.

By moving the new section away from the area, the surface begins to "unbunch".

You can continue this process of extruding or adding new surface sections until you have reached the final shape for your model.

Opening or Closing a Surface

Reference

Mode: Edit Mode
Menu: *Surface* → *Toggle Cyclic*
Hotkey: `Alt-C`

As in *curves*, surfaces can be closed (cyclic) or open. However, as surfaces are 2D, you can control this property independently along the U and V axes.

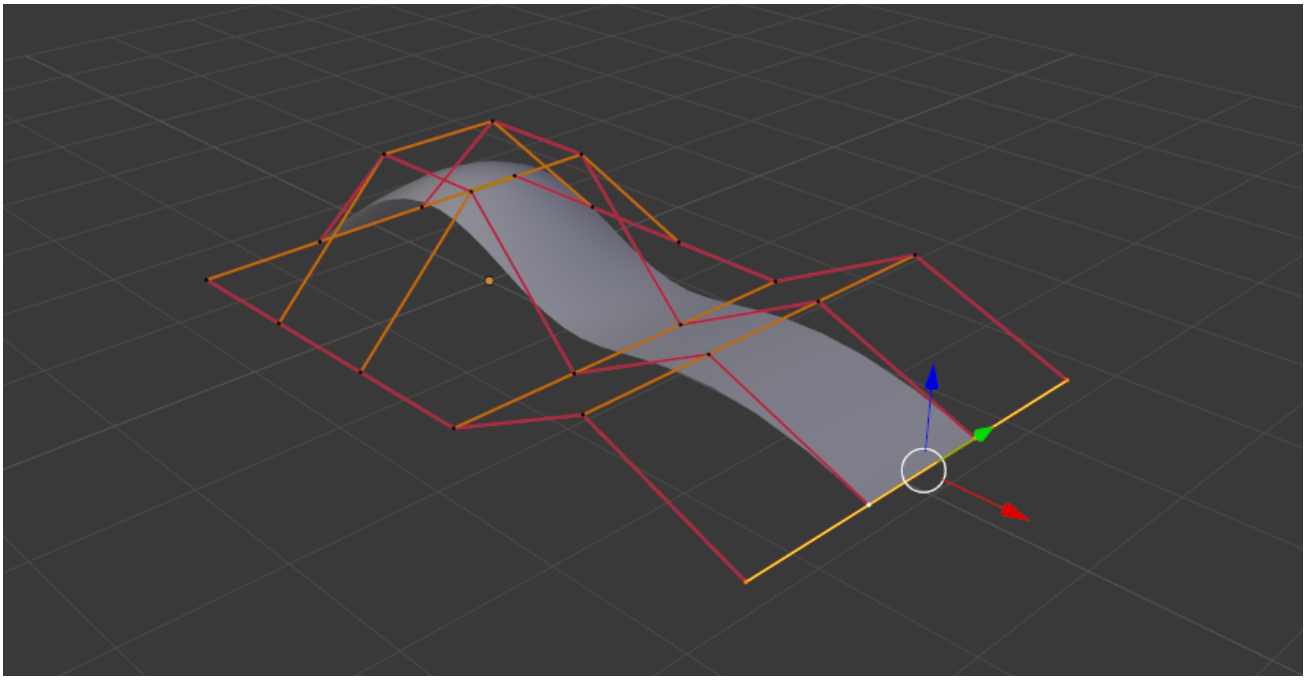


Fig. 2.709: Extruding.

To toggle the cyclic property of a surface along one axis, use `Alt-C` and choose either *cyclic U* or *cyclic V* from the pop-up menu. The corresponding surface's outer edges will join together to form a "closed" surface.

Note: Inner and Outer

Surfaces have an "inner" and "outer" face, the first being black whereas the latter is correctly shaded. (There does not seem to be any "double sided" shading option for surfaces...). When you close a surface in one or two directions, you might get an entirely black object! In this case, just *Switch Direction* of your surface...

Duplication

Reference

Mode: Edit Mode
Menu: *Curve* → *Duplicate*
Hotkey: `Shift-D`

Well, as with meshes and curves, this command just duplicates the selection. As usual, the copy is selected and placed in *Grab* mode, so you can move it to another place.

However, with surfaces there are some selections that cannot be duplicated, in which case they will just be placed in *Grab* mode... In fact, only selections forming a *single* valid sub-grid are copyable; let us see this in practice:

- You can copy a single control point. From it, you will be able to "extrude" a "surface curve" along the U axis, and then extrude this unique U-row along the V axis to create a real new surface.
- You can copy a single continuous part of a row (or a whole row, of course). This will give you a new *U-row*, even if you selected (part of) a V-row!

- You can copy a single whole sub-grid.

Note: Trying to duplicate several valid "sub-grids" (even being single points) at once will not work; you will have to do it one after the other...

Deleting Elements

Reference

Mode: Edit Mode
Menu: *Curve* → *Delete...*
Hotkey: `X`, `Delete`

The *Erase* pop-up menu of surfaces offers you two options:

Selected This will delete the selected rows, *without* breaking the surface (i.e. the adjacent rows will be directly linked, joined, once the intermediary ones are deleted). The selection must abide by the following rules:

- Whole rows, and only whole rows must be selected.
- Only rows along the same axis must be selected (i.e. you cannot delete both U- and V-rows at the same time).

Also remember that NURBS order cannot be higher than its number of control points in a given axis, so it might decrease when you delete some control points... Of course, when only one row remains, the surface becomes a "surface curve"; when only one point remains, there is no more visible surface; and when all points are deleted, the surface itself is deleted.

All As with meshes or curves, this deletes everything in the object!

Example

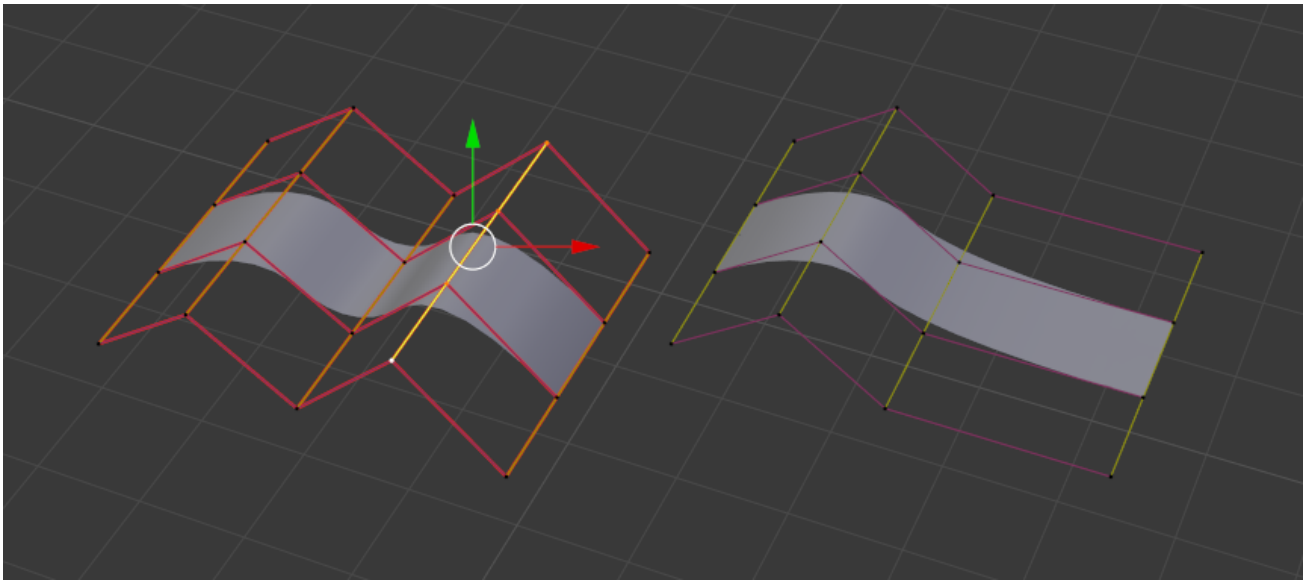


Fig. 2.710: Before and after.

In Fig. *Before and after (left)* a row of control points has been selected by initially selecting the one control point and using Shift-R to select the remaining control points. Then, using the *Delete Menu* X, the *selected* row of control points is erased, resulting in Fig. *Before and after (right)*.

Joining or Merging Surfaces

Reference

Mode: Edit Mode
Menu: *Surface* → *Make Segment*
Hotkey: F

Just like *curves*, merging two surfaces requires that a single edge, a border row of control points, from two separate surfaces are selected. This means that the surfaces must be part of the same object. For example, you cannot join two surfaces while in *Object Mode* - but you can of course, as with any objects of the same type, join two or more *Surface* objects into one object Ctrl-J - they just will not be "linked" or merged in a single one... Yes, it's a bit confusing!

This command is equivalent to creating edges or F aces for meshes (hence its shortcut), and so it only works in *Edit Mode*. The selection must contains only border rows of the same resolution (with the same number of control points), else Blender will try to do its best to guess what to merge with what, or the merge will fail (either silently, or stating that Resolution does not match if rows with different number of points are selected, or that there is Too few selections to merge if you only selected points in one surface...). To select control points of different surfaces, in the same object, you must use either border select or circle select. Holding down Ctrl while LMB will not work.

So to avoid problems, you should always only select border rows with the same number of points... Note that you can join a border U-row of one surface with a border V-row of another one, Blender will automatically "invert" the axis of one surface for them to match correctly.

NURBS surface curves are often used to create objects like hulls, as they define cross sections all along the object, and you just have to "skin" them as described above to get a nice, smooth and harmonious shape.

Examples

Fig. Joining ready is an example of two NURBS surface curves, **not** NURBS curves, in *Edit Mode*, ready to be joined. Fig. Joining complete is the result of joining the two curves.

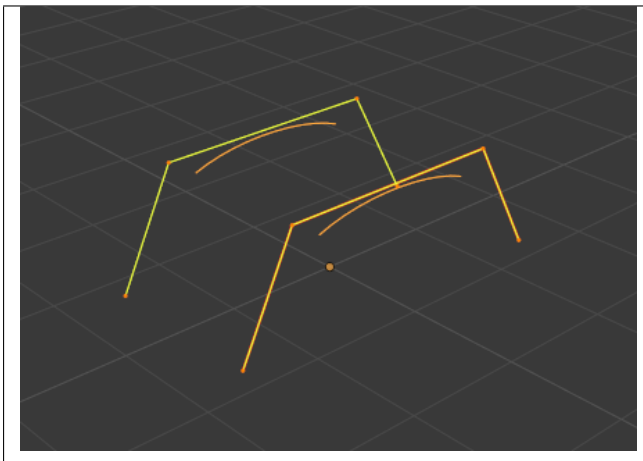


Fig. 2.711: Joining ready.

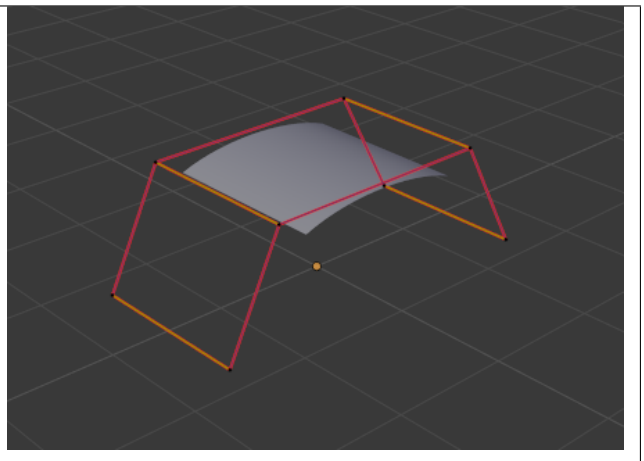


Fig. 2.712: Joining complete.

Subdivision

Reference

Mode: Edit Mode
Panel: Curve Tools
Menu: *Surface tools* → *Modeling* → *Subdivide*, *Specials* → *Subdivide*

Surface subdivision is most simple: using either the *Subdivide* entry in the *Specials* menu `W`, or the *Subdivide* button of the *Curve Tools1* panel, you will subdivide once all *completely* selected grids by subdividing each "quad" into four smaller ones.

If you apply it to a 1D surface (a "surface curve"), this tool works exactly as with *curves*.

Spin

Reference

Mode: Edit Mode
Panel: Curve Tools

This tool is a bit similar to its *mesh counterpart* but with less control and options (in fact, there is none!).

It only works on selected "surfaces" made of *one U-row* (and not with one V-row), so-called "surface curves", by "extruding" this "cross section" in a square pattern, automatically adjusting the weights of control points to get a perfect circular extrusion (this also implies closing the surface along the V axis), following exactly the same principle as for the *NURBS Tube* or *NURBS Donut* primitives.

Switch Direction

Reference

Mode: Edit Mode
Menu: *Surface* → *Segments* → *Switch Direction*, *Specials* → *Switch Direction*

This command will "reverse" the direction of any curve with at least one selected element (i.e. the start point will become the end one, and *vice versa*). Mainly useful when using a curve as path, or the bevel and taper options...

Other Specials Options

Reference

Mode: Edit Mode
Menu: Specials
Hotkey: W

The *Specials* menu contains exactly the same additional options as for curves, except for *Set Radius* and *Smooth Radius*.

Conversion

As there are only NURBS surfaces, there is no "internal" conversion here.

However, there is an "external" conversion available, from surface to mesh, that only works in *Object Mode*. It transforms a *Surface* object into a *Mesh* one, using the surface resolutions in both directions to create faces, edges and vertices.

Misc Editing

You have some of the same options as with meshes, or in *Object Mode*. You can *separate* a given surface P, make other selected objects *children* of one or three control points Ctrl-P, or *add hooks* to control some points with other objects.

The *Mirror* tool is also available, behaving exactly as with *mesh vertices*.

2.4.5 Metaball

Introduction

Reference

Mode: Object or Edit Modes
Menu: *Add* → *Meta*
Hotkey: Shift-A

Meta objects are *implicit surfaces*, meaning that they are *not explicitly* defined by vertices (as meshes are) or control points (as surfaces are): they exist *procedurally*. Meta objects are literally mathematical formulas that are calculated on-the-fly by Blender.

A very distinct visual characteristic of metas is that they are fluid *mercurial*, or *clay-like* forms that have a "rounded" shape. Furthermore, when two meta objects get close to one another, they begin to interact with one another. They "blend" or "merge", as water droplets do, especially in zero-g (which, by the way, makes them very handy for modeling streams of water when you do not want to do a fluid simulation). If they subsequently move away from one another, they restore their original shape.

Each of these is defined by its own underlying mathematical structure (*Structure*), and you can at any time switch between them using the *Active Element* panel.

Typically *Meta* objects are used for special effects or as a basis for modeling. For example, you could use a collection of metas to form the initial shape of your model and then convert it to a mesh for further modeling. Meta objects are also very efficient for ray-tracing.

Note: *Meta* objects have a slightly different behavior in *Object Mode*.

Visualization

In Object Mode, the calculated mesh is shown, along with a black "selection ring" (becoming pink when selected).

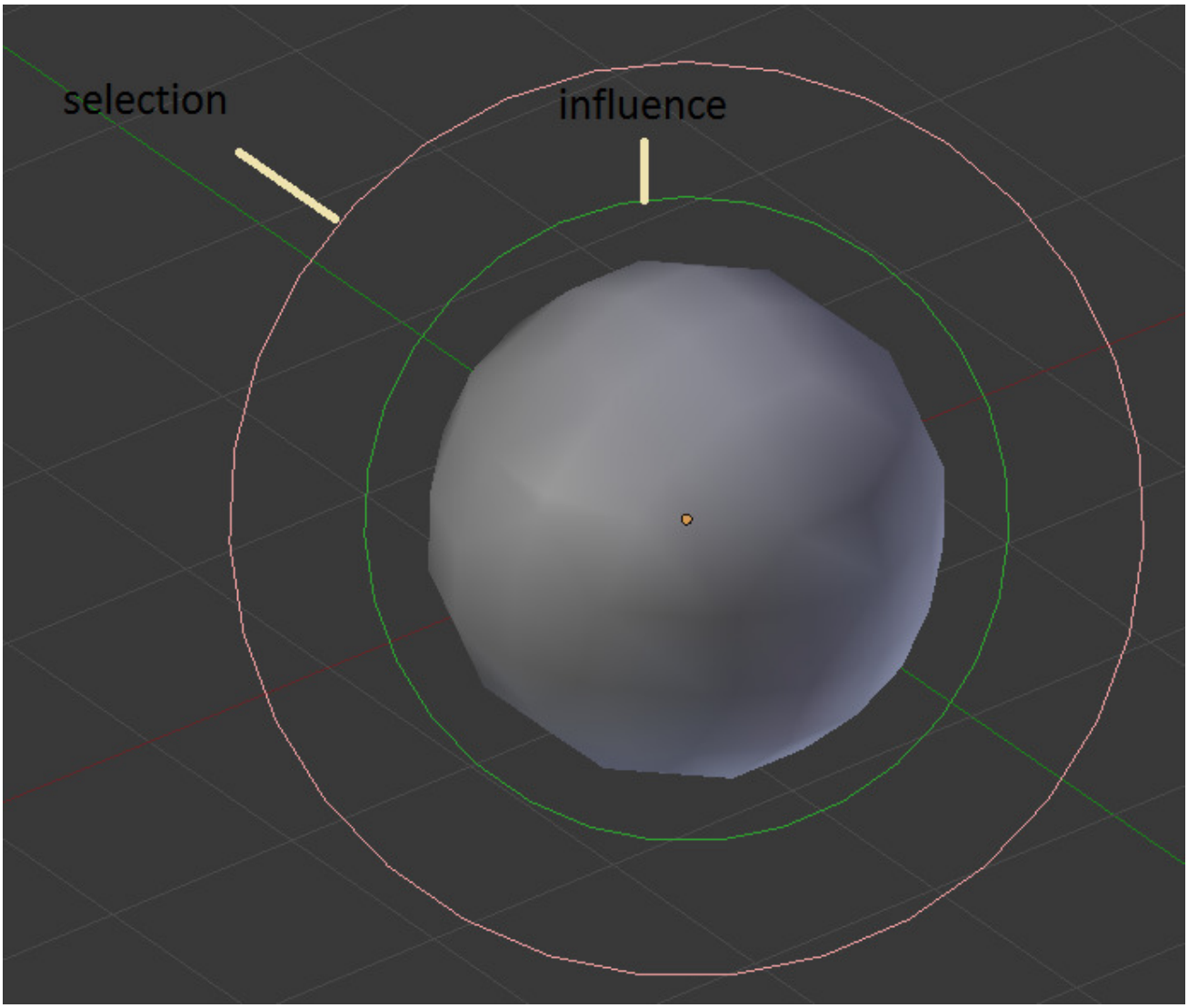


Fig. 2.713: Meta Ball example.

In *Edit Mode* (Fig. *Meta Ball example.*), a meta is drawn as a mesh (either shaded or as black wireframe, but without any vertex of course), with two colored circles: a red one for selection (pink when selected), and a green one for a direct control of the meta's stiffness (light green when active). Note that except for the *Scale* S transformation, having the green circle highlighted is equivalent to having the red one.

Primitives

There are five predefined meta "primitives" (or configurations) available in the *Add* → *Meta* sub-menu:

Meta Ball Adds a meta with a point underlying structure.

Meta Tube Adds a meta with a line segment underlying structure.

Meta Plane Adds a meta with a planar underlying structure.

Meta Ellipsoid Adds a meta with an ellipsoidal underlying structure.

Meta Cube Adds a meta with a volumetric cubic underlying structure.

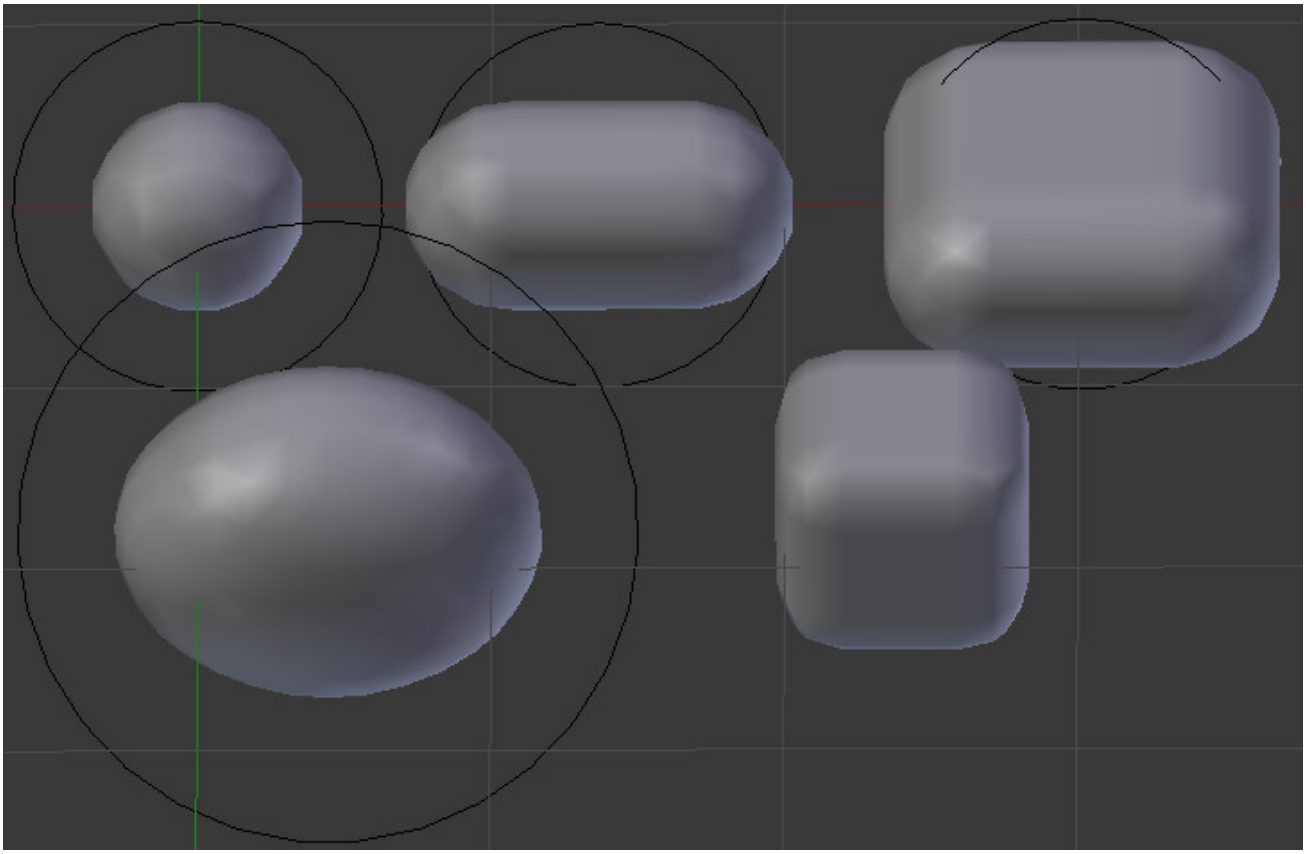


Fig. 2.714: The five Meta primitives.

Structure

Technical Details

A more formal definition of a meta object can be given as a *directing structure* which can be seen as the source of a static field. The field can be either positive or negative and hence the field generated by neighboring directing structures can attract or repel.

The implicit surface is defined as the surface where the 3D field generated by all the directing structures assume a given value. For example a meta ball, whose directing structure is a point, generates an isotropic (i.e. identical in all directions) field around it and the surfaces at constant field value are spheres centered at the directing point.

Meta objects are nothing more than mathematical formula that perform logical operations on one another (AND, OR), and that can be added and subtracted from each other. This method is also called *Constructive Solid Geometry* (CSG). Because of its mathematical nature, CSG uses little memory, but requires lots of processing power to compute.

Underlying Structure

Reference

Mode: Edit Mode
Panel: *Properties region* → *Transform panel* → *Type*, *Metaball tab* → *Active Element panel* → *Type*

Blender has five types of metas, each determined by its underlying (or directing) structure.

In *Edit Mode*, you can change this structure, either using the relevant buttons in the *Metaball tab* → *Active Element panel*, or the selector in the *Transform* panel in the Properties region. Depending on the structure, you might have additional parameters, located in both *Transform* panel and *Active Element* panel.

Ball (point, zero-dimensional structure) This is the simplest meta, without any additional setting. As it is just a point, it generates an isotropic field, yielding a spherical surface (this is why it is called *Meta Ball* or *Ball* in Blender).

Tube (straight line, uni-dimensional structure) This is a meta which surface is generated by the field produced by a straight line of a given length. This gives a cylindrical surface, with rounded closed ends. It has one additional parameter:

dx The length of the line (and hence, of the tube).

Plane (rectangular plane, bi-dimensional structure) This is a meta which surface is generated by the field produced by a rectangular plane. This gives a parallel-epipedal surface, with a fixed thickness, and rounded borders. It has two additional parameters:

dx, dy The length, width of the rectangle.

Note that by default, the plane is a square.

Ellipsoid (ellipsoidal volume, tri-dimensional structure) This is a meta which surface is generated by the field produced by an ellipsoidal volume. This gives an ellipsoidal surface. It has three additional parameters:

dx, dy, dz The length, width, height of the ellipsoid (defaults set to 1.0).

Note that by default, the volume is a sphere, producing a spherical meta, as the *Ball* option...

Cube (parallel-epipedal volume, tri-dimensional structure) This is a meta which surface is generated by the field produced by a parallel-epipedal volume. This gives a parallel-epipedal surface, with rounded edges. As you might have guessed, it has three additional parameters:

dx, dy, dz The length, width, height of the parallelepiped (defaults set to 1.0).

Note that by default, the volume is a cube.

Properties

All Meta objects in a scene interact with each other. The settings in the *Metaball* section apply to all meta objects. In *Edit Mode*, the *Active Element* panel appears for editing individual meta elements.

Fig. 2.715: global meta properties.

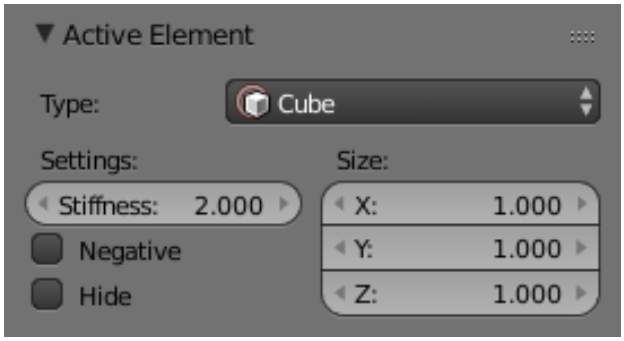


Fig. 2.716: individual meta properties.

Resolution

The *Resolution* controls the resolution of the resultant mesh as generated by the Meta object.

View The 3D View resolution of the generated mesh. The range is from (0.05 to 1.0) (finest to coarsest).

Render The rendered resolution of the generated mesh. The range is from (0.05 to 1.0) (finest to coarsest).

One way to see the underlying mathematical structure is to lower the *Resolution*, increase the *Threshold* and set the *Stiffness* (see below) a fraction above the *Threshold*. Fig. *Underlying.* is a *Meta cube* with the above mentioned configuration applied as follows: *Resolution* of 0.410, *Threshold* of 5.0 and *Stiffness* a fraction above at 5.01.

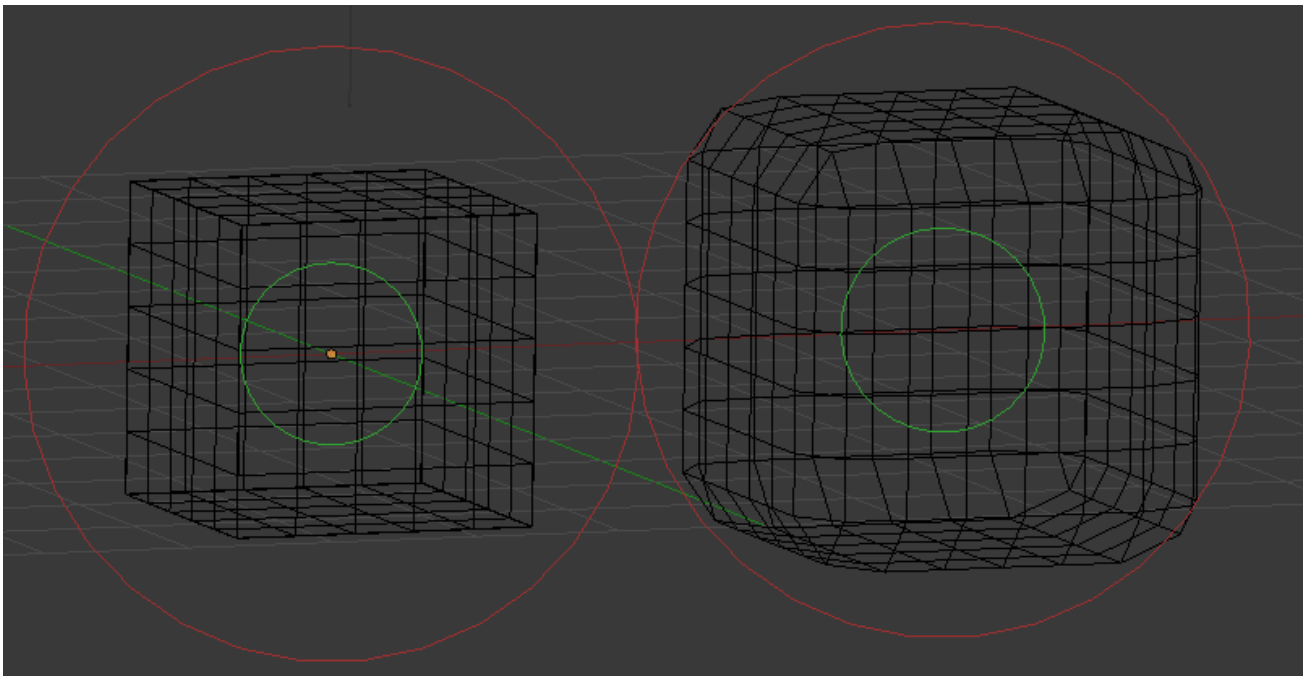


Fig. 2.717: Underlying.
Left: Underlying structure, Right: the shape.

You can clearly see the underlying cubic structure that gives the meta cube its shape.

Threshold (Influence)

Reference

Mode: Object or Edit Modes
Panel: Metaball

Threshold defines how much a meta's surface "influences" other metas. It controls the *field level* at which the surface is computed. The setting is global to a group of *Meta* objects. As the threshold increases, the influence that each meta has on each other increases.

There are two types of influence: *positive* or *negative*. The type can be toggled on the *Active Element* panel while in *Edit Mode*, using the *Negative* button. You could think of *positive* as attraction and *negative* as repulsion of meshes. A negative meta will push away or repel the meshes of positive *Meta* objects.

A *positive* influence is defined as an attraction, meaning the meshes will stretch towards each other as the *rings of influence* intersect. Fig. *Positive.* shows two meta balls' *rings of influence* intersecting with a *positive* influence.

Notice how the meshes have pulled towards one another. The area circled in white shows the green *influence* rings intersecting.

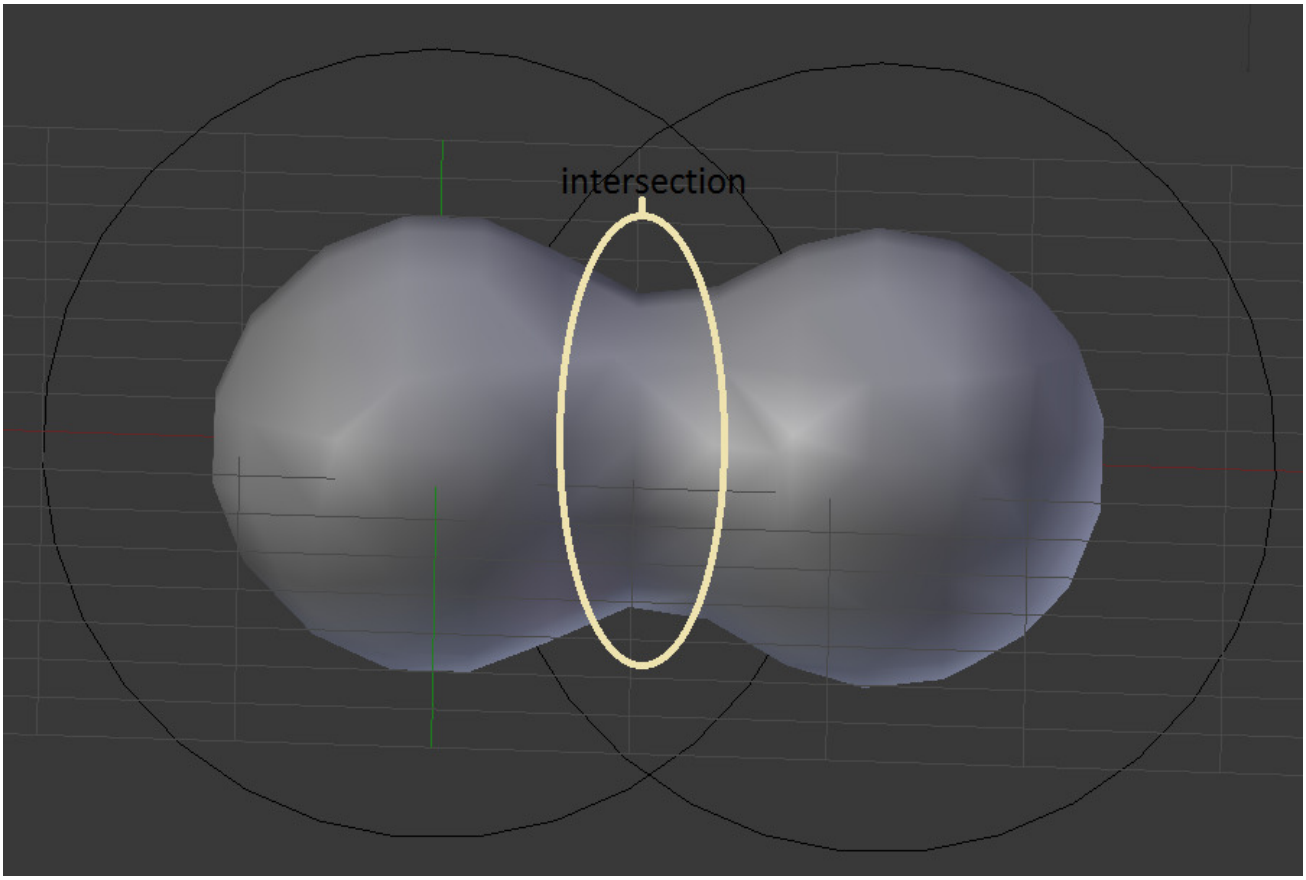


Fig. 2.718: Positive.

Update

While transforming metas (grab/move, scale, etc.), you have four "modes" of visualization, located in the *Update* buttons group of the *Metaball* panel:

Always fully draw the meta during transformations.

Half Res During transformations, draw the meta at half its *Wiresize* resolution.

Fast Do not show meta mesh during transformations.

Never Never show meta mesh (not a very recommended option, as the meta is only visible at render time!).

This should help you if you experience difficulties (metas are quite compute-intensive...), but with modern computers, this should not happen, unless you use many metas, or very high resolutions...

Editing Metas

When in *Edit Mode*, the *Active Element* panel appears. These settings apply only to the selected meta element.

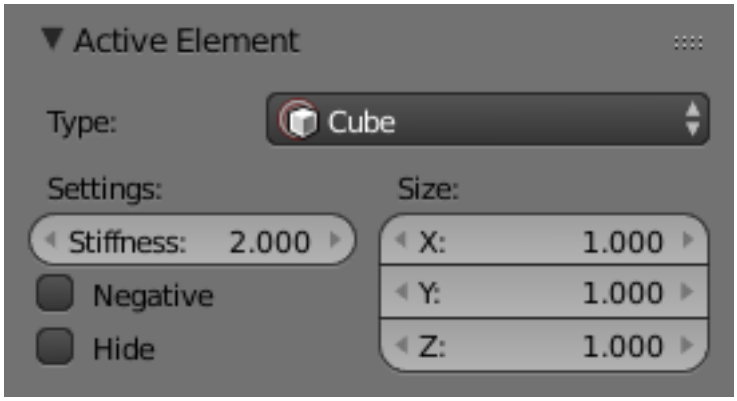


Fig. 2.719: Active element panel.

Meta Shape

The *Type* menu lets you change the shape of the meta object, as explained above.

Stiffness

Together with *Threshold*, *Stiffness* controls the influencing range. While the threshold is common to all metas in the same object (or even the same *Object Families*), the stiffness is specific to each meta.

Scaling the inner green circle changes the *Stiffness* value. Stiffness defines how much the meta object is filled. This essentially defines how sensitive a meta is to being affected by other metas. With a low stiffness, the meta will begin to deform from further away. A higher value means the meta needs to be close to another one to begin merging.

When a *Meta* object comes within "range" of another meta, the two will begin to interact with each other. They do not necessarily need to intersect, and depending on the *Threshold* and *Stiffness* settings, they most likely will not need to. *Stiffness* is materialized by the *green ring*

The range is from (0.0 to 10.0). But to be visible, the *Stiffness* must be slightly larger than the *Threshold* value. You can also visually adjust the *Stiffness* ring by using the RMB to select it and activate *Scale* mode with S.

In Fig. *Stiffness.*, the meta ball labeled "A", has a smaller *Stiffness* value than the one labeled "B". As you can see, the radius (green ring) is different for each of them.

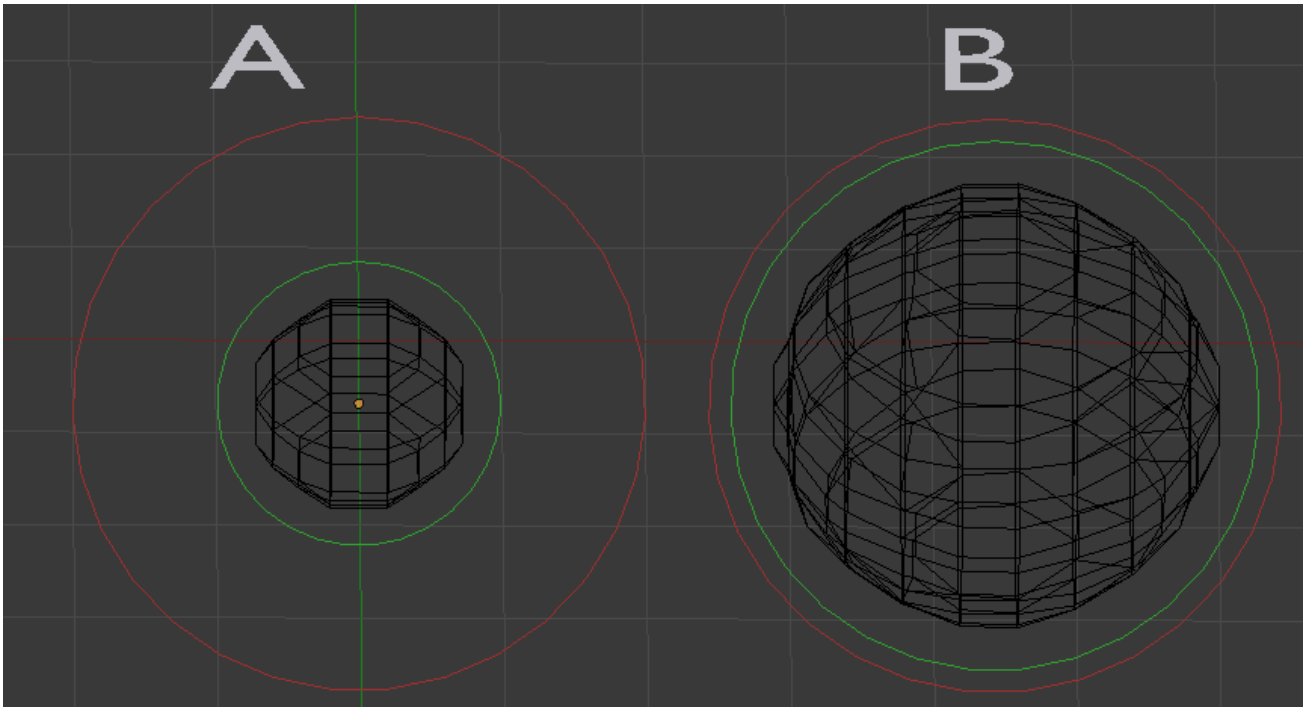


Fig. 2.720: Stiffness.

Negative Influence

The opposite effect of a *positive* influence would be a *negative* influence: the objects repel each other. Fig. *Negative.* shows a meta ball and a meta plane where the first is negative and the second, positive. Notice how the negative meta is not visible: only the surrounding circles appear. This is how Blender indicates that the object is negative.

Moving the sphere to the plane causes the plane's mesh to "cave in" or collapse inward. If you move the plane away from the sphere, the plane's mesh will restore itself.

To make a meta *negative*, just select the meta in edit mode, and check *negative* in the *active element* panel.

Hiding Elements

As in *Object Mode*, you can hide the selected meta(s), and then reveal what was hidden. This is very handy for cleaning your views up a bit... Note that the two red and green rings always remain visible in *Edit Mode*, as well as the select circle (in *Object Mode*...).

To hide the current selection, use `H`, the *Hide* toggle button in the *Metaball tools*, or the *Metaball* → *Show/Hide* → *Hide Selected* menu option.

To hide everything but the current selection, press `Shift-H` or use *Metaball* → *Show/Hide* → *Hide Deselected*.

To reveal what was hidden, use `Alt-H`, or the relevant option in the same *Metaball* → *Show/Hide* menu. You can also un-toggle the *Hide* button in the (*Metaball tools* panel).

Deleting Elements

There is no *Erase* menu for metas, just a confirmation pop-up asking you if you want to delete the selected metas. Clear and simple!

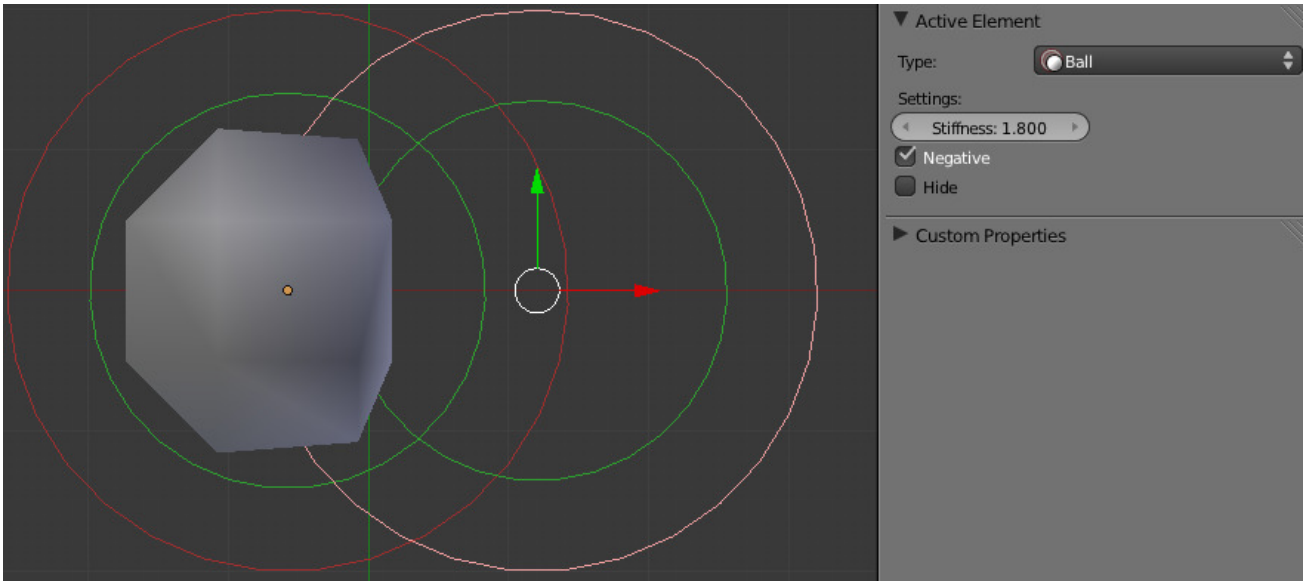


Fig. 2.721: Negative.

Conversion

You can only convert metas to meshes, but here you have the option to keep the original *Meta* object (i.e. create a new *Mesh* one, instead of a "real" conversion...). Note that the resolution used for the new mesh is the *Wiresize* one, not the *Rendersize* one.

To convert the meta, press `Alt-C` in *Object Mode*, and select *Mesh/Text*,

Object Families

Meta objects have different behavior in *Object Mode* than other object types. They can be "regrouped" into so-called "families".

A "family" is a way to regroup several meta objects, producing something very similar to having several metas inside the same object.

A family is defined by the left part of an object's name (the one before the dot). Remember, an object's name is the one in the *OB* field, in most panels, **not** the *MB* field, which is the meta data-block's name... For example, the *family* part of "MetaPlane.001" is `MetaPlane`. Each meta object in the same "family" is associated with one another as discussed below.

Families of metas are controlled by a *base* meta object which is identified by an object name **without** a right number part. For example, if we have five metas called "MetaThing", "MetaThing.001", "MetaThing.002", "MetaThing.003" and "MetaThing.004", the *base* meta object would be "MetaThing".

The *base* meta object determines the basis, the resolution, the threshold, *and* the transformations. It also has the material and texture area. The *base* meta is effectively the parent of (or perhaps a better word to use is "the owner of") the other metas in the group (i.e. it is as if the other metas were "included" or joined into the base one).

Hint: When working with multiple scenes, take care naming your meta objects so the *base* is always in the same scene as other metas.

Failing to do this will give confusing behavior (invisible meta objects).

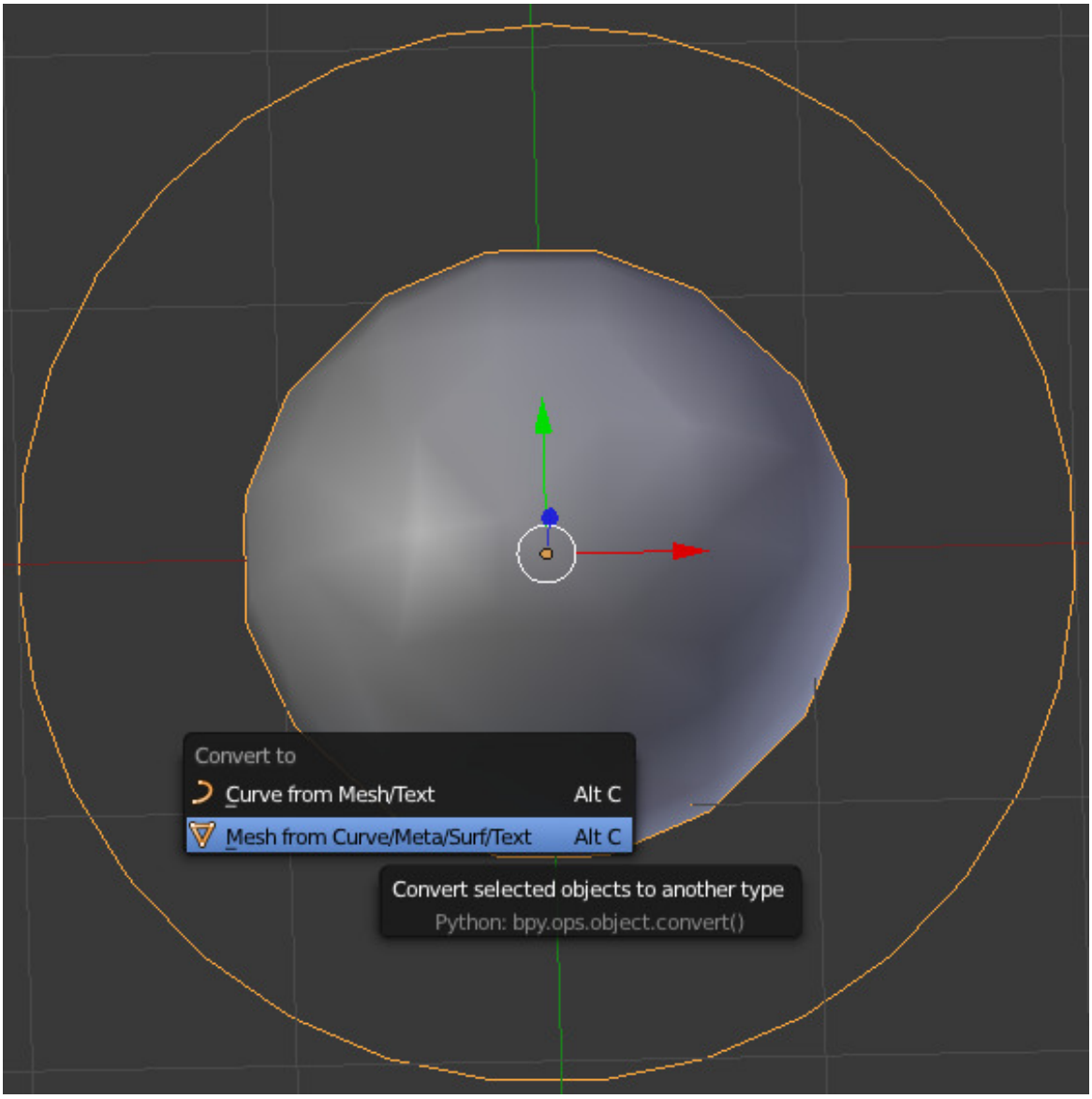


Fig. 2.722: Convert Menu.

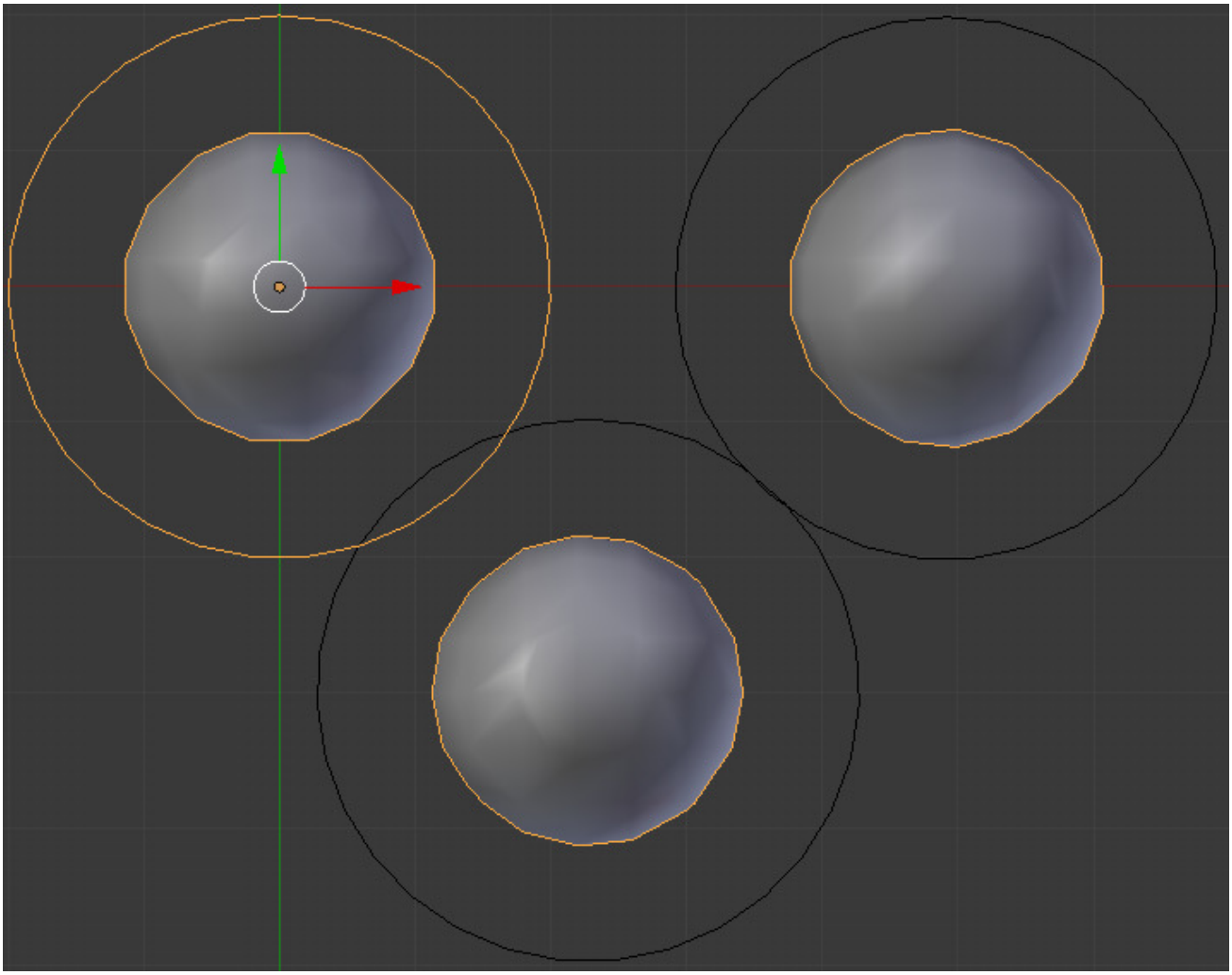


Fig. 2.723: Meta ball base.

Examples

Fig. *Meta ball base.* shows the *base* meta labeled "B". The other two *Meta* objects are *children*. Children's selection rings are always black, while the group's mesh is orange. Because the metas are grouped, they form a unified mesh which can always be selected by selecting the mesh of any meta in the group. For example, in the example Fig. *Meta ball base.*, only the lower sphere (the parent) has been selected, and you see that both the parent's mesh *and* all of the children's meshes are now highlighted.

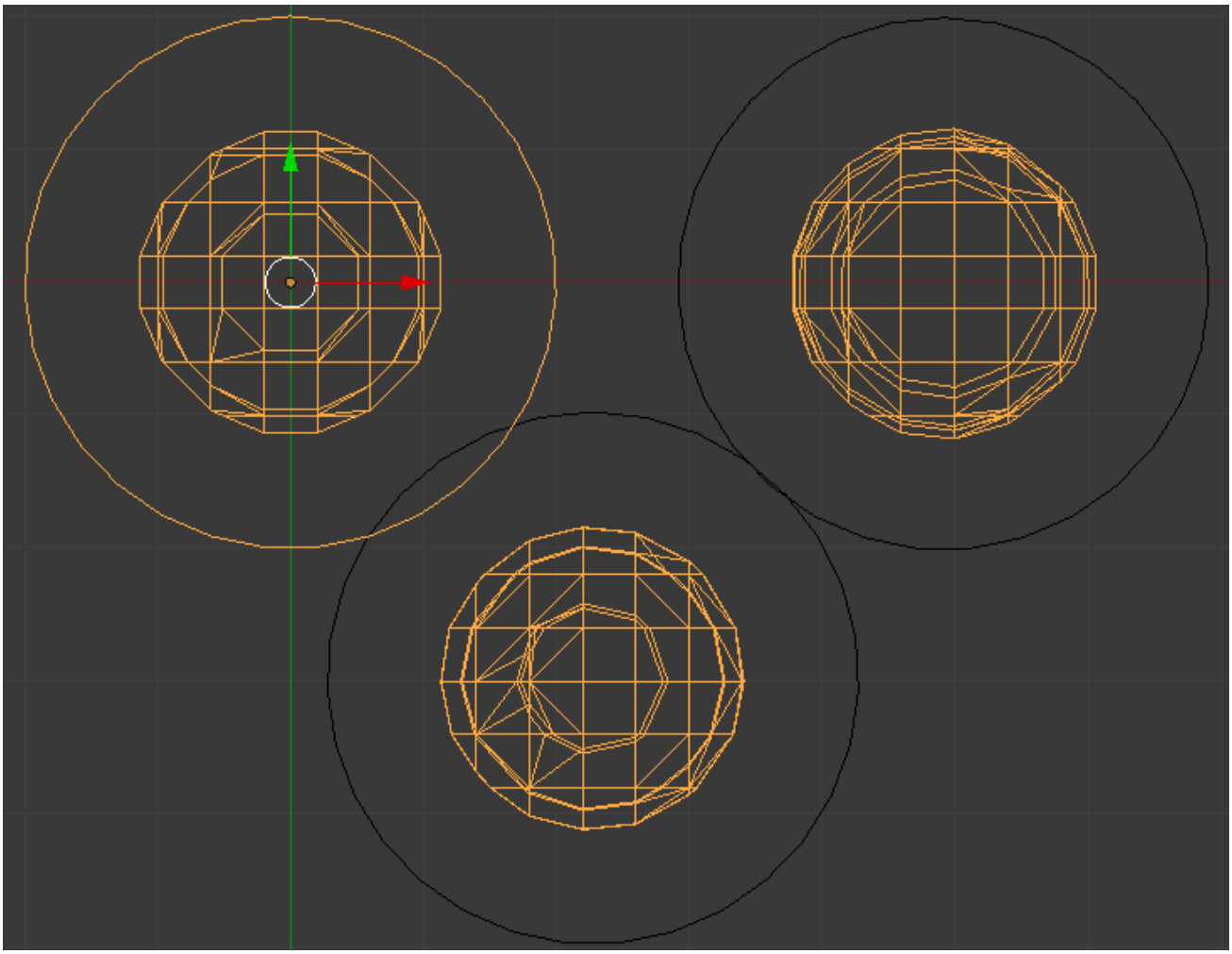


Fig. 2.724: Scaling the "base".

The *base* meta object controls the *polygonalization* (mesh structure) for the group, and as such, also controls the polygonalization for the children (non-base) metas. If we transform the *base* meta, the children's polygonalization changes. However, if we transform the children, the polygonalization remains unchanged.

Hint: This discussion of "polygonization" does *not* mean that the various meshes do not deform towards or away from each other (meta objects always influence one another in the usual way, whether or not they are members of the same family). Rather, it means that the underlying mesh structure changes only when the *base* object transforms. For example, if you scale the *base*, the children's mesh structure changes. In Fig. *Scaling the "base".*, the *base* has been scaled down, which has the effect of scaling the mesh structure of each of the children. As you can see, the children's mesh resolution has increased, while the *base* decreased. The children did *not* change size!

2.4.6 Text

Introduction

Reference

Mode: Edit Mode
Panel: Curve and Surface, Font and Char
Menu: *Add* → *Text*

Fig. 2.725: Text Examples.

Text objects are exactly what they sound like: they contain some text. They share the same object type as curves and surfaces, as modern fonts (OpenType, TrueType, etc.) are vectorial, made of curves (generally Béziers).

Blender uses a "Font System" to manage mapping "letter codes –> objects representing them in 3D views". This implies that not only does the font system have its own *built-in* font, but it can use external fonts too, including *PostScript Type 1*, *OpenType* and *TrueType* fonts. And last but not least, it can use any objects existing in the current blend-file as letters...

Texts in Bender allow you to create/render 2D or 3D text, shaded as you want, with various advanced layout options (like justifying and frames), as we will see below. By default, letters are just flat filled surfaces, exactly like any closed 2D curve. But you can of course extrude them... And texts can follow other curves.

Of course, once you are happy with the shape of your text, you can convert it (with `Alt-C`, in *Object Mode*), either to a curve, or directly to a mesh, allowing you to use all the powerful features of these types of objects on it...

Fig. *Text Examples.* shows some examples of various fonts in action, including the "blue" font that has been applied to a curve path.

Note: A maximum of 50000 characters is allowed per text object; however, be forewarned that the more characters a single text object has, the slower the object will respond interactively.

As you can see when you switch between *Object Mode* and *Edit Mode*, the *Font* panel remains the same. This means that its settings can be applied equally in both modes ... and this implies that you cannot apply them to just a part of the mesh. So font, size, and so on, are common to all letters in a *Text* object. There is just one exception: the *Bold* or *Italic* buttons control properties specific to each letter (this is a way to use up to four different fonts in a text).

For optimum resource usage, only characters that are being used consume memory (rather than the entire character set).

Shape

Reference

Mode: Object or Edit Mode
Panel: Curve and Surface

As you can see in the *Curve and Surface* panel, texts have most of the same options as curves.

Resolution

Preview, Render resolution. See *curve resolution*.

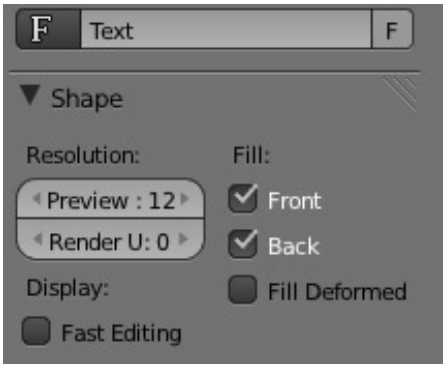


Fig. 2.726: the shape settings.

Fast Editing disables curve filling while in edit mode.

Fill

The fill options control how the text curves are filled in when text is *Extruded* or *Beveled* in the *Geometry* Panel.

Front Fills in the front side of the surface.

Back Fills in the back side of the surface.

Fill Deformed Fills the curves after applying shape keys and modifiers.

Textures

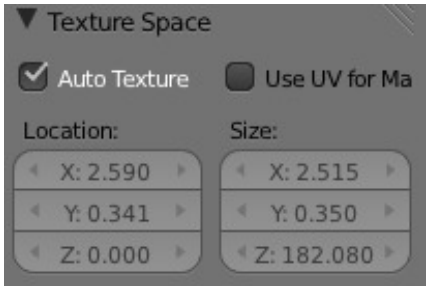


Fig. 2.727: Texture Settings.

Use UV for Mapping Use UV values as generated texture coordinates.

Auto Texture Space Adjusts the active object's texture space automatically when transforming object.

Geometry

Text objects have all the *curves extrusion features*.

Properties

Shape

Resolution

Preview The surface resolution in the U direction to use in the viewport.

Render he surface resolution in the U direction, set to zero to use the the *Preview* resolution.

Fill Determines the way a Curve is displayed when it is beveled.

Fill Deformed Fills the curve after applying all modification that might deform the curve (i.e. shape keys and modifiers).

Fast Editing Does not fill polygons while editing text.

Texture Space

TODO.

Geometry

Modification

Offset Alters the space between letters.

Extrude Will extrude the text along both the positive and negative local Z axes.

Bevel

Depth Changes the size of the bevel.

Resolution Alters the smoothness of the bevel.

Taper Object Used to select a curve object that can be used to cause the characters to get thinner towards one end. You can also alter the proportions of the Taper throughout the tapered object by moving/scaling/rotating the Control Points of the *Taper Object*. The *Taper Object* can only be a curve. Editing the Handles and Control Points of the *Taper Object* will cause the original object to change shape.

Bevel Object Used to select a curve object that can be used to give custom bevel results.

Font

Reference

Mode: Edit Mode
Panel: Font

The *Font* panel has several options for changing the look of characters.

Loading and Changing Fonts

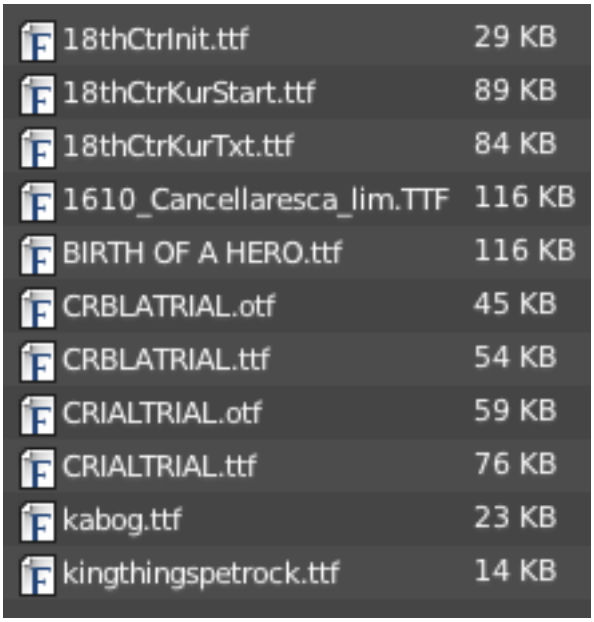


Fig. 2.728: Loading a Type 1 font file.

Blender comes with a *built-in* font by default and is displayed in each of the four font style data-block menus . The *built-in* font is always present and shows in this list as "Bfont". The data-block menu contains a list displaying the currently loaded fonts. Select one for each font style.

To load a different *Font*, click one of the *Load* buttons in the *Font* panel and navigate to a *valid* font. The *File Browser* will give all valid fonts a capital F icon, as seen in *Loading a Type 1 font file.*

Note: Location of fonts on Unix

Fonts are typically located under `/usr/lib/fonts`, or some variant like `/usr/lib/X11/fonts`, but not always. They may be in other locations as well, such as `/usr/share/local` or `/usr/local/share`, and possibly related sub-trees.

If you select a font that Blender cannot understand, you will get the error `Not a valid font.`

Remember the same font will be applied to all chars with same style in a text, but that a separate font is required for each style. For example, you will need to load an *Italics* font in order to make characters or words italic. Once the font is loaded you can apply that font "Style" to the selected characters or the whole object. In all, you would need to load a minimum of four different types of fonts to represent each style (Normal, Italics, Bold, Bold-Italics).

It is important to understand, that Blender does not care what font you load for "normal", "bold", etc., styles. This is how you can have up to four different fonts in use in the same text, but you have to choose between different styles of a same font, or different fonts. Blender has a number of typographic controls for changing the style and layout of text, found in the *Font* panel.

Size and Shear

Size Controls the size of the whole text (no way to control each char size independently). Note however, that chars with different fonts (different styles, see below) might have different visible sizes.

Shear Controls the inclination of the whole text. Different to as it may seems, this is not similar to italics style.

Fig. 2.729: shear: 'blender' has a shear value of 1, '2.59' a shear value of 0.

Objects as Fonts

You can also "create" your own "font" inside Blender! This is quite a complex process, so let us detail it:

1. First, you must create your chars. Each char, of any type, is an object (mesh, curve, meta...). They all must have a name following the schema: *common prefix* followed by the *char name* (e.g. "ft.a", "ft.b", etc.).
2. Then, for the *Text* object, you must enable the *Dupli Verts* button (*Object* → *Animation Settings* panel).
3. In the *Font* tap, fill the *Object Font* field with the *common prefix* of your "font" objects.

Now, each time a char in your text matches the *suffix part* of a "font" object's name, this object is duplicated on this char. The original chars remain visible. The objects are duplicated so that their center is positioned at the *lower right corner* of the corresponding characters.

Text on Curve Used to select a curve for the text object to follow.

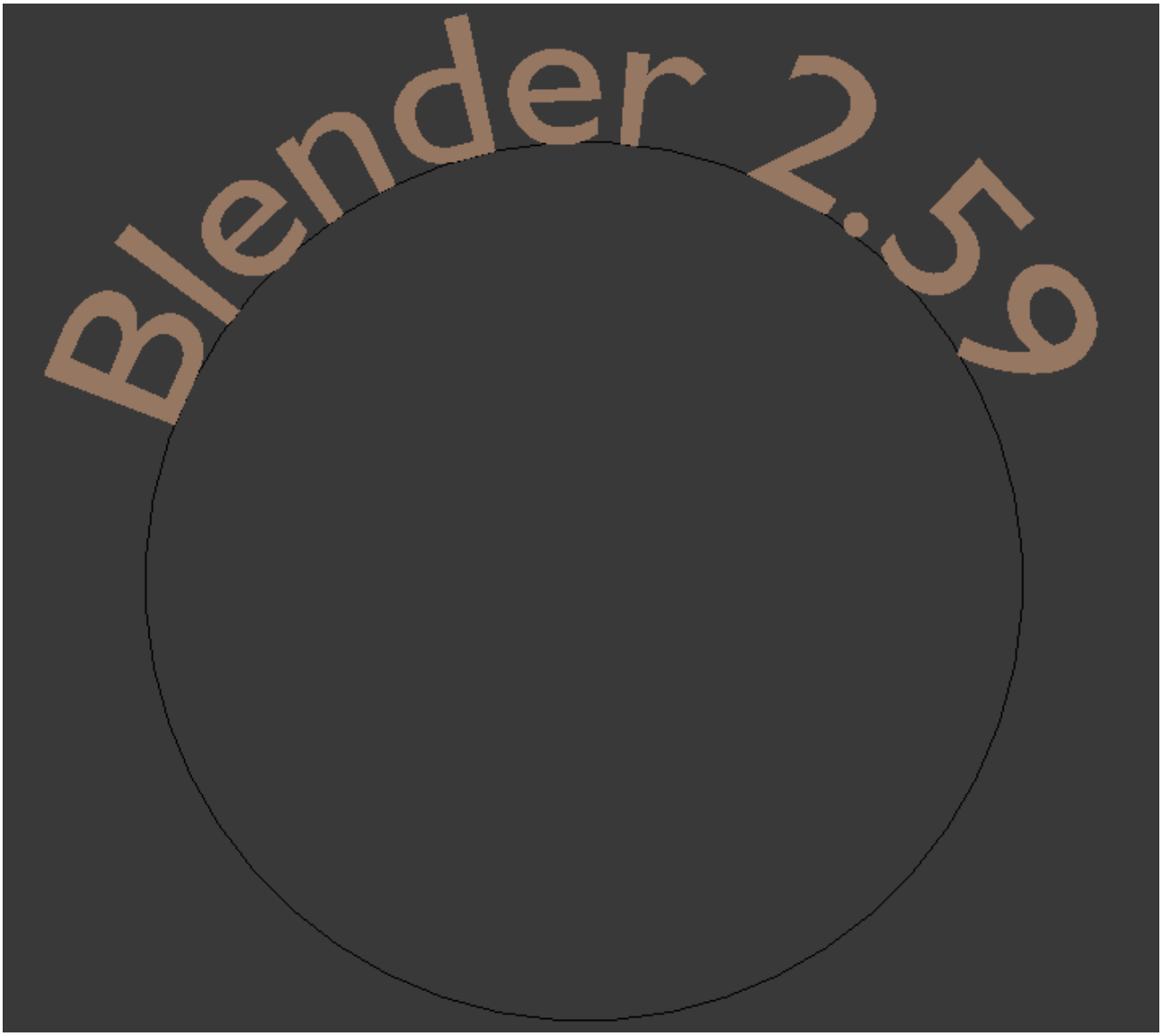


Fig. 2.730: Text on curve.

Tip: You can also use the *Curve Modifier* which offers more control.

Underline Toggled with the *Underline* button before typing. Text can also be set to Underlined by selecting it then using the *Underline* button in the Tool Shelf.

Position This allows you to shift vertically the position of the underline.

Thickness This controls the thickness of the underline.

Character

Fig. 2.731: Bold text.

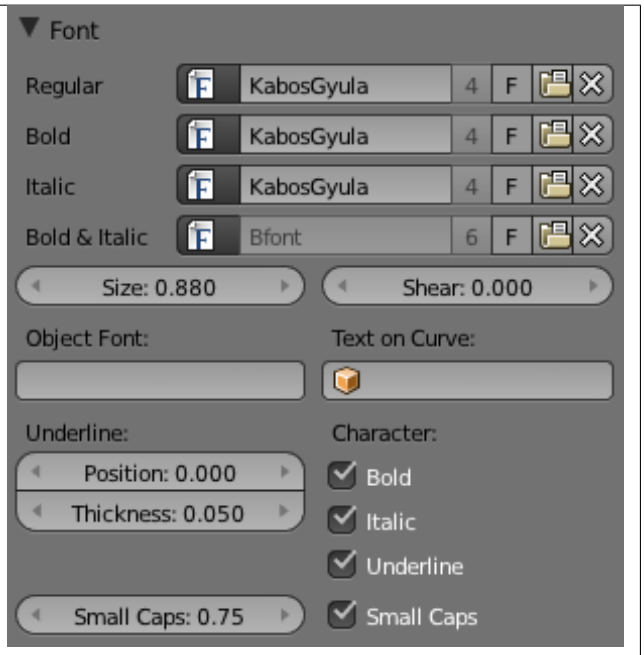


Fig. 2.732: Character options to, for example, type bold text.

Bold Toggled with the *Bold* button before typing. Text can also be set to Bold by selecting it then using the *Bold* button in the Tool Shelf.

Italics Toggled with the *Italic* button before typing. Text can also be set to Italic by selecting it then using the *Italic* button in the Tool Shelf.

Underline Enables underlining, as controlled by the Underline settings above.

Small Caps Type small capital text.

Blender's *Bold* and *Italic* buttons do not work the same way as other applications, as they also serve as placeholders for you to load up other fonts manually, which get applied when you define the corresponding style; see *Font*.

To apply the Bold/Italics/Underline attribute to a set of characters, you either turn on *Bold* / *Italics* / *Underline* prior to typing characters, or highlight (select) first and then toggle Bold/Italics/Underline.

Setting Case

You can change the text case by selecting it then clicking the *To Upper* or *To Lower* in the tool shelf.

Enable the *Small Caps* option to type characters as small caps.

The size of the *Small Caps* can be changed with the *Small Caps Scale* setting. Note that the *Small Caps Scale* is applied the same to all *Small Caps* formatted characters.

Paragraph

The *Paragraph* Panel has settings for the alignment and spacing of text.

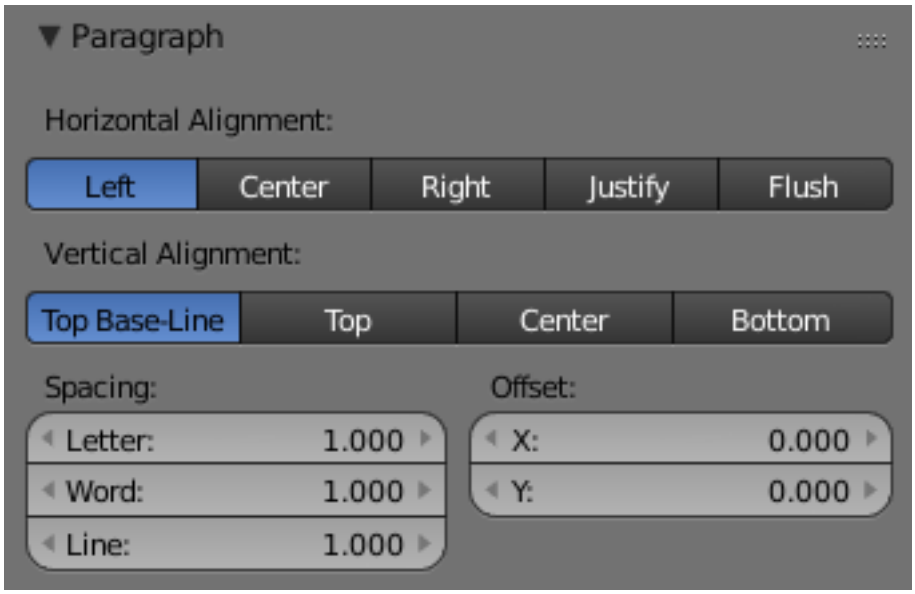


Fig. 2.733: The paragraph tab.

Horizontal Alignment

Left Aligns text to left of frames when using them, else uses the center point of the *Text* object as the starting point of the text (which grows to the right).

Center Centers text in the frames when using them, else uses the center point of the *Text* object as the mid-point of the text (which grows equally to the left and right).

Right Aligns text to right of frames when using them, else uses the center point of the *Text* object as the ending point of the text (which grows to the left).

Justify Only flushes a line when it is terminated by a wordwrap (**not** by `Return`), it uses *whitespace* instead of *character spacing* (kerning) to fill lines.

Flush Always flushes the line, even when it is still being entered; it uses character spacing (kerning) to fill lines.

Both *Justify* and *Flush* only work within frames.

Vertical Alignment

Top Base-Line Aligns the text base-line to top of frames when using them, else uses the center point of the *Text* object as the starting point of the text (which grows to the bottom).

Top Aligns top of text to the center point of the *Text* object (which grows to the bottom). It behaves as *Top Base-Line* when using frames. *Top* only works without frames.

Center Centers text in the frames when using them, else uses the center point of the *Text* object as the mid-point of the text (which grows equally to the top and bottom).

Bottom Aligns text to bottom of frames when using them, else uses the center point of the *Text* object as the ending point of the text (which grows to the top).

Spacing

Character A factor by which space between each character is scaled in width

Word A factor by which whitespace between words is scaled in width. You can also control it by pressing `Alt-Left` or `Alt-Right` to decrease/increase spacing by steps of 0.1 .

Line A factor by which the vertical space between lines is scaled.

Offset

X offset and Y offset Well, these settings control the X and Y offset of the text, regarding its "normal" positioning. Note that with frames (see *Text Boxes*), it applies to all frames' content...

Editing

Editing text is quite different from other object types in Blender, and happens mainly in two areas. First, the 3D View, of course, where you type your text, and have a few shortcuts, e.g. for applying styles (see *Character*) - note however, that most Blender hotkeys you know in *Edit Mode* do not exist for texts! The second place is the Properties Editor, especially the *Font* tab.

Fig. 2.734: Text in Edit Mode.

The menu of the 3D View header has nearly no use, and there is no *Specials* menu... You have no transform nor mirror tools, and so on. However, you can apply to texts the same modifiers as for curves.

Editing *Text* is similar to using a standard text editor but is not as full-featured and has some differences:

Exit *Edit Mode* `Tab` does not insert a tab character in the text, but rather enters and exits *Edit Mode*, as with other object types.

Copy To copy text to the buffer, use `Ctrl-C` or the *Copy* button in the tool shelf.

Cut and Copy To cut and copy text to the buffer, use `Ctrl-X` or the *Cut* button in the tool shelf.

Paste To paste text from the buffer, use `Ctrl-V` or the *Paste* button in the tool shelf.

Delete all text To completely erase or delete all text, use `Ctrl-Backspace`.

Home/End `Home` and `End` move the cursor to the beginning and end of a line respectively.

Next/Previous word To move the cursor on a word's boundary, use `Ctrl-Left` or `Ctrl-Right`.

The text buffer is in sync with the desktop clipboard. But if it is used within Blender the text formatting will be copied as well. For other ways of inserting a text, see *Inserting Text* below.

Inserting Text

You can insert text in two ways: from the internal text buffer (as described above), or from a text file.

To load text from a text file, use the *Text* → *Paste File* tool. This will bring up a *File Browser* for navigating to a valid UTF-8 file. As usual, be careful that the file does not have too many characters, as interactive response will slow down.

Special Characters

Reference

Mode: Edit Mode
Menu: *Text* → *Special Characters*

If you need special characters (such as accented chars, which are not on your keyboard) you can produce many of them using a combination of two other characters. To do so, type the main char, press `Alt-Backspace`, and then press the desired "modifier" to produce the special character. Some examples are given below:

ã	`A`, `Alt-Backspace`, `~`
á	`A`, `Alt-Backspace`, `'`
à	`A`, `Alt-Backspace`, `\`
å	`A`, `Alt-Backspace`, `O`
ë	`E`, `Alt-Backspace`, `"`
ø	`O`, `Alt-Backspace`, `/`

Converting Text Objects

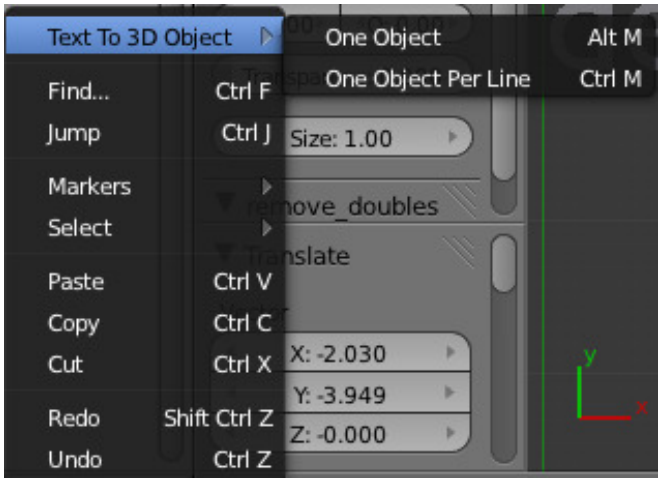
Converting to Text Object

Using an existing text-block, you can convert it to an object from the text editor's header, select *Edit* → *Text to 3D Object*, *One Object* or *One Object per Line* depending on your needs.

It is also possible to paste from the clipboard or a file from the *Edit* menu, while editing 3D Text.

Converting to 3D Mesh

It is possible to convert a Text Object to a 3D Mesh object. This can be useful so that you may edit the vertices in *Edit Mode*, but you will lose the ability to edit the text itself. To do this, go to *Object Mode* and select your Text Object. Press `Alt-C` and select *Mesh From Curve/Meta/Surf/Text*. Now you can return to *Edit Mode* and manually edit the vertices. They are usually a bit messy, so it may be useful to use a *Limited Dissolve* deletion or *Remesh* Object *Modifier* at a low threshold to clean up your mesh.



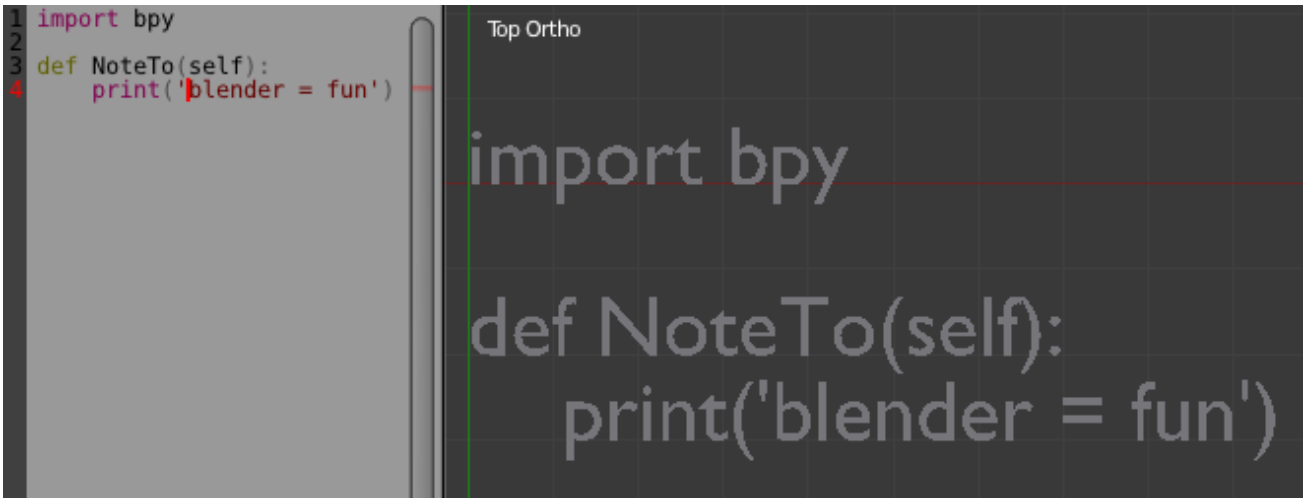


Fig. 2.735: left normal text, right the made text object.

Text Selection

Fig. 2.736: Text in Edit Mode.

In *Edit Mode*, your text has a white cursor, and as in any text editor, it determines where new chars will be inserted! You move this cursor with the arrow keys or `PageUp` / `PageDown` and `Home` / `End`.

Hold `Shift` while using the arrow keys to select a part of the text. You can use it to specify different materials, the normal/bold/italic state, and not much more...

Text Boxes

Reference

Mode: Object or Edit Modes
Panel: Font

Text "Boxes" allow you to distribute the text amongst rectangular areas within a single text object. An arbitrary number of freely positionable and re-sizable text frames are allowed per text object.

Text flows continuously from the lowest-numbered frame to the highest-numbered frame with text inside each frame word-wrapped. Text flows between frames when a lower-numbered frame cannot fit any more text. If the last frame is reached, text overflows out of it.

Text frames are very similar to the concept of *frames* from a desktop publishing application, like Scribus. You use frames to control the placement and flow of text.

Frames are controlled in the *Text Boxes* panel.

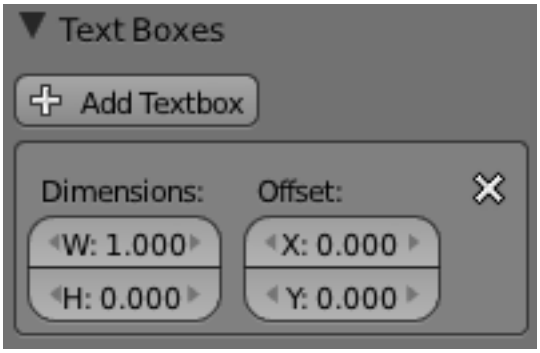


Fig. 2.737: Text frame.

Frame Size

By default the first frame for a new text object, and any additional frames, has a size of **zero** for both *Width* and *Height*, which means the frame is initially not visible.

Frames with a width of 0.0 are ignored completely during text flow (no wordwrap happens), and frames with a height of 0.0 flow forever (no flowing to the next text frame).

In order for the frame to become visible, the frame's *Width* must be greater than 0.0.

Note: Technically the height is never actually 0.0, because the font itself always contributes height.

Fig. 2.738: Frame width.

Fig. *Frame width.* is a text object with a width of 5.0. And because the frame width is greater than 0.0 it is now visible and is drawn in the active theme color as a dashed rectangle. The text has overflowed because the text has reached the end of the last frame, the default frame.

Adding/Deleting a Frame

To add a frame click the *Add Textbox* button on the *Text Boxes* panel. A new frame is inserted just after (in text flow order) the current one, with its attributes (position and size). Be sure to modify the offset for the new frame in the X and/or Y fields. Just an X modification will create a new column.

To delete the current frame, click the `Delete` button. Any text in higher frames will be re-flowed downward into lower frames.

Examples

Text Flow

Fig. 2.739: Wrapping.

With two or more frames you can organize text to a finer degree. For example, create a text object and enter "Blender is super duper". This text object has a frame; it just is not visible because its *Width* is 0.0.

Set the width to 5.0. The frame is now visible and text is wrapping according to the new width, as shown in Fig. *Wrapping.*. Notice that the text has overflowed out of the frame. This is because the text has reached the end of the last frame, which just happens to be the default/initial frame.

Fig. 2.740: Text flowing from box 1 to box 2.

When we add another frame and set its width and height, the text will flow into the new frame.

Multiple Columns

To create two columns of text just create a text object and adjust the initial frame's *Width* and *Height* to your requirements, then insert a new frame. The new frame will have the same size as the initial frame. Set the X position to something greater or less than the width of the initial frame; see Fig. *Text 5.*.

Assigning Materials

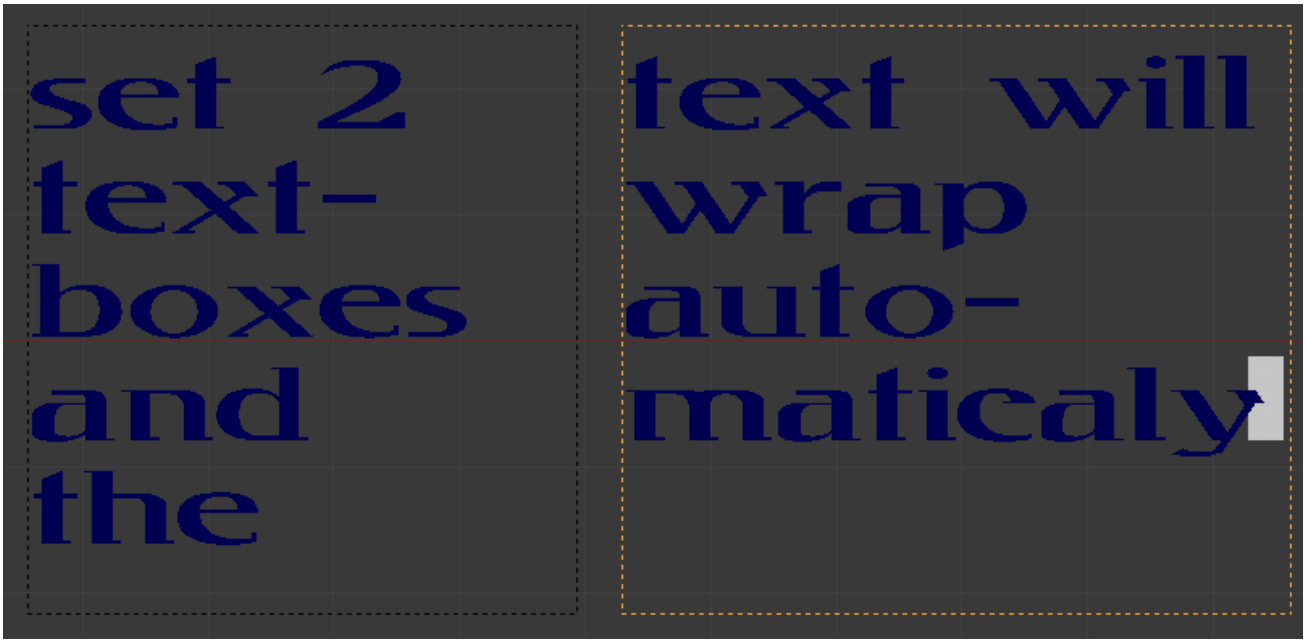


Fig. 2.741: Text 5.

Reference

Mode: Edit Mode
Panel: Link and Materials

Each character can have a different *Material index* in order to have different materials on different characters.

You can assign indices either as you type, or after by selecting blocks of text and clicking on the *Assign* button in the Materials panel.

Fig. 2.742: Red Green Blue.

For example, to create Fig. *Red Green Blue.* you would need to create three separate materials and three separate material indices. Each word would be assigned a *Material index* by selecting the characters for each word and clicking the *Assign* button. Fig. *Red Green Blue.* is still one single *Text* object.

2.4.7 Empties

The "Empty" is a single coordinate point with no additional geometry. Because an Empty has no volume and surface, it cannot be rendered. Still it can be used as a handle for many purposes.

Editing

An Empty can only be edited in *Object Mode*, which includes its transformation and parenting properties.

Properties

Display

Plain Axes Draws as six lines, initially with one pointing in each of the +X, -X, +Y, -Y, +Z, and -Z axis directions.

Arrows Draws as arrows, initially pointing in the positive X, Y, and Z axis directions, each with a label.

Single Arrow Draws as a single arrow, initially pointing in the +Z axis direction.

Circle Draws as a circle initially in the XZ plane.

Cube Draws as a cube, initially aligned to the XYZ axes.

Sphere Draws as an implied sphere defined by three circles. Initially, the circles are aligned, one each, to the X, Y, and Z axes.

Cone Draws as a cone, initially pointing in the +Y axis direction.

Image Empties can display images. This can be used to create reference images, including blueprints or character sheets to model from, instead of using background images. The image is displayed regardless of the 3D display mode. The settings are the same as in *Background Image Settings*

Note: While alpha-images can be used, there is a known limitation with object draw order, where alphas will not always draw on top of other objects when unselected.

Size Controls the size of the empties visualization. This does not change its scale, but functions as an offset.

Usage

Empties can serve as transform handles. Some examples of ways to use them include:

Parent object for a group of objects

An Empty can be parented to any number of other objects. This gives the user the ability to control a group of objects easily, and without affecting a render.

Target for constraints

An empty can also be used as a target for normal, or bone constraints. This gives the user far more control; for instance, a rig can easily be set up to enable a camera to point towards an empty using the *Track to* constraint.

Array offset

An empty can be used to offset an array modifier, meaning complex deformations can be achieved by only moving a single object.

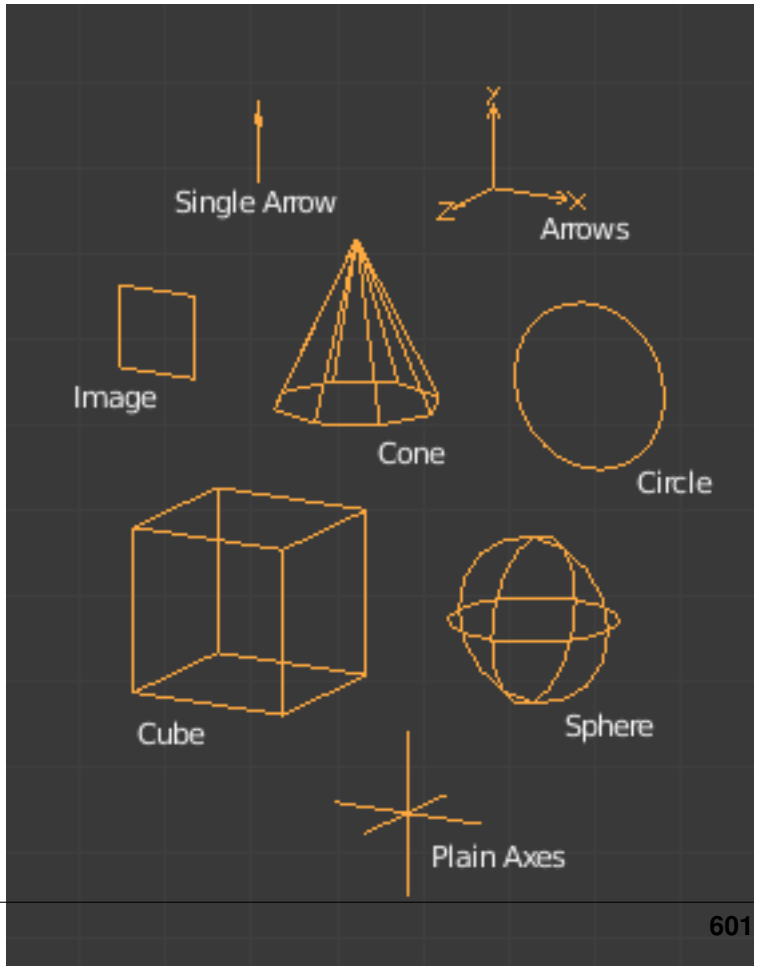


Fig. 2.743: Empty Draw Types.

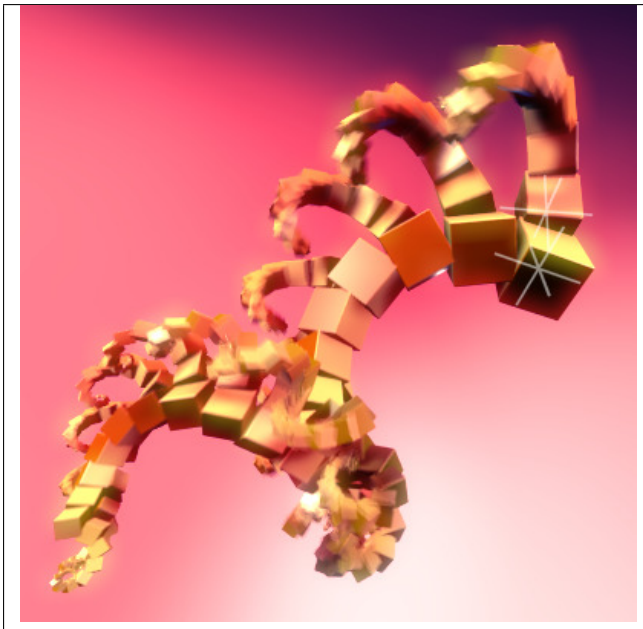


Fig. 2.744: An example of an empty being used to control an array.

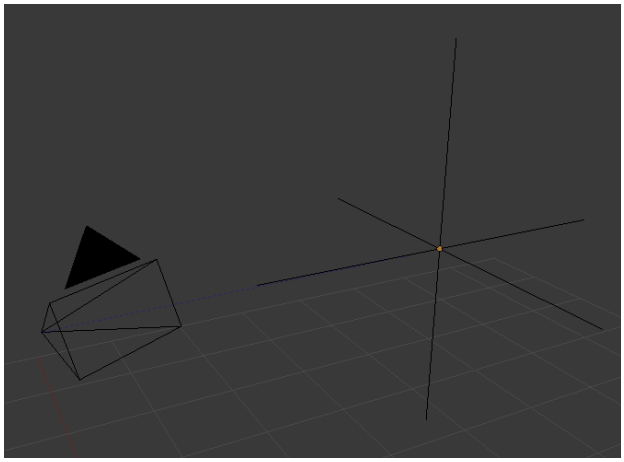


Fig. 2.745: An example of an empty being used to control the track to constraint.

Other common uses:

- Placeholders
- Rigging controls
- DOF distances
- Reference Images

2.4.8 Modifiers

Introduction

Modifiers are automatic operations that affect an object in a non-destructive way. With modifiers, you can perform many effects automatically that would otherwise be too tedious to do manually (such as subdivision surfaces) and without affecting the base geometry of your object.

They work by changing how an object is displayed and rendered, but not the geometry which you can edit directly. You can add several modifiers to a single object to form *The Modifier Stack* and *Apply* a modifier if you wish to make its changes permanent.

There are four types of modifiers:

Modify The *Modify* group of modifiers are tools similar to the *Deform Modifiers* (see below), but which do not directly affect the shape of the object; rather they affect some other data, such as vertex groups.

Fig. 2.746: Modifiers Menu.

Generate The *Generate* group of modifiers are constructive tools that either change the general appearance of or automatically add new geometry to an object.

Deform The *Deform* group of modifiers only change the shape of an object without adding new geometry, and are available for meshes, and often texts, curves, surfaces and/or lattices.

Simulate The *Simulate* group of modifiers activate simulations. In most cases, these modifiers are automatically added to the modifiers stack whenever a *Particle System* or *Physics* simulation is enabled. Their only role is to define the place in the modifier stack used as base data by the tool they represent. Generally, the attributes of these modifiers are accessible in separate panels.

Interface

Each modifier has been brought in from a different part of Blender, so each has its own unique settings and special considerations. However, each modifier's interface has the same basic components, see Fig. *Panel Layout (Subdivision Surface as an example).*.

At the top is the *panel header*. The icons each

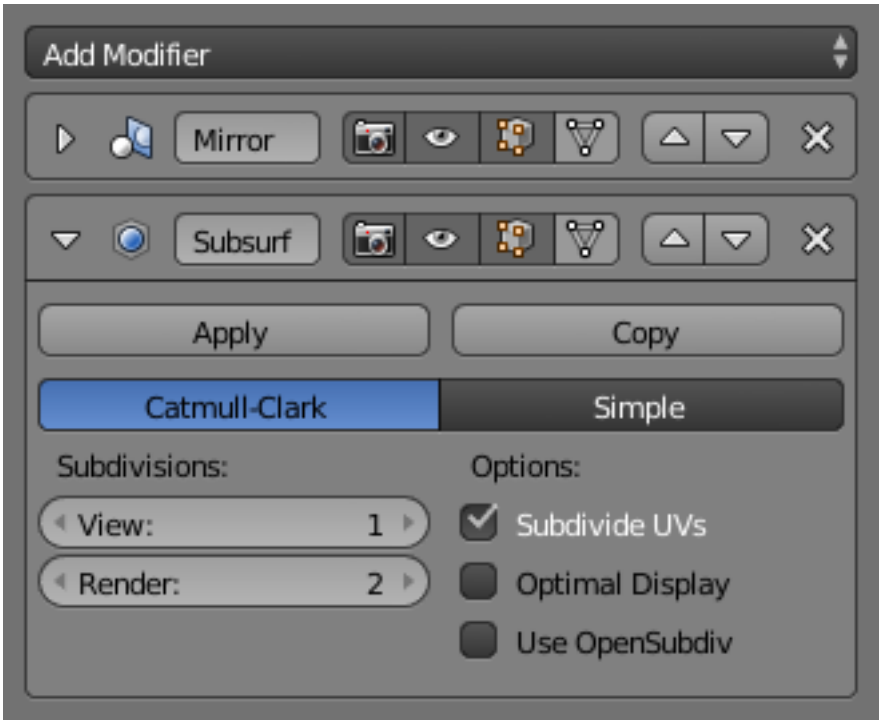


Fig. 2.747: Panel Layout (Subdivision Surface as an example).

represent different settings for the modifier (left to right):

Arrow Collapse modifier to show only the header and not its options.

Icon A quick visual reference of the modifier's type.

Name Every modifier has a unique name per object. Two modifiers on one object must have unique names, but two modifiers on different objects can have the same name. The default name is based off the modifier type.

Camera Toggles visibility of the modifier effect in the render.

Eye Toggles visibility of the modifier effect in the 3D View.

Box Displays the modified geometry in edit mode, as well as the original geometry which you can edit.

Triangle When enabled, the final modified geometry will be shown in Edit Mode and can be edited directly.

Up arrow Moves modifier up in the stack.

Down arrow Moves modifier down in the stack.

Cross Deletes the modifier.

Note: The *Box* and *Triangle* icons may not be available depending on the type of modifier.

Below the header are two buttons:

Apply Makes the modifier "real" - converts the object's geometry to match the applied modifier, and deletes the modifier.

Copy Creates a duplicate of the modifier at the bottom of the stack.

> **Warning:** Applying a modifier that is not first in the stack will ignore the stack order and could produce undesired results.

Below this header, all of the options unique to each modifier will be displayed.

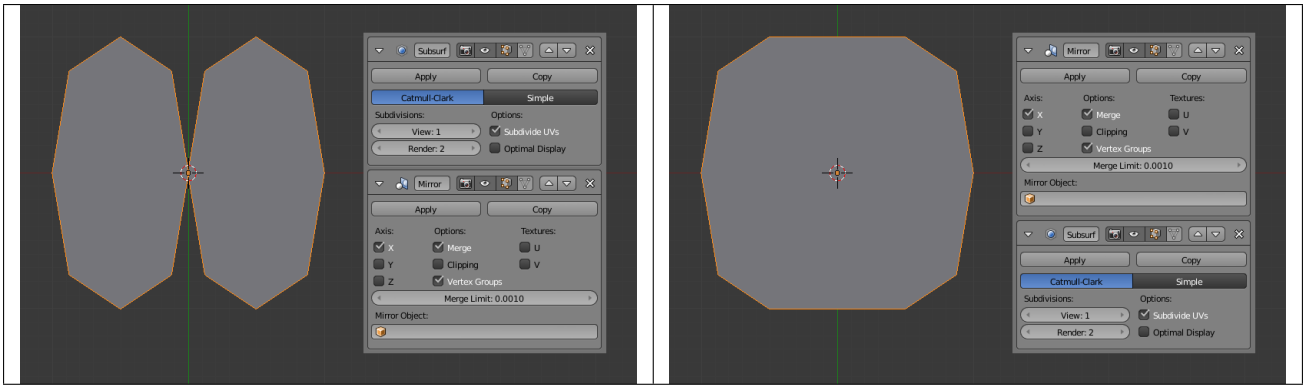
The Modifier Stack

Modifiers are a series of non-destructive operations which can be applied on top of an object's geometry. They can be applied in just about any order the users chooses.

This kind of functionality is often referred to as a "modifier stack" and is also found in several other 3D applications.

In a modifier stack the order in which modifiers are applied has an effect on the result. Fortunately modifiers can be rearranged easily by clicking the convenient up and down arrow icons. For example, the image below shows *Subdivision Surface* and *Mirror* modifiers that have switched places.

Table 2.24: The Subdivision surface modifier is the last item in the stack and the result is a single merged surface.



Modifiers are calculated from top to bottom in the stack. In this example, the desired result (on right) is achieved by first mirroring the object, and then calculating the subdivision surface.

Example

Download example file.

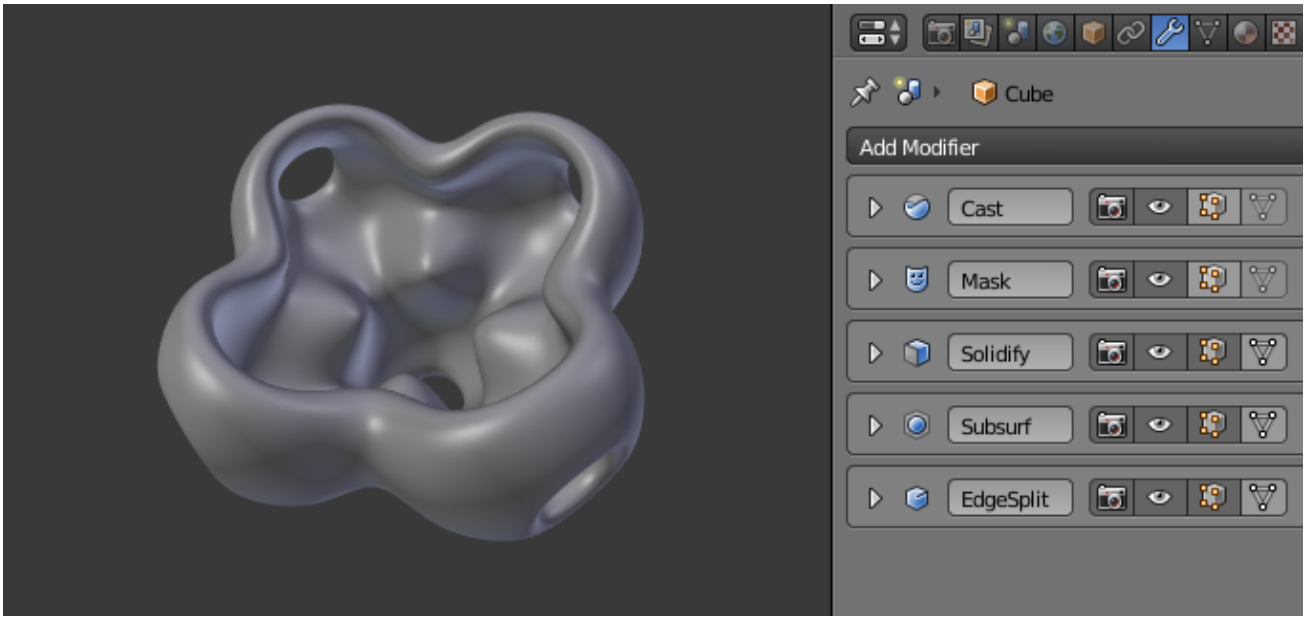


Fig. 2.750: In this example a simple subdivided cube has been transformed into a rather complex object using a stack of modifiers.

Modify

Data Transfer Modifier

The *Data Transfer* modifier transfers several types of data from one mesh to another. Data types include vertex groups, UV layers, vertex colors, custom normals...

Transfer works by generating a mapping between source mesh's items (vertices, edges, etc.) and destination ones, either on an one-to-one basis, or mapping several source items to a single destination one by interpolated mapping.

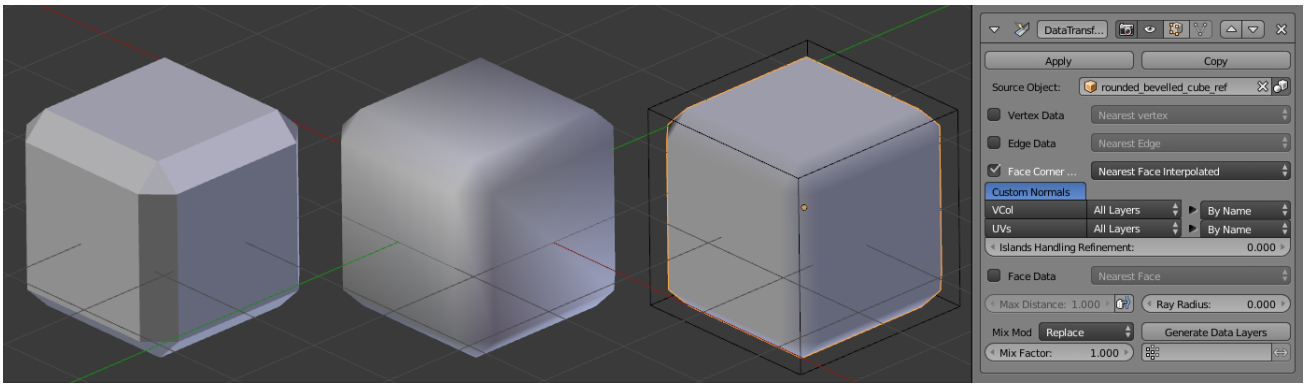


Fig. 2.751: From left to right, a flat-shaded beveled cube, a smooth-shaded beveled cube, and an autosmooth-shaded beveled cube copying its normals from the reference, flat-shaded cube shown as wire here, to achieve the 'fake round corners' effect.

Options

Source Object Mesh object to copy data from.

If the button to the right of the field is unset, source and destination geometries are considered in global space when generating the mapping, otherwise they are evaluated in local space (i.e. as if both object's centers were at the same place).

Max Distance When the icon "finger" button to the right is enabled, this is the maximum distance between source and destination to get a successful mapping. If a destination item cannot find a source one withing that range, then it will get no transfered data.

This allows to transfer a small sub-detailed mesh onto a more complete one (e.g. from a "hand" mesh towards a "full body" one).

Ray Radius For ray-casting-based mapping methods, the radius of the cast rays. Especially important for 1D and 2D items (i.e. vertices and edges), without some width there would be nearly no ray-casting matches...

Mix Mode Controls how destination data are affected:

All Replaces everything in destination (note that *Mix Factor* is still used).

Above Threshold Only replaces destination value if it's above given threshold *Mix Factor*. How that threshold is interpreted depends on data type, note that for boolean values this option fakes a logical AND.

Below Threshold Only replaces destination value if it's below given threshold *Mix Factor*. How that threshold is interpreted depends on data type, note that for boolean values this option fakes a logical OR.

Mix, Add, Subtract, Multiply Apply that operation, using mix factor to control how much of source or destination value to use. Only available for a few types (vertex groups, vertex colors).

Mix Factor How much of the transfered data gets mixed into existing one (not supported by all data types).

Vertex Group Allows per-item fine control of the mix factor. Vertex group influence can be reverted using the small "arrow" button to the right.

Generate Data Layers This modifier cannot generate needed data layers itself. Once the set of source data to transfer is selected, this button shall be used to generate matching destination layers.

Fig. 2.752: Data Transfer modifier.

Selection of Data to Transfer

To keep the size of the modifier reasonable, the kind of items to be affected must be selected first (vertices, edges, face corners and/or faces).

Mapping Type How is generated the mapping between those source and destination items. Each type has its own options, see *Geometry Mapping* below for details.

Data Types The left column of toggle buttons, to select which data types to transfer.

Multi-layers Data Types Options In those cases (vertex groups, vertex colors, UVs), one can select which source layers to transfer (usually, either all of them, or a single specified one), and how to affect destination (either by matching names, matching order/position, or, if a single source is selected, by specifying manually destination layer).

Islands Handling Refinement This setting only affects UV transfer currently. It allows to avoid a given destination face to get UV coordinates from different source UV islands. Keeping it at 0.0 means no island handling at all. Typically, small values like 0.02 are enough to get good results, but if you are mapping from a very high poly source towards a very low poly destination, you may have to raise it quite significantly.

Usage

First key thing to keep in mind when using this modifier is that it will **not** create destination data layers. *Generate Data Layers* button shall always be used for this purpose, once set of source data to transfer is selected. It should also be well understood that creating those data layers on destination mesh is **not** part of the modifier stack, which means e.g. that they will remain even once the modifier is deleted, or if source data selection is modified.

Geometry Mapping

Geometry mapping is the process by which a given destination vertex/edge/... knows **which part** of the source mesh to get its data from. It is crucial to understand this topic well to get good results with this modifier.

Topology The simplest option, expects both meshes to have identical number of items, and match them by order (indices). Useful e.g. between meshes

that were identical copies, and got deformed differently.

One-To-One Mappings Those always select only one source item for each destination one, often based on shortest distance.

Vertices

Nearest Vertex Uses source's nearest vertex.

Nearest Edge Vertex Uses source's nearest vertex of source's nearest edge.

Nearest Face Vertex Uses source's nearest vertex of source's nearest face.

Edges

Nearest Vertices Uses source's edge which vertices are nearest from destination edge's vertices.

Nearest Edge Uses source's nearest edge (using edge's midpoints).

Nearest Face Edge Uses source's nearest edge of source's nearest face (using edge's midpoints).

Face Corners A face corner is not a real item by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, ...) aspects are used to match them together.

Nearest Corner and Best Matching Normal Uses source's corner having the most similar *split* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner and Best Matching Face Normal Uses source's corner having the most similar *face* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner of Nearest Face Uses source's nearest corner of source's nearest face.

Faces

Nearest Face Uses source's nearest face.

Best Normal-Matching Uses source's face which normal is most similar with destination one.

Interpolated Mappings Those use several source items for each destination one, interpolating their data during the transfer.

Vertices

Nearest Edge Interpolated Uses nearest point on nearest source's edge, interpolates data from both source edge's vertices.

Nearest Face Interpolated Uses nearest point on nearest source's face, interpolates data from all that source face's vertices.

Projected Face Interpolated Uses point of face on source hit by projection of destination vertex along its own normal, interpolates data from all that source face's vertices.

Edges

Projected Edge Interpolated This is a sampling process. Several rays are cast from along the destination's edge (interpolating both edge's vertex normals), and if enough of them hit a source's edge, all hit source edges' data are interpolated into destination one.

Face Corners A face corner is not a real item by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, ...) aspects are used to match them together.

Nearest Face Interpolated Uses nearest point of nearest source's face, interpolates data from all that source face's corners.

Projected Face Interpolated Uses point of face on source hit by projection of destination corner along its own normal, interpolates data from all that source face's corners.

Faces

Projected Face Interpolated This is a sampling process. Several rays are cast from the whole destination's face (along its own normal), and if enough of them hit a source's face, all hit source faces' data are interpolated into destination one.

Mesh Cache Modifier

The *Mesh Cache* modifier is used so animated mesh data can be applied to a mesh and played back, deforming the mesh.

This works in a similar way to shape-keys but is aimed at playing back external files and is often used for interchange between applications.

Note: When using this modifier, the vertex locations are overwritten.

Options

Format The input file format (currently `.mdd` and `.pc2` are supported).

File Path Path to the cache file.

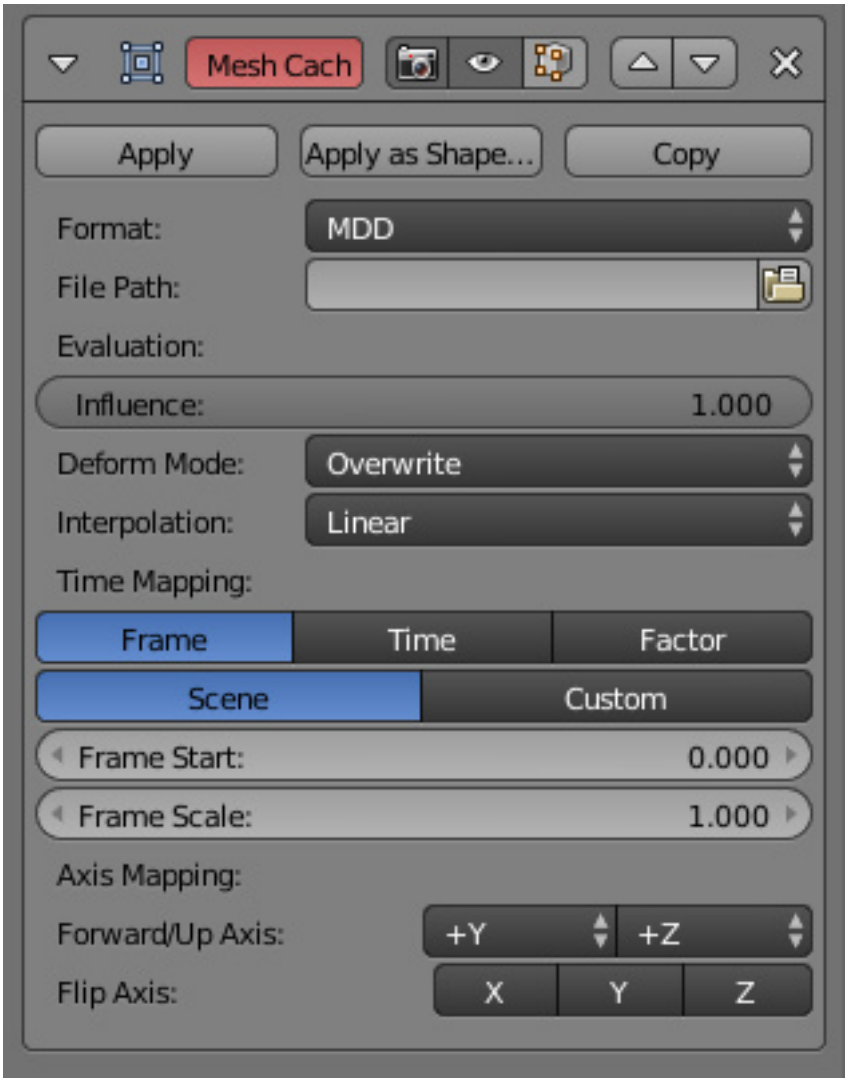


Fig. 2.753: Mesh Cache modifier.

Evaluation

Influence Factor to adjust the influence of the modifiers deformation, useful for blending in/out from the cache data.

Deform Mode This setting defaults to 'Overwrite' which will replace the vertex locations with those in the cache file. However, you may want to use shape-keys, for example, and mix them with the mesh-cache. In this case you can select the 'Deform' option which integrates deformations with the mesh-cache result.

Note: This feature is limited to making smaller, isolated edits and will not work for larger changes such as re-posing limbs.

Interpolation None or Linear which will blend between frames; use linear when the frames in the cache file do not match up exactly with the frames in the blend-file.

Time Mapping

Time Mode Select how time is calculated.

Frame Allows you to control the frames, which will ignore timing data in the file but is often useful since it gives simple control.

Time Evaluates time in seconds, taking into account timing information from the file (offset and frame-times).

Factor Evaluates the entire animation as a value from (0 - 1).

Play Mode Select how playback operates.

Scene Use the current frame from the scene to control playback.

Frame Start Play the cache starting from this frame.

Frame Scale Scale time by this factor (applied after the start value).

Custom Control animation timing manually.

Evaluation Value Property used for animation time, this gives more control of timing – typically this value will be animated.

Axis Mapping

Forward/Up Axis The axis for forward and up used in the source file.

Flip Axis In rare cases you may also need to flip the coordinates on an axis.

Tip: Both MDD and PC2 depend on the vertex order on the mesh remaining unchanged; this is a limitation with the method used so take care not to add/remove vertices once this modifier is used.

Mesh Sequence Cache Modifier

The *Mesh Sequence Cache Modifier* is used to **TODO**. Despite its name, this modifier supports meshes and curves. It also handles file sequences, as well as meshes and curves with varying number of vertices/control points.

Options

Cache File Data-block menu to select the Alembic file.

File Path Path to Alembic file.

Is sequence Whether or not the cache is separated in a series of files.

Override Frame Whether to use a custom frame for looking up data in the cache file, instead of using the current scene frame.

Frame The time to use for looking up the data in the cache file, or to determine which to use in a file sequence.

Manual Transform Scale Value by which to enlarge or shrink the object with respect to the world's origin. (only applicable through a *Transform Cache Constraint*)

Object Path The path to the Alembic object inside the archive.

Verts/Faces/UV/Color Type of data to read for a mesh object respectively: vertices, polygons, UV layers and Vertex Color layers.

Normal Edit Modifier

The *Normal Edit* modifier affects (or generates) custom normals. It uses a few simple parametric methods to compute normals (quite useful in game development and architecture areas), and mixes back those generated normals with existing ones.

Options

Radial/Directional The two modes currently available to generate normals.

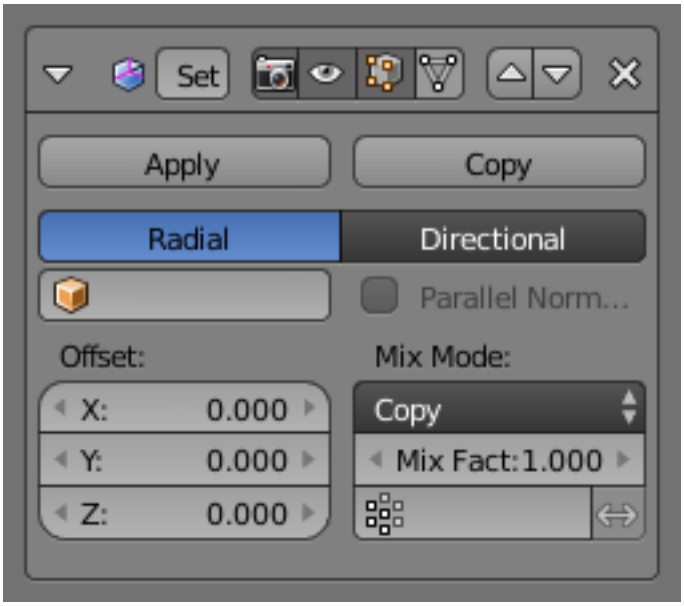


Fig. 2.754: Normal Edit modifier.

Radial aligns normals with the (origin, vertex coordinates) vector, in other words all normals seems to radiate from the given center point, as if they were emitted from an ellipsoid surface.

Directional makes all normals point (converge) towards a given target object.

Target Object Uses this object's center as reference point when generating normals.

Optional in *Radial* mode, mandatory in *Directional* one.

Parallel Normals Makes all normals parallel to the line between both objects' centers, instead of converging towards target's center.

Only relevant in *Directional* mode.

Offset Gives modified object's center an offset before using it to generate normals.

Only relevant in *Radial* mode if no *Target Object* is set, and in *Directional* mode when *Parallel Normals* is set.

Mix Mode How to affect existing normals with newly generated ones.

Note the *Multiply* option is **not** a cross product, but a mere component-by-component multiplication.

Mix Factor How much of the generated normals get mixed into existing ones.

Vertex Group Allows per-item fine control of the mix factor. Vertex group influence can be reverted using the small "arrow" button to the right.

Usage

This modifier can be used to quickly generate radial normals for low-poly tree foliage, or "fix" shading of toon-like rendering by partially bending default normals...

The only mandatory prerequisite to use it is to enable *Auto Smooth* option in Mesh properties, *Normals* panel.

Tip: More complex normal manipulations can be achieved by copying normals from one mesh to another, see the *Data Transfer modifier*.

UV Project Modifier

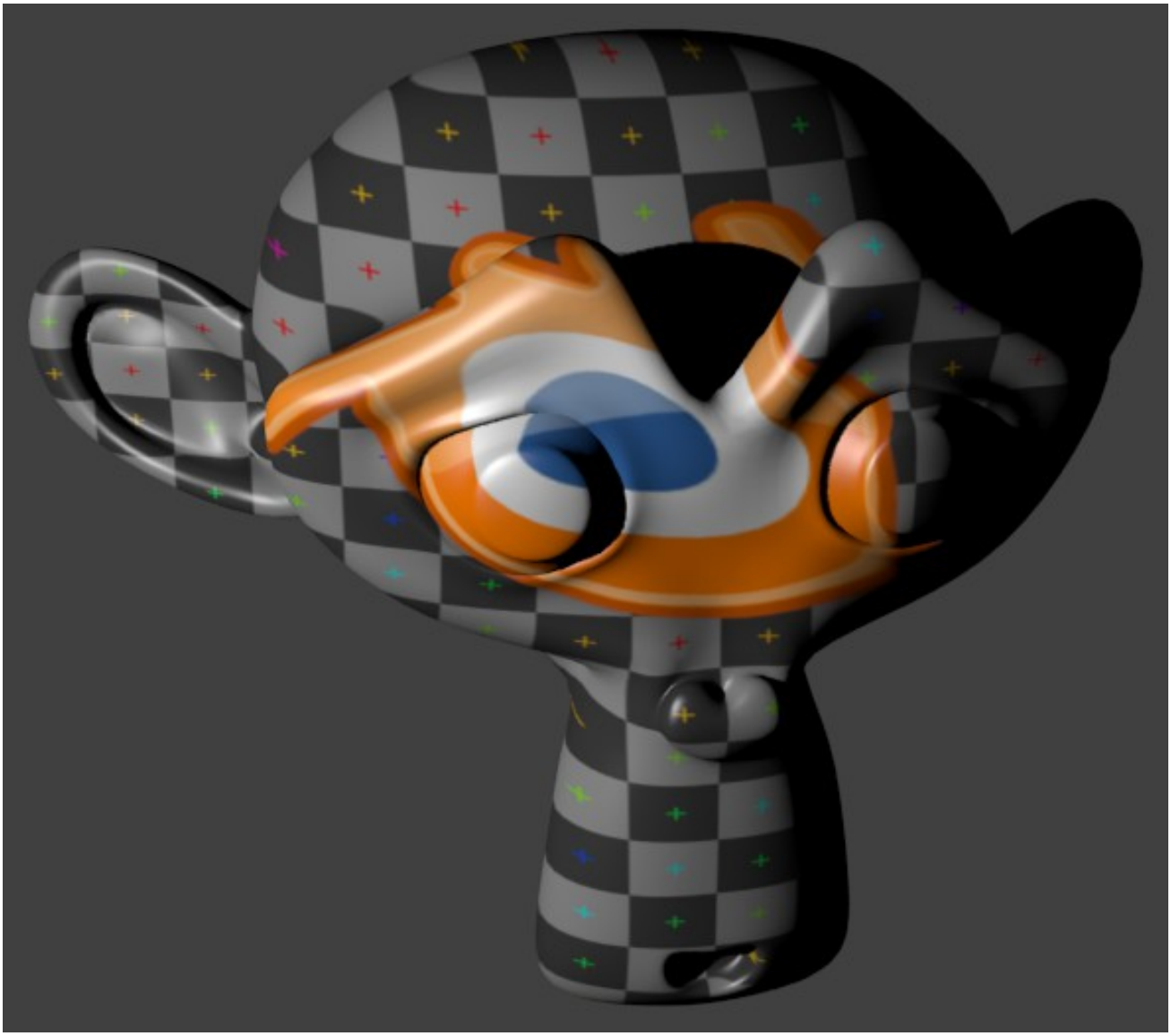
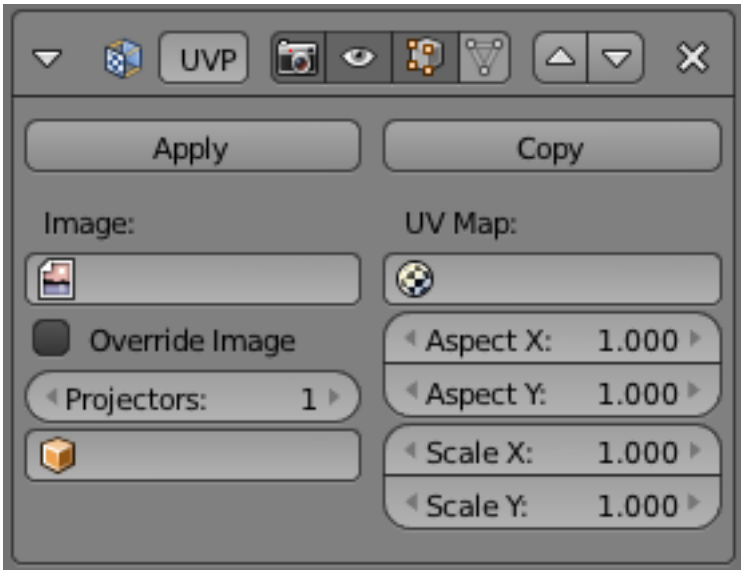


Fig. 2.755: Projecting the Blender logo onto Suzanne.

The *UV Project* Modifier acts like a slide projector. It emits a UV map from the negative Z-axis of a controller object (such as an *empty*), and applies it to the object as the "light" hits it. It can optionally override the objects face texture.

Download an example.

Options



UV layer Which UV layer to modify. Defaults to the active rendering layer.

Image The image associated with this modifier. Not required; you can just project a UV for use elsewhere. *Override Image*, below, defines how the image is used.

Override Image

- When true, the Face Texture of all vertices on the mesh is replaced with the Image. This will cause the image to repeat, which is usually undesirable.
- When false, the modifier is limited to faces with the Image as their Face Texture.

Projectors Up to ten projector objects are supported. Each face will choose the closest and aligned projector with its surface normal. Projections emit from the negative Z-axis (i.e. straight down a camera or lamp). If the projector is a camera, the projection will adhere to its perspective/orthographic setting.

Objects Specify the projector Object

Aspect X/Y and Scale X/Y These allow simple manipulation of the image. Only apply when a camera is used as projector Object.

Usage

General

UV Project is great for making spotlights more diverse, and also for creating decals to break up repetition.

The modifier's Image property is not generally used. Instead, a texture mapped to the UV layer that the modifier targets is added to the object's Material. This allows you to prevent the image from repeating by setting *Texture → Image Mapping → Extension to Clip*.

Perspective Cameras

When using perspective cameras or spot lamps, you will likely want to enable the *UV Project* Material Option (available in the materials panel), This uses a different UV interpolation to prevent distortion.

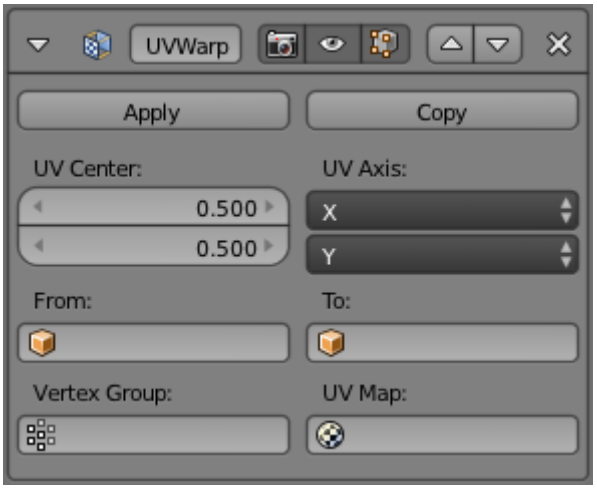
Note: This option is not yet available for Cycles

UV Warp Modifier

The *UV Warp* modifier uses two objects to define a transformation which is applied to the chosen UV coordinates.

Its purpose is to give you direct control over the object's UVs in the 3D View, allowing you to directly translate, rotate and scale existing UV coordinates using controller objects or bones.

Options



UV Center The center point of the UV map to use when applying scale or rotation. With (0, 0) at the bottom left and (1, 1) at the top right. Defaults to (0.5, 0.5).

UV Axis The axes to use when mapping the 3D coordinates into 2D.

From/To The two objects used to define the transformation. See *Usage* below.

Vertex Group The vertex group can be used to scale the influence of the transformation per-vertex.

UV Map Which UV map to modify. Defaults to the active rendering layer.

Usage

How the UVs are warped is determined by the difference between the transforms (location, rotation and scale) of the *from* and *to* objects.

If the *to* object has the same transforms as the *from* object, the UVs will not be changed.

Assuming the *UV Axis* of the modifier is X/Y and the scale of the objects are (1, 1, 1), if the *to* object is one unit away from the *from* object on the X-axis, the UVs will be transformed on the U-axis (horizontally) by one full UV space (the entire width of the image)

Vertex Weight Edit Modifier

This modifier is intended to edit the weights of one vertex group.

The general process is the following, for each vertex:

- (Optional) It does the mapping, either through one of the predefined functions, or a custom mapping curve.
- It applies the influence factor, and optionally the vertex group or texture mask (0.0 means original weight, 1.0 means fully mapped weight).
- It applies back the weight to the vertex, and/or it might optionally remove the vertex from the group if its weight is below a given threshold, or add it if it is above a given threshold.

Warning: This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

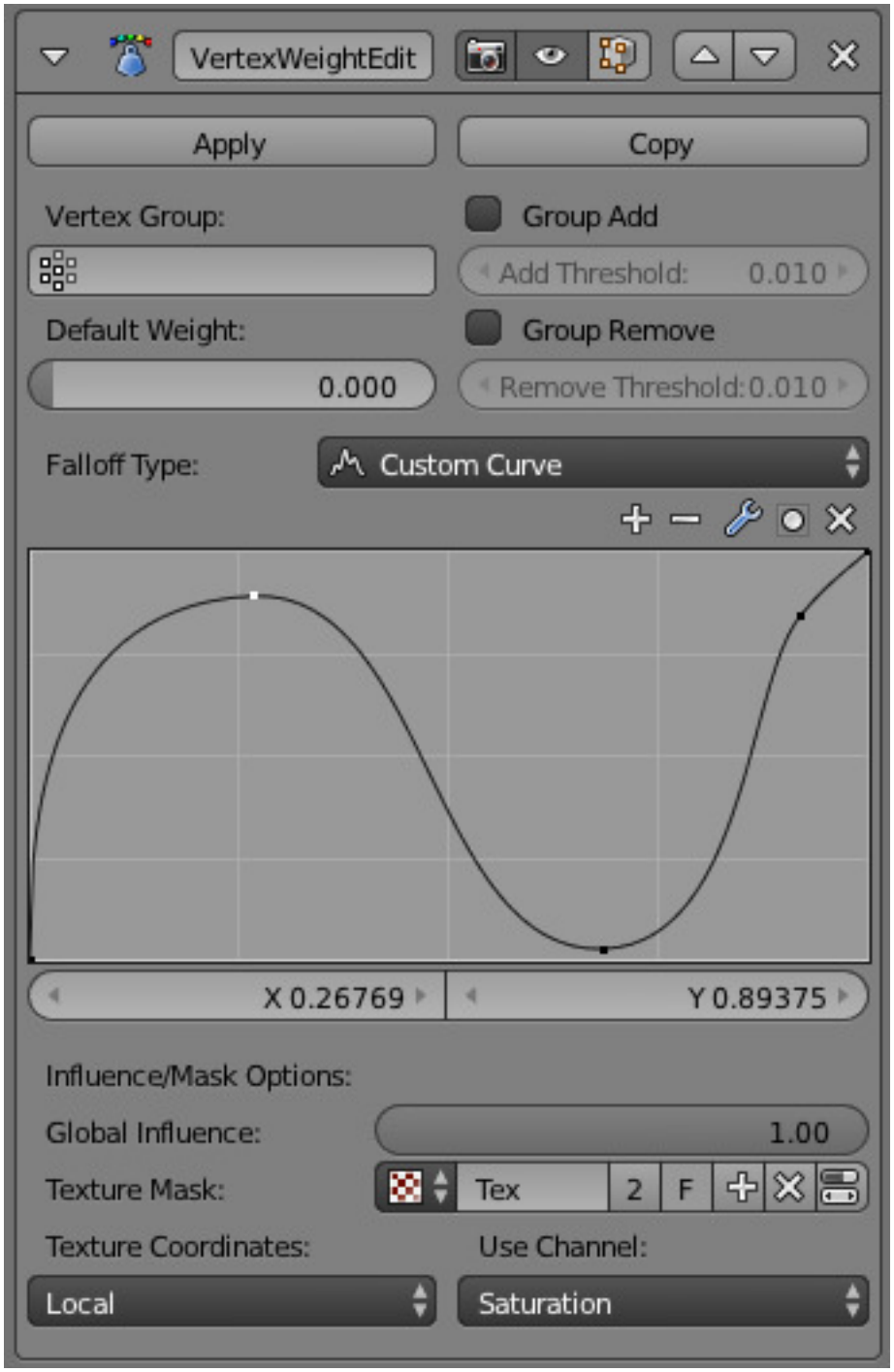


Fig. 2.756: The Vertex Weight Edit modifier panel.

Options

Vertex Group The vertex group to affect.

Default Weight The default weight to assign to all vertices not in the given vertex group.

Group Add Adds vertices with a final weight over *Add Threshold* to the vertex group.

Group Remove Removes vertices with a final weight below *Remove Threshold* from the vertex group.

Falloff Type Type of mapping:

Linear No mapping.

Custom Curve Allows the user to manually define the mapping using a curve.

Sharp, Smooth, Root and Sphere These are classical mapping functions, from spikiest to roundest.

Random Uses a random value for each vertex.

Median Step Creates binary weights (0.0 or 1.0), with 0.5 as cutting value.

Global Influence The overall influence of the modifier (0.0 will leave the vertex group's weights untouched, 1.0 is standard influence).

> **Warning:** Influence only affects weights, adding/removing of vertices to/from vertex group is not prevented by setting this value to 0.0.

Vertex Group Mask An additional vertex group, the weights of which will be multiplied with the global influence value for each vertex. If a vertex is not in the masking vertex group, its weight will be not be affected.

Texture Mask An additional texture, the values of which will be multiplied with the global influence value for each vertex.

This is a standard texture *data-block* control. When set, it reveals other settings:

Texture Coordinates How the texture is mapped to the mesh.

Local Use local vertex coordinates.

Global Use vertex coordinates in global space.

Object Use vertex coordinates in another object's space.

Object The object to be used as reference for *Object* mapping.

UV Use a UV layer's coordinates.

UV Layer The UV layer to be used for *UV* mapping.

Use Channel Which channel to use as weight factor source/

Red/Green/Blue/Alpha One of the color channels' values.

Intensity The average of the RGB channels (if RGB(1.0, 0.0, 0.0) value is 0.33)

Value The highest value of the RGB channels (if RGB(1.0, 0.0, 0.0) value is 1.0)

Hue Uses the hue value from the standard color wheel (e.g. blue has a higher hue value than yellow)

Saturation Uses the saturation value (e.g. pure red's value is 1.0, gray is 0.0)

Note: All of the channels above are gamma corrected, except for *Intensity*.

Note: You can view the modified weights in *Weight Paint Mode*. This also implies that you will have to disable the *Vertex Weight Edit Modifier* if you want to see the original weights of the vertex group you are editing.

Example

Using Distance from a Target Object's Geometry

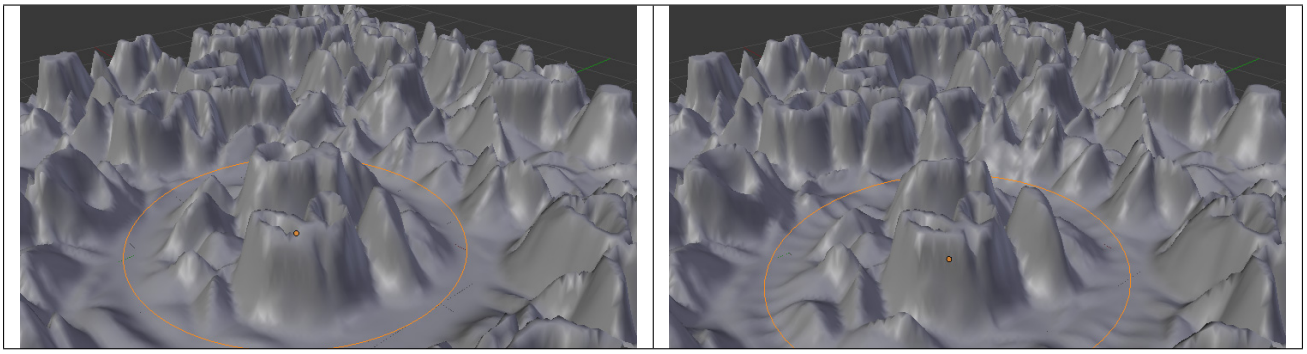
We are going to illustrate this with a *Displace* modifier.

Add a (10×10 BU) 100×100 vertices grid, and in *Edit Mode*, add to it a vertex group containing all of its vertices, as above. You can even further subdivide it with a first Subdivision Surface modifier.

Now add a curve circle, and place it 0.25 BU above the grid. Scale it up a bit (e.g. 4.0 BU).

Back to the grid object, add to it a *Vertex Weight Proximity* modifier, in *Geometry Distance* mode. Enable *Edge* (if you use *Vertex* only, and your curve has a low U definition, you would get wavy patterns, see Fig. *Wavy patterns.*).

Table 2.25: Distance from vertices.

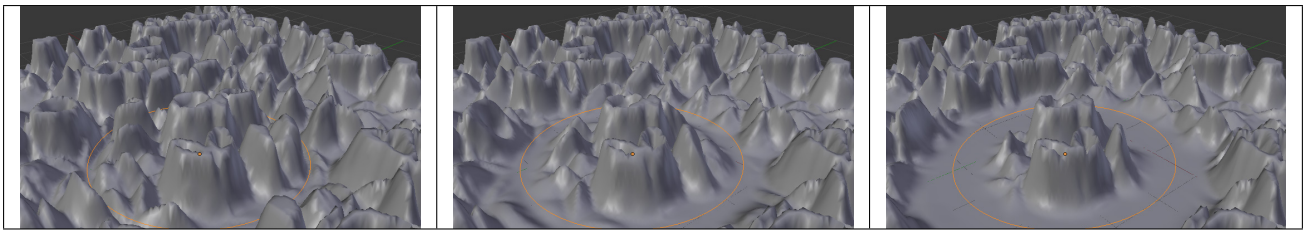


Set the *Lowest Distance* to 0.2, and the *Highest Distance* to 2.0, to map back the computed distances into the regular weight range.

Add a third *Displace* modifier and affect it the texture you like. Now, we want the vertices of the grid nearest to the curve circle to remain undisplaced. As they will get weights near zero, this means that you have to set the *Midlevel* of the displace to 0.0. Make it use our affected vertex group, and that is it! Your nice mountains just shrink to a flat plane near the curve circle.

As in the previous example, you can insert a *Vertex Weight Edit* modifier before the *Displace* one, and play with the *Custom Curve* mapping to get a larger/narrower "valley"...

Table 2.26: Convex-type mapping curve.



You can also add a fifth *Mask* modifier, and enable *Vertex Weight Edit* 's *Group Remove* option, with a *Remove Threshold* of 0.1, to see the bottom of your valley disappear.

The blend-file, TEST_2 scene.

Vertex Weight Mix Modifier

This modifier mixes a second vertex group (or a simple value) into the affected vertex group, using different operations.

Warning: This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

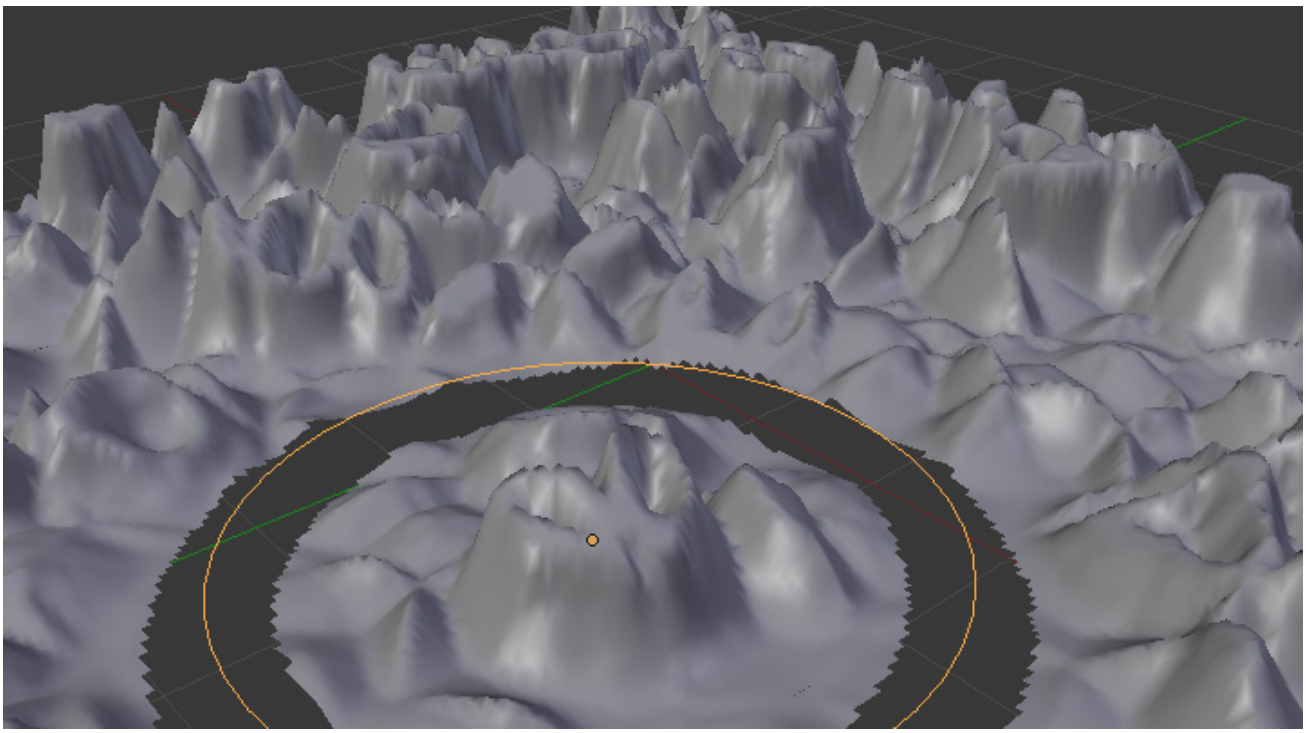


Fig. 2.762: Vertices with a computed weight below 0.1 removed from the vertex group.

Fig. 2.763: The Vertex Weight Mix modifier panel.

Options

Vertex Group A The vertex group to affect.

Default Weight A The default weight to assign to all vertices not in the given vertex group.

Vertex Group B The second vertex group to mix into the affected one. Leave it empty if you only want to mix in a simple value.

Default Weight B The default weight to assign to all vertices not in the given second vertex group.

Mix Mode How the vertex group weights are affected by the other vertex group's weights.

Replace weights Replaces affected weights with the second group's weights.

Add to weights Adds the values of *Group B* to *Group A*.

Subtract from weights Subtracts the values of *Group B* from *Group A*.

Multiply weights Multiplies the values of *Group B* with *Group A*.

Divide weights Divides the values of *Group A* by *Group B*.

Difference Subtracts the smaller of the two values from the larger.

Average Adds the values together, then divides by 2.

Mix Set Choose which vertices will be affected.

All vertices Affects all vertices, disregarding the vertex groups content.

Vertices from group A Affects only vertices belonging to the affected vertex group.

Vertices from group B Affects only vertices belonging to the second vertex group.

Vertices from one group Affects only vertices belonging to at least one of the vertex groups.

Vertices from both groups Affects only vertices belonging to both vertex groups.

> **Warning:** When using *All vertices*, *Vertices from group B* or *Vertices from one group*, vertices might be added to the affected vertex group.

Global Influence The overall influence of the modifier (0.0 will leave the vertex group's weights untouched, 1.0 is standard influence).

> **Warning:** Influence only affects weights, adding/removing of vertices to/from vertex group is not prevented by setting this value to 0.0.

Vertex Group Mask An additional vertex group, the weights of which will be multiplied with the global influence value for each vertex. If a vertex is not in the masking vertex group, its weight will be not be affected.

Texture Mask An additional texture, the values of which will be multiplied with the global influence value for each vertex.

This is a standard texture *data-block* control. When set, it reveals other settings:

Texture Coordinates How the texture is mapped to the mesh.

Local Use local vertex coordinates.

Global Use vertex coordinates in global space.

Object Use vertex coordinates in another object's space.

Object The object to be used as reference for *Object* mapping.

UV Use a UV layer's coordinates.

UV Layer The UV layer to be used for *UV* mapping.

Use Channel Which channel to use as weight factor source/

Red/Green/Blue/Alpha One of the color channels' values.

Intensity The average of the RGB channels (if RGB(1.0, 0.0, 0.0) value is 0.33)

Value The highest value of the RGB channels (if RGB(1.0, 0.0, 0.0) value is 1.0)

Hue Uses the hue value from the standard color wheel (e.g. blue has a higher hue value than yellow)

Saturation Uses the saturation value (e.g. pure red's value is 1.0, gray is 0.0)

Note: All of the channels above are gamma corrected, except for *Intensity*.

Note: You can view the modified weights in *Weight Paint Mode*. This also implies that you will have to disable the *Vertex Weight Mix Modifier* if you want to see the original weights of the vertex group you are editing.

Example

Using a Texture and the Mapping Curve

Here we are going to create a sort of strange alien wave (yes, another example with the *Wave* modi-

fier... but it is a highly visual one; it is easy to see the vertex group effects on it...).

So as above, add a 100×100 grid. This time, add a vertex group, but without assigning any vertex to it – we will do this dynamically.

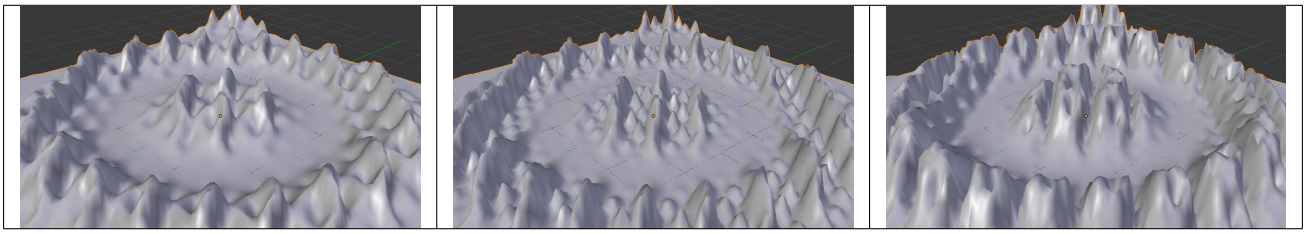
Add a first *Vertex Weight Mix* modifier, set the *Vertex Group A* field with a *Default Weight A* of 0.0, and set *Default Weight B* to 1.0.

Leave the *Mix Mode* to *Replace weights*, and select *All vertices* as *Mix Set*. This way, all vertices are affected. As none are in the affected vertex group, they all have a default weight of 0.0, which is replaced by the second default weight of 1.0. And all those vertices are also added to the affected vertex group.

Now, select or create a masking texture. The values of this texture will control how much of the "second weight" of 1.0 replaces the "first weight" of 0.0 ... In other words, they are taken as weight values!

You can then select which texture coordinates and channel to use. Leave the mapping to the default *Local* option, and play with the various channels...

Table 2.27: Using Saturation.

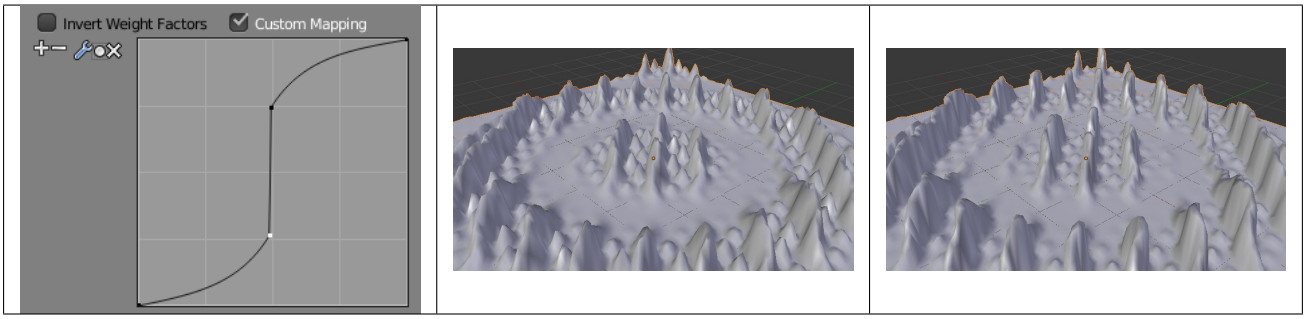


Do not forget to add a *Wave* modifier, and select your vertex group in it!

You can use the weights created this way directly, but if you want to play with the curve mapping, you must add the famous *Vertex Weight Edit* modifier, and enable its *Custom Curve* mapping.

By default, it is an one-to-one linear mapping – in other words, it does nothing! Change it to something like in Fig. *Custom mapping curve.*, which maps (0.0, 0.5) to (0.0, 0.25) and (0.5, 1.0) to (0.75, 1.0), thus producing nearly only weights below 0.25, and above 0.75: this creates great "walls" in the waves...

Table 2.28: Custom Mapping enabled.



The blend-file, TEST_4 scene.

Vertex Weight Proximity Modifier

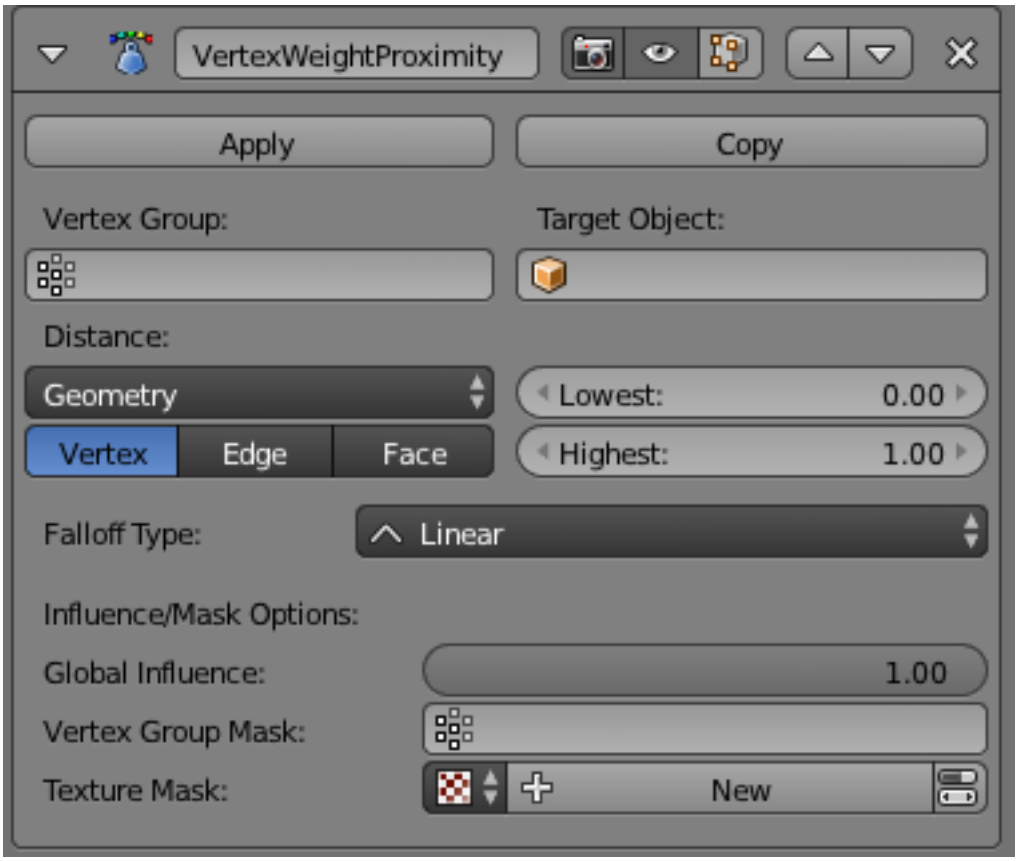


Fig. 2.770: The Vertex Weight Proximity modifier panel.

This modifier sets the weights of the given vertex group, based on the distance between the object (or its vertices), and another target object (or its geometry).

Warning: This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

Options

Vertex Group The vertex group to affect.

Target Object The object from which to compute distances.

Proximity mode

Object Distance Use the distance between the modified mesh object and the target object as weight for all vertices in the affected vertex group.

Geometry Distance Use the distance between each vertex and the target object, or its geometry.

Vertex This will set each vertex's weight from its distance to the nearest vertex of the target object.

Edge This will set each vertex's weight from its distance to the nearest edge of the target object.

Face This will set each vertex's weight from its distance to the nearest face of the target object.

Note: If you enable more than one of them, the shortest distance will be used. If the target object has no geometry (e.g. an empty or camera), it will use the location of the object itself.

Lowest Distance mapping to 0.0 weight.

Highest Distance mapping to 1.0 weight.

Tip: *Lowest* can be set above *Highest* to reverse the mapping.

Falloff Type Type of mapping:

Linear No mapping.

Custom Curve Allows the user to manually define the mapping using a curve.

Sharp, Smooth, Root and Sphere These are classical mapping functions, from spikiest to roundest.

Random Uses a random value for each vertex.

Median Step Creates binary weights (0.0 or 1.0), with 0.5 as cutting value.

Global Influence The overall influence of the modifier (0.0 will leave the vertex group's weights untouched, 1.0 is standard influence).

Warning: Influence only affects weights, adding/removing of vertices to/from vertex group is not prevented by setting this value to 0.0.

Vertex Group Mask An additional vertex group, the weights of which will be multiplied with the

global influence value for each vertex. If a vertex is not in the masking vertex group, its weight will be not be affected.

Texture Mask An additional texture, the values of which will be multiplied with the global influence value for each vertex.

This is a standard texture *data-block* control. When set, it reveals other settings:

Texture Coordinates How the texture is mapped to the mesh.

Local Use local vertex coordinates.

Global Use vertex coordinates in global space.

Object Use vertex coordinates in another object's space.

Object The object to be used as reference for *Object* mapping.

UV Use a UV layer's coordinates.

UV Layer The UV layer to be used for *UV* mapping.

Use Channel Which channel to use as weight factor source/

Red/Green/Blue/Alpha One of the color channels' values.

Intensity The average of the RGB channels (if RGB(1.0, 0.0, 0.0) value is 0.33)

Value The highest value of the RGB channels (if RGB(1.0, 0.0, 0.0) value is 1.0)

Hue Uses the hue value from the standard color wheel (e.g. blue has a higher hue value than yellow)

Saturation Uses the saturation value (e.g. pure red's value is 1.0, gray is 0.0)

Note: All of the channels above are gamma corrected, except for *Intensity*.

Note: You can view the modified weights in *Weight Paint Mode*. This also implies that you will have to disable the *Vertex Weight Proximity Modifier* if you want to see the original weights of the vertex group you are editing.

Example

Using Distance from a Target Object

In this example let us dynamically control a *Wave* modifier with a modified vertex group.

1. Add a *Grid* mesh with (100×100) x/y subdivisions and a 5 BU Radius.

2. Switch to *Edit Mode* `Tab`, and in the *Object Data* properties, *Vertex Groups* panel, add a vertex group. Assign to it all your mesh's vertices with 1.0 weight.
3. Go back to *Object Mode*. Then, go to the *Modifiers* properties, and add a *Vertex Weight Proximity* modifier. Set the Distance mode to *Object*. Select your vertex group, and the target object you want.

   You will likely have to adjust the linear mapping of the weights produced by the *Vertex Weight Proximity* modifier. To do so, edit *Lowest Distance* and *Highest Distance* so that the first corresponds to the distance between your target object and the vertices you want to have lowest weight, and similarly with the second and highest weight...
4. If your lamp is at Z-hight of 2 then set the settings for the weight proximity modifier to: Lowest: 2 and highest: 7 (this will stop the waves under the lamp) If you want waves to be only under the lamp, set the lowest to 7 and highest to 2.
5. Now add a *Wave* modifier, set it to your liking, and use the same vertex group to control it. Example settings-speed: 0.10 , Height: 1.0 , Width 1.50 , Narrowness: 1.50.
6. Animate your target object, making it move over the grid. As you can see, the waves are only visible around the reference object! Note that you can insert a *Vertex Weight Edit* modifier before the *Wave* one, and use its *Custom Curve* mapping to get larger/narrower "wave influence's slopes".

   The blend-file, TEST_1 scene.

Generate

Array Modifier

The Array modifier creates an array of copies of the base object, with each copy being offset from the previous one in any of a number of possible ways. Vertices in adjacent copies can be merged if they are nearby, allowing smooth *Subdivision Surface* frameworks to be generated.

This modifier can be useful when combined with tileable meshes for quickly developing large scenes. It is also useful for creating complex repetitive shapes.

Multiple array modifiers may be active for an object at the same time (e.g. to create complex three dimensional constructs).

Options

Fit Type Controls how the length of the array is determined. There are three choices, activating respectively the display of the *Curve*, *Length* or *Count* settings explained below:

Fit Curve Generates enough copies to fit within the length of the curve object specified in *Curve*.

Fit Length Generates enough copies to fit within the fixed length given by *Length*.

Fixed Count Generates the number of copies specified in *Count*.

Note:

- Both *Fit Curve* and *Fit Length* use the local coordinate system size of the base object, which means that scaling the base object in *Object Mode* will not change the number of copies generated by the *Array* modifier.
- *Fit Length* uses the local coordinate system length of the curve, which means that scaling the curve in *Object Mode* will not change the number of copies generated by the *Array* modifier.
- Applying the scale with `Ctrl-A` can be useful for each one.

Constant Offset, X, Y, Z Adds a constant translation component to the duplicate object's offset. X, Y and Z constant components can be specified.

Relative Offset, X, Y, Z Adds a translation equal to the object's bounding box size along each axis, multiplied by a scaling factor, to the offset. X, Y and Z scaling factors can be specified.

Object Offset Adds a transformation taken from an object (relative to the current object) to the offset. It is good practice to use an Empty object centered or near to the initial object. E.g. by rotating this Empty a circle or helix of objects can be created.

Merge If enabled, vertices in each copy will be merged with vertices in the next copy that are within the given *Distance*.

First Last If enabled **and** *Merge* is enabled, vertices in the first copy will be merged with vertices in the last copy (this is useful for circular objects).

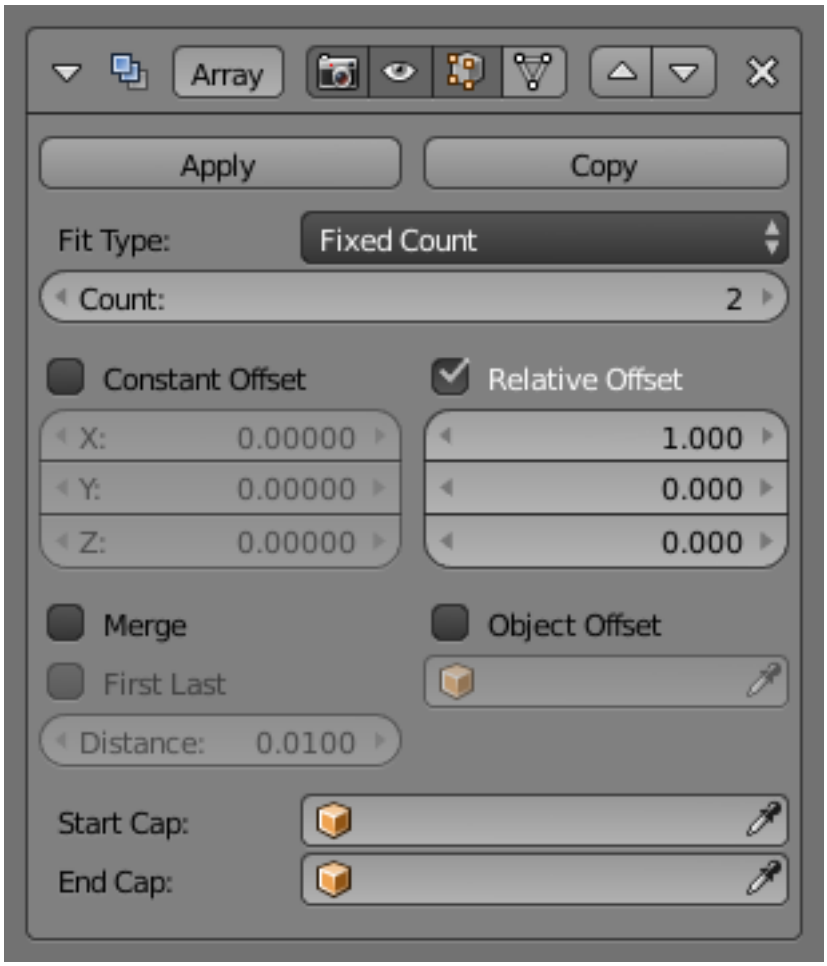


Fig. 2.771: Array modifier.

Fig. 2.772: Relative offset (0.5, 1.0 and 1.5) examples.

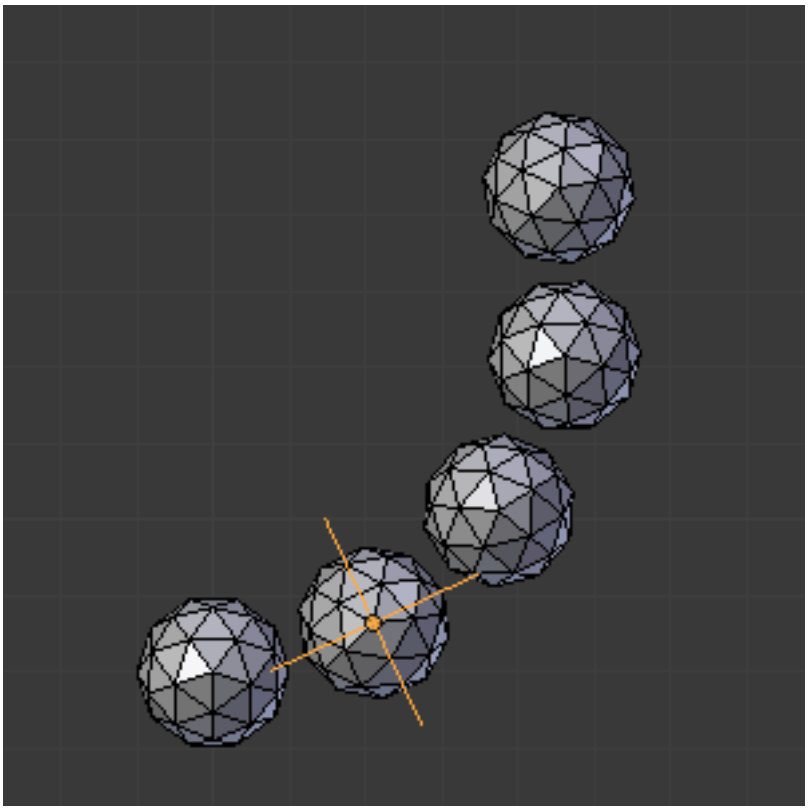
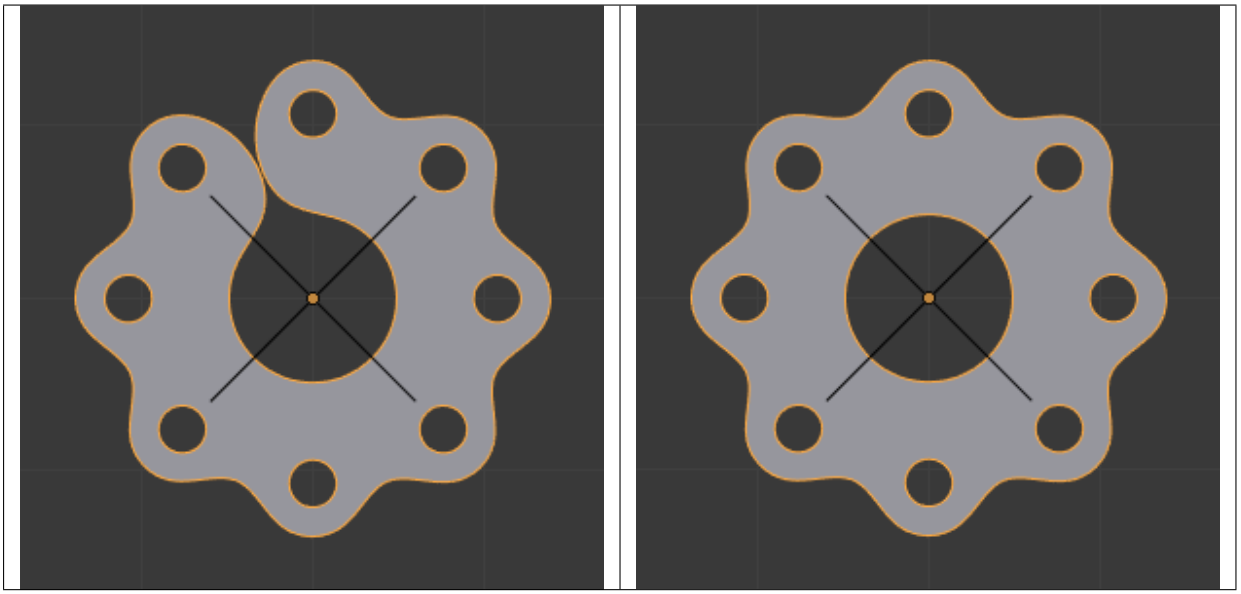


Fig. 2.773: Object offset example.

Table 2.29: Subdivision discontinuity eliminated by merging vertices between first and last copies (*First Last* on).



Distance Controls the merge distance for *Merge*.

Start Cap / End Cap This allows either endpoints of the array to have a different mesh subsisted.

For the *start*: as if it was in position -1, i.e. one "array step" before the first "regular" array copy. For the *end*: as if it was in position $n + 1$, i.e. one "array step" after the last "regular" array copy.

When *Merge* is activated, and the *cap* vertices are within the distance threshold, they will be merged.

Note: The start/end cap objects currently do not support the *First Last* option.

Hints

Offset Calculation

The transformation applied from one copy to the next is calculated as the sum of the three different components (*Relative*, *Constant* and *Object*), each of which can be enabled/disabled independently of the others. This allows, for example, a relative offset of (1.0, 0.0, 0.0) and a constant offset of (0.1, 0.0, 0.0), giving an array of objects neatly spaced along the X axis with a constant 0.1 unit between them, whatever the original object's size.

Examples

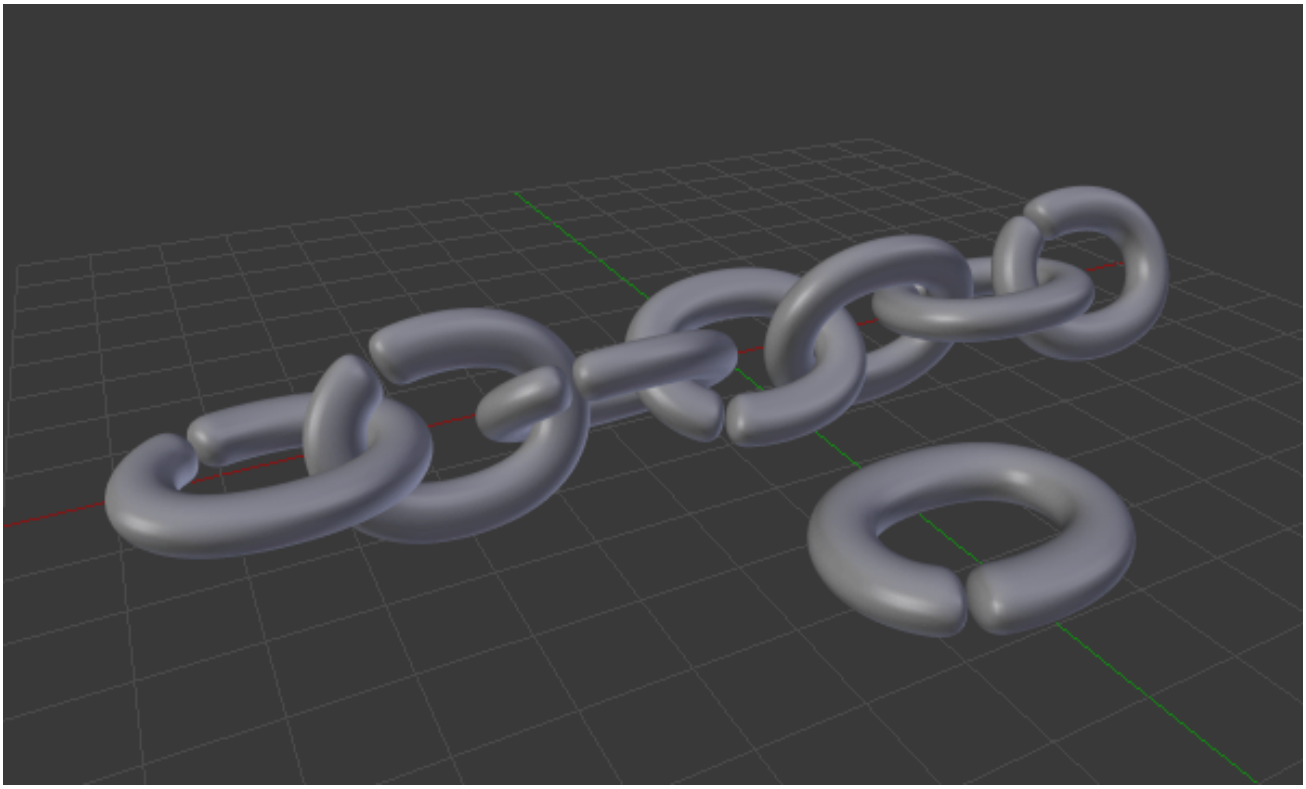


Fig. 2.776: A chain created from a single link. Sample blend-file.

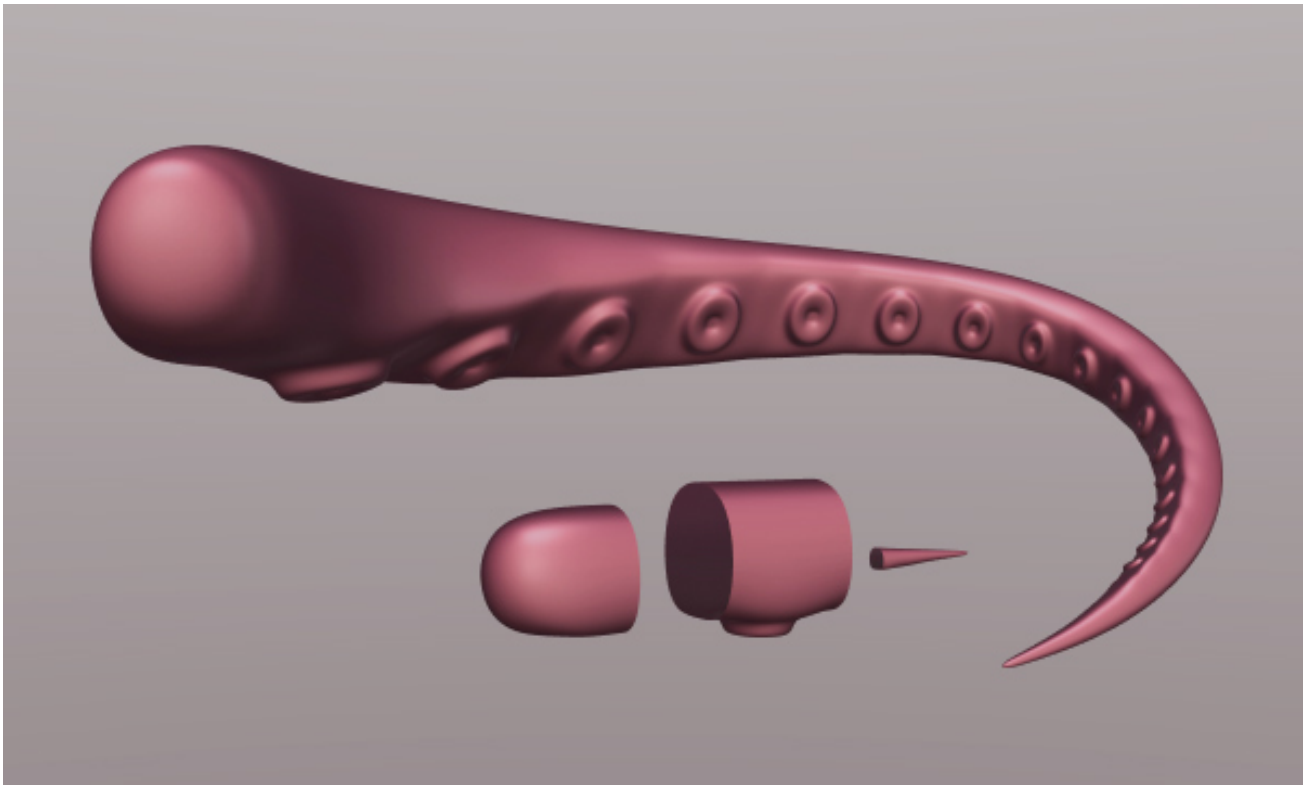


Fig. 2.777: A tentacle created with an Array modifier followed by a Curve modifier.
The segment in the foreground is the base mesh for the tentacle; the tentacle is capped by two specially-modeled objects deformed by the same Curve object as the main part of the tentacle. Sample blend-file.

Mechanical

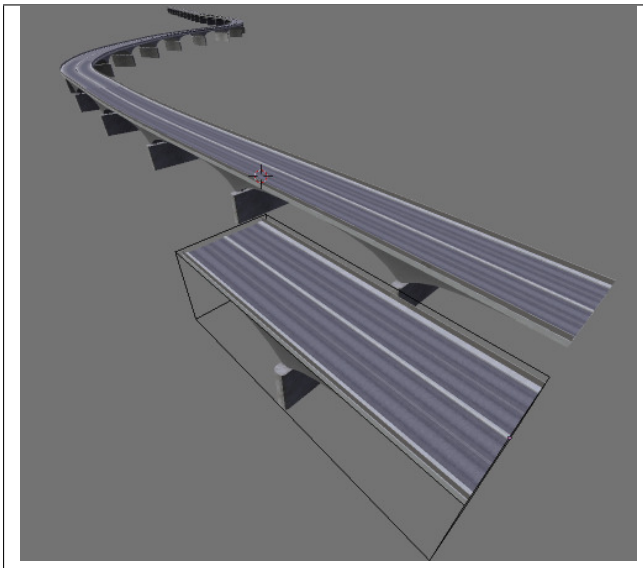


Fig. 2.778: A bridge made from a tileable mesh.

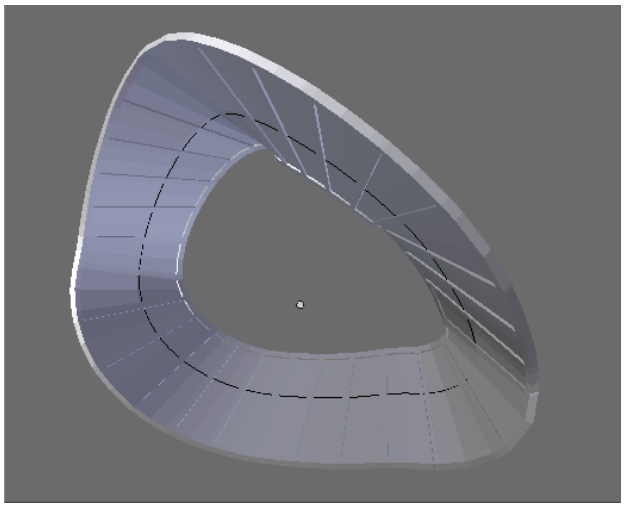


Fig. 2.779: A track. Sample blend-file.

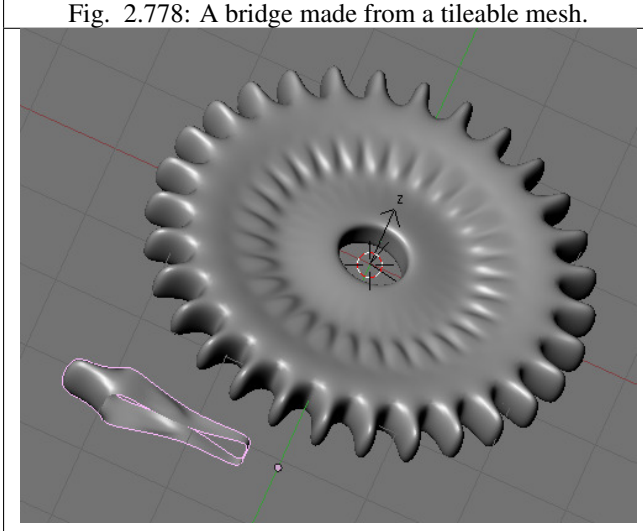


Fig. 2.780: A cog created from a single segment. Blend.

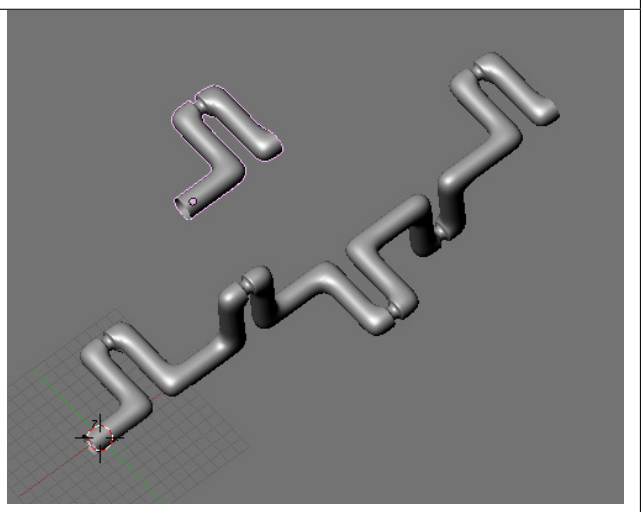


Fig. 2.781: A crankshaft. Sample blend-file.

Fractal

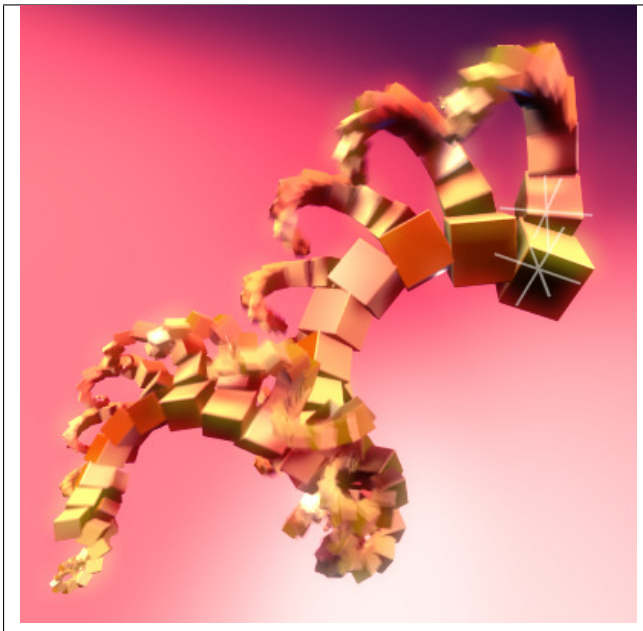


Fig. 2.782: Multi-level array animated with motion blur.

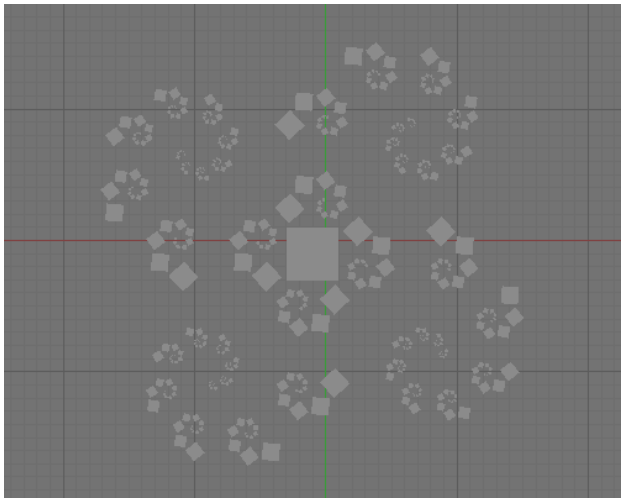


Fig. 2.783: Fractal created with multiple arrays. Blend.

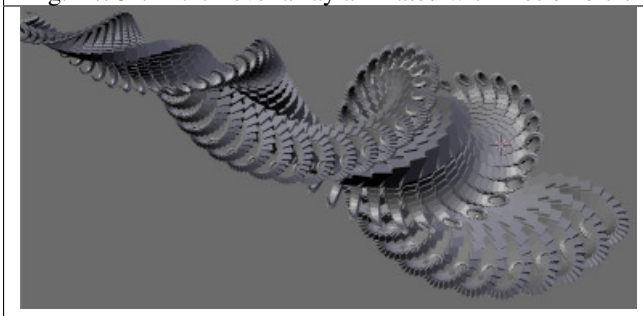


Fig. 2.784: A fractal fern image created with two array modifiers and one mirror applied to a cube.

Organic

Fig. 2.785: Subdivided cube array with one object offset, four cubes and a high vertex merge setting to give the effect of skinning.

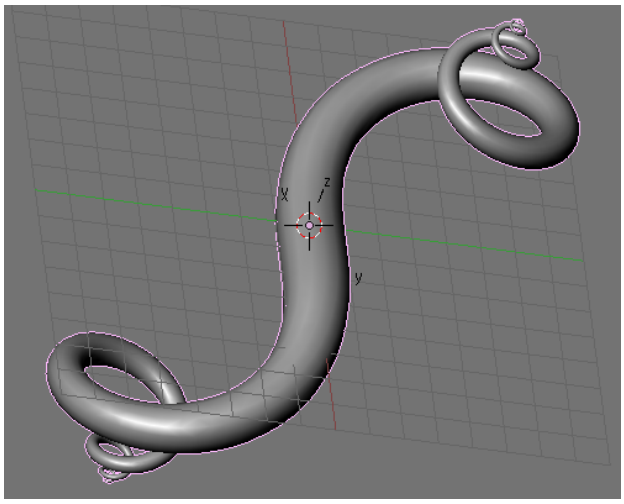


Fig. 2.786: A double spiral created with two array modifiers and one Subdivision Surface modifier applied to a cube. As above, the vertex merge threshold is set very high to give the effect of skinning. Sample blend-file.

Tutorials

- Neal Hirsig's Array Modifier Screencast on Vimeo.
- Creating A Double Helix With Modifiers.

The 'Double Helix' tutorial explains the Array modifier. It is for an old Blender Version (2.44) but except for the keyboard shortcuts it is still valid.

Bevel Modifier

The Bevel modifier adds the ability to bevel the edges of the mesh it is applied to, allowing control of how and where the bevel is applied to the mesh.

The Bevel modifier is a non-destructive alternative to the *Bevel Operation* in edit mode.

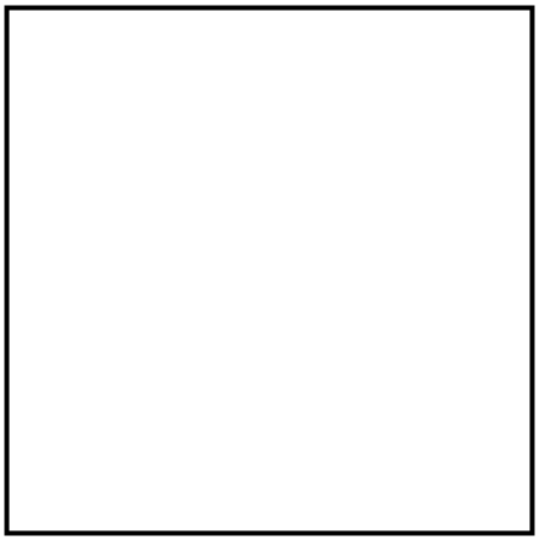


Fig. 2.787: Not beveled.

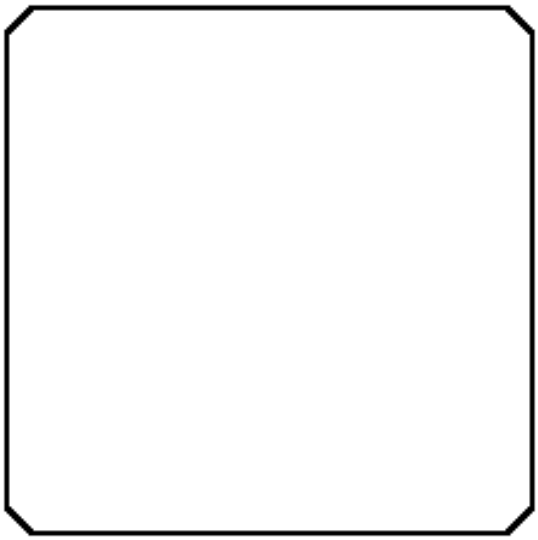


Fig. 2.788: Beveled.

The images above show the side views of a plain (Not beveled) cube and a beveled one.

Options

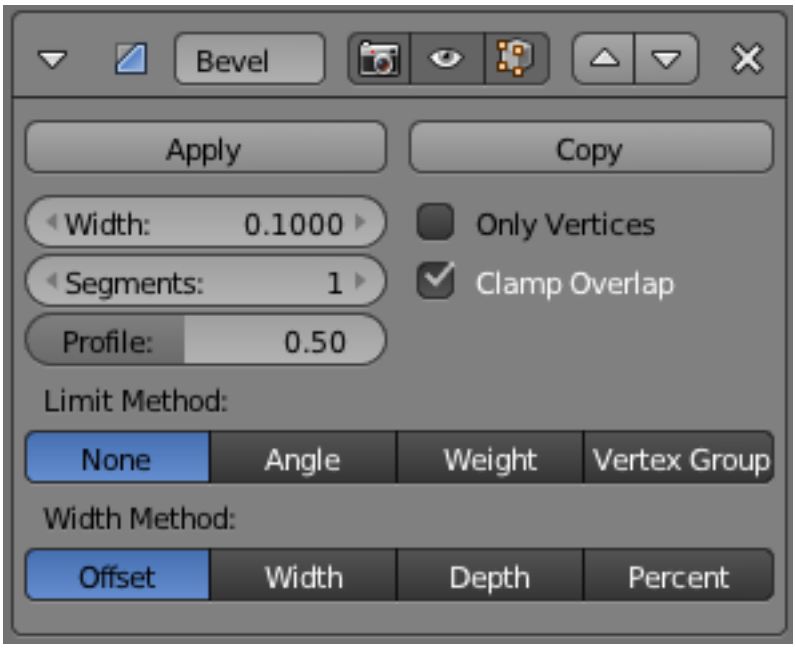


Fig. 2.789: Bevel modifier panel.

Width The size of the bevel affect. See *Width Method* below.

Segments The number of edge loops added along the bevel's face.

Profile The shape of the bevel, from concave to

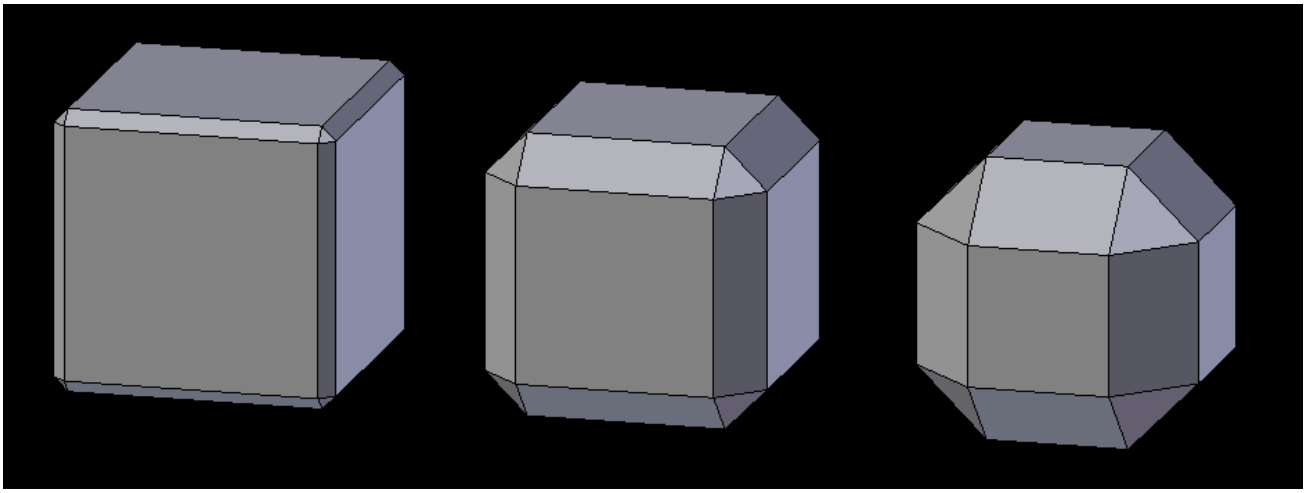


Fig. 2.790: Three Cubes with 0.1, 0.3 and 0.5 bevel Widths.

convex. It has no effect if *Segments* is less than 2.

Material The index of the material slot to use for the bevel. When set to -1, the material of the nearest original face will be used.

Only Vertices When enabled, only the areas near vertices are beveled; the edges are left not beveled.

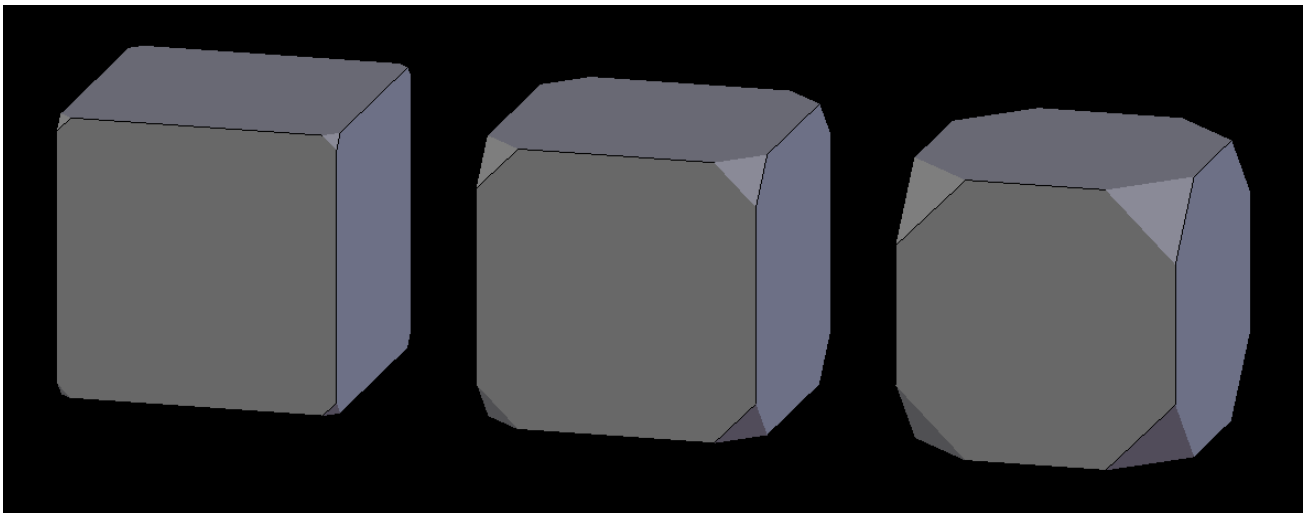


Fig. 2.791: Three cubes with 0.1, 0.3 and 0.5 bevel Widths, with Only Vertices option enabled.

Clamp Overlap When enabled, the width of each beveled edge will be limited such that they cannot intersect each other. Edges that are far apart will still bevel with the full width, only edges too close to each other are affected.

Limit Method Used to control where a bevel is applied to the mesh.

None No limit, all edges will be beveled.

Angle Only edges where the adjacent faces form an angle smaller than the defined threshold will be beveled. Intended to allow you to bevel only the sharp edges of an object without affecting its smooth surfaces.

Weight Use each edge's bevel weight to determine the width of the bevel. When the bevel weight is 0.0, no bevel is applied. See *here* about adjusting bevel weights.

Vertex Group Use weights from a vertex group to determine the width of the bevel. When the vertex weight is 0.0, no bevel is applied. An edge is only beveled if both of its vertices are in the vertex group. See *here* about adjusting vertex group weights.

Width Method Used to control how the *Width* is measured.

Offset Amount is offset of new edges from original.

Width Amount is width of new face.

Depth Amount is perpendicular distance from original edge to bevel face.

Percent Amount is percent of adjacent edge length.

Boolean Modifier

The Boolean modifier performs operations on meshes that are otherwise too complex to achieve with as few steps by editing meshes manually. The Boolean modifier uses one of three Boolean operations that can be used to create a single mesh out of two mesh objects:

Difference Negation

Union Disjunction

Intersect Conjunction

Note:

- The Boolean modifier works with open and closed volumes.
- The Boolean modifier does not work on edges without faces.
- The target topology determines the new topology of the modified mesh.
- The face normals are taken into account for the calculations.

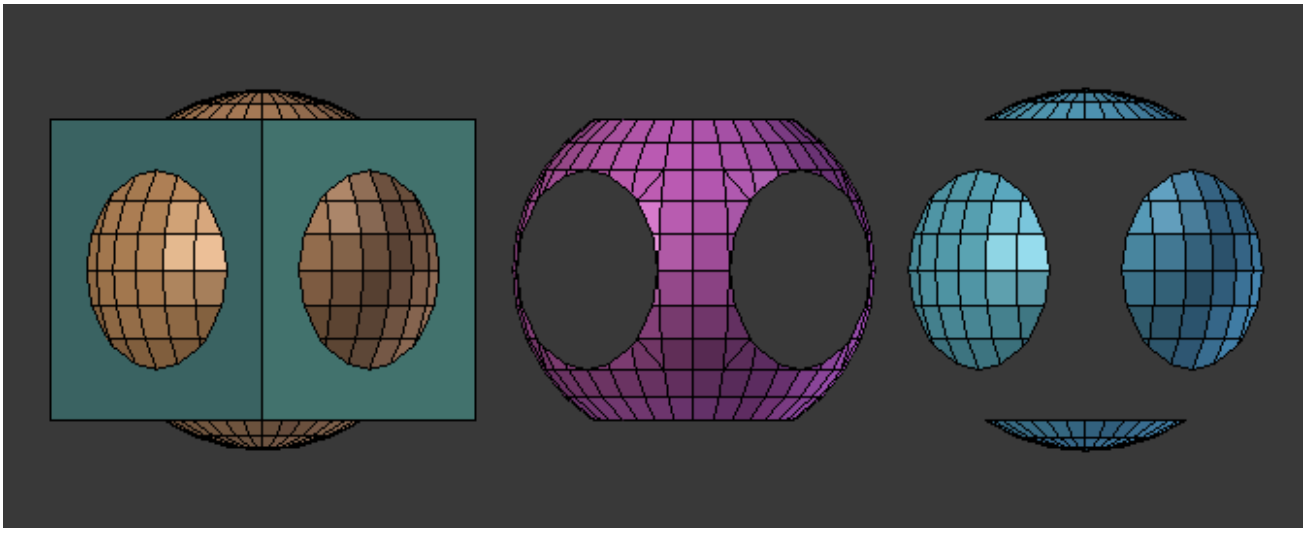


Fig. 2.792: The Union, Intersection and Difference between a Cube and a UV Sphere, with the modifier applied to the Sphere and using the cube as target.

- Whether faces are marked for smooth or flat for shading does not affect the calculations of this modifier.
- The line at which this modifier is calculated is delimited by the first tangential contact between faces of the modified mesh and target.

Tip: This is a dynamic real-time modifier!

If you have marked your Objects to show the Edges (in *Properties Editor* → *Object* → *Display*, enable *Wire*), you will see the Edge creation process while you are moving your Objects. Depending on your mesh topology, you can also enable X-Ray and Transparency and see the topology being created in real time.

Usage

To use the *Boolean Modifier* select the desired mesh Object then add a *Boolean modifier*. When you add the Boolean modifier for an object, Blender will need a second object to be the target of the operation. You can use open or closed meshes, as long as they have available face normals for the calculations to take effect. You can add one or more modifiers of this type for an Object but you can only apply one operation for the Boolean modifier at a time.

Options

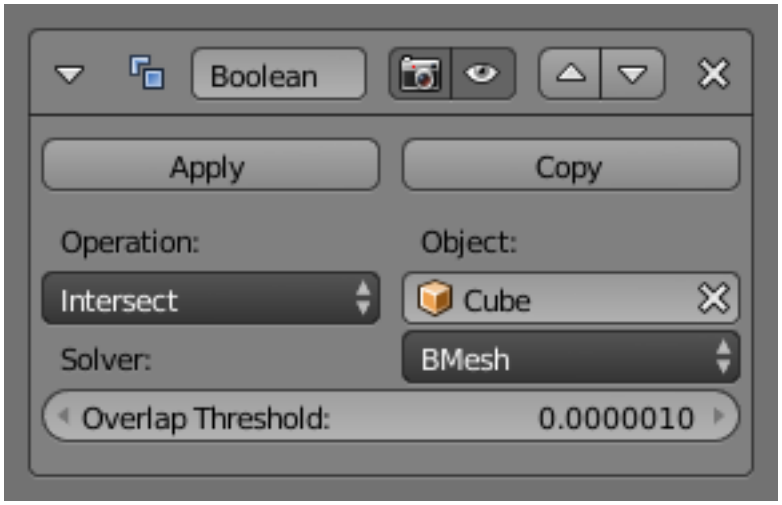


Fig. 2.793: Boolean Modifier Options.

Operations

Operation Which boolean operation will be used.

Difference The modified mesh is subtracted from the target mesh.

- If the target Mesh has inverted normals, Blender will Intersect the modified mesh.
- If the modified Mesh has inverted normals, Blender will add both meshes (Union).
- If both Meshes use inverted normals, Blender will Intersect the target Mesh.

Union The target mesh is added to the modified mesh.

- If the target Mesh has inverted normals, Blender will Intersect the target Mesh.
- If the modified Mesh has inverted normals, Blender will subtract the target Mesh.
- If both Meshes use inverted normals, Blender will Intersect the modified Mesh.

Intersect The target mesh is subtracted from the modified mesh.

- If the target Mesh has inverted normals, Blender will subtract the target Mesh.
- If the modified Mesh has inverted normals, Blender will intersect the target Mesh.
- If both Meshes use inverted normals, Blender will add both meshes (Union).

Object The name of the target mesh object.

Solver TODO

Materials

The Boolean modifier preserves the Materials of the participant Meshes, including their basic textures and mappings, and the modified mesh will receive its first active material index assigned to its new topology (the first active material).

Below, some examples are shown to exemplify how materials work with the Boolean modifier; we took the cube as the modified mesh, and the icosphere as the target with one material (white). We added four different indexes to one of the faces of the cube, leaving another basic material in the other faces. The top left image shows how the Boolean modifier interacts with the materials. The other three images show the three different Boolean operations applied to the modified mesh. In all the images the meshes have normals pointed outwards with the Icosphere as the target, and the Cube being the modified mesh.

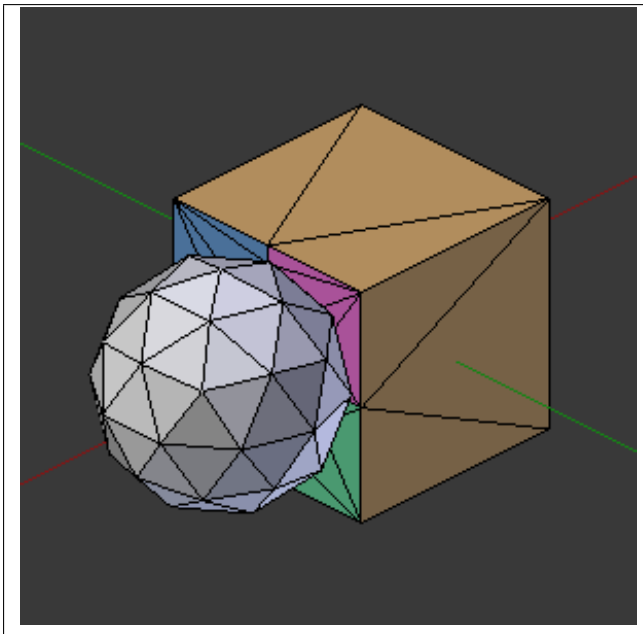


Fig. 2.794: Cube with Multi-Materials and Icosphere with basic Material.

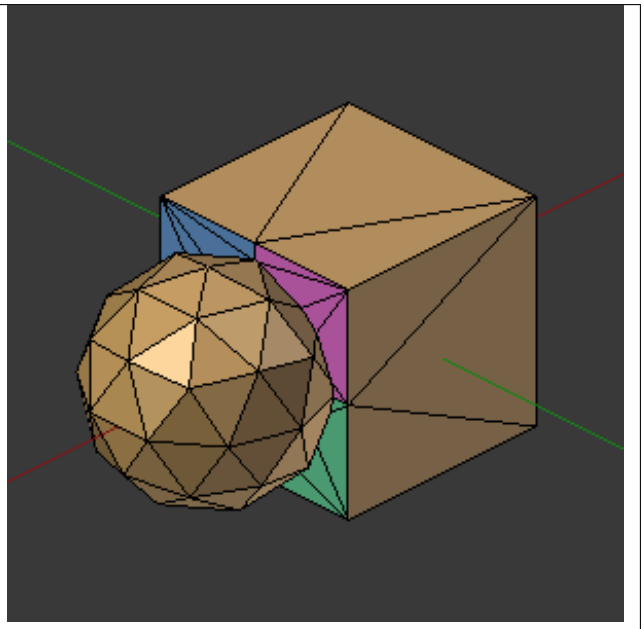


Fig. 2.795: Union: The first active Material of the Cube is added to the new topology.

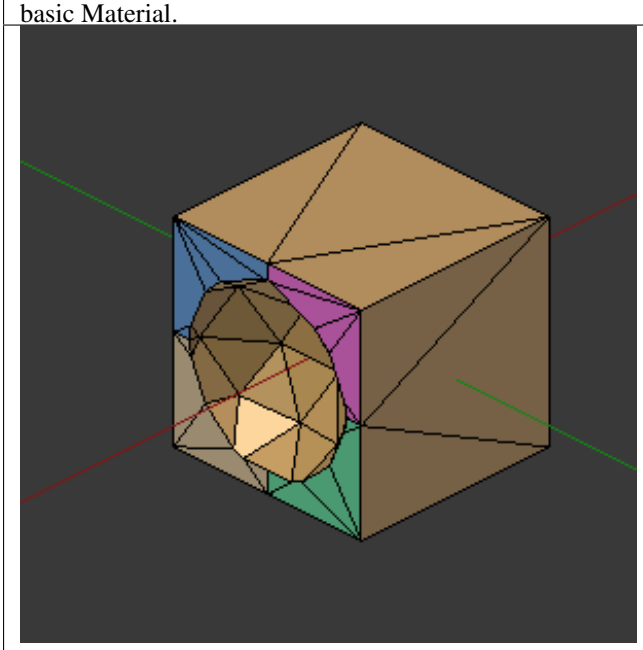


Fig. 2.796: Difference: The Icosphere was subtracted from the Cube.

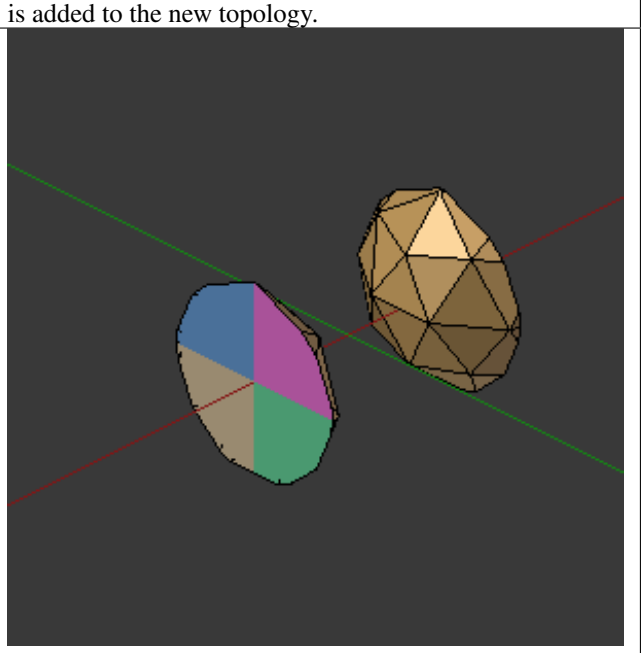


Fig. 2.797: Intersect: The resulting Mesh was copied and rotated 180.

The only exception is the difference operation when the normals of the target and modified mesh are inverted. In this case, Blender will project the textures in an inverted direction over the target using the center contact of the meshes as a pivot and the resulting mesh will have the modified mesh subtracted from the target. For complex target meshes in some particular cases, you may have to reassign materials to faces because Blender will use the possible projection, and this may result in a suboptimal texture assignment. You can see

this in the last example below.

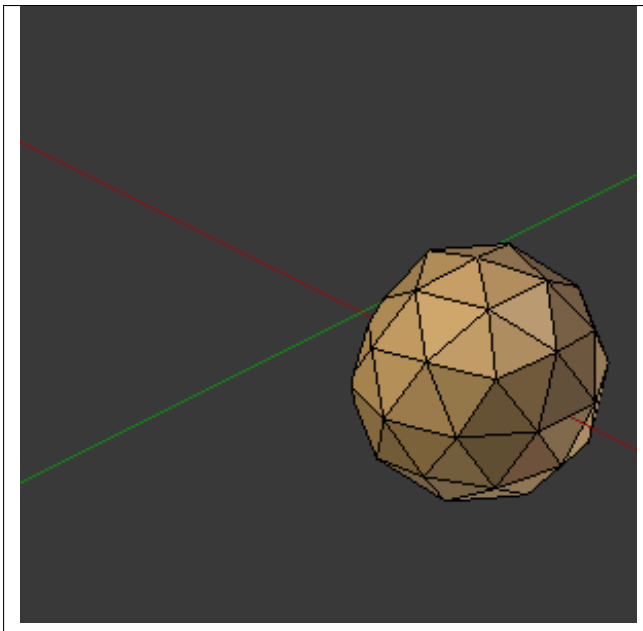


Fig. 2.798: Front of the target with the modified mesh materials.

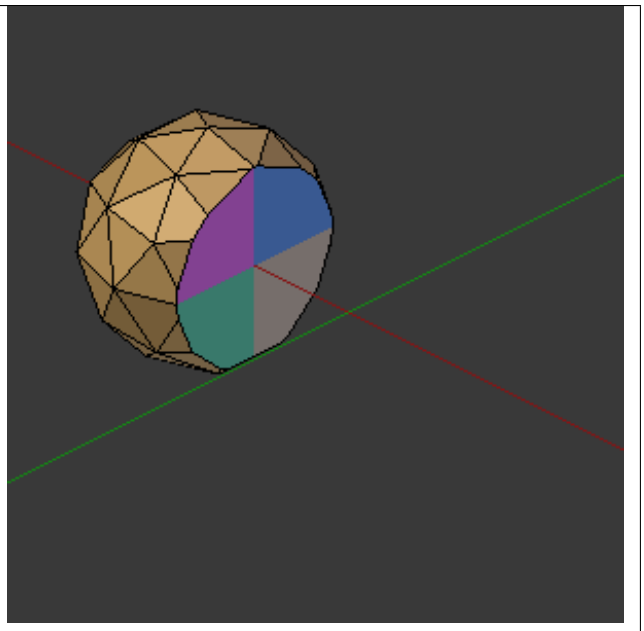


Fig. 2.799: Back of the target with the modified mesh materials.

UV Mappings

When you map UV Images to your target, Blender will add a map for each of the faces of the target. When you apply the Boolean modifier, Blender will follow the UV maps already assigned to the faces of the target topology that will be the result of the operation on the modified mesh. Blender will also use the same image mapped to the target faces in the modified mesh.

Warning: Depending on the way you have assigned textures to the faces during the UV unwrap, and the complexity of your meshes, the boolean operation may generate imperfect UVs for the new faces.

Below we have four Images, a UV sphere mapped with a test grid tinted blue and the other face tinted in purple, one face of the cube tinted in a light orange and the other faces using the normal test grid. The first image shows the operation at the start (difference), and on to the right of that shows the resulting mesh. And in the bottom row we show the unwrap in the Blender UV/Image Editor.

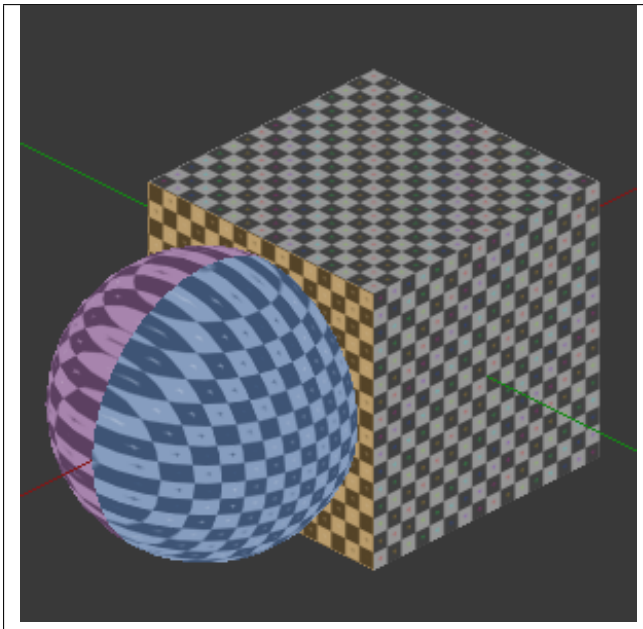


Fig. 2.800: A UV Sphere and a Cube with different UV Maps.

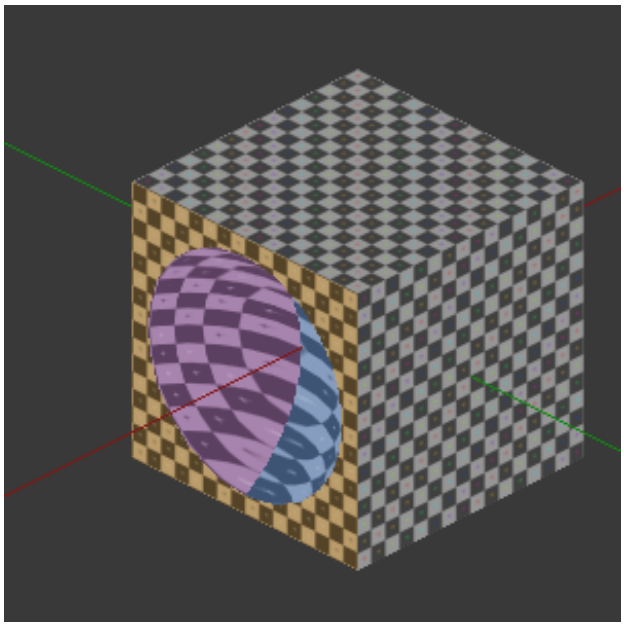


Fig. 2.801: Difference operation applied.

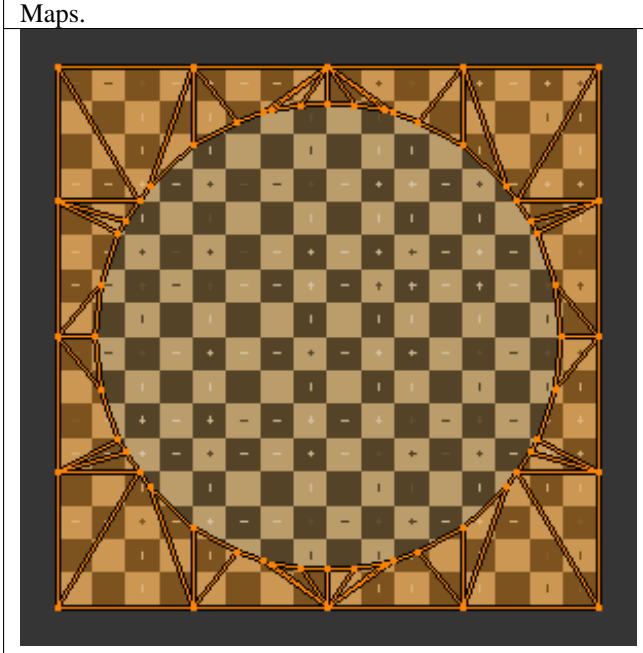


Fig. 2.802: Faces of the modified mesh mapped.

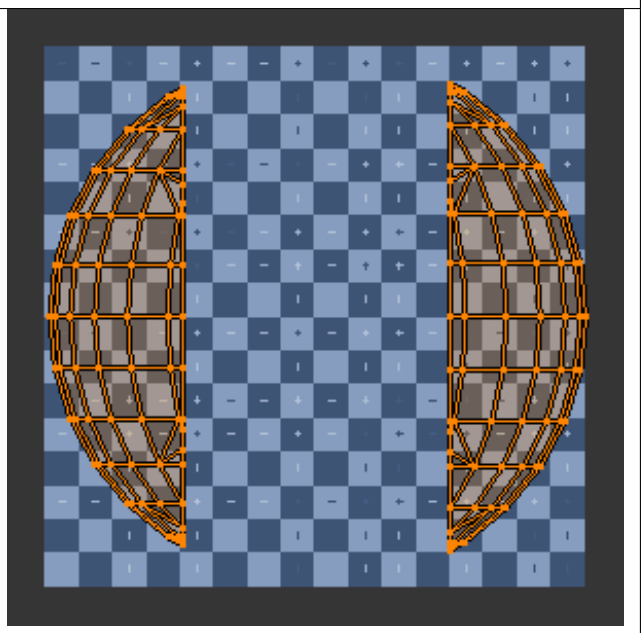


Fig. 2.803: New topology mapped and UV faces assigned.

Other Modifiers

The Boolean modifier calculation is performed using the target modified mesh topology and dimensions. Other modifiers added to the modified mesh are bypassed. This means that if a target is using another modifier, like Subdivision Surface, the resulting topology for the modified mesh will take into account the subdivision of the target; but for the modified mesh, the basic topology is used anyway (see examples).

If you add Subdivision Surface to the modified mesh with a Boolean modifier, Blender will visually add the subdivision for the modified mesh, but not for its calculations; it will only take into account its basic mesh topology. If you want to have a Subdivision Surface modifier added to the modified mesh, you have to apply the Subdivision Surface to the Boolean modified mesh before applying the Boolean operation.

The Boolean modifier can be added together with other modifiers in the modified mesh, but depending on the modifier, the calculations cannot be done and/or the modifier cannot execute. When the modifier cannot execute, it will show the message `"Cannot execute boolean operation"`, and when the modifier cannot be applied to the mesh, Blender will show the message `"Modifier is disabled,Skipping Apply."`. In this case, you either have to remove some modifiers or apply the necessary ones.

The most common case is when you add or copy a Boolean modifier to use the modified mesh in conjunction with another target later; Blender will place the warning in the subsequent Boolean modifiers in the stack depending on the operation, because you may be creating concurrent Boolean operations for the same modified mesh, which in most cases is impossible to execute depending on the chosen target. In this case, you can apply the first Boolean modifier of the stack for the target and then use the other Boolean modifier(s) in the stack for subsequent operations.

Also, if some other modifiers are placed above this modifier and you click on Apply, Blender will warn you with the message `"Applied Modifier was not first,results may not be as expected"`. The best usage scenario for this modifier is to prepare your modified mesh and target to work with the Boolean modifier.

When the Boolean modifier is the first of the stack and is applied, the other Modifiers will act over the resulting meshes using the resulting topology and will remain in the modifiers stack.

Below are two images: one with the Subdivision Surface modifier added to the target, and another with the resulting topology.

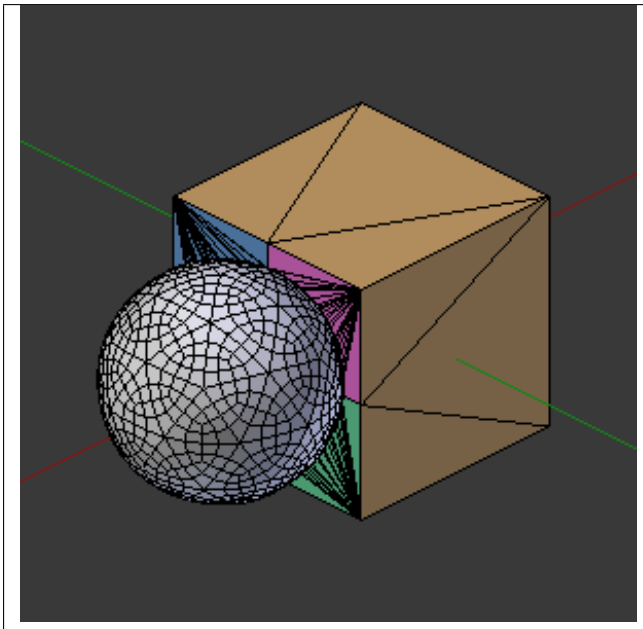


Fig. 2.804: Modifier with Subdivision Surface Target.

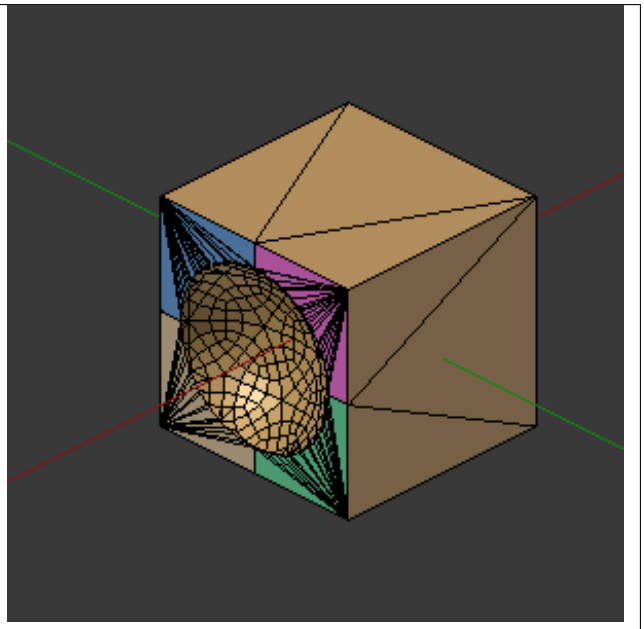


Fig. 2.805: The Resulting Topology.

As you can see, the added (not applied) Subdivision Surface modifier to the target was taken into consideration. The topology of the Icosphere with (Level 2) subdivision was completely transferred to the modified mesh.

Tip: The target topology determines the resulting topology

The target topology determines the results of the Boolean modifier operation. It means that any modifier added to the target which modifies its topology will affect the resulting mesh of the operation.

Concurrent Operations

For the modified meshes, you can only apply one operation at a time, but you can use the same target for other modified meshes and use modified meshes as a target for other meshes as well. Also, you can copy or add the same modifier to the modifiers stack as many times as you wish to suit the number of operations you need, but be aware that if you choose concurrent targets which are, at the same time, modified meshes pointing to each other, you can cause Blender to crash with closed loops!

Hints

Be aware that other modifiers and their stack position could cause this modifier to fail in certain circumstances.

Tip: The best way to work with this modifier when you need to make lots of sequential operations of the same modifier is to define the target at the time you need to apply the changes to the topology.

Face Normals

When using the Boolean modifier, Blender will use the face normal directions to calculate the three Boolean operations. The direction of the normals will define the result of the three available operations. When one of the participants has inverted normals, you are in fact multiplying the operation by -1 and inverting the calculation order. You can, at any time, select your modified mesh, enter Edit Mode and flip the normals to change the behavior of the Boolean modifier. See Tips: Fixing Mixed Normals below.

Blender also cannot perform any optimal Boolean operation when one or more of the mesh Normals of the participants that are touching has outwards/inwards normals mixed.

This means you can use the normals of the meshes pointed completely towards the inside or outside of your participants in the operation, but you cannot mix normals pointed inwards and outwards for the faces of the topology used for calculations. In this case, Blender will enable the modifier and you may apply the modifier, but with bad to no effects. We made some examples with a cube and an icosphere showing the results.

In the images below, all face normals are pointing outwards (Normal meshes).

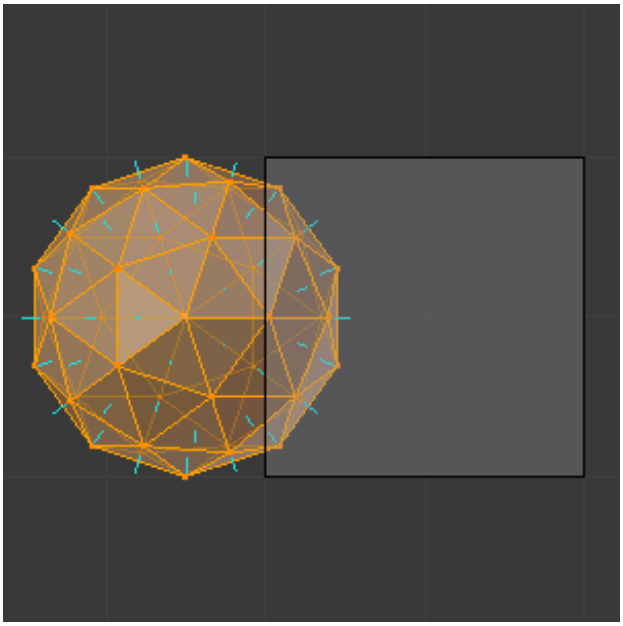


Fig. 2.806: Faces with normals pointing outwards.

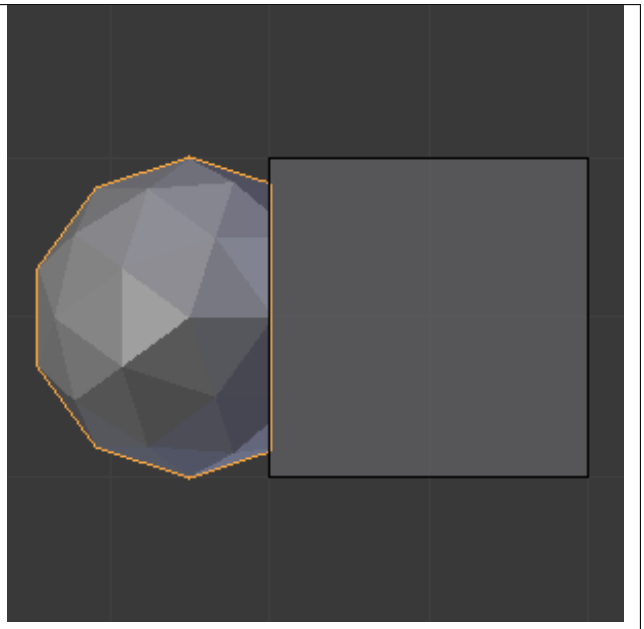


Fig. 2.807: Normal Boolean modifier operation (Difference operation).

In the images below, all face normals are inverted and using the intersection operation

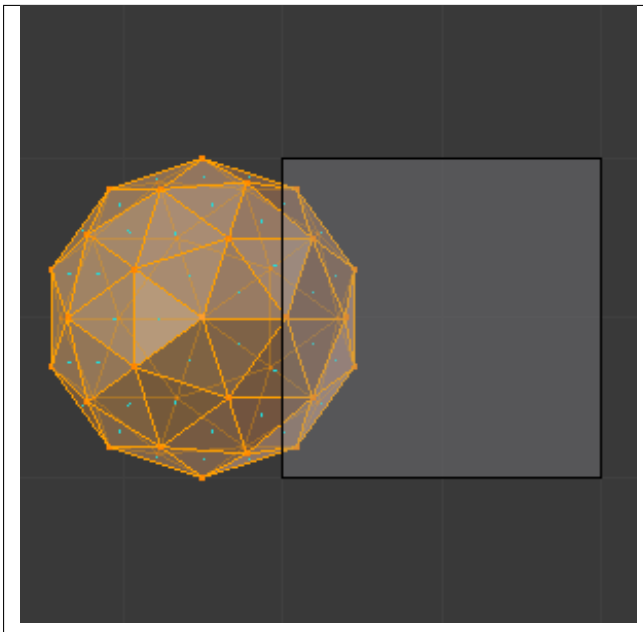


Fig. 2.808: Boolean Operation with inverted normals.

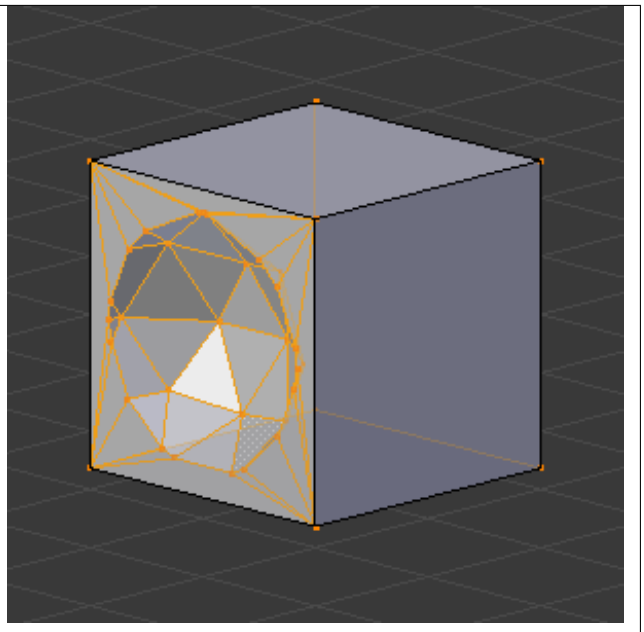


Fig. 2.809: Normal Boolean modifier operation.

Now, let us see what happens when the normal directions are mixed for one of the participants in the Boolean modifier operation. The images below show face normals mixed, pointed to different directions and the resulting operation, you can see that the modifier has bad effects when applied, leaving faces opened:

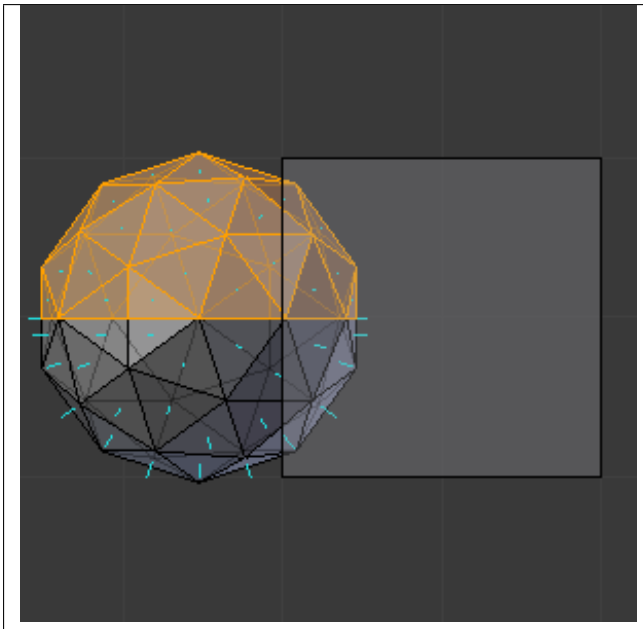


Fig. 2.810: Face normals mixed, pointed to different directions.

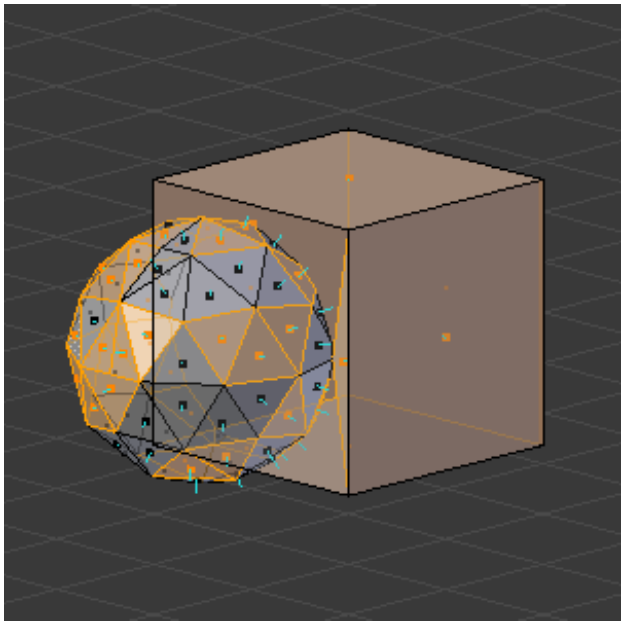


Fig. 2.811: Resulting operation leaves faces opened.

As you can see, the normal directions can be pointing to any of the Mesh sides, but cannot be mixed in opposite directions for the faces of the participants. The Library cannot determine properly what is positive and negative for the operation, so the results will be bad or you will have no effect when using the Boolean modifier operation.

A quick way to fix this is to use Blender's *Recalculate Normals* operation in Edit Mode.

If you still have some ugly black gouges you will have to *Manually Flip the Normals*.

Empty or Duplicated Faces

This modifier does not work when the modified and/or the target mesh uses empty faces in the topology used for calculations. If the modifier faces a situation where you have empty faces mixed with normal faces, the modifier will try, as much as possible, to connect the faces and apply the operation. For situations where you have two concurrent faces at the same position, the modifier will operate on the target mesh using both faces, but the resulting normals will get messed. To avoid duplicated faces, you can remove doubles for the vertices before recalculating the normals outside or inside. The button for remove doubles is

located in the *Mesh Tools* Panel in the 3D View, while in Edit Mode.

The best usage scenario for this modifier is when you have clean meshes with faces pointing clearly to a direction (inwards/outwards)

Below we show an example of meshes with open faces mixed with normal faces being used to create a new topology. In this example, a difference between the cube and the icosphere is applied, but Blender connected a copy of the icosphere to the Cube mesh, trying to apply what was possible.

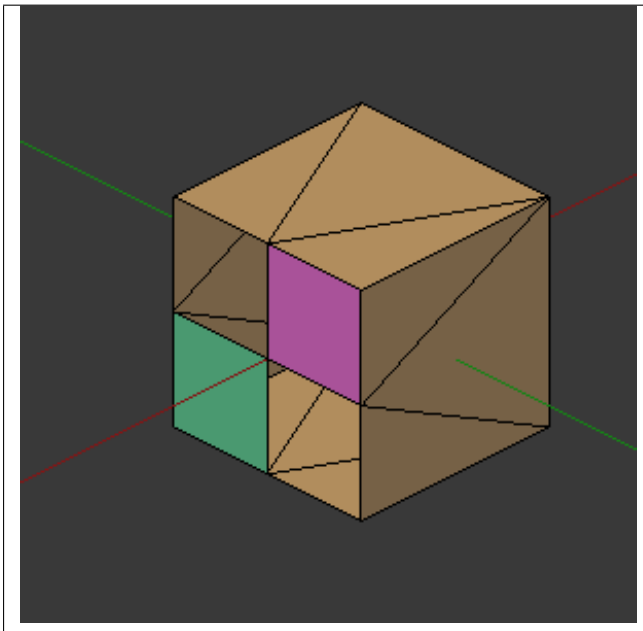


Fig. 2.812: Mesh with two empty faces mixed with normal faces.

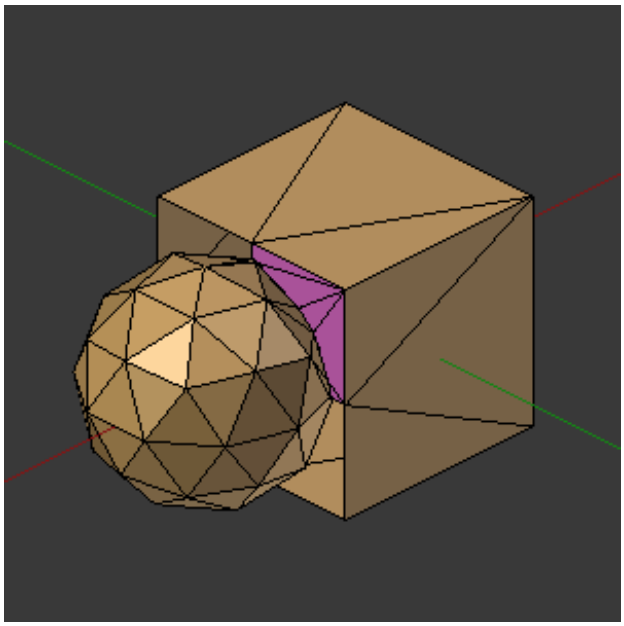


Fig. 2.813: Result of a difference operation applied.

Open Volumes

The Boolean Modifier permits you to use open meshes or non-closed volumes (not open faces).

When using open meshes or non-closed volumes, the Boolean modifier will not perform any operation in faces that do not create a new topology filled with faces using the faces of the target.

In the images below, is the resulting operation when using two non-closed volumes with faces forming a new topology.

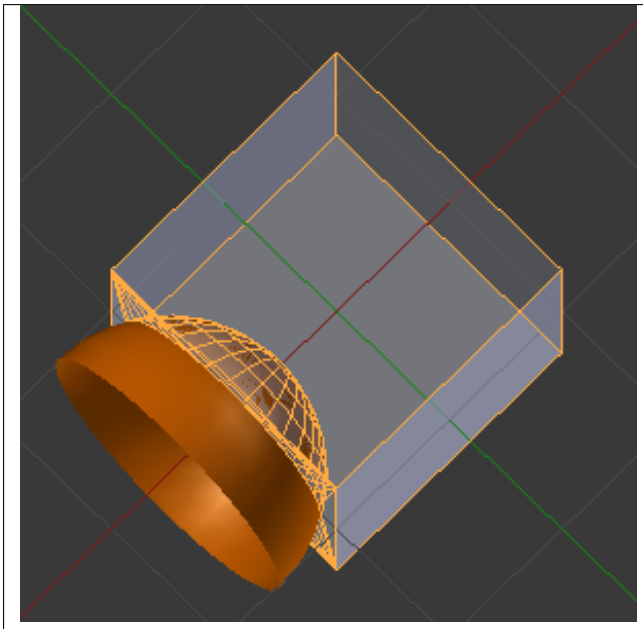


Fig. 2.814: Non-closed volumes forming a new topology.

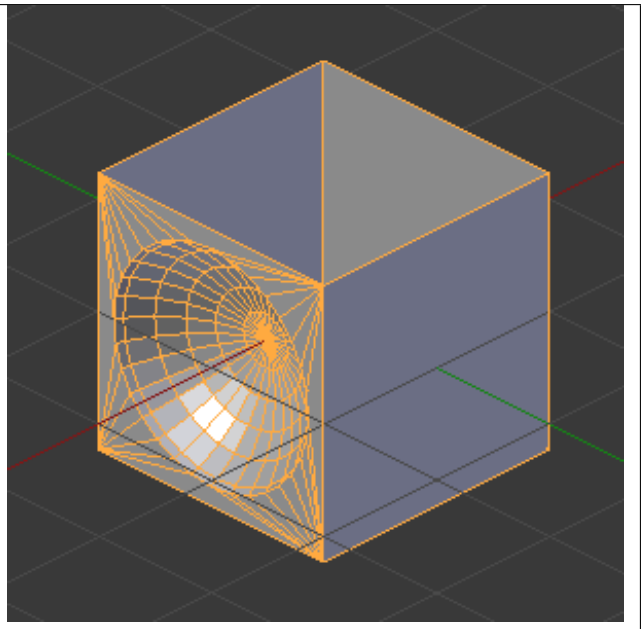


Fig. 2.815: Resulting operation using two open volumes.

Now, let us see what happens when we use meshes that are partially open, incomplete, or meshes that are not forming a new topology.

As you can see in the images below the faces of one participant in the Boolean operation gives incomplete information to the modifier. The resulting edges get messy and there is not enough information to create faces for the resulting Mesh. This example uses a smooth shaded UV-sphere cut in half. As explained before, the shading (smooth/flat) does not affect the calculations of the modifier.

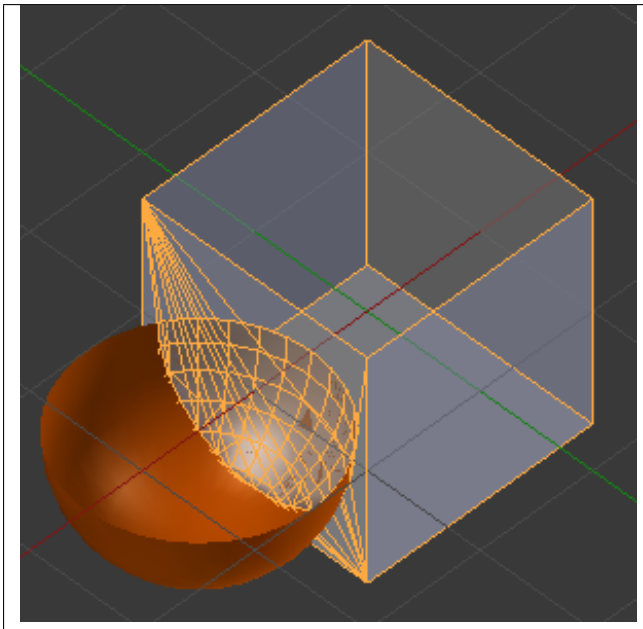


Fig. 2.816: Open volumes that are not forming a new topology.

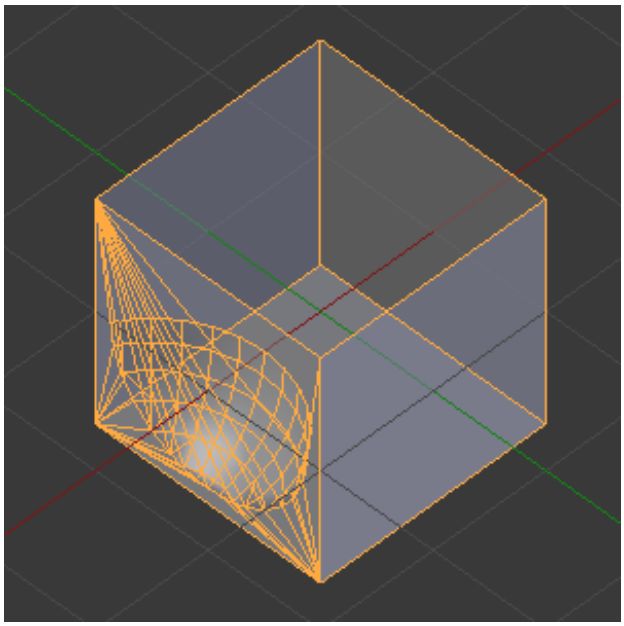


Fig. 2.817: Resulting Operation of image on the left

Build Modifier

The Build modifier causes the faces of the mesh object to appear one after the other over time.

By default, faces appear in the order in which they are stored in memory (by default, the order of creation). The face/vertex order can be altered in Edit Mode by selecting *Sort Mesh Elements* from the *Search Menu* `Spacebar`

Note: When using Blender Render, if the material of the mesh is a halo rather than a standard one, then the vertices of the mesh, not the faces, appear one after another.

Options

Fig. 2.818: Build modifier in action.

Start The start frame of the building process.

Length The number of frames over which to rebuild the object.

Randomize Randomizes the order in which the faces are built.

Seed The random seed. Changing this value gives a different "random" order when *"Randomize"* is checked – this order is always the same for a given seed/object set.

Decimate Modifier

The Decimate modifier allows you to reduce the vertex/face count of a mesh with minimal shape changes.

This is not usually used on meshes which have been created by modeling carefully and economically (where all vertices and faces are necessary to correctly define the shape). But if the mesh is the result of complex modeling, sculpting and/or applied Subdivision Surface/Multires modifiers, the Decimate modifier can be used to reduce the polygon count for a performance increase, or simply remove unnecessary vertices and edges.

The Decimate modifier is a quick and easy way of reducing the polygon count of a mesh non-destructively. This modifier demonstrates the advantages of a mesh modifier system because it shows how an operation which is normally permanent and destroys original mesh data, can be done interactively and safely using a modifier.

Unlike the majority of existing modifiers, the Decimate modifier does not allow you to visualize your changes in Edit Mode.

Options

Fig. 2.819: decimate modifier.

Decimate Type

Collapse Merge vertices together progressively, taking the shape of the mesh into account.

Ratio The ratio of faces to keep after decimation.

- On 1.0: the mesh is unchanged.
- On 0.5: edges have been collapsed such that half the number of faces remain (See *Note* below).
- On 0.0: all faces have been removed.

Note: Although the *Ratio* is directly proportional to the number of remaining faces, triangles are used when calculating the ratio.

This means that if your mesh contains quads, the number of remaining faces will be larger than expected, because quads remain unchanged if their edges are not collapsed.

This is only true if the *Triangulate* option is disabled.

Un-Subdivide Can be thought of as the reverse of subdivide. Attempts to remove edges that were the result of a subdivide operation. If additional editing has been done after the subdivide operation, the results may be unexpected.

Iterations The number of times to perform the un-subdivide operation. Two iterations is the same as one subdivide operation, so you will usually want to use even numbers.

Planar Dissolve geometry on the same plane.

Angle Limit Dissolve geometry which form angles lower than this setting.

All Boundaries When enabled, all vertices along the boundaries of faces are dissolved. This can give nicer results when using a high *Angle Limit*.

Delimit Prevent dissolving geometry in certain places.

Normal Does not dissolve edges on the borders of areas where the face normals are reversed.

Material Does not dissolve edges on the borders of where different materials are assigned.

Seam Does not dissolve edges marked as seams.

Face Count This field shows the number of remaining faces as a result of applying the Decimate modifier.

Edge Split Modifier

The Edge Split modifier splits edges within a mesh. The edges to split can be determined from the edge angle (i.e. angle between faces forming this edge), and/or edges marked as sharp.

Splitting an edge affects vertex normal generation at that edge, making the edge appear sharp. Hence, this modifier can be used to achieve the same effect as *Auto Smooth*, making edges appear sharp when their angle is above a certain threshold. It can also be used for manual control of the smoothing process, where the user defines which edges should appear smooth or sharp (see *Mesh Smoothing* for other ways to do this). If desired, both modes can be active at once.

The output of the Edge Split modifier is available to export scripts, making it quite useful for creators of game content.

Options

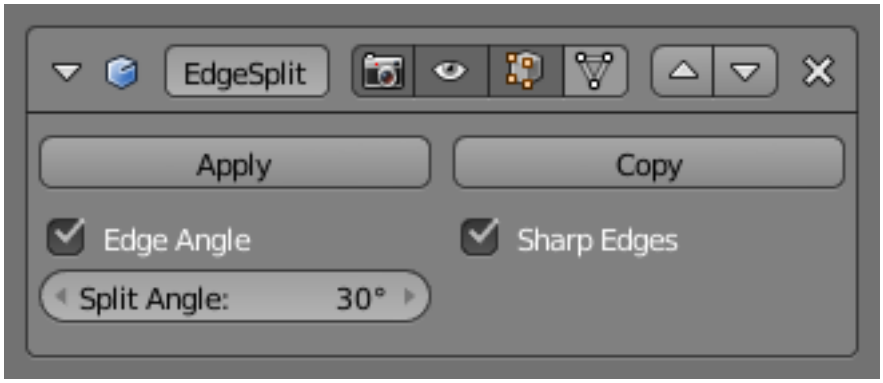


Fig. 2.820: Edge Split modifier.

Edge Angle When enabled, edges will be split if the angle between its two adjacent faces is greater than the *Split Angle*.

Split Angle On 0: all edges are split. On 180: no edges are split.

Sharp Edges When enabled, edges will be split if they were marked as sharp using *Edge Specials* → *Mark Sharp* (Menu shortcut: `Ctrl-E` in Edit Mode).

Note: *Non-manifold* edges (edges shared by more than two faces) will always be split.

Examples

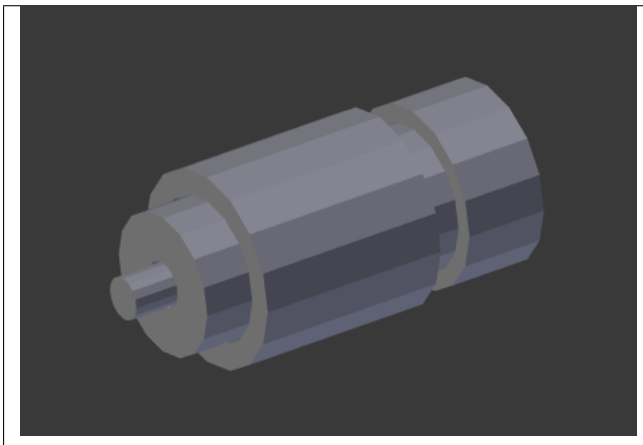


Fig. 2.821: Flat Shading.

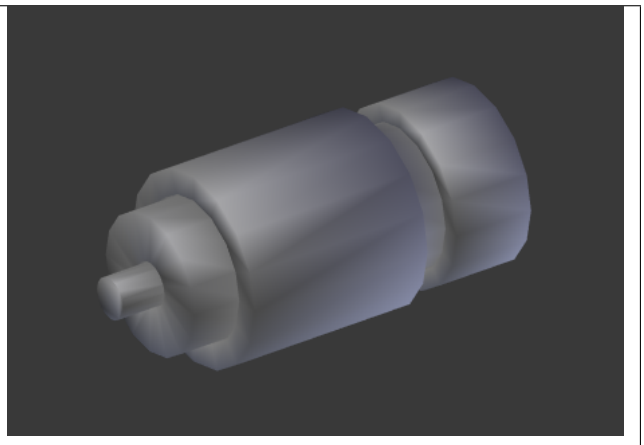


Fig. 2.822: Smooth Shading.

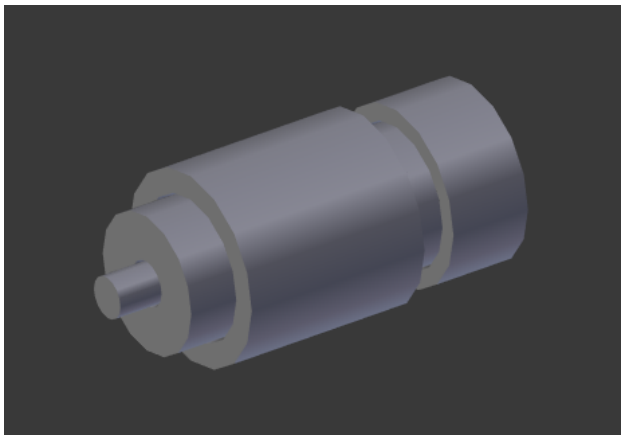


Fig. 2.823: Smooth Shading with Edge Split.

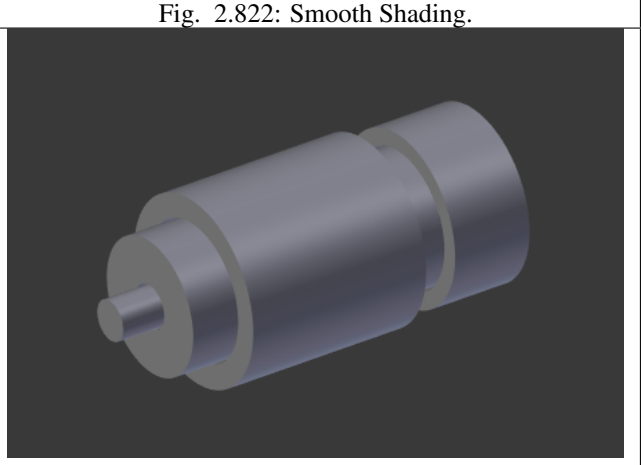


Fig. 2.824: Smooth Shading with Edge Split and Subdivision Surface.

Note: Splitting edges can also be performed manually in Edit Mode using: *Edge Specials* → *Edge Split* (Menu shortcut: `Ctrl-E`).

Mask Modifier

The Mask modifier allows vertices of an object to be hidden dynamically based on vertex groups.

Options

Mode The Mask modifier can hide parts of a mesh based on two different modes, selectable from this select menu.

Vertex Group When the *Vertex Group* option is selected, all vertices belonging to the chosen Vertex Group (with a weight above zero) will be visible, and all other vertices will be hidden.

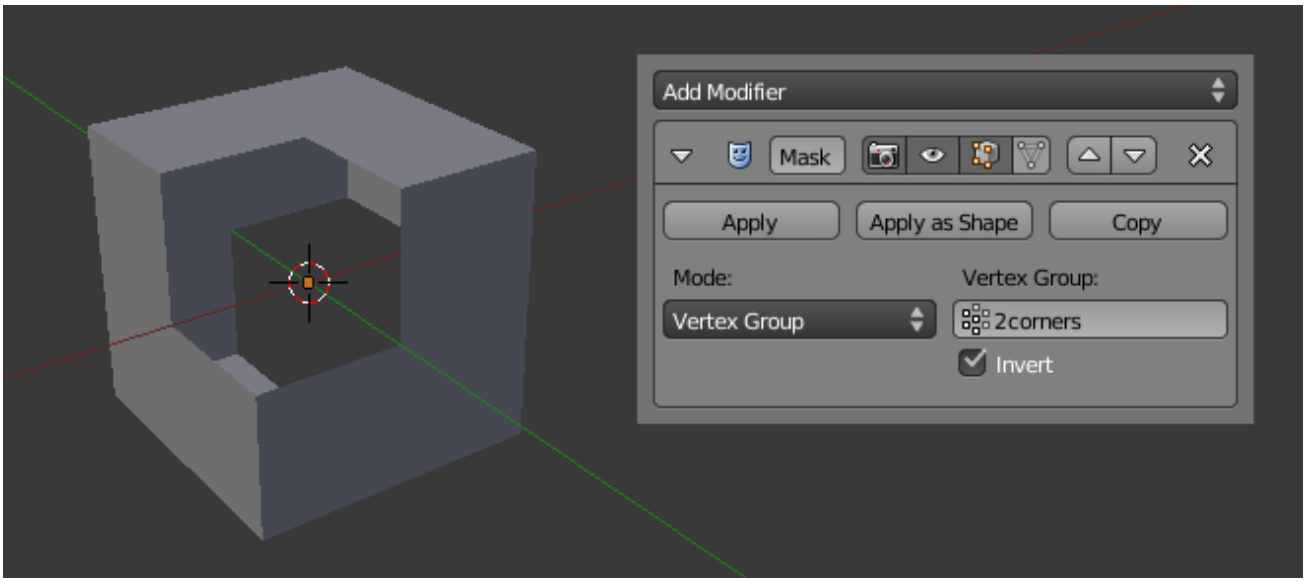


Fig. 2.825: Vertex Group.

Armature When in Pose Mode, vertices belonging to the Vertex Group associated with the active bone (same names) will be visible. Vertices not in that group will be hidden.

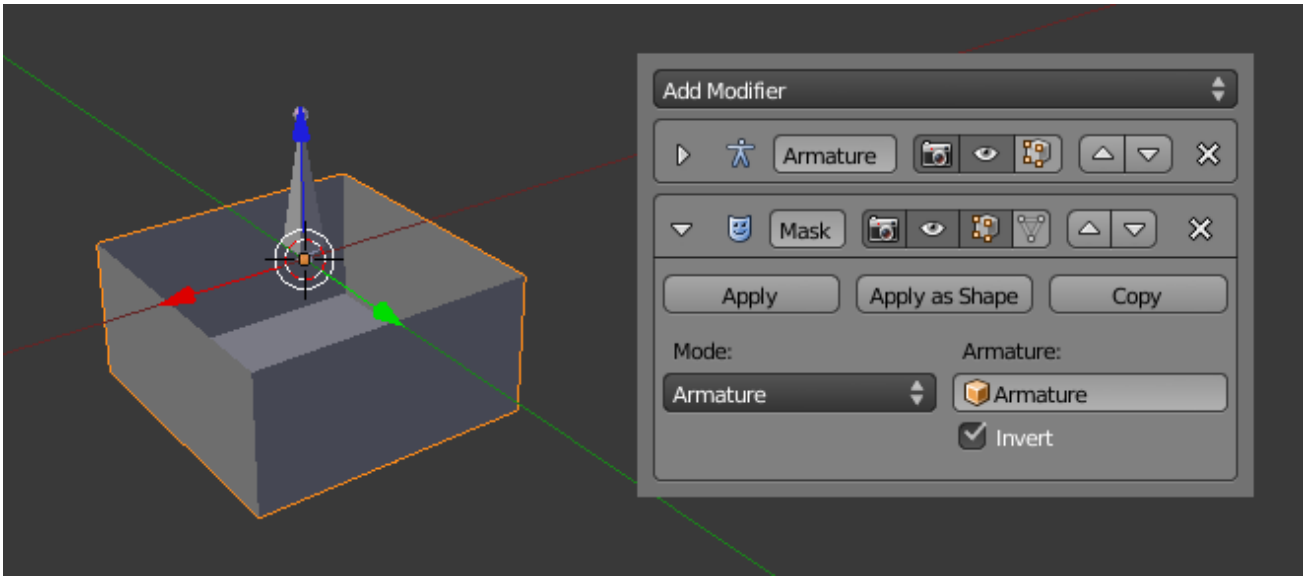


Fig. 2.826: Armature.

Inverse Normally, vertices belonging to the selected Vertex Group (or group associated with the active pose-bone) will be shown. The *Invert* toggle allows you to reverse

this behavior, instead showing all vertices which do not belong to the Vertex Group, and hiding those that do.

Mirror Modifier

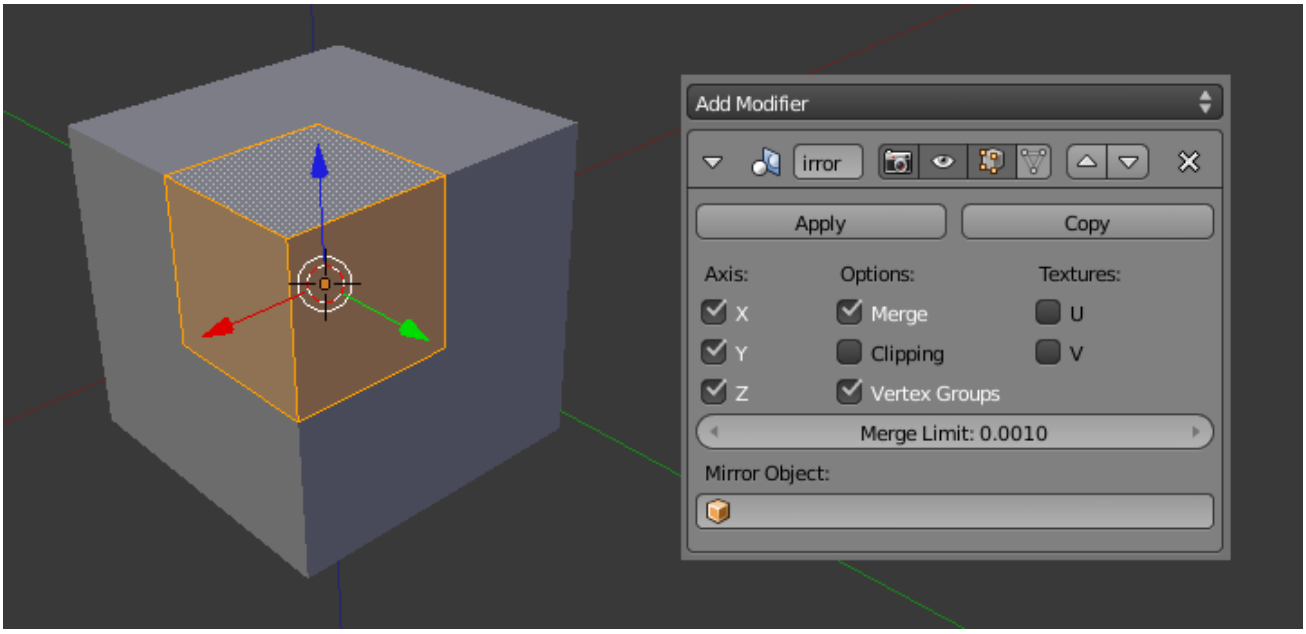


Fig. 2.827: The corner of a cube mirrored across three axes to form... well... a cube.

The Mirror modifier mirrors a mesh along its *local* X-, Y- and/or Z-Axes, across the object's center (the mirror plane is then defined by the two other axes).

It can also use another object as the mirror center, then use that object's local axes instead of its own.

Options

Axis The X-, Y-, Z-axis along which to mirror (i.e. the axis perpendicular to the mirror plane of symmetry).

To understand how the axis applies to the mirror direction, if you were to mirror on the X axis, the positive X values of the original mesh would become the negative X values on the mirrored side.

You can select more than one of these axes. And will then get more mirrored copies. With one axis you get a single mirror, with two axes four mirrors, and with all three axes eight mirrors.

Options

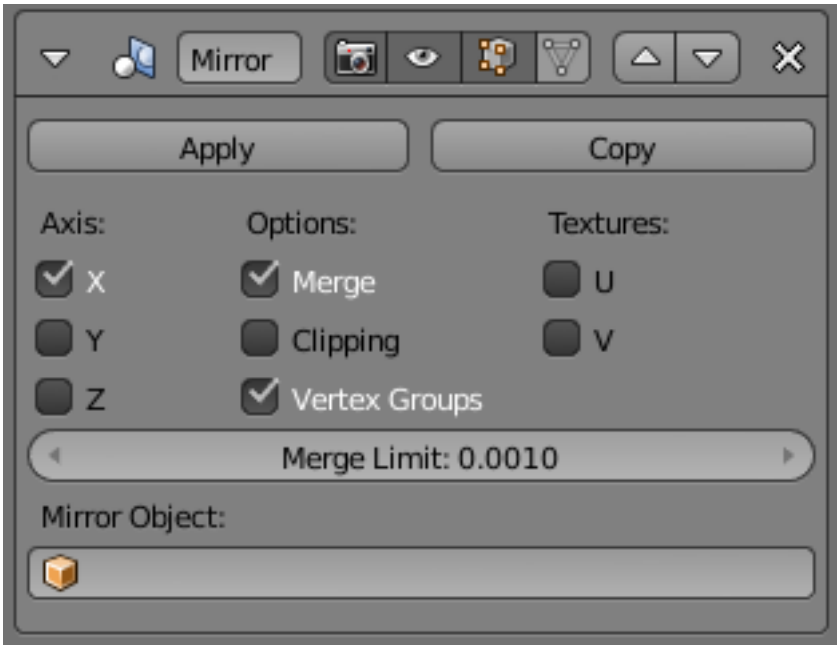


Fig. 2.828: Mirror modifier.

Merge Where a vertex is in the same place (within the *Merge Limit* distance) as its mirror it will be merged with the mirrored vertex.

Clipping Prevents vertices from moving through the mirror plane(s) while the user is transforming them in Edit Mode.

If *Clipping* is enabled but vertices are beyond the mirror plane and outside of the *Merge Limit*, the vertices will not be merged. But as soon as the vertices are within *Merge Limit* they are snapped together and cannot be moved beyond the mirror plane.

Note: Vertices on the mirror plane will be unable to move away from the mirror plane as long as *Clipping* is enabled. You must disable *Clipping* to be able to move the vertices along the mirror axis again.

Vertex Groups When enabled, the Mirror modifier will try to mirror existing vertex groups.

A very nice feature, but one that has very specific prerequisites:

- The vertex groups you want to mirror must be named following the usual left/right pattern (i.e. suffixed by something like ".R", ".right", ".L", etc).
- The mirror side vertex group must already

exist (it will not be created automatically). It must also be completely empty (no vertices assigned to it).

Textures The U and V options allows you to mirror the UV texture coordinates across the middle of the image.

E.g. if you have a vertex with UV coordinates of (0.3, 0.9), its mirror copy will have UV coordinates of (0.7, 0.1).

Merge Limit The maximum distance between a vertex and its mirror copy before they are merged together. In other words, a vertex may be half this distance away from the mirror plane before it snaps to it.

Mirror Object An *Object Selector* to select an object (usually an empty), to be used as the reference for the mirror process: its center and axes will drive the plane(s) of symmetry. You can of course animate its position/rotation to animate the mirror effect.

Hints

Many modeling tasks involve creating objects that are symmetrical. However, there used to be no quick way to model both halves of an object without using one of the workarounds that have been discovered by clever Blender artists over the years. A common technique was to model one half of an object and use `Alt-D` to create a linked duplicate which can then be scaled on one axis by -1 to produce a perfect mirror-image copy which updates in real time as you edit.

The Mirror modifier offers a simpler way to do this. Once your modeling is completed you can either click *Apply* to make a real version of your mesh or leave it as is for future editing.

Using the Mirror Modifier with a Subdivision Surface Modifier

When using the Mirror modifier along with a *Subdivision Surface* modifier, the order in which the modifiers are placed is important.

The above image shows the Subdivision Surface modifier placed before the Mirror one; as you can see the effect of this is that the mesh is split down the center line of

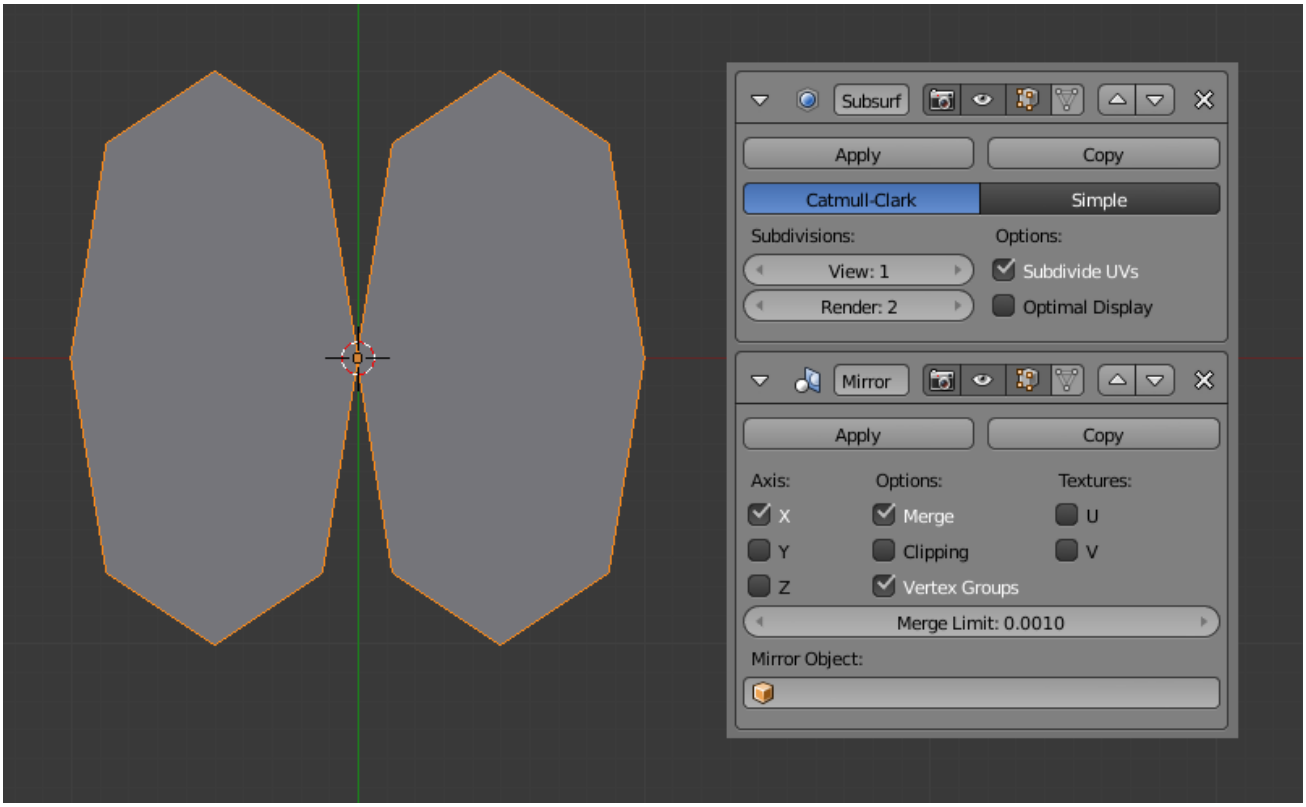


Fig. 2.829: Subdivision Surface modifier before Mirror modifier.

the mirror effect. This is because the Subdivision calculation moves vertices away from the mirror plane, too far away from the *Merge Limit*.

The above image shows the Mirror modifier placed before the Subdivision Surface modifier. In this order, the mirror calculation is done and the vertices are merged together. Only after that does the Subdivision Surface modifier move any vertices.

Accurately Positioning the Mirror Plane

To apply a Mirror modifier, it is common to have to move the object's center onto the edge or face that is to be the axis for mirroring. This can be tricky when attempted visually.

A good technique to achieve an exact position is to select the edge, then use `Shift-S` and choosing *Cursor to Selection*. This will position the 3D Cursor in the center of the edge. Finally, press `Ctrl-Alt-Shift-C` for the *Set Origin* menu, then select *Origin to 3D Cursor*. This will move the object's center (and

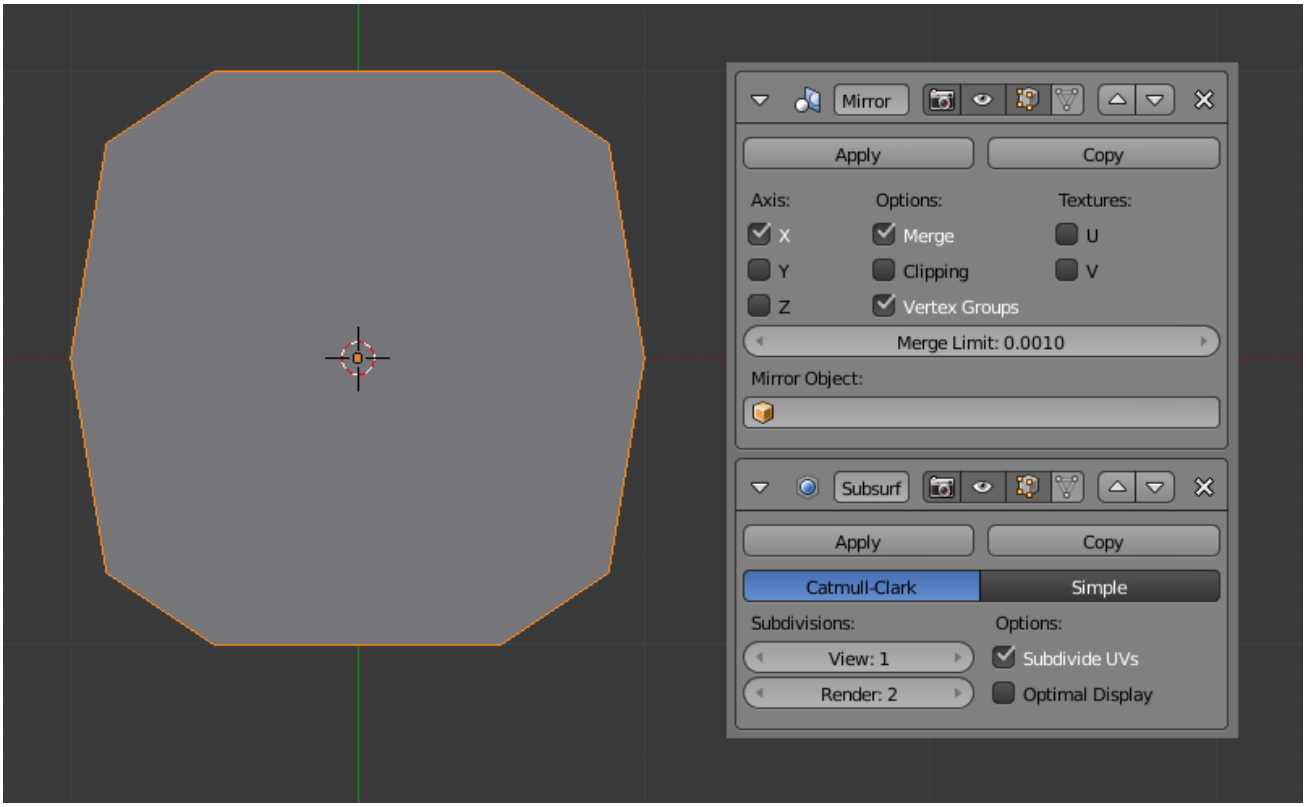


Fig. 2.830: Mirror modifier before Subdivision Surface modifier.

thus, the mirror plane) to where the 3D cursor is located, and the mirroring will be exact.

An alternative is to use an Empty as a *Mirror Object* that you move to the correct position.

Multiresolution Modifier

The Multiresolution modifier (often shortened to *Multires*) gives you the ability to subdivide a mesh similarly to the *Subdivision Surface Modifier*, but also allows you to edit the new subdivision levels in sculpt mode.

Note: The *Multiresolution Modifier* is the only modifier that cannot be repositioned in the stack if it means that there will be geometry or other object data created or removed before it (i.e. all *Generate*, some *Modify* and some *Simulate* modifiers cannot come before the Multiresolution modifier.)

Options

Catmull-Clark / Simple Set the type of subdivision.

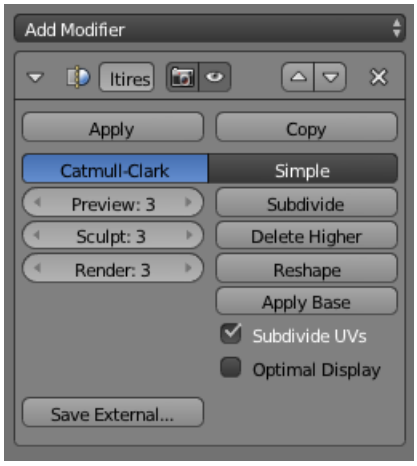


Fig. 2.831: Multires modifier.

Simple Maintains the current shape, and simply subdivides edges.

Catmull-Clark Creates a smooth surface, usually smaller than the original, using the standard Catmull-Clark subdivision surface algorithm.

Preview Set the level of subdivisions to show in the 3D View.

Sculpt Set the number of subdivisions to use in Sculpt Mode.

Render Set the number of subdivisions to show when rendering.

Subdivide Add another level of subdivision.

Delete Higher Deletes all subdivision levels that are higher than the current one.

Reshape Copies vertex coordinates from another mesh. To use, first select a different mesh object with matching topology and vertex indexes, then `Shift` select the object you wish to copy vertex coordinates to and click *Reshape*.

Apply Base Modifies the original unsubdivided mesh to match the form of the subdivided mesh.

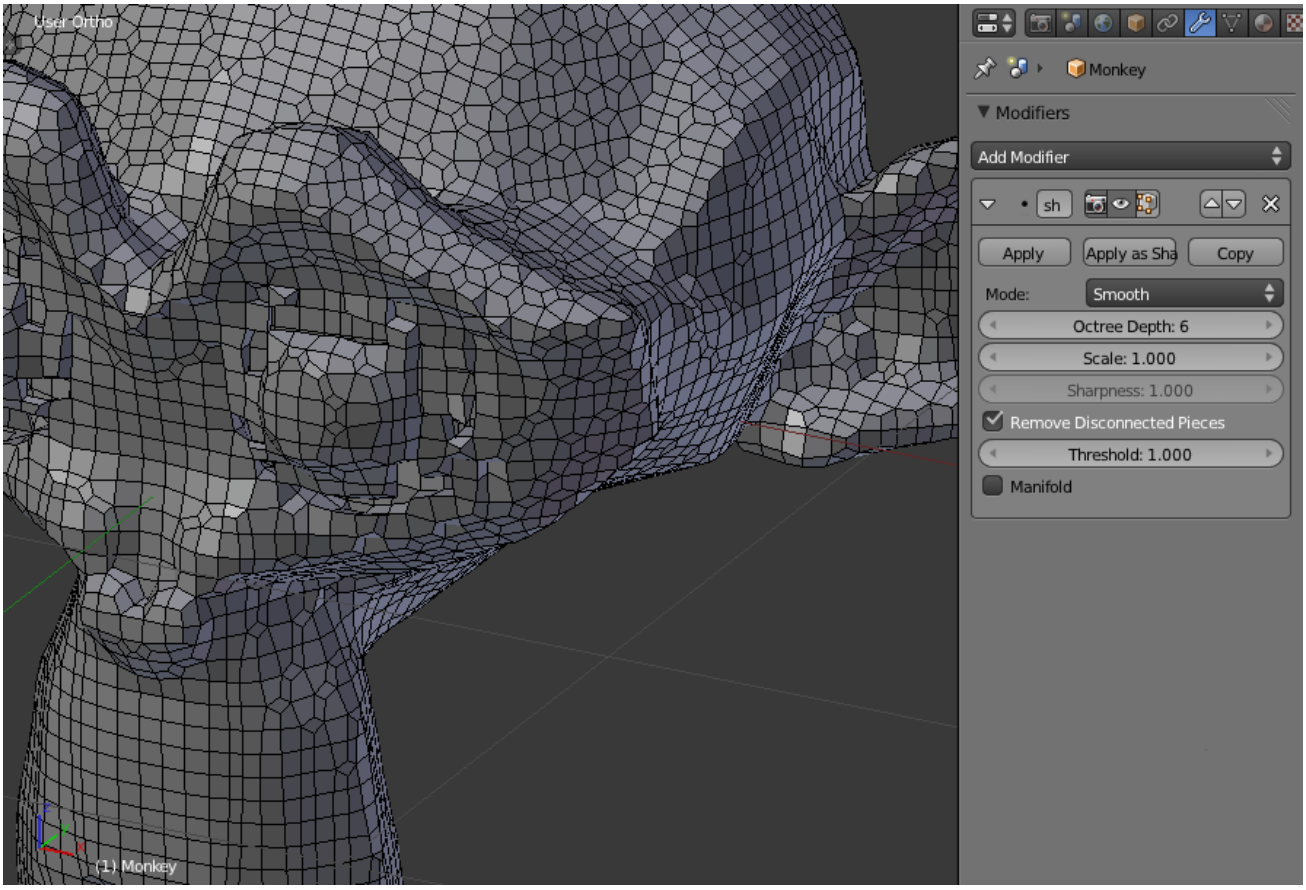
Subdivide UVs When enabled, the UV maps will also be subdivided. (i.e. Blender will add "virtual" coordinates for all sub-faces created by this modifier).

Optimal Display When drawing the wireframe of this object, the wires of the new subdivided edges will be skipped (only draws the edges of the original geometry).

Save External Saves displacements to an external .btx file.

Remesh Modifier

The Remesh modifier is a tool for generating new mesh topology. The output follows the surface curvature of the input, but its topology contains only quads.



Options

Mode There are three basic modes available in the remesh modifier: Blocks, Smooth and Sharp.

The output topology is almost identical between the three modes; what changes is the smoothing.

Blocks There is no smoothing at all.

Smooth Output a smooth surface.

Sharp Similar to *Smooth*, but preserves sharp edges and corners. In the above image, the circular bottom of the cone and the top point of the cone are more accurately reproduced in Sharp mode.

Octree Depth The Octree Depth sets the resolution of the output. Low values will gener-

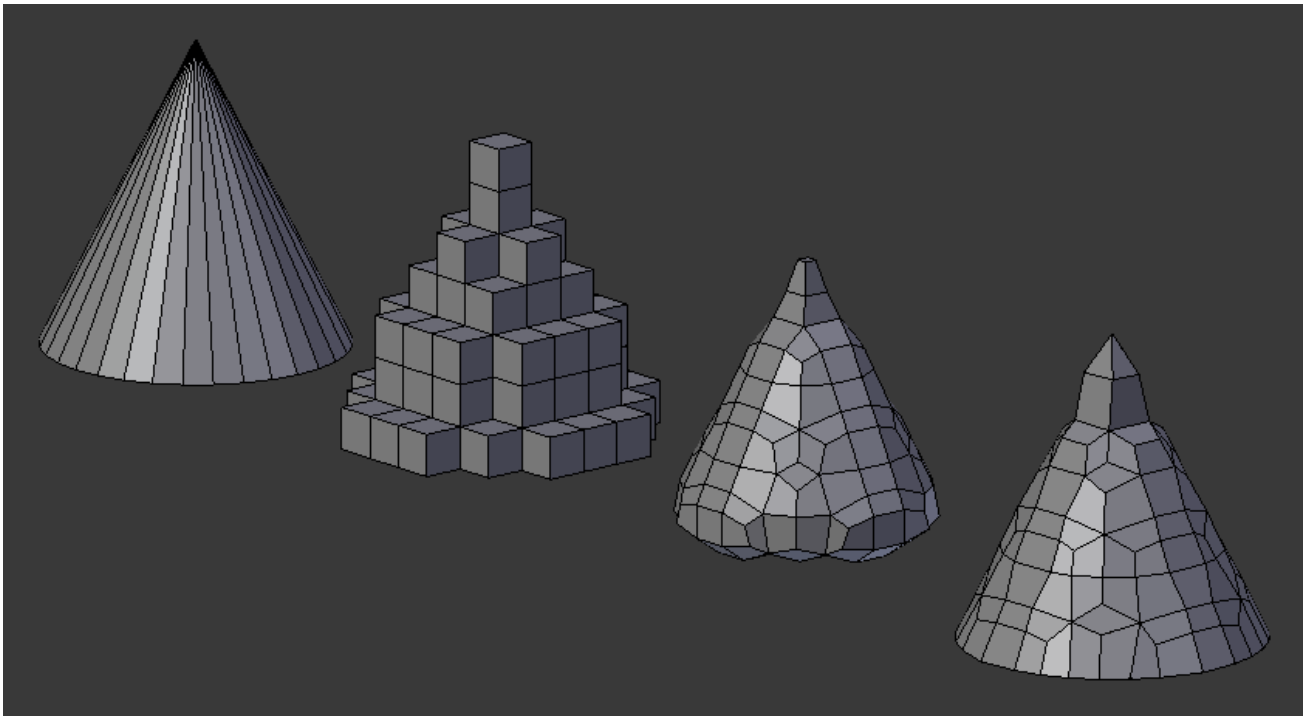


Fig. 2.832: This example shows a cone with each of the different remesh modes. From left to right: original cone, Blocks, Smooth, and Sharp

ate larger faces relative to the input, higher values will generate a denser output.

Scale The result can be tweaked further by setting the Scale; lower values effectively decrease the output resolution.

Sharpness Shown when using the *Sharp Mode* - Higher values produce edges more similar to the input, while lower values filter out noise.

Smooth Shading Output faces with smooth shading rather than flat shading. The smooth/flat shading of the input faces is not preserved.

Remove Disconnected Pieces Filter out small disconnected pieces of the output.

Threshold Use this to control how small a disconnected component must be to be removed.

Usage

In the modifier panel, add a Remesh modifier. The input mesh should have some thickness to it; if the input is completely flat, add a *solidify modifier* above the remesh modifier.

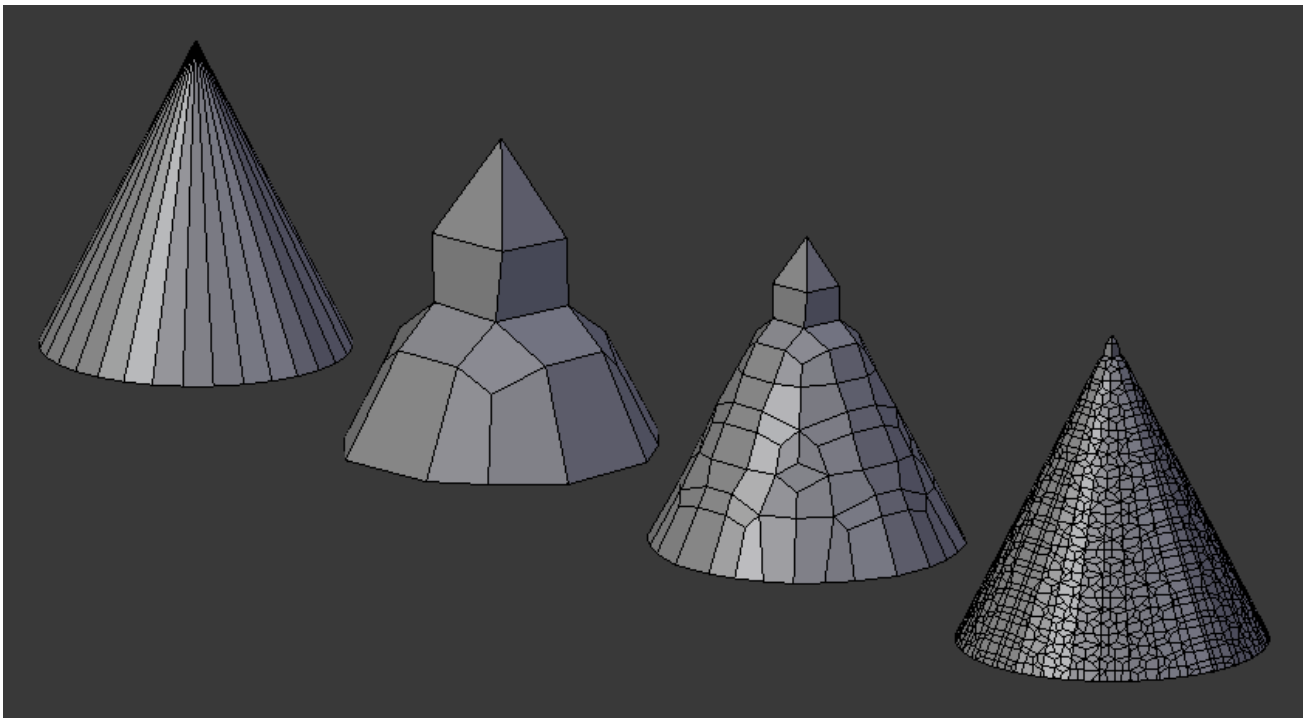


Fig. 2.833: Input mesh, and the low to high resolution output meshes.

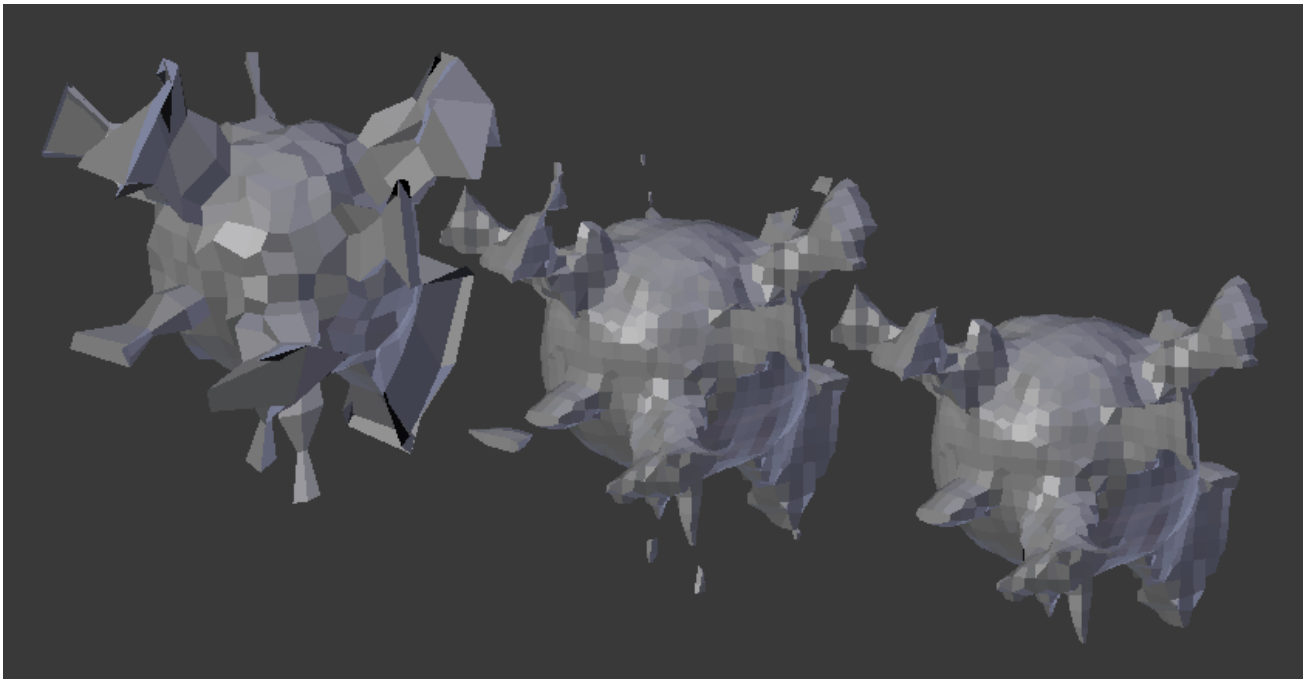


Fig. 2.834: The input mesh (left) is fairly noisy, so the initial output of the remesh modifier (center) contains small disconnected pieces. Enabling Remove Disconnected Pieces (right) deletes those faces.

Examples

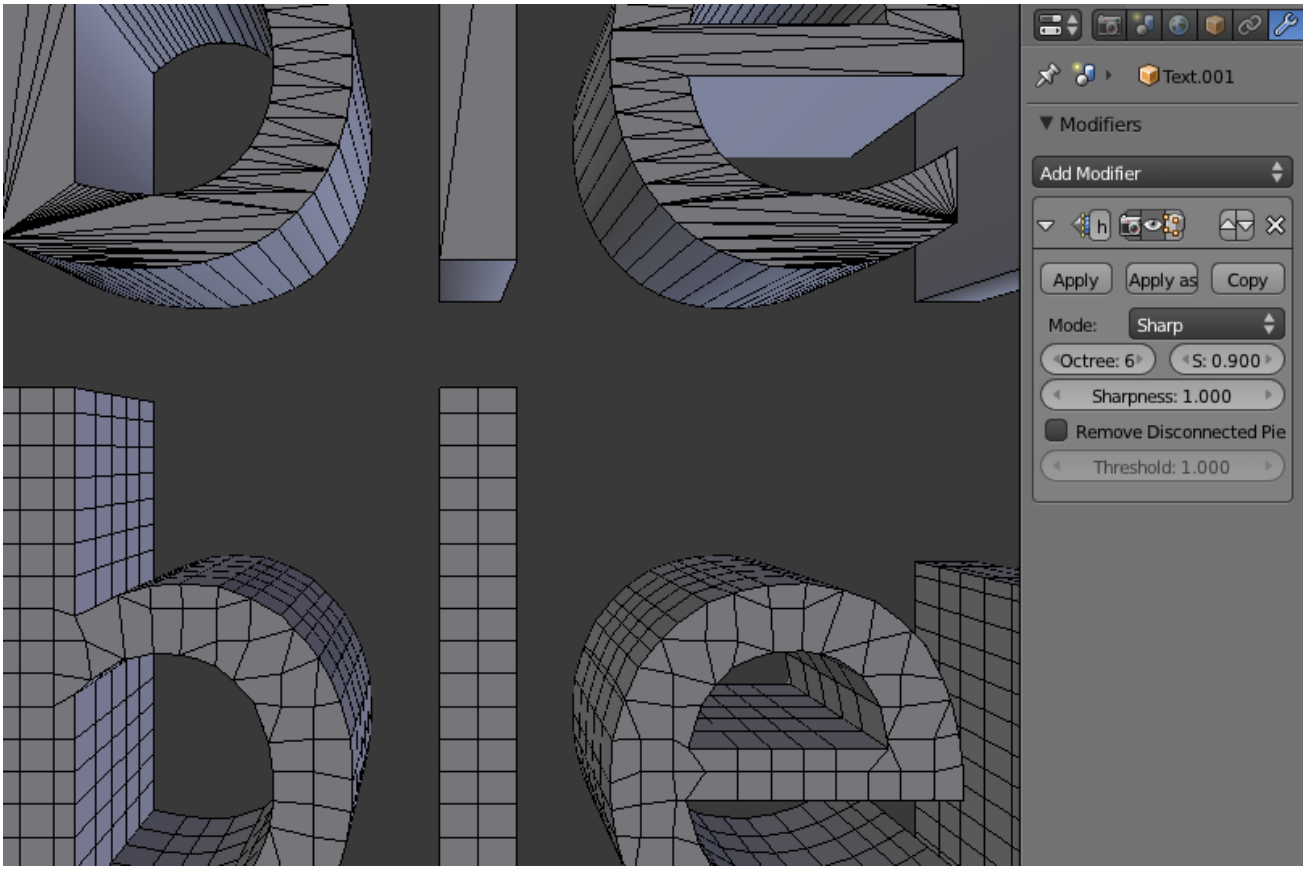


Fig. 2.835: Remesh modifier applied to text to improve topology.

Screw Modifier

The Screw modifier is similar to the *Screw tool* in the *Tool Shelf* in that it takes a profile object, a Mesh or a Curve, to create a helix-like shape.

The profile should be properly aligned to the cardinal direction of the object rather than to the screw axis.

Options

Axis The axis along which the helix will be built.

Screw The height of one helix iteration.

Axis Object The name of an object to define the axis direction.

Object Screw Use the distance from the *Axis Object* to define the height of one helix iteration.

Angle Degrees for a single helix revolution.

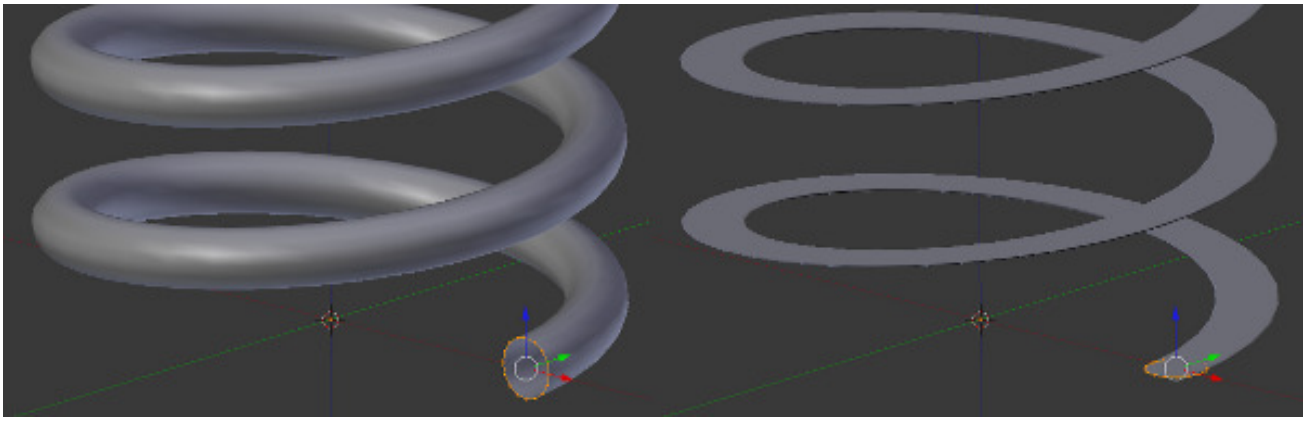


Fig. 2.836: Properly aligning the profile object is important.

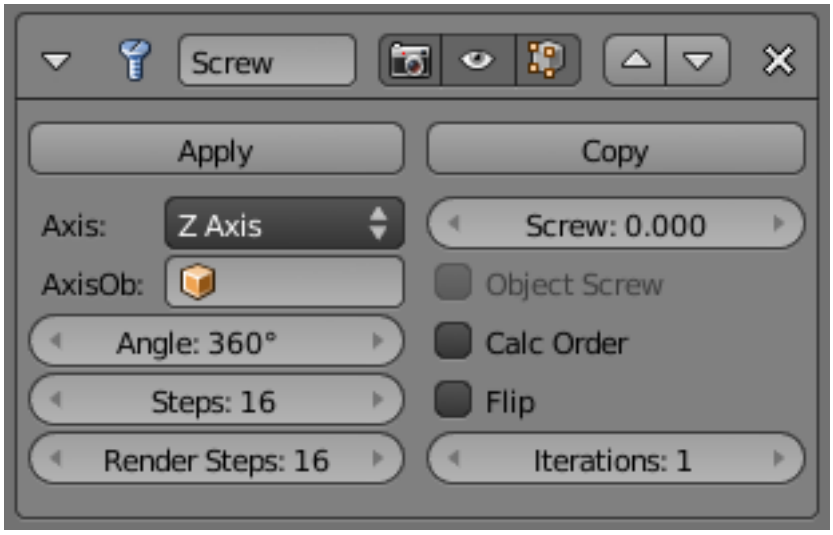


Fig. 2.837: Screw modifier.

Steps Number of steps used for a single revolution displayed in the 3D View. Beware of setting this higher than *Render Steps*, which is the value used for rendering.

Render Steps As above, but used during render time. Increase to improve quality.

Smooth Shading Output faces with smooth shading rather than flat shading. The smooth/flat shading of the input geometry is not preserved.

Calc Order Order of edges is calculated to avoid problems with normals and shading. Only needed for meshes, not curves.

Flip Flip normals direction.

Iterations Number of revolutions.

Stretch U/V Stretch the UV coordinates from (0.0 to 1.0) when UVs are present.

Skin Modifier

The Skin modifier uses vertices and edges to create a skinned surface, using a per-vertex radius to better define the shape. The output is mostly quads, although some triangles will appear around intersections.

It is a quick way to generate base meshes for sculpting and/or smooth organic shapes with arbitrary topology.

Note: Faces in the original geometry are ignored by the Skin modifier.

Options

Create Armature Create an armature on top of the object. Each edge becomes a bone.

Note: If the root vertex has more than one adjacent edge, an extra bone will be created to serve as the root.

This function does the following:

1. A new armature object is added with bones matching the input mesh. The active selection is switched to the new armature.
2. Weight groups are added to the input mesh. The Skin modifier propagates these weights to the output as well.

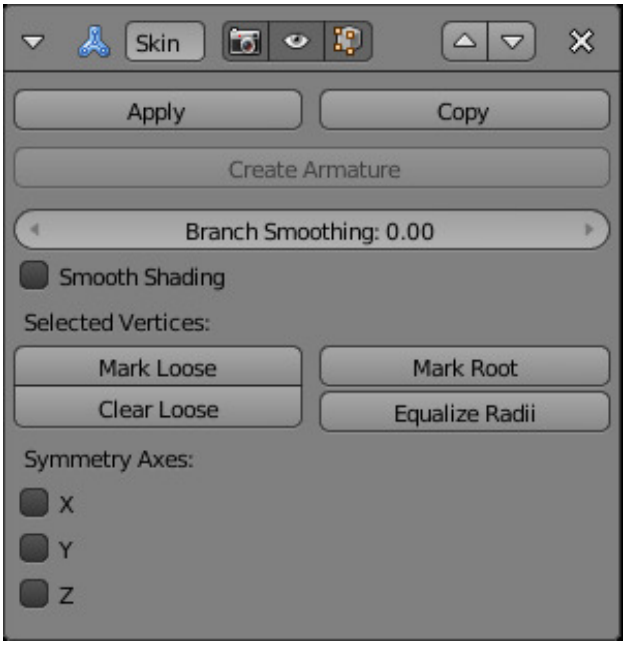


Fig. 2.838: Skin modifier UI.

3. An Armature modifier is added directly below the Skin modifier. Note that the Armature modifier is being applied after the Skin modifier because it should only deform the output, whereas if it were above the Skin modifier it might change the resulting topology.

Branch Smoothing A branch point is a vertex with three or more connected edges. These areas tend to produce more complicated topology, some of which may overlap. The *Branch Smoothing* setting relaxes the surface around these points, with the side effect of shrinking the surface.

Smooth Shading Output faces with smooth shading rather than flat shading. The smooth/flat shading of the input geometry is not preserved.

Selected Vertices

Mark/Clear Loose By default, a branch vertex (vertex with three or more connected edges) will generate extra edge loops along adjacent edges in order to keep the output tight. Branches can be made loose by clicking *Mark Loose*, which will allow the output to stretch between all adjacent vertices. This can be disabled again by clicking *Clear Loose* with the vertex selected.

Mark Root Marking a vertex as root causes that vertex to be used for calculating rotations for connected limbs. Root vertices also affect the armature output; they will be

used as the origin for the root bones.

Roots are shown in the 3D View with a red dashed circle around the vertex.

Each set of connected vertices should have one root node. *Mark Root* enforces the one-root per set rule, so it is not necessary to manually unmark roots.

Equalize Radii Makes the skin radii of selected vertices equal on each axis.

Symmetry Axes The Symmetry Axes checkboxes are used to keep the output topology symmetrical in their respective axes. In other words, using it avoids merging triangles across an axis unless the triangles form a symmetric quad.

Note: These symmetry axes checkboxes do not add geometry flipped across an axis. For that, the Mirror modifier should be used, typically placed above the Skin modifier.

Usage

Add the Skin modifier to a mesh. Disable *Limit selection to visible* in the 3D View so that you can see the vertices inside the new geometry.

The skin modifier uses ordinary vertices and edges as input. All of the regular Edit Mode tools (such as extrude, subdivide, grab, scale, and rotate) can be used when building a skinned mesh.

The radius of selected vertices can be adjusted in the *Transform* panel of the *Properties* region `N`

Examples

- In the modifiers menu, add a *Skin* modifier.
- `Tab` into edit mode and start extruding.
- Try to sketch results similar to Fig. *Simple creature, made with only the Skin modifier.*, through extruding the vertices of the object.
- Use `Ctrl-A` to change the size of the different regions within the creature.
- Use *Mark Loose* at regions like the neck, to merge these faces more together.

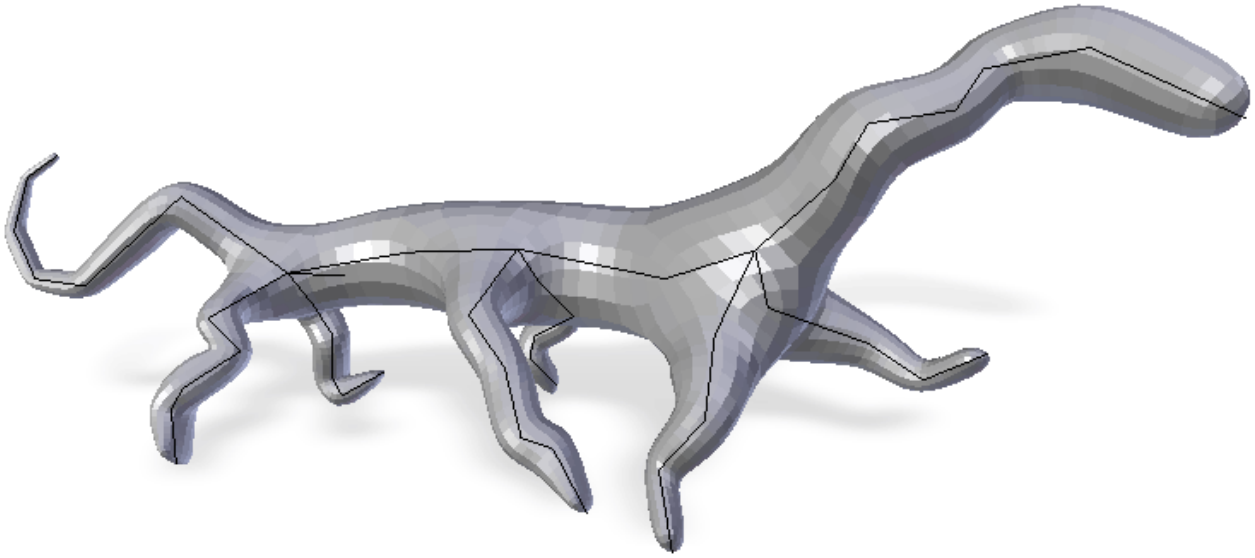


Fig. 2.839: Simple creature, made with only the Skin modifier.

- To get smoother results, activate *Smooth Shading* and add a *Subdivision Surface*

External links

- Skin Modifier Development at Blender Nation – An early demonstration of the skin modifier by Nicholas Bishop (March 2011).

- Ji, Zhongping; Liu, Ligang; Wang, Yigang (2010). B-Mesh: A Fast Modeling System for Base Meshes of 3D Articulated Shapes, Computer Graphics Forum 29(7), pp. 2169-2178. – The work this modifier is based on (direct link to PDF).
- Related thread on Blender artists.

Solidify Modifier

The Solidify modifier takes the surface of any mesh and adds depth to it.

Options

Thickness The depth to be solidified.

Offset A value between (-1 to 1) to locate the solidified output inside or outside the original mesh. Set to 0.0, the solidified output will be centered on the original mesh.

Clamp A value between (0 to 2) to clamp offsets to avoid self intersection.

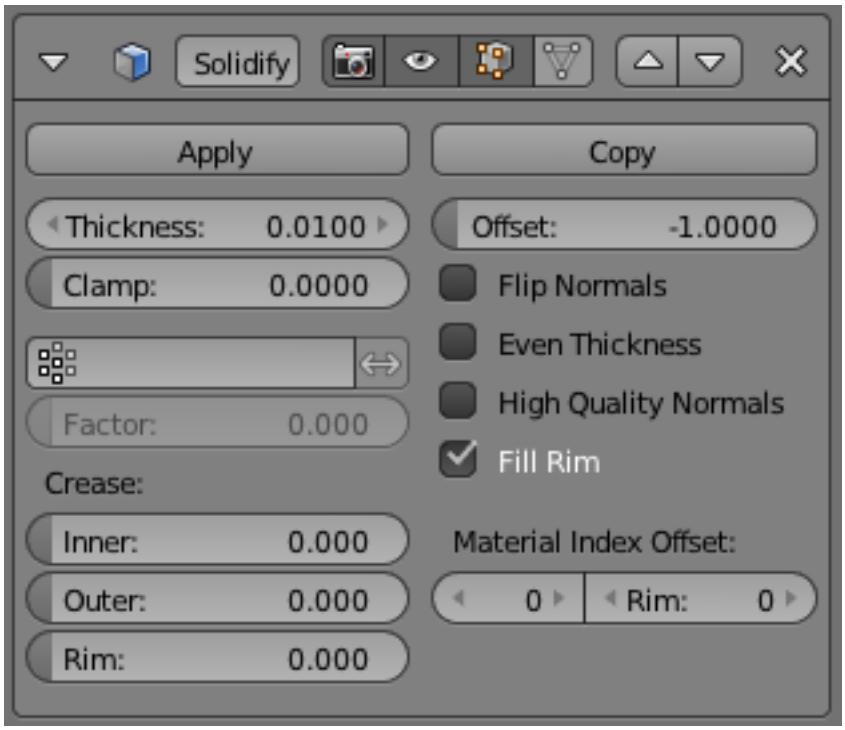


Fig. 2.840: Solidify Modifier.

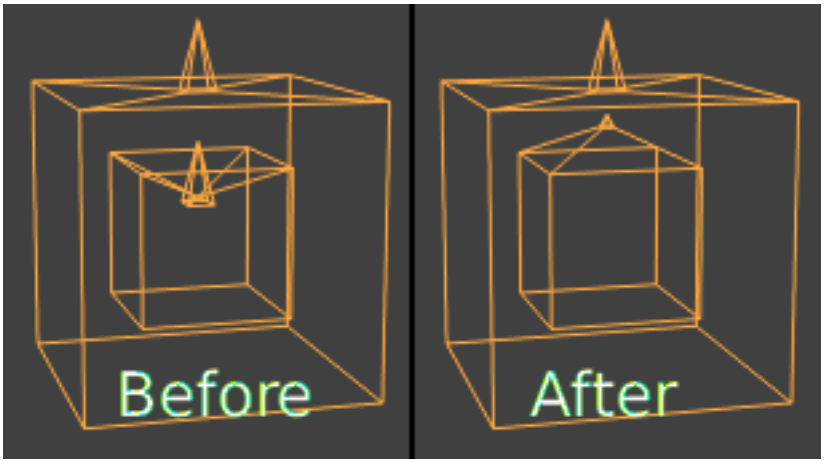


Fig. 2.841: Clamp Offset.

Vertex Group Only vertices in this group are solidified. Their weights are multiplied by the thickness, so vertices with lower weights will be less thick.

Invert Reverses the vertex group, so that only vertices which are **not** in the vertex group are solidified.

Factor How much the vertex weights are taken into account.

- On 0.0 , vertices with zero weight will have no thickness at all.
- On 0.5 , vertices with zero weight will be half as thick as those with full weight.
- On 1.0 , the weights are ignored and the *thickness* value is used for every vertex.

Crease These options are intended for usage with the *Subdivision Modifier*.

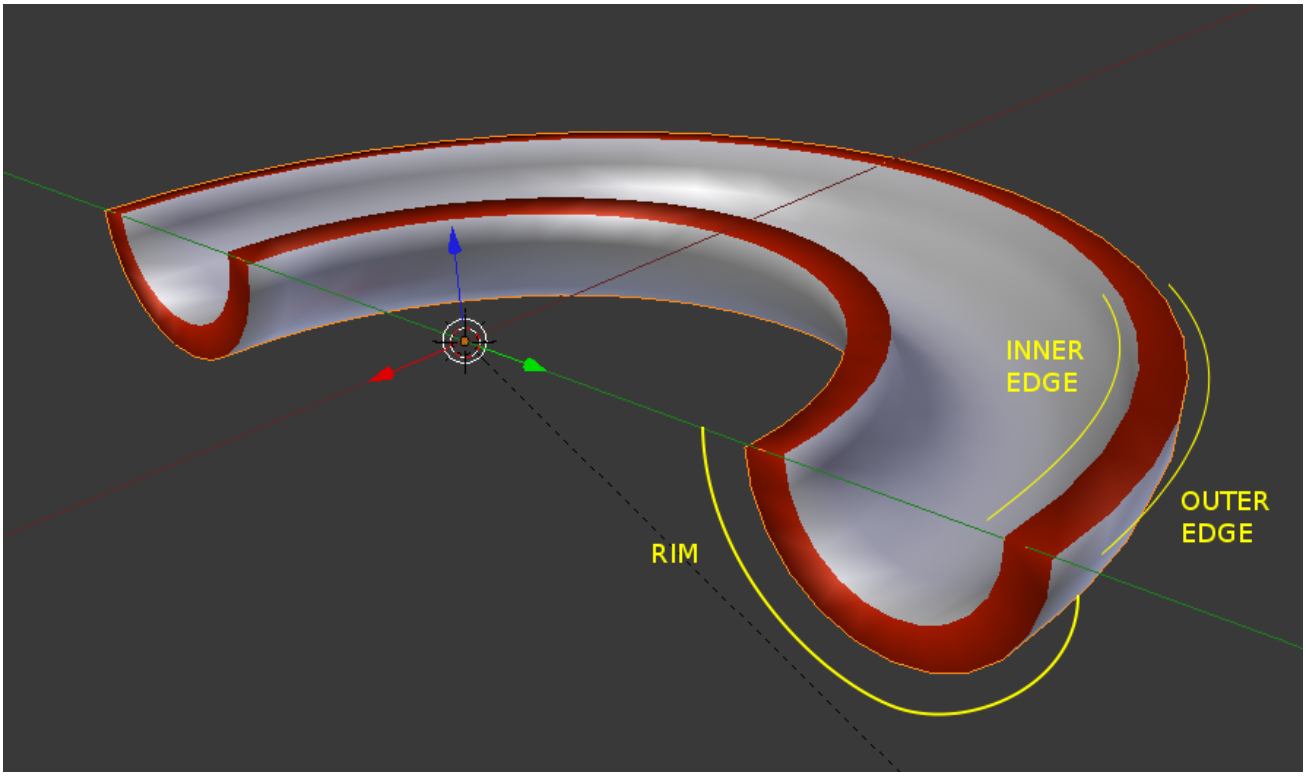


Fig. 2.842: Rim and edges. In this example, the object was assigned a second material used to color the rim red.

Inner Set a crease to the inner edges.

Outer Set a crease to the outer edges.

Rim Set a crease to the rim.

Flip Normals Reverse the normals of all geometry (both the inner and outer surfaces).

Even Thickness Maintain thickness by adjusting for sharp corners. Sometimes im-

proves quality but also increases computation time.

High Quality Normals Normals are calculated to produce a more even thickness. Sometimes improves quality but also increases computation time.

Fill Rim Fills the gap between the inner and outer edges.

Only Rim Will not have an extruded surface parallel to the original but instead will only have the perpendicular rim.

Note: *Fill Rim* and *Only Rim* only make a difference on *non-manifold* objects, since the "rims" are generated from the borders of the original geometry.

Material Index Offset Choose a different material to use for the new geometry; this is applied as an offset from the original material of the face from which it was solidified.

- A value of 0 means it will use the same material.
- A value of 1 means it will use the material immediately below the original material.
- A value of -2 means the material two positions above the original material will be used.

These are clamped to the top-most and bottom-most material slots.

Rim Similarly, you can give another material to the rim faces.

Warning: The modifier thickness is calculated using local vertex coordinates. If the object has non-uniform scale, the thickness will vary on different sides of the object.

To fix this, either apply `Ctrl-A` or clear `Alt-S` scale.

Known Limitations

Even Thickness

Solidify thickness is an approximation. While "Even Thickness" and "High Quality Normals" should yield good results, the final wall thickness is not guaranteed and may vary depending on the mesh topology.

In order to maintain precise wall thickness in every case, we would need to add/remove faces on the offset shell, something this modifier does not do since this

would add a lot of complexity and slow down the modifier.

Subdivision Surface Modifier

Subdivision Surface modifier is used to subdivide the faces of a mesh to give a smooth appearance, to enable modeling of complex smooth surfaces with simple, low-vertex meshes. This allows high resolution mesh modeling without the need to save and maintain huge amounts of data and gives a smooth *organic* look to the object.

This process creates virtual geometry that is generated non-destructively without modifying the original mesh, but it can be converted to real geometry that you could edit with the *Apply* button.

Also, like the rest of the Modifiers, order of execution has an important bearing on the results. For this, see the documentation on the *Modifier Stack*.

Keep in mind that this is a different operation than its companion, *Smooth Shading*. You can see the difference between the two in the grid image below.

Fig. 2.843: Subdivisions levels 0 to 3, without and with Smooth Shading.

Tip: The Subdivision Surface modifier does not allow you to edit the new subdivided geometry without applying it, but the *Multires* modifier does (in sculpt mode).

Options

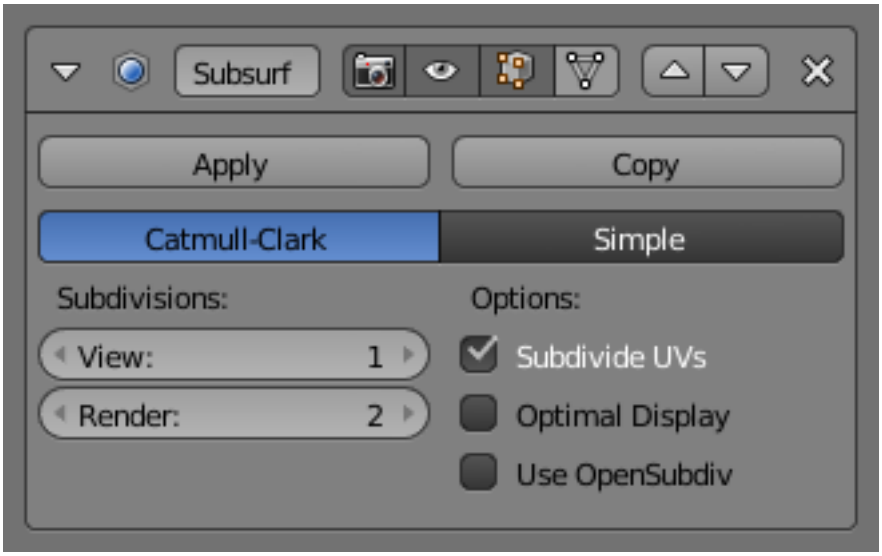


Fig. 2.844: Modifier's Panel.

Type This toggle button allows you to choose the subdivision algorithm:

Catmull-Clark The default option, subdivides and smooths the surfaces. According to its Wikipedia page, the "arbitrary-looking formula was chosen by Catmull and Clark based on the aesthetic appearance of the resulting surfaces rather than on a mathematical derivation."

Simple Only subdivides the surfaces, without any smoothing (the same as `W` → *Subdivide*, in Edit Mode). Can be used, for example, to increase base mesh resolution when using displacement maps.

Subdivisions Recursively adds more geometry. For details on polygon counts, see the *Performance Considerations* section.

View The number of subdivision levels shown in the 3D View.

Render The number of subdivision levels shown in renders.

The right combination of these settings will allow you to keep a fast and lightweight approximation of your model when interacting with it in 3D, but use a higher quality version when rendering.

> **Warning:** Be careful not to set the *View* subdivisions higher than the *Render* subdivisions, this would mean the 3D View will be higher quality than the render.

Options

Subdivide UVs When enabled, the UV maps will also be subdivided (i.e. Blender will add "virtual" coordinates for all sub-faces created by this modifier).

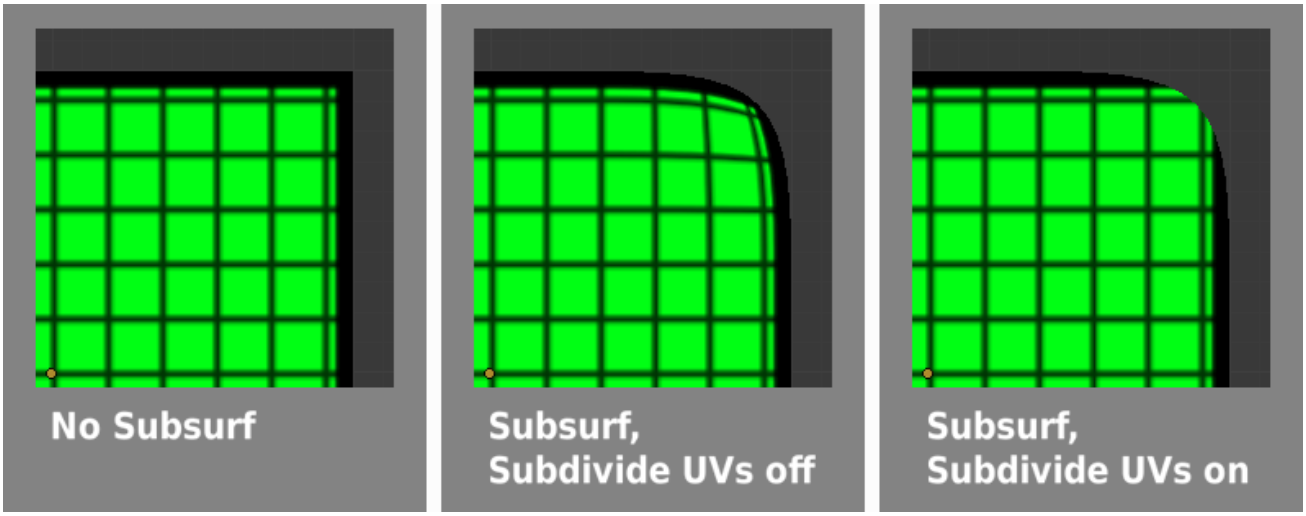


Fig. 2.845: Subdivide UVs on and off.

Optimal Display When drawing the wireframe of this object, the wires of the new subdivided edges will be skipped (only draws the edges of the original geometry)

Opensubdiv See the *OpenSubdiv* section.

OpenSubdiv

When *OpenSubdiv* is enabled, the modifier evaluation will happen on a compute device. To enable OpenSubdiv you must first choose the fastest compute device in the *User Preferences*. Most of the time the best performance will be achieved when using *GLSL*. As a result performance of the modifier will be much higher which is great for animations.

See also:

To find more on OpenSubdiv read the Release Notes.

Improving Performance

In order to utilize maximum performance form OpenSubdiv the following things are required:

- The modifier must be last in the *Modifier Stack*.

- There should be no modifiers prior to the which changes mesh topology across the time.
- Other objects should not use geometry of OpenSubdiv mesh

Control

Catmull-Clark subdivision rounds off edges, and often this is not what you want. There are several solutions that allow you to control the subdivision.

Weighted Edge Creases

Weighted edge creases for subdivision surfaces allows you to change the way the Subdivision Surface modifier subdivides the geometry to give the edges a smooth or sharp appearance.

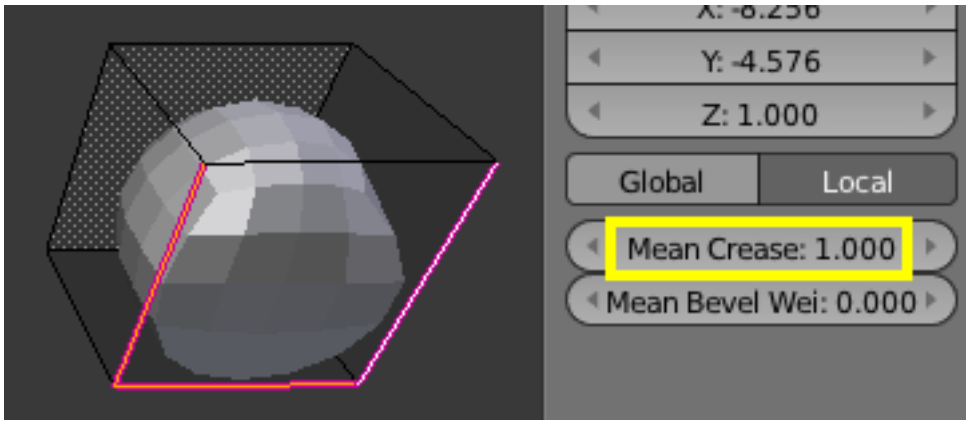


Fig. 2.846: A Subdivided Cube with Creased Edges.

The crease weight of selected edges can be changed in the *Transform* panel of the properties region `N`, or by using the shortcut `Shift-E` and moving the mouse closer or further from the selected edges to adjust the crease weight. A higher value makes the edge "stronger" and more resistant to the smoothing effect of subdivision surfaces.

Edge Loops

The Subdivision Surface modifier demonstrates why good, clean topology is so important. As you can see in the figure, the Subdivision Surface modifier has a drastic effect on a default Cube. Until you add

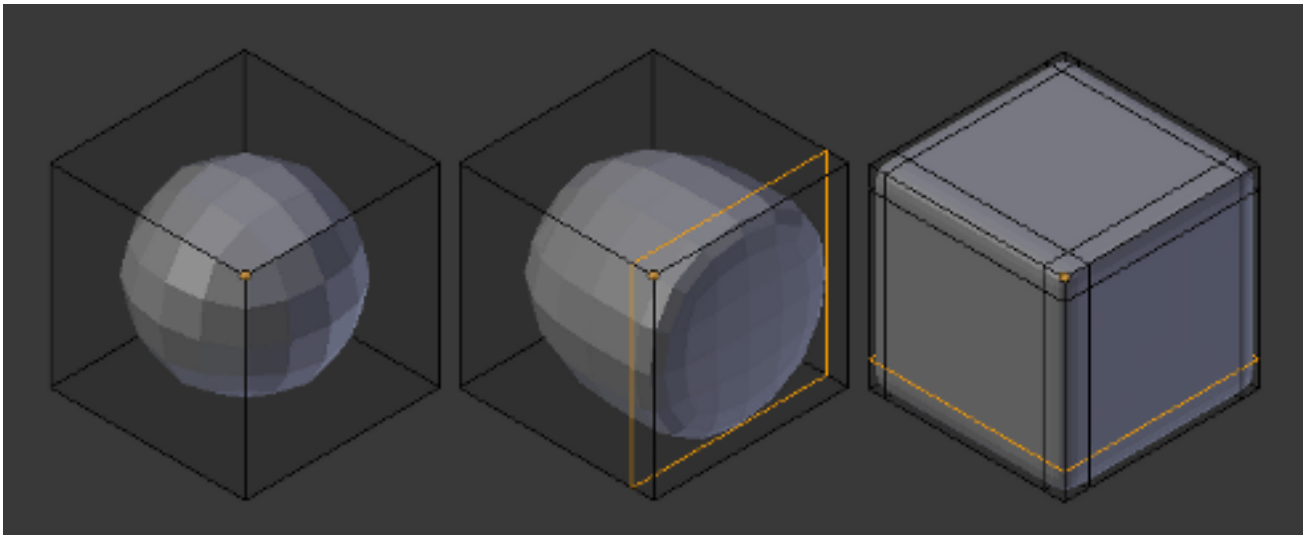


Fig. 2.847: Subdivision Level 2 Cube, the same with an extra Edge Loop, and the same with six extra Edge Loops.

in additional Loops (with Ctrl-R), the shape is almost unrecognizable as a cube.

A mesh with deliberate topology has good placement of Edge Loops, which allow the placement of more Loops (or removal of Loops, with X → *Edge Loop*) to control the sharpness/smoothness of the resultant mesh.

Performance Considerations

Higher levels of subdivisions mean more vertices, and more vertices means more memory will be used (both video memory for display, and system RAM for rendering). Blender could potentially crash or hang if you do not have enough memory.

When using high levels of subdivision with a graphics card that has a low total amount of Vram, some parts of the geometry will disappear visually. Your mesh will actually be intact, because the render is generated using your Object Data, (even though it cannot be shown by your graphics card).

Keyboard Shortcuts

To quickly add a Subdivision Surface modifier to one or more objects, select it/them and press Ctrl-1. That will add a Subdivision Surface modifier with *View Subdivisions* on 1.

You can use other numbers too, such as Ctrl-2, Ctrl-3, etc, to add a Subdi-

vision Surface modifier with that number of subdivisions. The *Render Subdivisions* will always be on 2 when added like this.

If an object already has a Subdivision Surface modifier, doing this will simply change its subdivision level instead of adding another modifier.

Known Limitations

Non Contiguous Normals

Blender's subdivision system produces nice smooth subdivided meshes, but any subdivided face (that is, any small face created by the algorithm from a single face of the original mesh), shares the overall normal orientation of that original face.

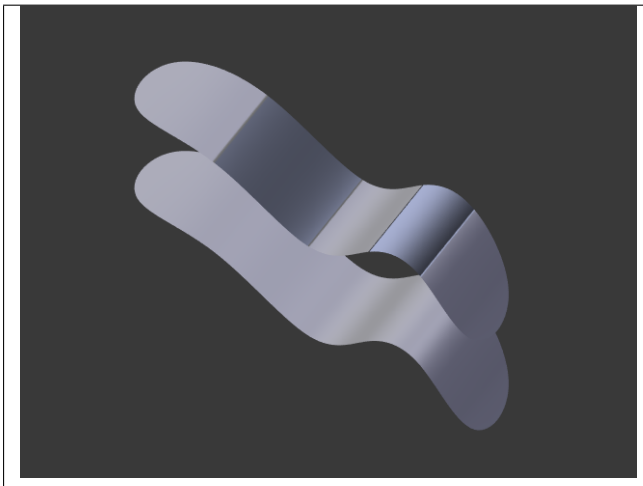


Fig. 2.848: Comparison of good normals and bad normals.

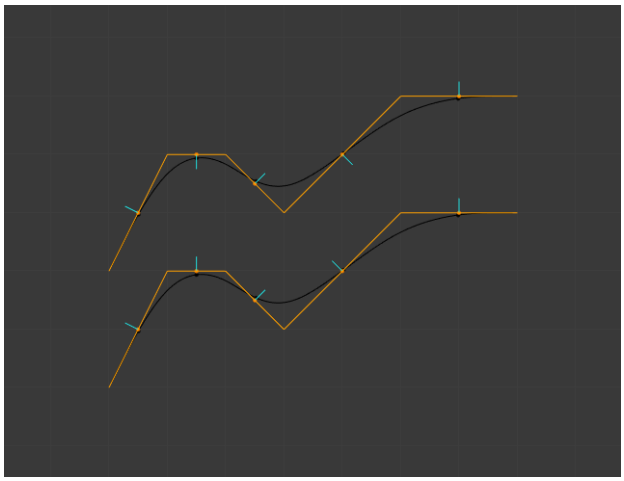


Fig. 2.849: Side view of image on left.

Abrupt normal changes can produce ugly black gouges even though these flipped normals are not an issue for the shape itself.

A quick way to fix this is to use Blender's *Recalculate Normals* operation in Edit Mode.

If you still have some ugly black gouges you will have to *Manually Flip the Normals*.

Concave N-Gons

While n-gons are supported, concave n-gons may give ugly overlapping results.

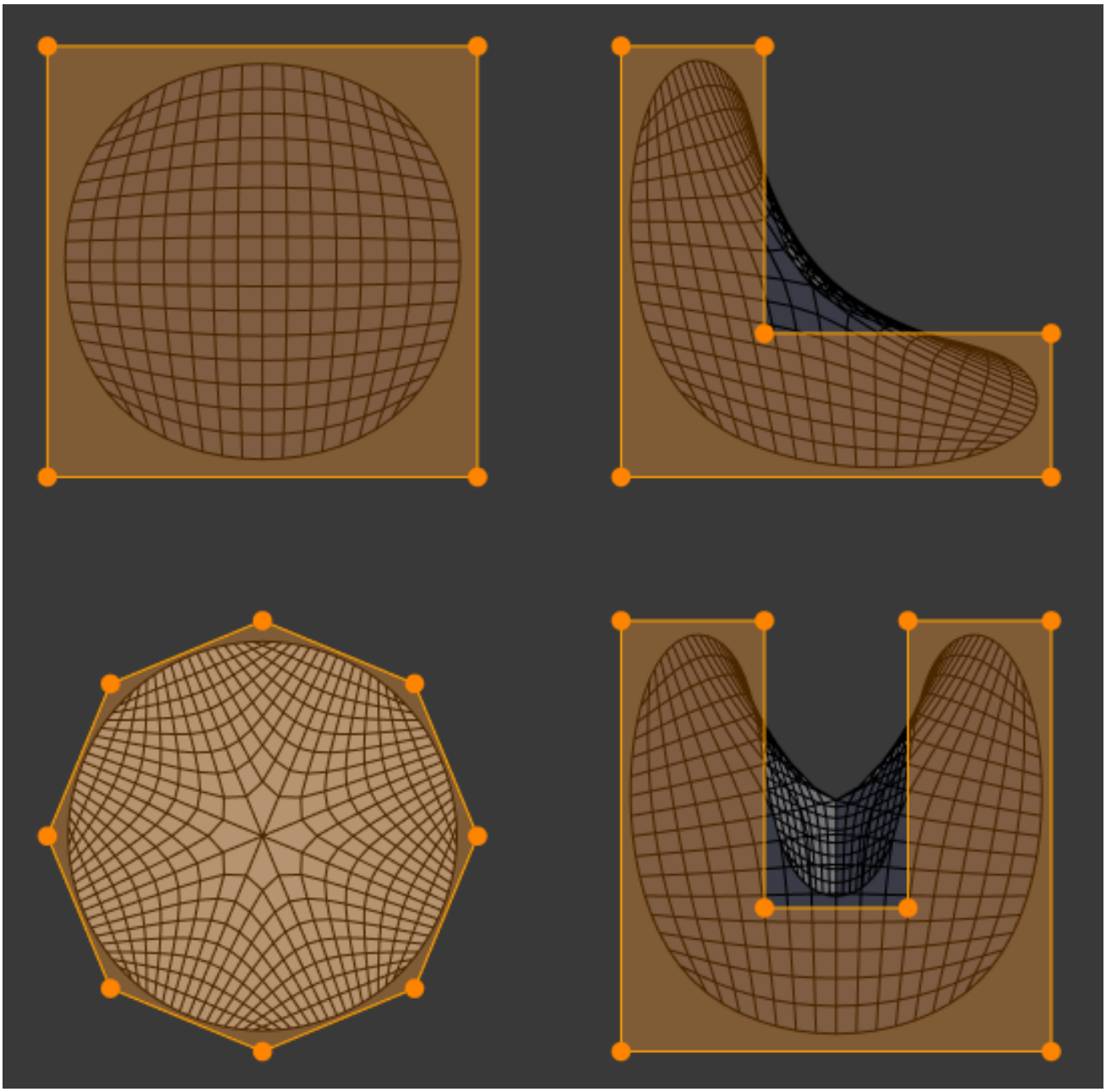


Fig. 2.850: The n-gons on the right show overlapping results.

Triangulate Modifier

The Triangulate modifier converts all faces in a mesh (whether it be quads or N-gons) to triangular faces. This modifier does the exact same function as the triangulate function `Ctrl-T` in Edit Mode.

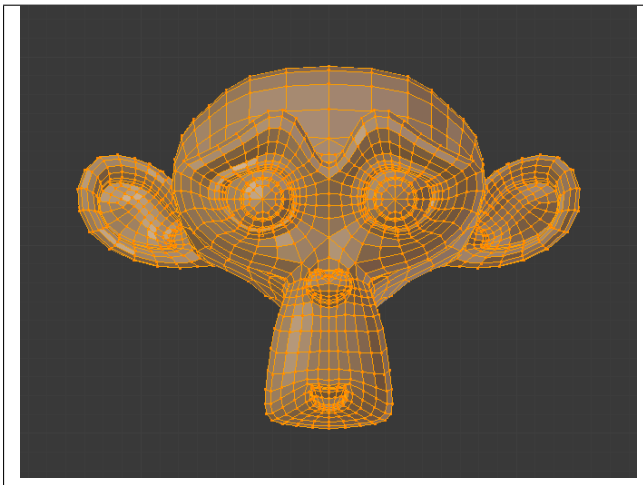


Fig. 2.851: Mesh before Triangulate Modifier.

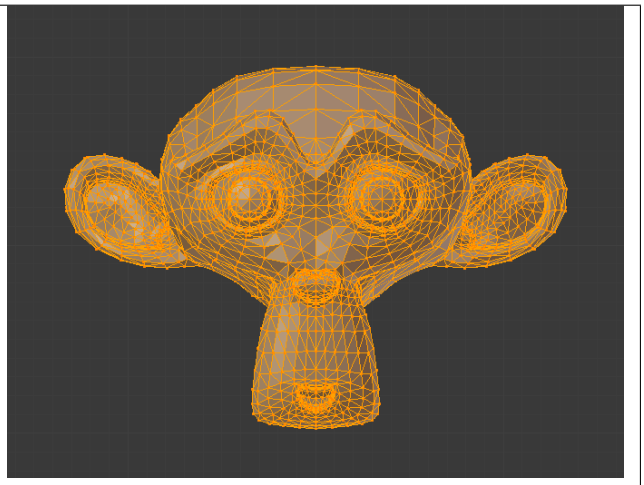


Fig. 2.852: Mesh after Triangulate Modifier.

Options

Quad Method

Beauty Split the quads in nice triangles, slower method.

Fixed Split the quads on the 1st and 3rd vertices.

Fixed Alternate Split the quads on the 2nd and 4th vertices.

Shortest Diagonal Split the quads based on the distance between the vertices.

Ngon Method

Beauty Arrange the new triangles nicely, slower method.

Scanfill Split the ngons using a scanfill algorithm.

Wireframe Modifier

The Wireframe modifier transforms a mesh into a wireframe by iterating over its faces, collecting all edges and turning those edges into four sided polygons. Be aware of the fact that your mesh needs to have faces to be wireframed. You can define the thickness, the material and several other param-

eters of the generated wireframe dynamically via the given modifier options.

Options

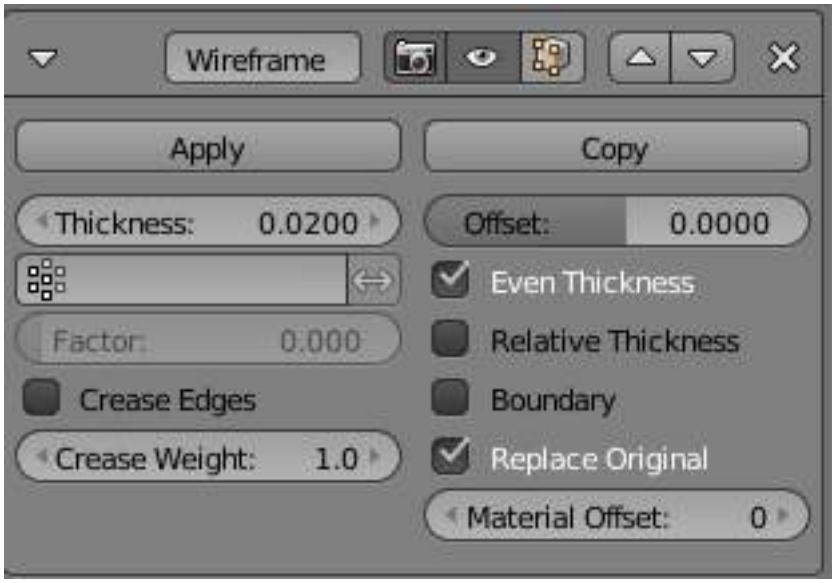


Fig. 2.853: Wireframe Modifier.

Thickness The depth or size of the wireframes.

Offset A value between (-1 to 1) to change whether the wireframes are generated inside or outside the original mesh. Set to zero, *Offset* will center the wireframes around the original edges.

Vertex Group Restrict the modifier to only this vertex group.

Invert Inverts the vertex group weights.

Crease Edges This option is intended for usage with the *Subdivision Modifier*. Enable this option to crease edges on their junctions and prevent large curved intersections.

Crease Weight Define how much crease (0 to 1) (no to full) the junctions should receive.

Even Thickness Maintain thickness by adjusting for sharp corners. Sometimes improves quality but also increases computation time.

Relative Thickness Determines the edge thickness by the length of the edge. Longer edges will be thicker.

Boundary Creates wireframes on mesh island boundaries.

Replace Original If this option is enabled, the original mesh is replaced by the generated

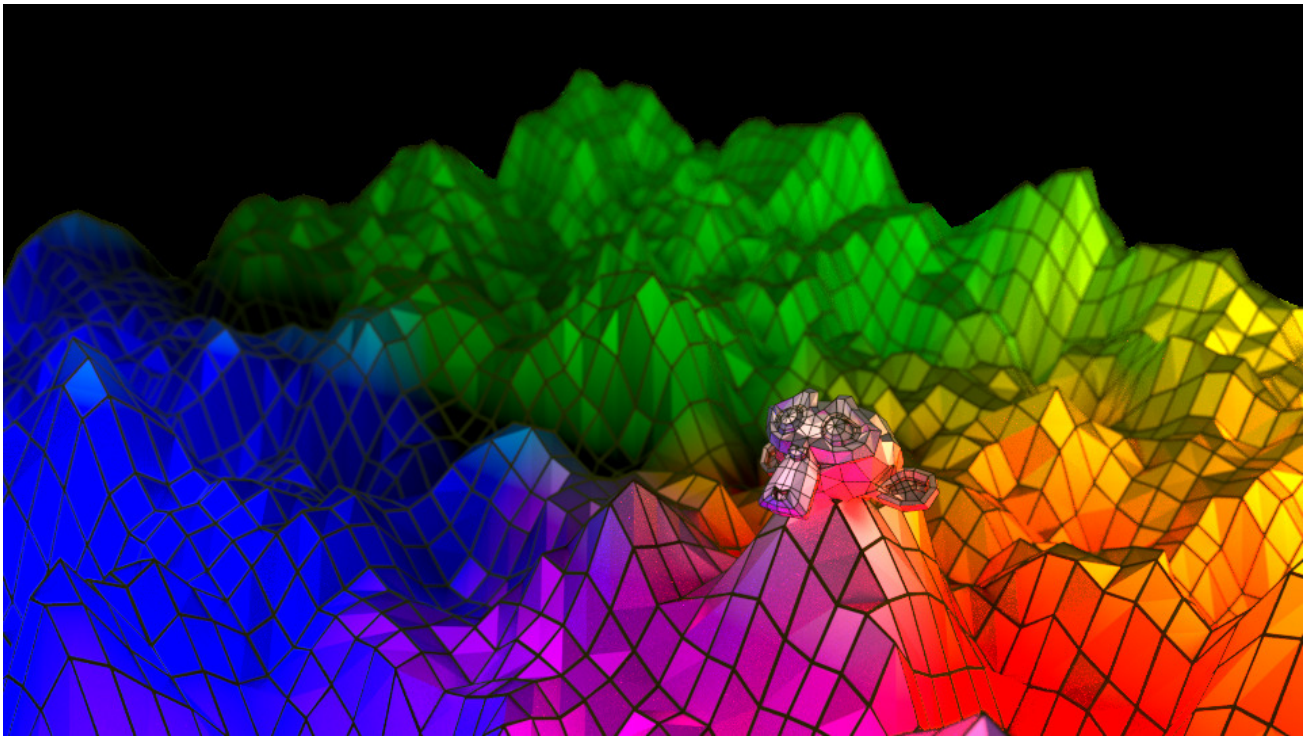


Fig. 2.854: Wireframes on a displaced plane. In this example, the wireframes carry a second (dark) material while the displaced plane uses its original one.

wireframe. If not, the wireframe is generated on top of it.

Material Offset Uses the chosen material index as the material for the wireframe; this is applied as an offset from the first material.

Examples

When you got more Faces that meet at one point they are forming a star like pattern like seen in the examples below.

Warning: Wireframe thickness is an approximation. While *Even Thickness* should yield good results in many cases, skinny faces can cause ugly spikes. In this case you can either reduce the extreme angles in the geometry or disable the *Even Thickness* option.

Deform

Armature Modifier

The *Armature* modifier is used for building skeletal systems for animating the poses of characters and anything else which needs to be posed.

By adding an armature system to an object, that object can be deformed accurately so

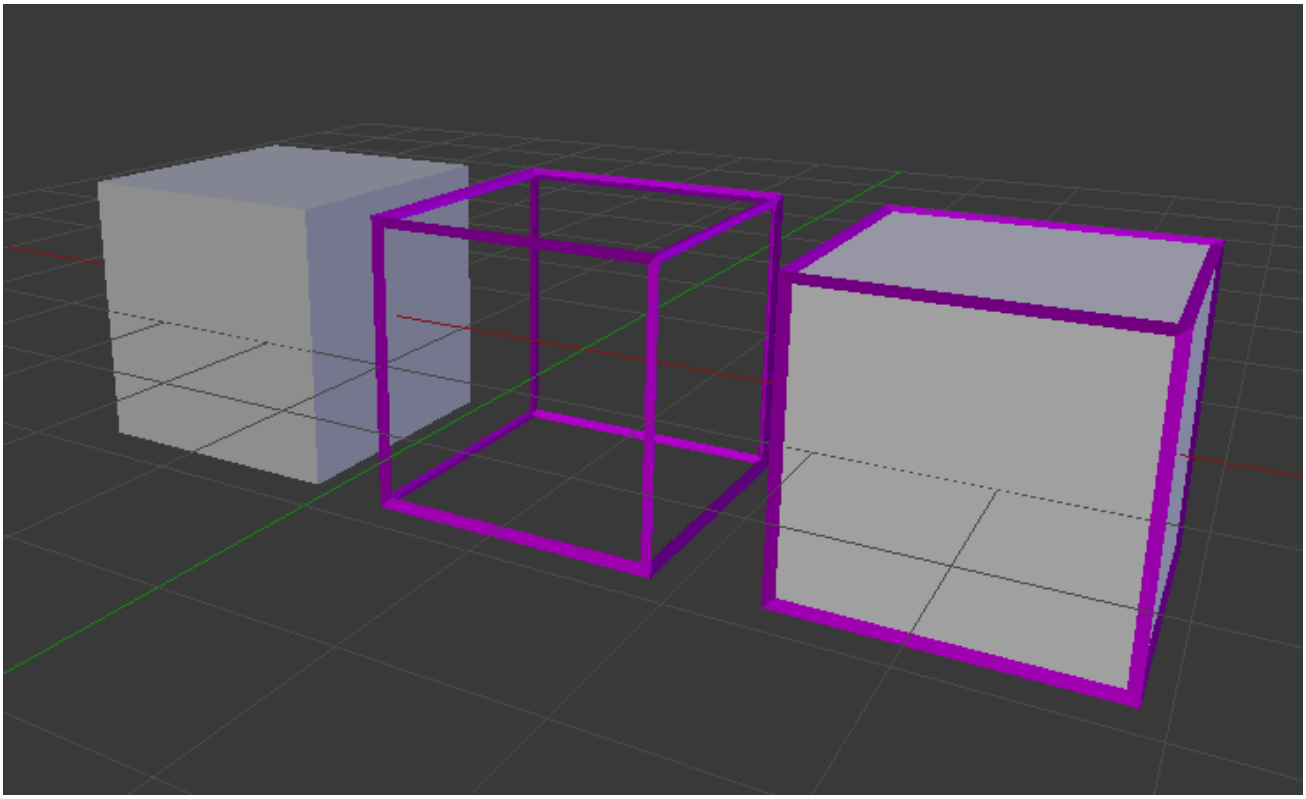


Fig. 2.855: Original / Wireframe / Original and Wireframe.

that geometry does not have to be animated by hand.

Note: For more details on armatures usage, see *this chapter*.

Options

Object The name of the armature object used by this modifier.

Preserve Volume Use quaternions for preserving volume of object during deformation. It can be better in many situations.

Vertex Group The name of a vertex group of the object, the weights of which will be used to determine the influence of this *Armature* modifier's result when mixing it with the results from other *Armature* ones.

Only meaningful when having at least two of these modifiers on the same object, with *Multi Modifier* activated.

Invert Inverts the influence set by the vertex group defined in previous setting (i.e. reverses the weight values of this group).

Multi Modifier Use the same data as a previous modifier (usually also an *Armature* mod-

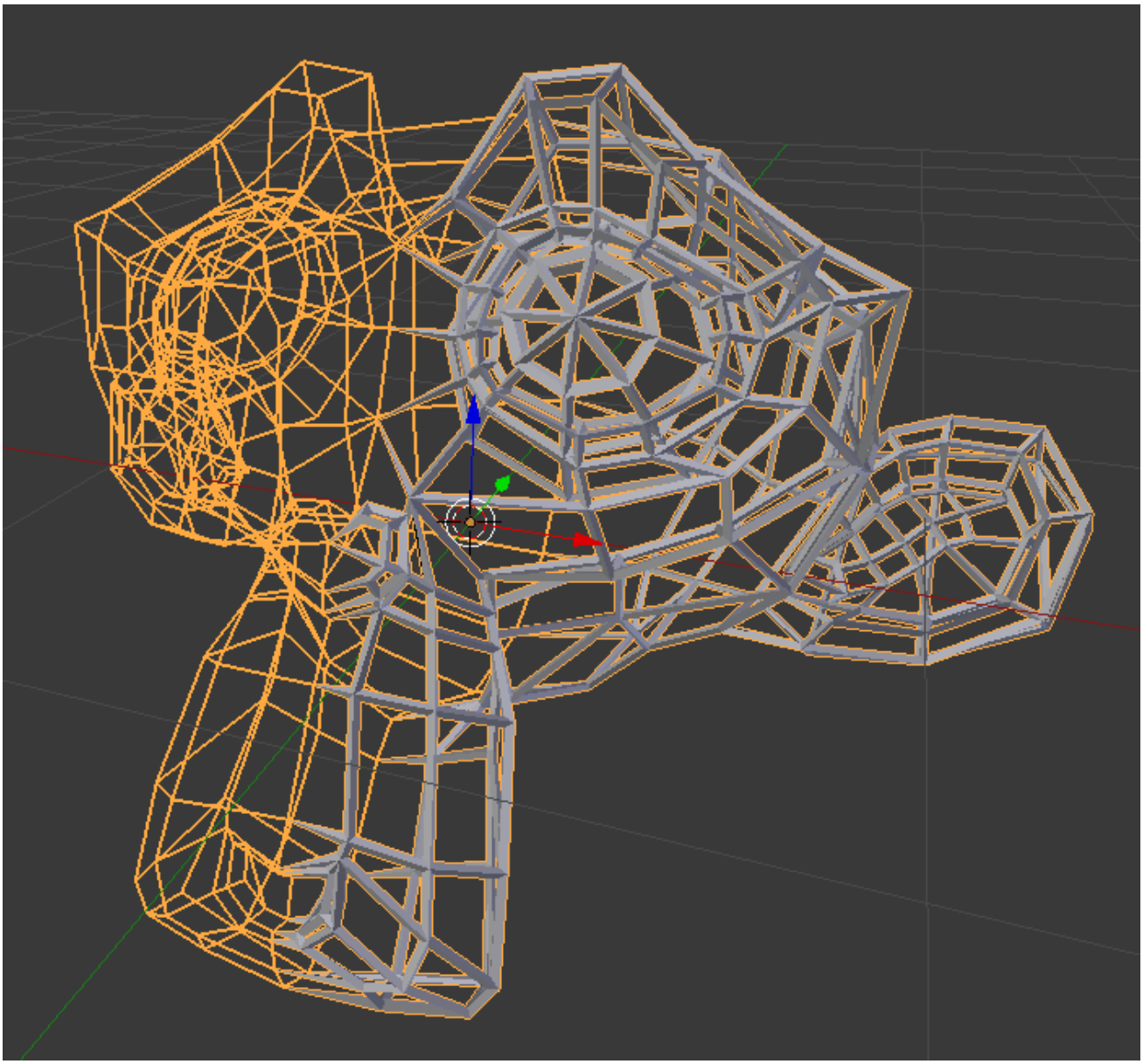


Fig. 2.856: Vertex Group weighting: One half 0 weighted, one half 1 weighted.

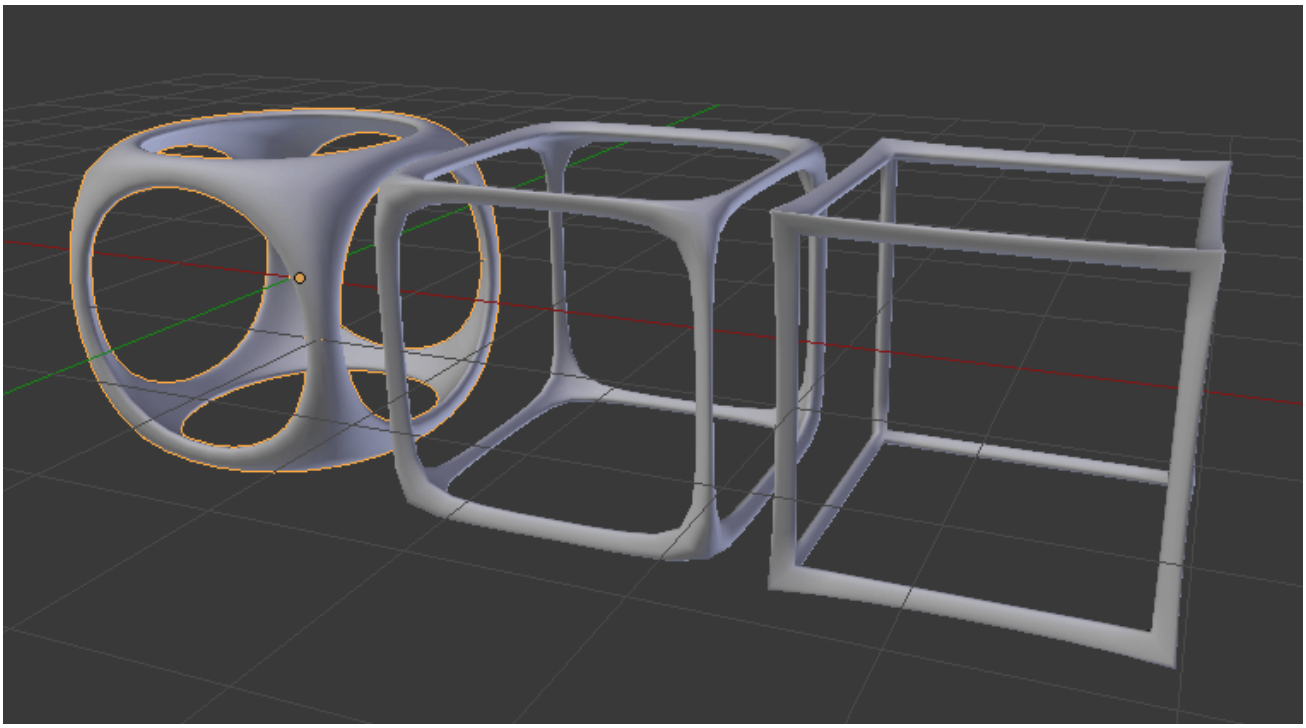


Fig. 2.857: Cube and Subdivision Surface with 0 / 0.5 / 1 crease weight.

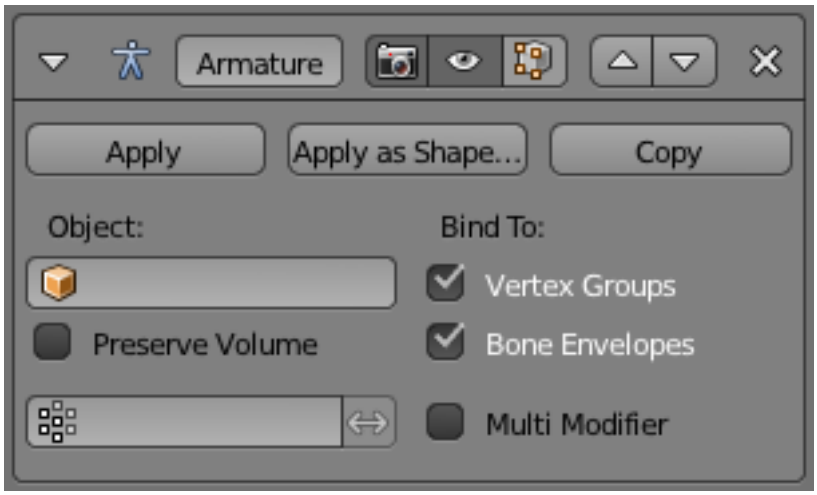


Fig. 2.858: Armature modifier.

ifier) as input. This allows you to use several armatures to deform the same object, all based on the "non-deformed" data (i.e. this avoids having the second *Armature* modifier deform the result of the first one...).

The results of the *Armature* modifiers are then mixed together, using the weights of the *Vertex Group* as "mixing guides".

Bind To Method to bind the armature to the mesh.

Vertex Groups When enabled, bones of a given name will deform vertices which belong to vertex groups of the same name.

Bone Envelopes When enabled, bones will deform vertices near them (defined by each bones envelope radius) Enable/Disable bone envelopes defining the deformation (i.e. bones deform vertices in their neighborhood).

Tip: Armature modifiers can quickly be added to objects using the parenting shortcut `Ctrl-P` when the active object is an armature.

Cast Modifier

This modifier shifts the shape of a mesh, curve, surface or lattice to any of a few predefined shapes (sphere, cylinder, cuboid).

It is equivalent to the *To Sphere* tool in *Edit Mode Mesh* → *Transform* → *To Sphere*, `Alt-Shift-S` and what other programs call "Spherify" or "Spherize", but, as written above, it is not limited to casting to a sphere.

Tip: The *Smooth modifier* is a good companion to *Cast*, since the cast shape sometimes needs smoothing to look nicer or even to fix shading artifacts.

Note: For performance reasons, this modifier only works with local coordinates. If the modified object looks wrong, you may need to apply its rotation `Ctrl-A`, especially when casting to a cylinder.

Options

Cast Type Menu to choose cast type (target shape): *Sphere*, *Cylinder* or *Cuboid*.

X, Y, Z Toggle buttons to enable/disable the modifier in the X, Y, Z axes directions (X and Y only for *Cylinder* cast type).

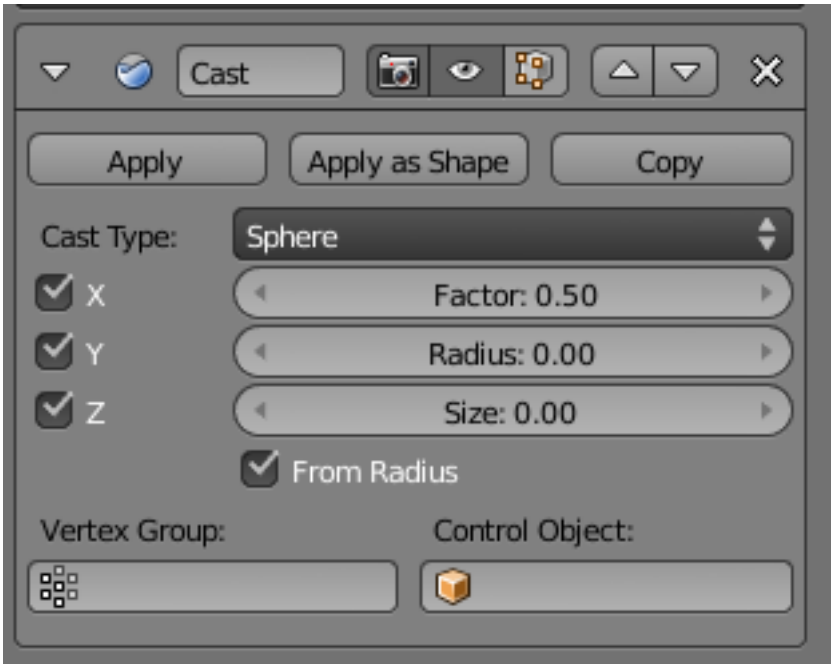


Fig. 2.859: Cast Modifier.

Factor The factor to control blending between original and cast vertex positions. It is a linear interpolation: 0.0 gives original coordinates (i.e. modifier has no effect), 1.0 casts to the target shape. Values below 0.0 or above 1.0 exaggerate the deformation, sometimes in interesting ways.

Radius If non-zero, this radius defines a sphere of influence. Vertices outside it are not affected by the modifier.

Size Alternative size for the projected shape. If zero, it is defined by the initial shape and the control object, if any.

From radius If activated, calculate *Size* from *Radius*, for smoother results.

Vertex Group A vertex group name, to restrict the effect to the vertices in it only. This allows for selective, real-time casting, by painting vertex weights.

Control Object The name of an object to control the effect. The location of this object's center defines the center of the projection. Also, its size and rotation transform the projected vertices.

Hint: Animating (keyframing) this control object also animates the modified object.

Example

Fig. 2.860: Top: Suzanne without modifiers. Middle: Suzanne with each type of Cast Modifier (Sphere, Cylinder and Cuboid). Bottom: Same as above, but now only X axis is enabled. Sample blend-file.

Corrective Smooth

This modifier is used to reduce highly distorted areas of a mesh by smoothing the deformations.

This is typically useful *after* an armature modifier, where distortion around joints may be hard to avoid, even with careful weight painting.

To use this modifier effectively, it is useful to understand the basics of how it works.

Rest State Used as a reference to detect highly distorted areas. The original vertex locations are used by default.

Smoothing Many options for this modifier relate to smoothing which is used internally to correct the distorted regions.

Options

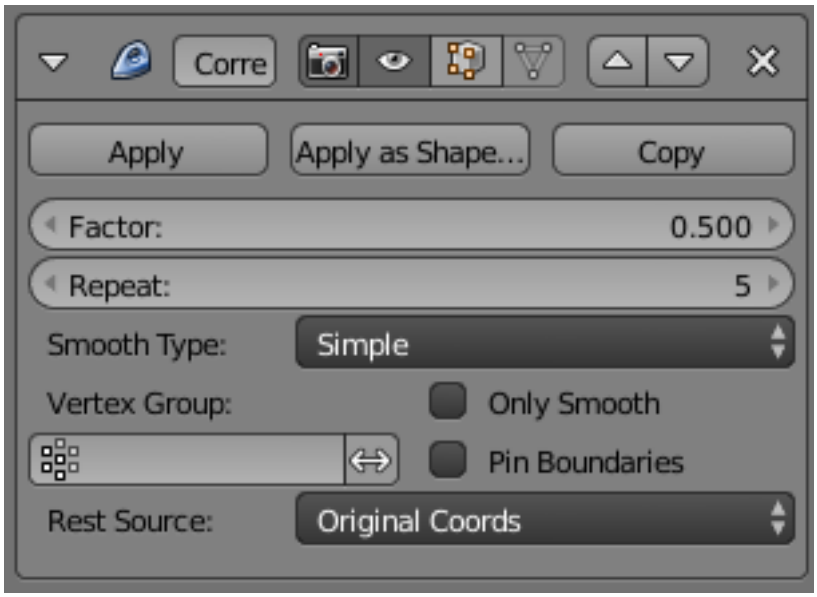


Fig. 2.861: Corrective smooth modifier.

The modifier also uses a *Rest* state, to use as a reference Internally this modifier uses smoothing, so some of the options adjust the kind of smoothing.

Factor The factor to control the smoothing amount. Higher values will increase the effect. Values outside this range (above 1.0 or below 0.0) distort the mesh.

Repeat The number of smoothing iterations. Higher values generally improve the quality of the smoothing, but the operation is slowed down.

Smooth Type Select the smoothing method used.

Simple This simply relaxes vertices to their connected edges.

Length Weight Uses a method of relaxing that weights by the distance of surrounding vertices. This option can give higher quality smoothing in some cases, by better preserving the shape of the original form.

Vertex Group Use to manually select regions to smooth.

Only Smooth This option is included to preview the smoothing used, before correction is applied.

Pin Boundaries Prevent boundary vertices from smoothing.

Rest Source Select the source for reference vertex positions that defines the undeformed

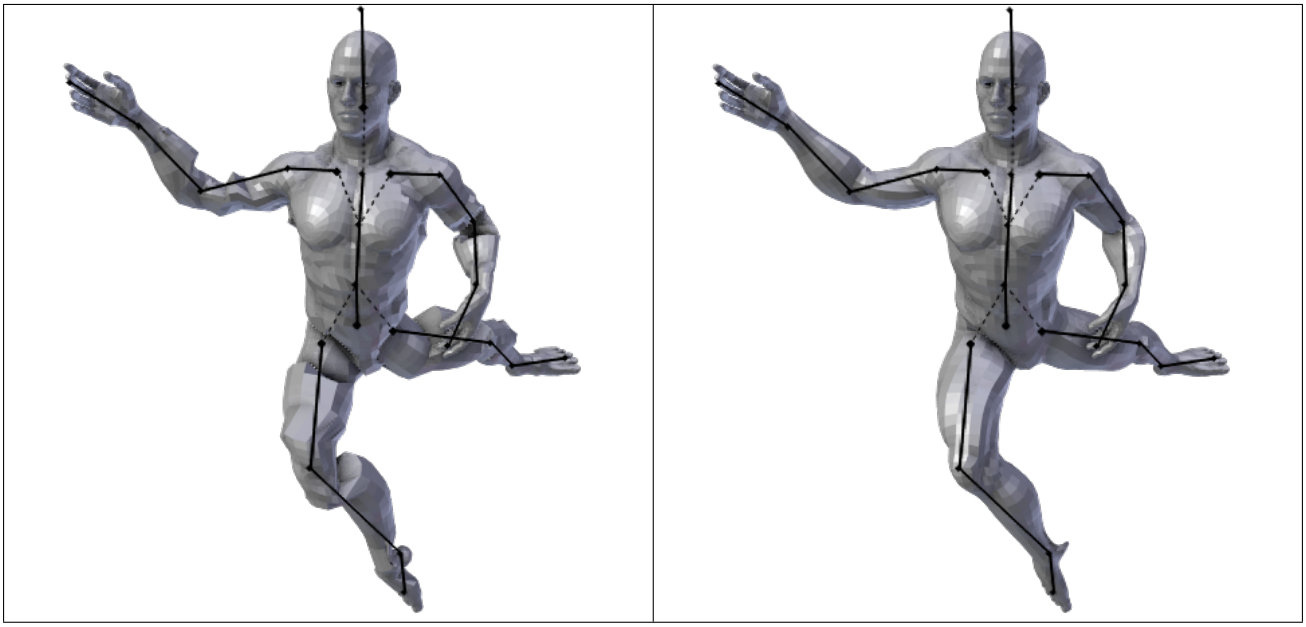
state.

Original Coords Use the original input vertex positions. This relies on the original mesh having the same number of vertices as the original mesh.

Bind Coords Optionally you may bind the modifier to a specific state. This requires that there are constructive modifiers such as Subdivision Surface or Mirror being applied before this modifier in the stack.

Example

Table 2.30: Armature & corrective smooth



Curve Modifier

The Curve Modifier provides a simple but efficient method of deforming a mesh along a curve object.

The Curve Modifier works on a (global) dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction (by default, the X-axis), the mesh will traverse along the curve. Moving the mesh perpendicularly to this axis, the object will move closer or further away from the curve.

When you move the object beyond the curve endings the object will continue to deform based on the direction vector of the curve endings.

Options

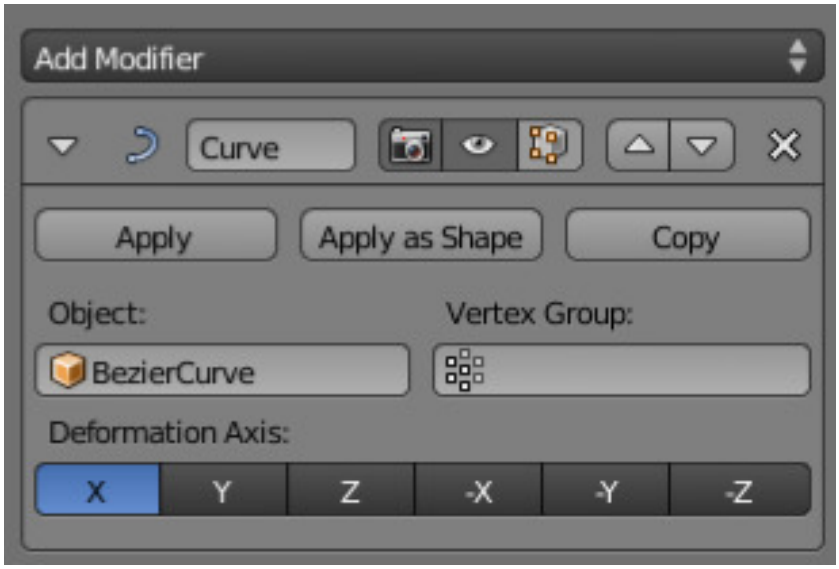


Fig. 2.864: Curve modifier.

Object The name of the curve object that will affect the deformed object.

Vertex Group A vertex group name within the deformed object. The modifier will only affect vertices assigned to this group.

Deformation Axis This is the axis that the curve deforms along.

X, Y, Z, -X, -Y, -Z

Example

Let us make a simple example:

- Remove default cube object from scene and add a Monkey with *Add* → *Mesh* → *Monkey*
- Now add a curve with *Add* → *Curve* → *Bézier Curve*
- While in Edit Mode, move the control points of the curve as shown in Fig. *Edit Curve.*, then exit Edit Mode `Tab`.
- Select the Monkey `RMB` in *Object Mode*
- Assign the curve to the modifier, as shown below. The Monkey should be positioned on the curve:
- Now if you select the Monkey, and move it in the Y-direction `G-Y`, the monkey will deform nicely along the curve.

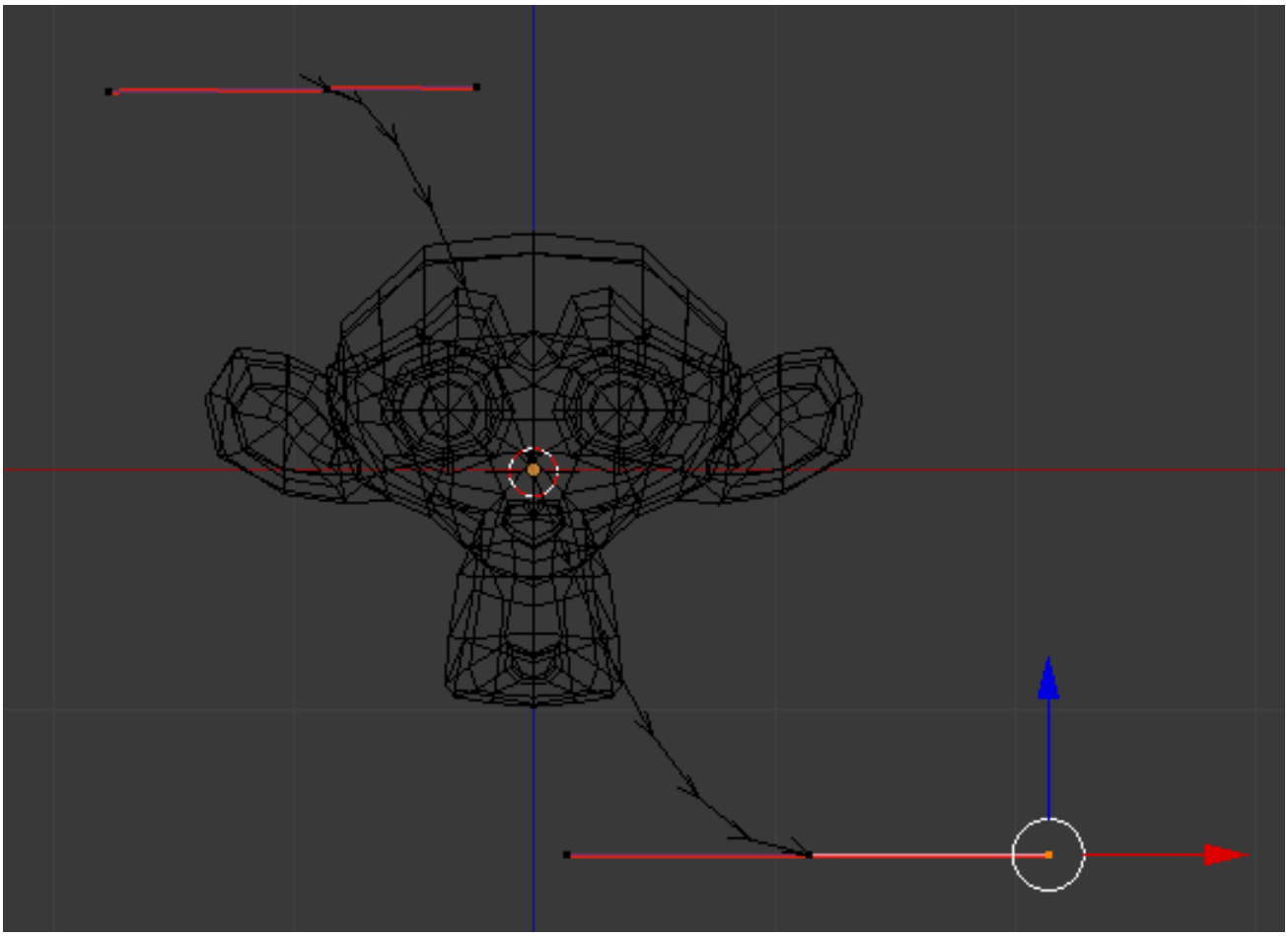


Fig. 2.865: Edit Curve.

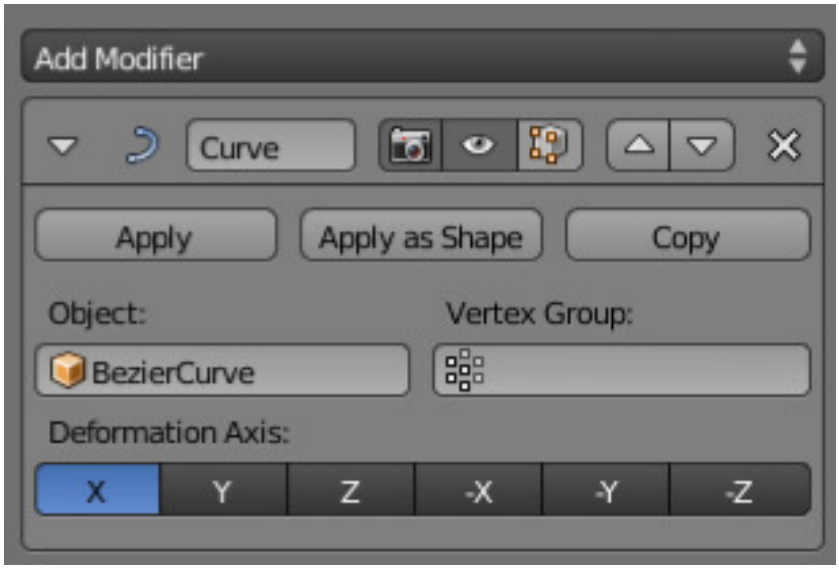


Fig. 2.866: Assign the Bézier curve to the Curve modifier (for Monkey).

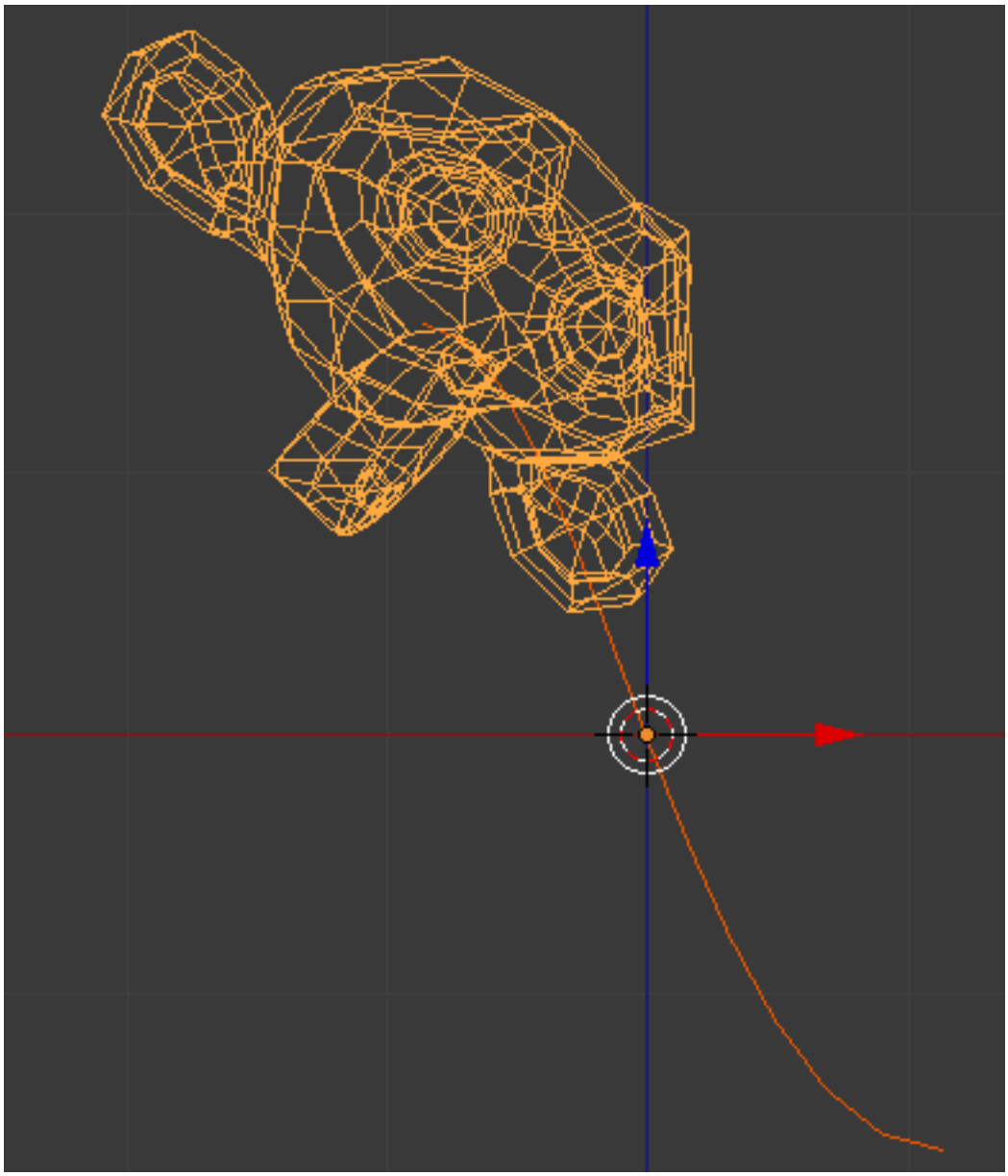


Fig. 2.867: Monkey on a Curve.

Tip: If you press MMB while moving the Monkey you will constrain the movement to one axis only.

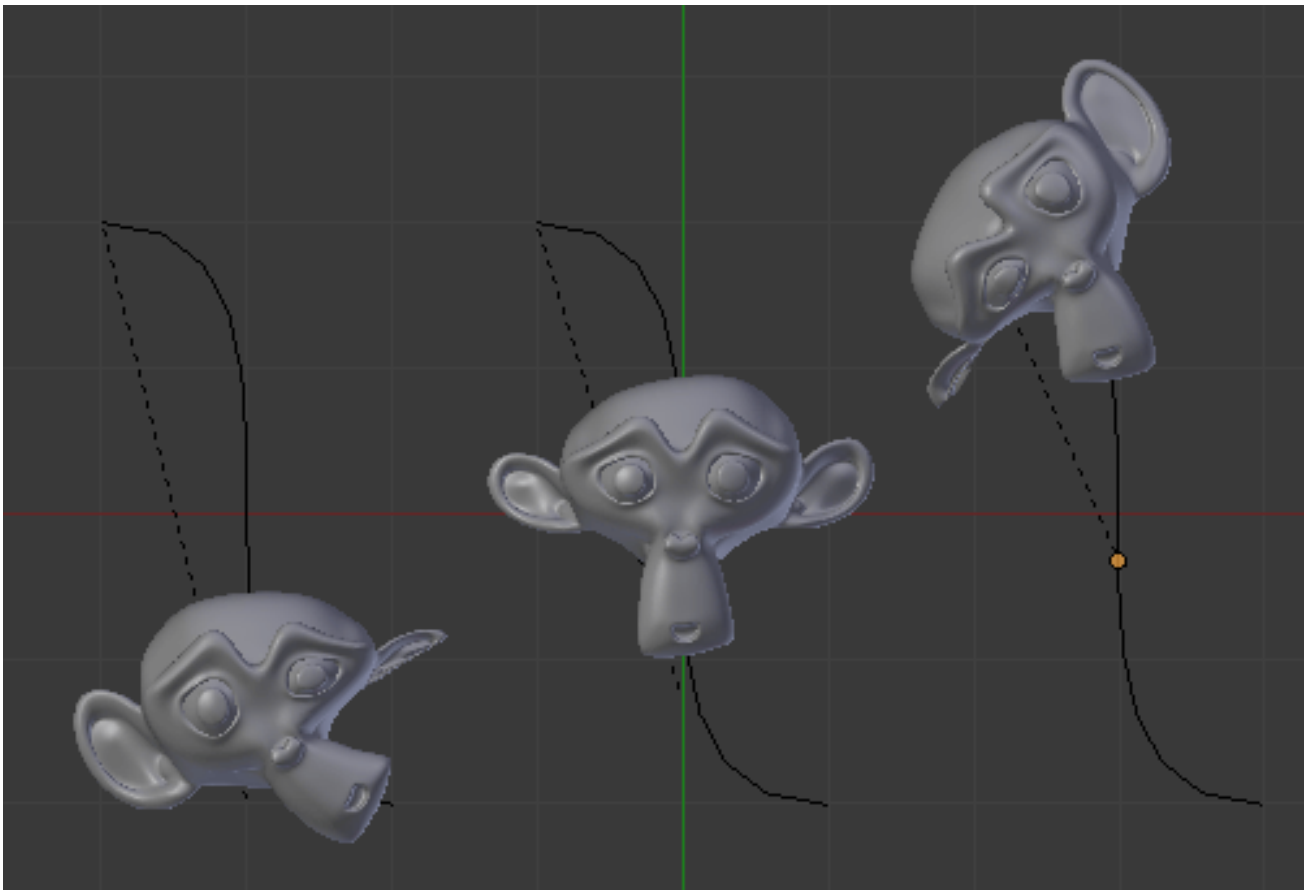


Fig. 2.868: Monkey deformations.

- In the image above you can see the Monkey at different positions along the curve. To get a cleaner view over the deformation, a *Subdivision Surface* modifier with two subdivision levels was applied, and *smooth* shading was used.

Displace Modifier

The Displace modifier displaces vertices in a mesh based on the intensity of a texture. Either procedural or image textures can be used. The displacement can be along a particular local axis, along the vertex normal, or the separate RGB components of the texture can be used to displace vertices in the local X, Y and Z directions simultaneously (sometimes referred to as *Vector Displacement*).

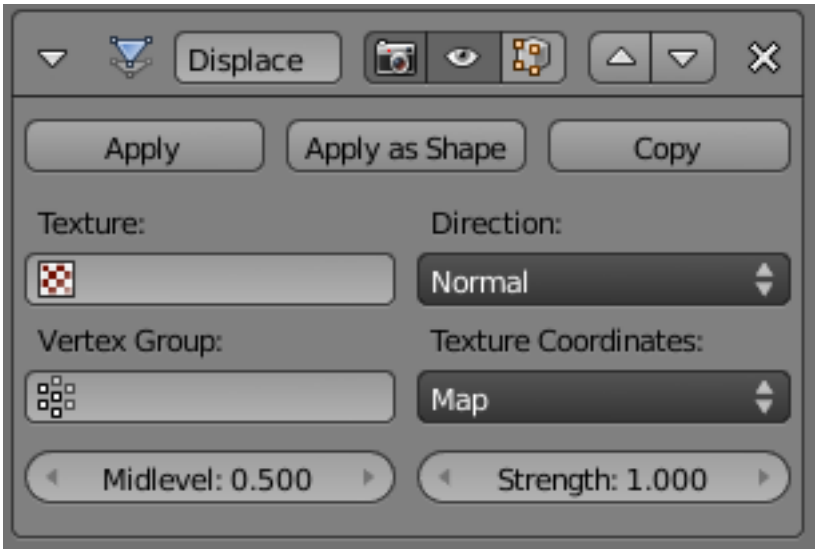


Fig. 2.869: Displace Modifier.

Options

Texture The name of the texture from which the displacement for each vertex is derived. If this field is empty, the modifier defaults to 1.0 (white).

Direction The direction along which to displace the vertices. Can be one of the following:

X, Y, Z Displace along a local axis.

Normal Displace along vertex normal.

RGB to XYZ Displace along local XYZ axes individually using the RGB components of the texture (Red values displaced along the X-axis, Green along the Y, Blue along the Z). This is sometimes referred to as *Vector Displacement*.

Texture Coordinates The texture coordinate system to use when retrieving values from the texture for each vertex. Can be one of the following:

UV Take texture coordinates from face UV coordinates.

UV Map The UV coordinate layer from which to take texture coordinates. If the object has no UV coordinates, it uses the *Local* coordinate system. If this field is blank, but there is a UV coordinate layer available (e.g. just after adding the first UV layer to the mesh), it will be overwritten with the currently active UV layer.

Note: Since UV coordinates are specified per face, the UV texture coordinate system

currently determines the UV coordinate for each vertex from the first face encountered which uses that vertex; any other faces using that vertex are ignored. This may lead to artifacts if the mesh has non-contiguous UV coordinates.

Object Take the texture coordinates from another object's coordinate system (specified by the *Object* field).

Object The object from which to take texture coordinates. Moving the object will therefore alter the coordinates of the texture mapping.

Take note that moving the original object will **also** result in a texture coordinate update. As such, if you need to maintain a displacement coordinate system while moving the modified object, consider parenting the coordinate object to the modified object.

If this field is blank, the *Local* coordinate system is used.

Global Take the texture coordinates from the global coordinate system.

Local Take the texture coordinates from the object's local coordinate system.

Vertex Group The name of a vertex group which is used to control the influence of the modifier. If left empty, the modifier affects all vertices equally.

Midlevel The texture value which will be treated as no displacement by the modifier. Texture values below this value will result in negative displacement along the selected direction, while texture values above this value will result in positive displacement.

displacement = *texture_value* - *Midlevel*

Recall that color/ luminosity values are typically between (0.0 to 1.0) in Blender, and not between (0 to 255).

Strength The strength of the displacement. After offsetting by the *Midlevel* value, the displacement will be multiplied by the *Strength* value to give the final vertex offset.

vertex_offset = *displacement* × *Strength*.

A negative strength can be used to invert the effect of the modifier.

Example

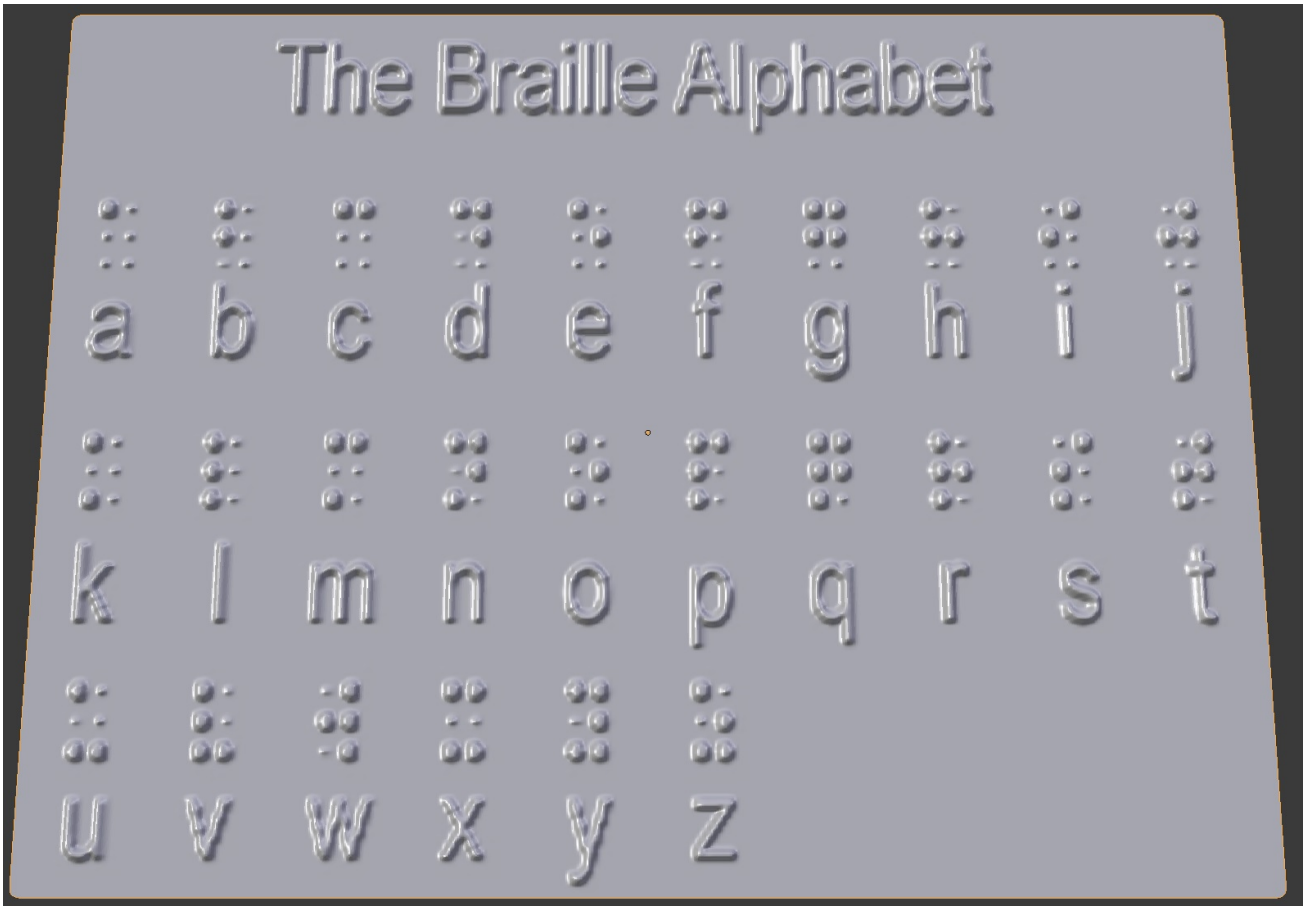


Fig. 2.870: A highly subdivided plane with an image of the Braille alphabet used as the displacement texture.

Hook Modifier

The Hook modifier is used to deform a *Mesh*, *Curve* a *Lattice* using another object (usually an *Empty* or a *Bone* but it can be any object).

As the hook moves, it pulls vertices from the mesh with it. You can think of it as animated *proportional editing*.

While hooks do not give you the fine control over vertices movement that shape keys do, they have the advantage that you can grab vertices directly for manipulation.

Options

Object The name of the object to hook vertices to.

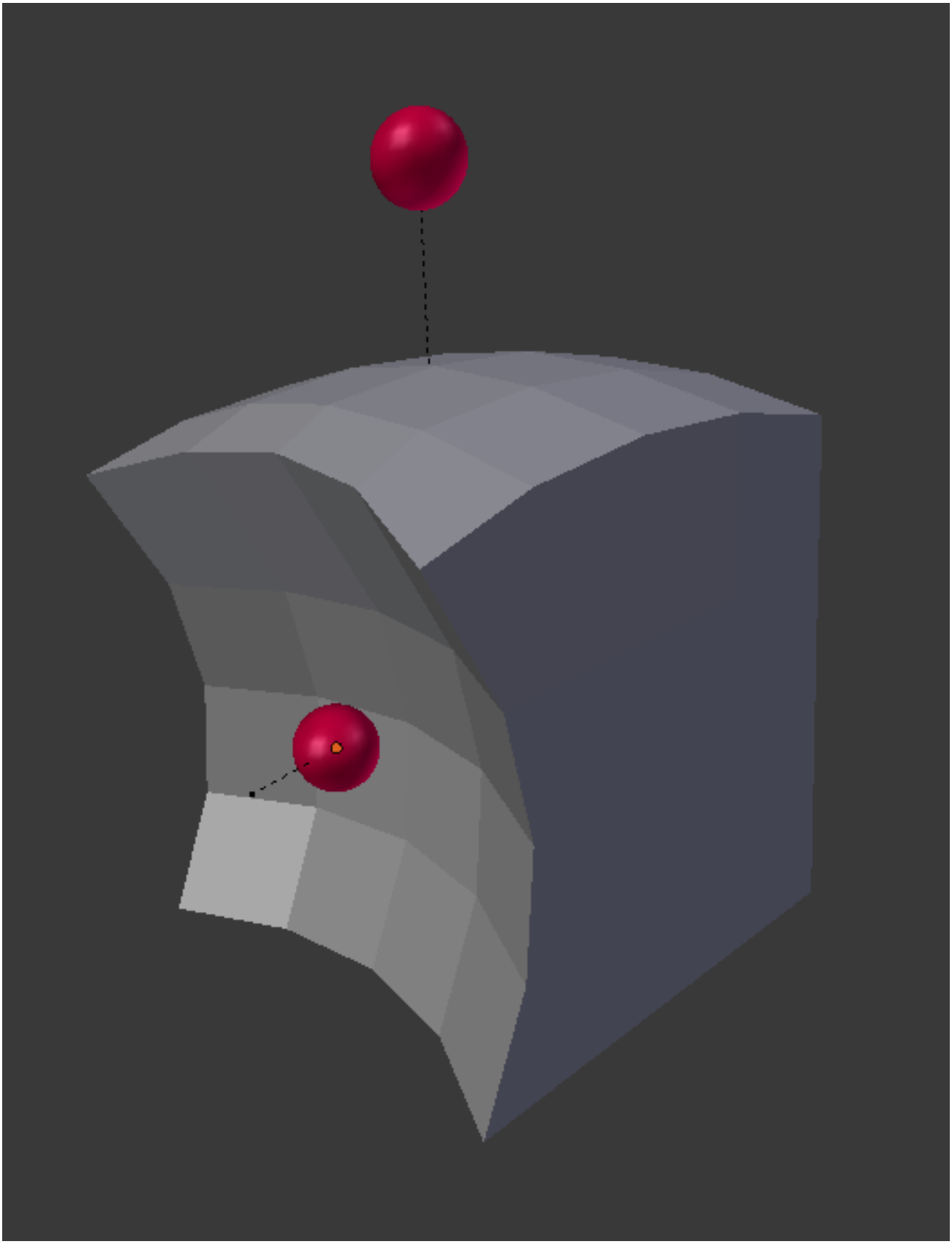


Fig. 2.871: Two spheres used as Hooks to deform a subdivided cube.

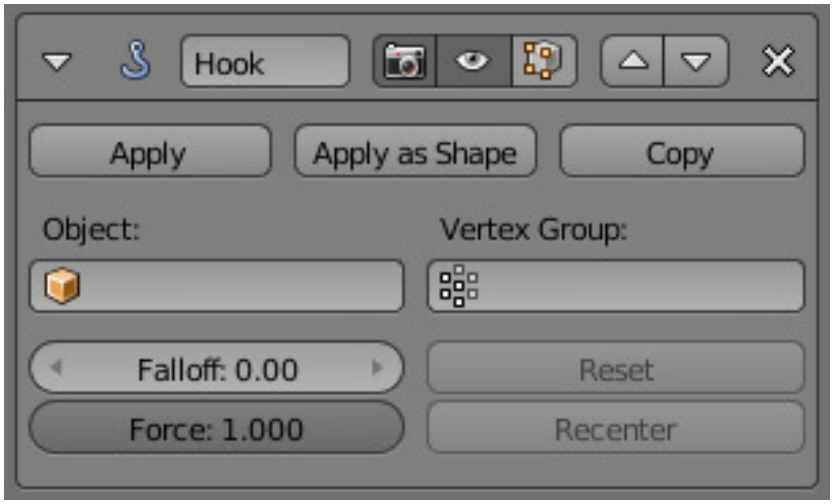


Fig. 2.872: Hook modifier.

Vertex Group Allows you to define the influence per-vertex.

Useful when you do not something other than a spherical field of influence.

Radius The size of the hooks influence.

Strength Adjust this hooks influance on the vertices, were (0.0 to 1.0) (no change to fully follows the hook).

Since multiple hooks can work on the same vertices, you can weight the influence of a hook using this property.

Falloff Type This can be used to adjust the kind of curve that the hook has on the mesh, You can also define a custom-curve to get a much higher level of control.

Uniform Falloff This setting is useful when using hooks on scaled objects, especially in cases where non-uniform scale would stretch the result of the hook.

This is especially useful for lattices, where its common to use non-uniform scaling.

The following settings are only available in Edit Mode:

Reset Recalculate and clear the offset transform of hook.

Recenter Set hook center to the 3D cursor position.

Select Select the vertices affected by this hook.

Assign Assigns selected vertices to this hook.

Note: The Hook Modifier stores vertex indices from the original mesh to determine what to effect; this means that modifiers that generate geometry, like a Subdivision Surface modifier, should always be applied **after** the hook modifier;

otherwise the generated geometry will be left out of the hook's influence.

Laplacian Smooth Modifier

The Laplacian Smooth modifier allows you to reduce noise on a mesh's surface with minimal changes to its shape.

It can also exaggerate the shape using a negative *Factor*.

The Laplacian Smooth is useful for objects that have been reconstructed from the real world and contain undesirable noise. It removes noise while still preserving desirable geometry as well as the shape of the original model.

The Laplacian Smooth modifier is based on a curvature flow Laplace Beltrami operator in a diffusion equation.

Options

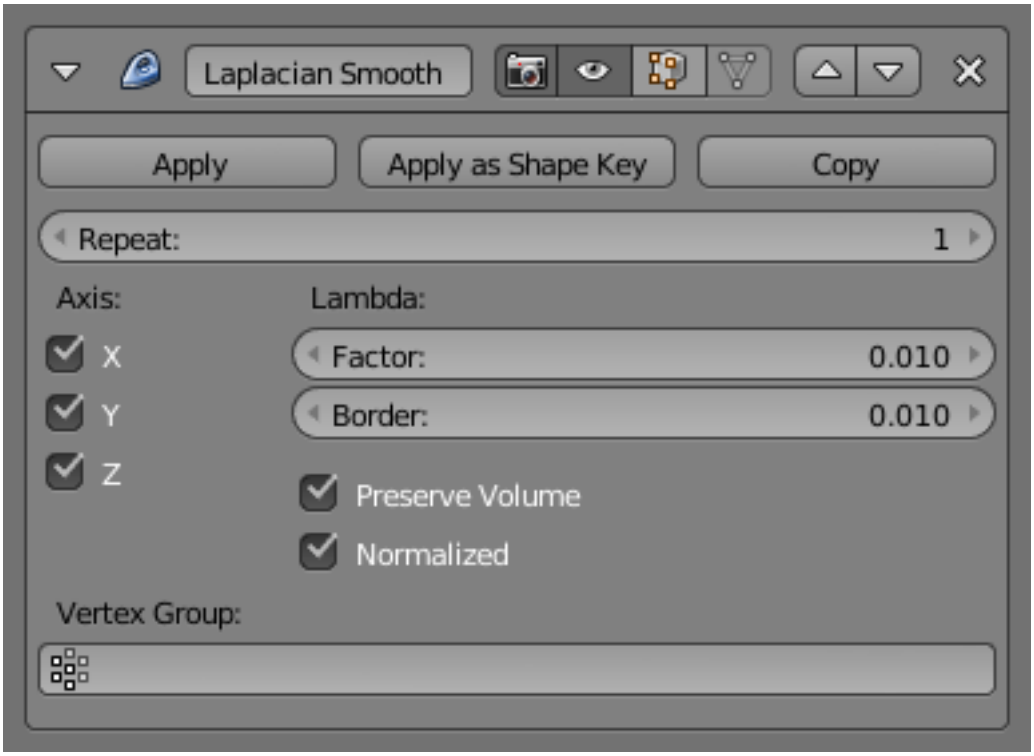


Fig. 2.873: Laplacian Smooth modifier.

Repeat Repetitions allow you to run the Laplacian smoothing multiple times. Each repetition causes the flow curvature of the mesh to be recalculated again, and as a result it removes more noise with every new iteration using a small *Factor* < 1.0 .

When on 0, no smoothing is done.

Note: More repetitions will take longer to calculate. So beware of doing so on meshes with a large number of vertices.

Table 2.31: Repeat: 10.

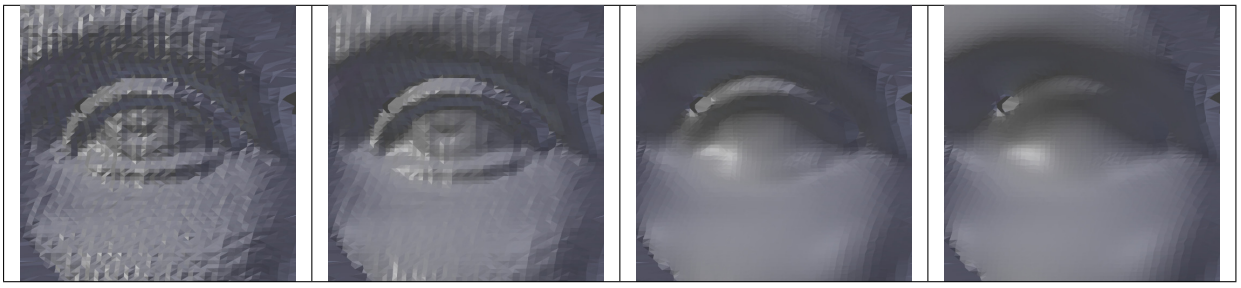


Table 2.32: Repeat: 10.

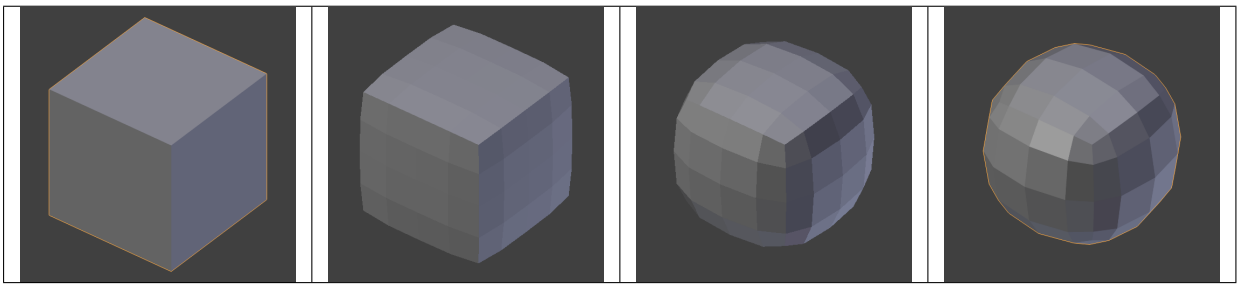
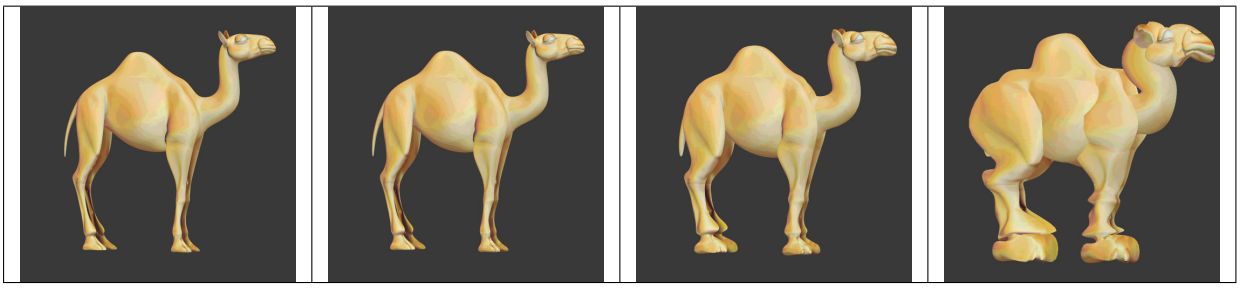


Table 2.33: Repeat: 10.



Factor Controls the amount of displacement of every vertex along the curvature flow.

- Using a small *Factor*, you can remove noise from the shape without affecting desirable geometry.
- Using a large *Factor*, you get smoothed versions of the shape at the cost of fine geometry details.
- Using a negative *Factor*, you can enhance the shape, preserving desirable geometry.
- When the *Factor* is negative, multiple iterations can magnify the noise.

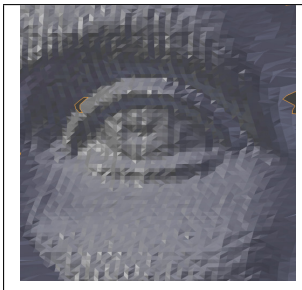


Fig. 2.886: Factor: 0.0.

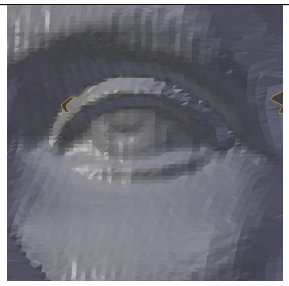


Fig. 2.887: Factor: 0.5.

Fig. 2.888: Factor: 2.5.

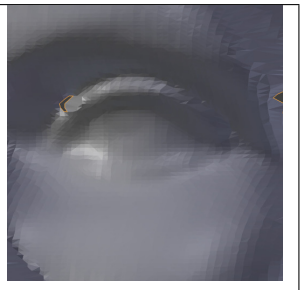


Fig. 2.889: Factor: 5.0.

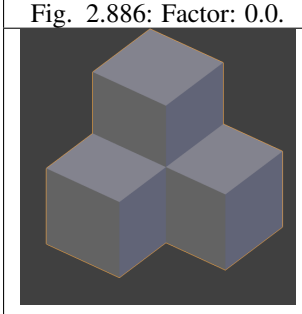


Fig. 2.890: Factor: 0.0.

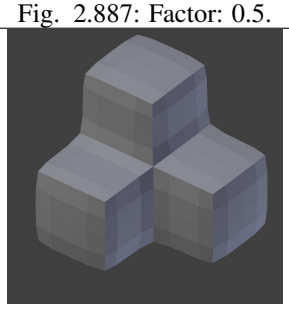


Fig. 2.891: Factor: 1.0.

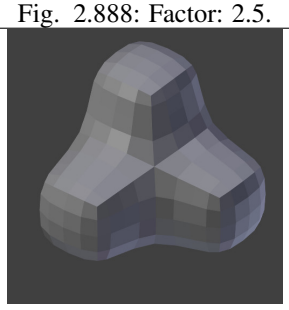


Fig. 2.892: Factor: 10.0.

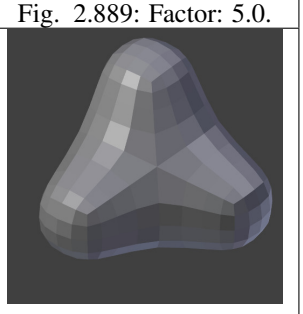


Fig. 2.893: Factor: 50.0.

Fig. 2.894: Factor: 0.0.

Fig. 2.895: Factor: -20.0.

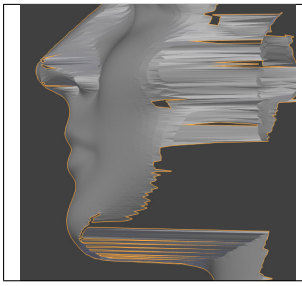
Fig. 2.896: Factor: -50.0.

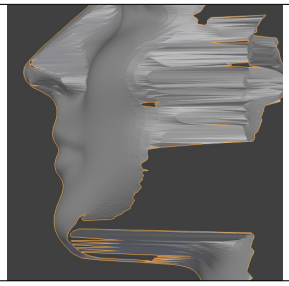
Fig. 2.897: Factor: -300.0.

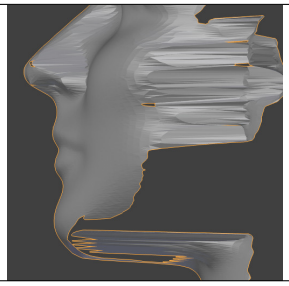
Border Since there is no way to calculate the curvature flow on border edges, they must be controlled separately. Border edges are smoothed using a much simpler method, using this property to control the influence.

Positive values will smooth the vertex positions, while negative values will "enhance" them by transforming them in the opposite direction.

Table 2.34: Border: 10.0.







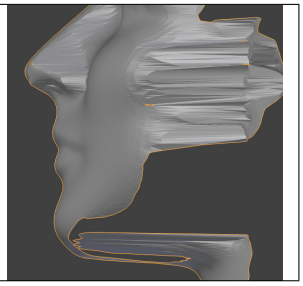


Table 2.35: Border: 20.0.

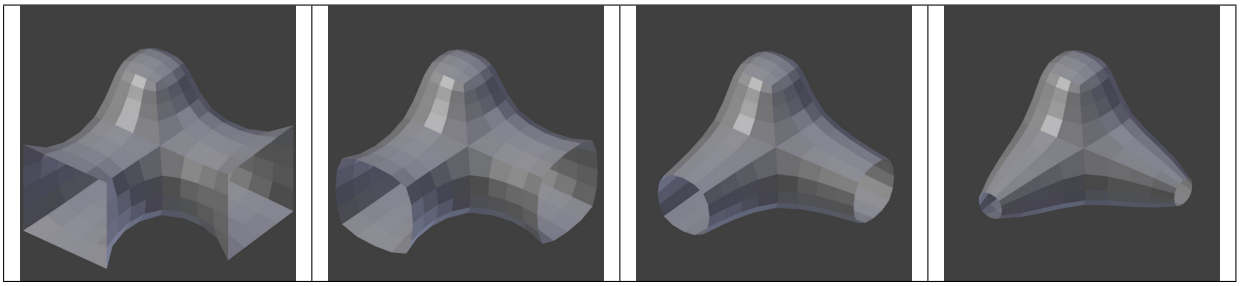
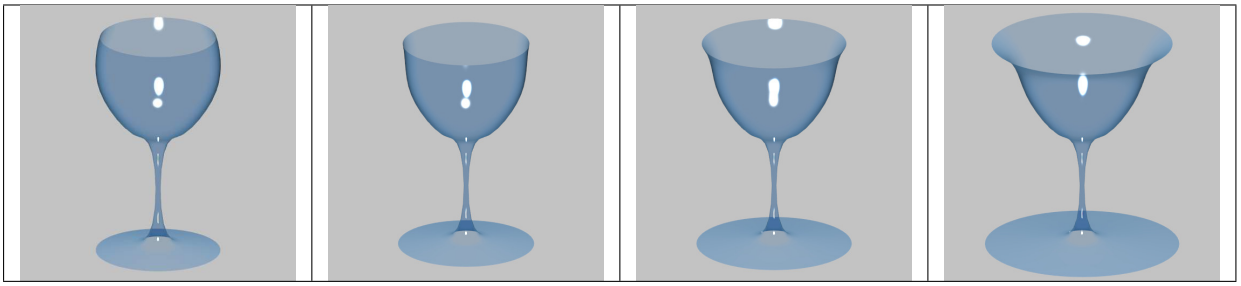


Table 2.36: Border: -200.0.



X, Y, Z Toggle buttons to enable/disable deforming vertices in the X, Y and/or Z axis directions.

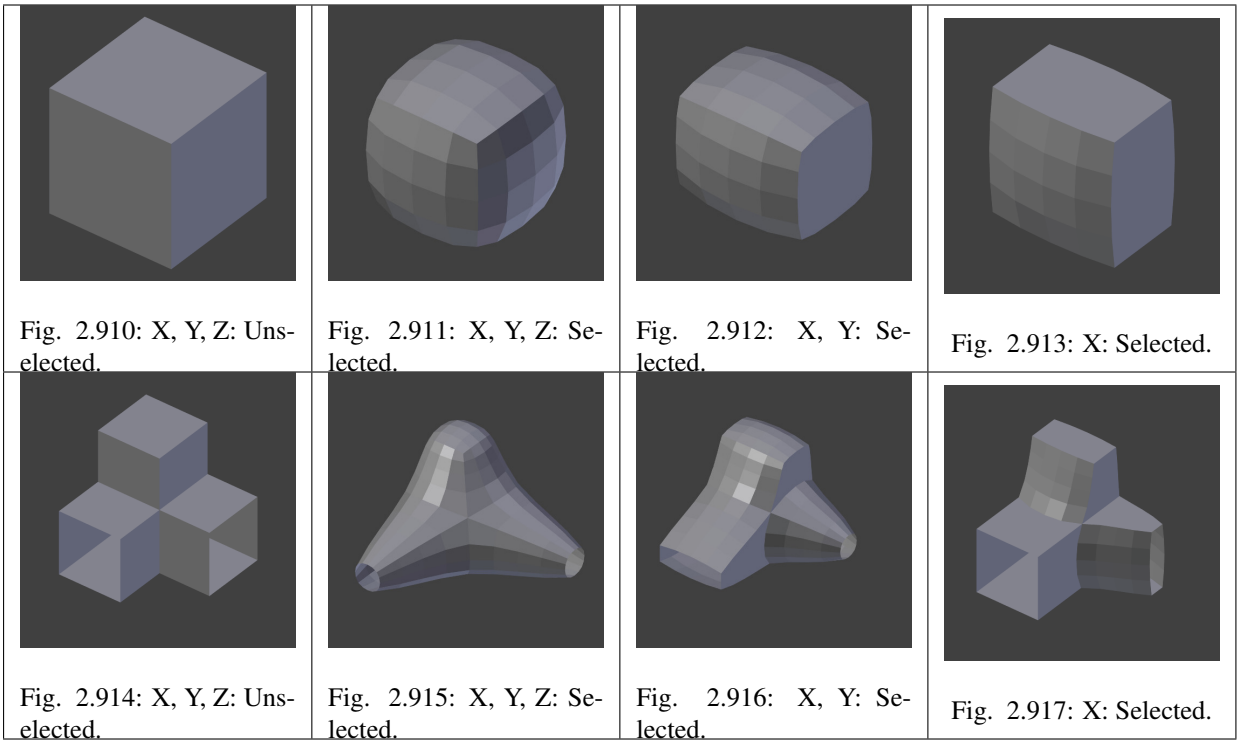


Fig. 2.910: X, Y, Z: Unselected.

Fig. 2.911: X, Y, Z: Selected.

Fig. 2.912: X, Y: Selected.

Fig. 2.913: X: Selected.

Fig. 2.914: X, Y, Z: Unselected.

Fig. 2.915: X, Y, Z: Selected.

Fig. 2.916: X, Y: Selected.

Fig. 2.917: X: Selected.

Preserve Volume The smoothing process can produce shrinkage. That is significant for large *Factor* or large *Repeat* values; to reduce that effect you can use this option.

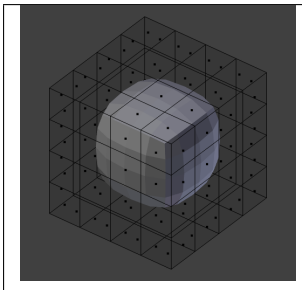


Fig. 2.918: Off.

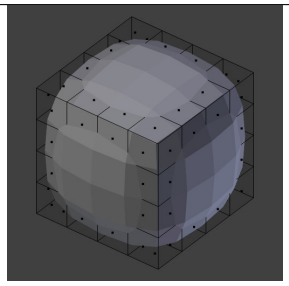


Fig. 2.919: On.

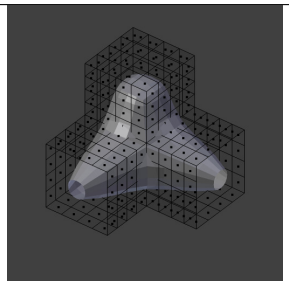


Fig. 2.920: Off.

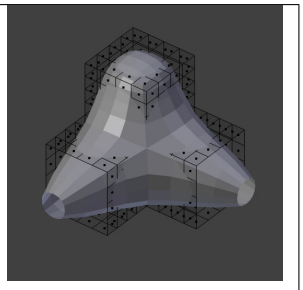
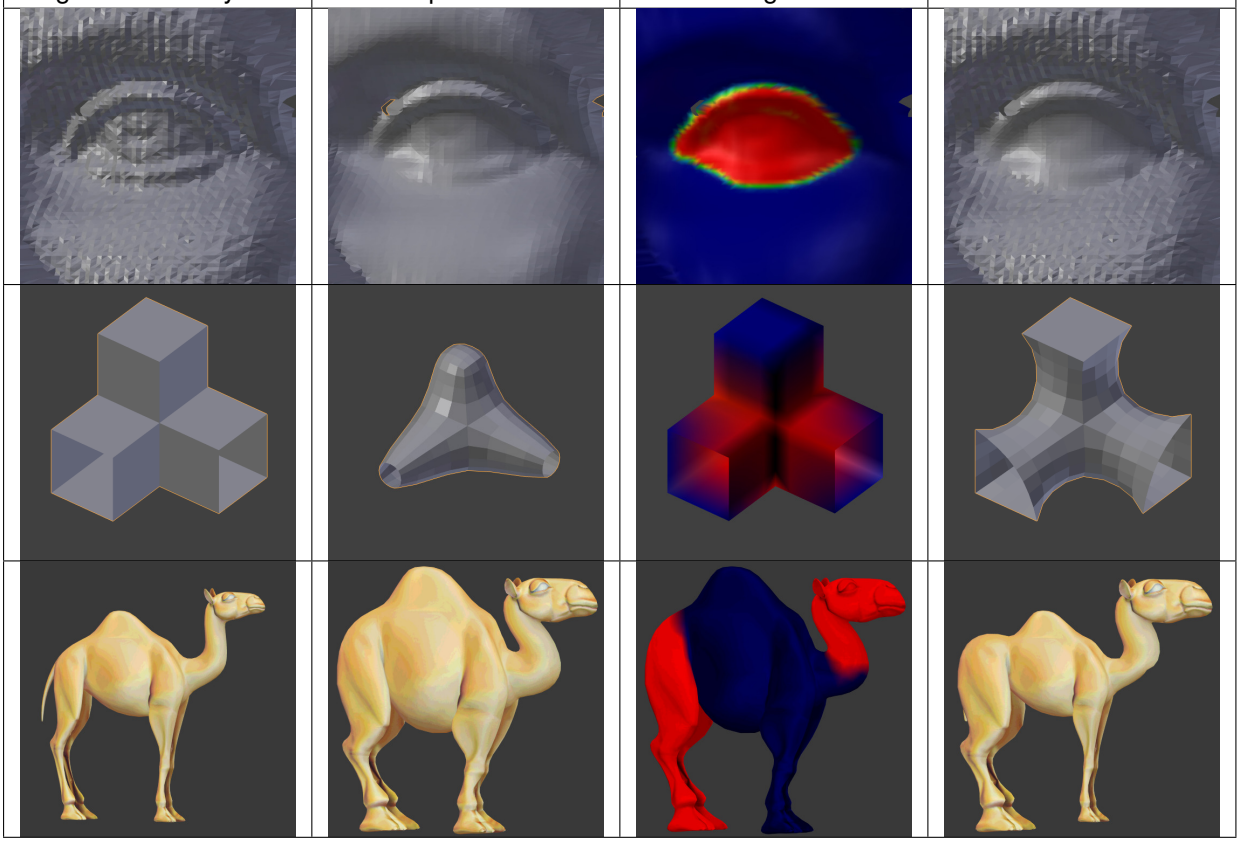


Fig. 2.921: On.

Vertex Group A vertex group name, to constrain the effect to a group of vertices only. Allows for selective, real-time smoothing or enhancing, by painting vertex weights.



Normalized When enabled, the results will depend on face sizes. When disabled, geometry spikes may occur.

Fig. 2.922: Original Geometry.

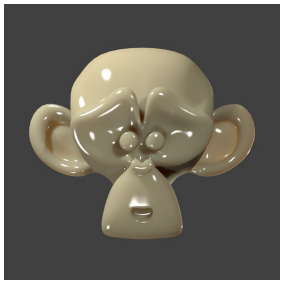


Fig. 2.923: On.

Fig. 2.924: Off.

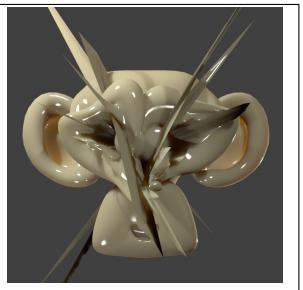


Fig. 2.925: Off, High Factor.

Hint: Meshes with a great number of vertices, more than ten thousand (10,000), may take several minutes for processing; you can use small portions of the mesh for testing before executing the modifier on the entire model.

Examples

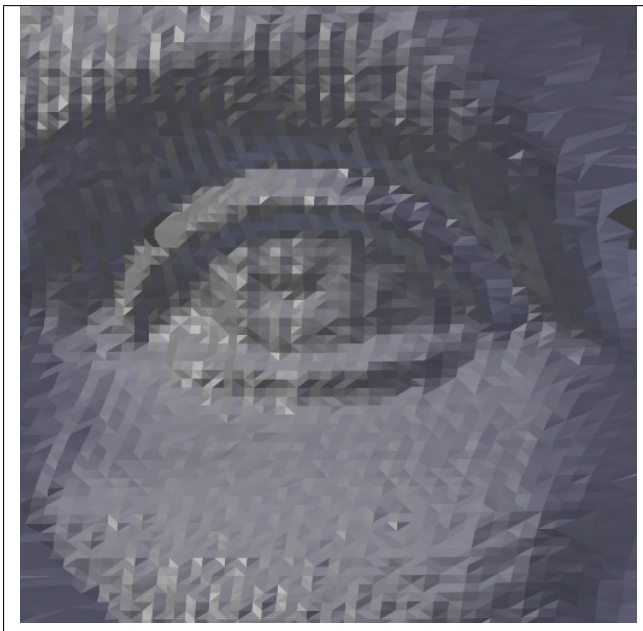


Fig. 2.926: Femme Front blend-file.

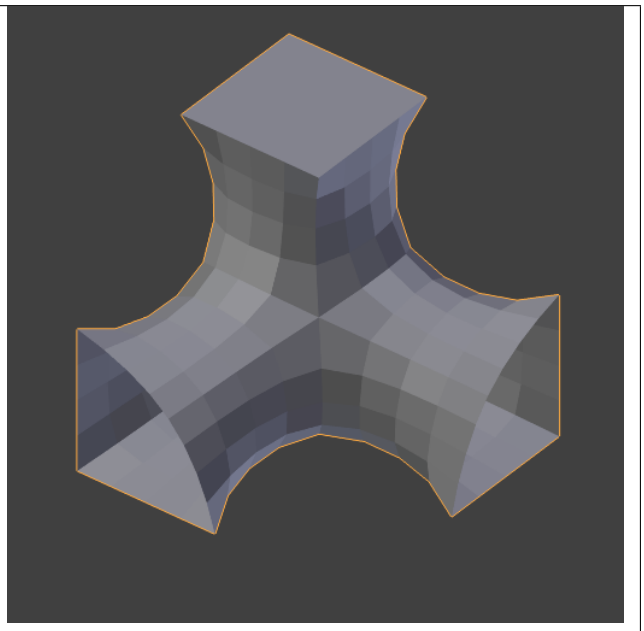


Fig. 2.927: Cube Smooth blend-file.

See also:

Smooth Modifier.

Laplacian Deform Modifier

The Laplacian Deform modifier allows you to pose a mesh while preserving geometric details of the surface.

The user defines a set of 'anchor' vertices, and then moves some of them around. The modifier keeps the rest of the anchor vertices in fixed positions, and calculates the best possible locations of all the remaining vertices to preserve the original geometric details.

This modifier captures the geometric details with the uses of differential coordinates. The differential coordinates captures the local geometric information how curvature and direction of a vertex based on its neighbors.

Note: You must define a *Anchors Vertex Group*. Without a vertex group modifier does nothing.

Options

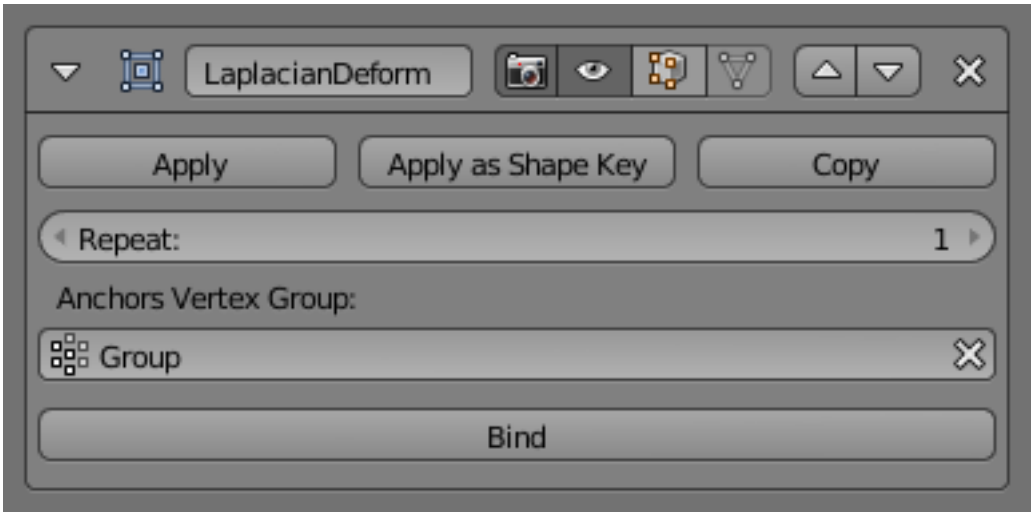


Fig. 2.928: Laplacian Deform modifier

Repeat Repetitions iteratively improve the solution found. The objective is to find the rotation of the differential coordinates preserving the best possible geometric detail. Details are retained better if more repetitions are used, however, it will take longer to calculate.

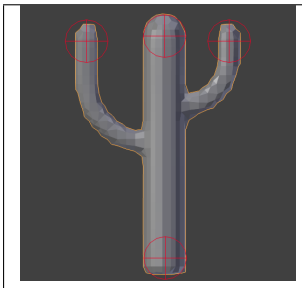


Fig. 2.929: Original Model.

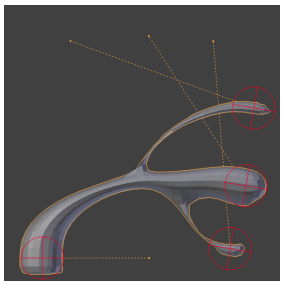


Fig. 2.930: Repeat: 1.

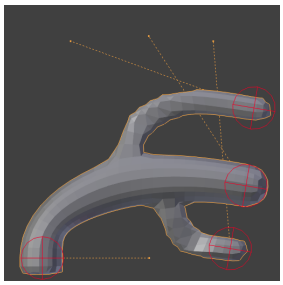


Fig. 2.931: Repeat: 2.

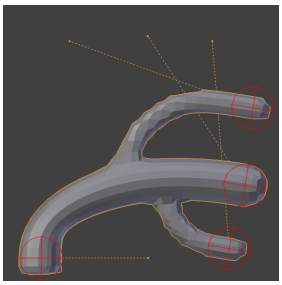


Fig. 2.932: Repeat: 5.

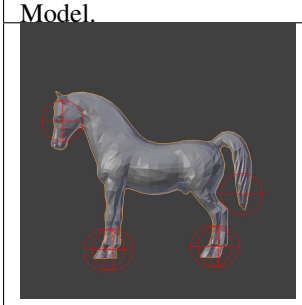


Fig. 2.933: Original Model.

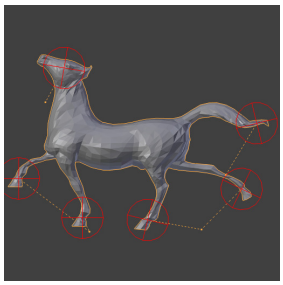


Fig. 2.934: Repeat: 1.

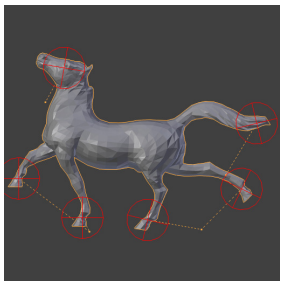


Fig. 2.935: Repeat: 2.

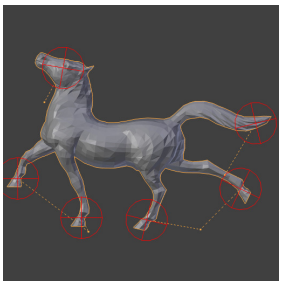


Fig. 2.936: Repeat: 10.

Anchors Vertex Group A vertex group name, to define the group of vertices that the user uses to transform the model. The weight of each vertex does not affect the behavior of the modifier; the method only takes into account vertices with weight greater than

0.

Bind The *Bind* button is what tells the Laplacian Deform modifier to actually capture the geometry details of the object, so that altering the anchors vertices actually alters the shape of the deformed object.

Unbind After binding the modifier, you may later decide to make changes to the Anchors Vertex Group. To do so you will first need to *Unbind* the modifier before binding again.

Error Messages

Vertex group *group_name* is not valid This message is displayed when a user deletes a Vertex Group or when the user changes the name of the Vertex Group.

Verts changed from X to Y This message is displayed when a user add or delete verts to the mesh.

Edges changed from X to Y This message is displayed when a user add or delete edges to the mesh.

The system did not find a solution This message is displayed if the solver SuperLU did not find a solution for the linear system.

Note: If the mesh is dense, with a number of vertices greater than 100,000, then it is possible that the nonlinear optimization system will fail.

History

Laplacian Surface Editing is a method developed by Olga Sorkine and others in 2004. This method preserves geometric details as much as possible while the user makes editing operations. This method uses differential coordinates corresponding to the difference between a vector and the weighted average of its neighbors to represent the local geometric detail of the mesh.

See also:

- Laplacian Surface Editing (Original paper)
- Differential Coordinates for Interactive Mesh Editing

Lattice Modifier

The Lattice modifier deforms the base object according to the shape of a *Lattice* object. Objects to be deformed can be meshes, curves, surfaces, text, lattices and even particles.

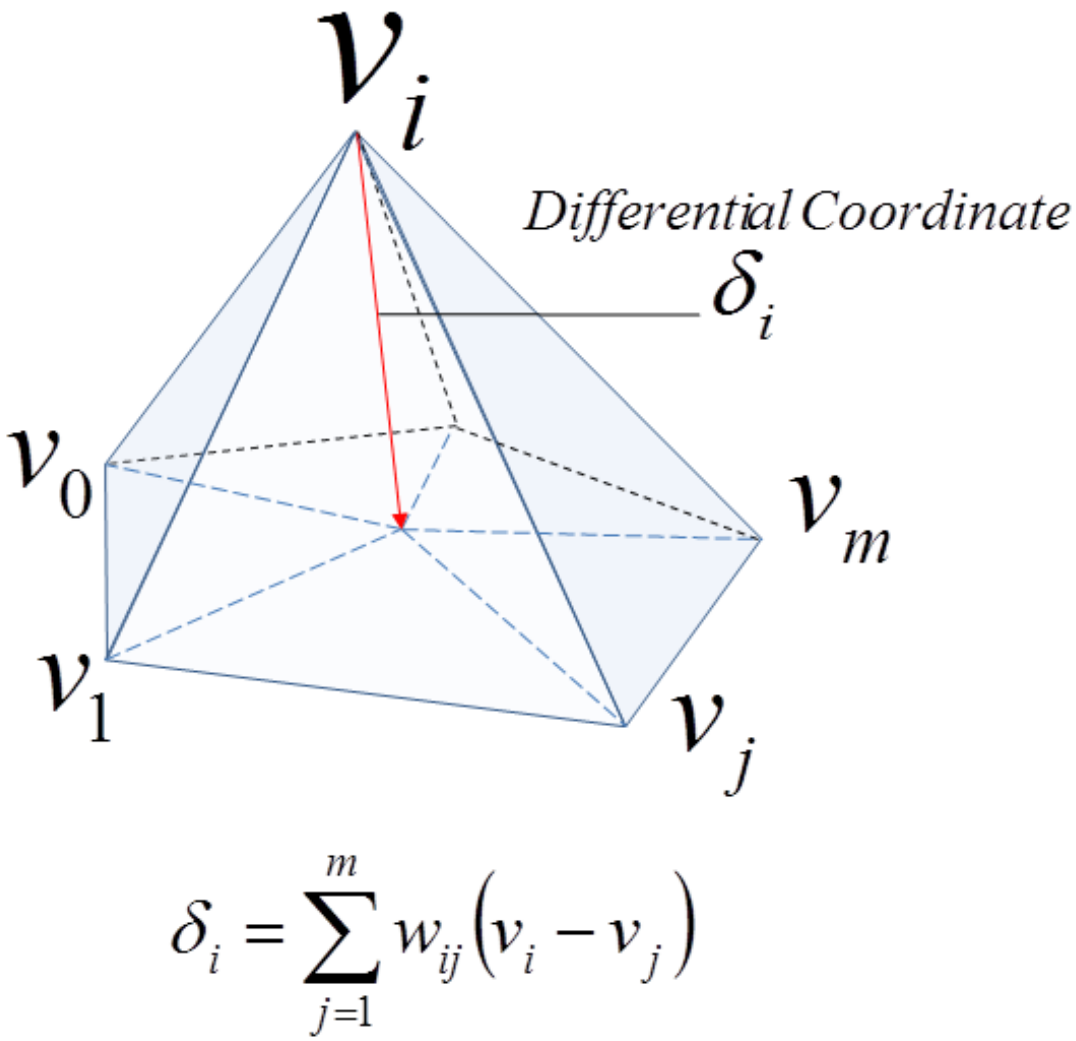


Fig. 2.937: Differential Coordinate.

Tip: A Lattice Modifier can quickly be added to selected objects by selecting them all, then selecting the lattice object last and pressing `Ctrl-P` and choosing *Lattice Deform*. This will both add Lattice modifiers to the selected objects and parent them to the lattice.

Options

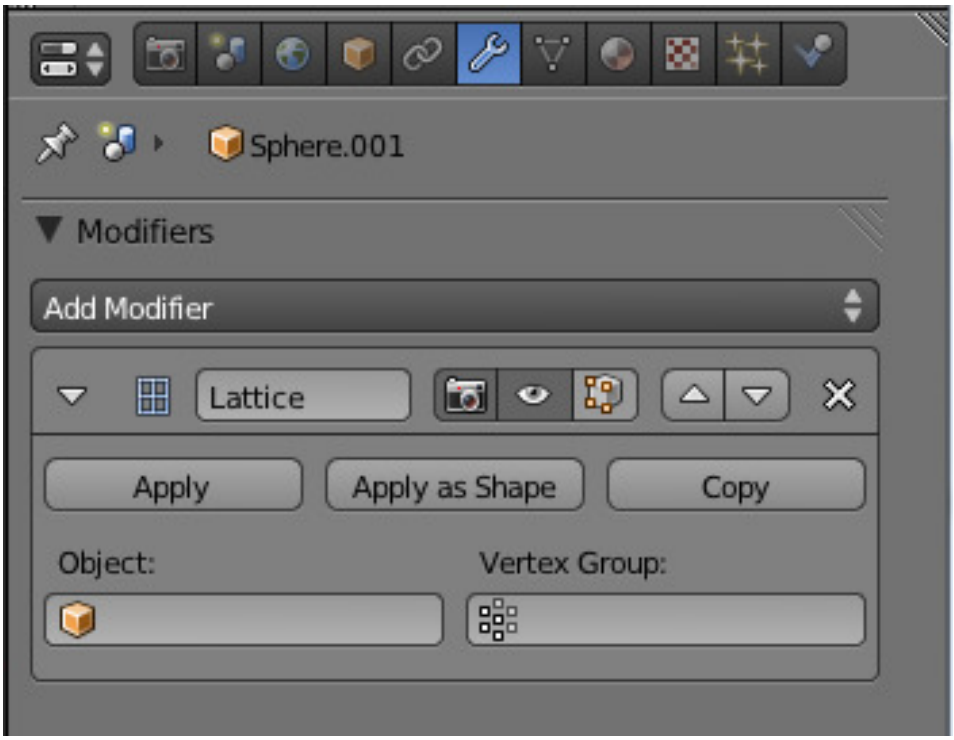


Fig. 2.938: Lattice modifier.

Object The *Lattice* object with which to deform the base object.

Vertex Group An optional vertex group name which lets you limit the modifier effect to a part of the base mesh.

Strength A factor to control blending between original and deformed vertex positions.

Hints

Why would you use a lattice to deform a mesh instead of deforming the mesh itself in Edit Mode ? There are a couple of reasons for that:

- If your object has a large number of vertices, it would be difficult to edit portions of it quickly in Edit Mode. Using a lattice will allow you to deform large portions efficiently.

- The smooth deformation you get from a lattice modifier can be hard to achieve manually in Edit Mode.
- Multiple objects can use the same lattice, thus allowing you to edit multiple objects at once.
- Like all modifiers, it is non-destructive. Meaning all changes happen on top of the original geometry, which you can still go back and edit without affecting the deformation.
- A lattice does not affect the texture coordinates of a mesh's surface.

Note: When using a lattice to deform particles, always remember to place the Lattice Modifier after the Particle System Modifier. Read more about the importance of the modifier stack *Here*.

Mesh Deform Modifier

The Mesh Deform modifier allows an arbitrary mesh (of any closed shape) to act as a deformation cage around another mesh.

Options

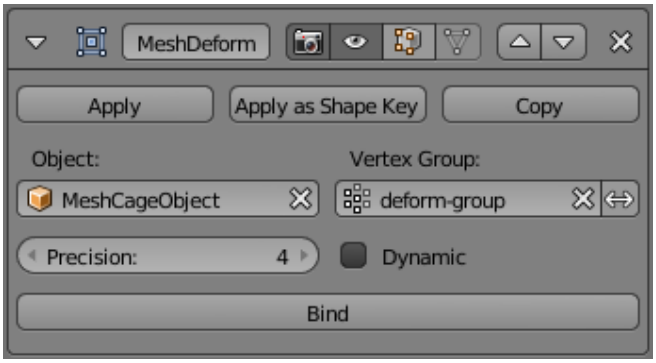


Fig. 2.939: Mesh Deform Modifier.

The Mesh Deform modifier is reasonably easy to use but it can be very slow to do the calculations needed to properly map the deform mesh cage to the deformed object.

Object The name of the mesh object to be used as a deforming mesh cage.

Vertex Group An optional vertex group of the object's mesh to restrict the vertices that will be affected by this modifier. Vertices not in this group will not be deformed.

Invert <–> Inverts the influence of the selected vertex group, meaning that the group now

represents vertices that will not be deformed by the modifier.

(The setting reverses the weight values of the group).

Precision The *Precision* number button controls the accuracy with which the deform mesh cage alters the deformed object, when the points on the cage are moved. Raising this value higher can greatly increase the time it takes the *Mesh Deform* modifier to complete its binding calculations, but it will get more accurate cage mapping to the deformed object.

This setting becomes unavailable once a cage has been bound.

Dynamic When activated, other mesh altering features (such as other modifiers and shape keys) are taken into account when binding, increasing deformation quality.

The setting is deactivated by default to save memory and processing time when binding. Like with *Precision*, this setting is unavailable once a cage has been bound.

Bind Links the current vertex positions of both the modified geometry and the deformer *Object* chosen together. An unbound Mesh Deform modifier has no effect – it must be bound so that altering the shape of the deform mesh cage actually alters the shape of the modified object.

> **Warning:** Depending on the settings of the Mesh Deform modifier and complexity of the deform mesh cage and/or deformed object, it can take a long time for this operation to complete. This can result in Blender not responding to user's actions until it has completed.
>
> It is also possible that Blender will run out of memory and crash.
>
> To be safe, save your blend-file before proceeding!

Unbind When a deformed object has been associated to a deform mesh cage, it can later be disassociated by clicking the *Unbind* button which replaced the *Bind* one. When *Unbind* is clicked, the *deform mesh cage* will keep its current shape; it will not reset itself back to its original start shape.

If you need its original shape, you will have to save a copy of it before you alter it.

The deformed object will, however, reset back to its original shape that it had before it was bound to the deform mesh cage.

Warning: Significant changes to the entire change mesh *(such as rotating the cage upside down)* can cause noticeable artifacts.

These can be reduced by binding with a higher *Precision*, however, it is a known limitation with this modifier and cannot be avoided entirely.

Hints

- Ensure that the normals on the cage mesh point to the outside; they are used to determine the inside and outside of the cage.
- Besides the outer cage, more faces within the cage, either loose or forming another smaller cage, can be used for extra control. Such smaller cages may also overlap with the main cage; for example, to get extra control over eyes, two small sphere cages could be added around them.

See also:

- The *Lattice modifier*.
- Original paper

Shrinkwrap Modifier

The *Shrinkwrap* modifier allows an object to "shrink" to the surface of another object. It moves each vertex of the object being modified to the closest position on the surface of the given mesh (using one of the three methods available).

It can be applied to meshes, lattices, curves, surfaces and texts.

Options

Target Shrink target, the mesh to shrink to/wrap around.

Offset The distance that must be kept from the calculated target position, in Blender Units.

Vertex Group The vertex group to control whether and how much each vertex is displaced to its target position. If a vertex is not a member of this group, it is not displaced (same as weight 0).

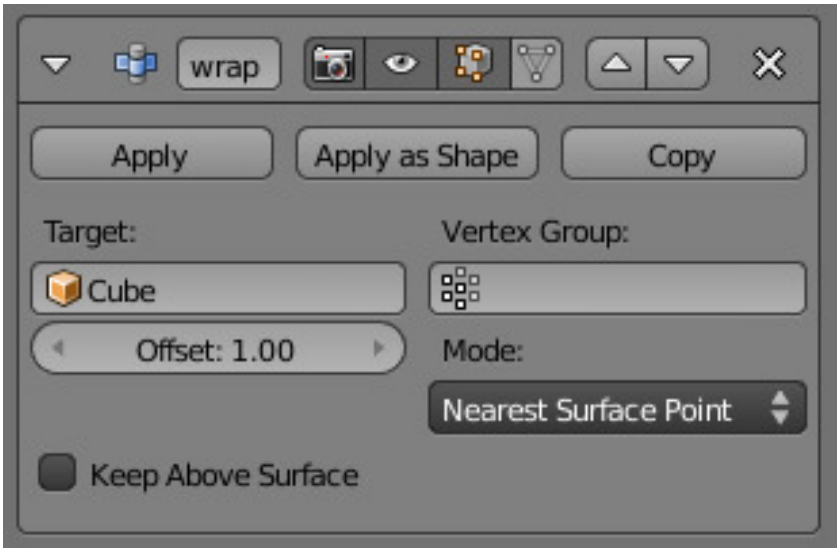


Fig. 2.940: Nearest Surface Point.

Mode

This selector specifies the method to be used to determine the nearest point on the target's surface for each vertex of the modified object. Some options will add some extra, specific controls to the panel.

Nearest Surface Point

This will select the nearest point over the surface of the shrink target. It adds the extra option *Above surface*, which always keep the computed vertices above their "floor faces". This is only meaningful when *Offset* is not null.

Projection

This will project vertices along a chosen axis until they touch the shrink target. Vertices that never touch the shrink target are left in their original position.

Subdivision Surface Levels This applies a (temporary) *Catmull-Clark* subdivision to the modified object, before computing the wrap when using Projection mode.

Limit This is a distance limit between original vertex and surface. If the distance is larger than this limit vertex would not be projected onto the surface,

Axis Along which local axis of the modified object the projection is done. These options

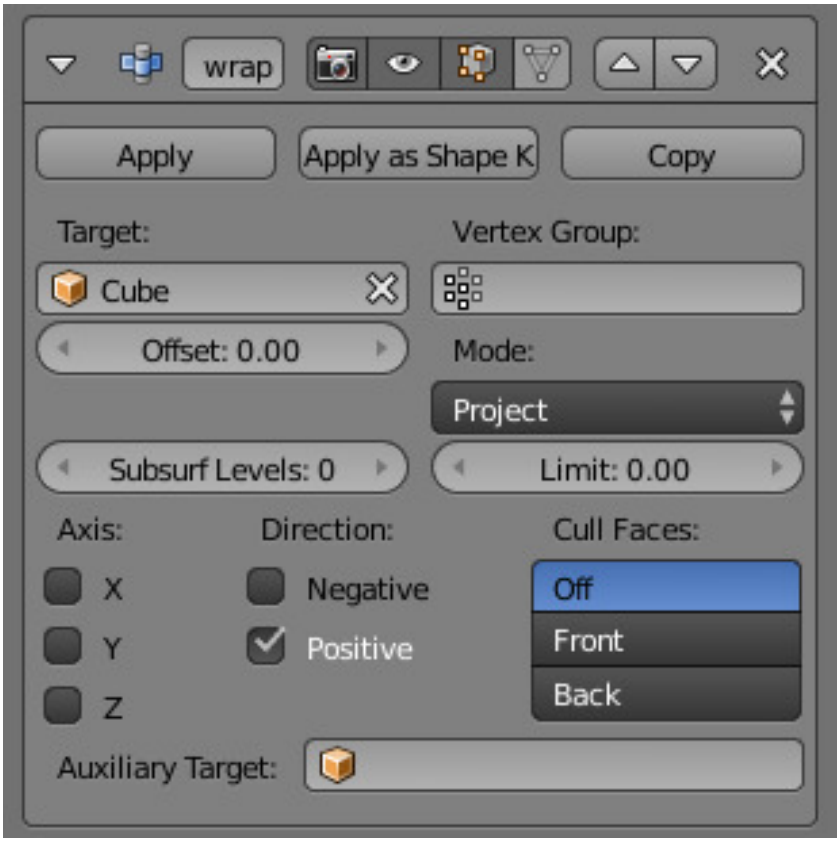


Fig. 2.941: Projection options.

can be combined with each other, yielding a "median axis" of projection.

X, Y, Z

Direction This allows you to select the allowed direction(s) of the shrink along the selected axis. With more than one *Shrinkwrap* modifier, negative and positive axes can be combined.

Negative, Positive

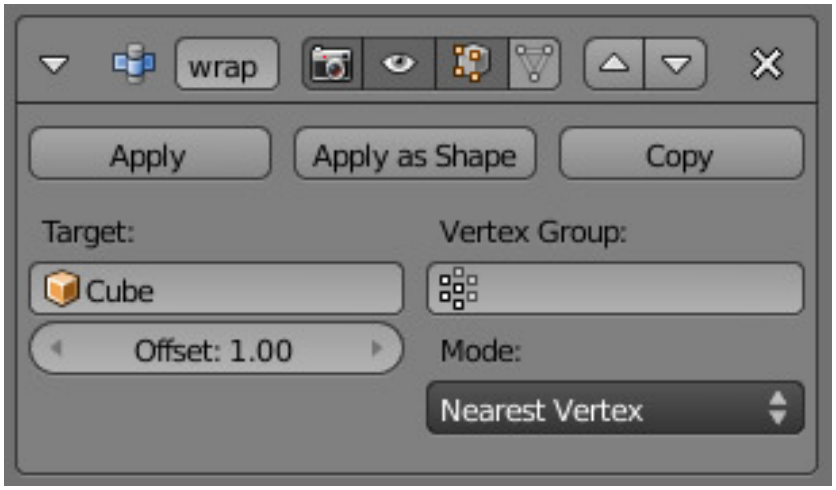
Cull Faces This radio button allows you to prevent any projection over the "front side" (respectively the "back side") of the target's faces. The "side" of a face is determined by its normal (front being the side "from where" the normal "originates").

Auxiliary Target An additional object to project over.

Nearest Vertex

This will snap vertices to the nearest vertex of the shrink target. It adds no extra options.

Nearest Vertex options.



See also:

Shrinkwrap Constraint.

Simple Deform Modifier

The Simple Deform modifier allows easy application of a simple deformation to an object (meshes, lattices, curves, surfaces and texts are supported).

Using another object, it is possible to define the axis and origin of the deformation, allowing application of very different effects.

Options

Mode This radio button defines the deform function applied, among four available:

Twist Rotates around the Z axis.

Bend Bends the mesh over the Z axis.

Taper Linearly scales along Z axis.

Stretch Stretches the object along the Z axis (negative *Factor* leads to squash), preserving volume by scaling inversely on the X and Y axes.

Vertex Group The name of the vertex group that indicates whether and how much each vertex is influenced by the deformation.

Origin The name of an object that defines the origin of deformation (usually an empty). This object can be:

- Rotated to control the axis (its local Z-axis is now used as the deformation axis).

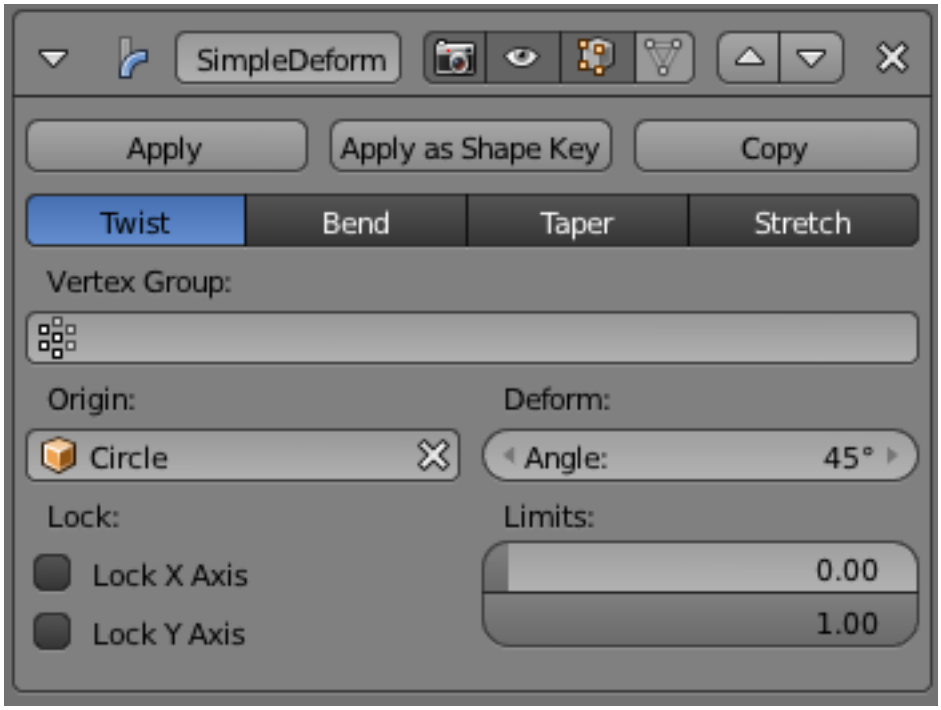


Fig. 2.942: Simple Deform.

- Translated to control the origin of deformation.
- Scaled to change the deform factor.

Note: When the object controlling the origin (the one in the *Origin* field) is a child of the deformed object, this creates a cyclic dependency in Blender's data system. The workaround is to create a new empty and parent both objects to it.

Angle/Factor The amount of deformation. Can be negative to reverse the deformation.

Limits These settings allow you to set the lower and upper limits of the deformation. The upper limit cannot be lower than lower limit.

Lock X Axis / Lock Y Axis (Taper and Stretch modes only) These controls whether the X and/or Y coordinates are allowed to change or not. Thus it is possible to squash the X coordinates of an object and keep the Y coordinates intact.

Smooth Modifier

The *Smooth Modifier* smooths a mesh by flattening the angles between adjacent faces in it, just like *Specials* → *Smooth* in

Edit Mode. It smooths without subdividing the mesh - the number of vertices remains the same.

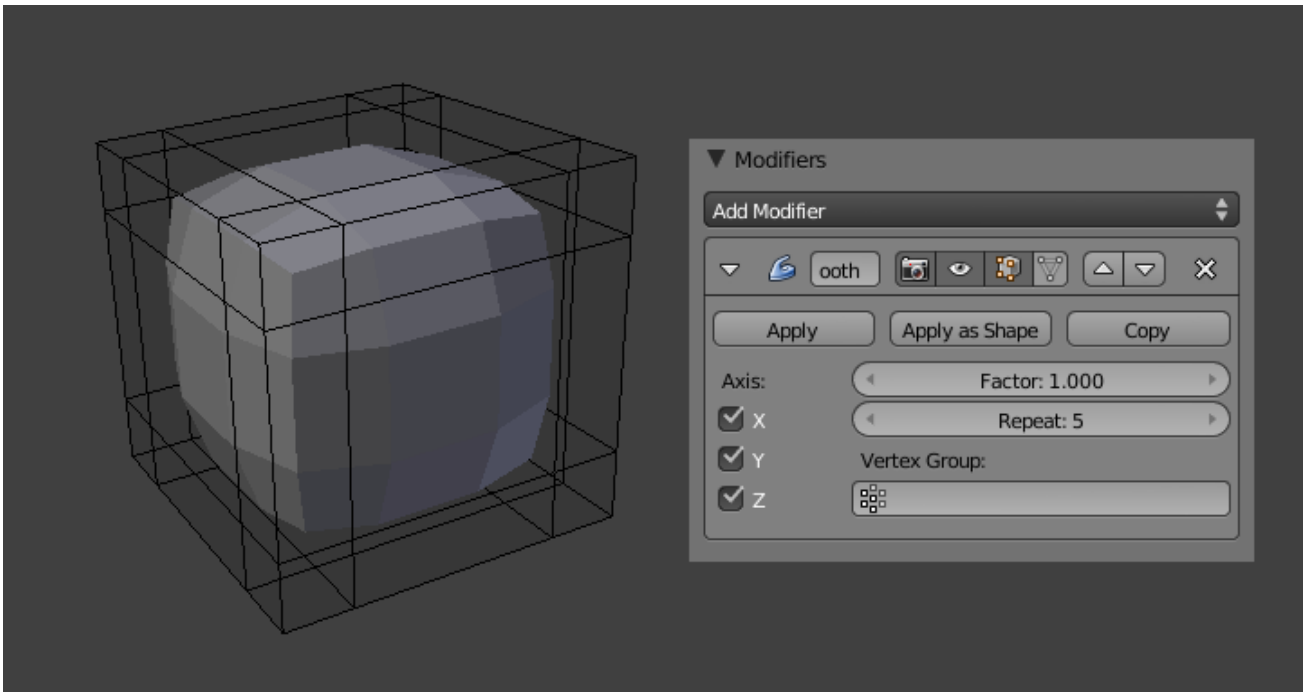


Fig. 2.943: Smooth modifier applied to a subdivided cube.

This modifier is not limited to smoothing, though. Its control factor can be configured outside the (0.0 to 1.0) range (including negative values), which can result in interesting deformations.

Options

X, Y, Z Toggle buttons to enable/disable the modifier in the X, Y and/or Z axes directions.

Factor The factor to control the smoothing amount. Higher values will increase the effect. Values outside this range (above 1.0 or below 0.0) distort the mesh.

Repeat The number of smoothing iterations, equivalent to executing the smooth tool multiple times.

Vertex Group A vertex group name, to restrict the effect to the vertices in it only. This allows for selective, real-time smoothing, by painting vertex weights.

Algorithm

The calculation done by the Smooth Modifier is a simple and logical one, and can be thought of as the geometric equivalent of blurring images.

Each new vertex position is simply the average position of surrounding vertices (the vertices connected to the same edge as it).

Warp Modifier

This deformation modifier can be used to warp parts of a mesh to a new location in a very flexible way by using two objects to select the "from" and "to" regions, with options for using a curve falloff, texture and vertex group.

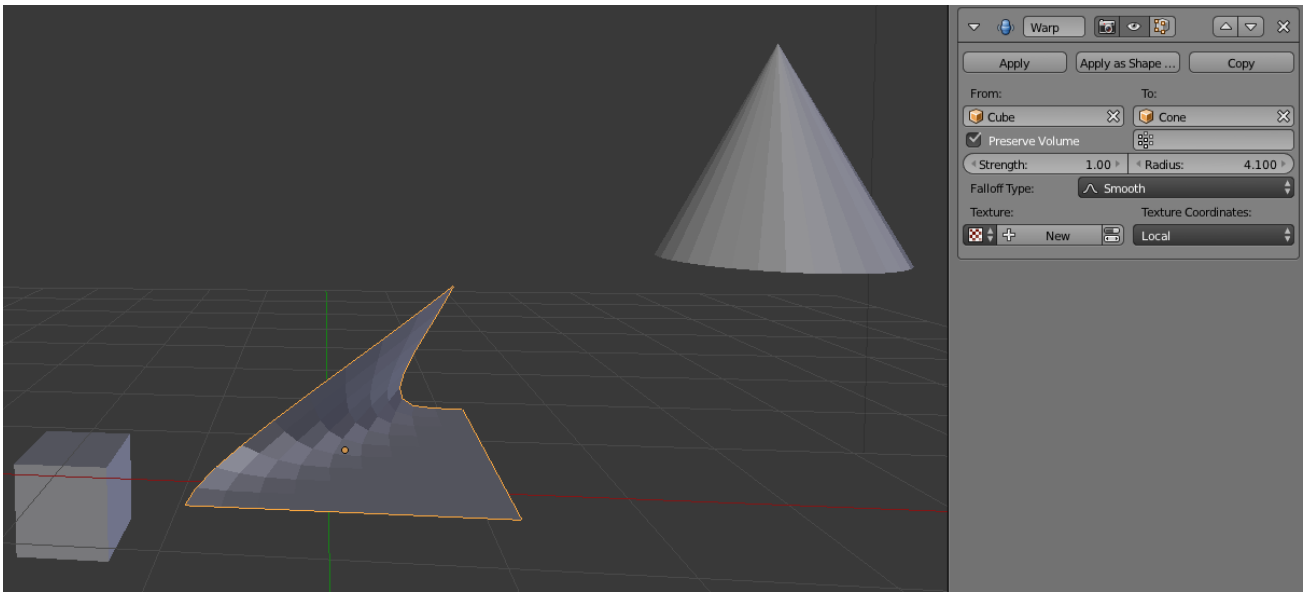


Fig. 2.944: Warp modifier applied to a grid.

The Warp Modifier is a bit tricky at first, but it helps to understand how it works. The modifier requires two points, specified by object centers. The "from" point designates a point in space that is pulled toward the "to" point. It is akin to using the *Proportional Editing* in Edit Mode.

Options

From Specify the origin object transformation of the warp.

To Specify the destination object transformation of the warp.

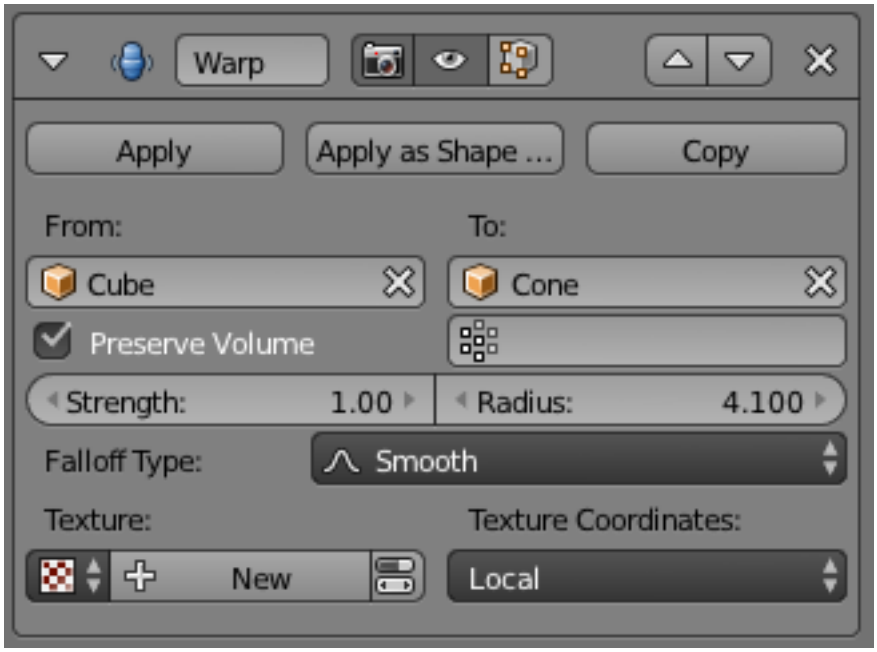


Fig. 2.945: Warp modifier.

Preserve Volume Enables volume preservation when rotating one of the transforms.

Vertex Group Limit the deformation to a specific vertex group.

Strength Sets how strong the effect is.

Radius Sets the distance from the transforms that can be warped by the transform handles.

Falloff Type Sets the way the strength of the warp change as it goes from the center of the transform to the Radius value. See *Proportional Editing* for descriptions of the falloff types.

Texture Specify a texture the strength is offset by to create variations in the displacement.

Texture Coordinates Set the way textures are applied to the mesh when using a textured warp.

Object Specify an object to use when set to Object.

UV Layer Specify a UV layer when set to UV.

Usage

The *Warp Modifier* can be awkward to use sometimes and the use case is rather small however, there are a couple of uses. For example, The *Warp Modifier* can be used

to have an interactive *Proportional Editing* that can be used for animation.

Another way to use the *Warp Modifier* is to use it similar to the *Deform Modifier*. This allows you to deform parts of the mesh without having to make a vertex group.

Wave Modifier

The Wave modifier adds a ripple-like motion to an object's geometry.

This modifier is available for meshes, lattices, curves, surfaces and texts, with one restriction for non-mesh objects: Activating *Normals* or typing a name in *VGroup* will simply deactivate the modifier.

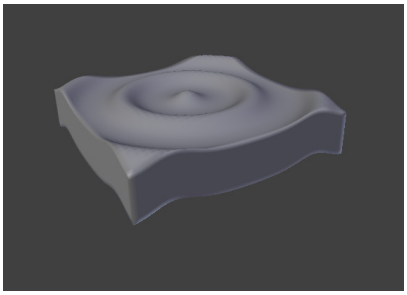


Fig. 2.946: Circular wave front.

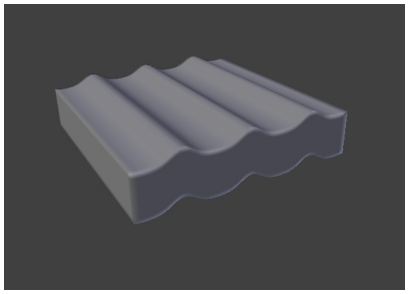


Fig. 2.947: Linear wave front.

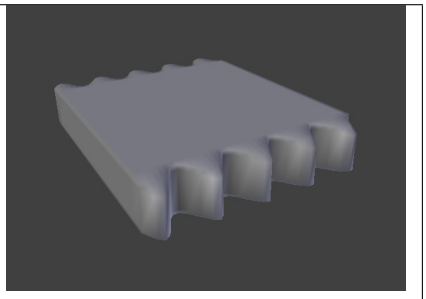


Fig. 2.948: Motion enabled for X, Normals enabled for Y.

Options

Motion

X, Y The wave effect deforms vertices/control points in the Z direction, originating from the given starting point and propagating along the object with circular wave fronts (if both X and Y are enabled), or with rectilinear wave fronts (if only one axis is enabled), then parallel to the axis corresponding to the X or Y button activated.

Cyclic Repeats the waves cyclically, rather than a single pulse.

Normals For meshes only. Displaces the mesh along the surface normals (instead of the object's Z-axis).

Time Settings to control the animation.

Offset Time offset in frames. The frame at which the wave begins (if *Speed* is positive), or ends (if *Speed* is negative). Use a negative frame number to prime and prestart the waves.

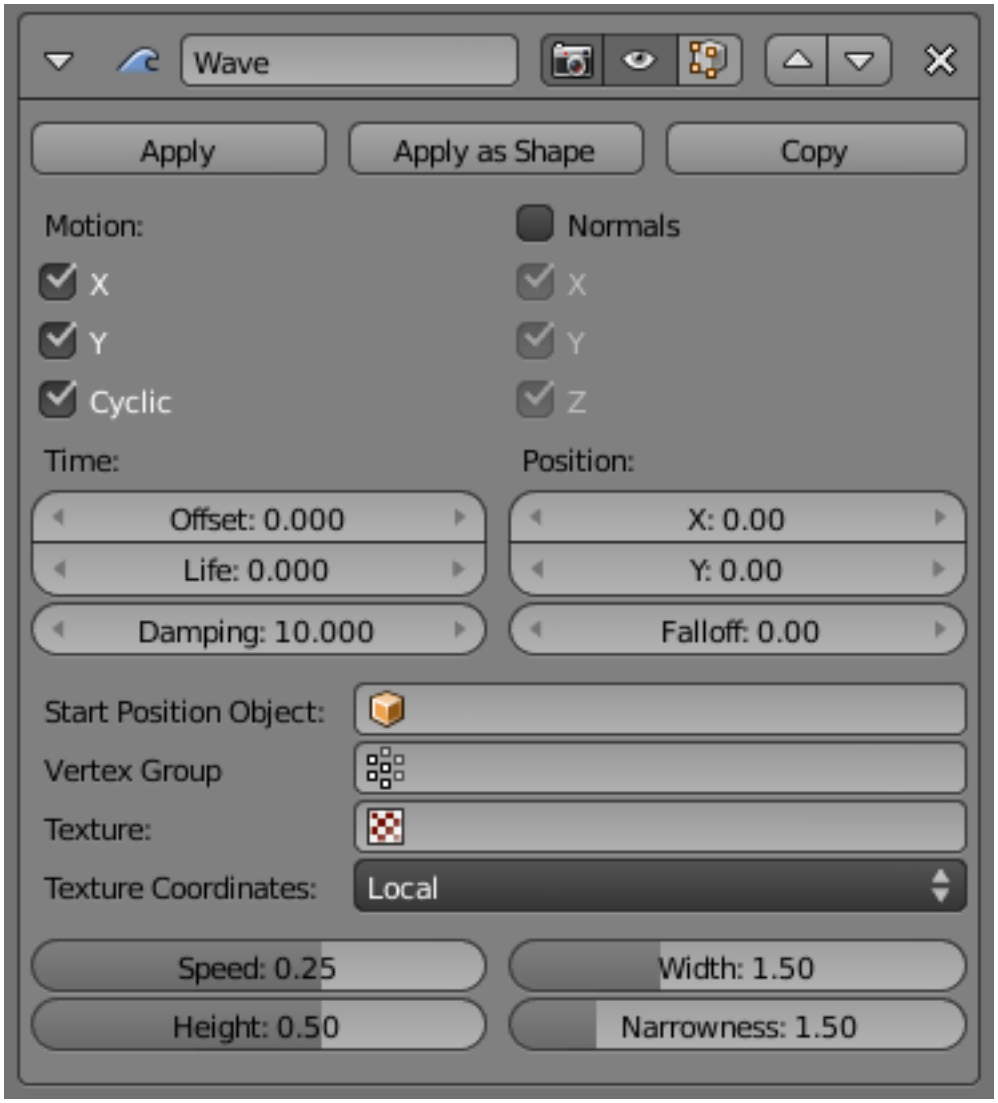


Fig. 2.949: Wave modifier.

Life Duration of animation in frames. When set to zero, loops the animation forever.

Damping An additional number of frames in which the wave slowly damps from the *Height* value to zero after *Life* is reached. The dampening occurs for all the ripples and begins in the first frame after the *Life* is over. Ripples disappear over *Damping* frames.

Position

X, Y Coordinates of the center of the waves, in the object's local coordinates.

Falloff Controls how fast the waves fade out as they travel away from the coordinates above (or those of the *Start Position Object*).

Start Position Object Use another object as the reference for the starting position of the wave. Note that you then can animate this object's position, to change the wave's origin across time.

Vertex Group For meshes only. A vertex group name, used to control the parts of the mesh affected by the wave effect, and to what extent (using vertex weights).

Texture Use this texture to control the object's displacement level. Animated textures can give very interesting results here.

Texture Coordinates This menu lets you choose the texture's coordinates for displacement:

Local Object's local coordinates.

Global Global coordinates.

Object Adds an additional field just below, to type in the name of the object from which to get the texture coordinates.

UV Adds an extra *UV Layer* property, to select the UV layer to be used.

Speed The speed, in BU (for "Blender Units") per frame, of the ripple.

Height The height or amplitude, in BU, of the ripple.

Width Half of the width, in BU, between the tops of two subsequent ripples (if *Cyclic* is enabled). This has an indirect effect on the ripple amplitude. If the pulses are too near to each other, the wave may not reach the zero Z-position, so in this case Blender actually lowers the whole wave so that the minimum is zero and, consequently, the

maximum is lower than the expected amplitude. See *Technical Details and Hints* below.

Narrowness The actual width of each pulse: the higher the value the narrower the pulse. The actual width of the area in which the single pulse is apparent is given by 4/ *Narrowness*. That is, if *Narrowness* is 1 the *pulse* is 4 units wide, and if *Narrowness* is 4 the *pulse* is 1 unit wide.

Warning: All the values described above must be multiplied with the corresponding *Scale* values of the object to get the real dimensions.

Technical Details and Hints

The relationship of the above values is described here:

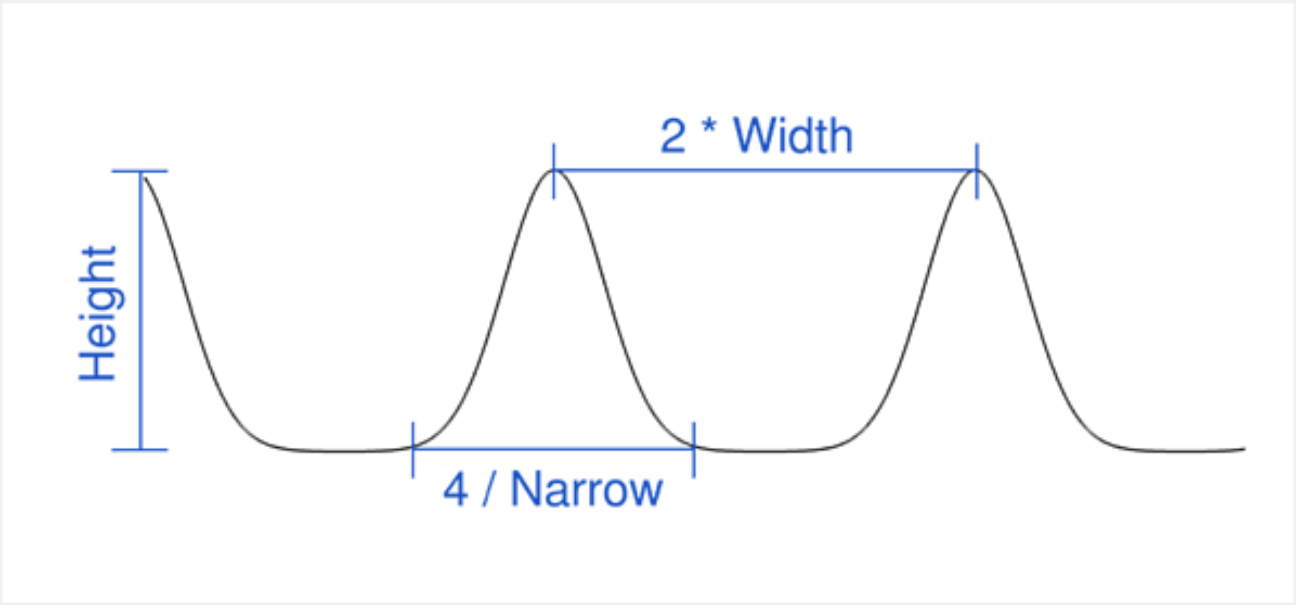


Fig. 2.950: Wave front characteristics.

To obtain a nice wave effect similar to sea waves and close to a sinusoidal wave, make the distance between following ripples and the ripple width equal; that is, the *Narrowness* value must be equal to 2/ *Width*. E.g. for *Width* to be 1, set *Narrow* to 2.

Simulate

Explode Modifier

The Explode Modifier is used to alter the mesh geometry by moving/rotating its

faces in a way that roughly tracks particles emitted by that object, making it look as if the mesh is being exploded (broken apart and pushed outward).

For the Explode Modifier to have a visible effect, there needs to be particle system on it. The particle system on the mesh is what controls how the mesh will be exploded, and therefore without the particle system the mesh will not appear to alter.

Both the number of emitted particles and number of faces determine how granular the Explode Modifier will be. More faces and more particles will mean more individual pieces.

Here is a demo video showing a cube with a particle system and Explode Modifier. (Blend file)

Note: The Explode modifier must come after the Particle System Modifier because the Particle System Modifier has the information needed to drive the Explode Modifier.

Options

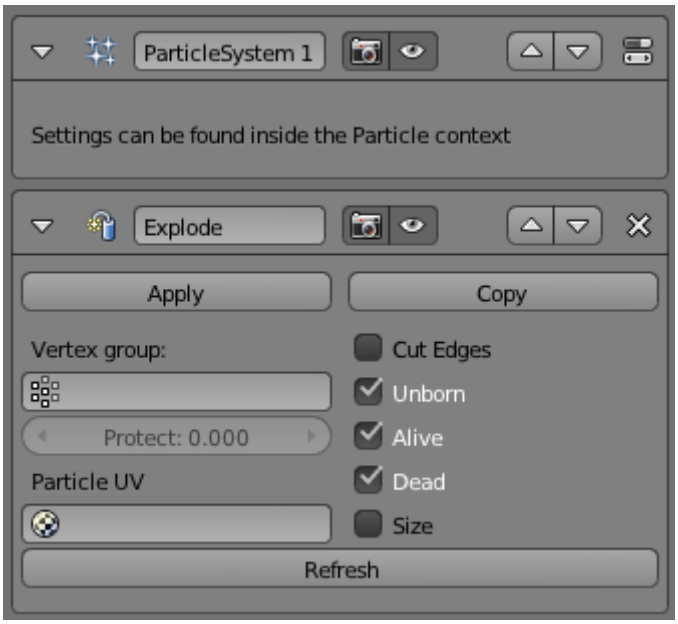


Fig. 2.951: Explode Modifier panel with Particle System Modifier above it.

Vertex group Vertices in this group may not be affected by the Explode Modifier. Vertices with full weight are not affected at all, while vertices with less weight have a higher chance of being affected.

Vertices with no weight will be treated like those which do not belong to the group at

all and explode normally.

Protect Clean vertex group edges. Depending on the weights assigned to that vertex group; either completely protect those faces from being affected by the Explode Modifier (which would happen if the faces had a weight value of 1) or completely remove protection from those faces (which would happen if the faces had a weight value 0).

Particle UV UV map to change with particle age.

Cut Edges Cut face edges for nicer shrapnel.

Unborn Show mesh when particles are unborn.

Alive Show mesh when particles are alive.

Dead Show mesh when particles are dead.

Size Use particle size for shrapnel.

Refresh Refresh data in the explode modifier.

Ocean Modifier

The *Ocean Modifier* is an ocean simulation tool to simulate and generate a deforming ocean surface, and associated texture, used to render the simulation data.

Note: The *Ocean Modifier* is ported from the open source Houdini Ocean Toolkit and is intended to simulate deep ocean waves and foam.

Options

Geometry

Geometry

Generate Creates a tiled mesh grid that exactly corresponds with the resolution of the simulation data.

When generating a mesh surface, the existing mesh object is completely overridden with the ocean grid. A UV channel is also added, mapping the (0.0 to 1.0) UV space to the simulation grid.

Displace Uses the existing geometry rather than replacing it. Vertices are displaced along the local Z-axis.

Repeat X, Repeat Y When generating a mesh surface, controls the number of times the grid is tiled in X and Y directions. UVs for

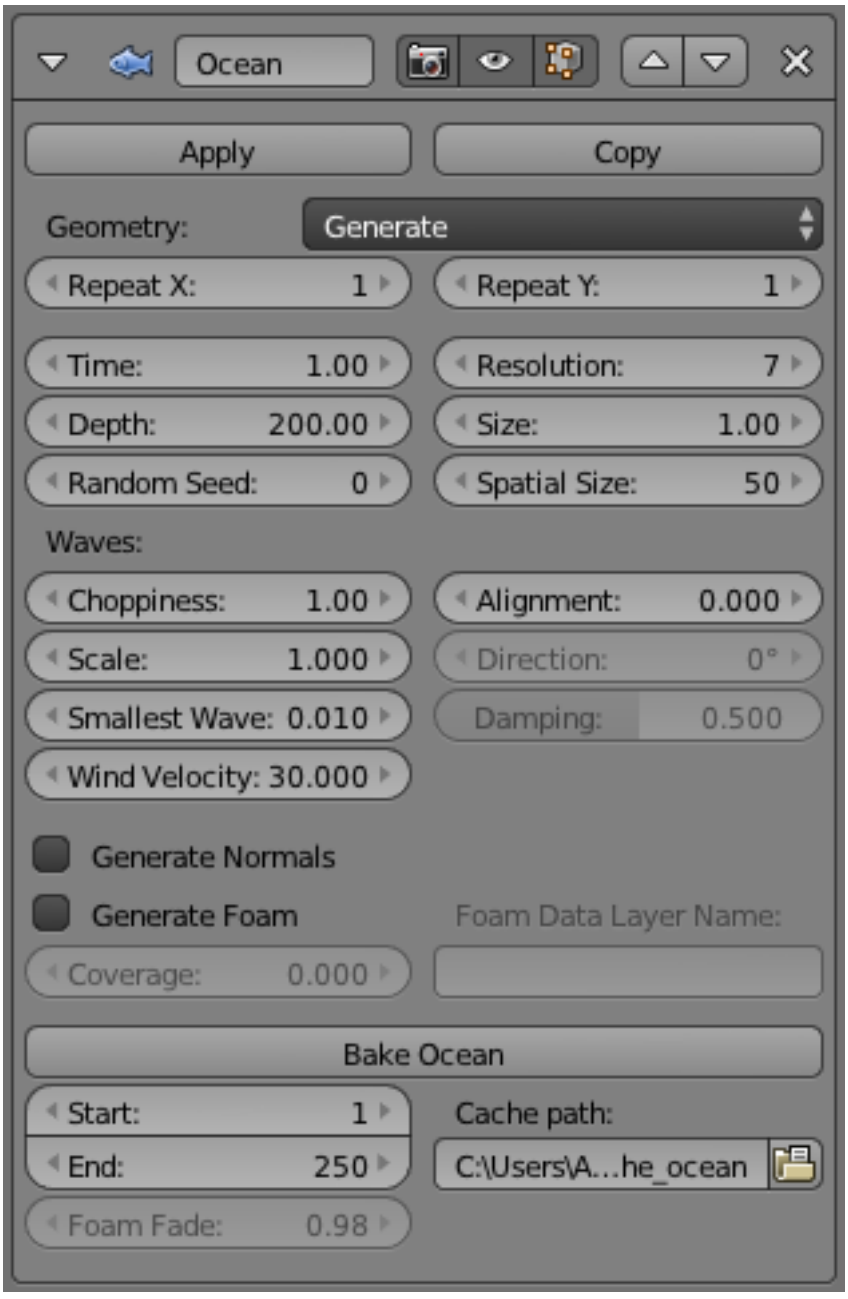


Fig. 2.952: Ocean Modifier.

these tiled mesh areas continue outside of the (0.0 to 1.0) UV space.

Time The time at which the ocean surface is being evaluated. To make an animated ocean, you will need to insert keyframes RMB and animate this time value. The speed that the time value is changing will determine the speed of the wave animation.

Depth The constant depth of the ocean floor under the simulated area. Lower values simulate shallower waters by producing higher frequency details and smaller waves.

Random Seed A different seed will produce a different simulation result.

Resolution The main control of quality vs speed in the simulation engine. This determines the resolution of the internal 2D grids generated by the simulation.

The internal grids are powers of two of the resolution value, so a resolution value of 16, will create simulation data of size 256×256. The higher the resolution, the more detail will be produced, but the slower it will be to calculate.

Note: When using the 'Generate' modifier geometry option, this resolution value also determines the resolution of the generated mesh surface, equal to the resolution of the internal simulation data.

Size A simple scaling factor that does not affect the height of the waves or behavior of the simulation.

Spatial Size The width of the ocean surface area being simulated, in meters. This also determines the size of the generated mesh, or the displaced area, in Blender units. Of course you can scale the object with ocean modifier in object mode to tweak the apparent size in your scene.

Wave

Choppiness The choppiness of the wave peaks. With a choppiness of 0, the ocean surface is only displaced up and down in the Z direction, but with higher choppiness, the waves are also displaced laterally in X and Y, to create sharper wave peaks.

Scale An overall scale control for the amplitude of the waves. It approximates the height or depth of the waves above or below zero.

Rather than just scaling the ocean object in Z, it scales all aspects of the simulation, displacement in X and Y, and corresponding foam and normals too.

Alignment The directionality of the wave shapes due to wind. At a value of 0, the wind and waves are randomly, uniformly oriented. With higher Alignment values, the wind is blowing in a more constant direction, making the waves appear more compressed and aligned to a single direction.

Direction When using Alignment, the direction in degrees that the waves are aligned to.

Damping When using Alignment, amount that inter-reflected waves are damped out. This has the effect of making the wave motion more directional (not just the wave shape). With damping of 0.0, waves are reflected off each other every direction, with damping of 1.0, these inter-reflected waves are damped out, leaving only waves traveling in the direction of the wind.

Smallest Wave A minimum limit for the size of generated waves. Acts similarly to a low-pass filter, removing higher frequency wave detail.

Wind Velocity Wind speed in meters/second. With a low velocity, waves are restricted to smaller surface waves.

Simulation Data Generation Options

By default, the simulator only generates displacement data, since it takes the least amount of work and gives the fastest feedback. Additional simulation data can be generated for rendering as well.

Generate Normals Simulates additional normal map data. This can be used by the Ocean texture, when mapped to Normals, as a bump map, and enables generating normal map image sequences when baking.

Generate Foam Simulates additional foam data. This can be retrieved by the Ocean texture for use in texturing (perhaps as a mask), and enables generating foam map image sequences when baking.

Coverage Tweaks the amount of foam covering the waves, negative values will reduce the amount of foam (leaving only the topmost peaks), positive values will add it. Typically ranges from (-1.0 to 1.0)

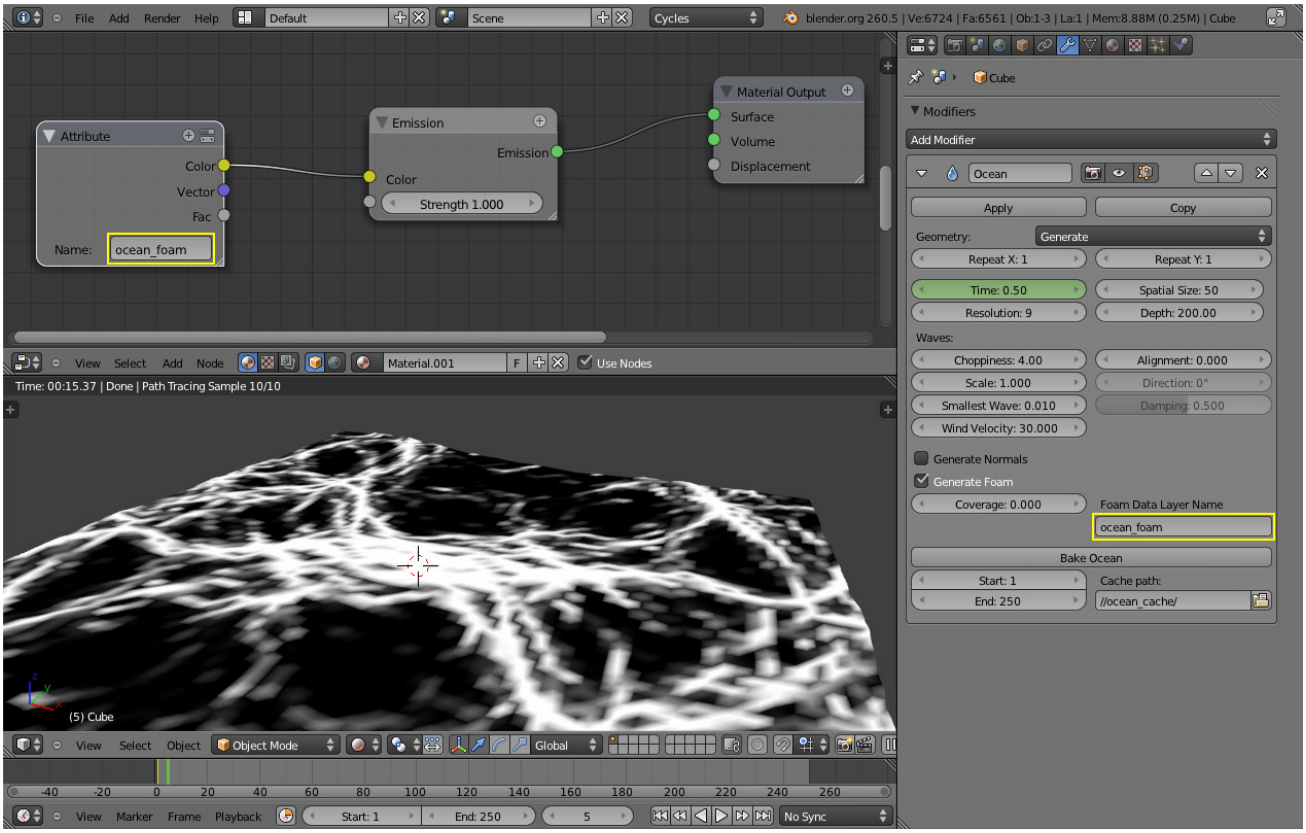


Fig. 2.953: Using foam vertex colors with a named data layer.

Foam Data Layer Name Optional name for the vertex data layer, used by the Ocean modifier to store foam maps as vertex colors. This is required for accessing the foam data in the renderer.

Baking

Rather than simulating the ocean data live, the ocean data can be baked to disk. When a simulation is baked, the simulator engine is completely bypassed, and the modifier/texture retrieves all information from the baked files.

Baking can be advantageous for a few reasons:

- It is faster to use the stored data rather than re-calculating it.
- Allows rendering ocean data in external renderers.
- Enables more advanced foam maps.

Data Files

Simulation data is stored in disk as sequences of OpenEXR image maps, one for each of displacement, normal and foam (if enabled to be generated). Upon loading the data from these baked files, when a frame of the bake sequence is read from disk, it is cached in memory. This means that accessing loaded frames subsequent times is fast, not incurring the overhead of disk access.

Since these baked files are plain OpenEXRs, they can also be opened and rendered in any other application or renderer that supports them.

Baking Foam

Baking also provides improved foam capabilities. When simulating live, the ocean simulator retrieves data for that current frame only. In the case of the foam map, this represents the tips of wave crests for that given frame. In reality, after foam is created by wave interactions, it remains sitting on the top of the wave surface for a while, as it dissipates. With baking, it is possible to approximate that behavior, by accumulating foam from previous frames, leaving it remaining on the surface.

Baking Options

Start, End Frames of the simulation to bake (inclusive). The start and end frames of the bake are repeated when accessing frames outside the baked range.

Cache Path Folder to store the baked EXR files in. The sequences will be in the form `disp_####.exr`, `normal_####.exr`, and `foam_####.exr` where `####` is the four digit frame number. If the cache path folder does not exist, it will be created.

Simulation Internals

The simulator itself uses FFT methods to generate 2D grids of simulation information internally, very similar to 2D texture maps. The simulator can generate three types of data: displacement, normals, and extra data, that is used to calculate wave

crest intersections (i.e. foam). After simulation, these maps are used to displace the ocean surface geometry in 3D, and also can be used for shading via the Ocean texture. The internal simulation engine is multi threaded with OpenMP to take advantage of multiple cores.

Examples

Simulated and baked to image maps in Blender, rendered in 3Delight.

Particle Instance Modifier

When a *Particle Instance* modifier is added to an object, that object will be used as a particle shape on an object which has a particle system associated with it. This means that to use this modifier you must also have another object which has a particle system on it, otherwise the *Particle Instance* modifier will appear to do nothing.

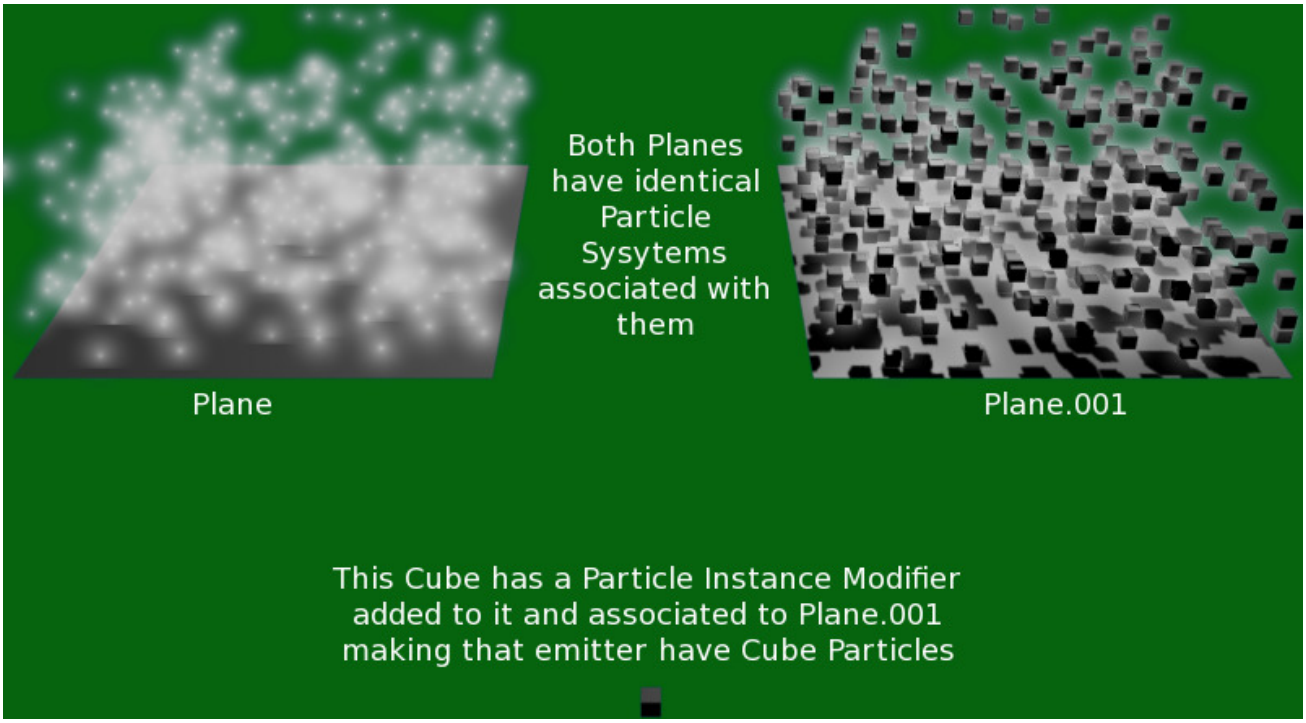


Fig. 2.954: Particle system on left has no Particle Instance modified object associated with it. The one on the right is associated with cube shown by using a Particle Instance modifier on the cube.

Options

Because of the co-dependent way in which the *Particle Instance* modifier is influenced

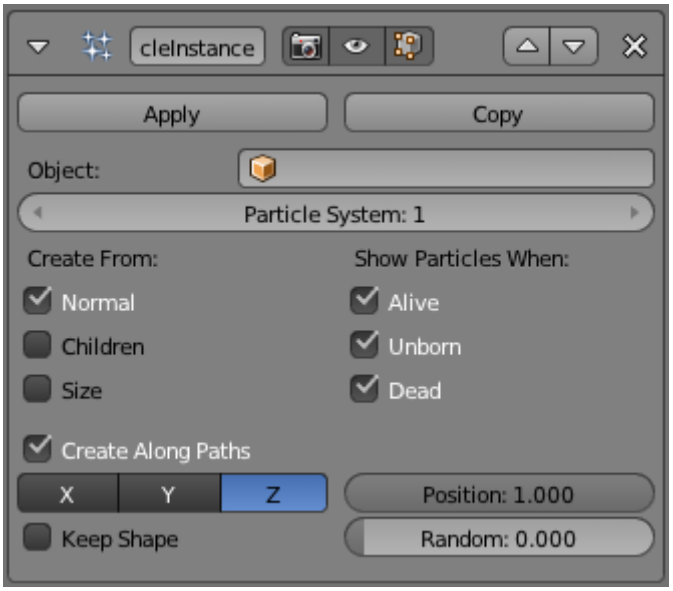


Fig. 2.955: Particle Instance Modifier.

by the underlying particle systems on other objects, some of the apparent effects generated by the *Particle Instance* modifier can look and act vastly different, depending on the underlying settings of the particle systems it is associated with. This is worth taking account of if the *Particle Instance* modifier settings do not appear to be giving the results expected, as it may indicate that the particle system settings may need altering rather than the *Particle Instance* modifier settings.

Object The *Object* field, associates this *Particle Instance* modifier with another object (usually an object having a particle system...). This indicates that when the object named in this field emits particles, those particles will have the mesh shape of the current *Particle Instance* modifier's mesh. If for example a sphere has a *Particle Instance* modifier added to it, when the *Object* field of this modifier is filled in with the name of an object that emits particles, those particle will be sphere shaped. Even though most of the time the *Object* field will have the name of an object with a particle system, this is not mandatory, you can enter an object's name which does not have a particle system, and it will be accepted by the *Object* field, as there do not appear to be any checks made to make sure the object's name entered into this field is "valid".

Particle System The *Particle System* field is used to select which particle system number to apply the *Particle Instance* modifier to, when the mesh which has the particle

system on it has more than one of these. The *Particle System* field can have a value between (1 to 10). It is possible to select any of the ten particle system numbers, however, a check will **not** be made with the underlying particle emitting object specified previously in the *Object* field. If you select a particle system number which does not exist on the particle emitting object, then the particles on the emitting mesh will keep their normal particle shapes. No warning will be given that the chosen particle system does not exist on a particular particle emitting mesh.

As an example, below is a single plane mesh with two areas (the first area shown in red and the second in white), with different particle systems applied to each area. The left side using a *Particle Instance* modifier which has the shape of a sphere and the right side having a *Particle Instance* modifier which has the shape of a cube.

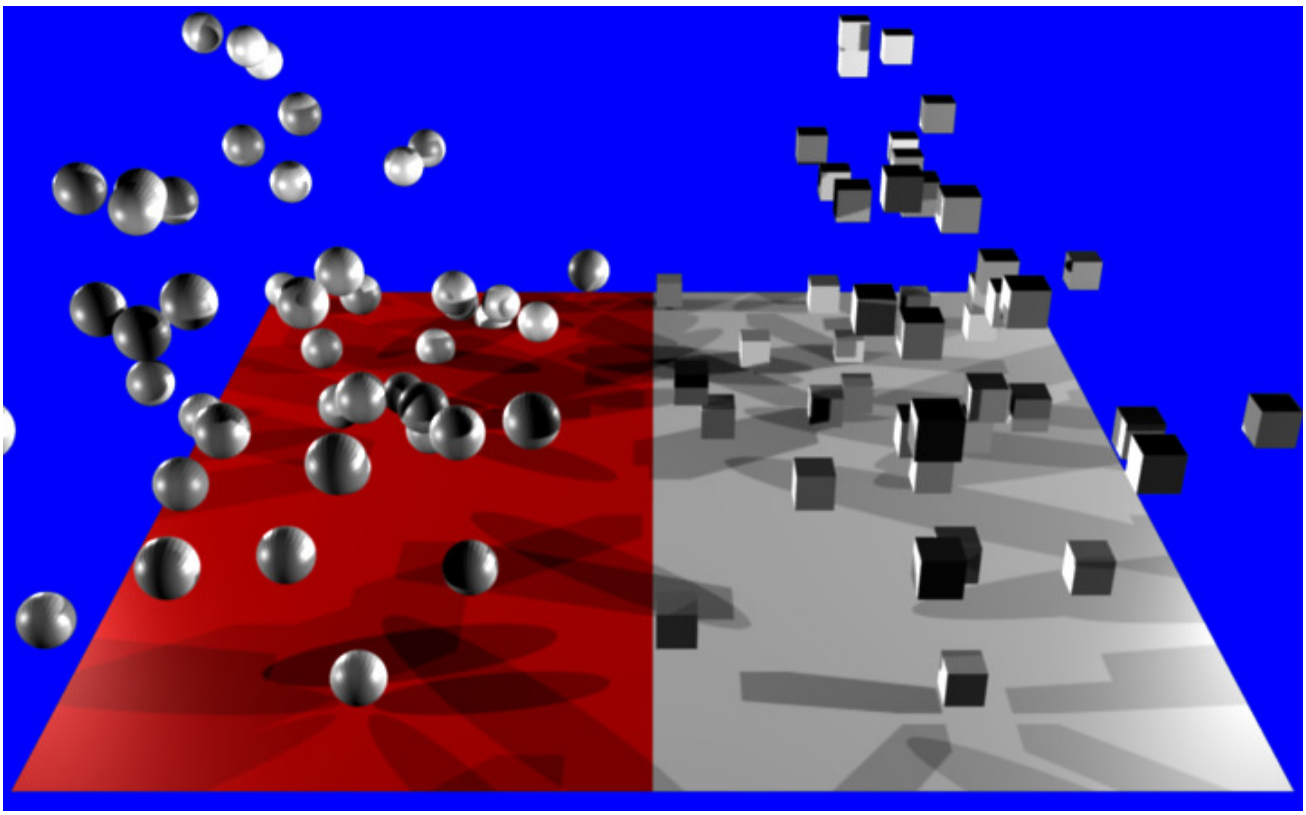


Fig. 2.956: Render showing a single Plain mesh object assigned to two different vertex groups and each of those vertex groups is assigned a separate and independent particle system, with each particle system being assigned a different Particle Instance modifier. In the case shown the Particle Instance modifiers are a sphere and a cube. Example Blend file.

Creation

Normal When selected, the *Normal* button tells the *Particle Instance* modifier to draw instances of itself wherever normal particle types are emitted from the underlying particle system. So if the current *Particle Instance* modifier is a sphere shape, when normal particles are emitted they will be spheres.

Children When selected, the *Children* button tells the *Particle Instance* modifier to draw instances of itself wherever children/child particles are emitted/used on the underlying particle system. So if the current *Particle Instance* modifier is a sphere shape, when children/child particles are emitted they will be spheres.

Size Scale the instanced objects by the particle size attribute. When this is disabled, all the copies appear the same size as the origin.

Display

Unborn When selected, the *Unborn* button tells the *Particle Instance* modifier to draw instances of itself wherever unborn particles will be emitted/used on the underlying particle system. So if the current *Particle Instance* modifier is a sphere shape, when unborn particles are present they will be spheres.

Alive When selected, the *Alive* button tells the *Particle Instance* modifier to draw instances of itself wherever alive particles will be emitted/used on the underlying particle system. So if the current *Particle Instance* modifier is a sphere shape, when alive particles are present they will be spheres.

Dead When selected, the *Dead* button tells the *Particle Instance* modifier to draw instances of itself wherever dead particles will occur on the underlying particle system. So if the current *Particle Instance* modifier is a sphere shape, when dead particles are present they will be spheres.

Using Paths

Create Along Paths This option tries to make the underlying mesh object of the *Particle Instance* modifier deform its mesh shape in such a way as to try and match the path

traveled by the particles/hair strands of the system associated with it. For example, below is a screen shot showing the path of a single keyed particle as it travels its way through each of the different way points (1 to 4) (target particle systems), when it reaches way point 4 the particle dies and ends its journey.

Rotation Axis Specify which pole axis to use for the rotation.

X, Y, Z

Keep Shape Enabling this prevents the object from being deformed. It instead simply aligns to the end of the path at the object's center.

Position Specify what percentage of the path the object fills. You could create a growing effect by animating this value over time.

Random Scales the position value of each instance a random value.

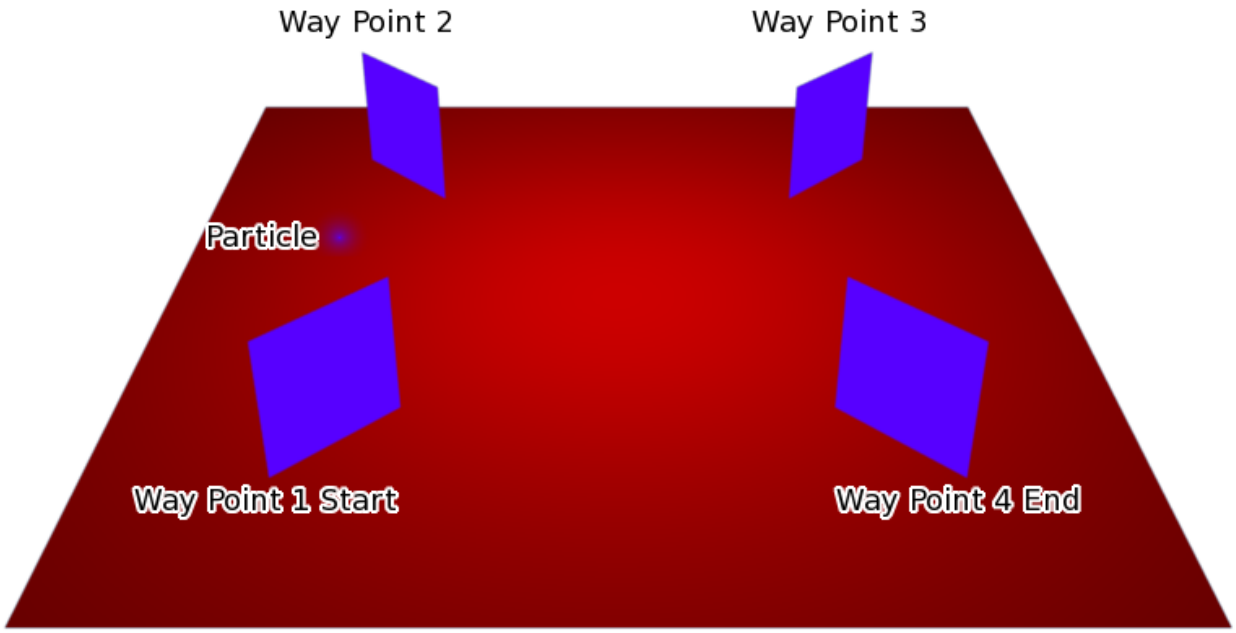


Fig. 2.957: Keyed particle following way points (showing one particle). Example Blend file.

When a *Particle Instance* modifier is added to a cylinder object and then associated with the way point particle system, the particle position is copied by the cylinder and placed at the particles position. So the mesh object follows the location of the particle. The cylinder does not alter any of its other properties when following the particle, only the cylinders location gets altered, shape and rotation do not get altered. See screenshot below:

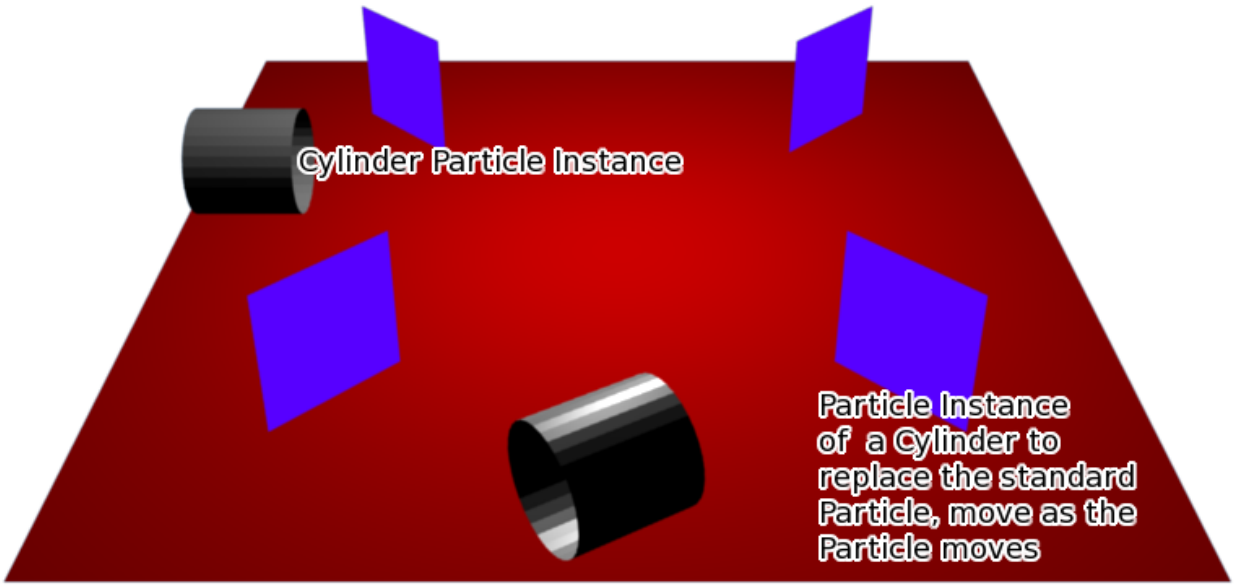


Fig. 2.958: Keyed particle following way points showing a mesh object (Particle Instance modifier) in place of the original particle. Example Blend file.

Both of the above examples had the *Particle Instance* modifier *Path* button deactivated. When the *Path* button is activated the effect can be seen in the screenshot below:

Instead of the cylinder location just following the position of the particle (and not altering its shape), the cylinder tries to fit its mesh to the shape of the path followed by the particle. The mesh geometry of the object which is trying to deform can have an

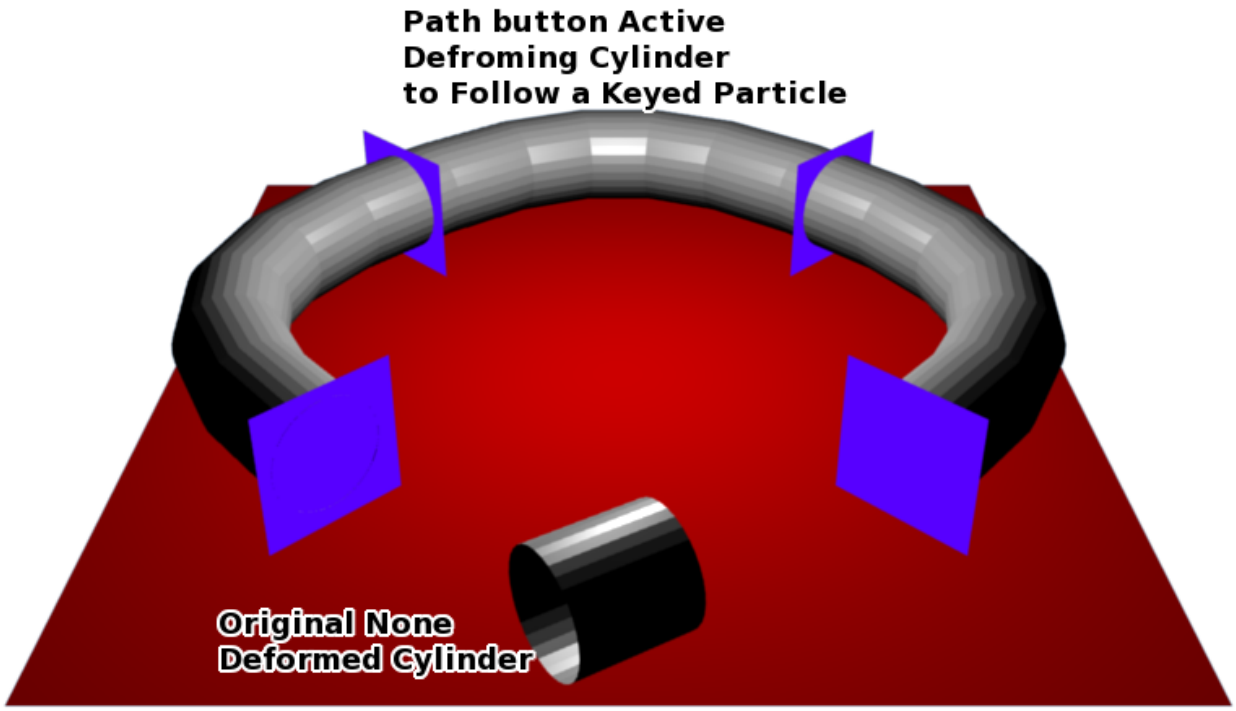


Fig. 2.959: Keyed particle following way points showing a mesh object (Particle Instance modifier) in place of the original particle, that is also being deformed to fit the travel path of the original particle. Example Blend file.

impact on how well the deformation is carried out. In the case of the cylinder, it has many loop cuts along its length so that it can bend at those points to deform along the particle path. For example here is the same scene with the number of loop cuts along the length of the cylinder reduced, showing the effect on the deformation of the cylinder along the particle path.

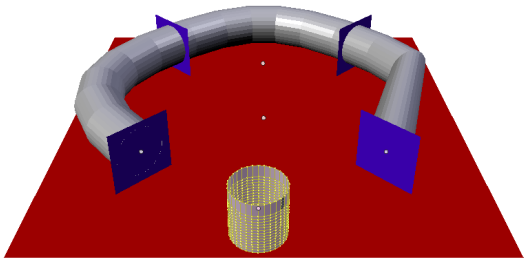


Fig. 2.960: The cylinder has most of its edge loops so most of the path deform is very regular apart from at the very end of the curve.

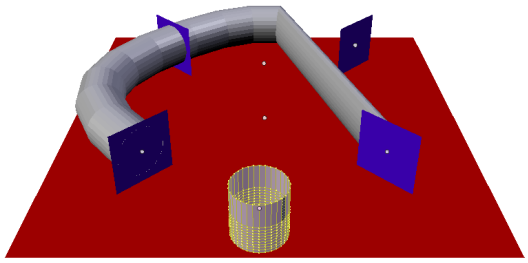


Fig. 2.961: The cylinder has some of its edge loops removed so the path of the deform starts to become less regular.

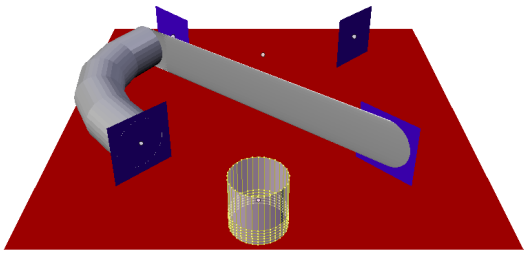


Fig. 2.962: Now the deform path is very rough.

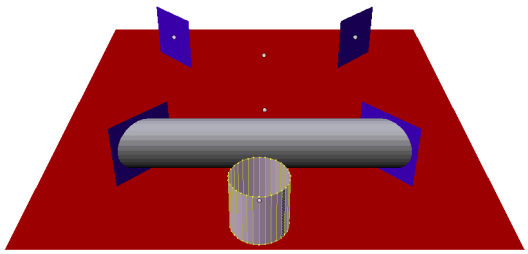


Fig. 2.963: At this point there are not any vertices to bend the cylinder to follow the path, and instead the cylinder just goes directly to the last way point 4.

Once all the extra edge loops around cylinder are removed so that there is only the top and bottom vertices left, meaning that the cylinder does not have enough geometry to bend, in that case it cannot follow the path of the particle, so it just goes from the start way point 1 to the ending way point 4. The *Particle Instance* modifier *Path* button works for hair (strand) particles as well as with keyed particles. In this case the mesh of the *Particle Instance* modifier will follow the length and profile of the hair strands paths. Below is a screenshot show-

ing the effect of the *Path* button on hair:

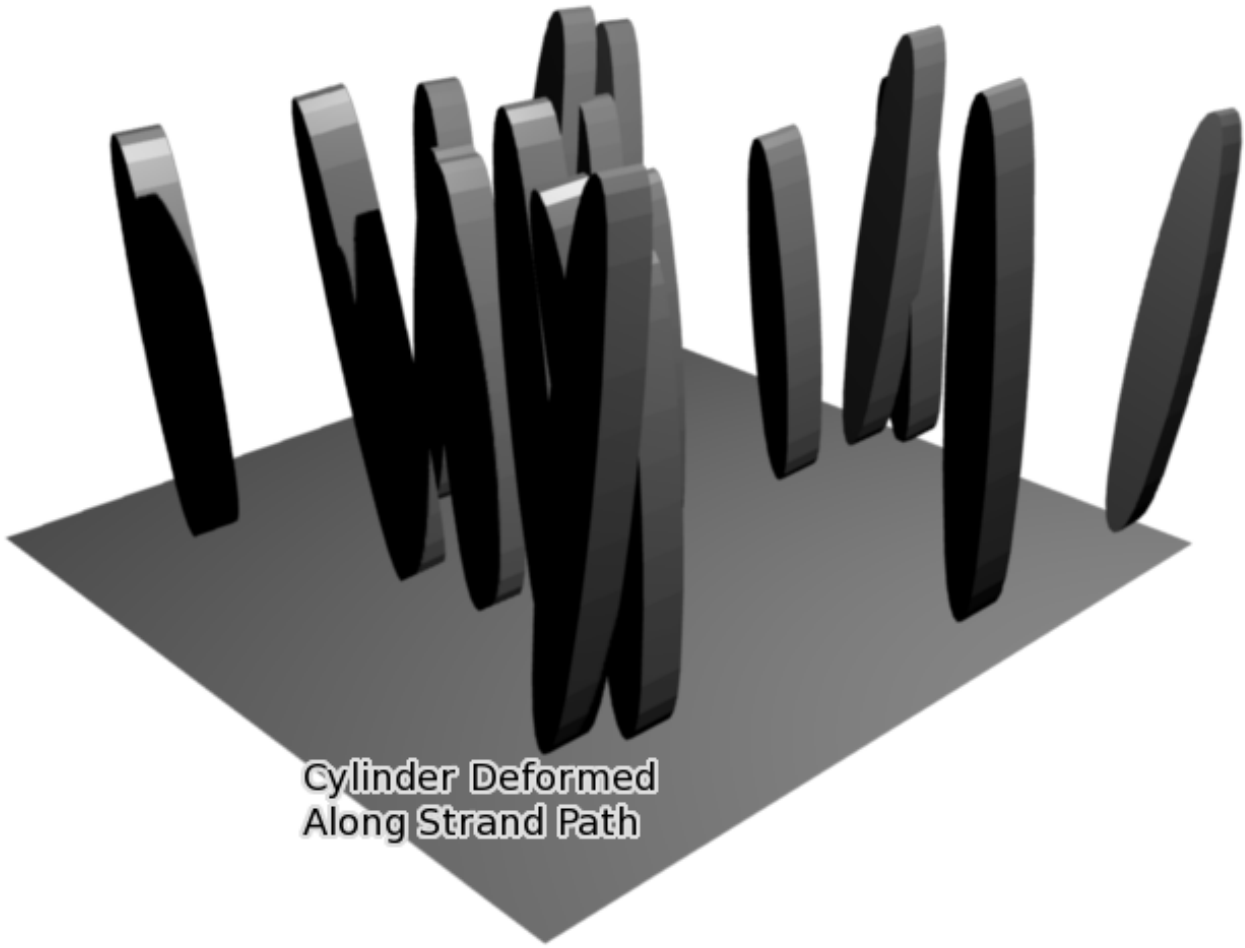


Fig. 2.964: Strand with a Particle Instance modifier associated with it and deforming the cylinder along the hair profile. Example Blend file.

Note: Strands when they are generated instantly die when created so for the *Path* button to be of any use, you must also have the *Dead* button activated. Otherwise the path a mesh took will not be visible!

See also:

Particles

2.5 Painting & Sculpting

2.5.1 Introduction

Brush

TODO.

2.5.2 Painting

Introduction

Todo.

Paint Modes

Texture Paint

Introduction

A UV Texture is a picture (image, sequence or movie) that is used to color the surface of a mesh. The UV Texture is mapped to the mesh through one or more UV maps. There are three ways to establish the image used by the UV Texture:

- Paint a flat image in the UV/Image Editor onto the currently selected UV Texture, using its UV map to transfer the colors to the faces of the mesh.
- Paint the mesh in the 3D View, and let Blender use the currently selected UV map to update the UV Texture. (see *Projection Painting*).
- Use any image-editing (paint) program to create an image. In the UV/Image Editor, select the UV Texture and load the image. Blender will then use that texture's UV map to transfer the colors to the faces of the mesh.

  Blender features a built-in paint mode called Texture Paint which is designed specifically to help you edit your UV Textures and images quickly and easily in either the UV/Image or the 3D View Editor. Since a UV Texture is just a special-purpose image, you can also use any external paint program. For example, GIMP is a full-featured image manipulation program that is also open-source.

  Since a mesh can have layers of UV Textures, there may be many images that color the mesh. However, each UV Texture only has one image.

  Texture Paint works in both a 3D View and the UV/Image Editor. In the 3D View in Texture Paint Mode, you paint directly on the mesh by *projecting onto the UVs*.

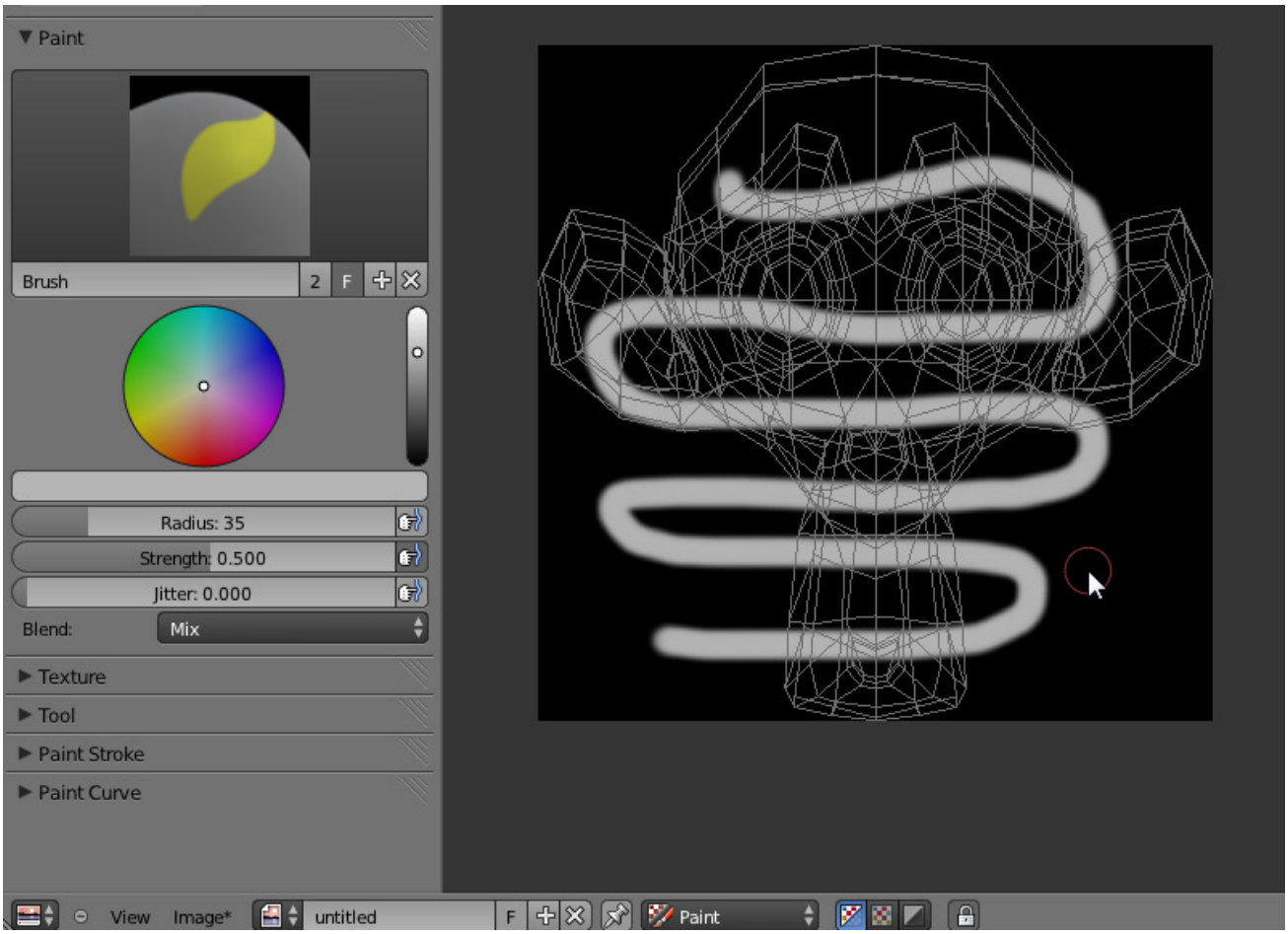


Fig. 2.965: Texture painting in Blender.

Getting Started

Once you have unwrapped your model to a UV Map (as explained in previous pages), you can begin the texturing process. You cannot paint on a mesh in Texture Paint Mode without **first** unwrapping your mesh, **and** doing one of the following steps. Either:

See: *Applying Image*.

After you have done one of these two things, you can modify the image using the Texture Paint Mode:

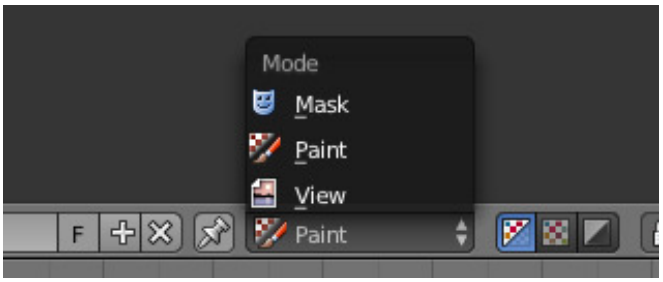


Fig. 2.966: Enabling paint mode.

- In the 3D View, select Texture Paint Mode from the mode selector in the header, and you can paint directly onto the mesh.
- In the UV/Image Editor, switch the mode from View to Paint (shown to the right).

Note: Square Power of Two

Texture paint is very fast and responsive when working in the 3D View and when your image is sized as a square where the side lengths are a power of two, e.g. 256×256, 512×512, 1024×1024, etc.

Once you enable Texture Painting, your mouse becomes a brush. To work with the UV layout (for example, to move coordinates) you must go back to "View" mode.

As soon as you enable Texture Painting or switch to Texture Paint Mode, brush settings become available in the Tool Shelf.

In the UV/Image Editor, you paint on a flat canvas that is wrapped around the mesh using UV coordinates. Any changes made in the UV/Image Editor show up immediately in the 3D View, and vice versa.

A full complement of brushes and colors can be selected from the Properties region

in the UV/Image Editor. Brush changes made in either panel are immediately reflected in the other panel. However, the modified texture will **not** be saved automatically; you must explicitly do so by *Image* → *Save* in the UV/Image Editor.

Texture Preview

If your texture is already used to color, bump map, displace, alpha-transparent, etc., a surface of a model in your scene (in other technical words, is mapped to some aspect of a texture via a texture channel using UV as a map input), you can see the effects of your painting in the context of your scene as you paint.

To do this, set up side-by-side areas, one area in 3D View set to *Texture* shading option, and in the second area the UV/Image Editor loaded with your image. Position the 3D View to show the object that is UV mapped to the loaded image. In the image to the right, the texture being painted is mapped to the "Normal" attribute, and is called "bump mapping", where the grayscale image is used to make the flat surface appear bumpy. See Texture Mapping Output for more information on bump mapping.

Saving

If the header menu item Image has an asterisk next to it, it means that the image has been changed, but not saved. Use the *Image* → *Save Image* option to save your work with a different name or overwrite the original image.

Note: UV Textures

Since images used as UV Textures are functionally different from other images, you should keep them in a directory separate from other images.

The image format for saving is independent of the format for rendering. The format for saving a UV image is selected in the header of the File browser, and defaults to `PNG` (`.png`).

If Packing is enabled in the File browsers header, or if you manually *Image* → *Pack*

Image, saving your images to a separate file is not necessary.

Using an External Image Editor

If you use an external program to edit your UV Texture, you must:

- run that paint program (GIMP, Photoshop® , etc.)
- load the image or create a new one
- change the image, and
- re-save it within that program.
- Back in Blender, you reload the image in the UV/Image Editor.

  You want to use an external program if you have teams of people using different programs that are developing the UV textures, or if you want to apply any special effects that Texture Paint does not feature, or if you are much more familiar with your favorite paint program.

Known Limitations

UV Overlap

In general overlapping UVs are not supported (as with texture baking).

However, this is only a problem when a single brush stroke paints onto multiple faces that share a texture.

Perspective View & Faces Behind the View

When painting onto a face which is partially behind the view (in perspective mode), the face cannot be painted on. To avoid, this zoom out or use an Ortho mode viewport.

Perspective View & Low Poly

When painting onto a face in perspective mode onto a low poly object with normals pointing away from the view, painting may fail; to workaround disable the *Normal* option in the paint panel.

Typically this happens when painting onto the side of a cube (see Bug report T34665).

Tools

Open the Tool Shelf in the 3D View or UV/Image Editor .

Press S on any part of the image to sample that color and set it as the brush color.

Brush

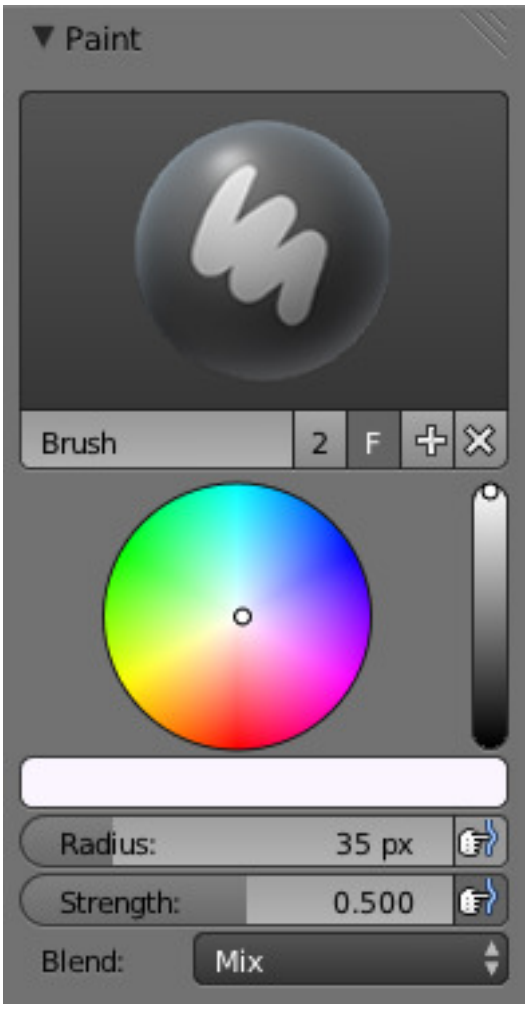


Fig. 2.967: Brush Settings.

With this panel, you can create many brushes, each with unique settings (such as color and width).

Brush *Data-Block Menu* to select a preset *Brush Types* or a custom brush.

Add + When you add a brush, the new brush is a clone of the current one.

Note: In order to save in a blend-user a custom brush set a Fake User.

Common

Most brushes have common settings.

Color The color of the brush.

Radius The radius of the brush in pixels.

Strength How powerful the brush is when applied.

Pressure Sensitivity The icon (hand and bulged in blue line) to the right of the following three settings will enable or disable tablet pressure sensitivity to control how strong the effect is.

Blend Set the way the paint is applied over the underlying color. See *Color Blend Modes*.

- Add Alpha: makes the image more opaque where painted.
- Erase Alpha: makes the image transparent where painted, allowing background colors and lower-level textures to show through. As you 'paint', the false checkerboard background will be revealed. Using a table pen's eraser end will toggle on this mode.
- Luminosity
- Exclusion
- Vivid light
- Pin light

Tip: In order to see the effects of the Erase and Add Alpha mix modes in the UV/Image Editor, you must enable the alpha channel display by clicking the Display Alpha or the Alpha-Only button. Transparent (no alpha) areas will then show a checkered background.

Alpha Opacity of the clone image display.

Brush Types

Draw

The normal brush; paints a swath of color.

Soften

Blends edges between two colors.

Smear

When you click, takes the colors under the cursor, and blends them in the direction you move the mouse. Similar to the "smudge" tool of *Gimp*.

Clone

Copies the colors from the image specified (Tex.Dirt in the example), to the active image.

Clone from paint slot The background image is shown when this brush is selected; use the *Strength* slider to control how prominent the background image is.

Source Clone Slot When using the clone brush, this allows you to select an image as a clone source.

Texture

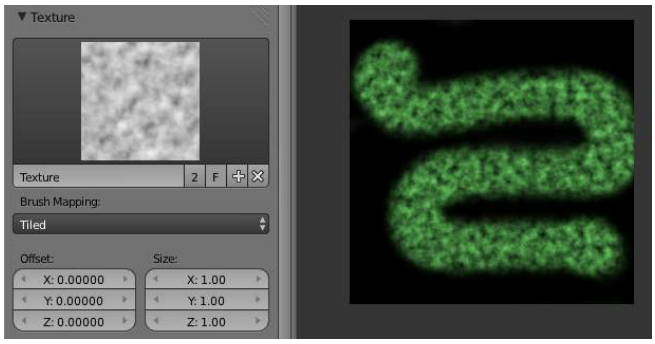


Fig. 2.968: Texture options and example.

Use the texture data-block at the bottom of the paint panel to select a pre-loaded image or procedural texture to use as your brush pattern. Note that in order to use it, you must have a placeholder material defined, and that particular texture defined using the Material and Texture buttons. It is not necessary to have that material or texture applied to any mesh anywhere; it must only be defined. The example to the right shows the effects of painting with a flat (banded) wood texture. Switching the texture to Rings makes a target/flower type of brush painting pattern.

Note: In Clone paint mode, this field changes to indicate the picture image or texture that you are cloning from.

Brush Mapping Sets how the texture is applied to the brush.

View Plane In 2D painting, the texture moves with the brush.

Tiled The texture is offset by the brush location.

3D Same as tiled mode.

Stencil Texture is applied only in borders of the stencil.

Random Random applying of texture.

Angle This is the rotation angle of the texture brush. It can be changed interactively via `Ctrl-F` in the 3D View. While in the interactive rotation you can enter a value numerically as well. Can be set to:

User Directly input the angle value.

Rake Angle follows the direction of the brush stroke. Not available with *3D* textures.

Random Angle is randomized.

Offset Offset the texture in X, Y, and Z.

Size Set the scale of the texture in each axis.

Texture Mask

TODO.

Stroke

Stroke Method Allows set the way applying strokes.

Airbrush Flow of the brush continues as long as the mouse click is held, determined by the *Rate* setting. If disabled, the brush only modifies the color when the brush changes its location.

Rate Interval between paints for airbrush.

Space Creates brush stroke as a series of dots, whose spacing is determined by the *Spacing* setting.

Spacing Represents the percentage of the brush diameter. Limit brush application to the distance specified by spacing.

Dots Apply paint on each mouse move step.

Jitter Jitter the position of the brush while painting.

Smooth stroke Brush lags behind mouse and follows a smoother path. When enabled, the following become active:

Radius Sets the minimum distance from the last point before stroke continues.

Factor Sets the amount of smoothing.

Input Samples Average multiple input samples together to smooth the brush stroke.

Wrap wraps your paint to the other side of the image as your brush moves off the **other** side of the canvas (any side, top/bottom, left/right). Very handy for making seamless textures.

Curve

The paint curve allows you to control the falloff of the brush. Changing the shape of the curve will make the brush softer or harder.

See also:

Read more about using the *Curve Widget*.

Options

Options tab.

Overlay

Allows you to customize the display of curve and texture that applied to the brush.

Appearance

Allows you to customize the color of the brush radius outline, as well as specify a custom icon.

Project Paint

TODO.

Vertex Paint

Introduction

Vertex Painting is a simple way of painting color onto an object, by directly manipulating the color of vertices, rather than textures, and is fairly straightforward.

When a vertex is painted, the color of the vertex is modified according to the rules of

the 'brush'. The color of all visible planes and edges attached to the vertex are then modified with a gradient to the color of the other connected vertices. (Note that the color of non-visible faces are not modified).

Vertex colors can be painted by first going into Edit Mode, then switching to *Vertex Paint Mode*; however, it will not show up in the render unless you check *Vertex Color Paint* in the *Materials Options* panel.

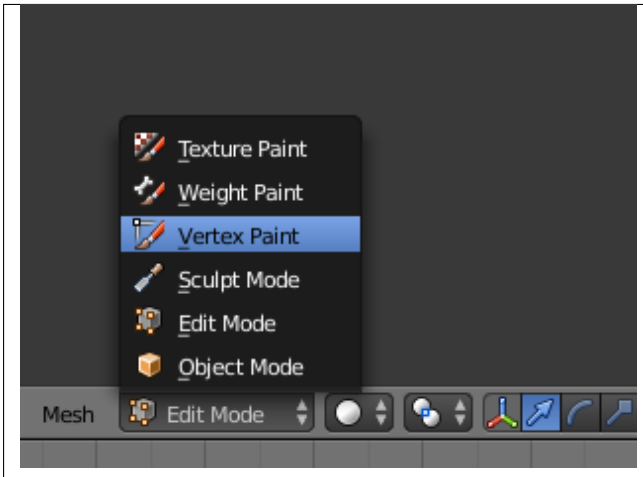


Fig. 2.969: Vertex Painting Mode.

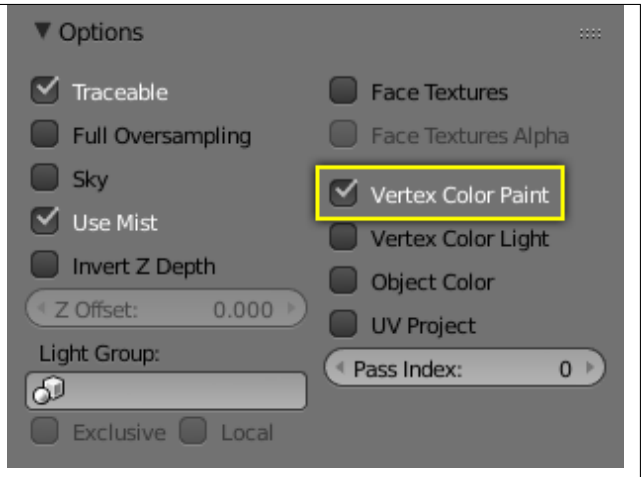


Fig. 2.970: Check this box.

Options

Tools

The Tools Shelf contains most of the options for vertex painting. The following sections describe the controls in each of the available panels.

Brush

Brush The *Data-Block menu* allows you to select brush presets, as well as custom brushes.

Color Color picker.

Radius Set the radius of the brush

Strength Set the strength of the brush's effect.

Blend

Mix Mixes RGB values. When set to a strength of 1.0, it will cover the underlying "paint".

Add Adds RGB values. Will eventually turn the entire object white as RGB values accumulate to (1.0, 1.0, 1.0): Pure White.

Subtract Subtracts RGB values. Usually results in Black.

Multiply Multiplies brush colors by the vertex colors.

Blur Blurs vertex colors.

Lighten Lightens the color of the vertices.

Darken Darkens the color of the vertices.

Texture

Use the texture data-block at the bottom of the paint panel to select a pre-loaded image or procedural texture to use as your brush pattern. Note that in order to use it, you must have a placeholder material defined, and that particular texture defined using the Material and Texture buttons. It is not necessary to have that material or texture applied to any mesh anywhere; it must only be defined.

Brush Mapping Mode Sets how the texture is applied to the brush.

View Plane In 2D painting, the texture moves with the brush.

Tiled The texture is offset by the brush location

3D Same as tiled mode

Stencil Texture is applied only in borders of the stencil.

Random Random applying of texture.

Angle This is the rotation angle of the texture brush. It can be changed interactively via `Ctrl-F` in the 3D View. While in the interactive rotation you can enter a value numerically as well. Can be set to:

User Directly input the angle value.

Rake Angle follows the direction of the brush stroke. Not available with *3D* textures.

Random Angle is randomized.

Offset Offset the texture in x, y, and z.

Size Set the scale of the texture in each axis.

Stroke

Stroke Method Allows set the way applying strokes.

Airbrush Flow of the brush continues as long as the mouse click is held, determined by the *Rate* setting. If disabled, the brush only modifies the color when the brush changes its location.

Rate Interval between paints for airbrush.

Space Creates brush stroke as a series of dots, whose spacing is determined by the *Spacing* setting.

Spacing Represents the percentage of the brush diameter. Limit brush application to the distance specified by spacing.

Dots Apply paint on each mouse move step.

Jitter Jitter the position of the brush while painting.

Smooth stroke Brush lags behind mouse and follows a smoother path. When enabled, the following become active:

Radius Sets the minimun distance from the last point before stroke continues.

Factor Sets the amount of smoothing.

Input Samples Average multiple input samples together to smooth the brush stroke.

Curve

Brush Curves affect how strongly the color is applied depending on distance from the center of the brush. In other words, they allow you to edit the Falloff of the brush intensity.

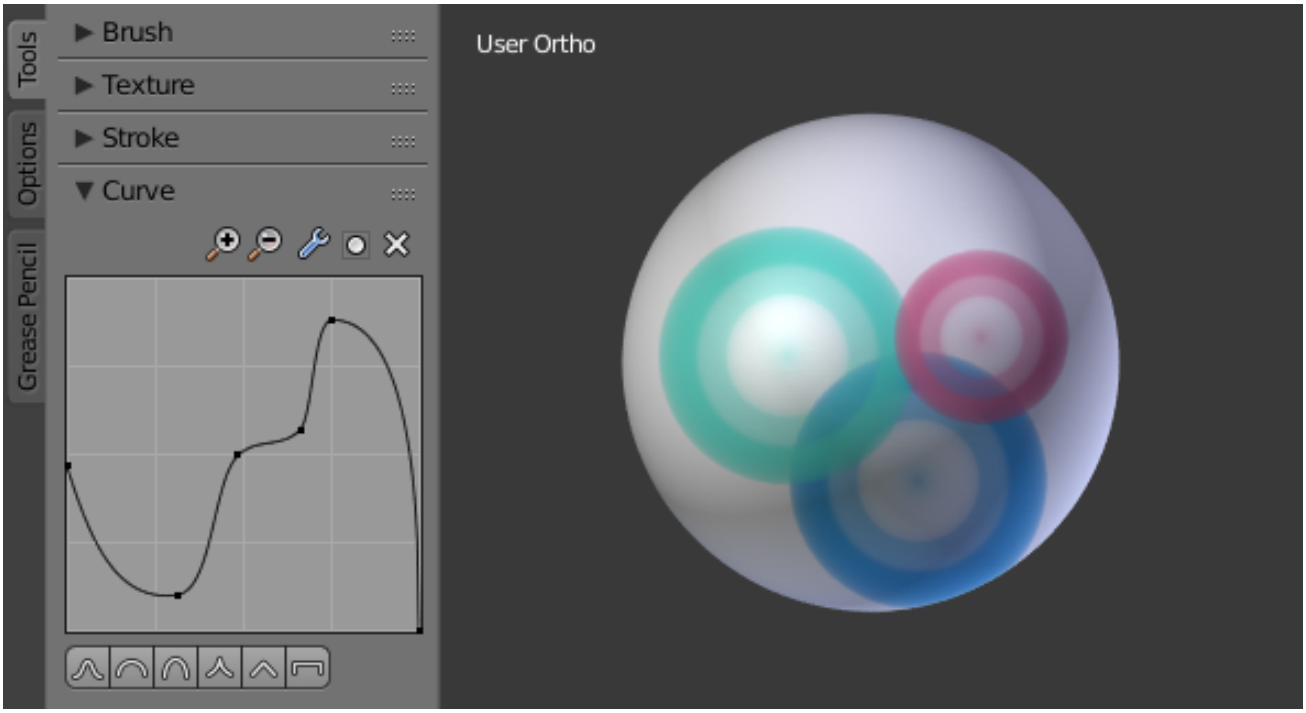
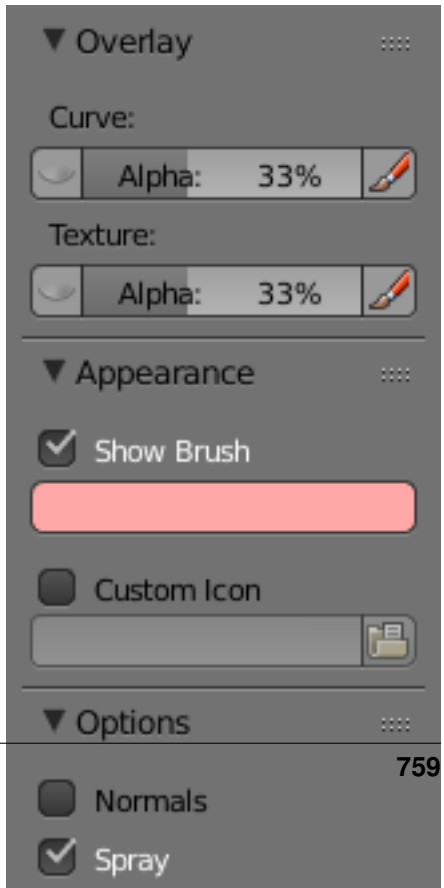


Fig. 2.972: Brush curve example.

Options

Overlay

Allows you to customize the display of curve and texture that applied to the brush.



Appearance

Allows you to customize the color of the brush radius outline, as well as specify a custom icon.

Options

Normals Applies the Vertex Normal before painting. This does not usually affect painting.

Spray Continues painting for as long as the mouse is held.

Unified Settings

Size All brushes use the same size.

Strength All brushes use the same strength.

Weight Paint

Introduction

Vertex Groups can potentially have a very large number of associated vertices and thus a large number of weights (one weight per assigned vertex). *Weight Painting* is a method to maintain large amounts of weight information in a very intuitive way. It is primarily used for rigging meshes, where the vertex groups are used to define the relative bone influences on the mesh. But we use it also for controlling particle emission, hair density, many modifiers, shape keys, etc.

The basic principle of the method is: the weight information is literally *painted* on top of the Mesh body by using a set of Weight brushes. And since painting is always associated with color, we also need to define ...

Weight Paint in a nutshell

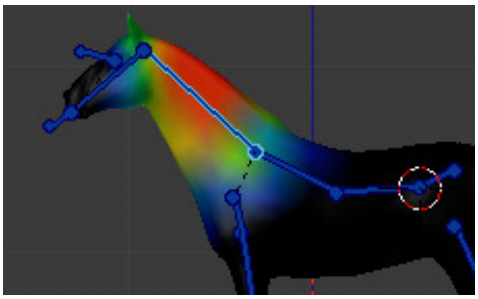


Fig. 2.974: Weight Painted Vertex Group.

- You enter *Weight Paint Mode* from the Mode Menu `Ctrl-Tab`. The selected Mesh Object is displayed slightly shaded with a rainbow color spectrum.
- The color visualizes the weights associated to each vertex in the active Vertex Group. Blue means unweighted; Red means fully weighted.
- You can customize the colors in the weight gradient by enabling *Custom Weight Paint Range* in the *System* tab of the *User Preferences*.
- You assign weights to the vertices of the Object by painting on it with weight brushes. Starting to paint on a mesh automatically adds weights to the active Vertex Group (a new Vertex Group is created if needed).

Tip: Useful Keyboard Shortcuts

The shortcuts can speed up your weight painting:

Weight color picker `Ctrl-LMB` change current weight value to the weight value of clicked vertex

Resize the brush `F` then drag to new brush size

Create linear gradient `Alt-LMB` then drag

Create radial gradient `Alt-Ctrl-LMB` then drag

Draw a *Clipping Border* `Alt-B` then drag the clipping border to select the part of the 3D View which shall be kept visible. You can then draw only in this part. Press `Alt-B` again to remove the *clipping border*.

The weighting Color Code

Weights are visualized by using a cold/hot color system, such that areas of low influence (with weights close to 0.0) are drawn in blue (cold) and areas of high influence (with weights close to 1.0) are drawn in red (hot). And all in-between influences are drawn in rainbow colors, depending on their value (blue, green, yellow, orange, red).

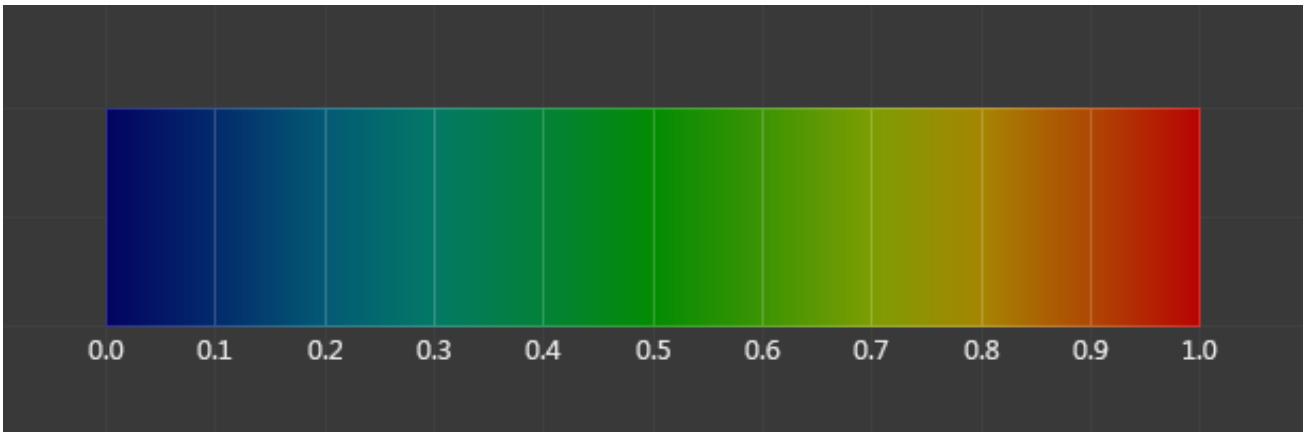


Fig. 2.975: The color spectrum and their respective weights.

In addition to the above described color code, Blender has added (as an option) a special visual notation for unreferenced vertices: They are drawn in black. Thus you can see the referenced areas (drawn in cold/hot colors) and the unreferenced areas (in black) at the same time. This is most practical when you look for weighting errors (we will get back to this later).

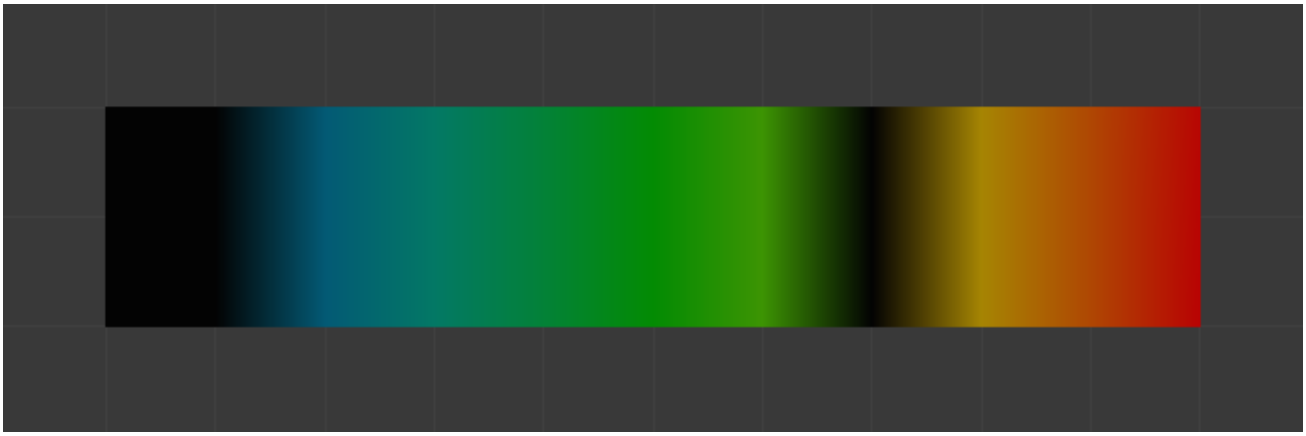


Fig. 2.976: Unreferenced vertices example.

Note: The color spectrum can be changed in the *User Preferences*.

Usage

Weight Painting for Bones

This is one of the main uses of weight painting. When a bone moves, vertices around the joint should move as well, but just a little, to mimic the stretching of the skin around the joint. Use a "light" weight (10 - 40%) paint on the vertices around the joint so that they move a little when the bone rotates. While there are ways to automatically assign weights to an armature (see the *Armature section*), you can do this manually. To do this from scratch, refer to the process below. To modify automatically assigned weights, jump into the middle of the process where noted:

1. Create an armature.
2. Create a mesh that will be deformed when the armature's bone(s) move.
3. With the mesh selected, create an *Armature* modifier for your mesh (located in the Properties editor, *Modifiers* tab). Enter the name of the armature.

   Pick up here for modifying automatically assigned weights.

1. Select the armature in 3D View, and bring the armature to *Pose Mode* with `Ctrl-Tab`, or the 3D View header mode selector.
2. Select a desired bone in the armature.
3. Select your mesh with `RMB` and change immediately to *Weight Paint Mode*. The mesh will be colored according to the weight (degree) that the selected bone movement affects the mesh. Initially, it will be all blue (no effect).
4. Weight paint to your heart's content. The mesh around the bone itself should be red (generally) and fade out through the rainbow to blue for vertices farther away from the bone.

   You may select a different bone with `RMB` while weight painting, provided the armature was left in *Pose Mode* as described above. This will activate the vertex group sharing the name with the selected bone, and display related weights. If the mesh skins the bones, you will not be able to see the bones because the mesh is painted. If so, turn on *X-Ray* view (Properties Editor, *Armature* tab).

   If you paint on the mesh, a vertex group is created for the bone. If you paint on vertices outside the group, the painted vertices are automatically added to the vertex group.

   If you have a symmetrical mesh and a symmetrical armature you can use the option *X-Mirror*. Then the mirrored groups with the mirrored weights are automatically created.

Weight Painting for Particles

Faces or vertices with zero weight generate no particles. A weight of 0. 1 will result in 10% of the amounts of particles. This option "conserves" the total indicated number of particles, adjusting the distributions so that the proper weights are achieved while using the actual number of particles called for. Use this to make portions of your mesh hairier than others by weight painting a vertex group, and then calling out the name of the vertex group in the *VGroup:* field (*Particles* panel, *Object* tab).

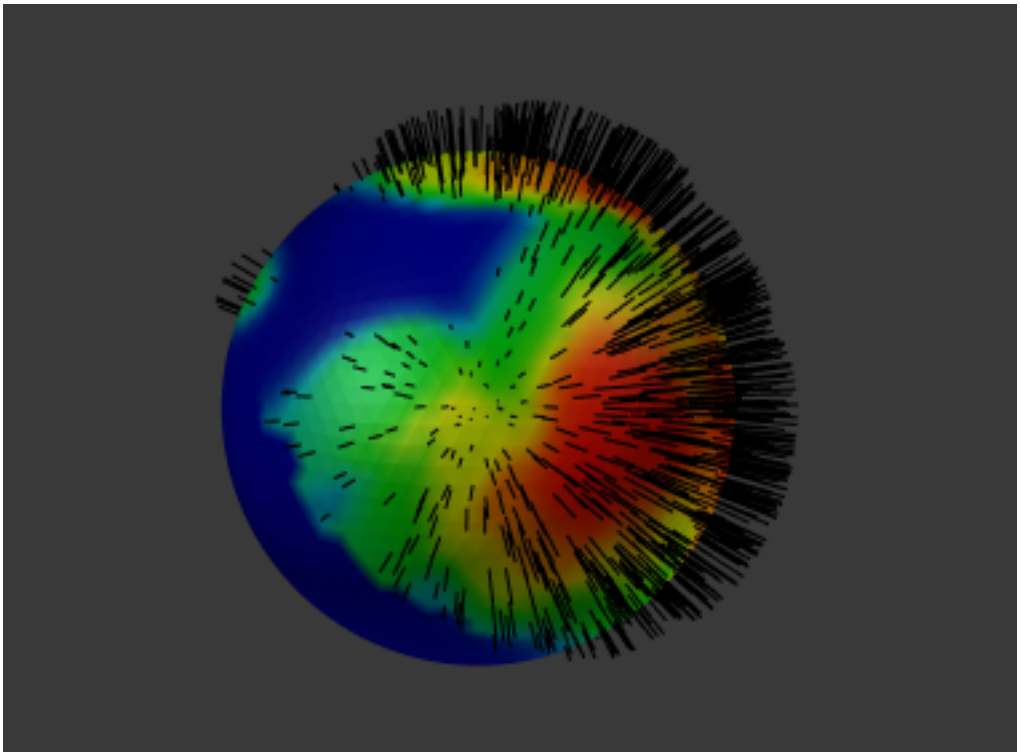


Fig. 2.977: Weight painted particle emission.

Properties

Tools

Brush

Painting needs paint brushes and Blender provides a Brush Panel within the Tool Shelf when it operates in *Weight Paint Mode*.

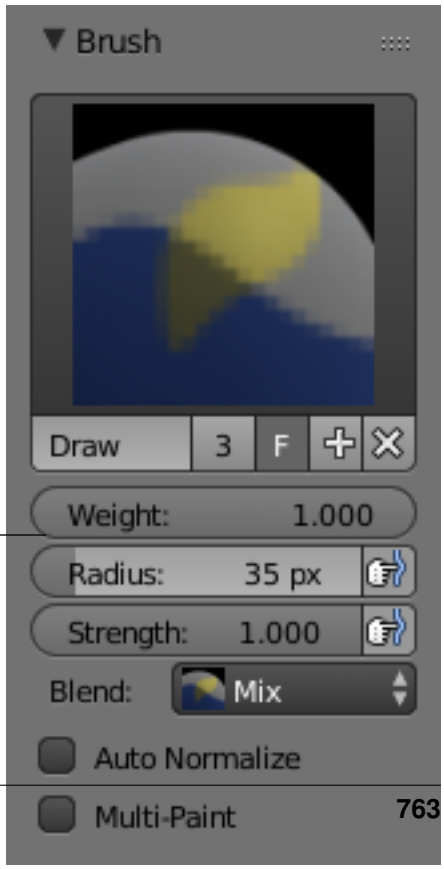


Fig. 2.978: Brush Panel.

Brush In the *Data-Block menu* you find predefined Brush Presets. And you can create your own custom presets as needed.

Weight The weight (color) to be used by the brush. However, the weight value is applied to the Vertex Group in different ways depending on the selected Brush Blending mode (see below).

Strength This is the amount of paint to be applied per brush stroke. What that means exactly also depends on the Brush Blending mode.

Radius The radius defines the area of influence of the brush.

> **Note:** You can also change the Brush radius with a keyboard shortcut while painting. Just press `F` at any time, then drag the mouse to increase/reduce the brush radius. Finally click `LMB` to use the new setting. Or press `Esc` at any time to return to the current settings.

Blend mode The brush Blending mode defines in which way the weight value is applied to the Vertex Group while painting.

Mix In this Blend mode the Weight value defines the *target weight* that will eventually be reached when you paint long enough on the

same location of the mesh. And the strength determines how many strokes you need to arrive at the target weight. Note that for strength = 1.0 the target weight is painted immediately, and for Weight = 0.0 the brush just does nothing.

Add In this blend mode the specified weight value is *added* to the vertex weights. The strength determines which fraction of the weight gets added per stroke. However, the brush will not paint weight values above 1.0.

Subtract In this blend mode the specified weight is *subtracted* from the vertex weights. The strength determines which fraction of the weight gets removed per stroke. However, the brush will not paint weight values below 0.0.

Lighten In this blend mode the specified weight value is interpreted as the target weight very similar to the Mix Blend mode. But only weights below the target weight are affected. Weights above the target weight remain unchanged.

Darken This Blend mode is very similar to the Lighten Blend mode. But only weights above the target weight are affected. Weights below the target weight remain unchanged.

Multiply Multiplies the vertex weights with the specified weight value. This is somewhat like subtract, but the amount of removed weight is now dependent on the Weight value itself.

Blur Smooths out the weighting of adjacent vertices. In this mode the Weight Value is ignored. The strength defines how much the smoothing is applied.

Accumulate This option keeps applying smoothing on top of the previous result.

Hint:

- Disable when painting individual vertices on lower poly modules.
- Enable for more dense geometry, or when you want to increase the blur effect.

Auto Normalize Ensures that all deforming vertex groups add up to one while painting. When this option is turned off, then all weights of a vertex can have any value between 0.0 and 1.0. However, when Vertex Groups are used as Deform Groups for character animation then Blender always interprets the weight values relative to each other. That is, Blender always does a normalization over all deform bones. Hence in practice it is not necessary to maintain a strict normalization and further normalizing weights should not affect animation at all.

This option works most intuitively when used to maintain normalization while painting on top of weights that are already normalized with some other tool.

Multi-Paint Paint on all selected Vertex Groups simultaneously, in a way that preserves their relative influence. This can be useful when tweaking weights in an area that is affected by more than three bones at once, e.g. certain areas on a character's face.

This option is only useful in the Armature tab, where you can select multiple Vertex Groups by selecting multiple Pose bones. Once at least two Vertex Groups are selected, viewport colors and paint logic switch to Multi-Paint Mode, using the sum of the selected groups' weights if Auto Normalize is enabled, and the average otherwise. Any paint operations

aimed at this collective weight are applied to individual Vertex Group weights in such way that their ratio stays the same.

Since the ratio is undefined if all weights are zero, Multi-Paint cannot operate on vertices that do not have any weight assigned to the relevant Vertex Groups. For this reason it also does not allow reducing the weight all the way to zero. When used with X-Mirror, it only guarantees completely symmetrical result if weights are initially symmetrical.

Tip: While Multi-Paint cannot directly paint on zero-weight vertices, it is possible to use the *Smooth Weight* tool to copy a reasonable non-zero weight distribution from adjacent vertices without leaving Multi-Paint Mode or changing bone selection.

To do that, enable vertex selection, select target vertices, and apply one iteration of the tool using vertex groups from *Selected Pose Bones* with low Factor. After that simply paint on top to set the desired collective weight.

Stroke

Stroke Method

Airbrush Keep applying paint effect while holding mouse down (spray).

Space Limit brush application to the distance specified by spacing (see below).

Dots Apply paint on each mouse move step.

Rate (only for Airbrush) Interval between paints for airbrush.

Spacing (only for Space) Limit brush application to the distance specified by spacing.

Jitter Jitter the position of the brush while painting.

Smooth Stroke Brush lags behind mouse and follows a smoother path.

Radius Minimum distance from last point before stroke continues.

Factor Higher values give a smoother stroke.

Curve

This *Curve widget* is used to control the brush falloff. Changing the curve allows you to specify the characteristics of your brushes to a large extent.

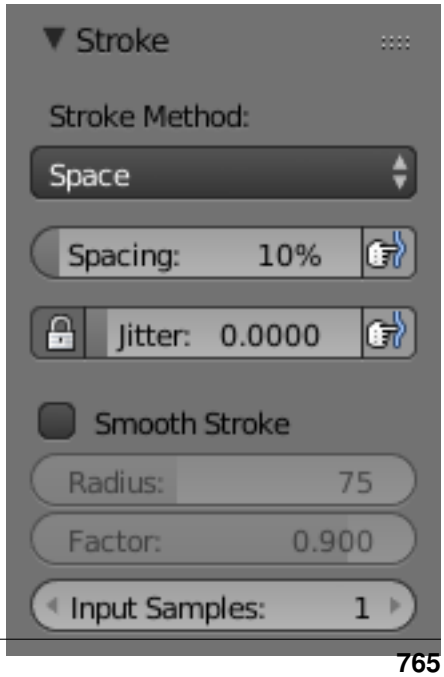


Fig. 2.979: Stroke Panel.

Weight Paint Tools

Blender provides a set of helper tools for Weight Painting. The tools are located in the weight tools panel.

The weight paint tools are full described in the *Weight Paint Tools* page.

Options

Overlay

Allows you to customize the display of curve and texture that applied to the brush.

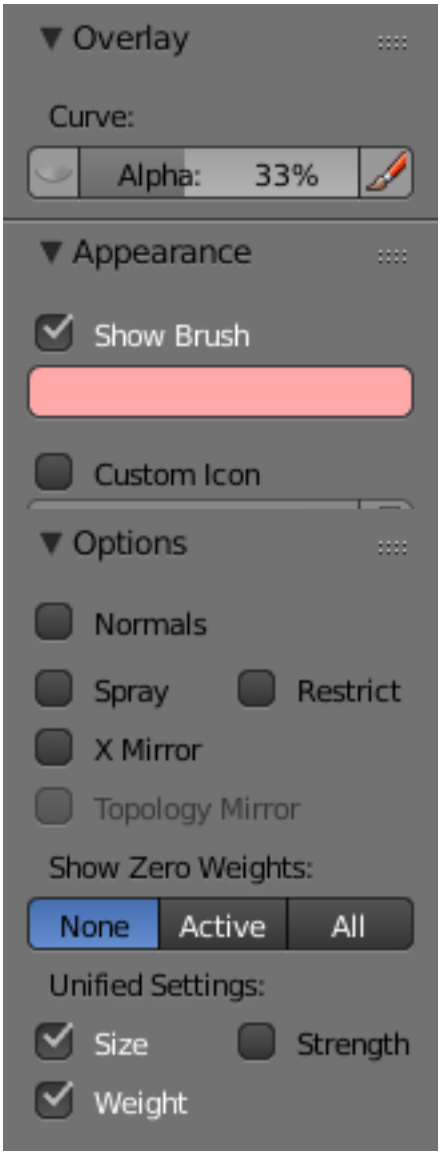


Fig. 2.981: Paint Options.

Appearance

Show Brush Makes the brush visible as a circle (on by default).

Custom Icon Allows definition of a custom brush icon.

Options

The Weight Paint Options modify the overall brush behavior:

Normals The vertex normal (helps) determine the extent of painting. This causes an effect as if painting with light.

Spray Constantly draw (opposed to drawing one stroke per mouse click).

Restrict This option limits the influence of painting to vertices belonging (even with weight 0) to the selected vertex group.

X-mirror Use the X-mirror option for mirrored painting on groups that have symmetrical names, like with extension ".R"/ ".L" or "_R" / "_L". If a group has no mirrored counterpart, it will paint symmetrically on the active group itself. You can read more about the naming convention in *Editing Armatures: Naming conventions*. The convention for armatures/bones apply here as well.

Topology Mirror Use topology-based mirroring, for when both side of a mesh have matching mirrored topology.

Show Zero Weights To display unreferenced and zero weighted areas in black (by default).

- None
- Active
- All

Unified Settings The *Size*, *Strength* and *Weight* of the brush can be set to be shared across different brushes, as opposed to per-brush.

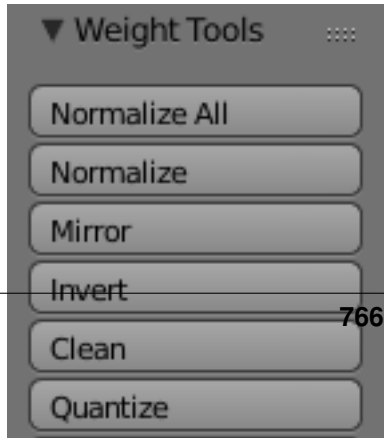
Weight Tools

Blender provides a set of helper tools for Weight Painting. The tools are accessible from the Tool Shelf in Weight Paint Mode. And they are located in the weight tools panel.

The Subset Option

Some of the tools also provide a Subset parameter (in the Operator panel, displayed after the tool is called) with following options:

- Active Group
- Selected Pose Bones
- Deform pose Bones



- All Groups

  All tools also work with Vertex Selection Masking and Face Selection masking. In these modes the tools operate only on selected verts or faces.

Tip: About the Blend tool

The Blend tool only works when "Vertex selection masking for painting" is enabled. Otherwise the tool button is grayed out.

Normalize All

For each vertex, this tool makes sure that the sum of the weights across all Vertex Groups is equal to 1. This tool normalizes all of the vertex groups, except for locked groups, which keep their weight values untouched.

Operator Parameters

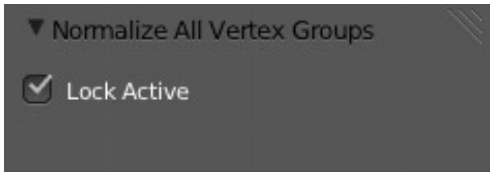


Fig. 2.983: Normalize All Options.

Lock Active Keep the values of the active group while normalizing all the others.

Note: Currently this tool normalizes **all** vertex groups except the locked vertex groups.

Normalize

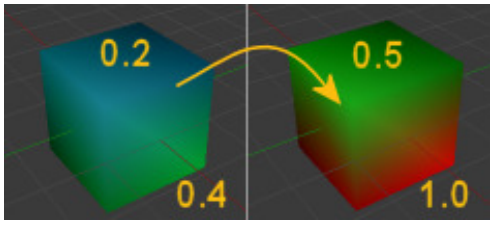


Fig. 2.984: Normalize All Options.

This tool only works on the active Vertex Group. All vertices keep their relative weights, but the entire set of weights is scaled up such that the highest weight value is 1.0 .

Mirror

This tool mirrors the weights from one side of the mesh to the opposite side (only mirroring along x-axis is supported). But note, the weights are not transferred to the corresponding opposite bone weight group. The mirror only takes place within the selected Vertex Group.

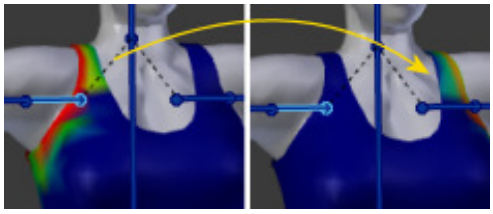


Fig. 2.985: Normalize All Options.

Operator Parameters

Fig. 2.986: Mirror Options.

Mirror Weights Mirrors the weights of the active group to the other side. Note, this only affects the active weight group.

Flip Group Names Exchange the names of left and right side. This option only renames the groups.

All Groups Operate on all selected bones.

Topology Mirror Mirror for meshes which are not 100% symmetric (approximate mirror).

Tip: Mirror to opposite bone

If you want to create a mirrored weight group for the opposite bone (of a symmetric character), then you can do this:

1. Delete the target Vertex Group (where the mirrored weights will be placed).
2. Create a copy of the source bone Vertex Group (the group containing the weights which you want to copy).
3. Rename the new Vertex Group to the name of the target Vertex Group (the group you deleted above).
4. Select the Target Vertex Group and call the Mirror tool (use only the Mirror weights option and optionally Topology Mirror if your mesh is not symmetric).

Invert

Replaces each Weight of the selected weight group by × -1.0 weight.

Examples:

- original 1.0 converts to 0.0
- original 0.5 remains 0.5
- original 0.0 converts to 1.0

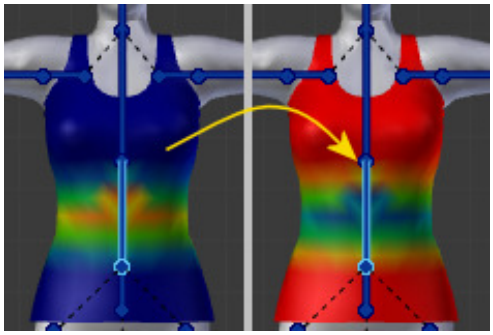


Fig. 2.987: Invert.

Operator Parameters

Fig. 2.988: Invert Options.

Subset Restrict the tool to a subset. See above *The Subset Option* about how subsets are defined.

Add Weights Add verts that have no weight before inverting (these weights will all be set to 1.0)

Remove Weights Remove verts from the Vertex Group if they are 0.0 after inverting.

Note: Locked vertex Groups are not affected.

Clean

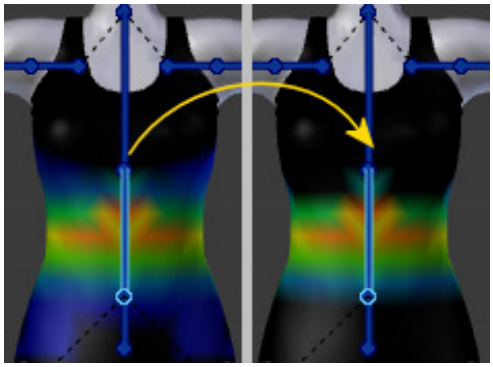


Fig. 2.989: Invert.

Removes weights below a given threshold. This tool is useful for clearing your weight groups of very low (or zero-) weights.

In the example shown, a cutoff value of 0.139 is used (see operator options below) so all blue parts (left side) are cleaned out (right side).

Note, the images use the *Show Zero weights* =Active option so that unreferenced Weights are shown in Black.

Operator Parameters

Fig. 2.990: Clean Options.

Subset Restrict the tool to a subset. See above *The Subset Option* for how subsets are defined.

Limit This is the minimum weight value that will be kept in the Group. Weights below this value will be removed from the group.

Keep Single Ensure that the Clean tool will not create completely unreferenced verts (verts which are not assigned to any Vertex Group), so each vertex will keep at least one weight, even if it is below the limit value!

Levels

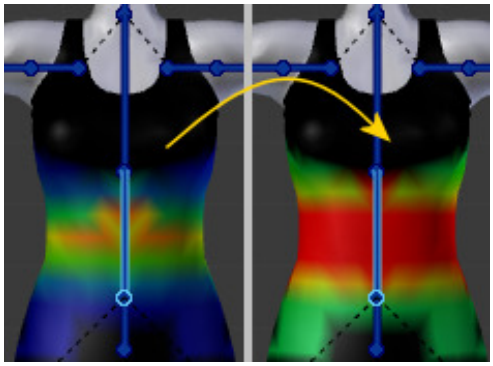


Fig. 2.991: Invert.

Adds an offset and a scale to all weights of the selected Weight Groups. with this tool you can raise or lower the overall "heat" of the weight group.

Note: No weight will ever be set to values above 1.0 or below 0.0 regardless of the settings.

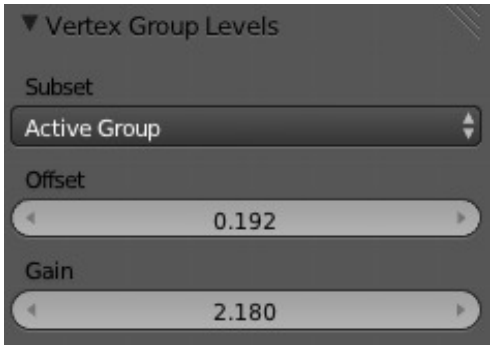


Fig. 2.992: Levels Options.

Operator Parameters

Subset Restrict the tool to a subset. See above *The Subset Option* for how subsets are defined.

Offset A value from the range (-1.0 - 1.0) to be added to all weights in the Vertex Group.

Gain All weights in the Subset are multiplied with the gain.

Note: Whichever *Gain* and *Offset* you choose, in all cases the final value of each weight will be clamped to the range (0.0 - 1.0). So you will never get negative weights or overheated areas (weight > 1.0) with this tool.

Blend

Blends the weights of selected vertices with adjacent unselected vertices. This tool only works in vertex select mode.

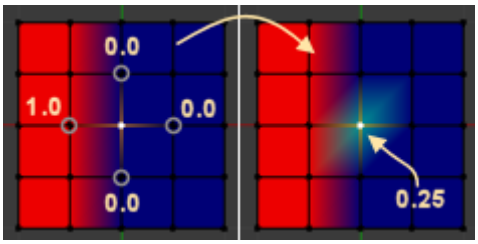


Fig. 2.993: Blending.

To understand what the tool really does, let us take a look at a simple example. The selected vertex is connected to four adjacent vertices (marked with a gray circle in the image). All adjacent vertices are unselected. Now the tool calculates the average weight of all connected **and** unselected verts. In the example this is:

$(1 + 0 + 0 + 0)/4 = 0.25$

This value is multiplied by the factor given in the Operator parameters (see below).

- If the factor is 0.0 then actually nothing happens at all and the vertex just keeps its value.
- If the factor is 1.0 then the calculated average weight is taken (0.25 here).
- Dragging the factor from 0 to 1 gradually changes from the old value to the calculated average.

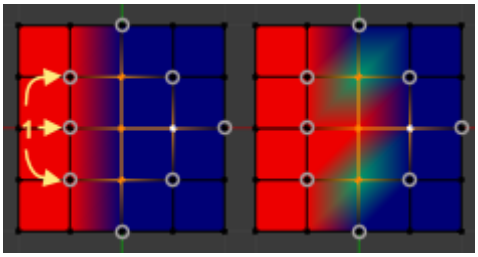


Fig. 2.994: Blending.

Now let us see what happens when we select all but one of the neighbors of the selected vert as well. Again all connected and unselected verts are marked with a gray circle. When we call the Blend tool now and set the Factor to 1.0, then we see different results for each of the selected verts:

- The topmost and bottommost selected verts:

  are surrounded by three unselected verts, with an average weight of $(1+0+0)/3 = 0.333$ So their color has changed to light green.

- The middle vertex:

  is connected to one unselected vert with `weight = 1`. So the average weight is 1.0 in this case, thus the selected vert color has changed to red.

- The right vert:

  is surrounded by three unselected verts with average weight = $(0+0+0)/3 = 0.0$ So the average weight is 0, thus the selected vert color has not changed at all (it was already blue before blend was applied).

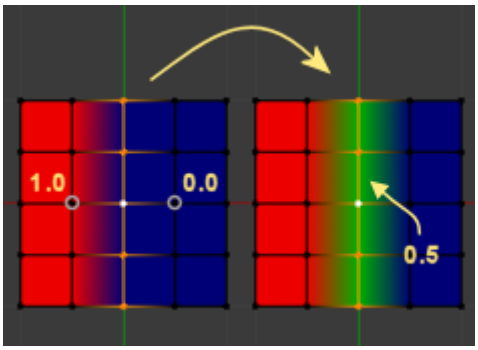


Fig. 2.995: Blending.

Finally let us look at a practical example (and explain why this tool is named Blend). In this example, the middle edge loop has been selected and it will be used for blending the left side to the right side of the area.

- All selected vertices have two unselected adjacent verts.
- The average weight of the unselected verts is $(1+0)/2 = 0.5$
- Thus when the Blend Factor is set to 1.0 then the edge loop turns to green and finally does blend the cold side (right) to the hot side (left).

Operator Parameters

Factor The effective amount of blending. When Factor is set to 0.0 then the Blend tool does not do anything. For Factor > 0 the weights of the affected vertices gradually

Fig. 2.996: Blend Options.

shift from their original value towards the average weight of all connected **and** unselected verts (see examples above).

Transfer Weights

Copy weights from other objects to the vertex groups of the active Object. By default this tool copies all vertex groups contained in the selected objects to the target object. However, you can change the tool's behavior in the operator redo panel (see below).

Prepare the Copy

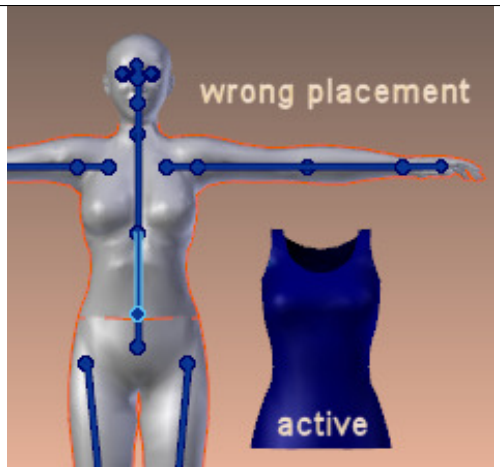


Fig. 2.997: Blending.

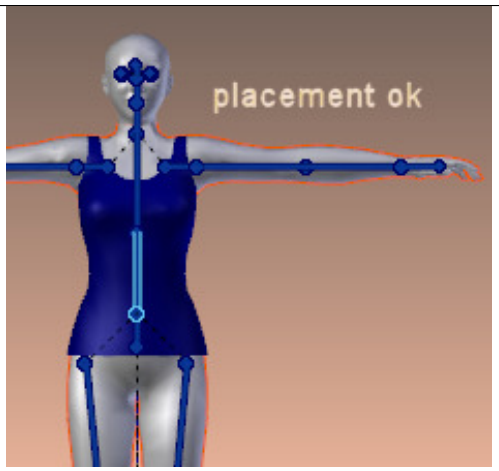


Fig. 2.998: Blending.

You first select all source objects, and finally the target object (the target object must be the active object).

It is important that the source objects and the target object are at the same location. If they are placed side by side, then the weight transfer will not work. You can place the objects on different layers, but you have to ensure that all objects are visible when you call the tool.

Now ensure that the Target Object is in Weight Paint Mode.

Call the Tool

Open the Tool Shelf and locate the Weight Tools panel. From there call the "Transfer weights" tool. The tool will initially copy all vertex groups from the source objects. However, the tool also has an operator redo panel (which appears at the bottom of the tool shelf). From the redo panel you can change the parameters to meet your needs. (The available Operator parameters are documented below.)

Redo Panel Confusion

You may notice that the Operator Redo Panel (see below) stays available after the weight transfer is done. The panel only disappears when you call another Operator that has its own redo Panel. This can lead to confusion when you use Transfer weights repeatedly after you changed your vertex groups. If you then use the still-visible redo panel, then Blender will reset your work to its state right before you initially called the Transfer Weights tool.

Workaround

When you want to call the Transfer Weights tool again after you made some changes to your vertex groups, then always use the "Transfer Weights" Button, even if the operator panel is still available. Unless you really want to reset your changes to the initial call of the tool.

Operator Parameters

Note: This tool now uses the generic 'data transfer' one. Please refer to the *Data Transfer* docs for options details and explanations.

Limit Total

Reduce the number of weight groups per vertex to the specified Limit. The tool removes lowest weights first until the limit is reached.

Hint: The tool can only work reasonably when more than one weight group is selected.

Operator Parameters

Subset Restrict the tool to a subset. See above *The Subset Option* for how subsets are defined.

Limit Maximum number of weights allowed on each vertex.

Weight Gradient

This is an interactive tool for applying a linear/radial weight gradient; this is useful at times when painting gradual changes in weight becomes difficult.

The gradient tool can be accessed from the Tool Shelf or as a key shortcut:

- Linear: `Alt-LMB` and drag.
- Radial: `Alt-Ctrl-LMB` and drag.

The following weight paint options are used to control the gradient:

Weight The gradient starts at the current selected weight value, blending out to nothing.

Strength Lower values can be used so the gradient mixes in with the existing weights (just like with the brush).

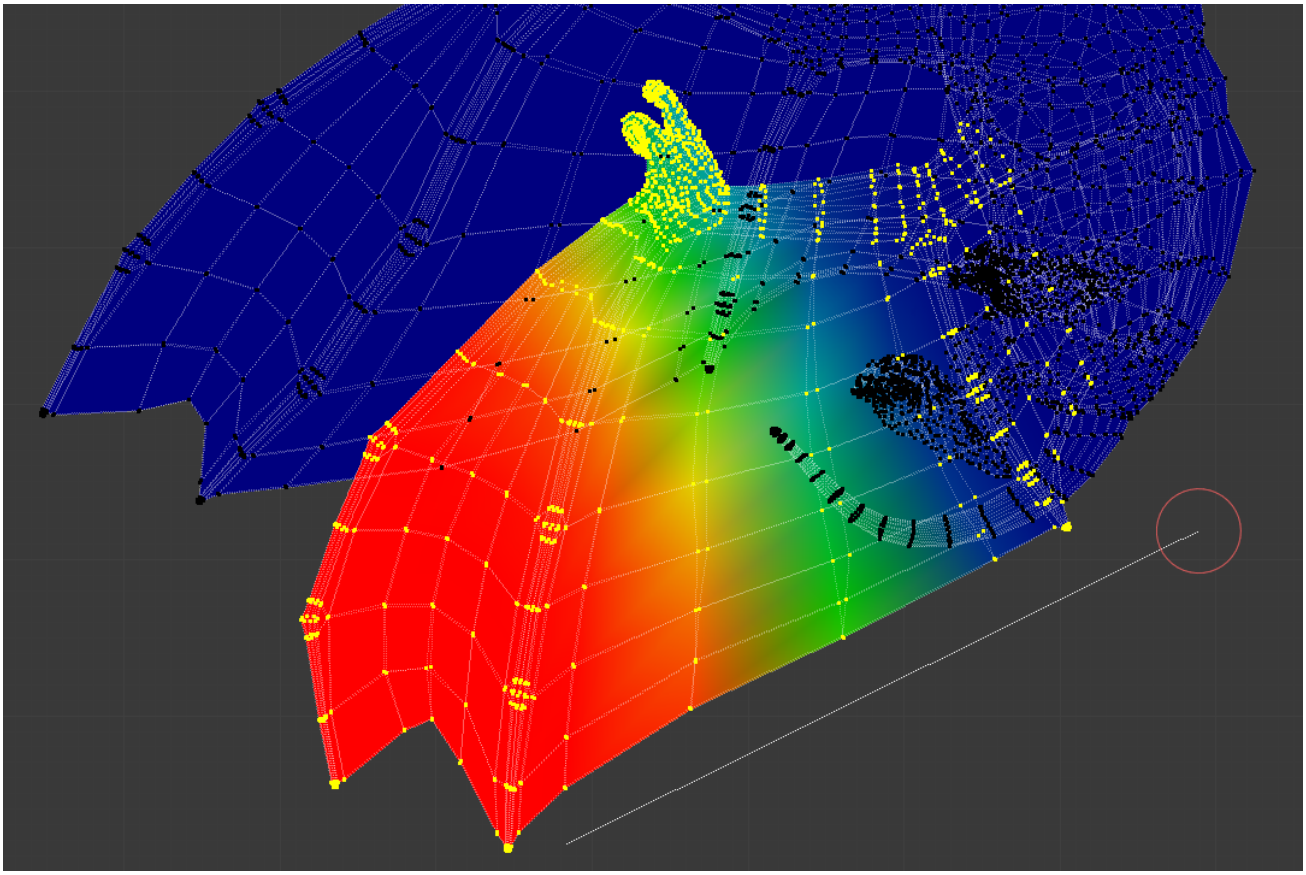


Fig. 2.999: Example of the gradient tool being used with selected vertices.

Curve The brush falloff curve applies to the gradient too, so you can use this to adjust the blending.

Blends the weights of selected vertices with unselected vertices.

Hint: This tool only works in vertex select mode.

Operator Parameters

Type

- Linear
- Radial

Hiding & Masking

Selection Masking

If you have a complex mesh, it is sometimes not easy to paint on all vertices in Weight Paint Mode. Suppose you only want to paint on a small area of the Mesh and keep the rest untouched. This is where *selection masking* comes into play. When this mode is enabled, a brush will only paint on the selected verts or faces. The option is available from the header of the 3D View (see icons surrounded by the yellow frame):

Fig. 2.1000: You can choose between *Face Selection masking* (left icon) and *Vertex selection masking* (right icon).

Select mode has some advantages over the default *Weight Paint Mode*:

- The original mesh edges are drawn, even when modifiers are active.
- You can select faces to restrict painting to the vertices of the selected faces.
- Selecting tools include:

Details about selecting

The following standard selection operations are supported:

- `RMB` - Single faces. Use `Shift-RMB` to select multiple.
- `A` - All faces, also to de-select.
- `B` - Block/Box selection.
- `C` - Select with brush.
- `L` - Pick linked (under the mouse cursor).
- `Ctrl-L` - Select linked.
- `Ctrl-I` - Invert selection *Inverse*.

Tip: Selecting Deform Groups

When you are doing weight painting for deform bones (with an Armature), you can select a deform group by selecting the corresponding bone. However, this Vertex Group selection mode is disabled when Selection Masking is active!

Vertex Selection Masking

In this mode you can select one or more vertices and then paint only on the selection.
All unselected vertices are protected from unintentional changes.

Note: This option can also be toggled with `V`.

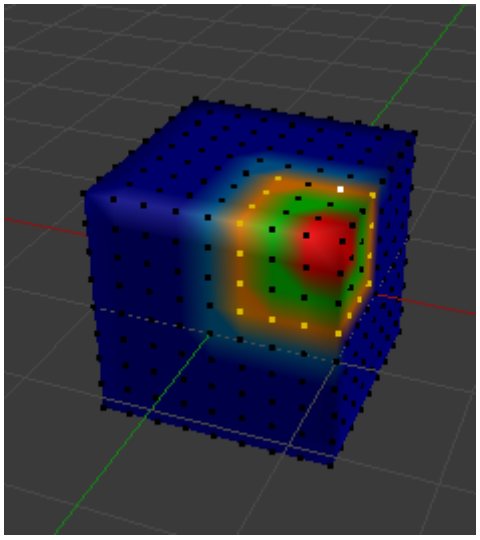


Fig. 2.1001: Vertex Selection masking.

Face Selection Masking

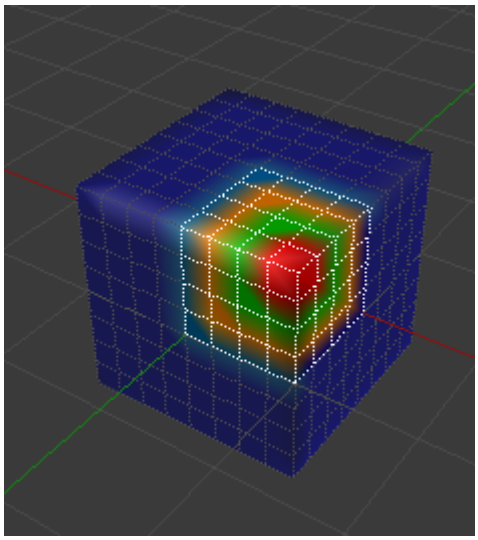


Fig. 2.1002: Face Selection masking.

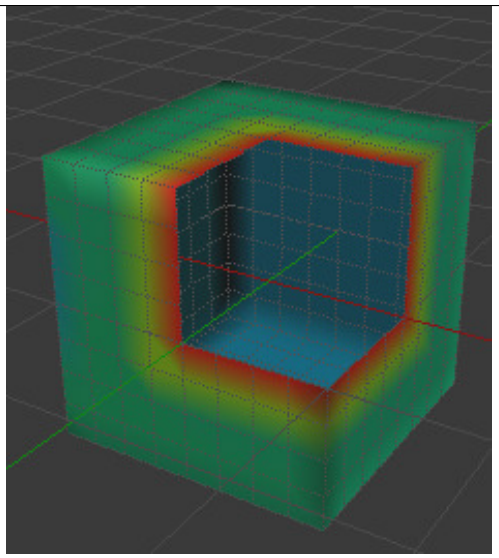


Fig. 2.1003: Hidden faces.

The *Face Selection masking* allows you to select faces and limit the weight paint tool to those faces, very similar to Vertex selection masking.

Hide/Unhide Faces

You also can hide selected faces as in Edit Mode with the keyboard Shortcut `H`, then paint on the remaining visible faces and finally unhide the hidden faces again by using `Alt-H`

Hide/Unhide Vertices

You cannot directly hide selected faces in vertex mask selection mode. However, you can use a trick:

1. First go to Face selection mask mode.
2. Select the areas you want to hide and then hide the faces (as explained above).
3. Switch back to Vertex Selection mask mode.

   Now the verts belonging to the hidden Faces will remain hidden.

The Clipping Border

To constrain the paint area further you can use the *Clipping Border*. Press `Alt-B` and `LMB` -drag a rectangular area. The selected area will be "cut out" as the area of interest. The rest of the 3D View gets hidden.

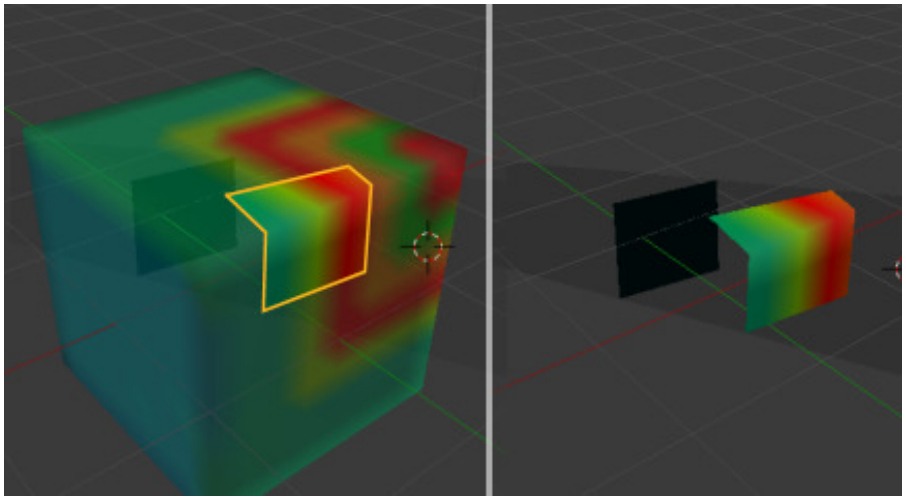


Fig. 2.1004: The Clipping Border is used to select interesting parts for local painting.

You make the entire mesh visible again by pressing `Alt-B` a second time.

All weight paint tools that use the view respect this clipping, including border select, weight gradient and of course brush strokes.

2.5.3 Sculpting

Introduction

Overview

Sculpt Mode is similar to *Edit Mode* in that it is used to alter the shape of a model, but Sculpt Mode uses a very different workflow: instead of dealing with individual elements (vertices, edges, and faces), an area of the model is altered using a brush. In other words, instead of selecting a group of vertices, Sculpt Mode automatically selects vertices based on where the brush is, and modifies them accordingly.

Sculpt Mode

Sculpt mode is selected from the mode menu of the *3D View* header. Once sculpt mode is activated the Tool Shelf of the *3D View* will change to sculpt mode specific panels. The panels will be *Brush*, *Texture*, *Tool*, *Symmetry*, *Stroke*, *Curve*, *Appearance*, and *Options*. Also a red circle will appear that follows the location of the cursor in the 3D View.

Note: To have a predictable brush behavior, apply the scale of your mesh.

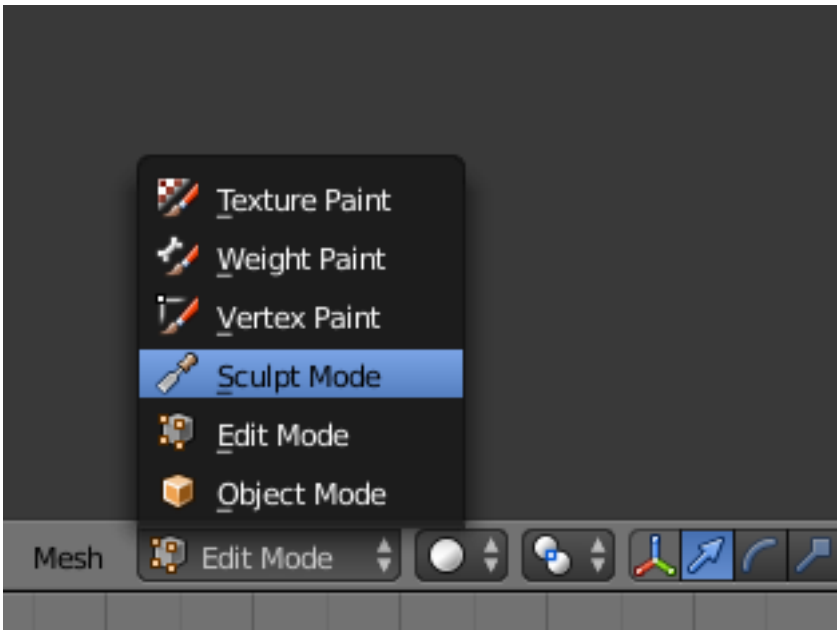


Fig. 2.1005: 3D View Mode selector: Sculpt Mode.

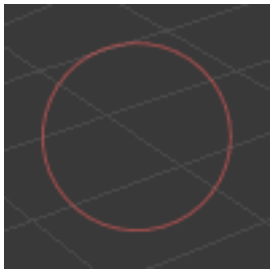


Fig. 2.1006: The cursor in Sculpt Mode.

Sculpt Menus

Tool Menu

Here you can select the type of brush preset to use. *Reset Brush* will return the settings of a brush to its defaults. You can also set Blender to use the current brush for *Vertex Paint Mode*, *Weight Paint Mode*, and *Texture Paint Mode* using the toggle buttons.

Keyboard Shortcuts

These shortcuts may be customized under *File* → *User Preferences* → *Input* → *3D View* → *Sculpt Mode*.

Table 2.37: Brush Selection Shortcuts.

Draw brush	X
Smooth brush	S
Pinch/Magnify brush	P
Inflate/Deflate brush	I
Grab brush	G
Layer brush	L
Flatten/Contrast brush	Shift-T
Clay brush	C
Crease brush	Shift-C
Snake Hook brush	K
Mask brush	M
Set brush by number	0 - 9 and Shift-0 to Shift-9

Table 2.38: Brush Option Shortcuts.

Interactively set brush size	F
Increase/decrease brush size	[and]
Interactively set brush strength	Shift-F
Interactively rotate brush texture	Ctrl-F
Brush direction toggle (*Add* / *Sub*)	Ctrl pressed while sculpting
Brush normal weight toggle	Ctrl toggle *Normal Weight*. (for *Grab* and *Snake Hook* brushes).
Set stroke method (airbrush, anchored, ..)	E
Toggle Smooth Stroke	Shift-S
Smooth stroke toggle	Shift
Set texture angle type	R
Translate/scale/rotate stencil texture	RMB, Shift-RMB, Ctrl-RMB
Translate/scale/rotate stencil mask	Alt-RMB, Alt-Shift-RMB, Alt-Ctrl-RMB

Table 2.39: Other Shortcuts.

Hide mesh inside selection	H then click & drag
Reveal mesh inside selection	Shift-H then click & drag
Show entire mesh	Alt-H
Mask clear	Alt-M
Mask invert	Ctrl-I
Step up one multires level	PageUp
Step down one multires level	PageDown
Set multires level	Ctrl-0 to Ctrl-5
Dynamic Topology toggle	Ctrl-D
Dynamic Topology detail	Shift-D

Options

Tools Tab

Brush

Brush Type Brushes are brush presets. They are a combination of a 'tool', along with stroke, texture, and options.

Blob Pushes mesh outward or inward into a spherical shape with settings to control the amount of pinching at the edge of the sphere.

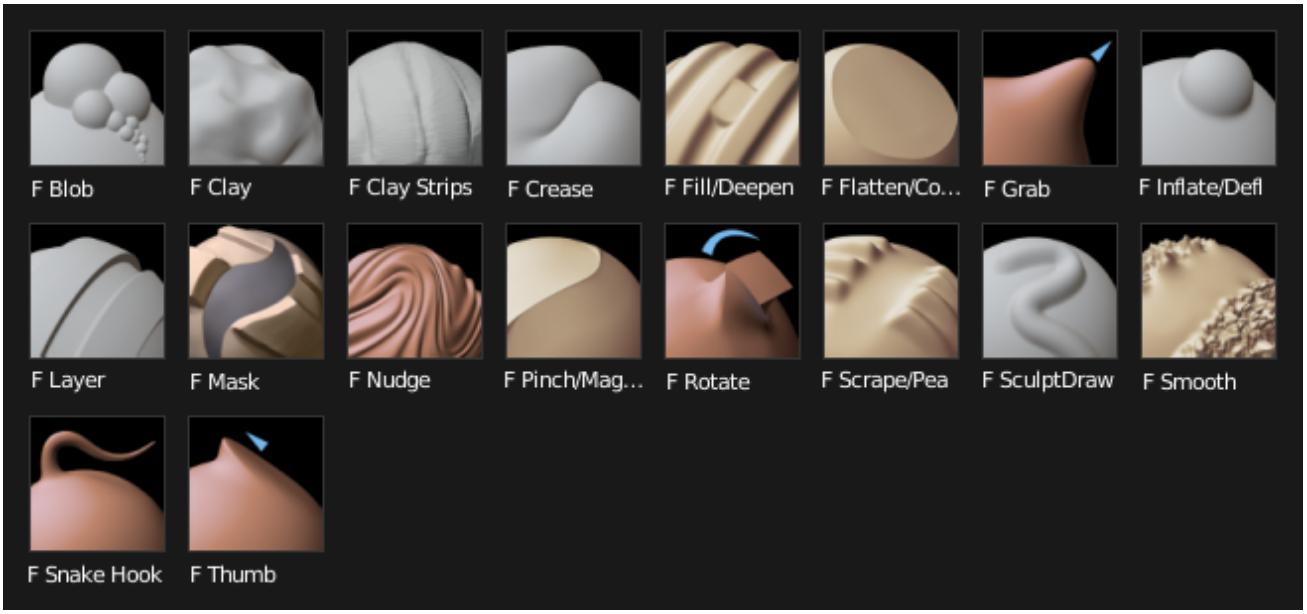


Fig. 2.1007: Sculpt brushes.

Clay Similar to the *Draw* brush, but includes settings to adjust the plane on which the brush acts.

Clay Strips Similar to the *Clay* brush, but it uses a cube test to define the brush area of influence rather than a sphere.

Crease Creates sharp indents or ridges by pushing or pulling the mesh, while pinching the vertices together.

Draw Moves vertices inward or outward, based the average normal of the vertices contained within the drawn brush stroke.

Fill Works like the Flatten brush, but only brings vertices below the brush plane upwards. The inverse of the Scrape brush is to *Deepen* by pushing vertices above the plane downward.

Flatten The *Flatten* brush finds an 'area plane' located by default at the average height above/below the vertices within the brush area. The vertices are then pulled towards this plane. The inverse of the Flatten brush is the *Contrast* brush which pushes vertices up or down away from the brush plane.

Grab Used to drag a group of points around. Unlike the other brushes, *Grab* does not modify different points as the brush is dragged across the model. Instead, *Grab* selects a group of vertices on mousedown, and pulls them to follow the mouse. The effect is similar to moving a group of vertices in *Edit Mode* with proportional-editing enabled, except that *Grab* can make use of other Sculpt Mode options (like textures and symmetry).

Inflate Similar to *Draw*, except that vertices in *Inflate* mode are displaced in the direction of their own normals.

Layer This brush is similar to *Draw*, except that the height of the displacement layer is capped. This creates the appearance of a solid layer being drawn. This brush does not draw on top of itself; a brush stroke intersects itself. Releasing the mouse button and starting a new stroke will reset the depth and paint on top of the previous stroke.

Nudge Moves vertices in the direction of the brush stroke.

Pinch Pulls vertices towards the center of the brush. The inverse setting is *Magnify*, in which vertices are pushed away from the center of the brush.

Rotate Rotates vertices within the brush in the direction the cursor is moved.

Scrape The *Scrape* brush works like the Flatten brush, but only brings vertices above the plane downwards. The inverse of the Scrape brush is to *Peak* by pushing vertices above the plane up away from the plane.

Smooth As the name suggests, eliminates irregularities in the area of the mesh within the brush's influence by smoothing the positions of the vertices.

Snake Hook Pulls vertices along with the movement of the brush to create long, snake-like forms.

Thumb Similar to the *Nudge* brush, this one flattens the mesh in the brush area, while moving it in the direction of the brush stroke.

Radius This option controls the radius of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse and then `LMB` (the texture of the brush should be visible inside the circle). Typing a number then enter while using `F` allows you to enter the size numerically. Brush size can be affected by enabling the pressure sensitivity icon, if you are using a *Graphics Tablet*.

Strength Controls how much each application of the brush affects the model. For example, higher values cause the *Draw* brush to add depth to the model more quickly, and cause the *Smooth* brush to smooth the model more quickly. This setting is not available for *Grab*, *Snake Hook*, or *Rotate*.

You can change the brush strength interactively by pressing `Shift-F` in the 3D View and then moving the brush and then `LMB`. You can enter the size numerically also while in `Shift-F` sizing. Brush strength can be affected by enabling the pressure sensitivity icon, if a supported tablet is being used.

Tip: If the range of strengths does not seem to fit the model (for example, if even the lowest strength setting still makes too large of a change on the model) then you can scale the model (in *Edit Mode*, not *Object Mode*). Larger sizes will make the brush's effect smaller, and vice versa.

Autosmooth Sets the amount of smoothing to be applied to each stroke.

Normal Weight Constrains brush movement along the surface normal. Especially useful with the *Grab Brush*, can be temporarily enabled by holding `Ctrl`.

Applies to *Grab* and *Snake Hook* brushes.

Sculpt Plane Use this menu to set the plane in which the sculpting takes place.

Front Faces Only When enabled, the brush only affects vertices that are facing the viewer.

Add/Subtract TODO.

Accumulate Causes stroke dabs to accumulate on top of each other.

Texture Panel

Texture Texture to be used to determine the strength of brush.

Brush Mapping Sets the way the texture is mapped to the brush stroke:

Fixed If *Fixed* is enabled, the texture follows the mouse, so it appears that the texture is being dragged across the model.

Tiled The *Tile* option tiles the texture across the screen, so moving the brush appears to move separately from the texture. The *Tile* option is most useful with tileable images, rather than procedural textures.

3D The *3D* option allows the brush to take full advantage of procedural textures. This mode uses vertex coordinates rather than the brush location to determine what area of the texture to use.

Angle This is the rotation angle of the texture brush. It can be changed interactively via `Ctrl-F` in the 3D View. While in the interactive rotation you can enter a value numerically as well. Can be set to:

User Directly input the angle value.

Rake Angle follows the direction of the brush stroke. Not available with *3D* textures.

Random Angle is randomized.

Rake TODO.

Random TODO.

Offset Fine tunes the texture map placement in the x, y, and z axes.

Size This setting allows you to modify the scaling factor of the texture. Not available for *Drag* textures.

Sample Bias Value added to texture samples.

Stroke Panel

Stroke Method Defines the way brush strokes are applied to the mesh:

Dots Standard brush stroke.

Drag Dot Creates a single displacement in the brush shape. Click then drag on mesh to desired location, then release.

Space Creates brush stroke as a series of dots, whose spacing is determined by the *Spacing* setting. *Spacing* represents the percentage of the brush diameter.

Anchored Creates a single displacement at the brush location. Clicking and dragging will resize the brush diameter. When *Edge to Edge* the brush location and orientation is determined by a two point circle, where the first click is one point, and dragging places the second point, opposite from the first.

Airbrush Flow of the brush continues as long as the mouse click is held, determined by the *Rate* setting. If disabled, the brush only modifies the model when the brush changes its location. This option is not available for the *Grab* brush.

The following parameters are available for the *Dots*, *Space*, and *Airbrush* strokes:

Jitter Jitters the position of the brush while painting.

Smooth stroke Brush lags behind mouse and follows a smoother path. When enabled, the following become active:

Radius Sets the minimum distance from the last point before stroke continues.

Factor Sets the amount of smoothing.

Curve Panel

The *Curve* section allows you to use a curve control to the right to modify the intensity of the brush from its center (left part of the curve) towards its borders (right part of the curve).

See also:

Read more about using the *Curve Widget*.

Symmetry Panel

Mirror Mirror the brush strokes across the selected local axes. Note that if you want to alter the directions the axes point in, you must rotate the model in *Edit Mode*, not *Object Mode*

Radial These settings allow for radial symmetry in the desired axes. The number determines how many times the stroke will be repeated within 360 degrees around the central axes.

Feather Reduces the strength of the stroke where it overlaps the planes of symmetry.

Lock These three buttons allow you to block any modification/deformation of your model along selected local axes, while you are sculpting it.

Tiling Using this option allows you to seamlessly tile your strokes along the given axes.

Tile Offset The default tile size is set to one BU (Blender Unit). The offset allows the option to alter the tile size along all three axes.

Options Tab

Overlay Panel

When enabled, the brush texture is shown in the viewport.

View The eye icon is used as a toggle to show or hide the given brush texture.

Alpha You can change the amount of transparency used when showing the texture using the Alpha slider.

Stroke Overlay The brush icon allows you to turn off the viewport overlay during strokes.

Options Panel

Gravity

Factor Setting the factor allows you to add gravity to your brush strokes, giving it a draping effect.

Orientation Using another object, the gravity can be oriented to the set object's local Z axis, changing the direction of the gravity.

Threaded Sculpt Takes advantage of multiple CPU processors to improve sculpting performance.

Fast Navigation For *Multires* models, show low resolution while navigation the viewport.

Use Deform Only Limits active modifiers on the active object to Deform modifiers, and Multiresolution.

Show Diffuse Color Allows the active object to show it's diffuse color when sculpting.

Unified Settings

Size Forces the brush size to be shared across brushes.

Strength Forces the brush strength to be shared across brushes.

Color Not Used in Sculpt Mode.

Show Brush Shows the brush shape in the viewport.

Color (Add/Subtract) Set the color of the brush ring when its particular effect is active.

Appearance Panel

Show Brush Shows the brush shape in the viewport.

Color (Add/Subtract) Set the color of the brush ring when its particular effect is active.

Custom Icon Append an image file to the active brush as an icon.

Adaptive Sculpting

Dynamic Topology

Dynamic topology (aka dyntopo) is a dynamic tessellation sculpting method, adds and removes details on the fly. Dyntopo is quick, just get a brush and start to sculpt. Dyntopo will add details base upon your brush size, detail type and strength.

Detail Type Dyntopo uses three different detail methods to create dynamic detail to an object. The methods available are Relative Detail (Default), Constant Detail, and Brush Detail.

Relative Detail This method uses a detail size based on the number of pixels, and in turn will create topology in that size. Zoom out big details, zoom in small fines details.

Constant Detail To keep detail uniform across the entire object, Constant Detail can be used. The Detail is based on the percentage of a single BU.

Brush Detail Giving more control over the topology, with this method you can create topology based on the brush size. You can increase and lower topology by simply resizing the brush itself. The detail size is based the size of the brush itself, where 100% will create topology the size of the brush ring itself.

Detail Size Each Detail Type's detail is set here. Depending on the Detail Type being used this property will rather show as a pixel count (px), or percentage.

Detail Refine Method When using Dynamic Topology, a certain method will be used to tell how topology is handled. Setting the option will determine which of the methods will be used when altering the topology.

Subdivide Just like the subdivide command, this method will only subdivide topology to match the detail given.

Collapse When topology is too dense, and is smaller than the detail given, edges will be collapse to fit the detail size appropriately.

Subdivide Collapse This method combines the two methods, subdividing edges smaller than the detail size, and collapsing topology.

Detail Flood Fill When using Constant Detail mode, this option is made available, allowing you to fill the entire object with a uniform detail, based on the detail size.

Direction Determines which direction the model will be symmetrized.

Dyntopo Symmetrize Uses direction orientation to symmetrize. Since Dyntopo adds details dynamical may happen that the model goes asymmetric, so this a good tool for that.

Multi-Resolution Modifier

The multires modifier is needed to sculpt. The modifier will subdivide the mesh. The more subdivision the more computing will be needed. With the Blender stack non-destructive data, multires sculpting will help when you have a clean topology base mesh.

When sculpting with multires we have the ability sculpt in different level of subdivision, this mean we can sculpt some details in subdivision level 1 and add more details in subdivision 2 and go back to subdivision 1 correct some mistakes. While this workflow is often used, multires modifier has some limitations. You may end up with some mesh distortions. As an advice, add as more details as possible before adding more subdivisions. Clay brush, SculptDraw work better with multi-resolution sculpting to sculpt secondary forms.

See also:

Read more about the *Multi Resolution Modifier*.

Hiding & Masking

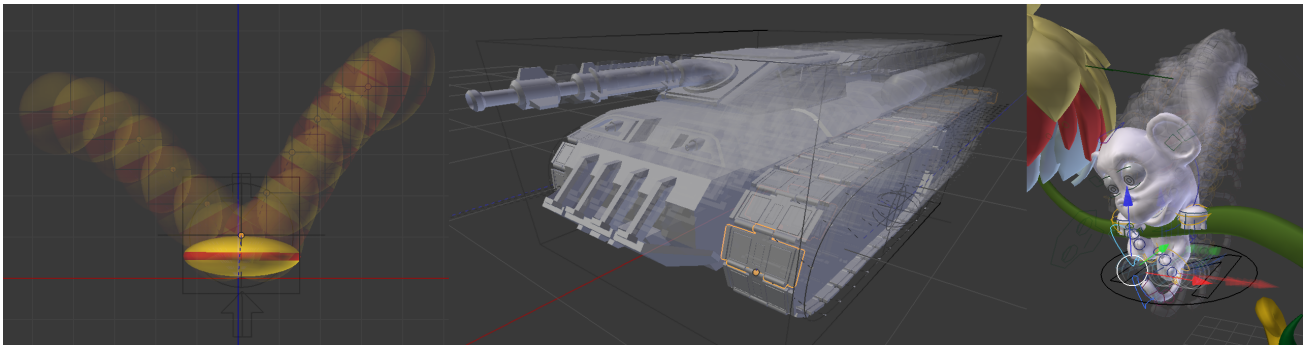
It is sometimes useful to isolate parts of a mesh to sculpt on. To hide a part of a mesh, press `H` then click & drag around the part you want to hide. To reveal a hidden part of a mesh, press `Shift-H` then click & drag around the part you want to reveal. To reveal all hidden parts, just press `Alt-H`. With the mask brush we can paint a part of the mesh and hide it.

Fig. 2.1008: Black part (hair) is masked.
The .blend file from OHA Studio © Mechanimotion Entertainment.

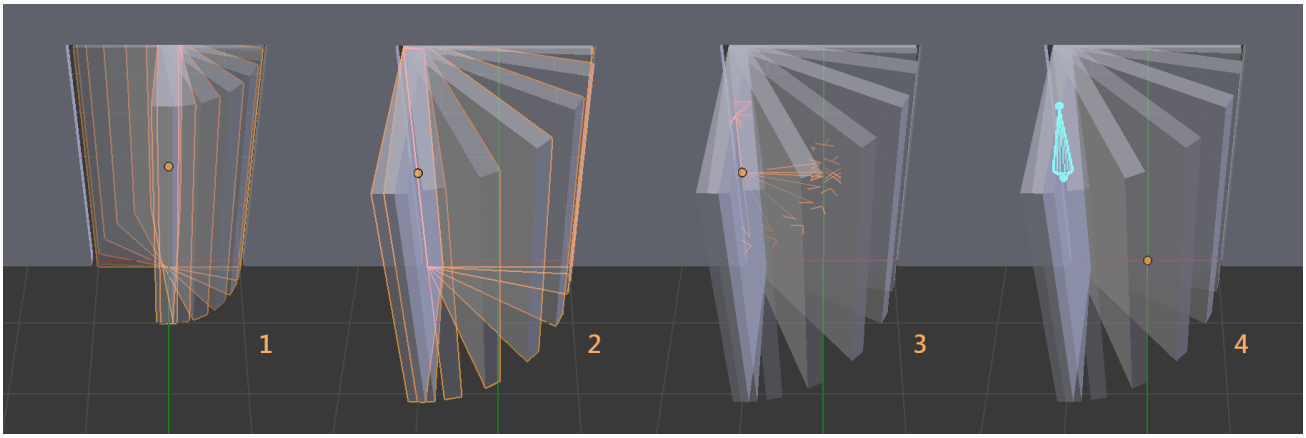
2.6 Rigging

2.6.1 Introduction

Rigging makes animation possible. Without a good rig animation is incredibly frustrating. Imagine animating a bouncing ball without the ability to squash it against the ground? Try animating a monkey swinging through the trees with no control to make the monkey's hands grab onto the branches. What if you had to animate an army tank speeding through the desert by positioning each tread on the tank one at a time?



At its most basic level, rigging solves motion problems. Imagine a door that opens into a hallway. Without a rig, the door will not swing open properly (1). A rig is needed to help the door swing open on its hinges (2, 3, 4), and there are many ways to rig the door. Door 2 gets rigged by repositioning the *Object Center* of the door. Door 3 gets rigged by *Parenting* the door to an *Empty*. Door 4 gets rigged by *Weight Painting* all of its *Vertices* to a *Bone* in an *Armature*.



Most production rigs are more complicated than a simple door, but be careful not to rush off building complicated rigs until you have developed some experience. Rigging is a discipline that takes practice. Start by building simple rigs (like a bouncing ball, a tumbling box, an odometer, a clock). Stay humble. Stay patient. Study the fundamental concepts that make a bouncing ball bounce. Add one rigging tool to your toolbox at a time. Test your simple rigs in actual animation projects. And only after much trial and error, consider putting everything together into the sophisticated character rig of your dreams.

See also:

The content of this chapter is simply a reference to how rigging is accomplished in Blender. It should be paired with additional resources such as Nathan Vegdahl's excellent (and free!) introduction to the fundamental concepts of character rigging, Humane Rigging.

2.6.2 Constraints

Introduction

Constraints control the behavior of one object with data from another. Constraints can make the eyes of a tennis player track a tennis ball bouncing across the court. Constraints allow the wheels on a bus to all rotate together. Constraints help a di-

nosaur's legs bend at the knee automatically. Constraints make it easy for a hand to grip the hilt of a sword and the sword to swing with the hand.

Constraints, in Blender, work with *Objects* and *Bones*.

Fig. 2.1009: *Object* Constraints.

Fig. 2.1010: *Bone* Constraints.

Constraints work in combination with each other to form a Constraint Stack.

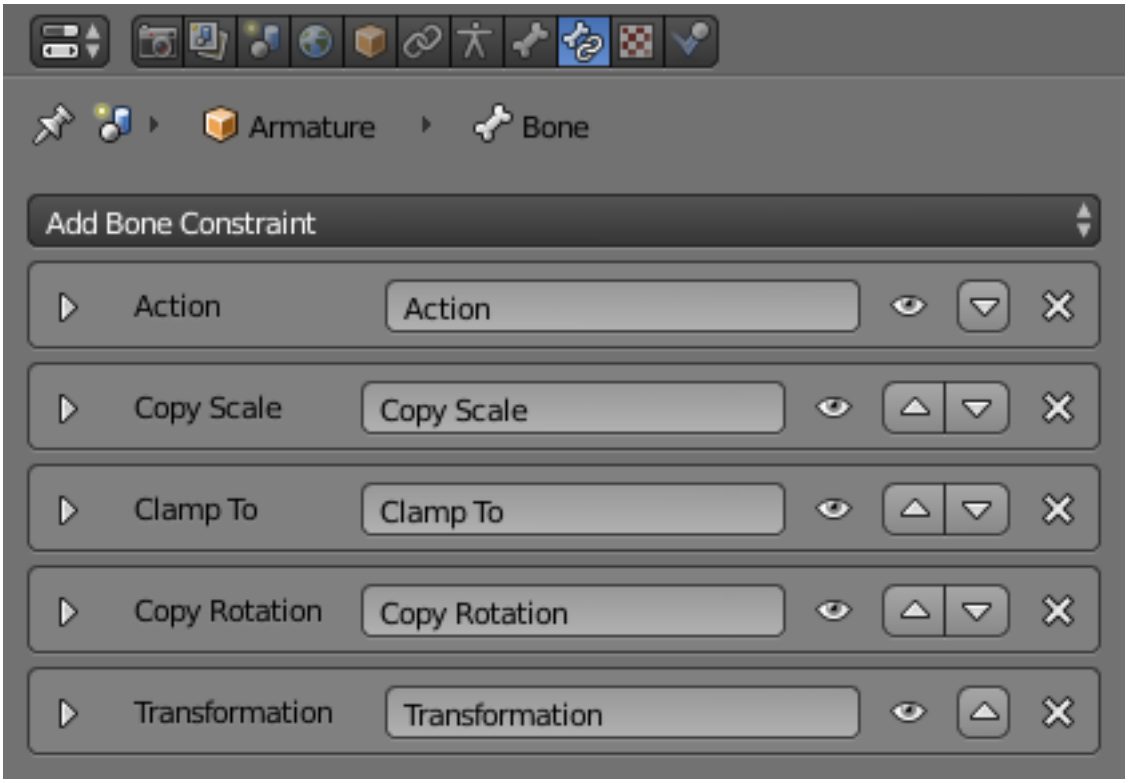
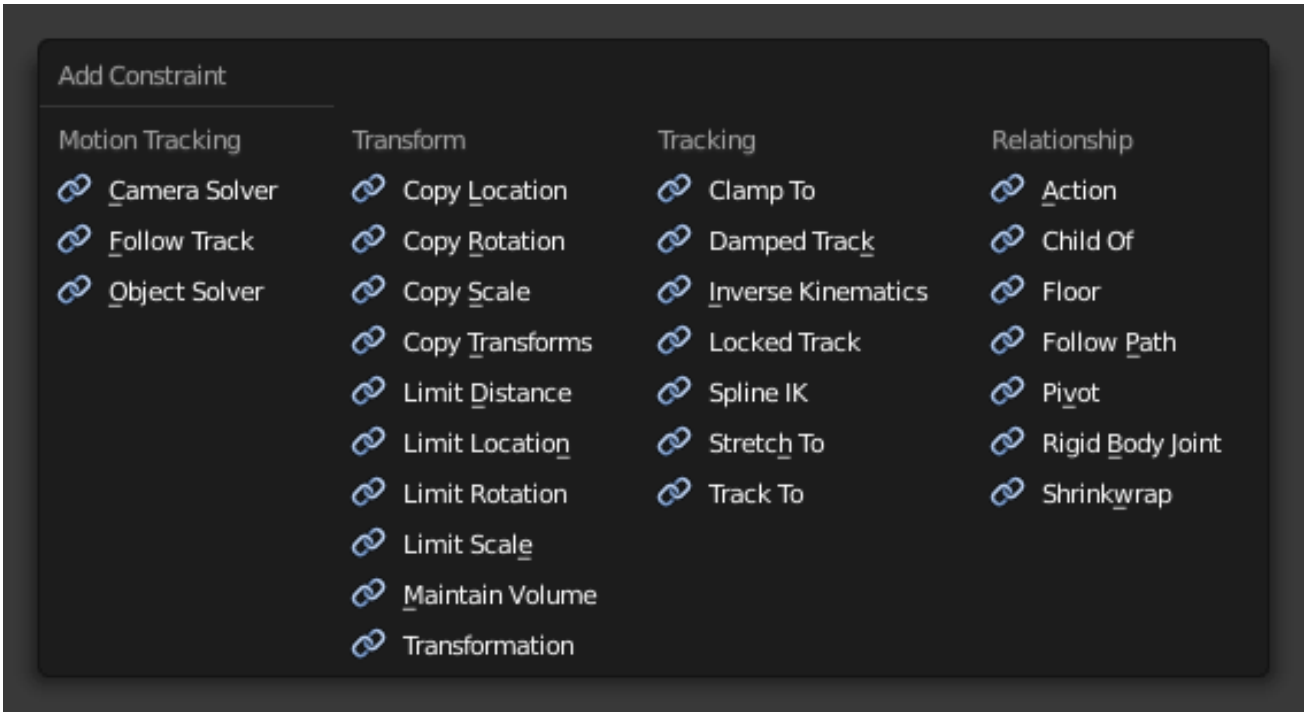


Fig. 2.1011: The Constraint Stack is evaluated from top to bottom.

Constraints are a fantastic way to add sophistication and complexity to a rig. But be careful not to rush in too quickly, piling up constraint upon constraint until you lose all sense of how they interact with each other.

Start simply. Get to know a single constraint inside and out. Copy Location is a good first constraint to explore. Take the time to understand every fundamental concept behind it, and the other constraints will make far more sense.



Interface

Adding/Removing a Constraint

To add a constraint in the *Constraints Panel*: Click on the "Add Constraint" menu.

To add a constraint in the 3D View: Select the object you would like to constrain. Press `Ctrl-Shift-C` and choose a constraint from the pop-up menu. If the chosen constraint needs a target, Blender will add an empty automatically as the target and position it at the center of the constrained object.

To add a constraint in the 3D View and simultaneously give it a target: Select the target first and then shift-select the object you would like to constrain. Press `Ctrl-Shift-C` and choose a constraint from the pop-up menu.

Note: When using a bone from another armature as the target for a constraint, `Ctrl-Shift-C` will look inside the non-active armature and use its active bone, provided that armature is in Pose Mode.

To remove a constraint: Click on the "X" button in the *header*.

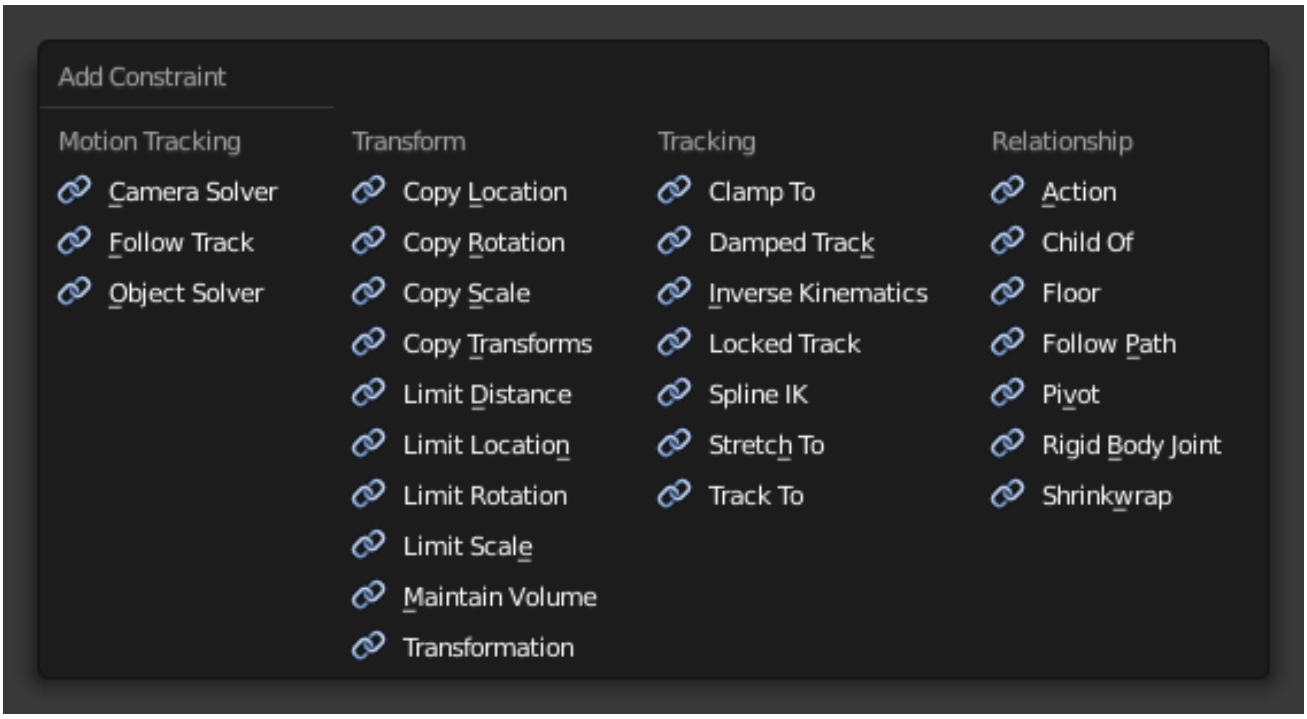
To remove all constraints from all selected object(s):

- Click *Object* → *Constraints* → *Clear Object Constraints* in the 3D View Header.
- or *Pose* → *Constraints* → *Clear Pose Constraints* (for bone constraints).
- or press `Ctrl-Alt-C`.

Header

Every constraint has a header. The interface elements of the header are explained below using a Copy Location constraint as an example.

Expansion Arrow (pointing down or right) Show or Hide the settings of the constraint. Tidy up the *constraint stack* by hiding constraints that do not currently need attention.



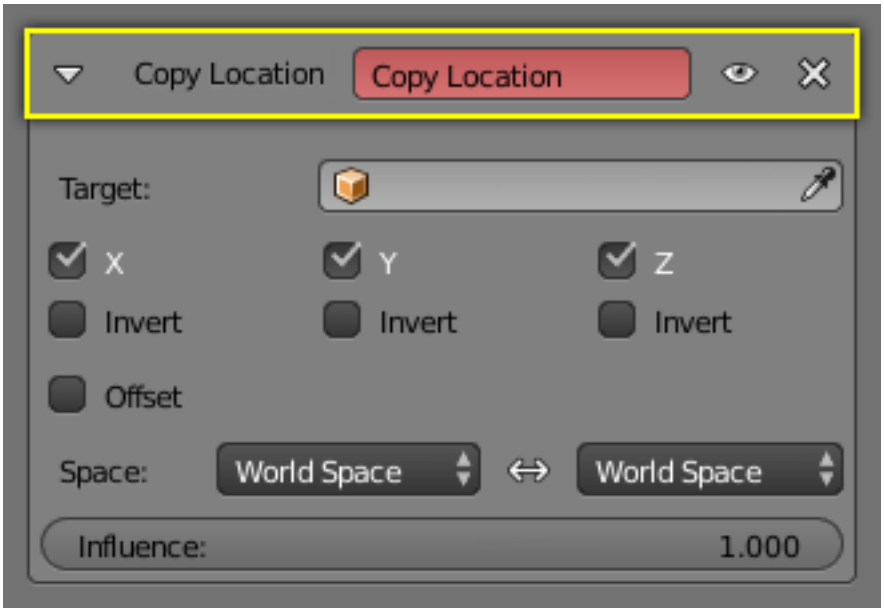


Fig. 2.1012: A Header sits at the top of every constraint.

Constraints will continue to affect the scene even when hidden.

"Copy Location" (first occurrence) The type of constraint. This is determined at the time the constraint is created.

"Copy Location" (second occurrence) Give the constraint a meaningful name in this field, something that describes its intent. Meaningful names help you and your team members understand what each constraint is supposed to do.

The *red* background is a warning that the constraint is not yet functional. The background will turn *gray* when the constraint is functioning. When this Copy Location constraint has a valid target in the "Target Field" it will turn gray and begin to function.

Eyeball (open or closed) Enable or Disable (Mute/Unmute) the constraint. Disabling a constraint will stop its affect on the scene.

Disabling a constraint is useful for turning off a constraint without losing all of its settings. Disabling means you can enable the constraint at a later time with the settings intact. Disabling is similar to setting the *influence* slider to 0.0.

Up/Down Arrows Move a constraint up or down in the *constraint stack*. Since the stack is evaluated from top to bottom, moving a constraint in the stack can significantly affect the final outcome of the stack.

If there is only one constraint in the stack, the arrows will not be drawn. If the constraint is at the top of the stack, only the down arrow will be drawn. If the constraint is at the bottom of the stack, only the up arrow will be drawn.

X Delete the constraint from the stack. The settings will be lost. The constraint will no longer affect the final outcome of the stack.

Target

The Target field lets you link the constraint to a Target object of your choosing. This link provides data to the constraint so that it can begin to function. For example, the Copy Location Constraint needs location data to function. Fill in the Target field, and the Copy Location constraint will begin to use location data from the Target object.

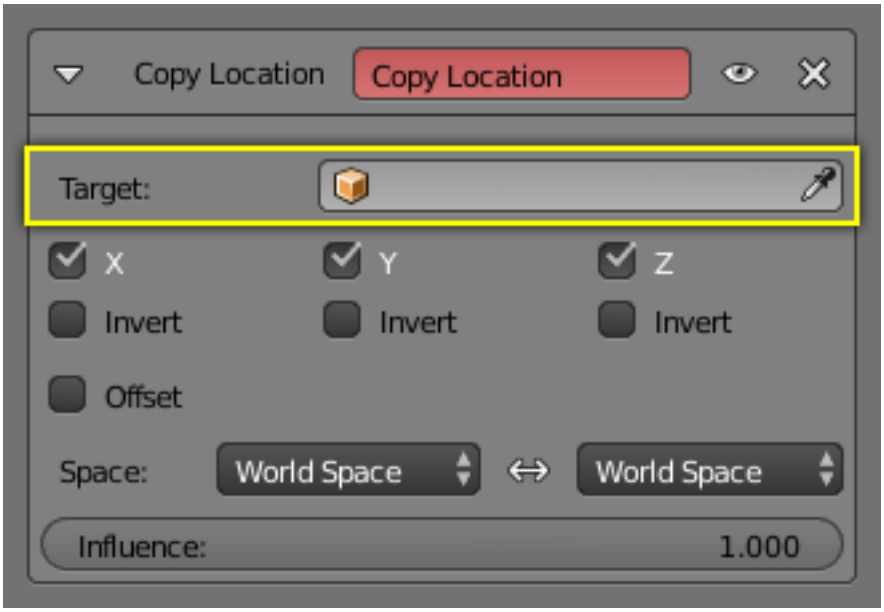
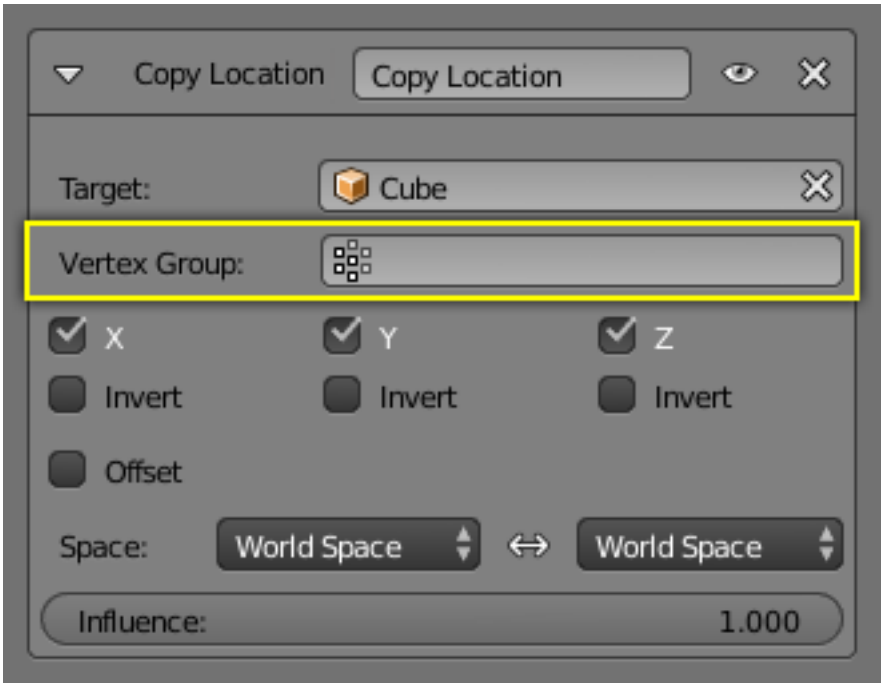


Fig. 2.1013: The Target field must be filled in for the constraint to function.

By default, the Target will use the *Object Center* as the target point.

If the Target field links to a *Mesh* or *Lattice* object, a *Vertex Group* field will appear. Enter the name of a vertex group and the constraint will target the median point of this vertex group instead of the object center.



If the Target field links to an *Armature*, a *Bone* field will appear along with a *Head* or *Tail* slider. Enter the name of a bone and the constraint will target the bone instead of the entire armature object center. Slide the slider and the constraint will target the head, the tail or somewhere inbetween.

Space

Constraints need a frame of reference in order to function. This frame of reference is called the "space" of the constraint. Choosing one space vs. another will change this frame of reference and substantially alter the behavior of a constraint.

To understand how changing the space will change the behavior of the constraint, consider experimenting with two empties. Make sure they display as arrows so that you can see the local axes for each empty. Make sure to size one empty a little larger than the other so that they are both always visible even if directly on top of each other. Then add a constraint to one empty that targets the other and experiment thoroughly by moving, rotating and scaling the target in many different ways.

Target Space & Owner Space

The space used to evaluate the target of the constraint is called the Target Space. The space used to evaluate the constrained object (the object that owns the constraint) is called the owner space. Hover over the space select menu(s) to learn whether it affects the space of the target or the space of the owner.

When the constraints use a Target and/or/nor a Owner space there will be no, one or two selector(s). The Copy Location constraint in example use both Target **and** Owner space.

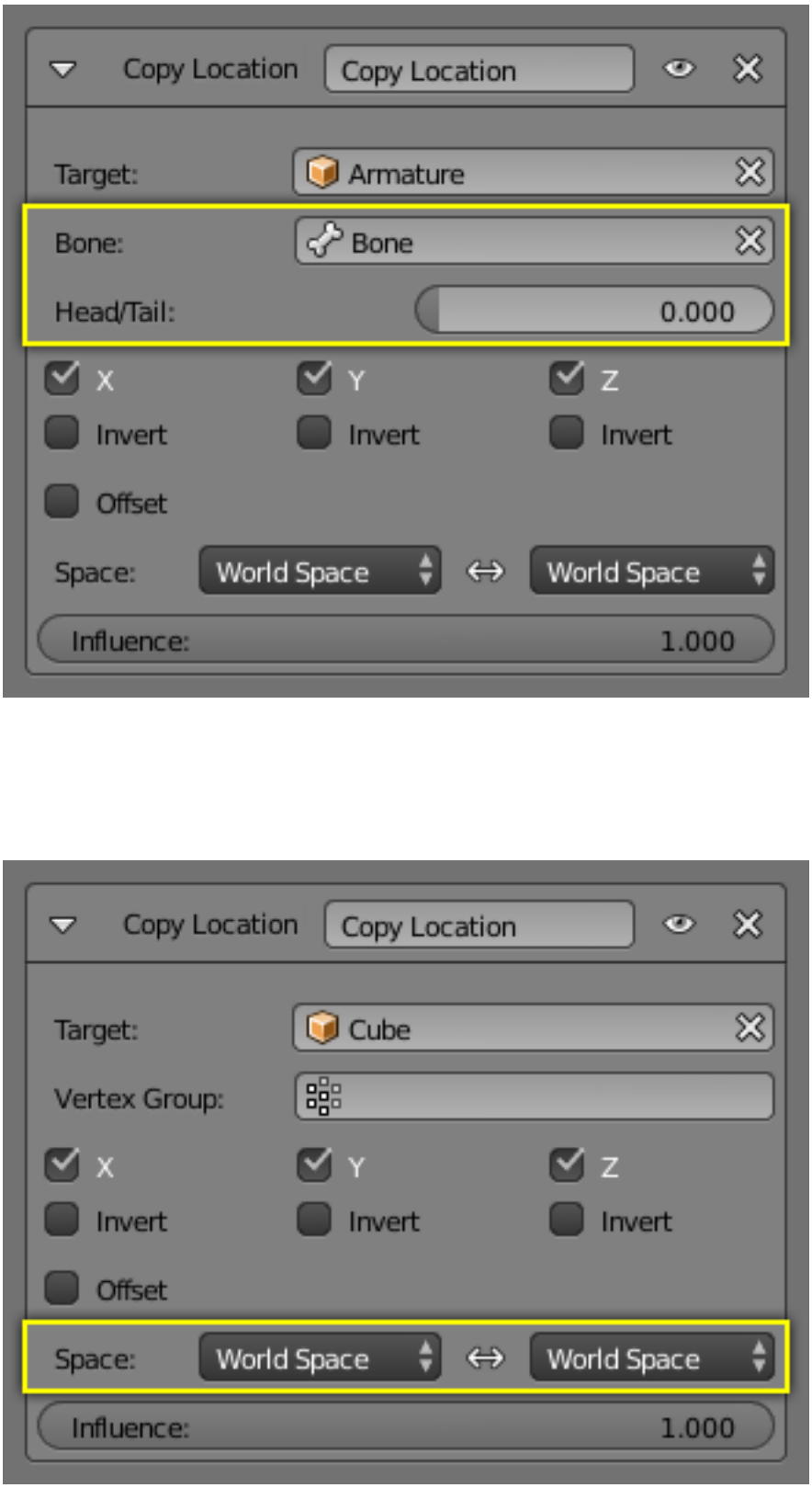


Fig. 2.1014: This constraint is set to use World Space as the frame of reference for both its Target Space and its Owner Space.

When a constraint uses both Target and Owner space, the Target and Owner can be any combination of space types.

Space Types

World Space In this space type the world is the frame of reference for the object (or bone). Location is relative to the world origin. Rotation and Scale are oriented to the world axes. Transformations to the object, the object's parent and any other constraints higher up in the constraint stack are all taken into account.

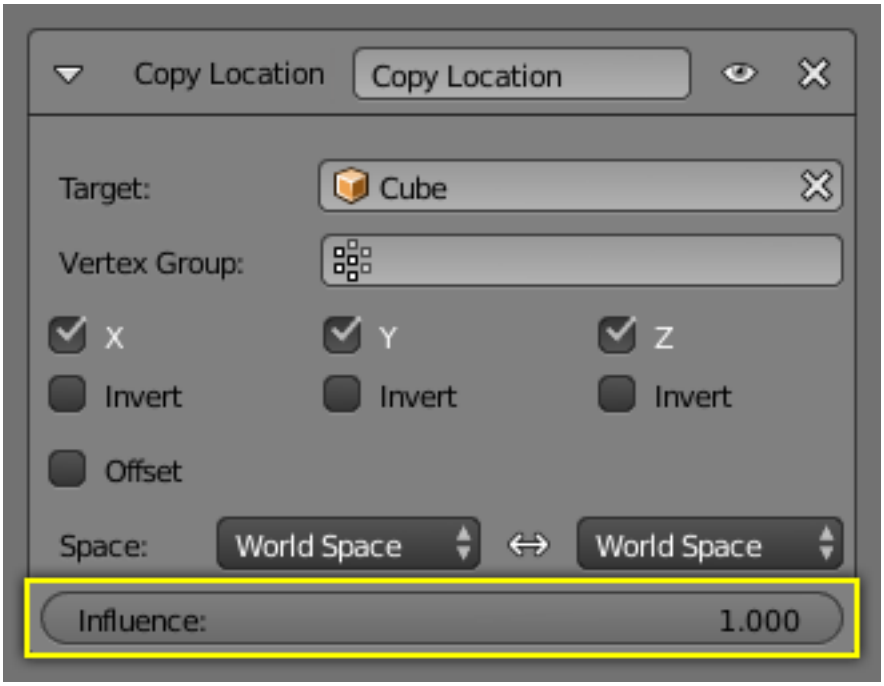
Local Space In this space type the parent of the object (or bone) is the frame of reference. Location is relative to the parent object origin. Rotation and Scale are oriented to the parent object axes. Only transformations to the object itself are taken into account. Transformations to the object's parent and any other constraints higher up in the constraint stack are **not** taken into account.

Local With Parent (bones only) The bone properties are evaluated in its own local space, *including* the transformations due to a possible parent relationship (i.e. due to the chain's transformations above the bone).

Pose Space (bones only) The bone properties are evaluated in the armature object local space (i.e. independently from the armature transformations in *Object Mode*). Hence, if the armature object has null transformations, *Pose Space* will have the same effect as *World Space*.

Influence

The influence slider determines how much the constraint will affect the constrained object.



An influence of 0.0 will have no effect. An influence of 1.0 will have the full effect.

Values between (0.0 and 1.0), will have a partial effect, but be careful. These partial effects can be difficult to control, especially as the *constraint stack* grows in complexity.

The influence value is animatable, allowing constraints to be turned off, or partially on as needed. (see *Using Constraints in Animation*)

The Constraints Stack

The combination of all the constraints affecting an object is called the Constraints Stack. The Stack is in the Constraints panel, below the "Add Constraint" menu.

Constraints in the stack are evaluated from top to bottom. The order of each constraint has a substantial impact on the final outcome of the stack. Changing the order of the constraints can change the behavior of the entire stack.

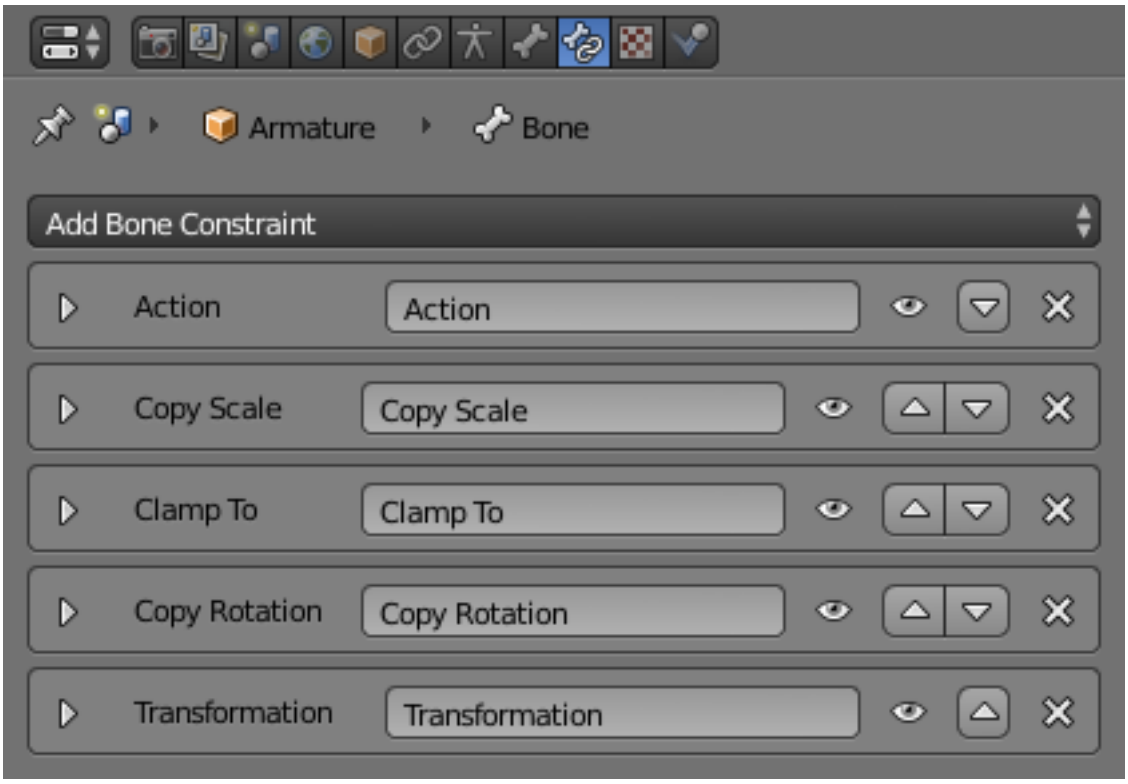


Fig. 2.1015: The seven constraints in this example stack are evaluated from top to bottom starting with the "Action" constraint and ending with the final "Transformation" constraint.

To change the order of a constraint use the up/down arrows in the *header*.

Motion Tracking

Camera Solver Constraint

The *Camera Solver* constraint gives the owner of this constraint, the location and rotation of the "solved camera motion".

The "solved camera motion" is where Blender thinks the physical, real world camera was, when it filmed the video footage, relative to the thing being tracked.

Note: This constraint only works after you have set up a minimum of eight markers and pressed *Solve Camera Motion*. (*Movie Clip Editor* → *Tool Shelf* → *Solve* → *Solve Camera Motion*)

Options

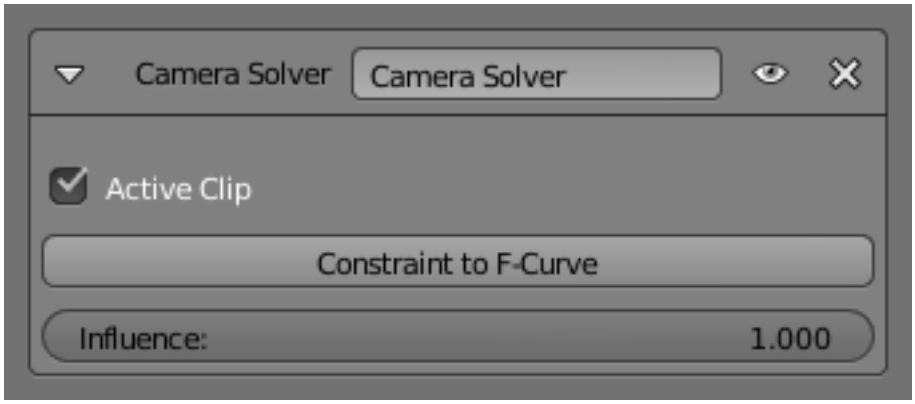


Fig. 2.1016: Camera Solver Constraint panel.

Active Clip Receive tracking data from the movie clip active in the movie clip editor. If unchecked, an option appears to choose from the other clips.

Constraint to F-Curve Applies the constraint, creating Keyframes for the transforms.

Influence Control how much this constraint effects the owner.

Object Solver Constraint

The *Object Solver* constraint gives the owner of this constraint, the location and rotation of the "solved object motion".

The "solved object motion" is where Blender thinks the physical, real world (tracked) object was, relative to the camera that filmed it.

Can be used to add a mesh to video for example.

Note: This constraint only works after you have set up a minimum of eight markers and pressed *Solve object Motion*. Located at *Movie Clip Editor* → *Tool Shelf* → *Solve* → *Solve Camera Motion*

If it says *Solve Camera Motion* instead of *Solve Object Motion* then go into the *Movie Clip Editor* → *Properties region* → *Objects* and switch it from the camera, to an object.

Options

Active Clip Receive tracking data from the active movie clip in the movie clip editor. If unchecked, an option appears to choose from the other clips.

Object Select a tracked object to receive transform data from.

Camera Select the camera to which the motion is parented to (if active empty scene camera is used)

Set Inverse Moves the origin of the object to the origin of the camera

Clear Inverse Moves the origin of the object back to the spot set in the Movie Clip Editor *Tool Shelf* → *Solve* → *Orientation* → *Set Origin*.

Constraint to F-Curve Applies the constraint, creating keyframes for the transforms.

Influence Control how much this constraint effects the owner.

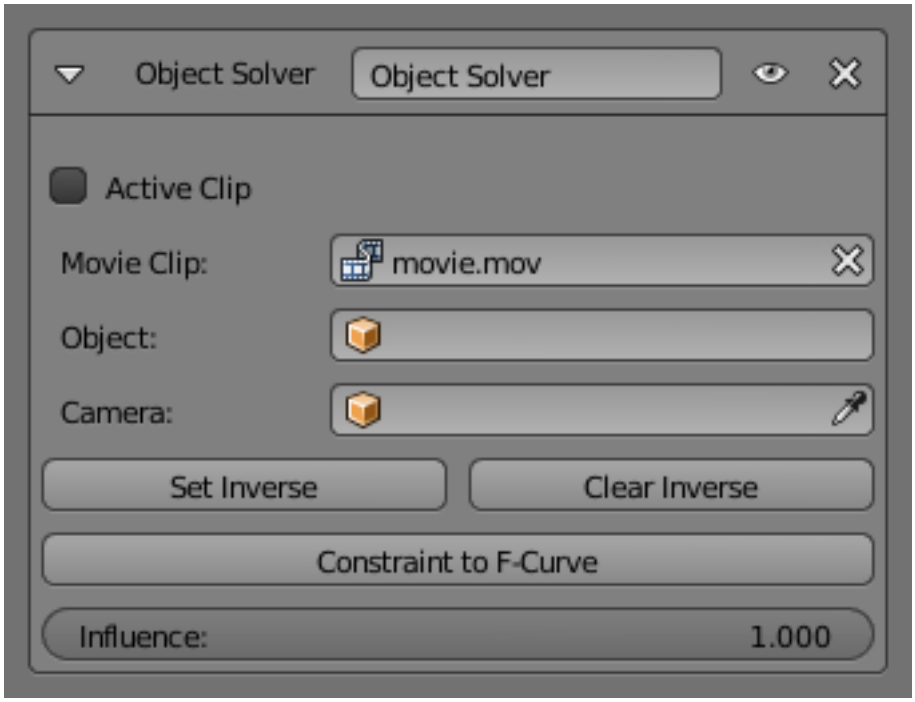


Fig. 2.1017: Object Solver Constraint panel.

Follow Track Constraint

TODO - see: https://developer.blender.org/T46926

Transform

Copy Location Constraint

The *Copy Location* constraint forces its owner to have the same location as its target.

> **Warning:** Note that if you use such a constraint on a *connected* bone, it will have no effect, as it is the parent's tip which controls the position of your owner bone's root.

Options

Target This constraint uses one target, and is not functional (red state) when it has none.

Bone If *Target* is an *Armature*, a new field is displayed offering the optional choice to set an individual bone as *Target*.

Head/Tail If a *Bone* is set as *Target*, a new field is displayed offering the optional choice of where along this bone the target point lies.

Vertex Group If *Target* is a *Mesh*, a new field is displayed offering the optional choice to set a *Vertex Group* as target.

X, Y, Z These buttons control which axes (i.e. coordinates) are constrained.

Invert The *Invert* buttons invert their respective preceding coordinates.

Offset When enabled, this control allows the owner to be translated (using its current transform properties), relative to its target's position.

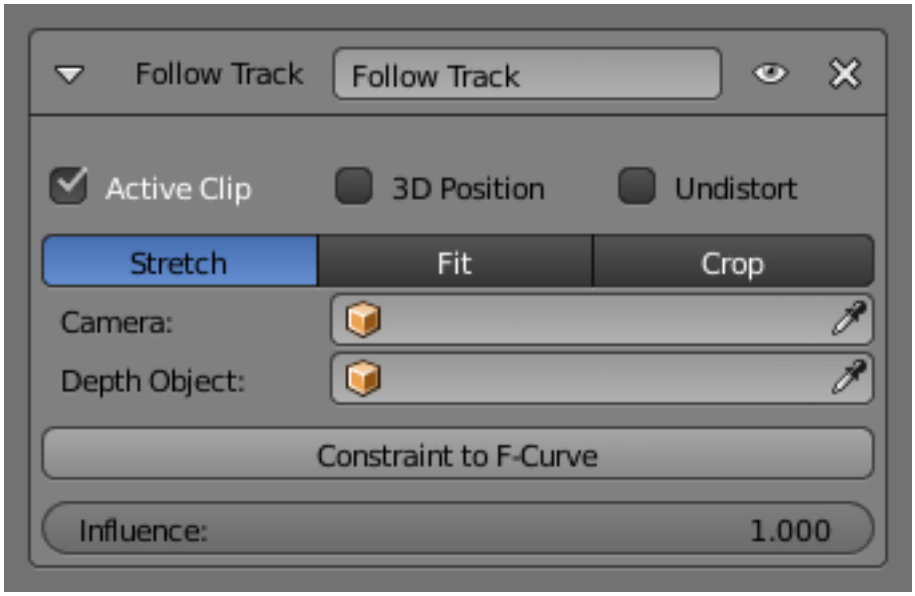


Fig. 2.1018: Follow Track Constraint panel.

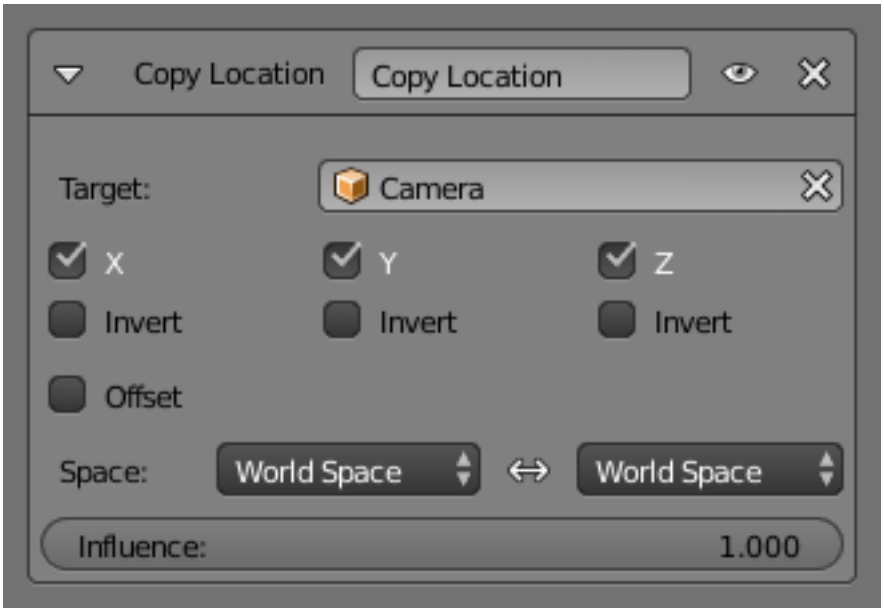


Fig. 2.1019: Copy Location panel.

Space This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

Copy Rotation Constraint

The *Copy Rotation* constraint forces its owner to match the rotation of its target.

Options

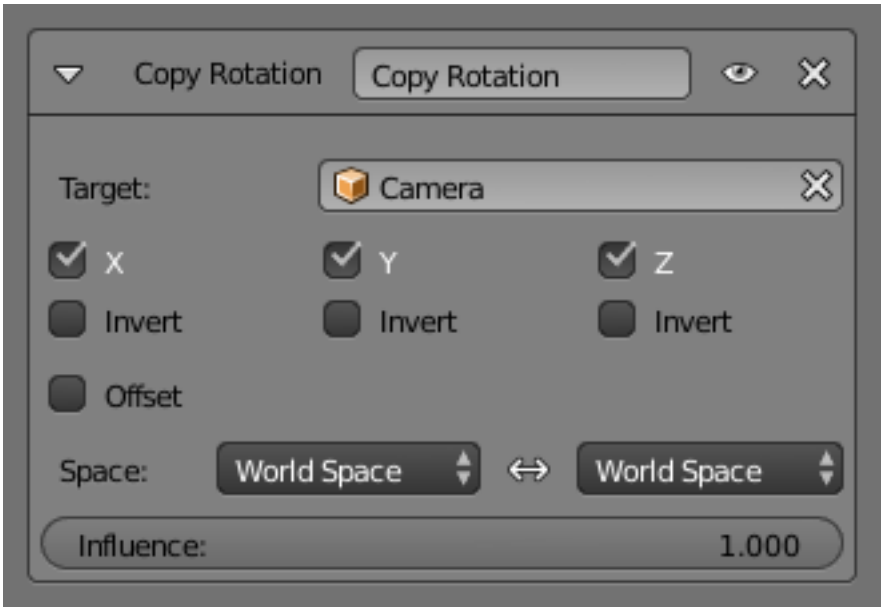


Fig. 2.1020: Copy Rotation panel.

Target This constraint uses one target, and is not functional (red state) when it has none.

Bone If *Target* is an *Armature*, a new field is displayed offering the optional choice to set an individual bone as *Target*.

Head/Tail If a *Bone* is set as *Target*, a new field is displayed offering the optional choice of where along this bone the target point lies.

Vertex Group If *Target* is a *Mesh*, a new field is displayed offering the optional choice to set a *Vertex Group* as target.

X, Y, Z These buttons control which axes are constrained.

Invert The *Invert* buttons invert their respective rotation values.

Offset When enabled, this control allows the owner to be rotated (using its current transform properties), relative to its target's orientation.

Space This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

Copy Scale Constraint

The *Copy Scale* constraint forces its owner to have the same scale as its target.

Warning: Here we talk of *scale*, not of *size*! Indeed, you can have two objects, one much bigger than the other, and yet both of them have the same scale. This is also true with bones: in *Pose Mode*, they all have a unitary scale when they are in rest position, represented by their visible length.

Options

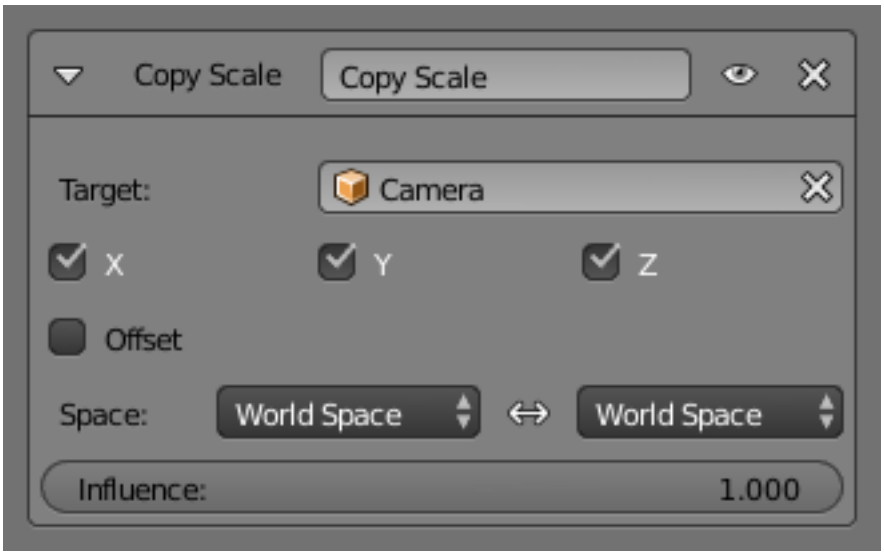


Fig. 2.1021: Copy Scale panel.

Target This constraint uses one target, and is not functional (red state) when it has none.

Bone If *Target* is an *Armature*, a new field is displayed offering the optional choice to set an individual bone as *Target*.

Head/Tail If a *Bone* is set as *Target*, a new field is displayed offering the optional choice of where along this bone the target point lies.

Vertex Group If *Target* is a *Mesh*, a new field is displayed offering the optional choice to set a *Vertex Group* as target.

X, Y, Z These buttons control along which axes the scale is constrained.

Offset When enabled, this control allows the owner to be scaled (using its current transform properties), relatively to its target's scale.

Space This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

Copy Transforms Constraint

The *Copy Transforms* constraint forces its owner to have the same transforms as its target.

Options

Target This constraint uses one target, and is not functional (red state) when it has none.

Bone If *Target* is an *Armature*, a new field is displayed offering the optional choice to set an individual bone as *Target*.

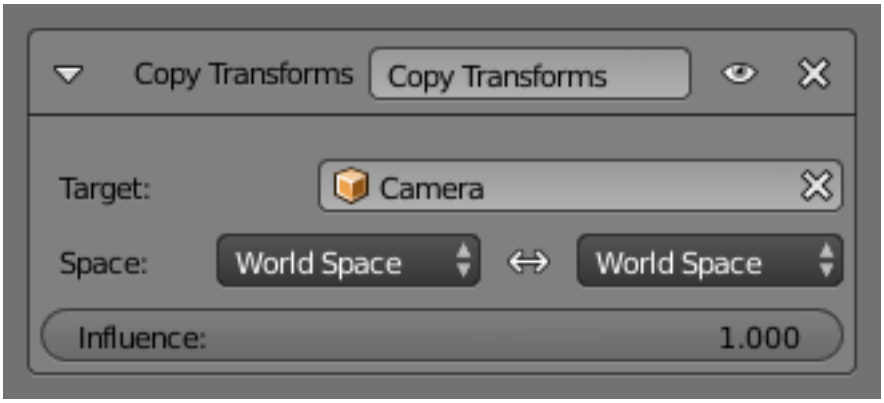


Fig. 2.1022: Copy Transforms panel.

Head/Tail If a *Bone* is set as *Target*, a new field is displayed offering the optional choice of where along this bone the target point lies.

Vertex Group If *Target* is a *Mesh*, a new field is displayed offering the optional choice to set a *Vertex Group* as target.

Space This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

Limit Distance Constraint

The *Limit Distance* constraint forces its owner to stay either further from, nearer to, or exactly at a given distance from its target. In other words, the owner's location is constrained either outside, inside, or at the surface of a sphere centered on its target.

When you specify a (new) target, the *Distance* value is automatically set to correspond to the distance between the owner and this target.

> **Warning:** Note that if you use such a constraint on a *connected* bone, it will have no effect, as it is the parent's tip which controls the position of your owner bone's root.

Options

Target This constraint uses one target, and is not functional (red state) when it has none.

Bone If *Target* is an *Armature*, a new field is displayed offering the optional choice to set an individual bone as *Target*.

Head/Tail If a *Bone* is set as *Target*, a new field is displayed offering the optional choice of where along this bone the target point lies.

Vertex Group If *Target* is a *Mesh*, a new field is displayed offering the optional choice to set a *Vertex Group* as target.

Distance This number button sets the limit distance, i.e. the radius of the constraining sphere.

Reset Distance When clicked, this small button will reset the *Distance* value, so that it corresponds to the actual distance between the owner and its target (i.e. the distance before this constraint is applied).

Clamp Region The *Limit Mode* select menu allows you to choose how to use the sphere defined by the *Distance* setting and target's center:

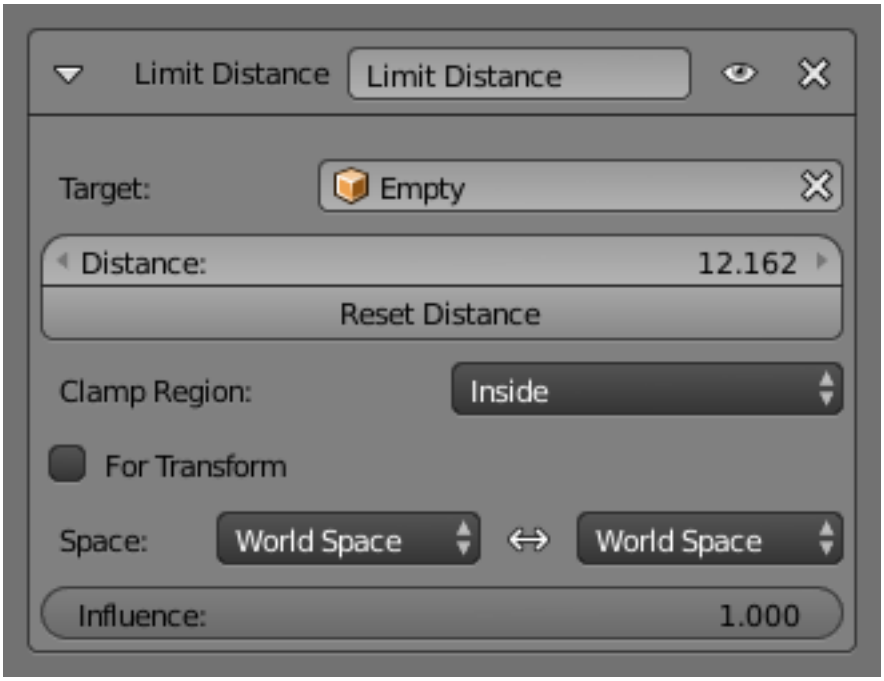


Fig. 2.1023: Limit Distance panel.

Inside The owner is constrained *inside* the sphere.

Outside The owner is constrained *outside* the sphere.

Surface The owner is constrained *on the surface* of the sphere.

For Transform ToDo.

Limit Location Constraint

An object or *unconnected* bone can be moved around the scene along the X, Y and Z axes. This constraint restricts the amount of allowed translations along each axis, through lower and upper bounds.

The limits for an object are calculated from its center, and the limits of a bone, from its root.

It is interesting to note that even though the constraint limits the visual and rendered location of its owner, its owner's data-block still allows (by default) the object or bone to have coordinates outside the minimum and maximum ranges. This can be seen in its *Transform* panel.

When an owner is grabbed and attempted to be moved outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its coordinates will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally specified location.

Similarly, if its owner has an internal location that is beyond the limits, dragging it back into the limit area will appear to do nothing until the internal coordinates are back within the limit threshold (unless you enabled the *For Transform* option, see below).

Setting equal the min and max values of an axis, locks the owner's movement along that axis... Although this is possible, using the *Transformation Properties* axis locking feature is probably easier!

Options

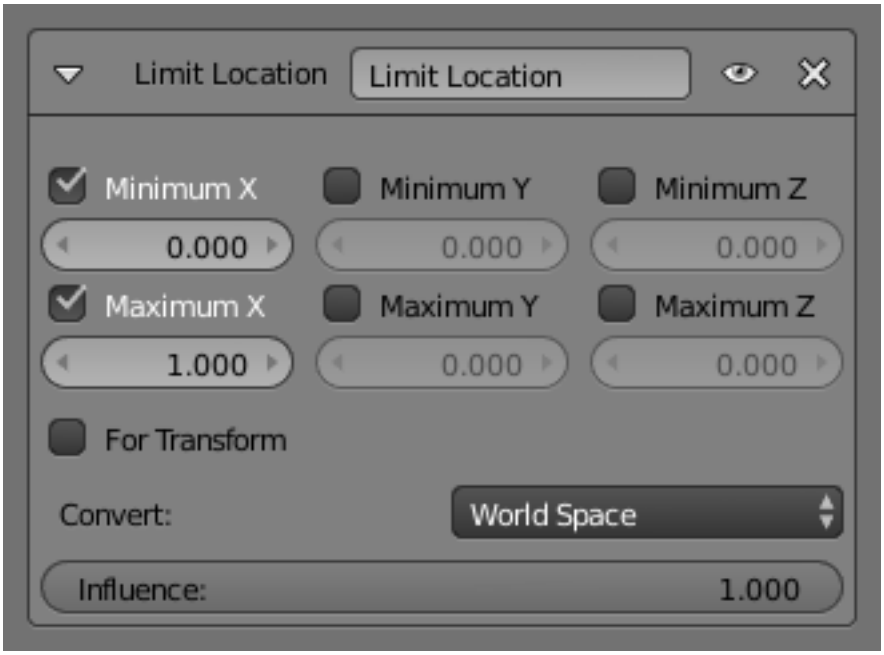


Fig. 2.1024: Limit Location panel.

Minimum X, Minimum Y, Minimum Z These buttons enable the lower boundary for the location of the owner's center along, respectively, the X, Y and Z axes of the chosen *Space*. The number button below them controls the value of their limit. Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

Maximum X, Maximum Y, Maximum Z These buttons enable the upper boundary for the location of the owner's center along, respectively, the X, Y and Z axes of the chosen *Space*. Same options as above.

For Transform We saw that by default, even though visually constrained, the owner can still have coordinates out of bounds (as shown by the *Transform* panel). Well, when you enable this button, this is no longer possible – the owner's transform properties are also limited by the constraint. Note however, that the constraint does not directly modify the coordinates: you have to grab its owner one way or another for this to take effect...

Convert This constraint allows you to choose in which space to evaluate its owner's transform properties.

Limit Rotation Constraint

An object or bone can be rotated around the X, Y and Z axes. This constraint restricts the amount of allowed rotations around each axis, through lower and upper bounds.

It is interesting to note that even though the constraint limits the visual and rendered rotations of its owner, its owner's data-block still allows (by default) the object or bone to have rotation values outside the minimum and maximum ranges. This can be seen in the *Transform* panel. When an owner is rotated and attempted to be rotated outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its rotation values will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally specified rotation.

Similarly, if its owner has an internal rotation that is beyond the limit, rotating it back into the limit area will appear to do nothing until the internal rotation values are back within the limit threshold (unless you enabled the *For Transform* option, see below).

Setting equal the min and max values of an axis, locks the owner's rotation around that axis... Although this is possible, using the *Transformation Properties* axis locking feature is probably easier.

This transform does not constrain the bone if it is manipulated by the IK solver. For constraining the rotation of a bone for IK purposes, see the "Inverse Kinematics" section of Bone properties.

Options

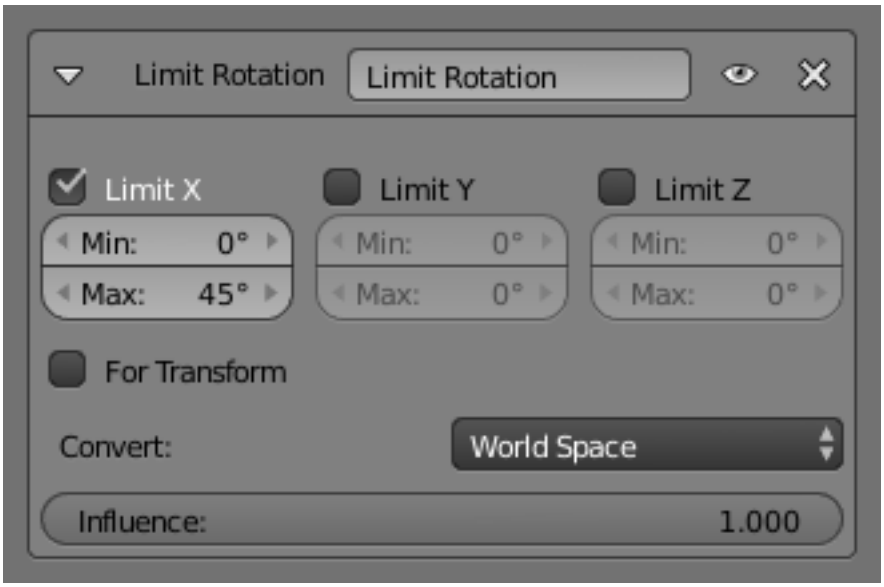


Fig. 2.1025: Limit Rotation panel.

Limit X, Y, Z These buttons enable the rotation limit around respectively the X, Y and Z axes of the owner, in the chosen *Space*. The *Min* and *Max* numeric fields to their right control the value of their lower and upper boundaries, respectively.

Note that:

- If a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.
- Unlike the *Limit Location constraint*, you cannot enable separately lower or upper limits...

For Transform We saw that by default, even though visually constrained, the owner can still have rotations out of bounds (as shown by the *Transform* panel). Well, when you enable this button, this is no more possible – the owner transform properties are also limited by the constraint. Note however, that the constraint does not directly modifies the rotation values: you have to rotate one way or the other its owner, for this to take effect...

Convert This constraint allows you to chose in which space evaluate its owner's transform properties.

Limit Scale Constraint

An object or bone can be scaled along the X, Y and Z axes. This constraint restricts the amount of allowed scalings along each axis, through lower and upper bounds.

Warning: This constraint does not tolerate negative scale values (those you might use to mirror an object...): when you add it to an object or bone, even if no axis limit is enabled, nor the *For Transform* button, as soon as you scale your object, all negative scale values are instantaneously inverted to positive ones... And the boundary settings can only take strictly positive values.

It is interesting to note that even though the constraint limits the visual and rendered scale of its owner, its owner's data-block still allows (by default) the object or bone to have scale values outside the minimum and maximum ranges (as long as they remain positive!). This can be seen in its *Transform* panel. When an owner is scaled and attempted to be moved outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its coordinates will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally-specified scale.

Similarly, if its owner has an internal scale that is beyond the limits, scaling it back into the limit area will appear to do nothing until the internal scale values are back within the limit threshold (unless you enabled the *For Transform* option, see below, or your owner has some negative scale values).

Setting equal the min and max values of an axis locks the owner's scaling along that axis. Although this is possible, using the *Transformation Properties* axis locking feature is probably easier.

Options

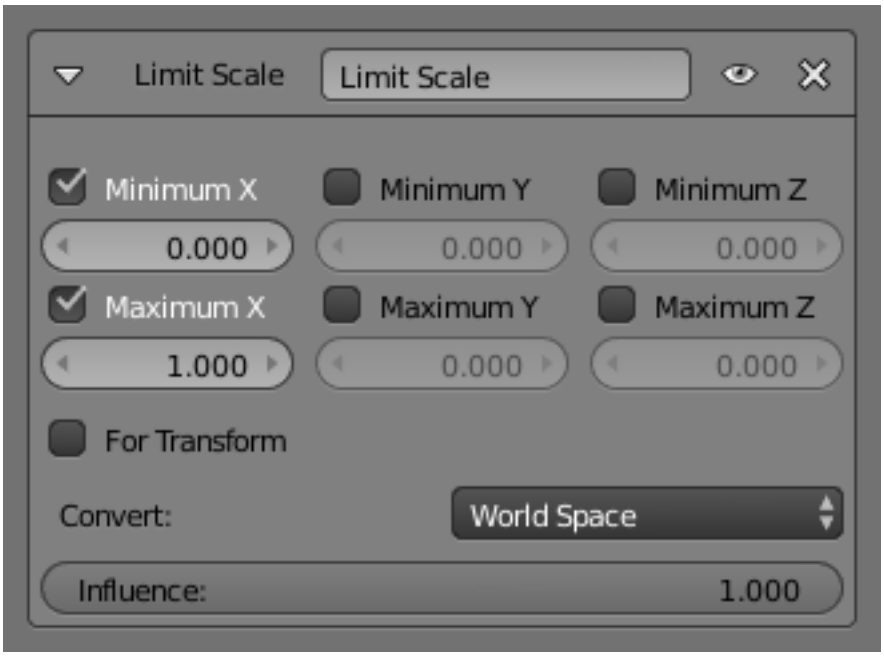


Fig. 2.1026: Limit Scale panel.

Minimum/Maximum X, Y, Z These buttons enable the lower boundary for the scale of the owner along respectively the X, Y and Z axes of the chosen *Space*. The *Min* and

Max numeric fields to their right control the value of their lower and upper boundaries, respectively.

Note: If a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

For Transform We saw that by default, even though visually constrained, and except for the negative values, the owner can still have scales out of bounds (as shown by the *Transform* panel). Well, when you enable this button, this is no longer possible, the owner transform properties are also limited by the constraint. Note however, that the constraint does not directly modify the scale values: you have to scale its owner one way or another for this to take effect.

Convert This constraint allows you to choose in which space to evaluate its owner's transform properties.

Maintain Volume Constraint

The *Maintain Volume* constraint limits the volume of a mesh or a bone to a given ratio of its original volume.

Options

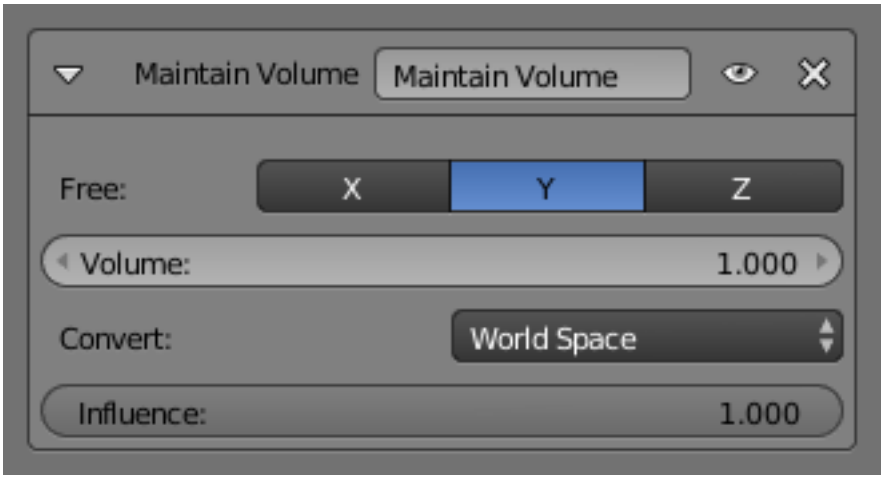


Fig. 2.1027: Maintain Volume Constraint.

Free X, Y, Z The free-scaling axis of the object.

Volume The bone's rest volume. Default is 1.0 .

Space This constraint allows you to choose in which space to evaluate its owner's transform properties.

See also:

Harkyman on the development of the Maintain Volume constraint.

Transformation Constraint

This constraint is more complex and versatile than the other "transform" constraints. It allows you to map one type of transform properties (i.e. location, rotation or scale)

of the target, to the same or another type of transform properties of the owner, within a given range of values (which might be different for each target and owner property). You can also switch between axes, and use the range values not as limits, but rather as "markers" to define a mapping between input (target) and output (owner) values.

So, e.g. you can use the position of the target along the X axis to control the rotation of the owner around the Z-Axis, stating that 1 BU along the target X-Axis corresponds to10 BU around the owner Z-Axis. Typical uses for this include gears (see note below), and rotation based on location setups.

Options

Target This constraint uses one target, and is not functional (red state) when it has none.

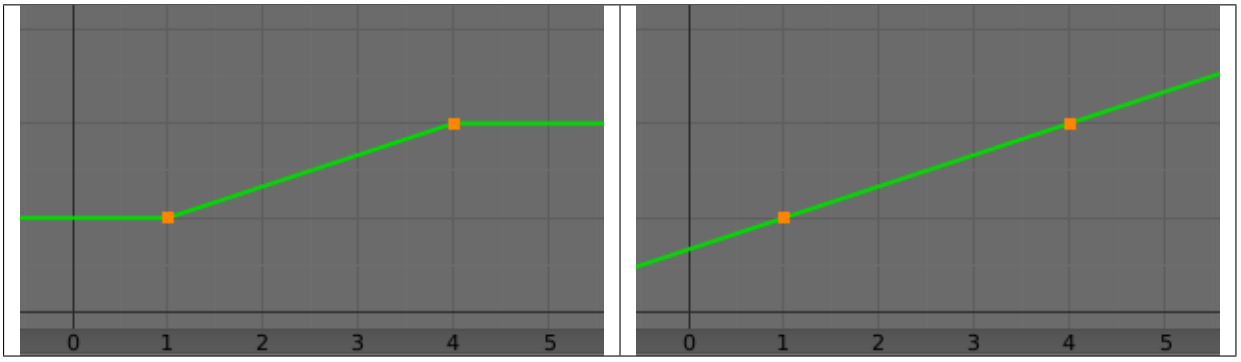
Bone If *Target* is an *Armature*, a new field is displayed offering the optional choice to set an individual bone as *Target*.

Head/Tail If a *Bone* is set as *Target*, a new field is displayed offering the optional choice of where along this bone the target point lies.

Vertex Group If *Target* is a *Mesh*, a new field is displayed offering the optional choice to set a *Vertex Group* as target.

Extrapolate By default, the *min* and *max* values bound the input and output values; all values outside these ranges are clipped to them. When you enable this button, the *min* and *max* values are no longer strict limits, but rather "markers" defining a proportional (linear) mapping between input and corresponding output values. Let us illustrate that with two graphs Fig. *The Extrapolate principles.*. In these pictures, the input range (in abscissa) is set to (1.0 to 4.0), and its corresponding output range (in ordinate), to (1.0 to 2.0). The yellow curve represents the mapping between input and output.

Table 2.40: Extrapolate enabled: the output values are "free" to proportionally follow the input ones.



Source

It contains the input (from target) settings.

Map From The radio buttons, allow you to select which type of property to use.

Location, Rotation, and Scale

From Independently for each axis (X, Y, and Z) the min and max number buttons control the lower and upper bounds of the input value range. Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

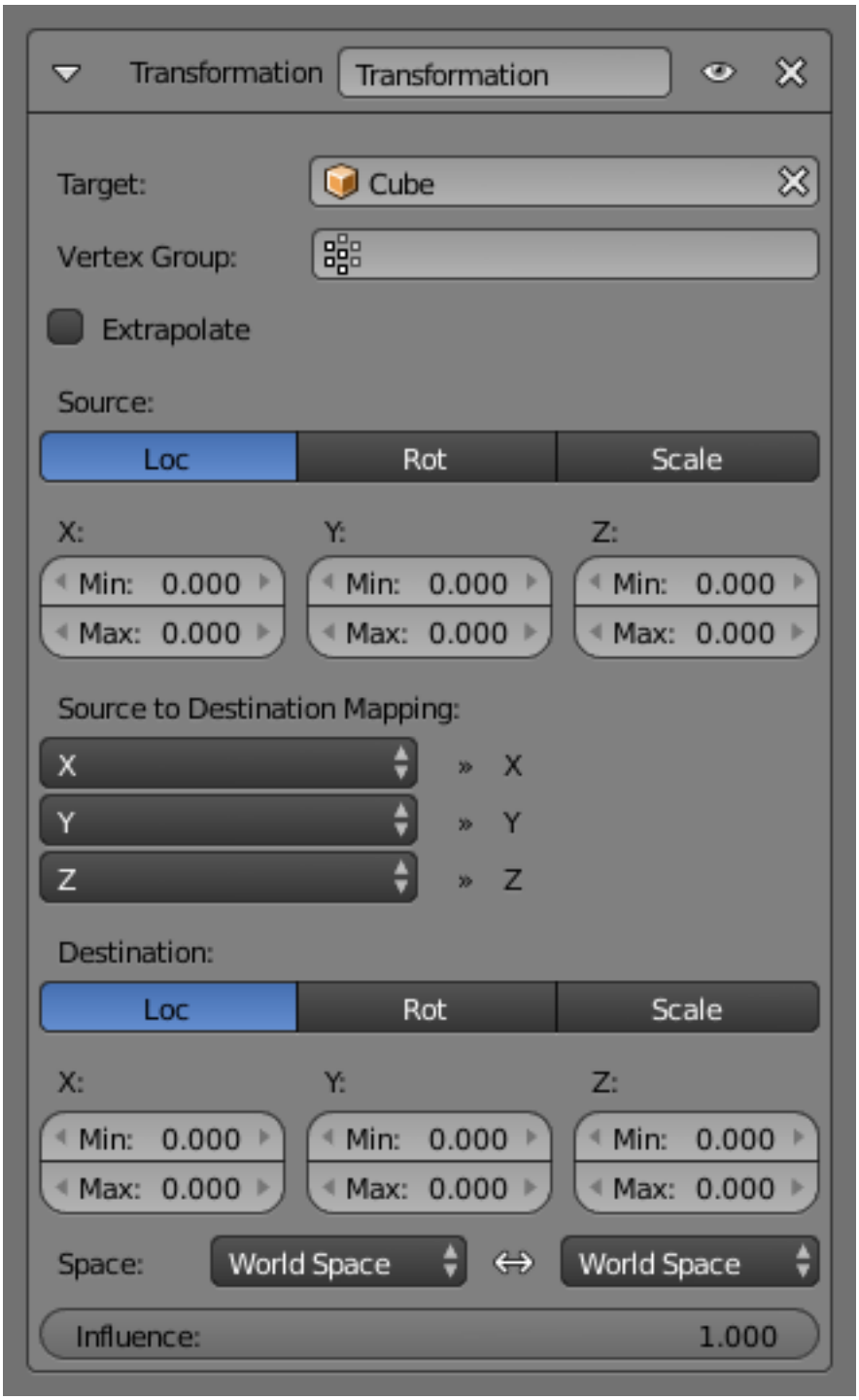


Fig. 2.1028: Transformation panel.

Source to Destination Mapping The three *Axis Mapping* selectors allow you to select which input axis to map to, respectively (from top to bottom), the X, Y and Z output (owner) axes.

Destination

It contains the output (to owner) settings.

Map To The three radio buttons allow you to select which type of property to control.

Location, Rotation, and Scale

To The *min* and *max* number buttons control the lower and upper bounds of the output value range, independently for each mapped axis. Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

Space This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

Note:

- When mapping transform properties to location (i.e. *Location*, *Destination* button is enabled), the owner's existing location is added to the result of evaluating this constraint (exactly like when the *Offset* button of the *Copy Location constraint* is enabled...).
- Conversely, when mapping transform properties to rotation or scale, the owner's existing rotation or scale is overridden by the result of evaluating this constraint.
- When using the rotation transform properties of the target as input, whatever the real values are, the constraint will always "take them back" into the (-180 to 180) range (e.g. if the target has a rotation of 420 degrees around its X-Axis, the values used as *X* input by the constraint will be:

  $((420 + 180) modulo 360) - 180 = 60 - ...$

  This is why this constraint is not really suited for gears!
- Similarly, when using the scale transform properties of the target as input, whatever the real values are, the constraint will always take their absolute values (i.e. invert negative ones).
- When a *min* value is higher than its corresponding *max* one, both are considered equal to the *max* one. This implies you cannot create "reversed" mappings...

Transform Cache Constraint

The *Transform Cache Constraint* is used to stream animations made at the transformation matrix level (for example rigid bodies, or camera movements).

Options

Cache File Data-block menu to select the Alembic file.

File Path Path to Alembic file.

Is Sequence Whether or not the cache is separated in a series of files.

Override Frame Whether to use a cuctom frame for looking up data in the cache file, instead of using the current scene frame.

Frame The time to use for looking up the data in the cache file, or to determine which to use in a file sequence.

Manual Transform Scale Value by which to enlarge or shrink the object with respect to the world's origin.

Object Path The path to the Alembic object inside the archive.

Verts/Faces/UV/Color Type of data to read for a mesh object respectively: vertices, polygons, UV layers and Vertex Color layers.

Tracking

Clamp To Constraint

The *Clamp To* constraint clamps an object to a curve. The *Clamp To* constraint is very similar to the *Follow Path* constraint, but instead of using the evaluation time of the target curve, *Clamp To* will get the actual location properties of its owner (those shown in the *Transform* panel), and judge where to put it by "mapping" this location along the target curve.

One benefit is that when you are working with *Clamp To*, it is easier to see what your owner will be doing; since you are working in the 3D View, it will just be a lot more precise than sliding keys around on a F-Curve and playing the animation over and over.

A downside is that unlike in the *Follow Path constraint*, *Clamp To* does not have any option to track your owner's rotation (pitch, roll, yaw) to the banking of the targeted curve, but you do not always need rotation on, so in cases like this it's usually a lot handier to fire up a *Clamp To*, and get the bits of rotation you do need some other way.

The mapping from the object's original position to its position on the curve is not perfect, but uses the following simplified algorithm:

- A "main axis" is chosen, either by the user, or as the longest axis of the curve's bounding box (the default).
- The position of the object is compared to the bounding box of the curve in the direction of the main axis. So for example if X is the main axis, and the object is aligned with the curve bounding box's left side, the result is 0; if it is aligned with the right side, the result is 1.
- If the cyclic option is unchecked, this value is clamped in the range 0-1.
- This number is used as the curve time, to find the final position along the curve that the object is clamped to.

This algorithm does not produce exactly the desired result because curve time does not map exactly to the main axis position. For example an object directly in the center of a curve will be clamped to a curve time of 0.5 regardless of the shape of the curve, because it is halfway along the curve's bounding box. However, the 0.5 curve time position can actually be anywhere within the bounding box!

Options

Target The Target: field indicates which curve object the Clamp To constraint will track along. The Target: field must be a curve object type. If this field is not filled in then it will be highlighted in red indicating that this constraint does not have all the information it needs to carry out its task and will therefore be ignored on the constraint stack.

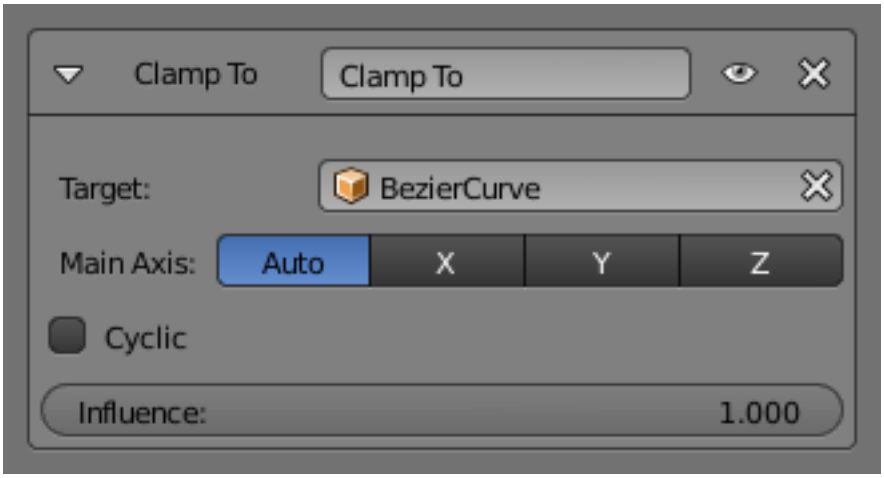


Fig. 2.1031: Clamp To panel.

Main Axis This button group controls which global axis (X, Y or Z) is the main direction of the path. When clamping the object to the target curve, it will not be moved significantly on this axis. It may move a small amount on that axis because of the inexact way this constraint functions.

For example if you are animating a rocket launch, it will be the Z axis because the main direction of the launch path is up. The default *Auto* option chooses the axis which the curve is longest in (or X if they are equal). This is usually the best option.

Cyclic By default, once the object has reached one end of its target curve, it will be constrained there. When the *Cyclic* option is enabled, as soon as it reaches one end of the curve, it is instantaneously moved to its other end. This is of course primarily designed for closed curves (circles & co), as this allows your owner to go around it over and over.

Damped Track Constraint

The *Damped Track* constraint constrains one local axis of the owner to always point towards *Target*. In other 3D software you can find it with the name "Look at" constraint.

Options

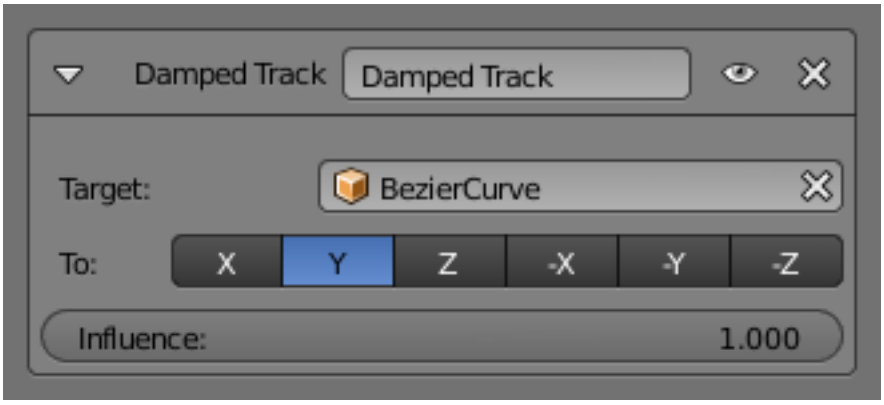


Fig. 2.1032: Damped Track panel.

Target (Mesh Object Type) This constraint uses one target, and is not functional (red state) when it has none.

Vertex Group If *Target* is a *Mesh*, a new field is displayed offering the optional choice to set a *Vertex Group* as target.

Target (Armature Object Type)

Bone If *Target* is an *Armature*, a new field is displayed offering the optional choice to set an individual bone as *Target*.

Head/Tail If *Target* is an *Armature*, a new field is displayed offering the optional choice to set whether the Head or Tail of a Bone will be pointed at by the *Target*. It is a slider value field which can have a value between 0 and 1. A value of 0 will point the Target at the Head/Root of a Bone while a value of 1 will point the Target at the Tail/Tip of a Bone.

To Once the owner object has had a Damped Track constraint applied to it, you must then choose which axis of the object you want to point at the Target object. You can choose between 6 axis directions (-X, -Y, -Z, X, Y, Z). The negative axis direction cause the object to point away from the Target object along the selected axis direction.

IK Solver Constraint

The *Inverse Kinematics* constraint implements the *inverse kinematics* armature posing technique. Hence, it is only available for bones. To quickly create an IK constraint with a target, select a bone in pose mode, and press `Shift-I`.

This constraint is fully documented in the *Inverse Kinematics* page, part of the rigging chapter.

Options

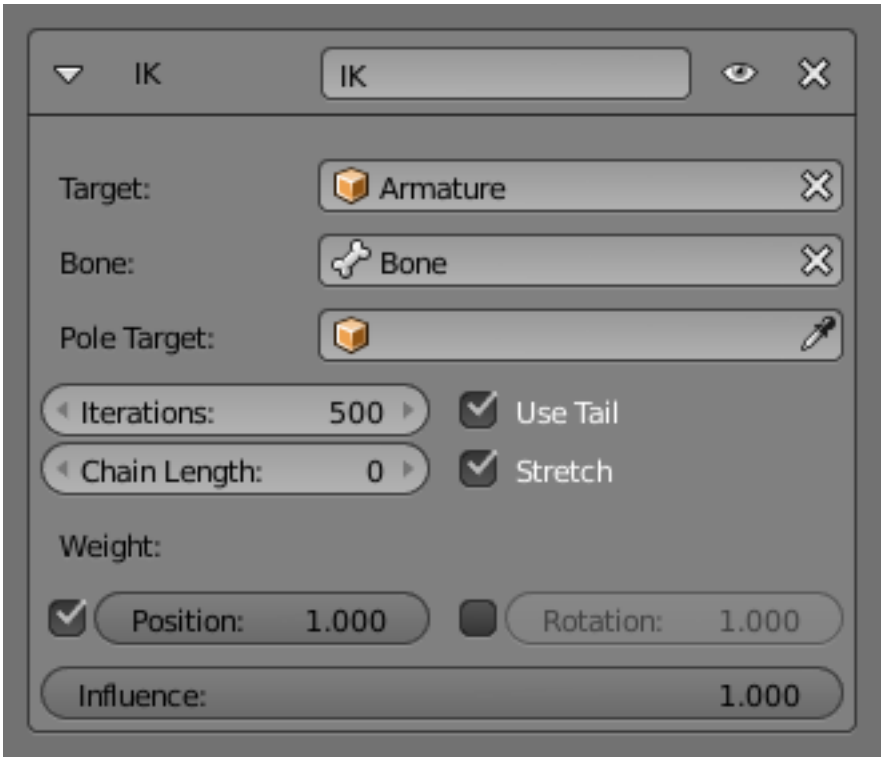


Fig. 2.1033: Inverse Kinematics panel.

Target Must be an armature.

Bone A bone in the armature.

Pole Target Object for pole rotation.

Iterations Maximum number of solving iterations.

Chain Length How many bones are included in the IK effect. Set to 0 to include all bones.

Use Tail Include bone's tail as last element in chain.

Stretch Enable IK stretching.

Weight

Position For Tree-IK: Weight of position control for this target.

Rotation Chain follow rotation of target.

Target Disable for targetless IK.

Rotation Chain follows rotation of target.

Locked Track Constraint

The *Locked Track* constraint is a bit tricky to explain, both graphically and textually. Basically, it is a *Track To constraint*, but with a locked axis, i.e. an axis that cannot rotate (change its orientation). Hence, the owner can only track its target by rotating around this axis, and unless the target is in the plane perpendicular to the locked axis, and crossing the owner, this owner cannot really point at its target.

Let us take the best real world equivalent: a compass. It can rotate to point in the general direction of its target (the magnetic North, or a neighbor magnet), but it cannot point *directly at it*, because it spins like a wheel on an axle. If a compass is sitting on a table and there is a magnet directly above it, the compass cannot point to it. If we move the magnet more to one side of the compass, it still cannot point *at* the target, but it can point in the general direction of the target, and still obey its restrictions of the axle.

When using a *Locked Track* constraint, you can think of the target as a magnet, and the owner as a compass. The *Lock* axis will function as the axle around which the owner spins, and the *To* axis will function as the compass' needle. Which axis does what is up to you!

If you have trouble understanding the buttons of this constraint, read the tool-tips, they are pretty good. If you do not know where your object's axes are, turn on the *Axis* button in the *Object* menu's *Draw* panel. Or, if you are working with bones, turn on the *Axes* button in the *Armature* menu's *Display* panel.

This constraint was designed to work cooperatively with the *Track To* constraint. If you set the axes buttons right for these two constraints, *Track To* can be used to point the axle at a primary target, and *Locked Track* can spin the owner around that axle to a secondary target.

This constraints also works very well for 2D billboarding.

Options

Target This constraint uses one target, and is not functional (red state) when it has none.

To The tracking local axis, i.e. the owner's axis to point at the target. The negative options force the relevant axis to point away from the target.

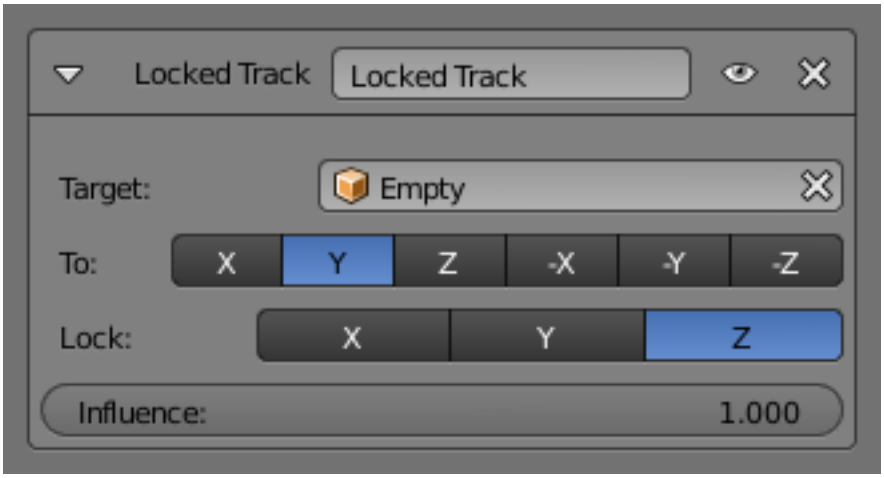


Fig. 2.1034: Locked track panel.

Lock The locked local axis, i.e. the owner's axis which cannot be re-oriented to track the target.

> **Warning:** If you choose the same axis for *To* and *Lock*, the constraint will no longer be functional (red state).

Spline IK Constraint

The *Spline IK* constraint aligns a chain of bones along a curve. By leveraging the ease and flexibility of achieving aesthetically pleasing shapes offered by curves and the predictability and well-integrated control offered by bones, *Spline IK* is an invaluable tool in the riggers' toolbox. It is particularly well suited for rigging flexible body parts such as tails, tentacles, and spines, as well as inorganic items such as ropes.

To set up *Spline IK*, it is necessary to have a chain of connected bones and a curve to constrain these bones to:

- With the last bone in the chain selected, add a *Spline IK* constraint from the *Bone Constraints* tab in the *Properties Editor*.
- Set the 'Chain Length' setting to the number of bones in the chain (starting from and including the selected bone) that should be influenced by the curve.
- Finally, set *Target* to the curve that should control the curve.

Options

Target The target curve.

Spline Fitting

Chain Length How many bones are included in the chain.

Even Division Ignore the relative length of the bones when fitting to the curve.

Chain Offset Offset the entire chain relative to the root joint.

Chain Scaling

Y stretch Stretch the Y axis of the bones to fit the curve.

XZ Scale Mode

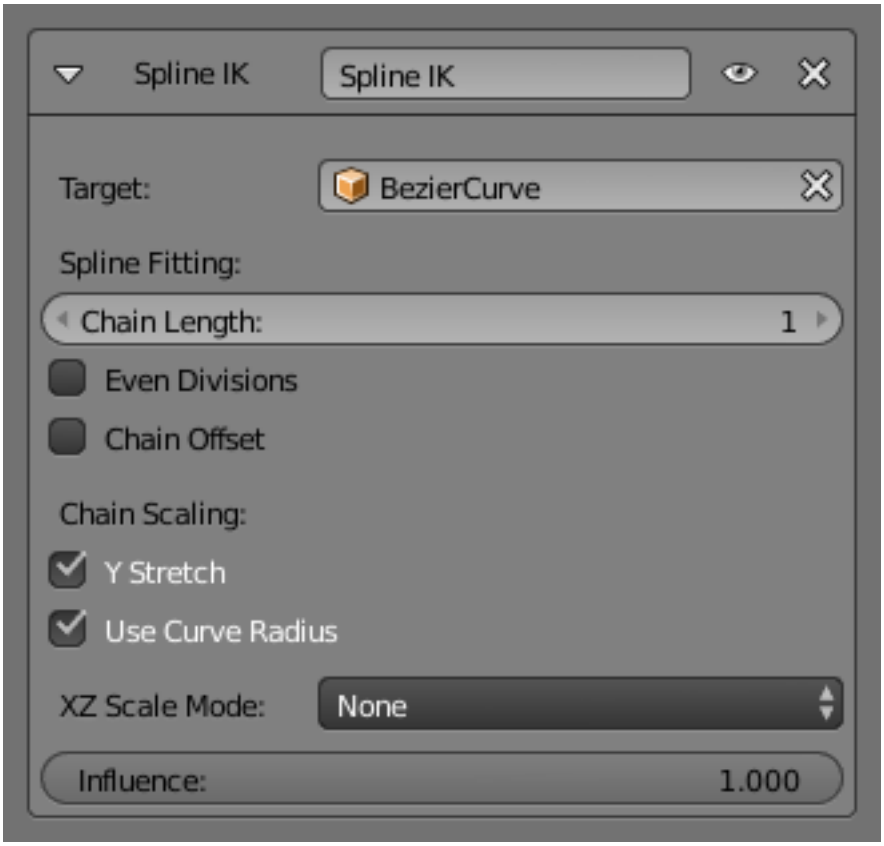


Fig. 2.1035: Spline IK panel.

None Do not scale the X and X axes.

Bone Original Use the original scaling of the bones.

Volume Preservation Scale of the X and Z axes is the inverse of the Y scale.

Use Curve Radius Average radius of the endpoints is used to tweak the X and Z scaling of the bones, on top of the X and Z scale mode.

See also:

This subject is seen in depth in the *Armature Posing section*.

Stretch To Constraint

The *Stretch To* constraint causes its owner to rotate and scale its Y axis towards its target. So it has the same tracking behavior as the *Track To constraint*. However, it assumes that the Y axis will be the tracking and stretching axis, and does not give you the option of using a different one.

It also optionally has some raw volumetric features, so the owner can squash down as the target moves closer, or thin out as the target moves farther away. Note however, that it is not the real volume of the owner which is thus preserved, but rather the virtual one defined by its scale values. Hence, this feature works even with non-volumetric objects, like empties, 2D meshes or surfaces, and curves.

With bones, the "volumetric" variation scales them along their own local axes (remember that the local Y axis of a bone is aligned with it, from root to tip).

Options

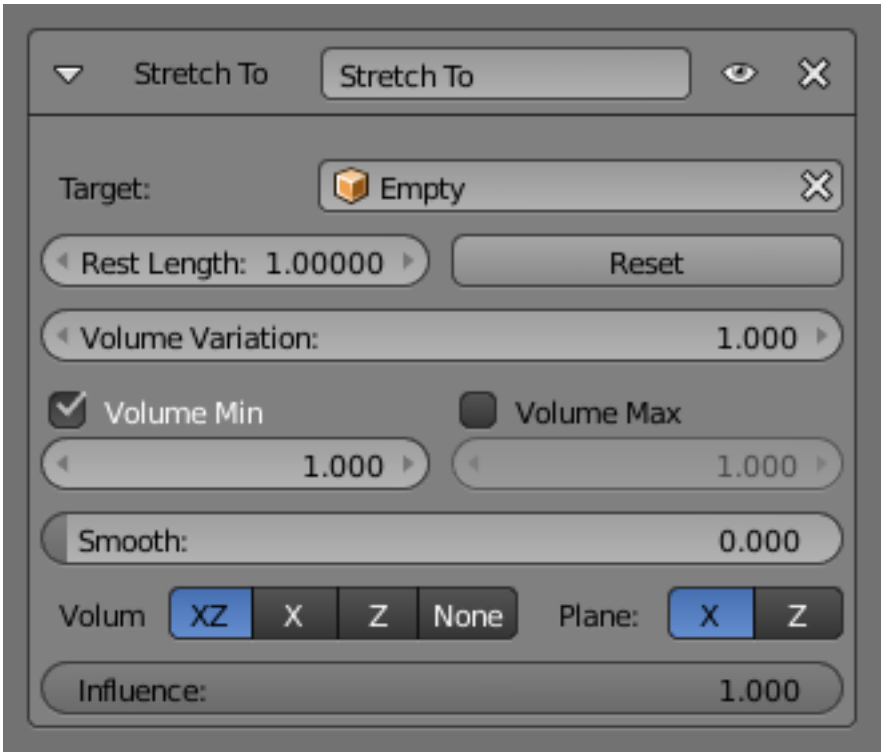


Fig. 2.1036: Stretch To panel.

Target (Mesh Object Type) This constraint uses one target, and is not functional (red state) when it has none.

Vertex Group When *Target* is a mesh, a new field is display where a vertex group can be selected.

Target (Armature Object Type) This constraint uses one target, and is not functional (red state) when it has none.

Bone When *Target* is an armature, a new field for a bone is displayed.

Head/Tail When using a Bone *Target*, you can choose where along this bone the target point lies.

Rest Length This number button sets the rest distance between the owner and its target, i.e. the distance at which there is no deformation (stretching) of the owner.

Reset When clicked, this small button will recalculate the *Rest Length* value, so that it corresponds to the actual distance between the owner and its target (i.e. the distance before this constraint is applied).

Volume Variation This number button controls the amount of "volume" variation proportionally to the stretching amount. Note that the 0.0 value is not allowed, if you want to disable the volume feature, use the *None* button (see below).

Volume These buttons control which of the X and/or Z axes should be affected (scaled up/down) to preserve the virtual volume while stretching along the Y axis. If you enable the *none* button, the volumetric features are disabled.

Plane These buttons are equivalent to the *Up* ones of the *Track To constraint*: they control which of the X or Z axes should be maintained (as much as possible) aligned with the global Z axis, while tracking the target with the Y axis.

Track To Constraint

The *Track To* constraint applies rotations to its owner, so that it always points a given "To" axis towards its target, with another "Up" axis permanently maintained as much aligned with the global Z axis (by default) as possible. This tracking is similar to the "billboard tracking" in 3D (see note below).

This is the preferred tracking constraint, as it has a more easily controlled constraining mechanism.

This constraint shares a close relationship to the *Inverse Kinematics constraint* in some ways.

Tip: Billboard tracking

The term "billboard" has a specific meaning in real-time CG programming (i.e. video games!), where it is used for plane objects always facing the camera (they are indeed "trackers", the camera being their "target"). Their main usage is as support for tree or mist textures: if they were not permanently facing the camera, you would often see your trees squeezing to nothing, or your mist turning into a millefeuille paste, which would be funny but not so credible.

Options

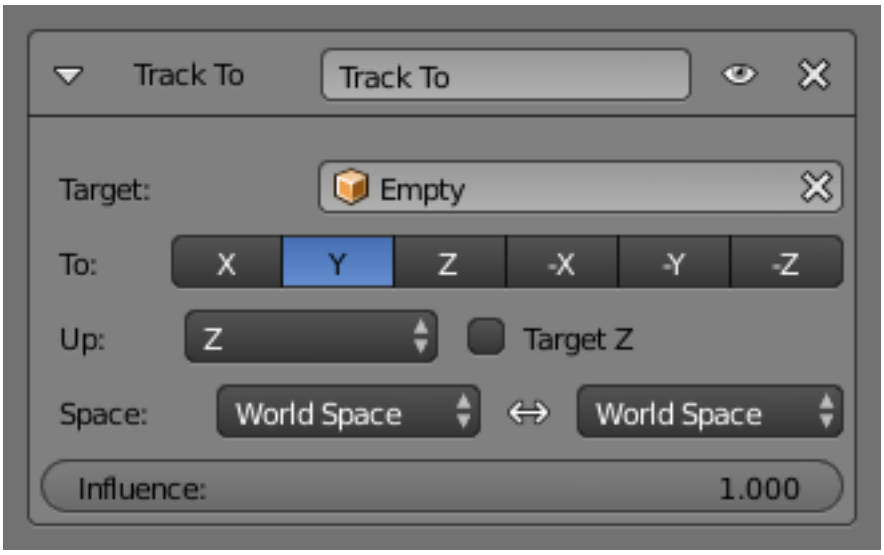


Fig. 2.1037: Track To panel.

Targets This constraint uses one target, and is not functional (red state) when it has none.

Bone When *Target* is an armature, a new field for a bone is displayed.

Head/Tail When using a bone target, you can choose where along this bone the target point lies.

Follow Bendy Bones When using a b-bone as a target, click on this button to make the target point between head and tail follow the lenght of the B-Spline curve instead of the absolute distance of the head and tail of the original b-bone.

Vertex Group When *Target* is a mesh, a new field is display where a vertex group can be selected.

To The tracking local axis, i.e. the owner's axis to point at the target. The negative options force the relevant axis to point away from the target.

Up The "upward-most" local axis, i.e. the owner's axis to be aligned (as much as possible) with the global Z axis (or target Z axis, when the *Target* button is enabled).

Target Z By default, the owner's *Up* axis is (as much as possible) aligned with the global Z axis, during the tracking rotations. When this button is enabled, the *Up* axis will be (as much as possible) aligned with the target's local Z axis?

Space This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

> **Warning:** If you choose the same axis for *To* and *Up*, the constraint will not be functional anymore (red state).

Relationship

Action Constraint

The *Action* constraint is powerful. It allows you control an *Action* using the transformations of another object.

The underlying idea of the *Action* constraint is very similar to the one behind the *Drivers*, except that the former uses a whole action (i.e. a bunch a F-Curves of the same type), while the latter controls a single F-curve of their "owner"...

Note that even if the constraint accepts the *Mesh* action type, only the *Object*, *Pose* and *Constraint* types are really working, as constraints can only affect objects' or bones' transform properties, and not meshes' shapes. Also note that only the object transformation (location, rotation, scale) is affected by the action, if the action contains keyframes for other properties they are ignored, as constraints do not influence those.

As an example, let us assume you have defined an *Object* action (it can be assigned to any object, or even no object at all), and have mapped it on your owner through an *Action* constraint, so that moving the target in the (0.0 to 2.0) range along its X-Axis maps the action content on the owner in the (0 to 100) frame range. This will mean that when the target's *X* property is 0.0 the owner will be as if in frame 0 of the linked action; with the target's *X* property at 1.0 the owner will be as if in frame 50 of the linked action, etc.

Options

Target This constraint uses one target, and is not functional (red state) when it has none.

Bone When target is an armature object, use this field to select the target bone.

Transform Channel This selector controls which transform property (location, rotation or scale along/around one of its axes) from the target to use as "action driver".

Target Space This constraint allows you to choose in which space to evaluate its target's transform properties.

To Action Select the name of the action you want to use.

> **Warning:** Even though it might not be in red state (UI refresh problems...), this constraint is obviously not functional when this field does not contain a valid action.

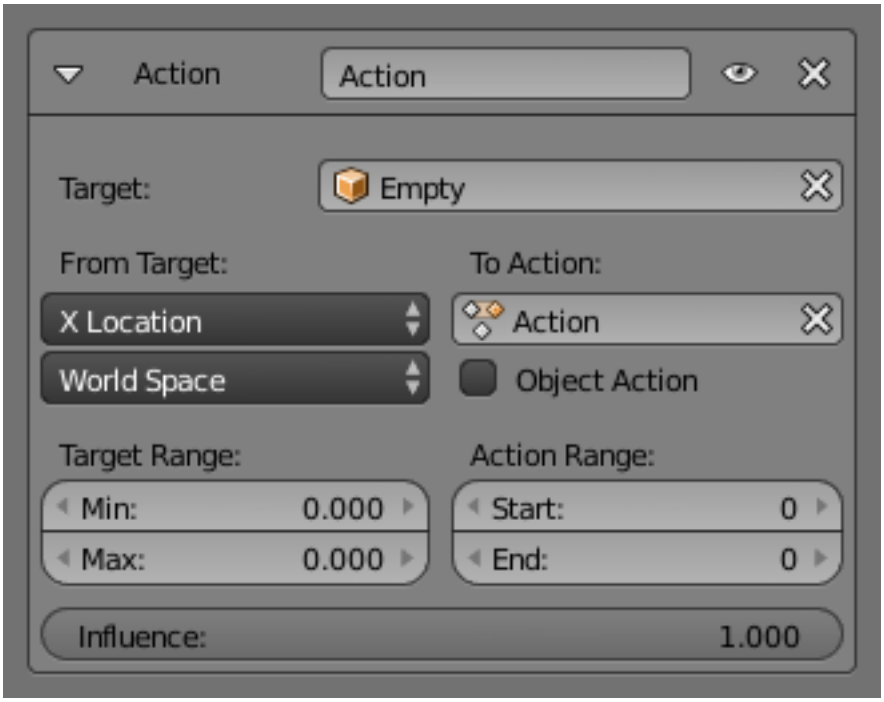


Fig. 2.1038: Action panel.

Object Action Bones **only**, when enabled, this option will make the constrained bone use the "object" part of the linked action, instead of the "same-named pose" part. This allows you to apply the action of an object to a bone.

Target Range Min/Max The lower and upper bounds of the driving transform property value.

> **Warning:** Unfortunately, here again we find the constraints limitations:
>
> - When using a rotation property as "driver", these values are "mapped back" to the (-180.0 to 180.0) range.
> - When using a scale property as "driver", these values are limited to null or positive values.

Action Range Start/End The starting and ending frames of the action to be mapped.

Note:

- These values must be strictly positive.
- By default, both values are set to 0 which disables the mapping (i.e. the owner just gets the properties defined at frame 0 of the linked action...).

Notes

- When the linked action affects some location properties, the owner's existing location is added to the result of evaluating this constraint (exactly as when the *Offset* button of the *Copy Location constraint* is enabled...).
- When the linked action affects some scale properties, the owner's existing scale is multiplied with the result of evaluating this constraint.

- When the linked action affects some rotation properties, the owner's existing rotation is overridden by the result of evaluating this constraint.
- Unlike usual, you can have a *Start* value higher than the *End* one, or a *Min* one higher than a *Max* one: this will reverse the mapping of the action (i.e. it will be "played" reversed...), unless you have both sets reversed, obviously!
- When using a *Constraint* action, it is the constraint *channel's names* that are used to determine to which constraints of the owner apply the action. E.g. if you have a constraint channel named "trackto_empt1", its keyed *Influence* and/or *Head/Tail* values (the only ones you can key) will be mapped to the ones of the owner's constraint named "trackto_empt1".
- Similarly, when using a *Pose* action (which is obviously only meaningful and working when constraining a bone!), it is the bone's name that is used to determine which bone *channel's names* from the action to use (e.g. if the constrained bone is named "arm", it will use and only use the action's bone channel named "arm"...). Unfortunately, using a *Pose* action on a whole armature object (to affect all the keyed bones in the action at once) will not work...
- Note also that you can use the *pose library feature* to create/edit a *Pose* action datablock... just remember that in this situation, there is one pose per frame!

Child Of Constraint

Child Of is the constraint version of the standard parent/children relationship between objects (the one established through the `Ctrl-P` shortcut, in the 3D Views).

Parenting with a constraint has several advantages and enhancements, compared to the traditional method:

- You can have several different parents for the same object (weighting their respective influence with the *Influence* slider).
- As with any constraint, you can key (i.e. animate) its Influence setting. This allows the object which has a Child Of constraint upon it to change over time which target object will be considered the parent, and therefore have influence over the Child Of constrained object.

> **Warning:** Do not confuse this "basic" object parenting with the one that defines the *chains of bones* inside of an armature. This constraint is used to parent an object to a bone (the so-called *object skinning*), or even bones to bones. But do not try to use it to define chains of bones.

Options

Target The target object that this object will act as a child of. This constraint uses one target, and is not functional (red state) when it has none. If *Target* is an armature or a mesh, a new name field appears where a name of a *Bone* or a *Vertex Group* can be selected.

Location X, Y, Z Each of these buttons will make the parent affect or not affect the location along the corresponding axis.

Rotation X, Y, Z Each of these buttons will make the parent affect or not affect the rotation around the corresponding axis.

Scale X, Y, Z Each of these buttons will make the parent affect or not affect the scale along the corresponding axis.

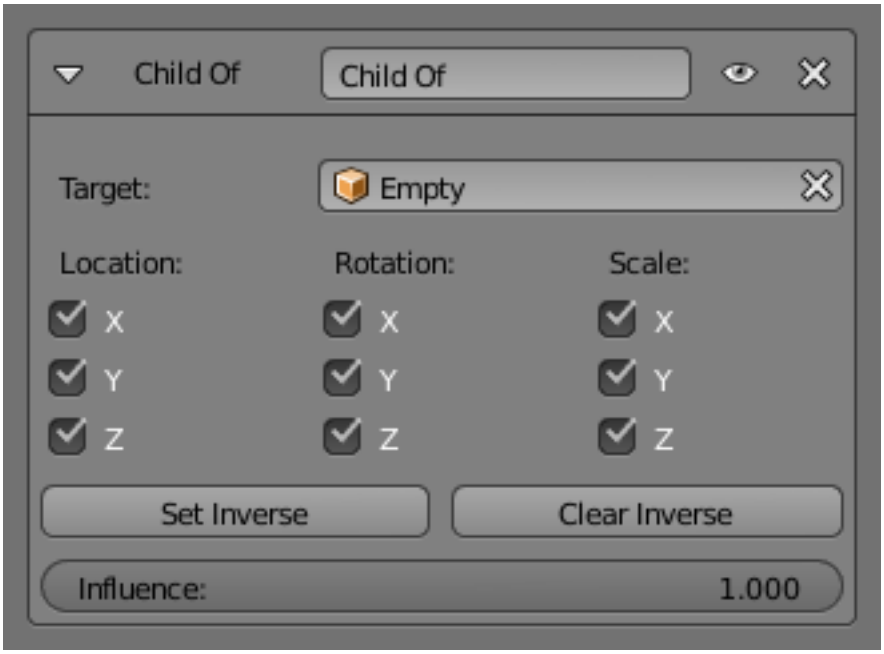


Fig. 2.1039: Child Of panel.

Set Inverse By default, when you parent your owner to your target, the target becomes the origin of the owner's space. This means that the location, rotation and scale of the owner are offset by the same properties of the target. In other words, the owner is transformed when you parent it to your target. This might not be desired! So, if you want to restore your owner to its before-parenting state, click on the *Set Inverse* button.

Clear Inverse This button reverses (cancels) the effects of the above one, restoring the owner/child to its default state regarding its target/parent.

Tips

When creating a new parent relationship using this constraint, it is usually necessary to click on the *Set Inverse* button after assigning the parent. As noted above, this cancels out any unwanted transform from the parent, so that the owner returns to the location/rotation/scale it was in before the constraint was applied. Note that you should apply *Set Inverse* with all other constraints disabled (their *Influence* set to 0.0) for a particular *Child Of* constraint, and before transforming the target/parent (see example below).

About the toggle buttons that control which target's (i.e. parent's) individual transform properties affect the owner, it is usually best to leave them all enabled, or to disable all three of the given Location, Rotation or Scale transforms.

Technical Note

If you use this constraint with all channels on, it will use a straight matrix multiplication for the parent relationship, not decomposing the parent matrix into loc/rot/size. This ensures any transformation correctly gets applied, also for combinations of rotated and non-uniform scaled parents.

Examples

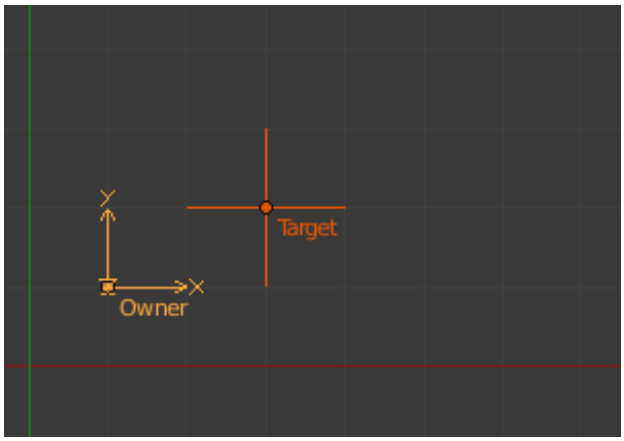


Fig. 2.1040: No constraint.
Note the position of Owner empty 1.0 BU along X- and Y-Axis.

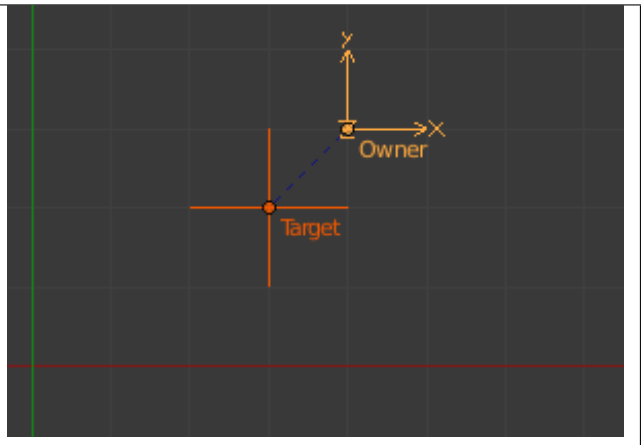


Fig. 2.1041: Child Of just added.
Here you can see that Owner empty is now 1.0 BU away from Target_1 empty along X- and Y-Axis.

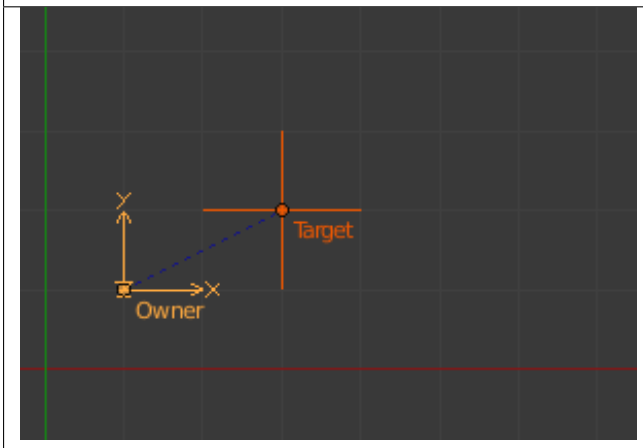


Fig. 2.1042: Offset set.
Set Inverse has been clicked, and Owner is back to its original position.

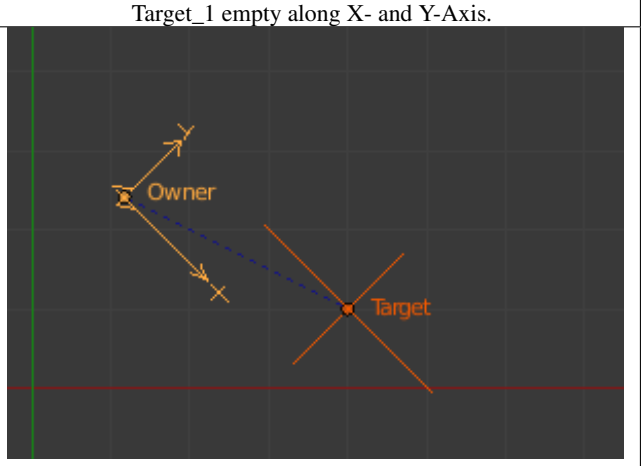


Fig. 2.1043: Target/parent transformed.
Target_1 has been translated in the XY plane, rotated around the Z-Axis, and scaled along its local X-Axis.

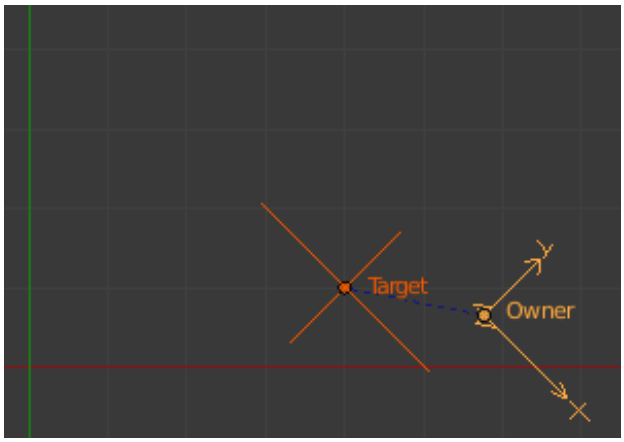


Fig. 2.1044: Offset cleared.
Clear Inverse has been clicked. Owner is fully again controlled by Target_1.

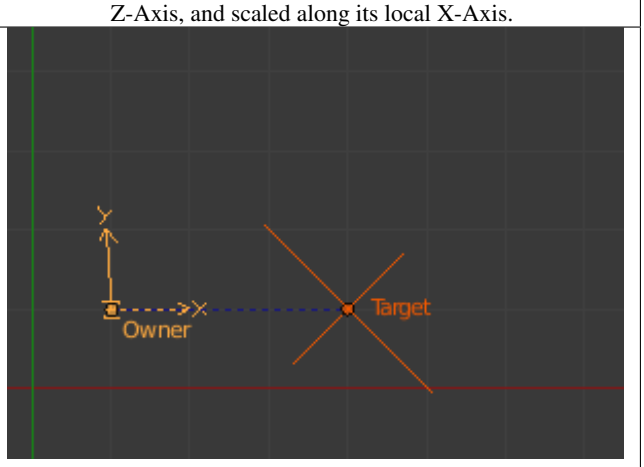


Fig. 2.1045: Offset set again.
Set Offset has been clicked again. As you can see, it does not gives the same result as in (Target/parent transformed). As noted above, use Set Inverse only once, before transforming your target/parent.

Floor Constraint

The *Floor* constraint allows you to use its target position (and optionally rotation) to specify a plane with a "forbidden side", where the owner cannot go. This plane can have any orientation you like. In other words, it creates a floor (or a ceiling, or a wall)! Note that it is only capable of simulating entirely flat planes, even if you use the *Vertex Group* option. It cannot be used for uneven floors or walls.

Options

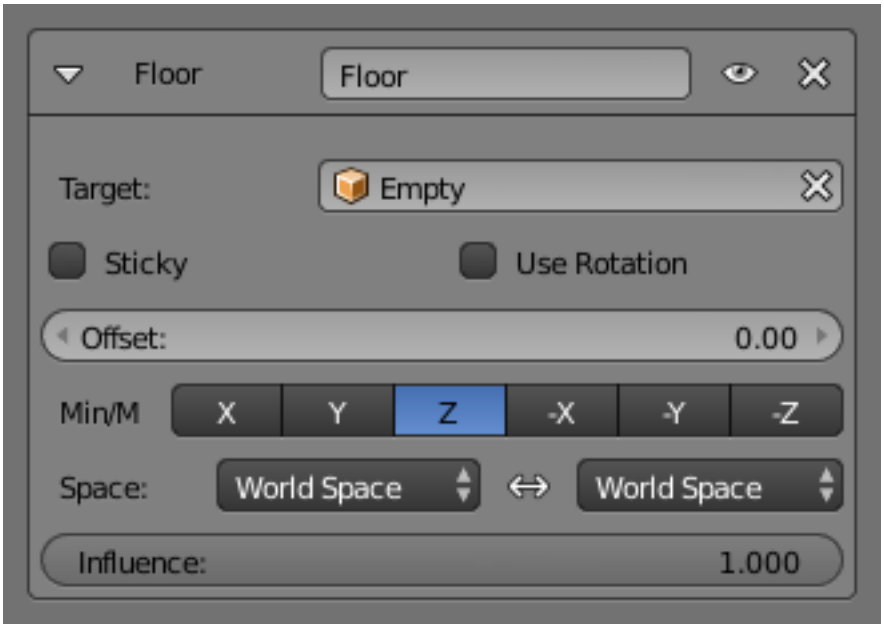


Fig. 2.1046: Floor panel.

Targets This constraint uses one target, and is not functional (red state) when it has none.

Bone When *Target* is an armature, a new field for a bone is displayed.

Vertex Group When *Target* is a mesh, a new field is display where a vertex group can be selected.

Sticky This button makes the owner immovable when touching the "floor" plane (it cannot slide around on the surface of the plane any more). This is fantastic for making walk and run animations!

Use Rotation This button forces the constraint to take the target's rotation into account. This allows you to have a "floor" plane of any orientation you like, not just the global XY, XZ and YZ ones...

Offset This number button allows you to offset the "floor" plane from the target's center, by the given number of Blender Units. Use it e.g. to account for the distance from a foot bone to the surface of the foot's mesh.

Max/Min This set of (mutually exclusive) buttons controls which plane will be the "floor". The buttons' names correspond indeed to the *normal* to this plane (e.g. enabling Z means "XY plane", etc.) By default, these normals are aligned with the *global* axes. However, if you enable *Use Rotation* (see above), they will be aligned with the *local target's axes*. As the constraint does not only define an uncrossable plane, but also a side of it which is forbidden to the owner, you can choose which side by enabling either the positive or negative normal axis... e.g. by default Z, the owner is stuck in the positive Z coordinates.

Space This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

Follow Path Constraint

The *Follow Path* constraint places its owner onto a *curve* target object, and makes it move along this curve (or path). It can also affect its owner's rotation to follow the curve's bends, when the *Follow Curve* option is enabled.

The owner is always evaluated in the global (world) space:

- Its location (as shown in the *Transform* panel) is used as an offset from its normal position on the path. E.g. if you have an owner with the (1.0, 1.0, 0.0) location, it will be one BU away from its normal position on the curve, along the X and Y axis. Hence, if you want your owner *on* its target path, clear its location `Alt-G`!
- This location offset is also proportionally affected by the scale of the target curve. Taking the same (1.0, 1.0, 0.0) offset as above, if the curve has a scale of (2.0, 1.0, 1.0), the owner will be offset *two* BU along the X axis (and one along the Y one)...
- When the *Curve Follow* option is enabled, its rotation is also offset to the one given by the curve (i.e. if you want the Y axis of your object to be aligned with the curve's direction, it must be in rest, non-constrained state, aligned with the global Y axis). Here again, clearing your owner's rotation `Alt-R` might be useful...

The movement of the owner along the target curve/path may be controlled in two different ways:

- The most simple is to define the number of frames of the movement, in the Path Animation panel of the Object Data tab, via the number button Frames, and its start frame via the constraint's Offset option (by default, start frame: 1 [= offset of 0)], duration: 100).
- The second way, much more precise and powerful, is to define a *Evaluation Time* interpolation curve for the *Target* path (in the *Graph Editor*. See the *animation chapter* to learn more about F-Curves.
- If you do not want your owner to move along the path, you can give to the target curve a flat *Speed* F-Curve (its value will control the position of the owner along the path).

Follow Path is another constraint that works well with the *Locked Track one*. One example is a flying camera on a path. To control the camera's roll angle, you can use a *Locked Track* and a target object to specify the up direction, as the camera flies along the path.

Note: *Follow Path* and *Clamp To*

Do not confuse these two constraints. Both of them constraint the location of their owner along a curve, but *Follow Path* is an "animation-only" constraint, inasmuch that the position of the owner along the curve is determined by the time (i.e. current frame), whereas the *Clamp To constraint* determines the position of its owner along the curve using one of its location properties' values.

Note: Note that you also need to keyframe Evaluation Time for the Path. Select the path, go to the path properties, set the overall frame to the first frame of the path (e.g. frame 1), set the value of Evaluation time to the first frame of the path (e.g. 1), right click on Evaluation time, select create keyframe, set the overall frame to the last frame of the path (e.g. frame 100), set the value of Evaluation time to the last frame of the path (e.g. 100), right click on Evaluation time, select create keyframe.

Options

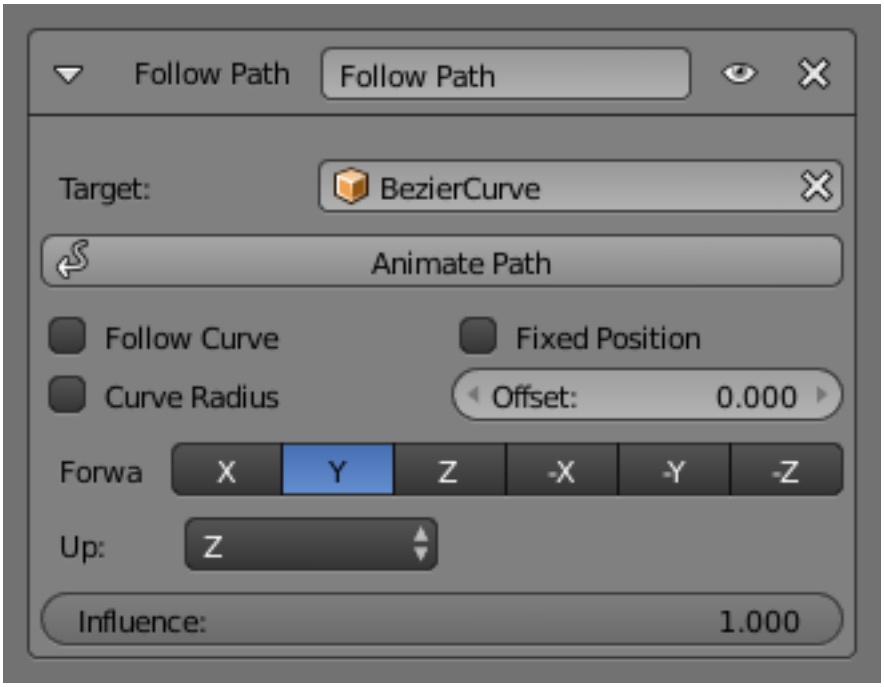


Fig. 2.1047: Follow Path panel.

Target This constraint uses one target, which *must* be a curve object, and is not functional (red state) when it has none.

Curve Radius Objects scale by the curve radius. See *Curve Editing*

Fixed Position Object will stay locked to a single point somewhere along the length of the curve regardless of time

Offset The number of frames to offset from the "animation" defined by the path (by default, from frame 1).

Follow Curve If this option is not activated, the owner's rotation is not modified by the curve; otherwise, it is affected depending on the following options:

Forward The axis of the object that has to be aligned with the forward direction of the path (i.e. tangent to the curve at the owner's position).

Up The axis of the object that has to be aligned (as much as possible) with the world Z axis. In fact, with this option activated, the behavior of the owner shares some properties with the one caused by a *Locked Track constraint*, with the path as "axle", and the world Z axis as "magnet".

Pivot Constraint

The *Pivot* constraint allows the owner to rotate around a target object.

It was originally intended for foot rigs.

Options

Target The object to be used as a pivot point.

Bone When *Target* is an armature, a new field for a bone is displayed.

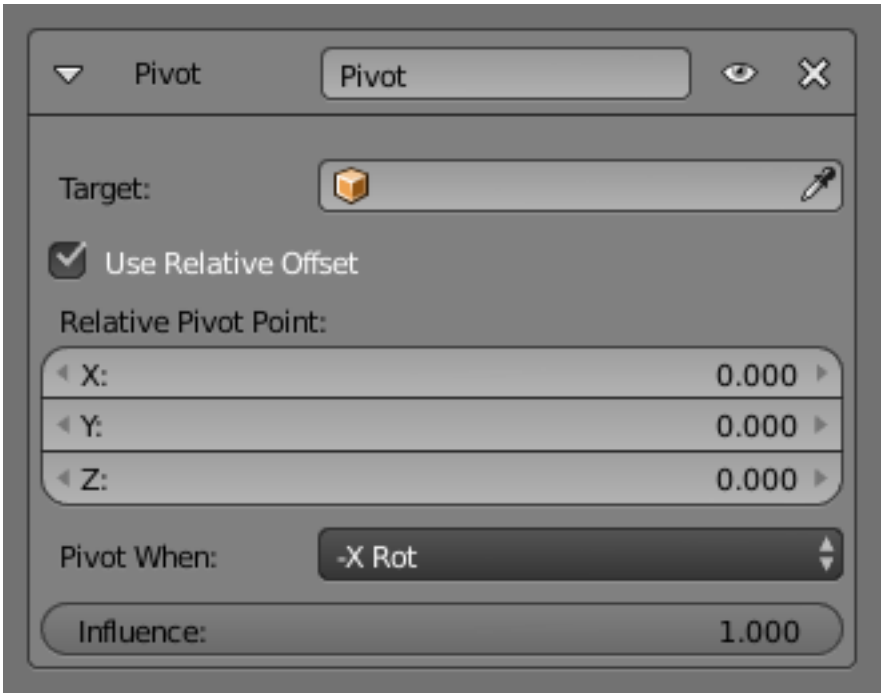


Fig. 2.1048: Pivot panel.

Head/Tail When using a bone target, you can choose where along this bone the target point lies.

Vertex Group When *Target* is a mesh, a new field is displayed where a vertex group can be selected.

Pivot Offset Offset of pivot from target.

Pivot When Always, Z Rot, Y Rot...

Example

Rigid Body Joint Constraint

The *Rigid Body Joint* constraint is very special, it is used by the physics part of the Blender Game Engine to simulate a joint between its owner and its target. It offers four joint types: hinge type, ball-and-socket type, cone-twist, and generic six-DoF (degrees of freedom) type.

> **Warning:** This constraint only works with the *Game Engine*.

The joint point and axes are defined and fixed relative to the owner. The target moves as if it were stuck to the center point of a stick, the other end of the stick rotating around the joint/pivot point...

This constraint is of no use in most "standard" static or animated projects. However, you can use its results outside of the BGE, through the *Game* → *Record Animation*. see *Rigid Bodies* for more info on this topic).

For a demo file that shows some of the different types, see: BGE-Physics-RigidBodyJoints.blend.

Note: In order for this constraint to work properly, both objects (so the owner and the target object) need to have *Collision Bounds* enabled.

Options

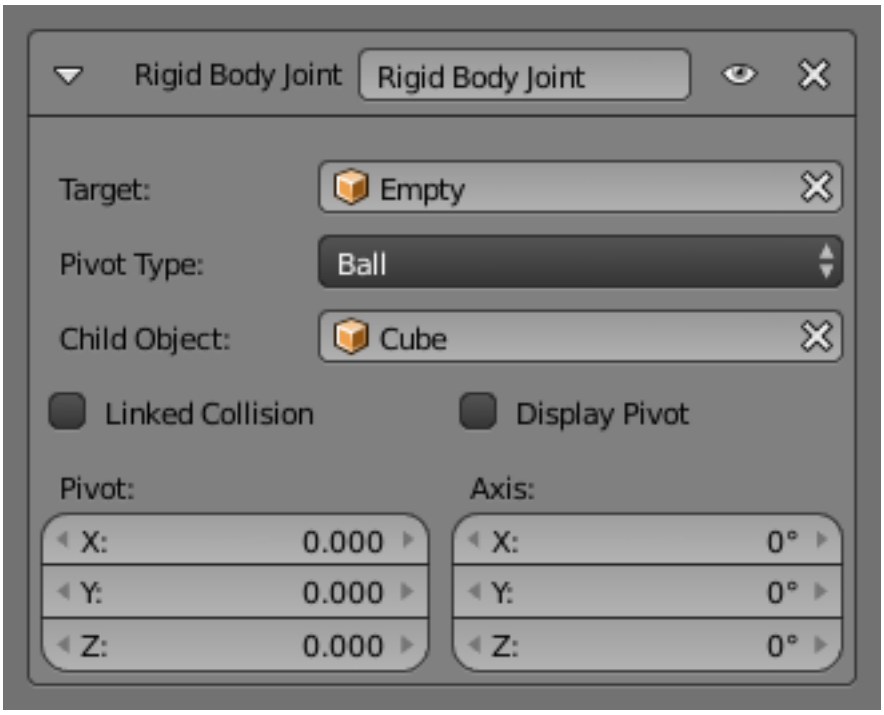


Fig. 2.1049: Rigid Body Joint panel.

Target This constraint uses one target, and is not functional (red state) when it has none.

Joint Type

Ball works like an ideal ball-and-socket joint, i.e. allows rotations around all axes like a shoulder joint.

Hinge works in one plane, like an elbow: the owner and target can only rotate around the X axis of the pivot (joint point).

Limits Angular limits for the X axis

Cone Twist similar to *Ball*, this is a point-to-point joint with limits added for the cone and twist axis

Limits Angular limits

Generic 6DOF works like the *Ball* option, but the target is no longer constrained at a fixed distance from the pivot point, by default (hence the six degrees of freedom: rotation and translation around/along the three axes). In fact, there is no longer a joint by default, with this option, but it enables additional settings which allow you to restrict some of these DoF:

Limits Linear and angular limits for a given axis (of the pivot) in Blender Units and degrees respectively.

Child Object normally, leave this blank. You can reset it to blank by right clicking and selecting Reset to Default Value.

Linked Collision When enabled, this will disable the collision detection between the owner and the target (in the physical engine of the BGE).

Display Pivot When enabled, this will draw the pivot of the joint in the 3D Views. Most useful, especially with the *Generic 6DOF* joint type!

Pivot These three numeric fields allow you to relocate the pivot point, *in the owner's space*.

Axis These three numeric fields allow you to rotate the pivot point, *in the owner's space*.

Shrinkwrap Constraint

The *Shrinkwrap* constraint is the "object counterpart" of the *Shrinkwrap modifier*. It moves the owner origin and therefore the owner object's location to the surface of its target.

This implies that the target *must* have a surface. In fact, the constraint is even more selective, as it can only use meshes as targets. Hence, the *Shrinkwrap* option is only shown in the *Add Constraint to Active Object* menu, `Ctrl-Alt-C`, (or its bone's equivalent), when the selected inactive object is a mesh.

Options

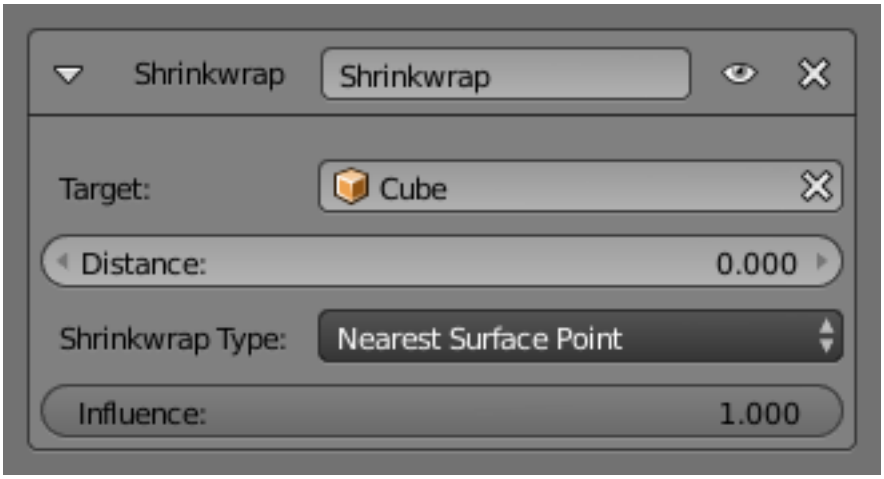


Fig. 2.1050: Shrinkwrap panel.

Target This constraint uses one target, which *must* be a mesh object, and is not functional (red state) when it has none.

Distance This number button controls the offset of the owner from the shrunk computed position on the target's surface. Positive values place the owner "outside" of the target, and negative ones, "inside" the target. This offset is applied along the straight line defined by the original (i.e. before constraint) position of the owner, and the computed one on the target's surface.

Shrinkwrap Type

This selector allows you to select which method to use to compute the point on the target's surface to which to translate the owner's center. You have three options:

Nearest Surface Point

The chosen target's surface's point will be the nearest one to the original owner's location. This is the default and most commonly useful option.

Projection

The target's surface point is determined by projecting the owner's center along a given axis.

Projection Axis This axis is controlled by the radio buttons that show up when you select this type. This mean the projection axis can only be aligned with one of the global axes, median to both of them (XY, XZ or YZ), or to the three ones (XYZ). When the projection of the owner's center along the selected direction does not hit the target's surface, the owner's location is left unchanged.

+X, +Y, +Z, -X, -Y, -Z

Axis Space ToDo.

Projection Distance ToDo.

Nearest Vertex

This method is very similar to the *Nearest Surface Point* one, except that the owner's possible shrink locations are limited to the target's vertices.

2.6.3 Armatures

Introduction

An Armature in Blender can be thought of as similar to the armature of a real skeleton, and just like a real skeleton an Armature can consist of many bones. These bones can be moved around and anything that they are attached to or associated with will move and deform in a similar way.

An "armature" is a type of object used for *rigging*. Armature object borrows many ideas from real life skeletons.

Your first armature

In order to see what we are talking about, let us try to add the default armature in Blender.

(Note that armature editing details are explained in the *armatures editing section*).

Open a default scene, then:

1. Delete all objects in the scene.
2. Make sure the cursor is in the world origin with `Shift-C`.
3. Press `Numpad1` to see the world in Front view.
4. Add a *Single Bone* (*Add* → *Armature* → *Single Bone*).
5. Press `NumpadDelete` to see the armature at maximum zoom.

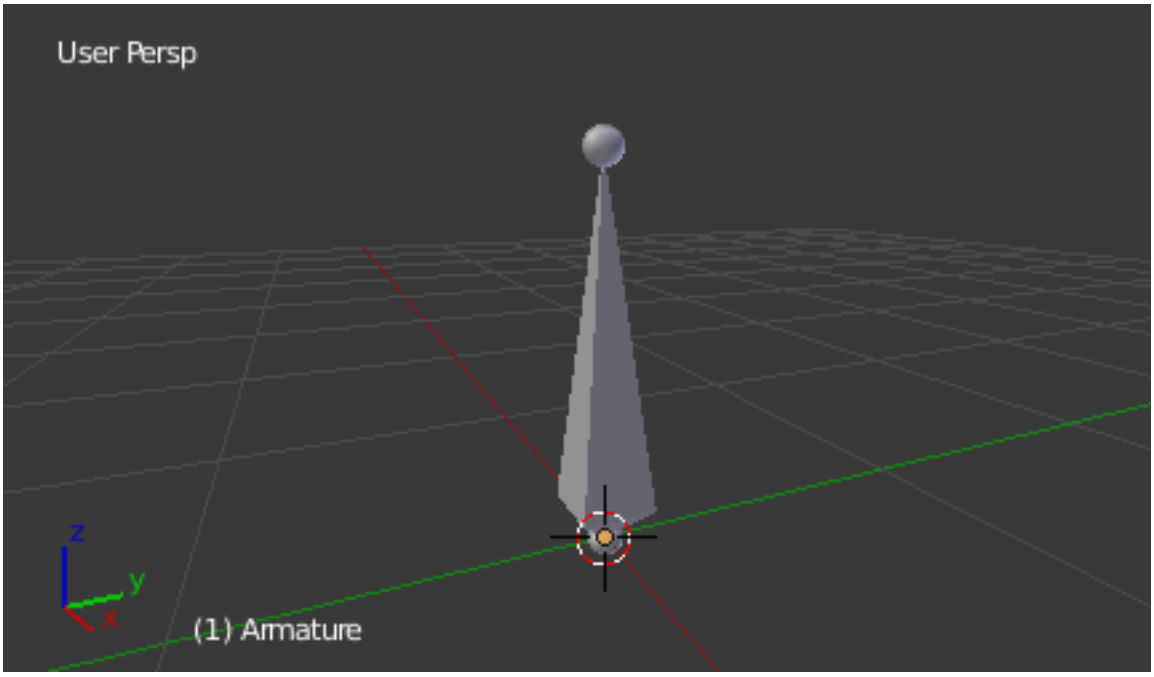


Fig. 2.1051: The default armature.

The Armature Object

As you can see, an armature is like any other object type in Blender:

- It has a center, a position, a rotation and a scale factor.
- It has an Object Data data-block, that can be edited in *Edit Mode*.
- It can be linked to other scenes, and the same armature data can be reused on multiple objects.
- All animation you do in *Object Mode* is only working on the whole object, not the armature's bones (use the *Pose Mode* to do this).

  As armatures are designed to be posed, either for a static or animated scene, they have a specific state, called "rest position". This is the armature's default "shape", the default position/rotation/scale of its bones, as set in *Edit Mode*.

  In *Edit Mode*, you will always see your armature in rest position, whereas in *Object Mode* and *Pose Mode*, you usually get the current "pose" of the armature (unless you enable the *Rest Position* button of the *Armature* panel).

Armature Chapter Overview

In the "Armatures" section, we will only talk about armatures themselves, and specifically we will talk about:

- The basics of *bones*.
- The different *armature visualizations*.
- The armature *structure types*.
- How to *Select Bones*,
- How to *Edit Armatures*,
- How to *Edit Bones*,

- How to *edit bones properties*,
- How to sketch armatures with the *Etch-a-Ton tool*,
- How to use *templates*.

Bones

Introduction

Bones are the base elements of armatures. The visualization of bones can be set in the Armatures *Display Panel*.

Structure

They have three elements:

- The "start point" named *root* or *head*,
- the "body" itself,
- and the "end point" named *tip* or *tail*.

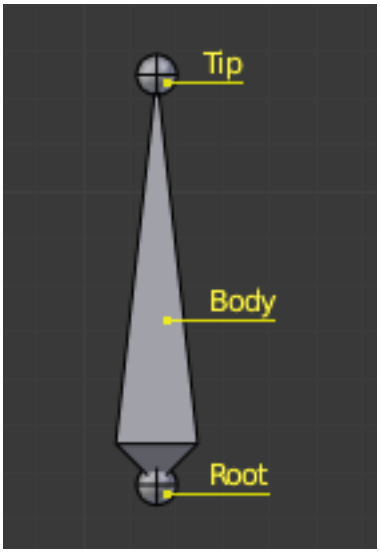


Fig. 2.1052: The elements of a bone.

With the default armature in edit-mode, you can select the root and the tip, and move them as you do with mesh vertices.

Both root and tip (the "ends") define the bone by their respective position.

They also have a radius property, only useful for the envelope deformation method (see below).

Roll

Activating *Axes* checkbox on the *Armature tab* → *Display panel*, will show local axes for each bone's tip. The Y axis is always aligned along the bone, oriented from root to tip. So, this is the "roll" axis of the bones.

Bones Influence

Basically, a bone controls a geometry when vertices "follow" the bone. This is like how the muscles and skin of your finger follow your finger-bone when you move a finger.

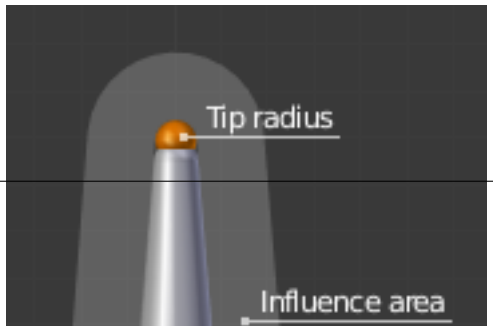
To do this, you have to define the strength of *influences* a bone has on a certain vertex.

The simplest way is to have each bone affecting those parts of the geometry that are within a given range from it. This is called the *envelope technique*, because each bone can control only the geometry "enveloped" by its own influence area.

If a bone is visualized as *Envelope*, in *Edit Mode* and in *Pose Mode* you can see the area of influence, which depends on:

- The *distance* property and
- the root's radius and the tip's radius.

All these influence parameters are further detailed in the *skinning pages*.



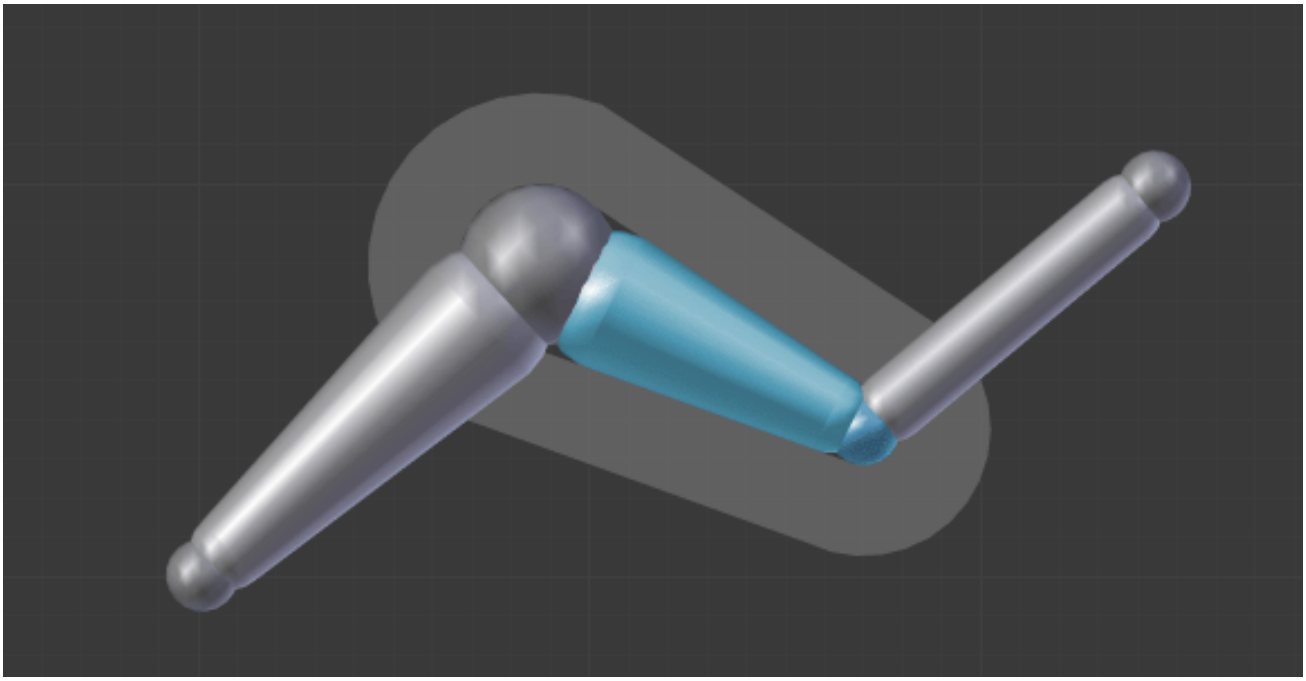


Fig. 2.1054: Our armature in Envelope visualization, in Pose Mode.

Selecting

Reference

Mode: Edit Mode
Panel: *Bone* panel

You can select and edit bones of armatures in *Edit Mode* and in *Pose Mode*. Here, we will see how to select bones in *Edit Mode*. Selecting bones in *Pose Mode* is similar to selecting in *Edit Mode* with a few specific differences that will be detailed in the *posing part*.

Similar to *vertices/edges selection* in meshes, there are two ways to select whole bones in *Edit Mode*:

- directly, by selecting the bone's body
- selecting both of its end points (root and tip)

This is an important point to understand, because selecting bones' ends only might lead to non-obvious behavior, with respect to which bone you actually select, see the.

Note that unlike the mesh draw type the armature draw type has no effect on selection behavior. In other words, you can

select a bone's end or body the same way regardless of the bone visualization chosen.

Selecting bones' ends

To select bones' ends you have the standard selection methods.

action	shortcut	menu	mouse
Select a bone's end			RMB -click on it
Add or Remove from the current selection			Shift−RMB
(De)select the ends of all bones	A	*Select* → *Select/Deselect All*	
Invert the current selection	Ctrl−I	*Select* → *Inverse*	
Box selection tool activated	B	*Select* → *Border Select*	
Box selection	Click and drag LMB the box around the ends you want to add to the current selection Click and drag LMB to remove from the current selection release LMB to validate press Esc or click RMB to cancel		
Box selection tool deactivated	B or Esc		RMB
Lasso selection	Click and drag Ctrl−LMB the lasso around the ends you want to add to the current selection Click and drag Ctrl−Shift−LMB to remove from the current selection Release LMB to validate Hit Esc or click RMB to cancel		

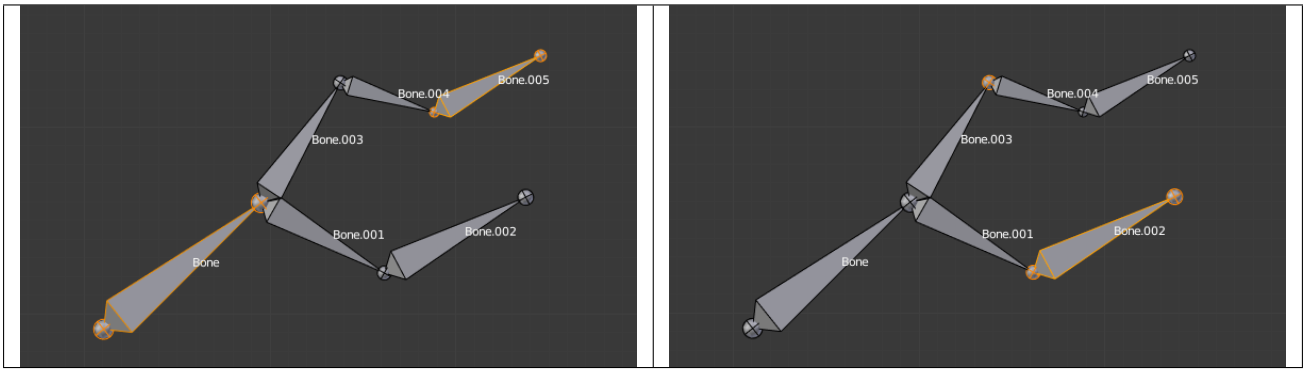
Inverse selection

As stated above, you have to remember that these selection tools are for bones' ends only, not the bones' bodies.

For example, the *Inverse* selection option Ctrl−I inverts the selection of bones' ends, not of bones (see *Inverse selection*).

Remember that a bone is selected only if both its ends are selected. So, when the selection status of bones' ends is inverted, a new set of bones is selected.

Table 2.41: The result of the inverse selection Ctrl-I the bones ends selection has been inverted, and not the bones selection.



Selecting connected bones' ends

Another example is: when you select the root of a bone connected to its parent, you also implicitly select the tip of its parent (and vice versa).

Note: Remember that when selecting bones' ends, the tip of the parent bone is the "same thing" as the root of its children bones.

Selecting Bones

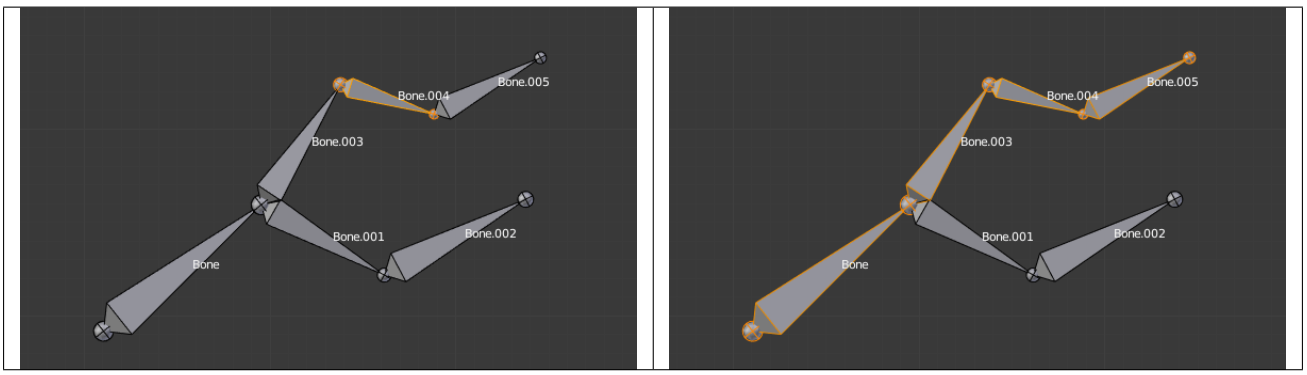
By RMB -clicking on a bone's body, you will select it (and hence you will implicitly select its root and tip).

Using Shift-RMB, you can add to/remove from the selection.

You also have some *advanced selection* options, based on their relations.

You can select at once all the bones in the chain which the active (last selected) bone belongs to by using the *linked selection* tool, L.

Table 2.42: Its whole chain selected with L.



You can deselect the active bone and select its immediate parent or one of its children using respectively *Select* → *Select Parent*, [or *Select* → *Select Child*,]. If you prefer to keep the active bone in the selection, use *Select* → *Extend Select Parent*, Ctrl-[or *Select* → *Extend Select Child*, Ctrl-].

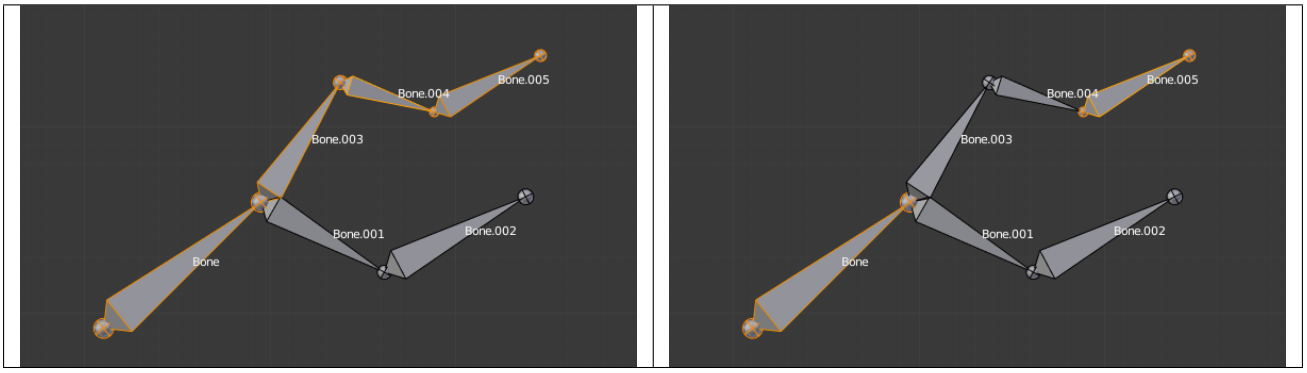
Deselecting connected bones

There is a subtlety regarding connected bones.

When you have several connected bones selected, if you deselect one bone, its tip will be deselected, but not its root, if it is also the tip of another selected bone.

To understand this, look at Fig. *Bone deselection in a selected chain.*.

Table 2.43: Two selected bones.



After `Shift−RMB` -clicking “Bone.003”:

- “Bone.003” ‘s tip (which is same as “Bone.004” ‘s root) is deselected.
- “Bone” is “Bone.003” ‘s parent. Therefore “Bone.003” ‘s root is same as the tip of “Bone”. Since “Bone” is still selected, its tip is selected. Thus the root of “Bone.003” remains selected.

Editing

Introduction

Reference

Mode: Edit Mode
Hotkey: `Tab`

As with any other object, you edit your armature in *Edit Mode* `Tab`.

Editing an armature means two main domains of action:

- *Editing the bones* - i.e. adding/inserting/deleting/extruding/sub-dividing/joining them...
- *Editing the bones' properties* - this includes key features, like transform properties (i.e. grab, scale, etc...) and relationships between bones (parenting and connecting), as well as bones' names, influence, behavior in *Pose Mode*, etc.

These are standard editing methods, quite similar for example to *meshes* editing. Blender also features a more advanced “armature sketching” tool, called *Etch-a-Ton*. The same tool might also be used in *templating*, i.e. using another armature as template for the current one...

Important: One important thing to understand about armature editing is that you edit the *rest position* of your armature, i.e. its “default state”. An armature in its *rest position* has all bones with *no* rotation and scaled to 1.0 in their own local

space.

The different *poses* you might create afterwards are based on this rest position. So if you modify it in *Edit Mode*, all the poses already existing will also be modified. Thus you should in general be sure that your armature is definitive before starting to *skin* and *pose* it!

Note: Please note that some tools work on bones' ends, while others work on bones themselves. Be careful not to get confused.

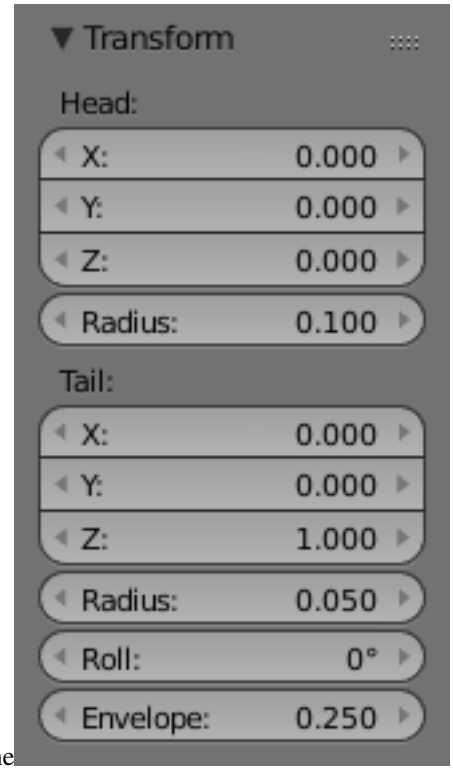
Transform

Transform

We will not detail here the various transformations of bones, nor things like axis locking, pivot points, and so on, as they are common to most object editing, and already described *here* (note however, that some options, like snapping, do not seem to work, even though they are available...). The same goes for mirroring, as it is nearly the same as with *mesh editing*. Just keep in mind that bones' roots and tips behave more or less like meshes' vertices, and bones themselves act like edges in a mesh.

As you know, bones can have two types of relationships: They can be parented, and in addition connected. Parented bones behave in *Edit Mode* exactly as if they had no relations. They can be grabbed, rotated, scaled, etc. a parent bone without affecting its descendants. However, connected bones must always have parent's tips connected to child's roots, so by transforming a bone, you will affect all its connected parent/children/siblings.

Finally, you can edit in the *Transform* panel in the Properties region the positions and radius of both ends of the active selected bone, as well as its *roll rotation*.



None

Fig. 2.1061: The Transform panel for armatures in Edit Mode.

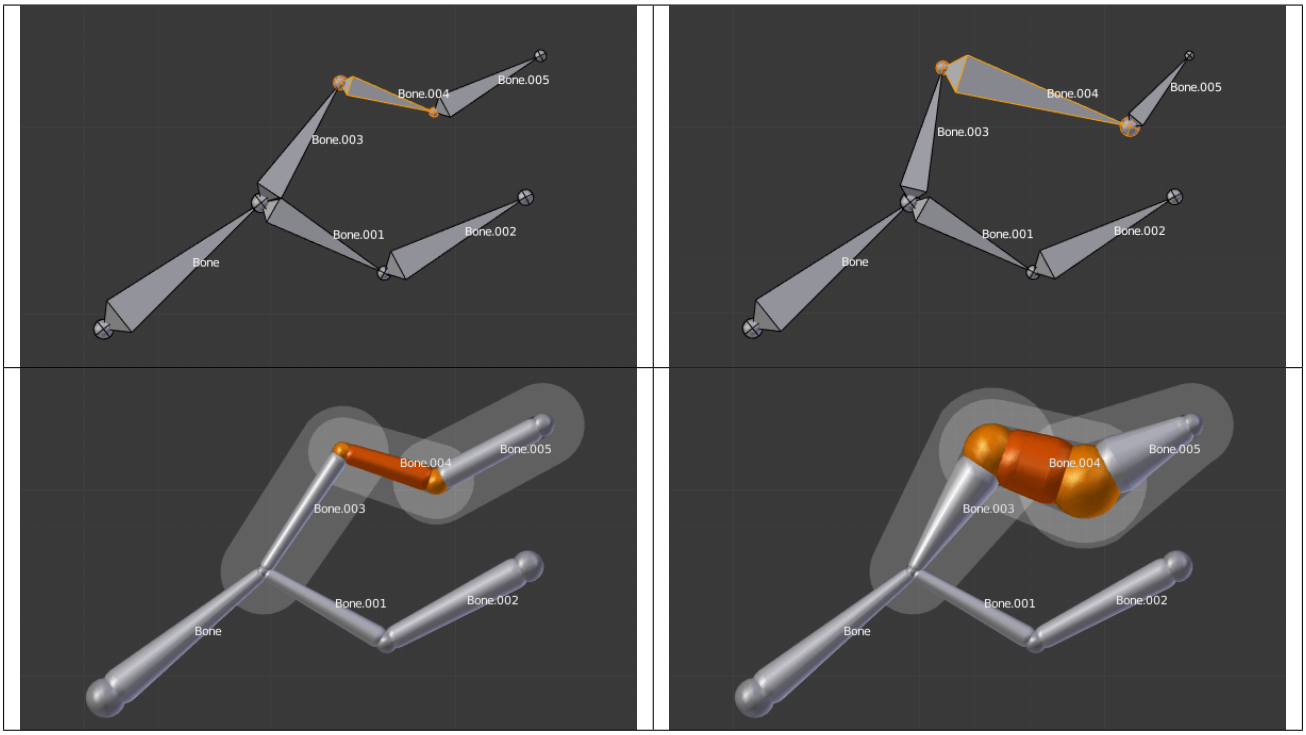
Radius and Scaling in Envelope Visualization

Reference

Mode: Edit Mode, Envelope visualization
Menu: *Armature* → *Transform* → *Scale*
Hotkey: `S`

When bones are displayed using *Octahedron*, *Stick* or *B-Bone* visualizations, scaling will behave as expected, similar to scaling mesh objects. When bones are displayed using *Envelope* visualization, scaling will have a different effect: it will scale the radius of the selected bones's ends. (see: *skinning part*). As you control only one value (the radius), there is no axis locking here. And as usual, with connected bones, you scale at the same time the radius of the parent's tip and of the children's roots.

Table 2.44: ...Scaled in Envelope visualization. Its length remains the same, but its ends' radius are bigger.



Note that when you resize a bone (either by directly scaling it, or by moving one of its ends), Blender automatically adjusts the end-radii of its envelope proportionally to the size of the modification. Therefore, it is advisable to place all the bones first, and only then edit these properties.

ScaleB and Envelope

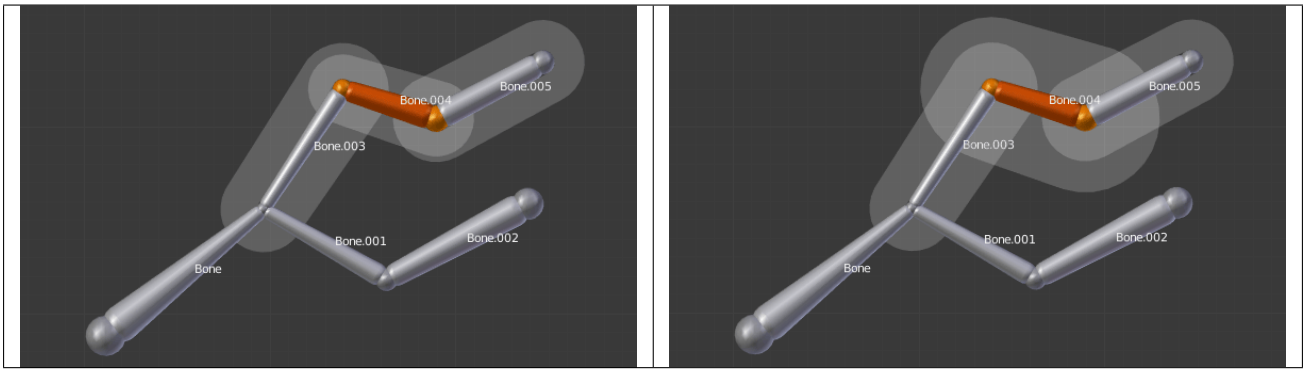
Reference

Mode: Edit Mode
Hotkey: `Ctrl-Alt-S`

`Ctrl-Alt-S` activates a transform tool that is specific to armatures. It has different behavior depending on the active visualization, as explained below:

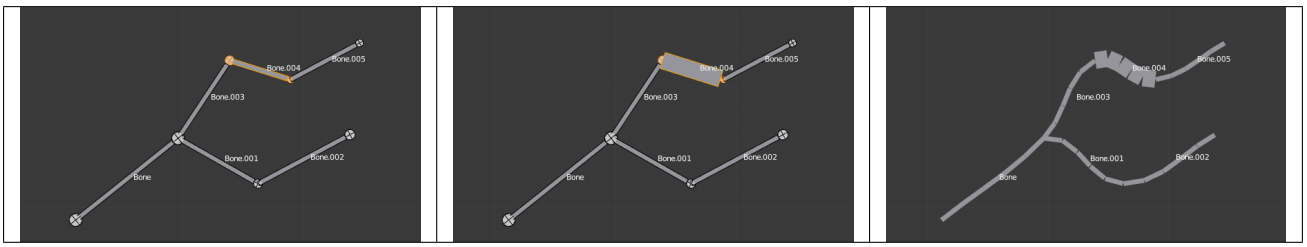
In *Envelope* visualization, it allows you to edit the influence of the selected bones (their *Distance* property, see the *skinning part*) – as with the "standard" scaling with this visualization (see the previous section), this is an one-value property, so there is no axis locking and such.

Table 2.45: Its envelope scaled with `Ctrl-Alt-S`.



In the other visualizations, it allows you to edit the "bone size". This seems to only have a visible effect in *B-Bone* visualization, but is available also with *Octahedron* and *Stick* ... This tool in this situation has another specific behavior: While with other transform tools, the "local axes" means the object's axes, here they are the bone's own axes (when you lock to a local axis, by pressing the relevant key twice, the constraint is applied along the selected bone's local axis, not the armature object's axis).

Table 2.46: The same armature in Object Mode and B-Bone visualization, with Bone.004's size scaled up.



Bone Roll

In *Edit Mode*, you can control of the bones roll (i.e. the rotation around the Y axis of the bone).

However, after editing the armature, or when using *euler rotation*, you may want to set the bone roll.

Set Bone Roll

Reference

Mode: Edit Mode
Menu: *Armature* → *Bone Roll* → *Set*
Hotkey: `Ctrl-R`

This is a transform mode where you can edit the roll of all selected bones.

Recalculate Bone Roll

Reference

Mode: Edit Mode
Menu: *Armature* → *Bone Roll* → *Recalculate*
Hotkey: `Ctrl-N`

Axis Orientation

Local Tangent Align roll relative to the axis defined by the bone and its parent.

X, Z

Global Axis Align roll to global X, Y, Z axis.

X, Y, Z

Active Bone Follow the rotation of the active bone.

View Axis Set the roll to align with the view-port.

Cursor Set the roll towards the 3D cursor.

Flip Axis Reverse the axis direction.

Shortest Rotation Avoids rolling the bone over 90 degrees from its current value.

Bone Direction

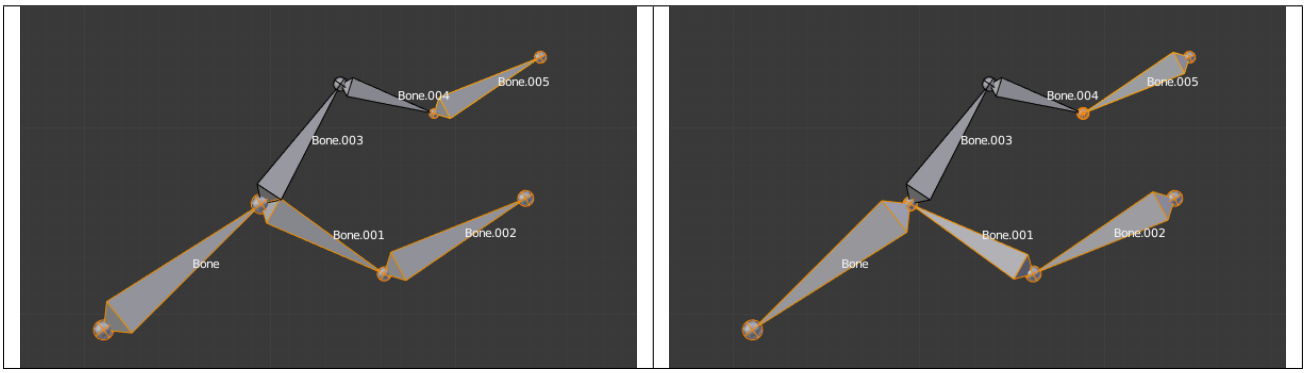
Reference

Mode: Edit Mode
Menu: *Specials* → *Switch Direction*
Hotkey: `W-3`

This tool is not available from the *Armature* menu, but only from the *Specials* pop-up menu `W`. It allows you to switch the direction of the selected bones (i.e. their root will become their tip, and vice versa).

Switching the direction of a bone will generally break the chain(s) it belongs to. However, if you switch a whole (part of a) chain, the switched bones will still be parented/connected, but in "reversed order". See the Fig. *Switching example.*.

Table 2.47: The selected bones have been switched. Bone.005 is no more connected nor parented to anything. The chain of switched bones still exists, but reversed (Now Bone.002 is its root, and Bone is its tip). Bone.003 is now a free bone.



Editing Bones

Reference

Mode: Edit Mode
Hotkey: `Tab`

You will learn here how to add (*Adding Bones*), delete (*Deleting Bones*) or subdivide (*Subdividing Bones*) bones. We will also see how to prevent any bone transformation (*Locking Bones*) in *Edit Mode*, and the option that features an automatic mirroring (*X-Axis Mirror Editing*) of editing actions along the X axis.

Adding Bones

To add bones to your armature, you have more or less the same options as when editing meshes:

- *Add* menu,
- extrusion,
- `Ctrl-LMB` clicks,
- fill between joints,
- duplication.

Add Menu

Reference

Mode: Edit Mode
Hotkey: `Shift-A`

In the 3D View, `Shift-A` → *Bone* to add a new bone to your armature.

This bone will be:

- of one Blender Unit of length,
- oriented towards the positive Y axis of the view,
- with its root placed at the 3D cursor position,
- with no relationship with any other bone of the armature.

Extrusion

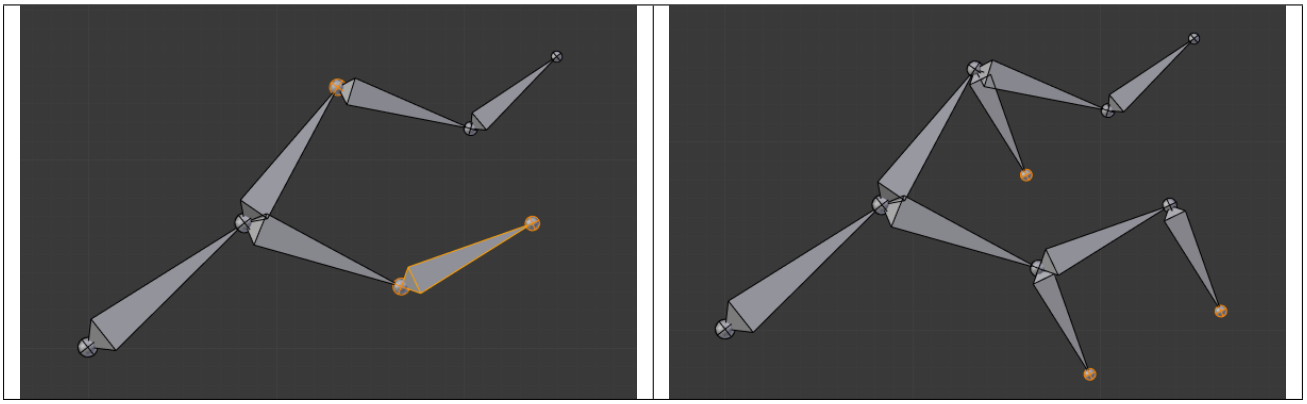
Reference

Mode: Edit Mode

Menu: *Armature* → *Extrude*
Hotkey: `E`, `Shift-E`

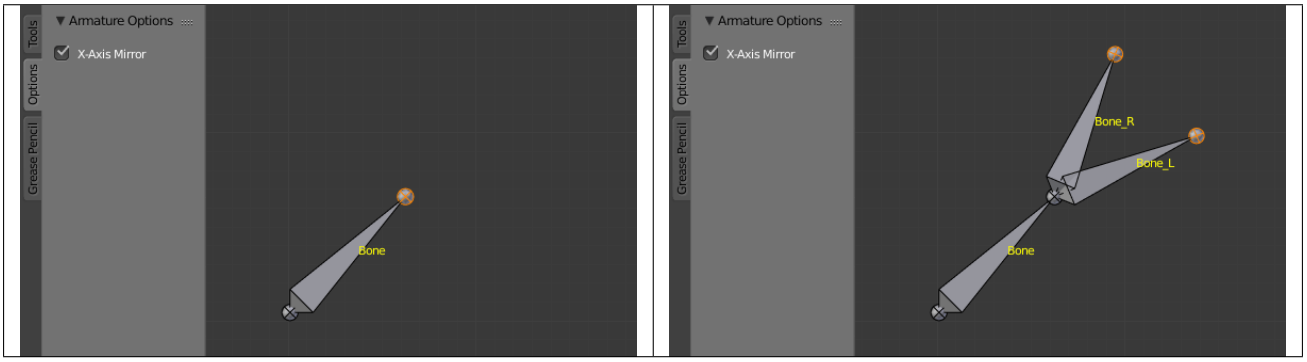
When you press `E`, for each selected tip (either explicitly or implicitly), a new bone is created. This bone will be the child of "its" tip owner, and connected to it. As usual, once extrusion is done, only the new bones' tips are selected, and in grab mode, so you can place them to your liking. See Fig. *Extrusion example.*.

Table 2.48: The three extruded bones.



You also can use the rotating/scaling extrusions, as with meshes, by pressing respectively `E-R` and `E-S` – as well as *locked* extrusion along a global or local axis.

Table 2.49: The two mirror-extruded bones.



Bones have an extra "mirror extruding" tool, called by pressing `Shift-E`. By default, it behaves exactly like the standard extrusion. But once you have enabled the X-Axis mirror editing option (see *X-Axis Mirror Editing*), each extruded tip will produce *two new bones*, having the same name except for the "_L"/ "_R" suffix (for left/right, see the *next page*). The "_L" bone behaves like the single one produced by the default extrusion – you can grab/rotate/scale it exactly the same way. The "_R" bone is its mirror counterpart (along the armature's local X axis), see Fig. *Mirror extrusion example.*.

Warning: Canceling the extrude action causes the newly created bones to snap back to the source position, (creating zero length bones). These will be removed when exiting Edit Mode, however, they can cause confusion and it's unlikely you want to keep them. If you realize the problem immediately undo the extrude action.

In case you are wondering, you cannot just press `X` to solve this as you would in mesh editing, because extrusion selects the newly created tips, and as explained below the delete command ignores bones' ends. To get rid of these extruded bones without undoing, you would have to move the tips, then select the bones and delete (*Deleting Bones*) them.

Mouse Clicks

Reference

Mode: Edit Mode
Hotkey: `Ctrl-LMB`

If at least one bone is selected, `Ctrl-LMB` -clicking adds a new bone.

About the new bone's tip:

- after you `Ctrl-LMB` -clicked it becomes the active element in the armature,
- it appears to be right where you clicked, but...
- ...(as in mesh editing) it will be on the plane parallel to the view and passing through the 3D cursor.

The position of the root and the parenting of the new bone depends on the active element:

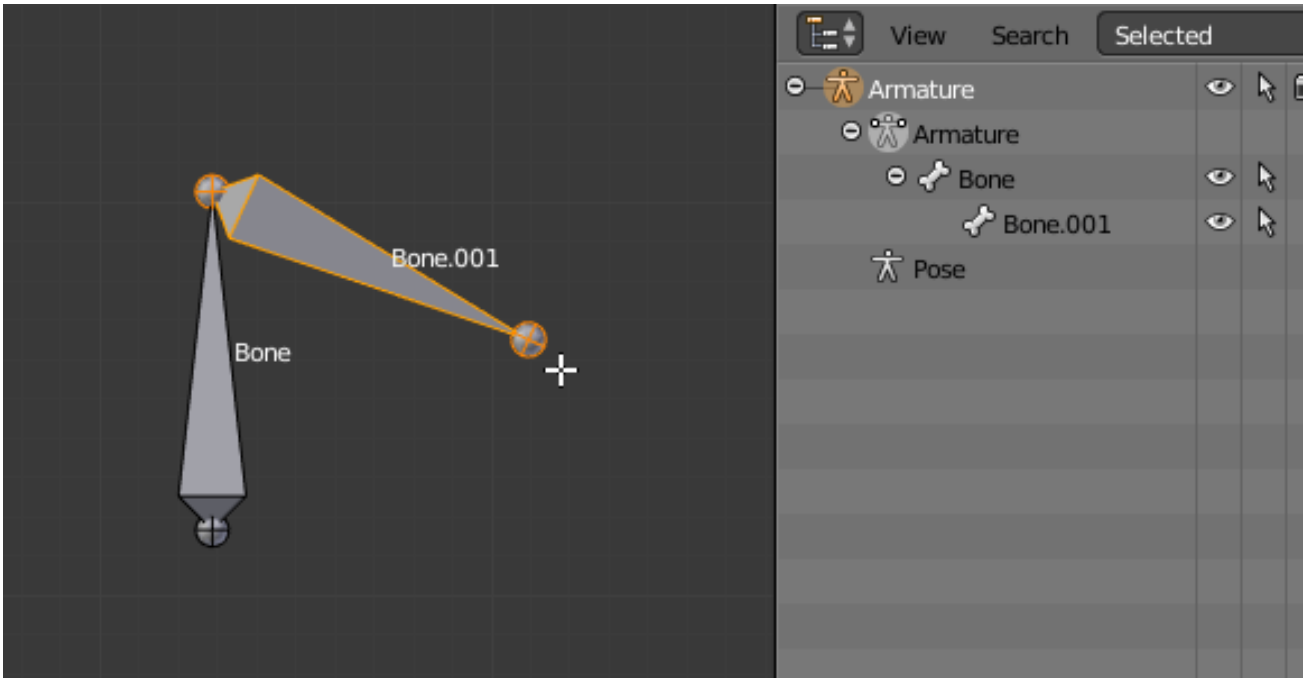


Fig. 2.1077: Ctrl-clicking when the active element is a bone.

If the active element is a *bone*

- the new bone's root is placed on the active bone's tip
- the new bone is parented and connected to the active bone (check the outliner in Fig. *Ctrl-clicking when the active element is a tip.*).

If the active element is a *tip* :

- the new bone's root is placed on the active tip
- the new bone is parented and connected to the bone owning the active tip (check the outliner in Fig. *Ctrl-clicking when the active element is a tip.*).

If the active element is a *disconnected root* :

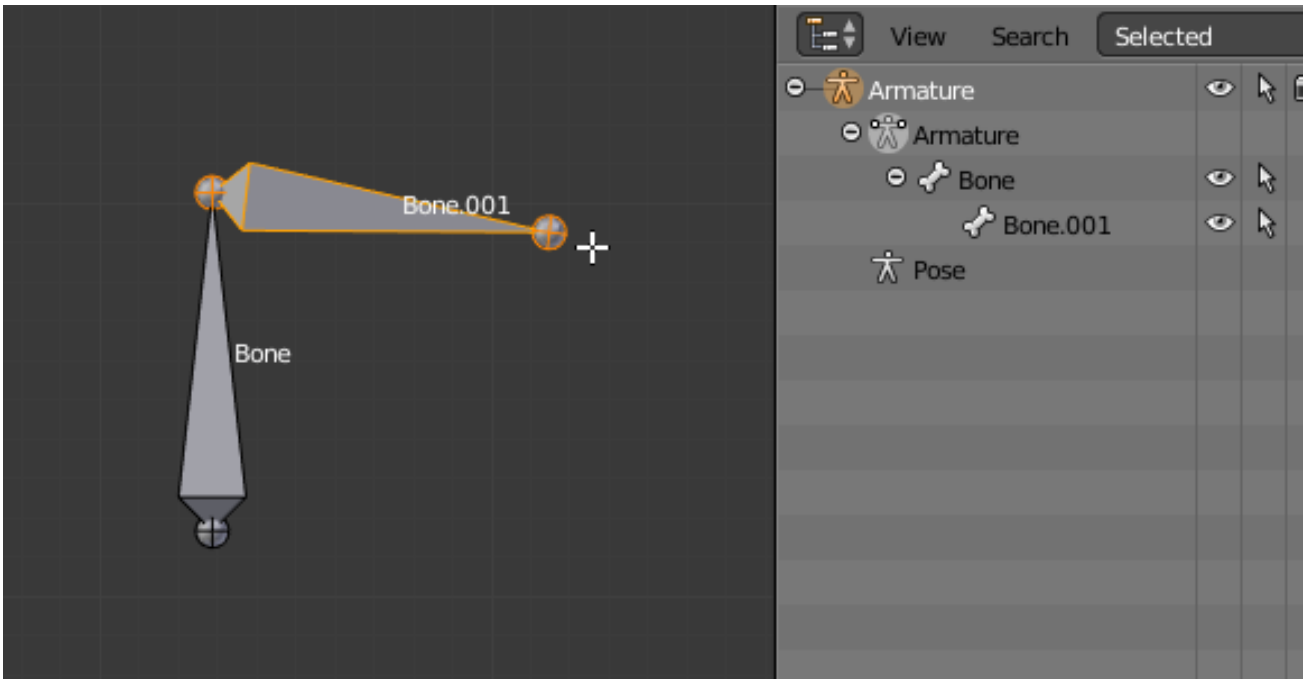


Fig. 2.1078: Ctrl-clicking when the active element is a tip.

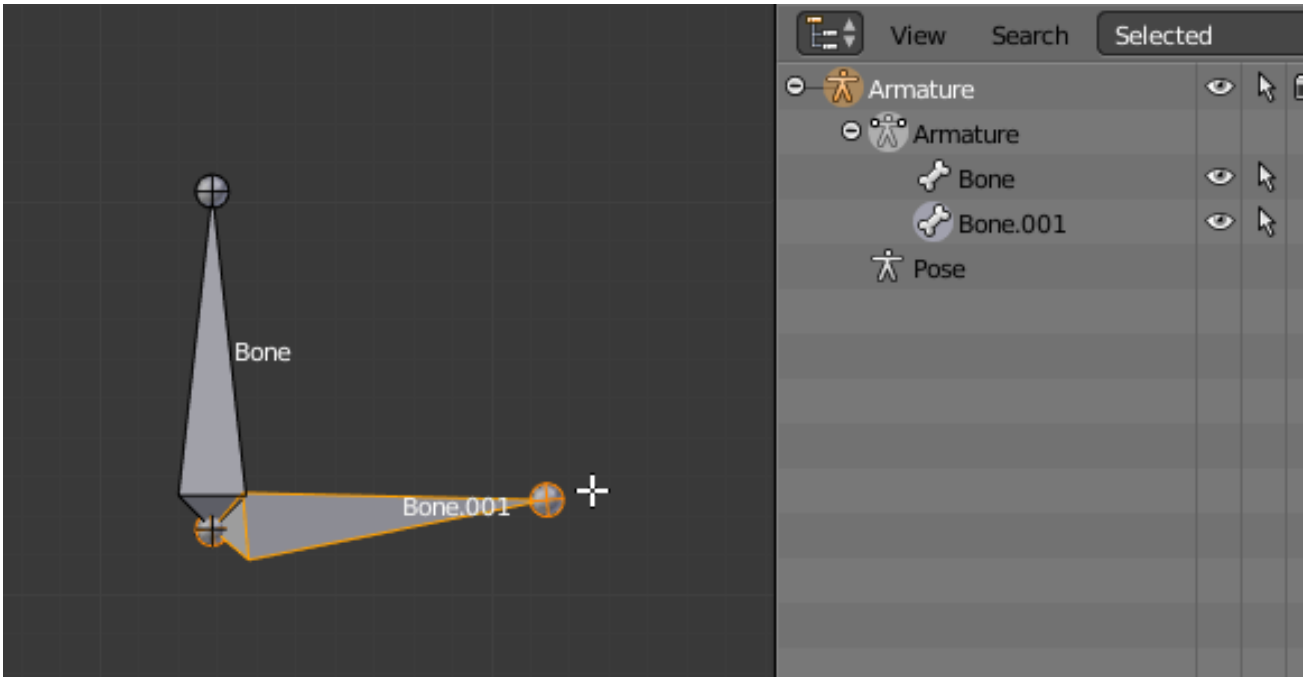


Fig. 2.1079: Ctrl-clicking when the active element is a disconnected root.

- the new bone's root is placed on the active root
- the new bone is **not** parented to the bone owning the active root (check the outliner in Fig. *Ctrl-clicking when the active element is a disconnected root.*).

And hence the new bone will **not** be connected to any bone.

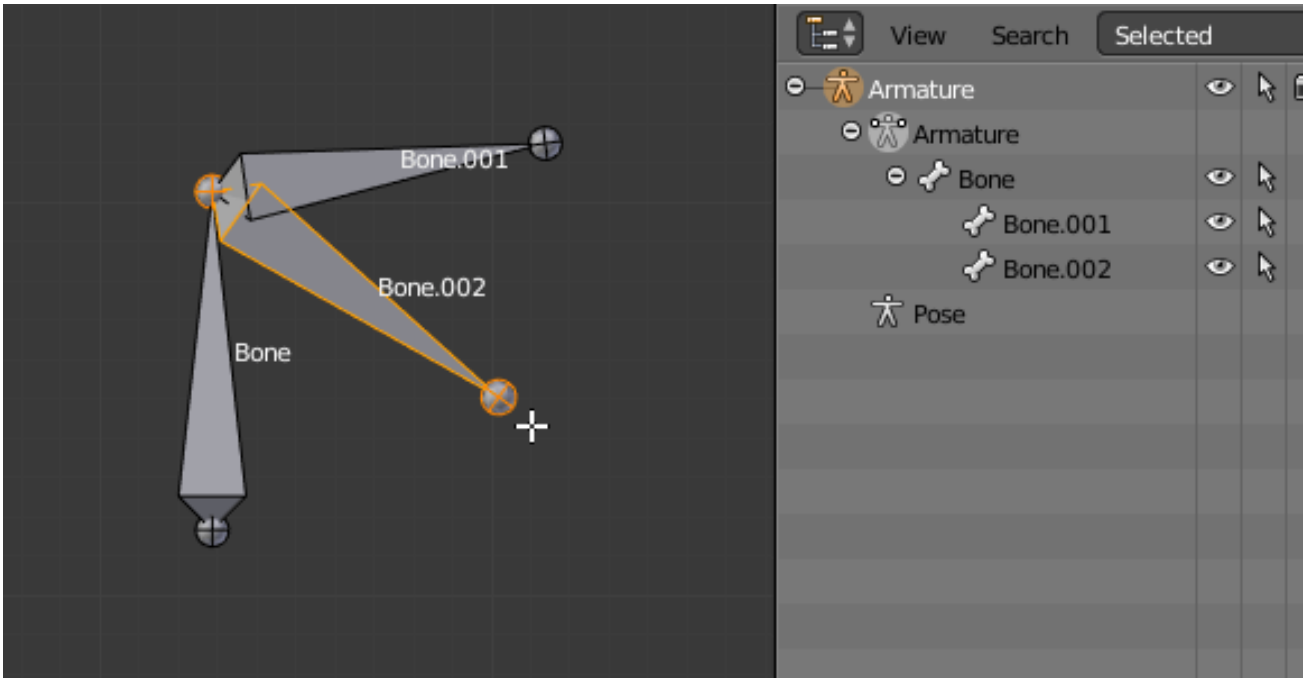


Fig. 2.1080: Ctrl-clicking when the active element is a connected root.

If the active element is a *connected root* :

- the new bone's root is placed on the active root
- the new bone **is** parented and connected to the parent of the bone owning the active root (check the outliner in Fig. *Ctrl-clicking when the active element is a connected root.*).

This should be obvious because if the active element is a connected root then the active element is also the tip of the parent bone, so it is the same as the second case.

As the tip of the new bone becomes the active element, you can repeat these `Ctrl-RMB` several times, to consecutively add several bones to the end of the same chain.

Fill between joints

Reference

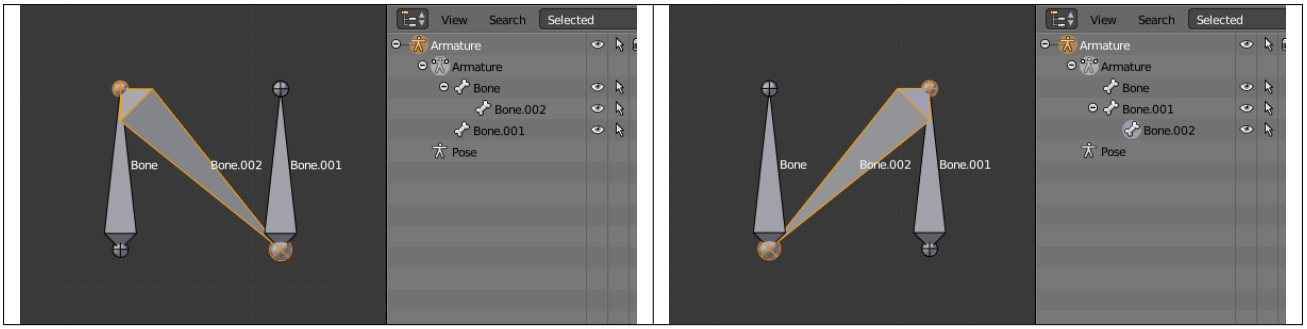
Mode: Edit Mode
Menu: *Armature* → *Fill Between Joints*
Hotkey: `F`

The main use of this tool is to create one bone between two selected ends by pressing `F`, similar to how in mesh editing you can "create edges/faces".

If you have one root and one tip selected, the new bone:

- Will have the root placed on the selected tip.
- Will have the tip placed on the selected root.
- Will be parented and connected to the bone owning the selected tip.

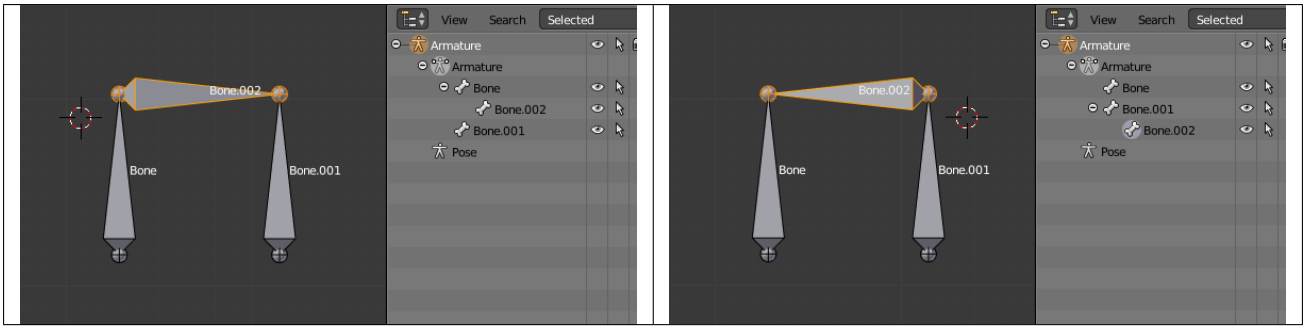
Table 2.50: Active tip on the right.



If you have two tips selected, the new bone:

- Will have the root placed on the selected tip closest to the 3D cursor.
- Will have the tip placed on the other selected tip.
- Will be parented and connected to the bone owning the tip used as the new bone's root.

Table 2.51: 3D cursor on the right.



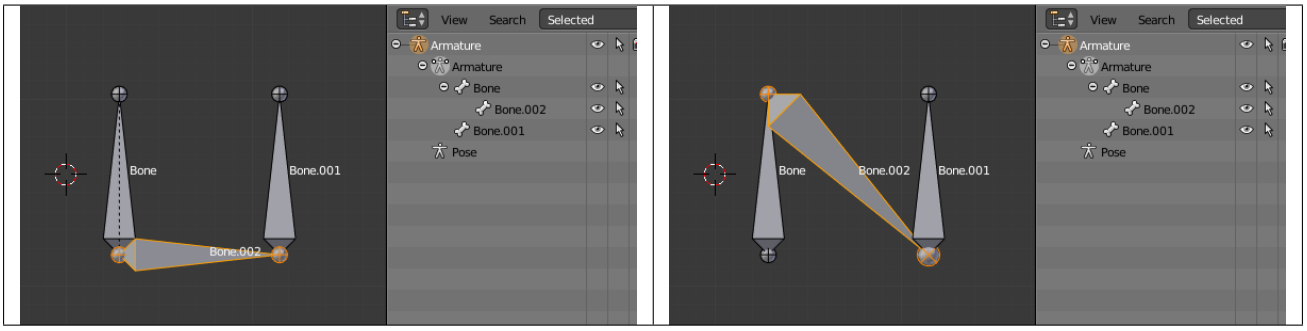
If you have two roots selected, you will face a small problem due to the event system in Blender not updating the interface in real time.

When clicking `F`, similar to the previous case, you will see a new bone:

- With the root placed on the selected root closest to the 3D cursor.
- With the tip placed on the other selected root.
- Parented and connected to the bone owning the root used as the new bone's root.

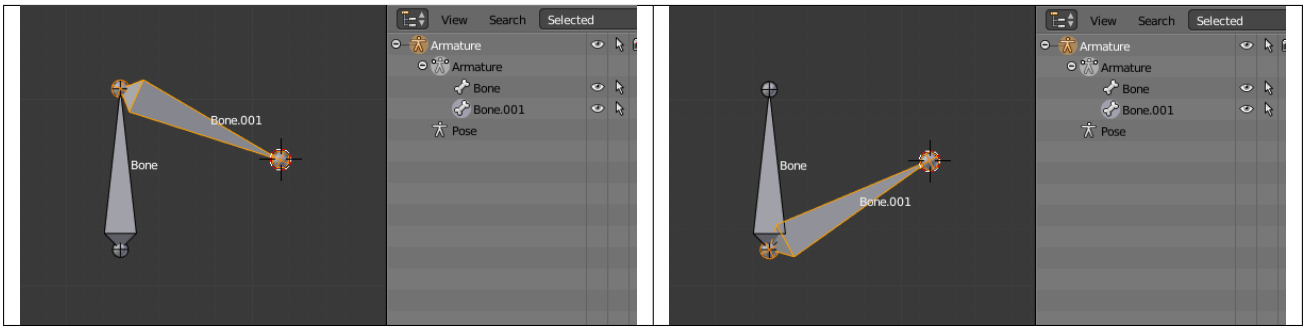
If you try to move the new bone, Blender will update the interface and you will see that the new bone's root moves to the tip of the parent bone.

Table 2.52: After UI update, correct visualization.



Clicking `F` with only one bone end selected will create a bone from the selected end to the 3D cursor position, and it will not parent it to any bone in the armature.

Table 2.53: Fill with only one root selected.



You will get an error when:

- trying to fill two ends of the same bone, or
- trying to fill more than two bone ends.

Duplication

Reference

Mode: Edit Mode
Menu: *Armature* → *Duplicate*
Hotkey: `Shift-D`

Note: This tool works on selected bones; selected ends are ignored.

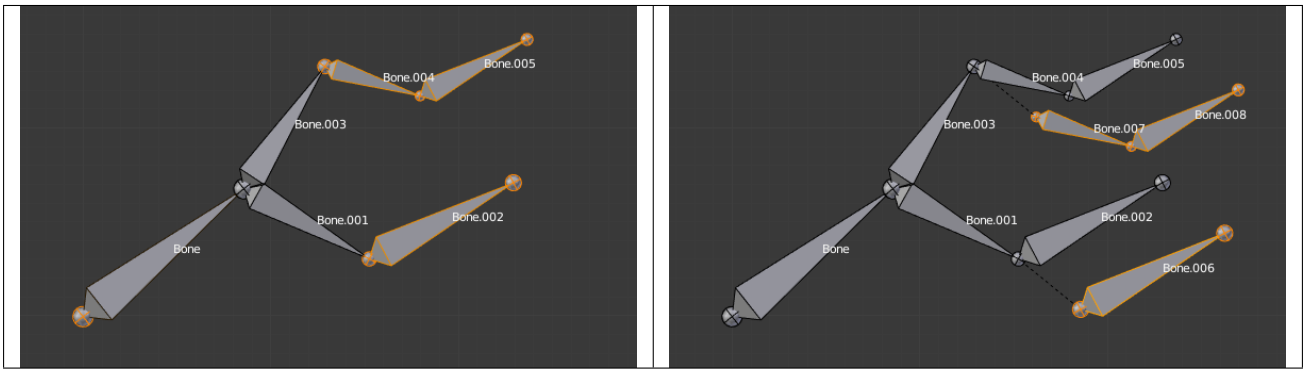
As in mesh editing, by pressing `Shift-D`:

- the selected bones will be duplicated,
- the duplicates become the selected elements and they are placed in grab mode, so you can move them wherever you like.

If you select part of a chain, by duplicating it you will get a copy of the selected chain, so the copied bones are interconnected exactly like the original ones.

The duplicate of a bone which is parented to another bone will also be parented to the same bone, even if the root bone is not selected for the duplication. Be aware, though, that if a bone is parented **and** connected to an unselected bone, its copy will be parented, but **not** connected to the unselected bone (see Fig. *Duplication example.*).

Table 2.54: The three duplicated bones. Note that the selected chain is preserved in the copy, and that Bone.006 is parented but not connected to Bone.001, as indicated by the black dashed line. Similarly, Bone.007 is parented but not connected to Bone.003.



Deleting Bones

You have two ways to remove bones from an armature: the standard deletion, and merging several bones in one.

Standard deletion

Reference

Mode: Edit Mode
Menu: *Armature* → *Delete*
Hotkey: X

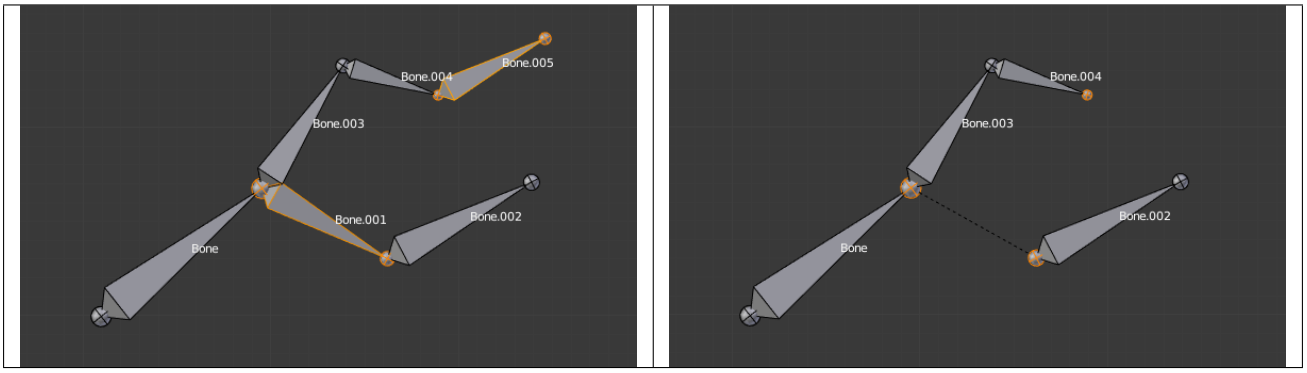
Note: This tool works on selected bones: selected ends are ignored.

To delete a bone, you can:

- press X and confirm, or
- use the menu *Armature* → *Delete* and confirm.

If you delete a bone in a chain, its child(ren) will be automatically re-parented to its own parent, but **not** connected, to avoid deforming the whole armature.

Table 2.55: The two bones have been deleted. Note that Bone.002, previously connected to the deleted Bone.001, is now parented but not connected to Bone.



Merge

Reference

Mode: Edit Mode
Menu: *Armature* → *Merge*
Hotkey: `Alt-M`

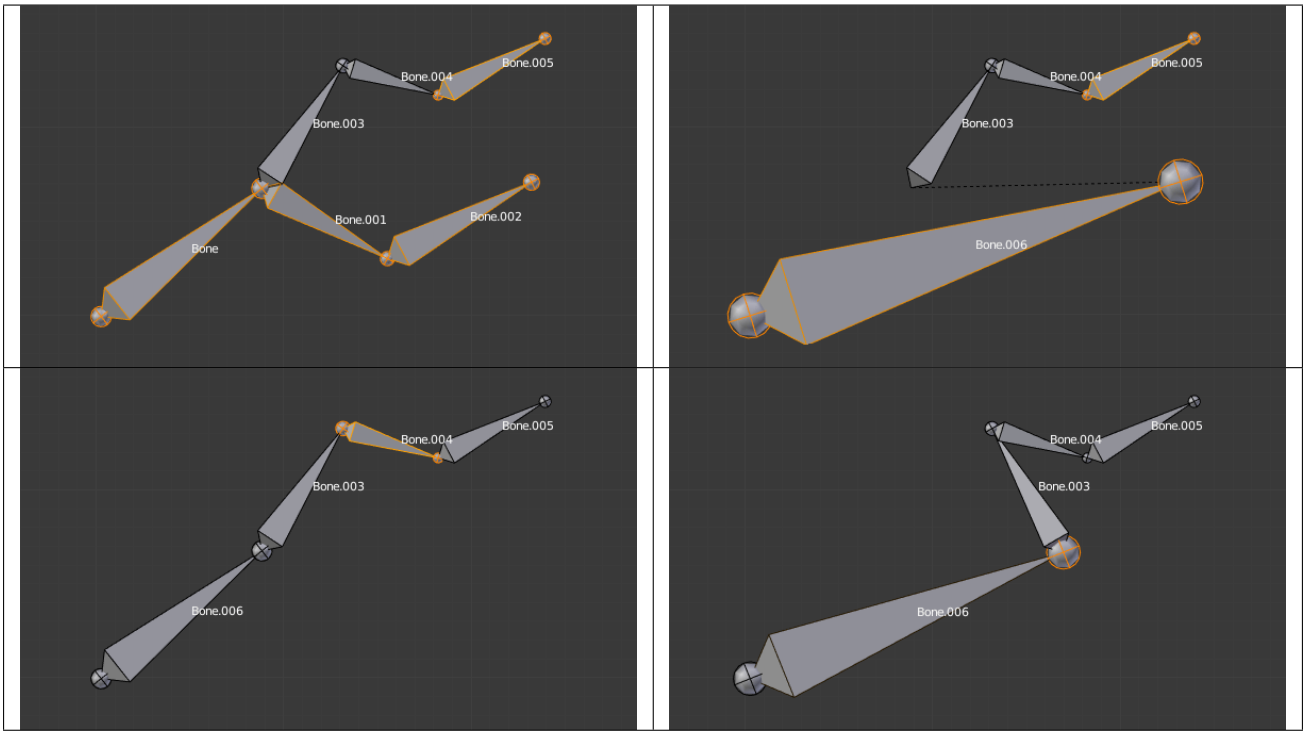
You can merge together several selected bones, as long as they form a chain. Each sub-chain formed by the selected bones will give one bone, whose root will be the root of the root bone, and whose tip will be the tip of the tip bone.

Confirm by clicking on *Merge Selected Bones* → *Within Chains*.

If another (non-selected) chain origins from inside of the merged chain of bones, it will be parented to the resultant merged bone. If they were connected, it will be connected to the new bone.

Here is a strange subtlety (see Fig. *Merge example.*): even though connected (the root bone of the unmerged chain has no root sphere), the bones are not visually connected. This will be done as soon as you edit one bone, differently depending in which chain is the edited bone (compare the bottom two images of the example to understand this better).

Table 2.56: The tip of Bone.006 has been translated, and hence the root of Bone.003 was moved to the tip of "Bone.006"



Subdividing Bones

Reference

Mode: Edit Mode
Menu: *Armature* → *Subdivide*
Hotkey: `W 1`

You can subdivide bones, to get two or more bones where there was just one bone. The tool will subdivide all selected bones, preserving the existing relationships: the bones created from a subdivision always form a connected chain of bones.

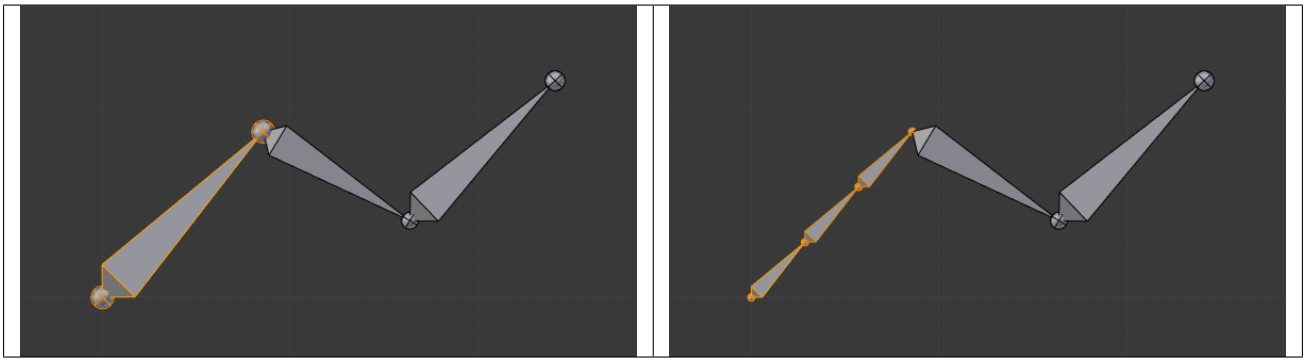
To create two bones out of each selected bone:

- Press `W` → *Subdivide*, same as `W-1`, or
- select *Armature* → *Subdivide* from the header menu.

To create an arbitrary number of bones from each selected bone in the Subdivide Multi Operator panel.

Number of Cuts Specifies the number of cuts. As in mesh editing, if you set *n* cuts, you will get *n* + 1 bones for each selected bone.

Table 2.57: The selected bone has been "cut" two times, giving three sub-bones.



Locking Bones

You can prevent a bone from being transformed in *Edit Mode* in several ways:

- All bones can be locked clicking on the *Lock* checkbox of their Transform panel in the *Bones* tab;
- Press `Shift-W` → *Toggle Settings* → *Locked*
- Select *Armature* → *Bone Settings* → *Toggle a Setting*).

If the root of a locked bone is connected to the tip of an unlocked bone, it will not be locked, i.e. you will be able to move it to your liking. This means that in a chain of connected bones, when you lock one bone, you only really lock its tip. With unconnected bones, the locking is effective on both ends of the bone.

X-Axis Mirror Editing

Another very useful tool is the *X-Axis Mirror* editing option by *Tool panel* → *Armature Options*, while Armature is selected in *Edit Mode*. When you have pairs of bones of the same name with just a different "side suffix" (e.g. ".R"/".L", or "_right"/"_left" ...), once this option is enabled, each time you transform (move/rotate/scale...) a bone, its "other side" counterpart will be transformed accordingly, through a symmetry along the armature local X axis. As most rigs have at least one axis of symmetry (animals, humans, ...), it is an easy way to spare you half of the editing work!

See also:

naming bones.

Separating Bones in a new Armature

You can, as with meshes, separate the selected bones in a new armature object *Armature* → *Separate*, `Ctrl-Alt-P` and of course, in *Object Mode*, you can join all selected armatures in one *Object* → *Join Objects*, `Ctrl-J`.

Naming

Reference

Mode: Edit Mode
Panel: *Properties region* → *Item*, *Bones tab* → *Bones panel*

You can rename your bones, either using the *name* field of the *Item* panel in the 3D Views, for the active bone, or using the *name* field in each bone of the *Bones* tab in *Edit Mode*.

Blender also provides you some tools that take advantage of bones named in a left/right symmetry fashion, and others that automatically name the bones of an armature. Let us look at this in detail.

Naming Conventions

Naming conventions in Blender are not only useful for you in finding the right bone, but also to tell Blender when any two of them are counterparts.

In case your armature can be mirrored in half (i.e. it is bilaterally symmetrical), it is worthwhile to stick to a left/right naming convention. This will enable you to use some tools that will probably save you time and effort (like the *X-Axis Mirror* editing tool we saw above...).

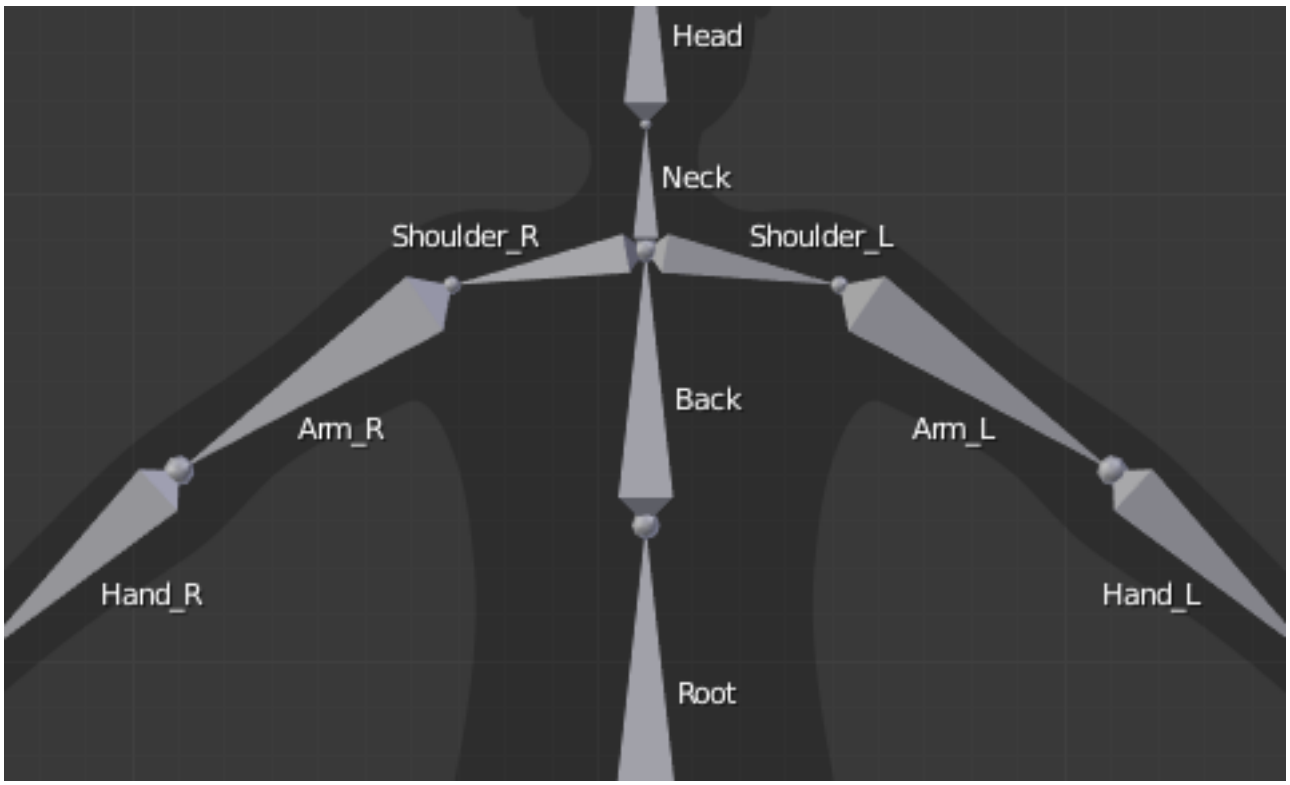


Fig. 2.1099: An example of left/right bone naming in a simple rig.

1. First you should give your bones meaningful base-names, like “leg”, “arm”, “finger”, “back”, “foot”, etc.
2. If you have a bone that has a copy on the other side (a pair), like an arm, give it one of the following separators:
   - Left/right separators can be either the second position “L_calfbone” or last-but-one “calfbone**.**R”
   - If there is a lower or upper case “L”, “R”, “left” or “right”, Blender handles the counterpart correctly. See below for a list of valid separators. Pick one and stick to it as close as possible when rigging; it will pay off.

   Examples of valid saparators:

   - (nothing): handLeft –> handRight
   - _ (underscore): hand_L –> hand_R
   - **.** (dot): hand**.**l –> hand**.**r
   - – (dash): hand**-**l –> hand**-**r
   - ‘‘ ‘‘ (space): hand LEFT –> hand RIGHT

Note: Note that all examples above are also valid with the left/right part placed before the name. You can only use the short "L"/ "R" code if you use a separator (i.e. "handL"/ "handR" will not work!).

3. Before Blender handles an armature for mirroring or flipping, it first removes the number extension, e.g. ".001".
4. You can copy a bone named "bla.L" and flip it over using `W` → *Flip Left-Right Names*. Blender will name the copy "bla.L.001" and flipping the name will give you "bla.R".

Bone Name Flipping

Reference

Mode: Edit Mode
Menu: *Armature* → *Flip Left & Right Names*
Hotkey: `W-4`

You can flip left/right markers (see above) in selected bone names, using either *Armature* → *Flip Left & Right Names*, or *Specials* → *Flip Left-Right Names*, `W-4`. This can be useful if you have constructed half of a symmetrical rig (marked for a left or right side) and duplicated and mirrored it, and want to update the names for the new side. Blender will swap text in bone names according to the above naming conventions, and remove number extensions if possible.

Auto Bone Naming

Reference

Mode: Edit Mode
Menu: *Armature* → *AutoName Left-Right*, *Armature* → *AutoName Front-Back*, *Armature* → *AutoName Top-Bottom*
Hotkey: `W-5`, `W-6`, `W-7`

The three *AutoName* entries of the *Armature* and *Specials* `W` menus allows you to automatically add a suffix to all selected bones, based on the position of their root relative to the armature center and its local coordinates :

AutoName Left-Right will add the ".L" suffix to all bones with a *positive* X-coordinate root, and the ".R" suffix to all bones with a *negative* X-coordinate root. If the root is exactly at 0.0 on the X-axis, the X-coordinate of the tip is used. If both ends are at 0.0 on the X-axis, the bone will just get a period suffix, with no "L"/ "R" (as Blender cannot decide whether it is a left or right bone...).

AutoName Front-Back will add the ".Bk" suffix to all bones with a *positive* Y-coordinate root, and the ".Fr" suffix to all bones with a *negative* Y-coordinate root. The same as with *AutoName Left-Right* goes for 0.0 Y-coordinate bones...

AutoName Top-Bottom will add the ".Top" suffix to all bones with a *positive* Z-coordinate root, and the ".Bot" suffix to all bones with a *negative* Z-coordinate root. The same as with *AutoName Left-Right* goes for 0.0 Z-coordinate bones...

Parenting

Reference

Mode: Edit Mode
Panel: Armature
Menu: *Armature* → *Parent* → ...
Hotkey: `Ctrl-P`, `Alt-P`

You can edit the relationships between bones (and hence create/modify the chains of bones) both from the 3D Views and the Properties editor. Whatever method you prefer, it's always a matter of deciding, for each bone, if it has to be parented to another one, and if so, if it should be connected to it.

To parent and/or connect bones, you can:

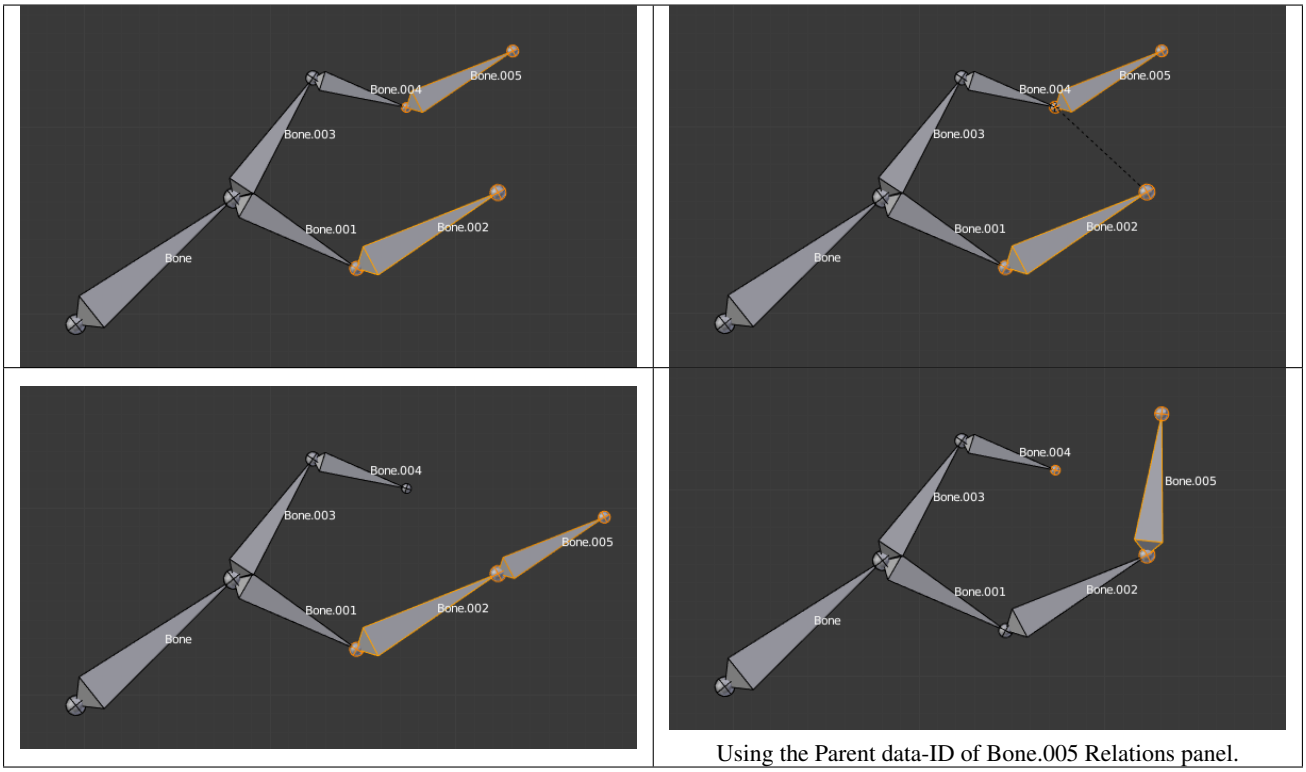
- In a 3D View, select the bone and *then* its future parent, and press `Ctrl-P` (or *Armature* → *Parent* → *Make Parent...*). In the small *Make Parent* menu that pops up, choose *Connected* if you want the child to be connected to its parent, else click on *Keep Offset*. If you have selected more than two bones, they will all be parented to the last selected one. If you only select one already-parented bone, or all selected bones are already parented to the last selected one, your only choice is to connect them, if not already done. If you select only one non-parented bone, you will get the *Need selected bone(s)* error message...

  Note: With this method, the newly-children bones will not be scaled nor rotated – they will just be translated if you chose to connect them to their parent's tip.

- In the Properties editor, *Bones* tab, for each selected bone, you can select its parent in the *Parent* data-ID to the upper right corner of its Relations panel. If you want them to be connected, just enable the checkbox to the right of the list.

  Note: With this method, the tip of the child bone will never be translated – so if *Connected* is enabled, the child bone will be completely transformed by the operation.

Table 2.58: Bone.005 parented and connected to Bone.002.



Using the Parent data-ID of Bone.005 Relations panel.

To disconnect and/or free bones, you can:

- In a 3D View, select the desired bones, and press `Alt-P` (or *Armature* → *Parent* → *Clear Parent...*). In the small *Clear Parent* menu that pops up, choose *Clear Parent* to completely free all selected bones, or *Disconnect Bone* if you just want to break their connections.
- In the Properties editor, *Bones* tab, for each selected bone, you can select no parent in the *Parent* data-ID of its Relations panel, to free it completely. If you just want to disconnect it from its parent, disable the *Connected* checkbox.

Note that relationships with non-selected children are never modified.

Properties

Reference

Mode: Edit Mode
Menu: *Armature* → *Bone Settings* → ...
Hotkey: `Shift-W`, `Ctrl-Shift-W`, `Alt-W`

Most bones' properties (excepted the transform ones) are regrouped in each bone's panels, in the *Bones* tab in *Edit Mode*. Let us detail them.

Note that some of them are also available in the 3D Views, through the three pop-up menus within the same entry:

- *Toggle Setting*: `Shift-W` or *Armature* → *Bone Settings* → *Toggle a Setting*
- *Enable Setting*: `Ctrl-Shift-W` or *Armature* → *Bone Settings* → *Enable a Setting*

- *Disable Setting*: `Alt-W` or *Armature* → *Bone Settings* → *Disable a Setting*

Draw Wire

Deform (also `Shift-W` → *(Deform, ...)*),

Multiply Vertex Group by Envelope (also `Shift-W` → *(Multiply Vertex Group by Envelope, ...)*)

These settings control how the bone influences its geometry, along with the bones' ends radius. This will be detailed in the *skinning part*.

Inherit Rotation These settings affect the behavior of children bones while transforming their parent in *Pose Mode*, so this will be detailed in the *posing part* !

Inherit Scale

Lock (also `Shift-W` → *(Locked, ...)*) This will prevent all editing of the bone in *Edit Mode*; see *previous page*.

Skeleton Sketching

If you think that creating a whole rig by hand, bone after bone, is quite boring, be happy: some Blender developers had the same feeling, and created the Skeleton Sketching tool, formerly the Etch-a-ton tool, which basically allows you to "draw" (sketch) whole chains of bones at once.

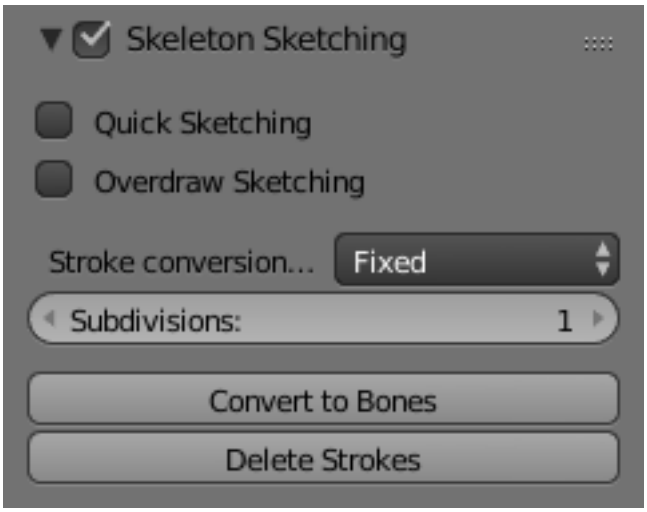


Fig. 2.1104: The Bone Sketching panel.

Skeleton Sketching is obviously only available in *Edit Mode*, in the 3D Views. You control it through its *Skeleton Sketching* panel in the *Transform panel*, which you can open with `N`. Use the mouse `LMB` to draw strokes, and `RMB` for gestures. Showing its tool panel will not enable sketching. You must tick the checkbox next to *Skeleton Sketching* to start drawing bone chains (otherwise, you remain in the standard *Edit Mode*...).

Sketching is done in two steps:

- *Drawing Chains* (called "strokes"). Each stroke corresponds to a chain of bones.
- *Converting to Bones*, using different methods.

The *point of view* is important, as it determines the future bones' roll angle: the Z axis of a future bone will be aligned with the view Z axis of the 3D View in which you draw its "parent" stroke (unless you use the *Template* converting method...). Strokes are drawn in the current view plane passing through the 3D cursor, but you can create somewhat "3D" strokes using the *Adjust* drawing option in different views (see below).

If you enable the small *Quick Sketch* option, the two steps are merged into one: once you have finalized the drawing of a stroke (see *Drawing Chains*), it is immediately converted to bones (using the current active method) and deleted. This option makes bone sketching quick and efficient, but you lose all the advanced stroke editing possibilities.

Sketches are **not** saved into blend-files, so you cannot interrupt a sketching session without losing all your work! Note also that the sketching is common to the whole Blender session, i.e. there is only one set of strokes (one sketch) in Blender, and not one per armature, or even per file...

Drawing Chains

So, each stroke you draw will be a chain of bones, oriented from the starting point (the reddest or most orange part of the stroke) to its end (its whitest part). A stroke is made of several segments, delimited by small black dots. There will be at least one bone per segment (except with the *Template* conversion method, see *next page*), so all black points represents future bones' ends. There are two types of segments, which can be mixed together:

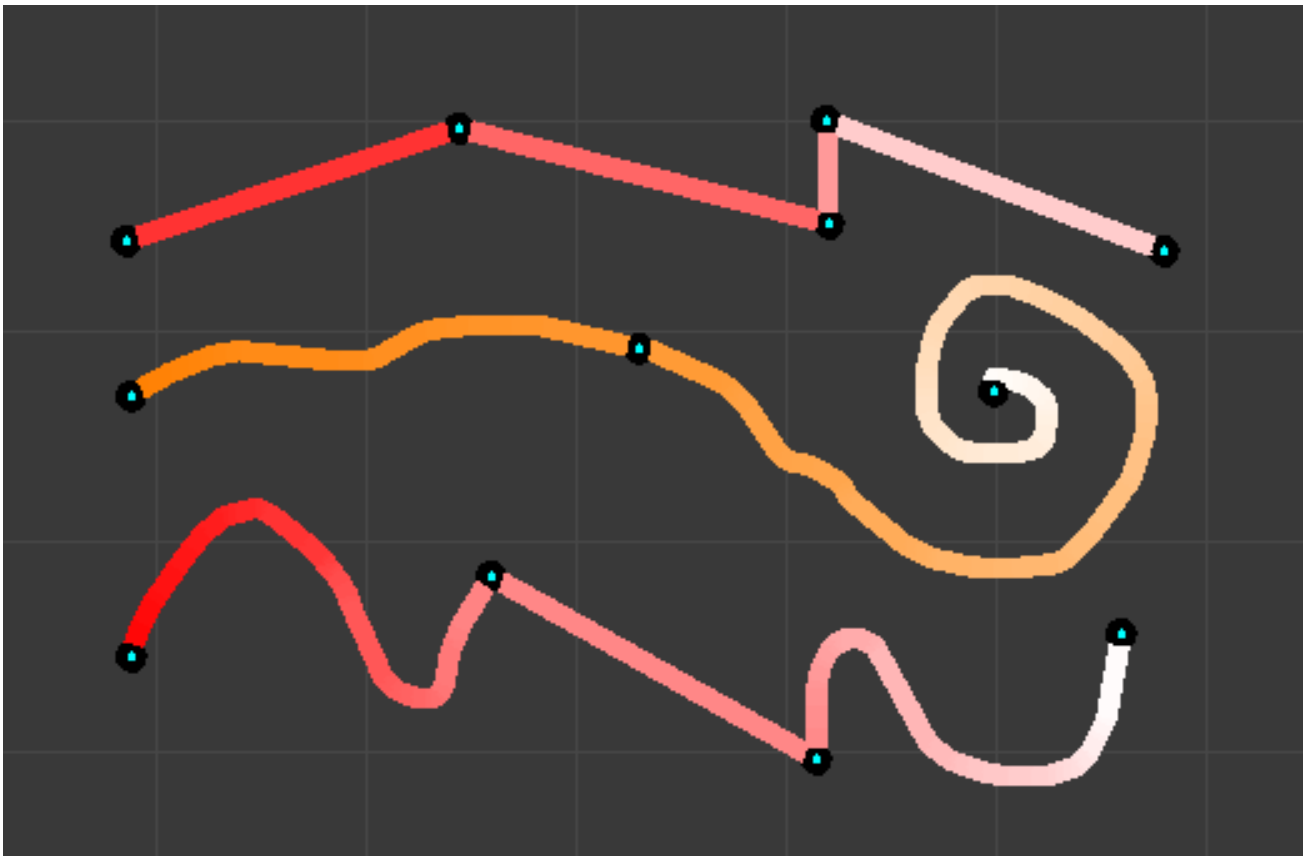
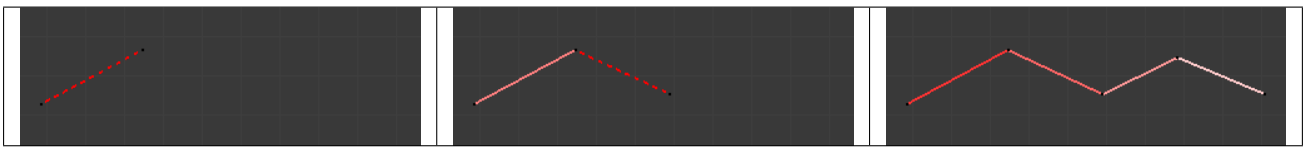


Fig. 2.1105: Strokes example. From top to bottom: A selected polygonal stroke of four straight segments, oriented from left to right. An unselected free stroke of two segments, oriented from left to right. A mixed stroke, with one straight segment between two free ones, right to left.

Straight Segments

To create a straight segment, click LMB at its starting point. Then move the mouse cursor, without pressing any button, a dashed red line represents the future segment. Click LMB again to finalize it. Each straight segment of a stroke will always create one and only one bone, whatever convert algorithm you use (except for the *Template* conversion method).

Table 2.59: Repeating these steps, we now have a four-segment polygonal stroke.



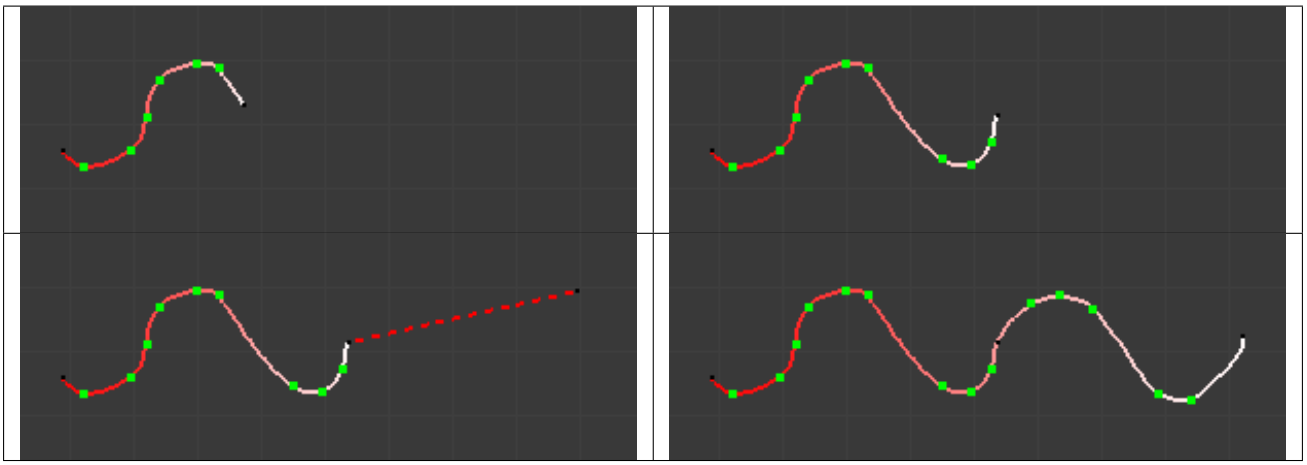
Free Segments

To create a free (curved) segment, click and hold LMB at its starting point. Then draw your segment by moving the mouse cursor – as in any paint program! Release LMB to finalize the segment. You will then be creating a new straight segment, so if you would rather start a new free segment, you must immediately re-press LMB.

The free segments of a stroke will create different number of bones, in different manners, depending on the conversion method used. The future bones' ends for the current selected method are represented by small green dots for each one of those segments, for the selected strokes only.

The free segment drawing uses the same *Manhattan Distance* setting as the *grease pencil tool* (*User Preferences*, *Edit Methods* "panel", *Grease Pencil* group) to control where and when to add a new point to the segment. So if you feel your free segments are too detailed, raise this value a bit, and if you find them too jagged, lower it.

Table 2.60: But if you immediately click again and drag LMB you will instead start a new free segment.



You finalize a whole stroke by clicking RMB. You can cancel the stroke you are drawing by pressing Esc. You can also snap strokes to underlying meshes by holding Ctrl while drawing. By the way, the *Peel Objects* button at the bottom of the *Bone Sketching* panel is the same thing as the "monkey" button of the snapping header controls shown when *Volume* snap element is selected. See the *snap to mesh* page for details.

Selecting Strokes

A stroke can be selected (materialized by a solid red-to-white line), or not (shown as an orange-to-white line) - see (Strokes example) above. As usual, you select a stroke by clicking RMB on it, you add one to/remove one from the current selection with a Shift−RMB click, and A (de)selects all strokes...

Deleting

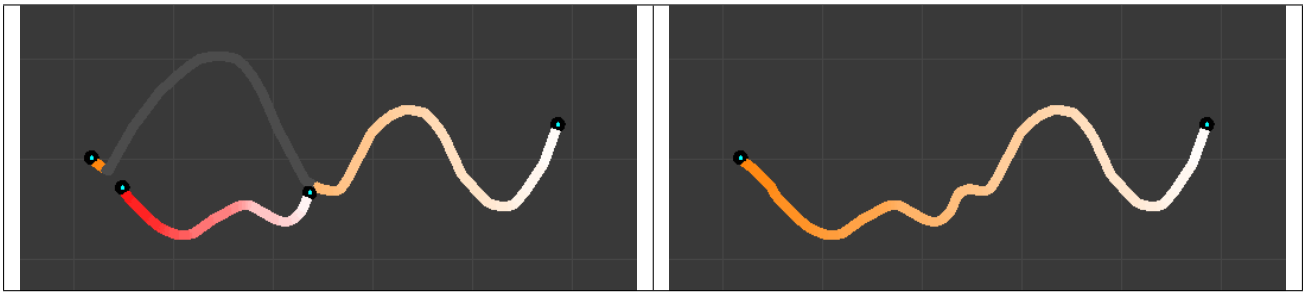
Hitting X or clicking on the *Delete* button (*Bone Sketching* panel) deletes the selected strokes (be careful, no warning/confirmation pop-up menu here). See also *Gestures*.

Modifying Strokes

You can adjust, or "redraw" your strokes by enabling the *Overdraw Sketching* option of the *Bone Sketching* panel. This will modify the behavior of the strokes drawing (i.e. LMB clicks and/or hold): when you draw, you will not create a new stroke, but rather modify the nearest one.

The part of the old stroke that will be replaced by the new one are drawn in gray. This option does not take into account stroke selection, i.e. all strokes can be modified this way, not just the selected ones... Note also that even if it is enabled, when you draw too far away from any other existing stroke, you will not modify any of them, but rather create a new one, as if *Overdraw Sketching* was disabled.

Table 2.61: Stroke adjusted.



Warning: There is no undo/redo for sketch drawing...

Gestures

There quite a few things about strokes editing that are only available through gestures. Gestures are started by clicking and holding Shift-LMB (when you are not already drawing a stroke), and materialized by blue-to-white lines. A gesture can affect several strokes at once.

There is no direct way to cancel a gesture once you have started "drawing" it. So the best thing to do, if you change your mind (or made a "false move"), is to continue to draw until you get a disgusting scribble, crossing your stroke several times. In short, something that the gesture system would never recognize!

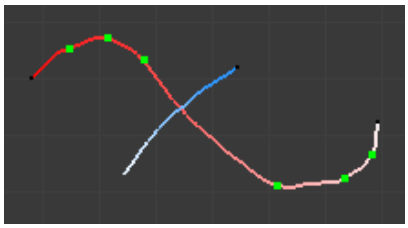


Fig. 2.1115: An unwanted cut stroke.

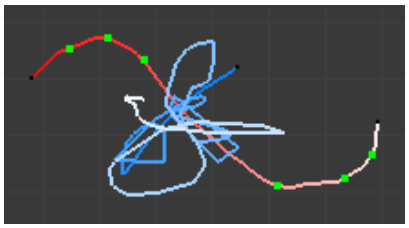


Fig. 2.1116: Some random drawing.

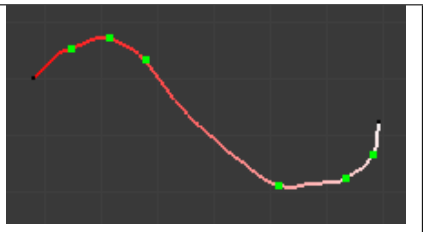


Fig. 2.1117: The stroke is still in one piece.

Cut

To *cut* a segment (i.e. add a new black dot inside it, making two segments out of one), "draw" a straight line crossing the chosen segment where you want to split it.

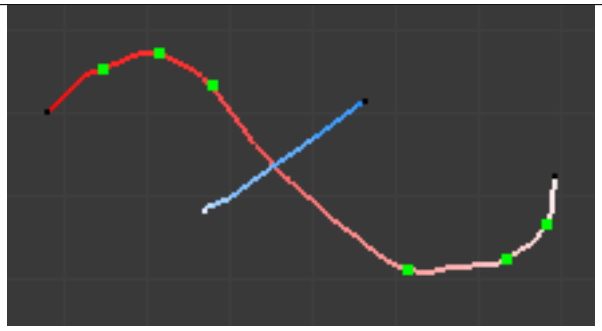


Fig. 2.1118: Gesture.

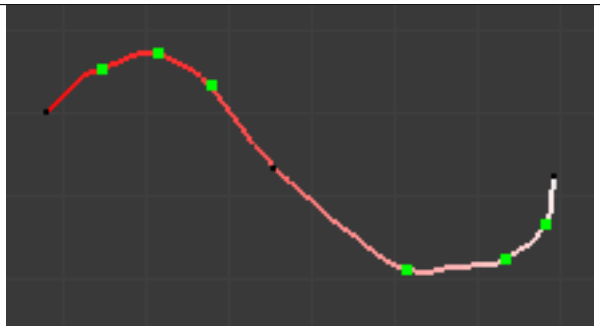


Fig. 2.1119: Result.

Delete

To *delete* a stroke, draw a "V" crossing the stroke to delete twice.

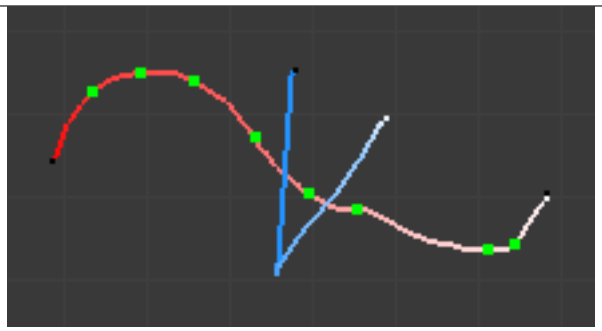


Fig. 2.1120: Gesture.

Fig. 2.1121: Result.

Reverse

To *reverse* a stroke (i.e. the future chain of bones will be reversed), draw a "C" crossing twice the stroke to reverse.

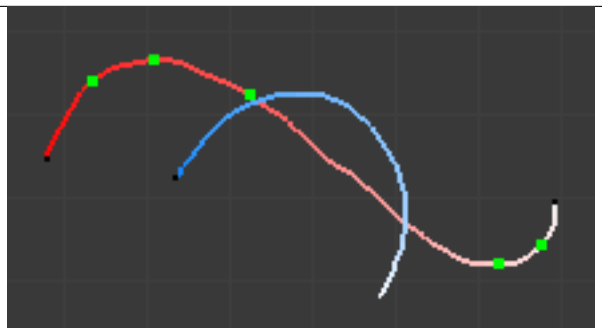


Fig. 2.1122: Gesture.

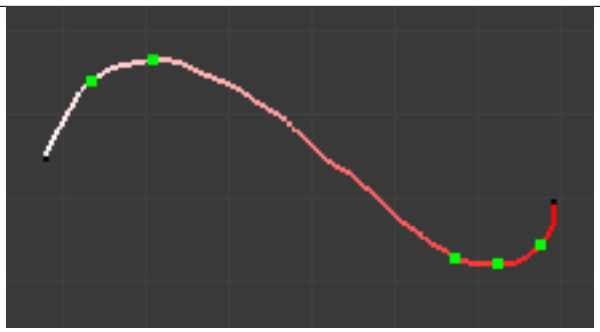


Fig. 2.1123: Result.

Converting to Bones

Once you have one or more selected strokes, you can convert them to bones, using either the *Convert* button of the *Bone Sketching* panel, or the corresponding gesture (see *Gestures*). Each selected stroke will generate a chain of bones, oriented from its reddest end to its whitest one. Note that converting a stroke does not delete it.

There are four different conversion methods with three "simple" ones, and one more advanced and complex, *Template*, that reuses bones from the same armature or from another one as a template for the strokes to convert, and which is detailed

in *the next page*. Anyway, remember that straight segments are always converted to one and only one bone (except for the *Template* conversion method), and that the future bones' ends are shown as green dots on selected free segments.

Remember also that the roll rotation of the created bones has been set during their "parent" stroke drawing (except for the *Template* conversion method) - their Z axis will be aligned with the view Z axis of the active 3D View at draw time.

Fixed

With this method, each free segment of the selected strokes will be uniformly divided in *n* parts (set in *Number* number button), i.e. will give *n* bones.

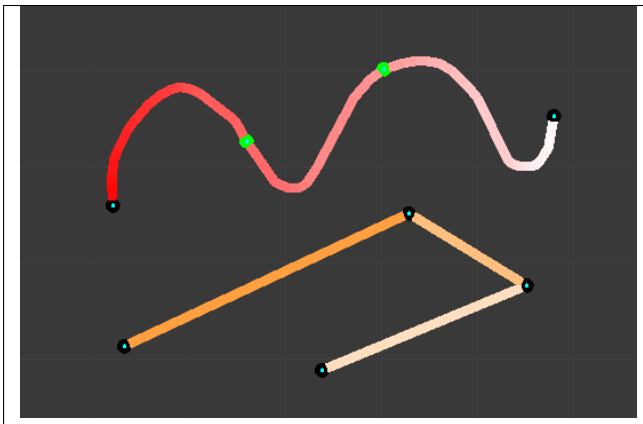


Fig. 2.1124: The Fixed conversion preview on selected strokes.

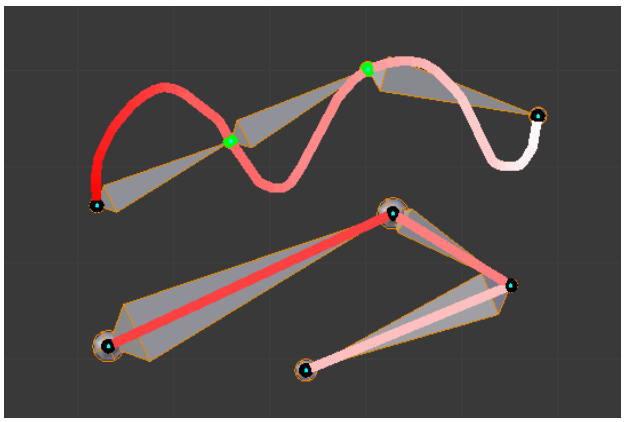


Fig. 2.1125: The Fixed conversion result.

Adaptive

With this method, each free segment of the selected strokes will create as many bones as necessary to follow its shape closely enough. This "closely enough" parameter being set by the *Threshold* number button; higher values giving more bones, following more closely the segments' shape. So the more twisted a free segment, the more bones it will generate.

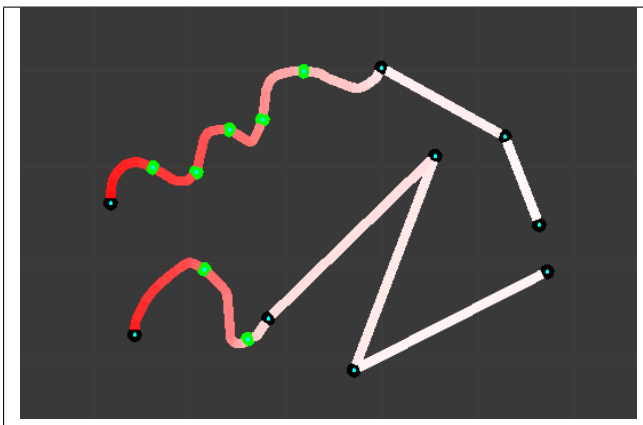


Fig. 2.1126: The Adaptive conversion preview on selected strokes.

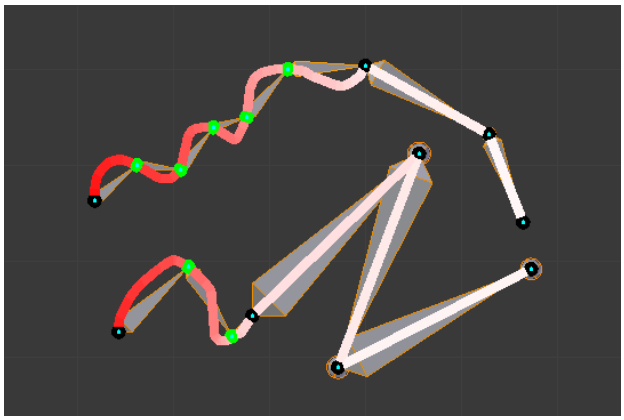


Fig. 2.1127: The Adaptive conversion result.

Length

With this method, each free segment of the selected strokes will create as many bones as necessary, so that none of them is longer than the *Length* number button value (in Blender Units).

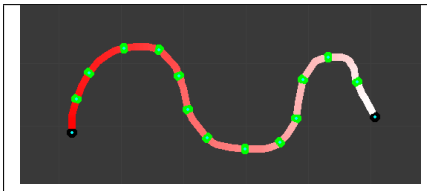


Fig. 2.1128: The Length conversion preview on selected strokes.

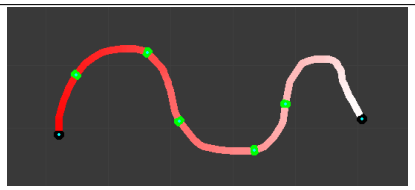


Fig. 2.1129: Using a larger length value.

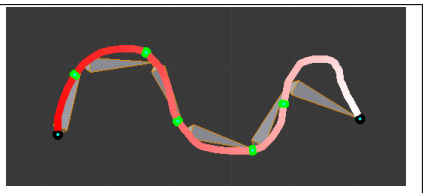


Fig. 2.1130: The Length conversion result.

Retarget

Retarget template bone chain to stroke.

Template Template armature that will be retargeted to the stroke. This is a more complex topic, detailed in its *own page*.

Retarget roll mode

None Do not adjust roll.

View Roll bones to face the view.

Joint Roll bone to original joint plane offset.

Autoname Todo.

Number Todo.

Side Todo.

Armature Templating

The idea of templating is to use an already existing armature as base ("template") to create a new armature. It differs from a simple copy in that you can directly define the new armature different in some aspects than its reference rig.

In Blender, the only templating tool is the bone sketching one (Etch-a-ton, described in *the previous page*), with its *Template* conversion method, so you should have read its page before this one!

Using Bone Sketching

Reference

Mode: Edit Mode
Panel: Bone Sketching (3D View editor)
Menu: *Armature* → *Bone Sketching*
Hotkey: P

The *Template* conversion method of *Bone Sketching* tool maps a copy of existing bones to each selected stroke. The new bones will inherit some of their properties (influence, number of segments, etc.) from the corresponding bones in the template, but they will acquire their lengths, rolls and rotation from the sketch; so these properties would be different as compared to the template.

This is easier to understand with some examples.

In the following image, "armature.002" is set as the template, and the stroke maps with "chain_a" of this template. None of the bones are selected in the template. Note that there is no second stroke to map with chain "chain_b" of the template. The result is shown at right: Blender creates a copy of "chain_a" and matches the bones with the stroke.

Blender also creates a copy of "chain_b", but this chain is not altered in any way; because this command can map only one selected chain with a stroke.

In the following example, no template is selected. (In other words, all the action is within the armature itself.)

Two bones are selected in "chain_b", and the property panel is set to map the joints with the stroke. So these two selected bones are copied and the newly created copy of the chain is matched with the stroke. (Note that the newly created bones are named in continuation of the original chain.)

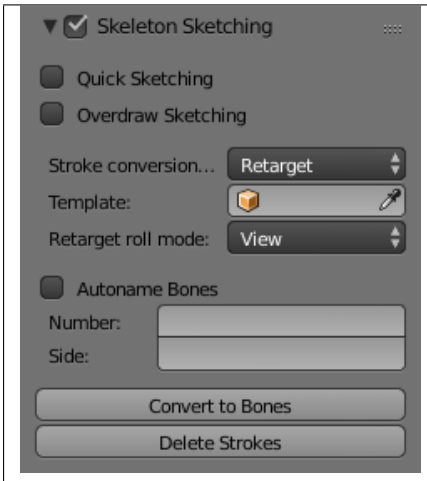


Fig. 2.1131: Conversion settings.

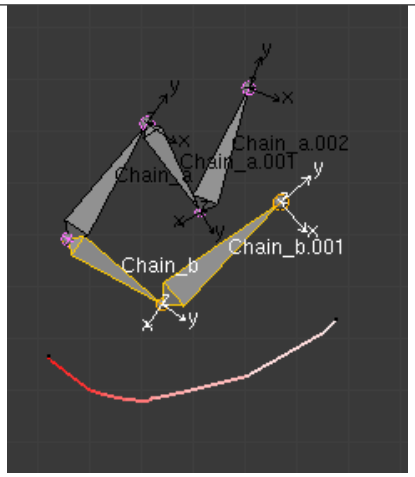


Fig. 2.1132: Before conversion.

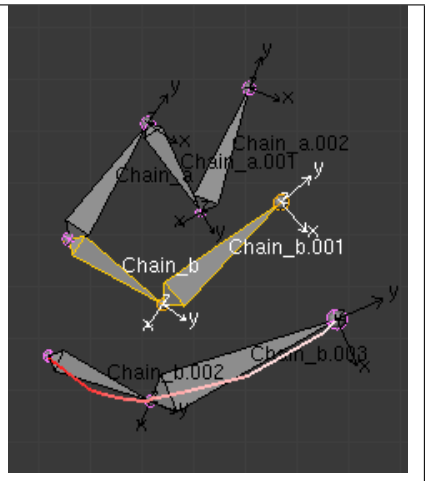
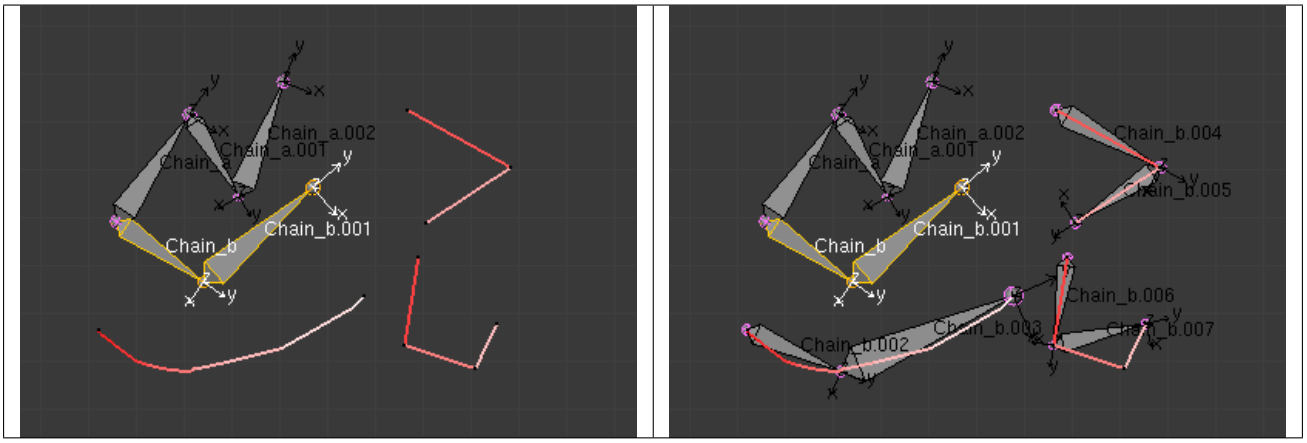


Fig. 2.1133: After conversion.

If you had selected both the chains ("Chain_a" and "Chain_b"), you would have still got the same result as in the example above, because the command maps to stroke only one selected chain.

In the following example also, only one chain is selected, but there are three strokes to map to. In this case, the same chain is copied three times (once for each stroke) and then mapped to individual strokes. Note how a two-bone chain is fitted to a three-segment stroke.

Table 2.62: After conversion.

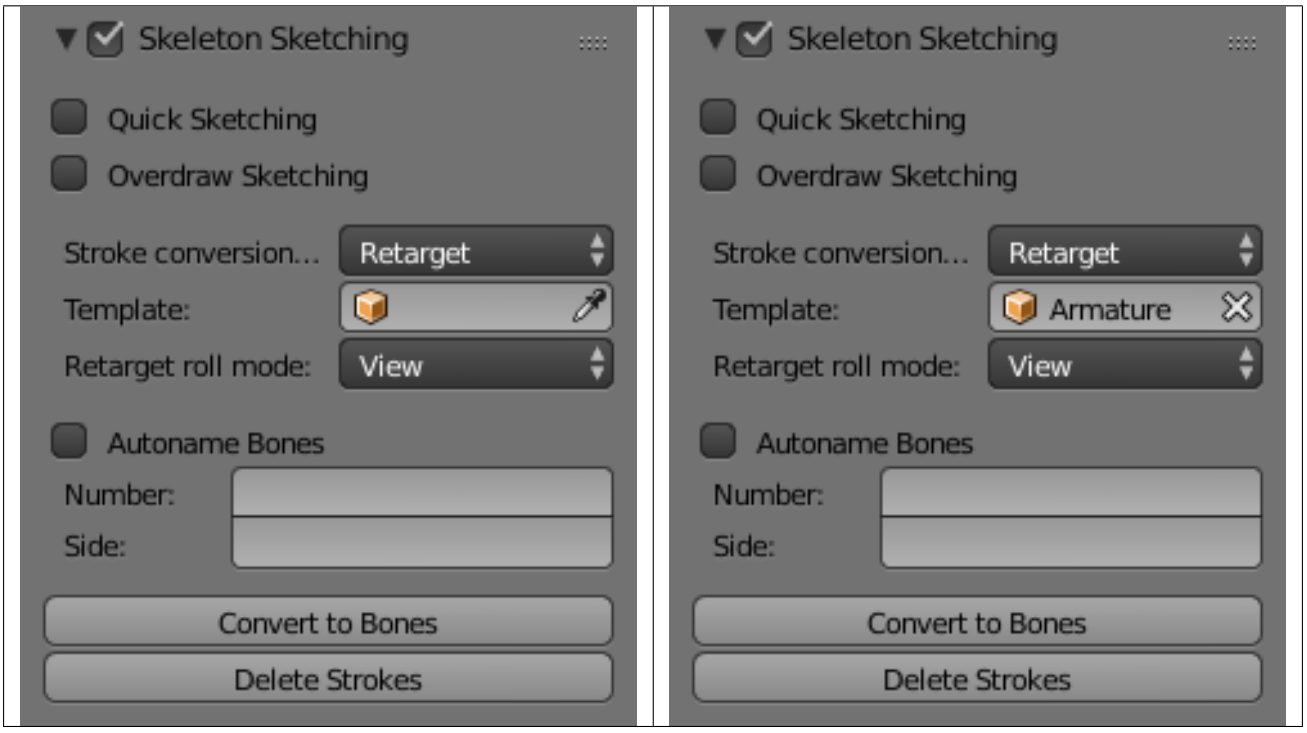


OK now, here are some important ground rules:

- This conversion method can use as reference bones either the selected bones in the *currently* edited armature, or *all* bones from another armature. In general, it is a better idea to create new "templated" bones inside the "reference" armature, so you can precisely select which bones to use as template – if you want the new bones in a different armature, you can then use the *Separate* `Ctrl-Alt-P` and optionally *Join* (`Ctrl-J` in *Object Mode*) commands...
- This tool only considers *one* chain of bones, so it is better to select only one chain of bones inside the current armature (or use a single-chain armature object as template). Else, the chain of the template containing the first created bones will be mapped to the selected strokes, and the other chains will just be "copied" *as is*, without any modification.

- This tool maps the same chain of bones on all selected strokes, so you cannot use multiple strokes to map a multi-chains template – you will rather get a whole set of new bones for each selected stroke!
- If you have strokes only made of straight segments, they must have *at least* as much segments as there are bones in the template chain (else, the newly created chain is not mapped at all to the stroke, and remains an exact duplicate of its template). If there are more segments than necessary, the conversion algorithm will chose the best "joints" for the bones to fit to the reference chain, using the same influence settings as for free segments (*Angle*, *Length* and *Definition* settings, see below).
- If you try to *Convert* without template bones (i.e. either an empty armature selected as template, or no bones selected in the current edited armature), you will get the error message `No Template and no deforming bones selected`, and nothing will occur.

Table 2.63: With another armature as template.



Now, here are the settings of this conversion method:

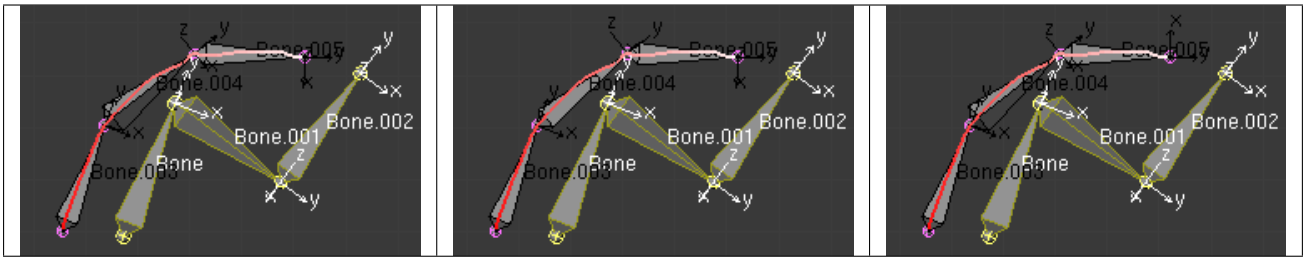
No, View, Joint buttons These three toggle buttons (mutually exclusive) control how the roll angle of newly created bones is affected:

No Do not alter the bones roll (i.e. the new bones' rolls fit their reference ones).

View Roll each bone so that one of its X, Y or Z local axis is aligned (as much as possible) with the current view's Z axis.

Joint New bones roll fit their original rotation (as *No* option), but with regards to the bend of the joint with its parent.

Table 2.64: With Joint roll option.



The “Bone.003” to “Bone.005” chain is the mapped-to-stroke version of “Bone” to “Bone.002” selected one, and “Bone.001” has a modified roll angle.

Template In this data-ID you can select the armature to use as template. If you choose *None*, the selected bones from the currently edited armature will be used as reference, else all bones of the other armature will be used.

***Angle*, *Length*, *Definition* are numeric fields.** These settings control how the template is mapped to the selected strokes. Each one can have a value between (0.0 and 10.0), the default being 1.0.

Angle Controls the influence of the angle of the joints (i.e. angle between bones). The higher this value, the more the conversion process will try to preserve these joints angle in the new chain.

Length Controls the influence of the bones’ length. The higher this value, the more the conversion process will try to preserve these lengths in the new bones.

Definition Controls the influence of the stroke’s shape. The higher this value, the more the conversion process will try to follow the stroke with the new chain.

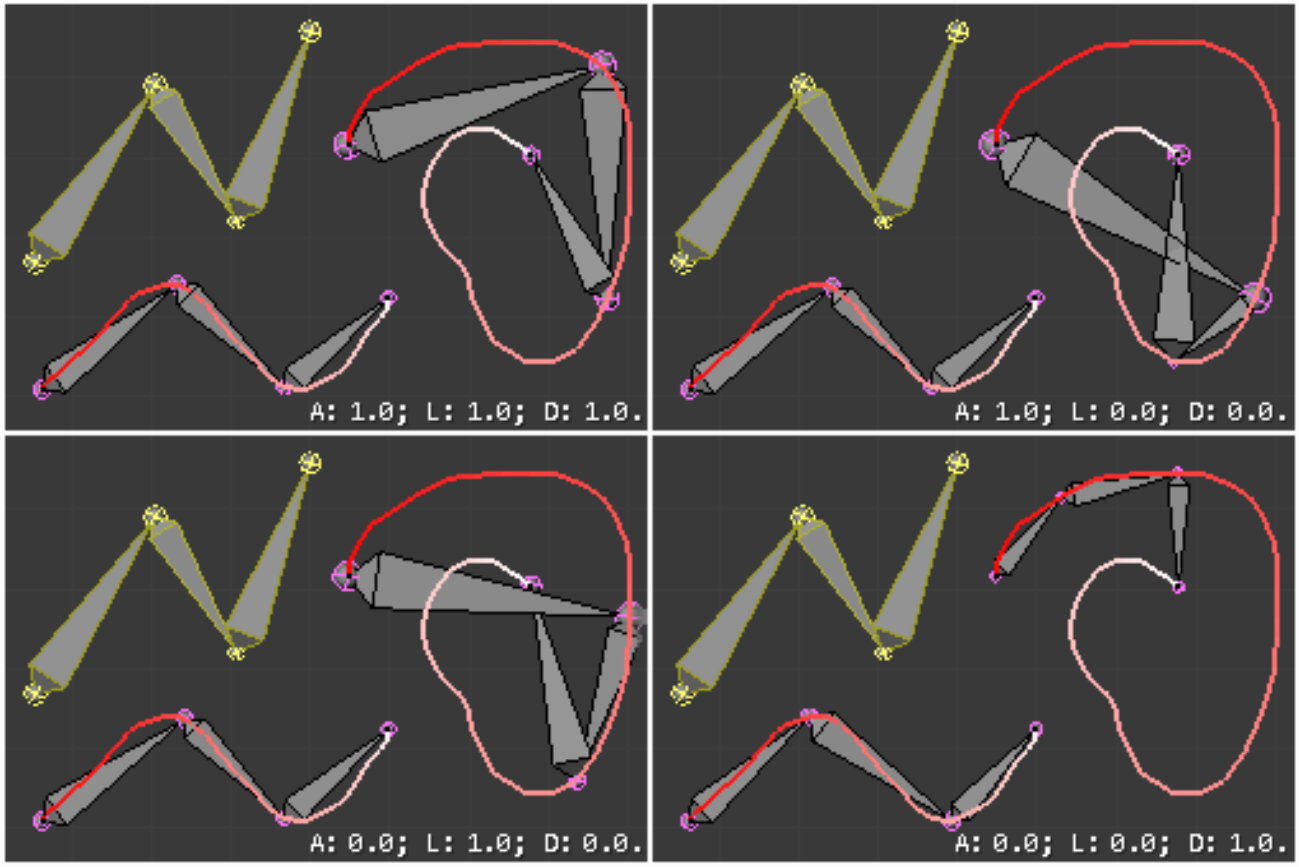


Fig. 2.1141: Examples of Template conversions for various influence weights values, with one stroke quite similar to the template chain’s shape, and one stroke very different.

Side and Number text fields, *auto* button These control how the new bones are named. By default, they just take the same names as the originals from the template, except for the final number, increased as needed. However, if the template bones have “&s” somewhere in their name, this “placeholder” will be replaced in the “templated” bones’ names by the content of the *Side* text field. Similarly, a “&n” placeholder will be replaced by the *Number* field content. If you enable the small *auto* button, the *Number* field content is auto-generated, producing a number starting from nothing, and increased each time you press the *Convert* button, and the “&s” placeholder is replaced by the side of the bone (relative to the local X axis: “r” for negative X values, “l” for positive ones).

Table 2.65: After conversion: the placeholders have been replaced by the content of the S and N text fields of the Bone Sketching panel.

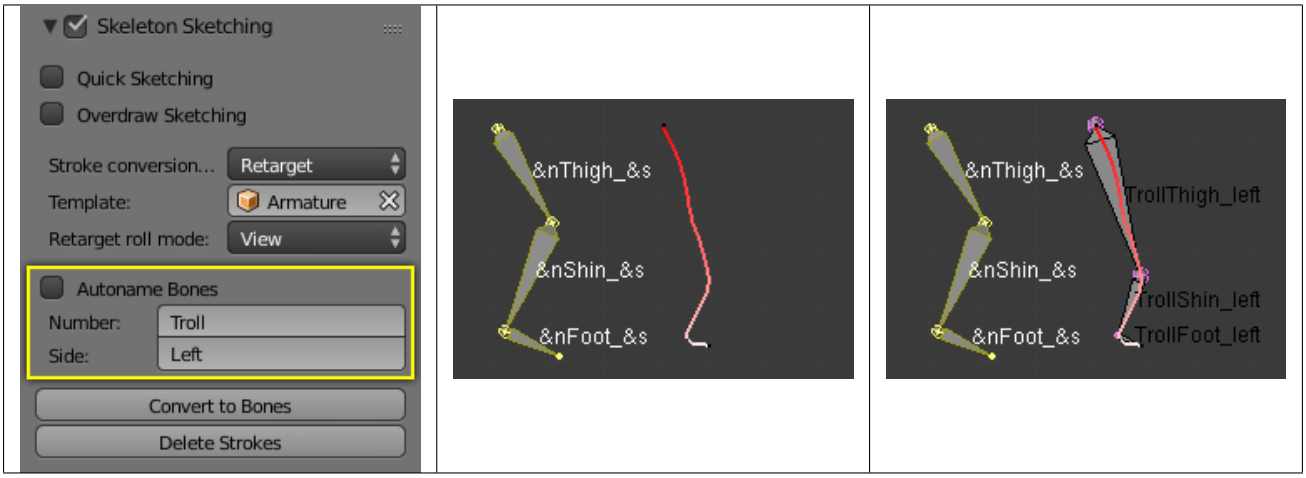
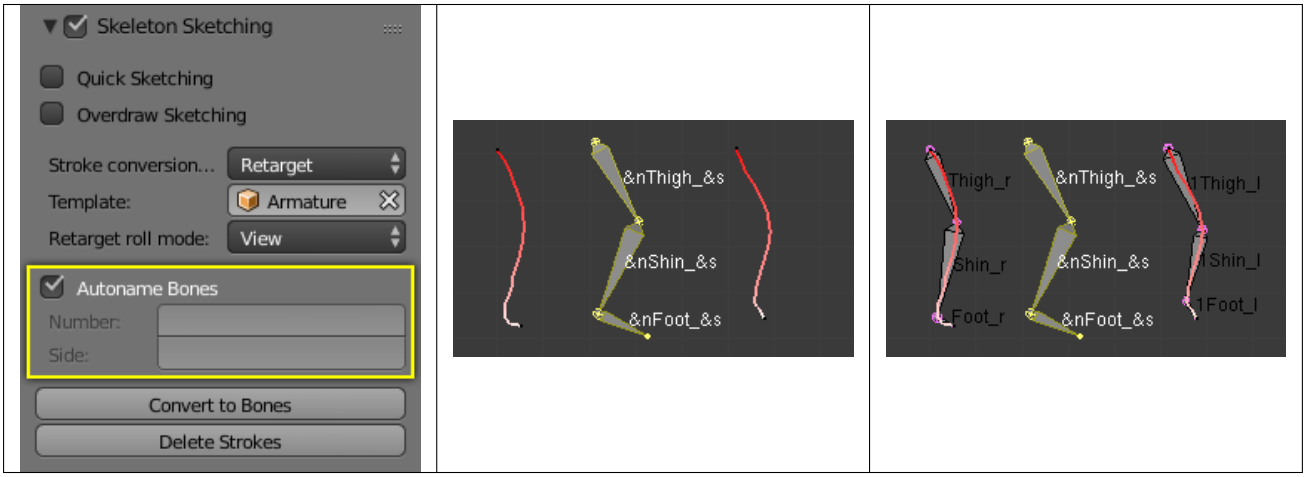


Table 2.66: After conversion.



Static text line The line just above the *Peel Objects* button gives you two informations:

- The *n* joints part gives you the number of joints (i.e. bones' ends, with connected ends considered as one joint), either from the selected bones of the edited armature, or in the whole other template armature.
- The second part is only present when another armature has been selected as template – it gives you the *root bone's name* of the chain that will be mapped to the strokes. Or, while you are drawing a stroke with straight segments, the name of the bone corresponding to the current segment (and "Done" when you have enough segments for all bones in the template chain).

Properties

Introduction

Reference

Mode: Object Mode, Edit Mode and Pose Mode
Panel: All in Properties editor, *Bone* property

When bones are selected (hence in *Edit Mode* and *Pose Mode*), their properties are shown in the *Bone* tab of the Properties editor. This shows different panels used to control features of each selected bone; the panels change depending on which mode you are working in.

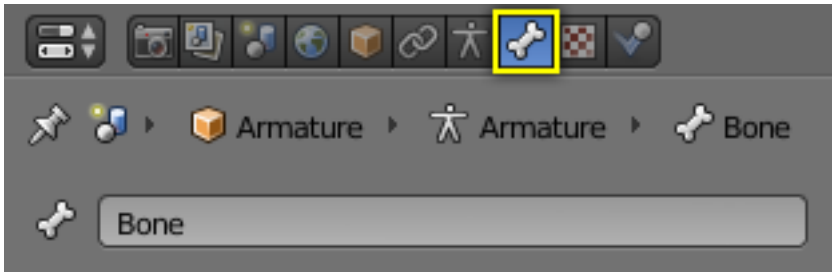


Fig. 2.1148: The Bone tab.

Relations In this panel you can arrange sets of bones in different layers for easier manipulation.

Display Display panel lets you customize the look of your bones taking the shape of another existing object.

Deform In this panel you can set basic deformation properties of the bones.

Transform

Fig. 2.1149: The Transform panel (edit mode).

When in edit mode you can use this panel to control position and roll of individual bones.

When in pose mode you can only set location for the main bone, and you can now set rotation and scale.

Note: This mode is only available in Edit Pose Modes.

Transform Locks

This panel appears only in pose mode and allows you to restrict position, rotation and scale by axis on each bone in the armature.

Note: This mode is only available in Pose Mode.

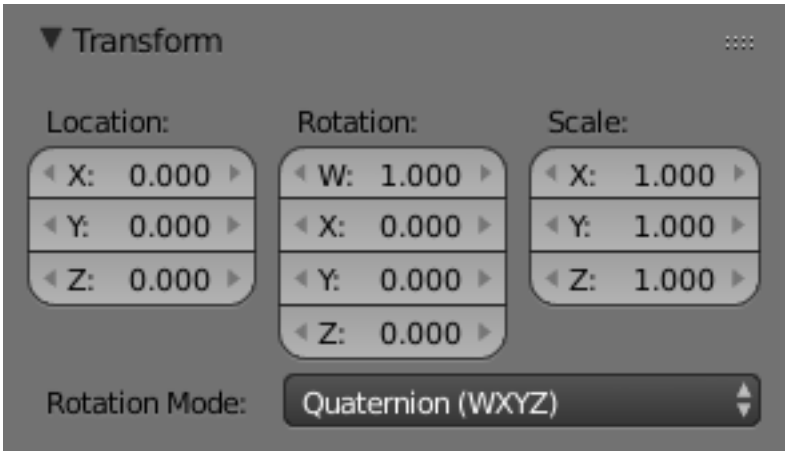


Fig. 2.1150: The Transform panel (pose mode).

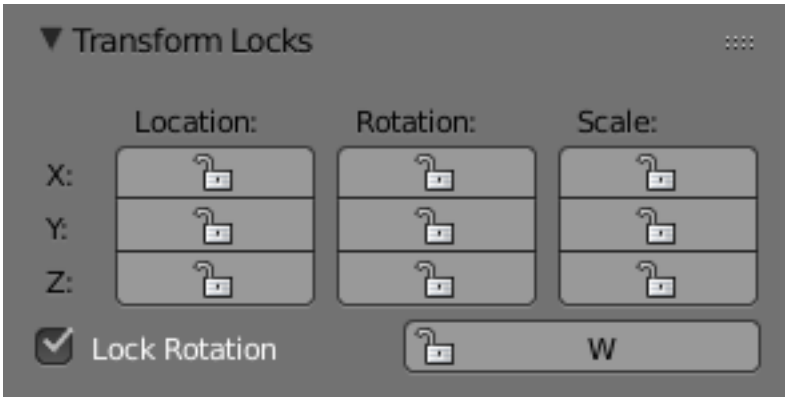


Fig. 2.1151: The Transform Locks panel.

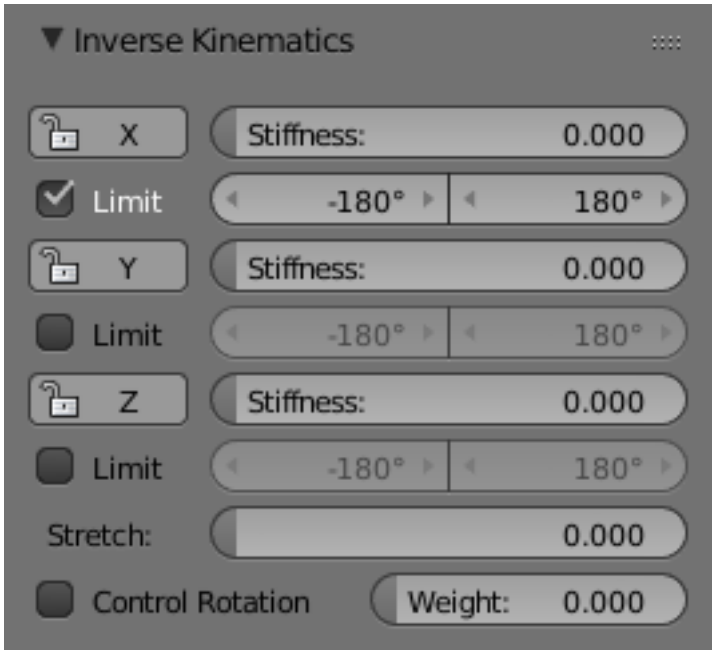


Fig. 2.1152: The Inverse Kinematics panel.

Inverse Kinematics

This panel controls the way a bone or set of bones behave when linked in an inverse kinematic chain.

Note: This mode is only available in Pose Mode.

Custom Properties

See the *Custom Properties* page for more information.

Display Panel

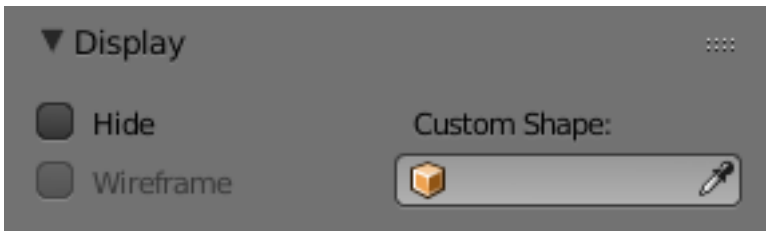


Fig. 2.1153: The Display panel.

Display panel lets you customize the look of your bones taking the shape of another existing object.

Hide Hides the selected bone.

Wireframe When enabled, bone is displayed in wireframe mode regardless of the viewport drawing mode. Useful for non-obstructive custom bone chains.

Shaped Bones

Reference

Mode: Object and Pose Mode
Panel: *Bone* → *Display*

Blender allows you to give to each bone of an armature a specific shape (in *Object Mode* and *Pose Mode*), using another object as "template". First of all, you have to enable the *Shapes* button (*Armature* panel).

Options

Custom Shape Object that defines the custom shape of the selected bone.

Custom At Bone that defines the display transform of this shape bone.

Workflow

To assign a custom shape to a bone, you have to:

- Switch to *Pose Mode* `Ctrl-Tab`.
- Select the relevant bone by clicking on it with `RMB`.
- Go to the *Display* panel *Custom Shape* field and select the 3D object previously created in the scene; in this example we are using a cube and a cone. You can optionally set the *At* field to another bone.

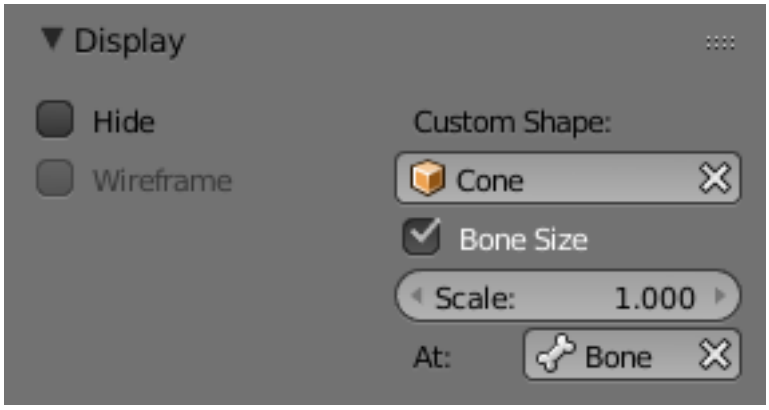


Fig. 2.1154: The Display panel.

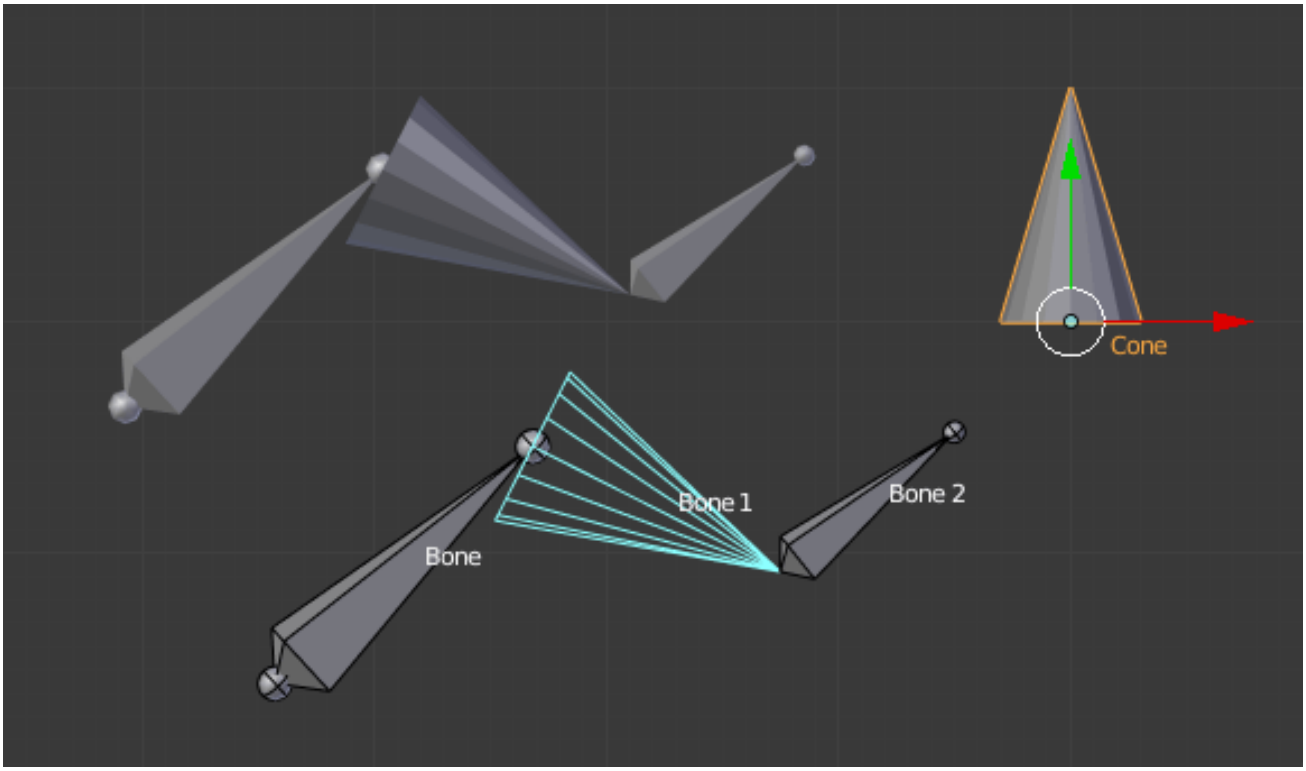


Fig. 2.1155: The armature with shape assigned to bone. Note the center of the Cone object.

Note:

- These shapes will never be rendered, like any bone, they are only visible in 3D Views.
- Even if any type of object seems to be accepted by the *OB* field (meshes, curves, even metas...), only meshes really work. All other types just make the bone invisible; nothing is drawn...

- The center of the shape object will be at the *root of the bone* (see the *bone page* for root/tip).
- The object properties of the shape are ignored (i.e. if you make a parallelepiped out of a cube by modifying its dimensions in *Object Mode*, you will still have a cube shaped bone...).
- The "along bone" axis is the Y one, and the shape object is always scaled so that one Blender Unit stretches along the whole bone length.
- If you need to remove the custom shape of the bone, just right click in the *Custom Shape* field and select *Reset to default value* in the pop-up menu.

So to summarize all this, you should use meshes as shape objects, with their center at their lower -Y end, and an overall Y length of 1.0 BU.

Relations

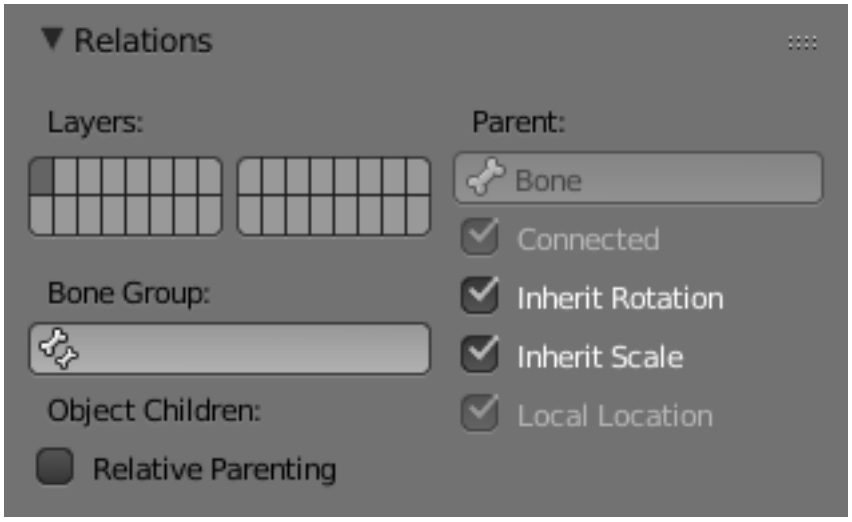


Fig. 2.1156: The Relations panel.

In this panel you can arrange sets of bones in different layers for easier manipulation.

Bone Layers

Reference

Mode: Object, Edit and Pose Mode
Panel: *Bone* → *Relations*

Moving bones between layers

Obviously, you have to be in *Edit Mode* or *Pose Mode* to move bones between layers. Note that as with objects, bones can lay in several layers at once, just use the usual `Shift-LMB` clicks... First of all, you have to select the chosen bone(s)!

- In the Properties editor, use the "layer buttons" of each selected bone Relations panel (*Bones* tab) to control in which layer(s) it lays.

- In the *3D View* editor, use the menu *Armature* → *Move Bone To Layer* or *Pose* → *Move Bone To Layer* or press M to show the usual pop-up layers menu. Note that this way, you assign the same layers to all selected bones.

Bone Group

Reference

Mode: Pose Mode

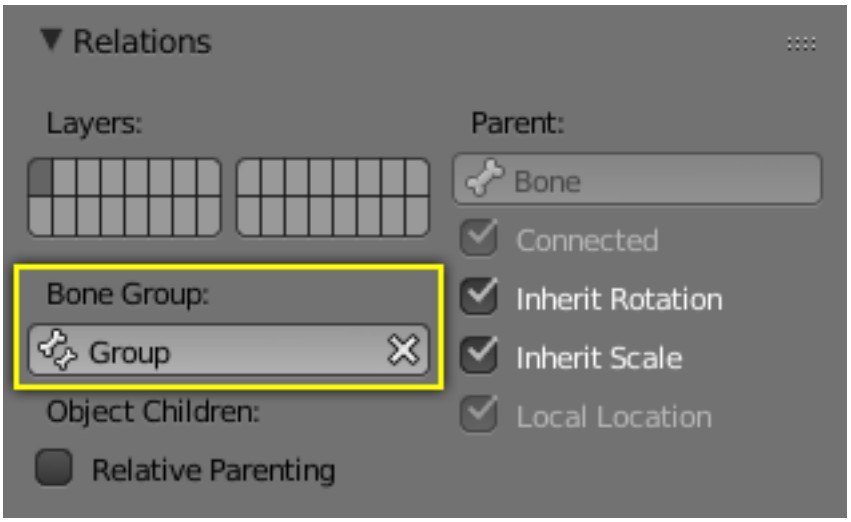


Fig. 2.1157: The Bone Group data-ID.

To assign a selected bone to a given bone group use the *Bone Group* data-ID.

Object Children

Reference

Mode: Pose Mode

Relative Parenting ToDo.

Parenting

Parent A data-ID to select the bone to set as a parent.

Connected The *Connected* checkbox set the head of the bone to be connected with its parent root.

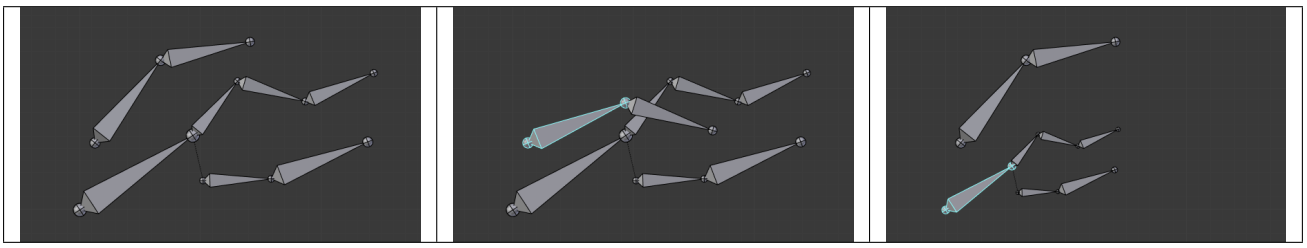
Transformations

Bones relationships have effects on transformations behavior.

By default, children bones inherit:

- Their parent position, with their own offset of course.
- Their parent rotation (i.e. they keep a constant rotation relatively to their parent).
- Their parent scale, here again with their own offset.

Table 2.67: Scaling of a root bone.



Exactly like standard children objects. You can modify this behavior on a per-bone basis, using the Relations panel in the *Bones* tab:

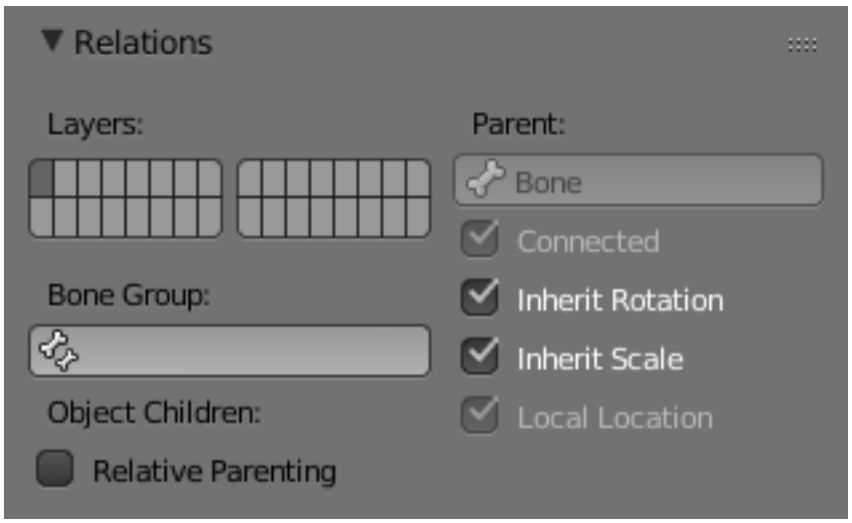


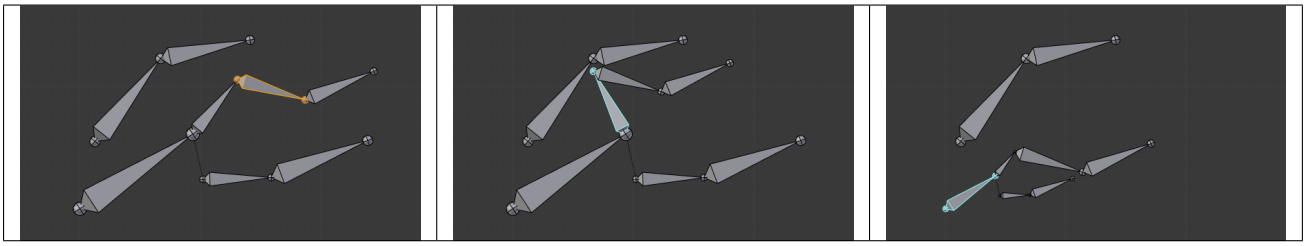
Fig. 2.1161: Relations panel in Pose Mode.

Inherit Rotation When disabled, this will "break" the rotation relationship to the bone's parent. This means that the child will keep its rotation in the armature object space when its parent is rotated.

Inherit Scale When disabled, this will "break" the scale relationship to the bone's parent.

These inheriting behaviors propagate along the bones' hierarchy. So when you scale down a bone, all its descendants are by default scaled down accordingly. However, if you set one bone's *Inherit Scale* or *Inherit Rotation* property on in this "family", this will break the scaling propagation, i.e. this bone *and all its descendants* will no longer be affected when you scale one of its ancestors.

Table 2.68: Scaling of a bone with an Inherit Rotation disabled bone among its descendants.



Connected bones have another specificity: they cannot be translated. Indeed, as their root must be at their parent's tip, if you do not move the parent, you cannot move the child's root, but only its tip, which leads to a child rotation. This is exactly what happens, when you press G with a connected bone selected, Blender automatically switches to rotation operation.

Bones relationships also have important consequences on how selections of multiple bones behave when transformed. There are many different situations which may not be included on this list, however, this should give a good idea of the problem:

- Non-related selected bones are transformed independently, as usual.

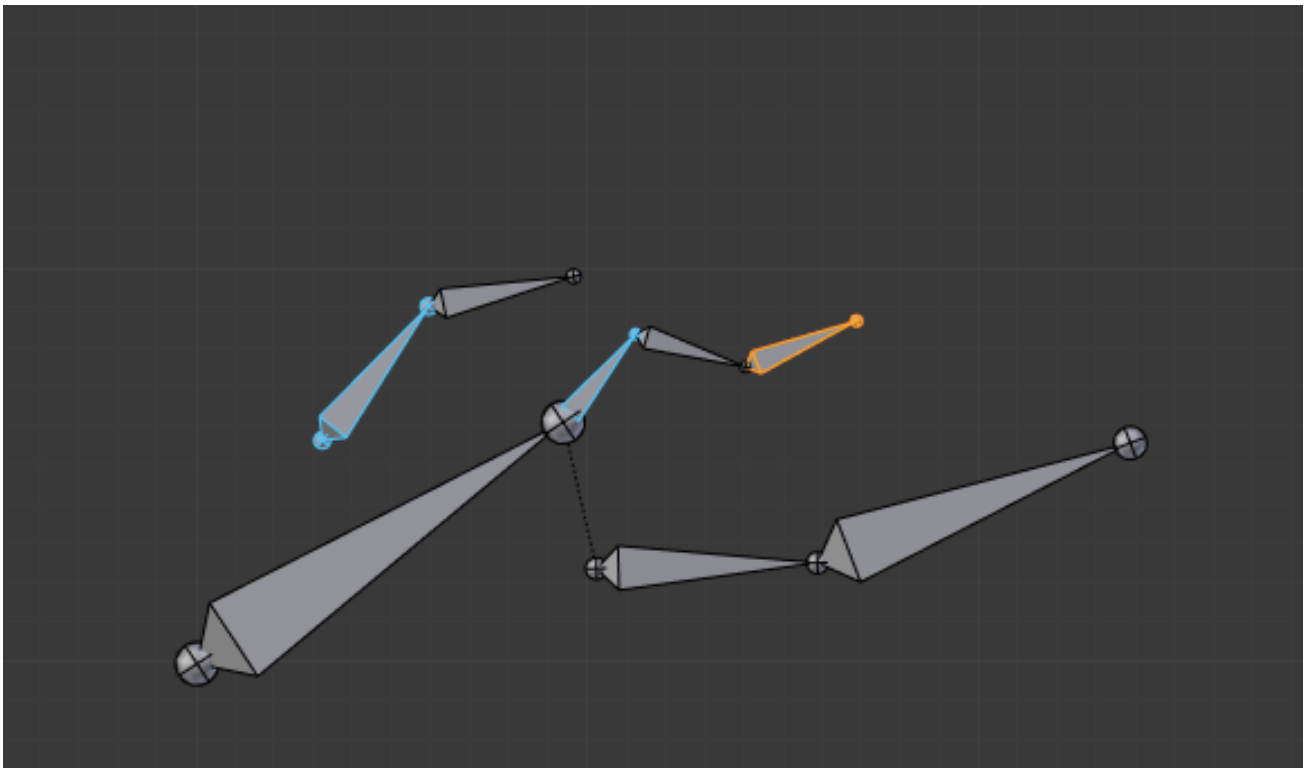


Fig. 2.1165: Scaling bones, some of them related.

- When several bones of the same "family" are selected, *only* the "most parent" ones are really transformed – the descendants are just handled through the parent relationship process, as if they were not selected (see Fig. *Scaling bones, some of them related.* the third tip bone, outlined in yellow, was only scaled down through the parent relationship, exactly as the unselected ones, even though it is selected and active. Otherwise, it should have been twice smaller!).
- When connected and unconnected bones are selected, and you start a grab operation, only the unconnected bones are affected.
- When a child connected hinge bone is in the selection, and the "most parent" selected one is connected, when you press G, nothing happens, because Blender remains in grab operation, which of course has no effect on a connected bone.

So, when posing a chain of bones, you should always edit its elements from the root bone to the tip bone. This process is known as *forward kinematics* (FK). We will see in a *later page* that Blender features another pose method, called *inverse kinematics* (IK), which allows you to pose a whole chain just by moving its tip.

Note: This feature is somewhat extended/completed by the *pose library* tool.

Deform

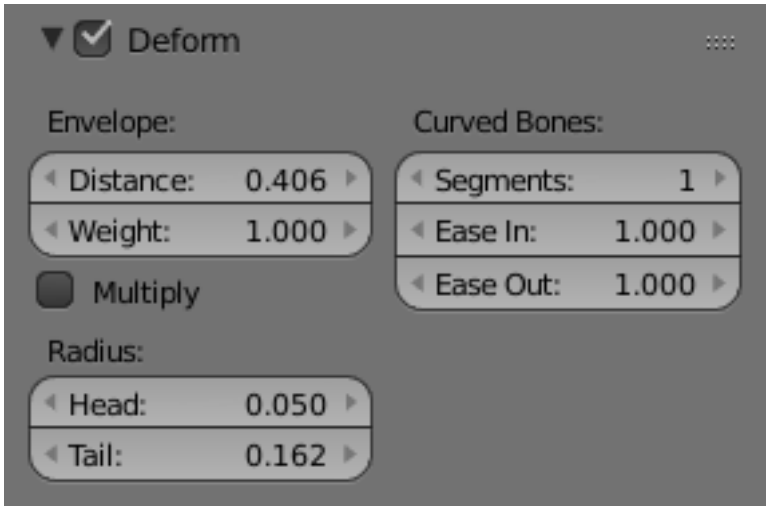


Fig. 2.1166: The Deform panel.

In this panel you can set basic properties of the bones.

Turning the Deform option on and off, includes the active bone in the Automatic Weight Calculation when the Mesh is Parented to the Armature using the Armature Deform with the "With Automatic Weights" option.

Also it is worth noting that by turning off a bone's deform option, makes it not influence the mesh at all, overriding any weights that it might have been assigned before; It mutes its influence.

Envelope

Curved Bones

Segments

Segments The *Segments* number button allows you to set the number of segments, which the given bone is subdivided into. Segments are small, rigid linked child bones that interpolate between the root and the tip. The higher this setting, the smoother "bends" the bone, but the heavier the pose calculations...

Technical Details

When you connect bones to form a *chain*, Blender calculates a Bézier curve passing through all the bones' ends, and bones' segments in the chain will bend and roll to follow this invisible curve. There is no direct access to the curve. It can only be controlled by some extent using bone properties.

However, if the chain has an influence on objects rather than geometry, the segments' orientation is not taken in account (details are explained in the *skinning part*).

Display

You can see these segments in *Object Mode* and in *Pose Mode*, and only if bones are visualized as *B-bones* or *Wire*; while in *Edit Mode* bones are always drawn as rigid sticks. Note that in the special case of a single bone, you cannot see these segments in *Object Mode*, because they are aligned.

When not visualized as *B-Bone* s, bones are always shown as rigid sticks, even though the bone segments are still present and effective (see *skinning to Object Data*). This means that even in e.g. *Octahedron* visualization, if some bones in a chain have several segments, they will nonetheless smoothly deform their geometry...

Example

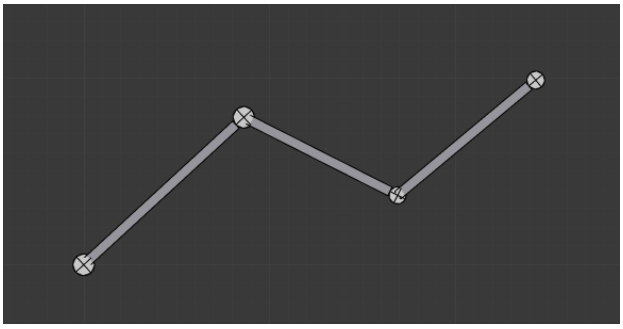


Fig. 2.1167: An armature of B-Bones, in Edit Mode.

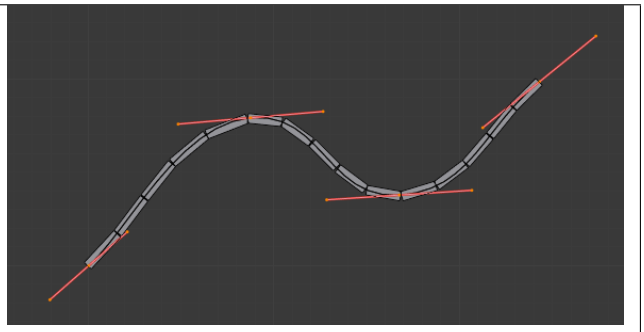


Fig. 2.1168: The Bézier curve superposed to the chain, with its handles placed at bones' ends.

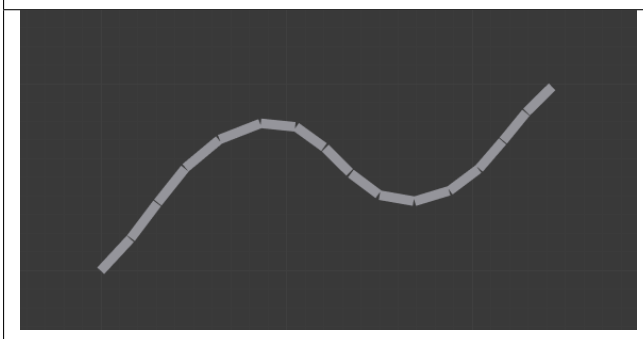


Fig. 2.1169: The same armature in Object Mode.

In Fig. *An armature of B-Bones, in Edit Mode.* we connected three bones, each one made of five segments. These are *B-bones* but as you see, in *Edit Mode* they are shown as rigid elements. Look at Fig. *The same armature in Object Mode.*, we can see how the bones' segments smoothly "blend" into each other, even for roll.

Usage

Curve bones are an easy way to replace long chains of many small rigid bones. A common use case for curve bones is to model spine columns.

Ease

Ease In, Ease Out The *Ease In/Out* number buttons, change the "length" of the *"auto"* Bézier handle to control the "root handle" and "tip handle" of the bone, respectively. These values are proportional to the default length, which of course automatically varies depending on bone length, angle with previous/next bones in the chain, and so on.

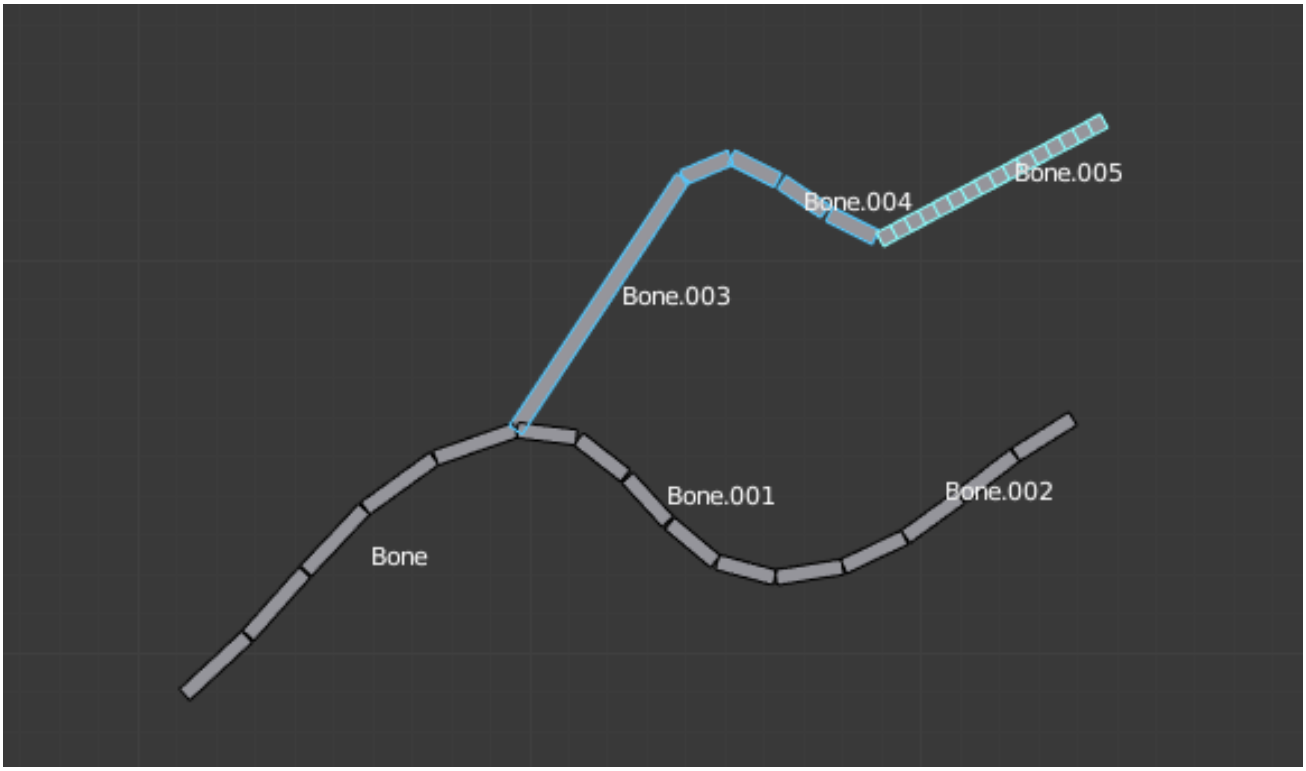
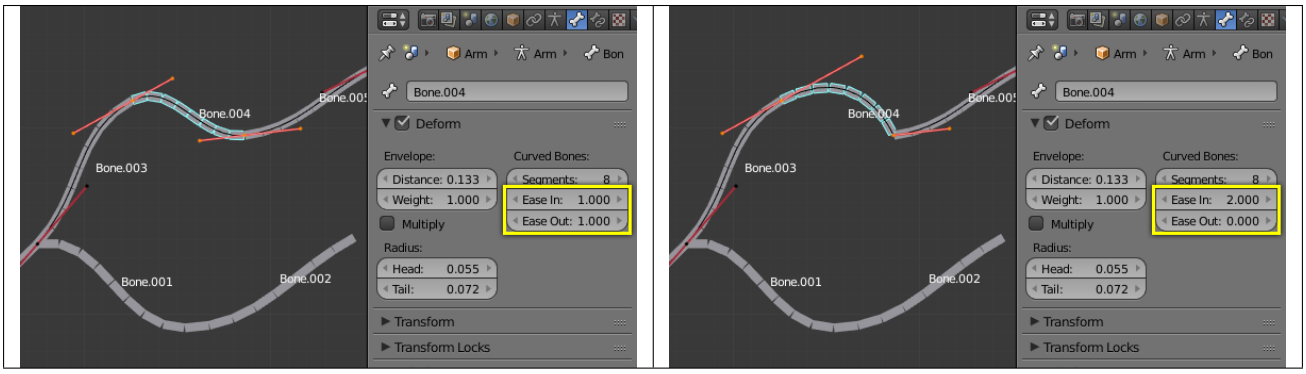


Fig. 2.1170: An armature in Pose Mode, B-Bone visualization: Bone.003 has one segment, Bone.004 has four, and Bone.005 has sixteen.

Table 2.69: Bone.004 with In at 2.0, and Out at 0.0.



Properties

Introduction

Reference

Mode: Object Mode, Edit Mode and Pose Mode
Panel: All in Properties editor, *Object* property

Let us first have a general overview of the various panels gathering the armature settings, in Properties editor, *Object* tab:

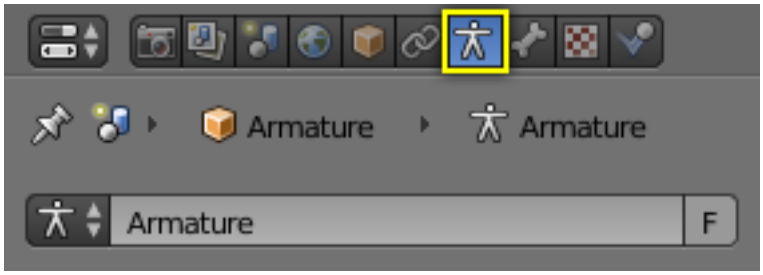


Fig. 2.1173: The Object property in the Properties editor.

Skeleton In this panel you can arrange sets of bones into different layers for easier manipulation.

Display This controls the way the bones appear in 3D View.

Bone Groups Bone Groups are meant to be used during the rig creation to define and assign a color to a meaningful set of bones.

Pose Library Allows you to save different properties (location, rotation, scale) for selected bones for later use.

Ghost Allows you to see a set of different poses, very useful when animating.

Motion Paths In this panel you can enable visualization of the motion path your skeleton leaves when animated.

Inverse Kinematics

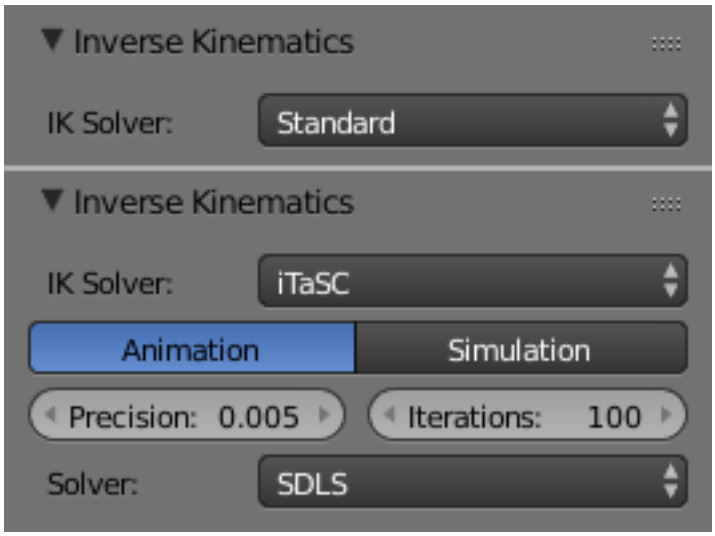


Fig. 2.1174: The Inverse Kinematics panel.

Defines the type of IK solver used in your animation.

Custom Properties

See the *Custom Properties* page for more information.

Skeleton

In this panel you can arrange sets of bones into different layers for easier manipulation.

Fig. 2.1175: The Skeleton panel.

Position

A radio button to switch between Pose Position and Rest Position.

Whereas in *Edit Mode*, you always see your armature in its rest position, in *Object Mode* and *Pose Mode* you see it by default in its *Pose Position* (i.e. as it was transformed in the *Pose Mode*). If you want to see it in the rest position in all modes, enable the *Rest Position* button in the *Armature* tab (*Edit Mode*).

Armature Layers

Reference

Mode: Object, Edit and Pose Mode
Panel: *Object data* → *Skeleton*

Each armature has 32 "Armature layers" which allow you to organize your armature by "regrouping" sets of bones into layers; this works similar to scene layers (those containing your objects). You can then "move" a bone to a given layer, hide or show one or several layers, etc.

Showing/hiding bone layers

Only bones in active layers will be visible/editable, but they will always be effective (i.e move objects or deform geometry), whether in an active layer or not. To (de)activate a layer, you have several options, depending in which mode you are in:

- In all modes, use the row of small buttons at the top of the *Display Options* group, *Armature* panel. If you want to enable/disable several layers at once, as usual, hold `Shift` while clicking...
- In *Edit Mode* and *Pose Mode*, you can also do this from the *3D View*, either by using the menu *Armature* → *Switch Armature Layers* or *Pose* → *Switch Armature Layers*, or the `Shift-M` shortcut, to display a small pop-up menu containing the same buttons as described above (here again, you can use `Shift-LMB` clicks to (de)select several layers at once).

Protected Layers

You can lock a given bone layer for all *proxies* of your armature, i.e. all bones in this layer will not be editable. To do so, in the *Skeleton* panel, Ctrl−LMB click on the relevant button, the layer lock will be enabled.

Protected layers in proxy are restored to proxy settings on file reload and undo.

Display Panel

Reference

Mode: Object, Edit and Pose Mode
Panel: *Object Data* → *Display*

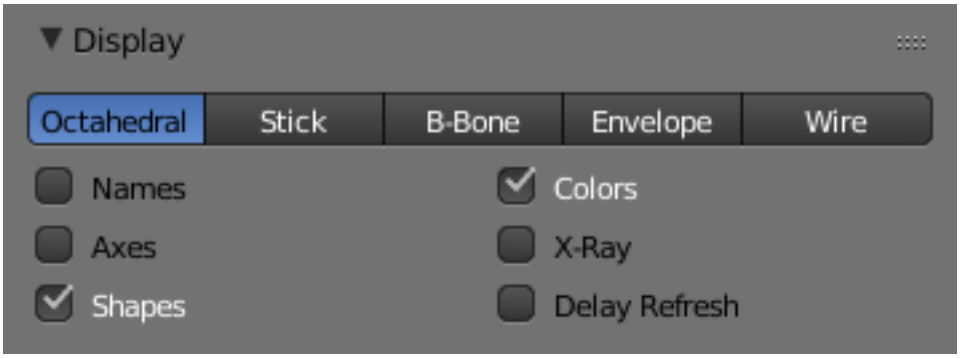


Fig. 2.1176: The Display panel.

This controls the way the bones appear in 3D View; you have four different visualizations you can select.

Bone Types

We have four basic bone visualization: Octahedral, Stick, B-Bone, Envelope and Wire:

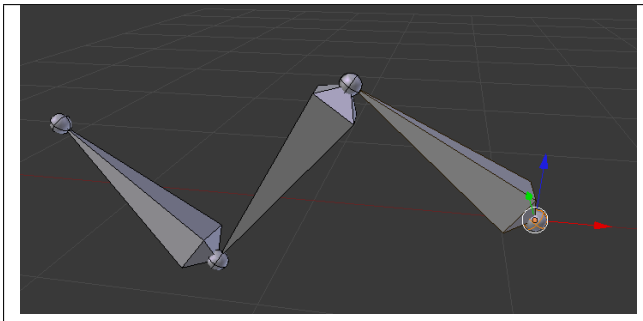


Fig. 2.1177: Octahedral bone display.

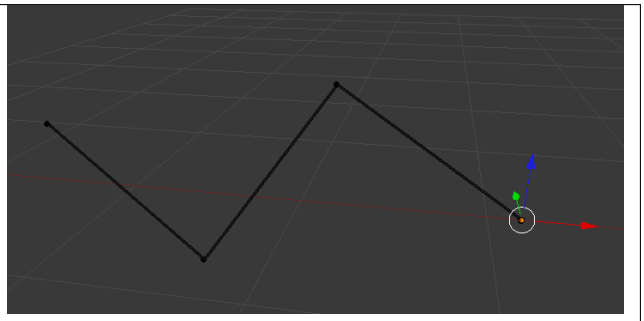


Fig. 2.1178: Stick bone display.

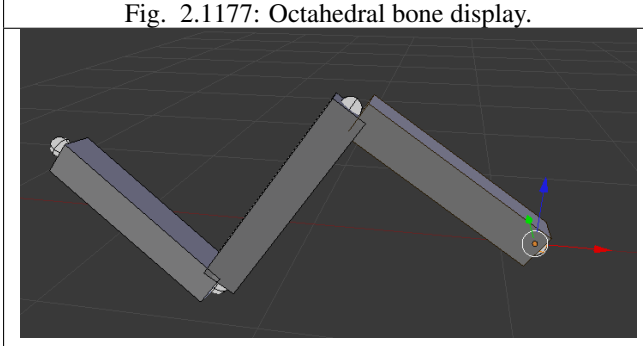


Fig. 2.1179: B-Bone bone display.

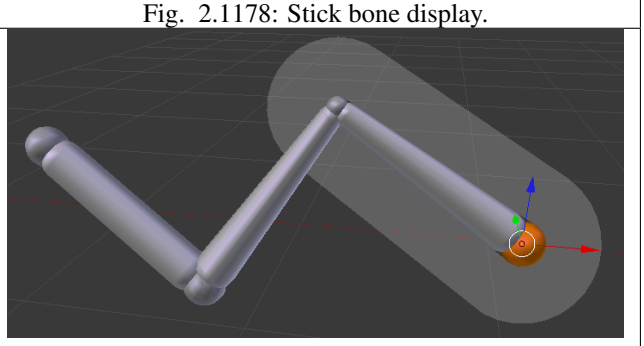


Fig. 2.1180: Envelope bone display.

Octahedral bone This is the default visualization, well suited for most of editing tasks. It materializes:

- The bone root (“big” end) and tip (“small” end).
- The bone “size” (its thickness is proportional to its length).
- The bone roll (as it has a square section).

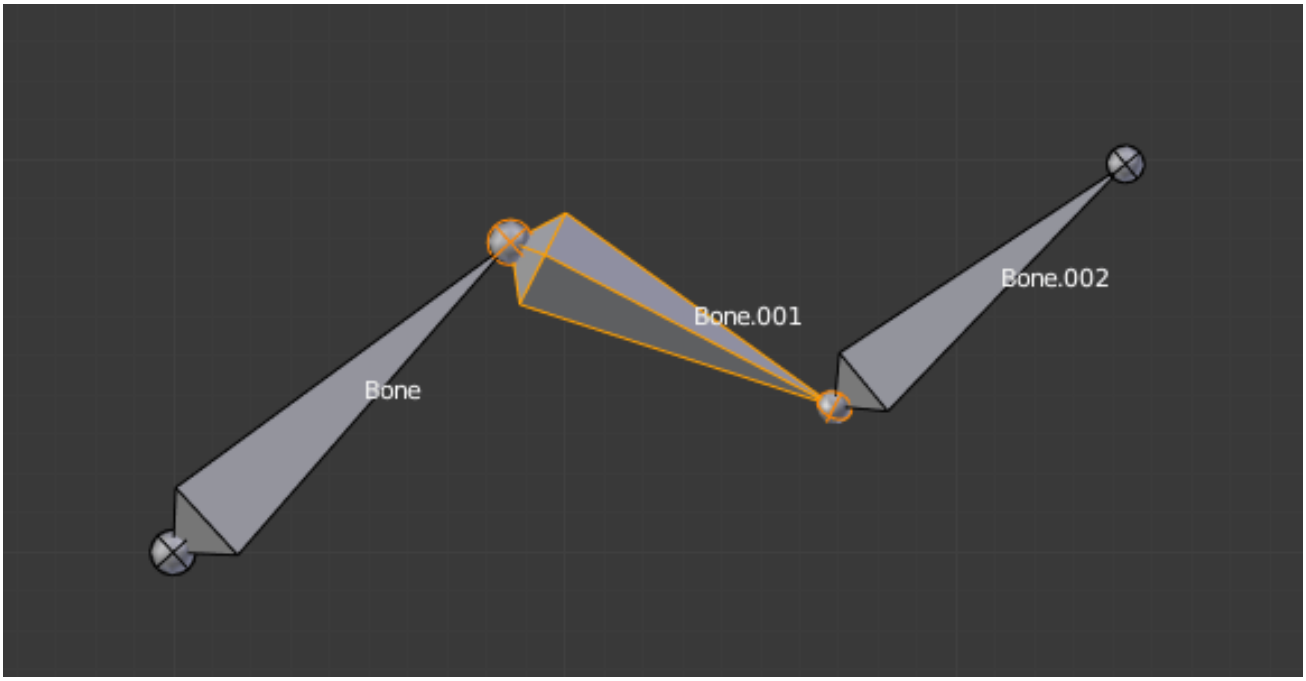


Fig. 2.1181: Note the 40° rolled Bone.001 bone.

Stick bone This is the simplest and most non-intrusive visualization. It just materializes bones by sticks of constant (and small) thickness, so it gives you no information about root and tip, nor bone size or roll angle.

B-Bone bone This visualization shows the curves of “smooth” multi-segmented bones; see the *bone page* for details.

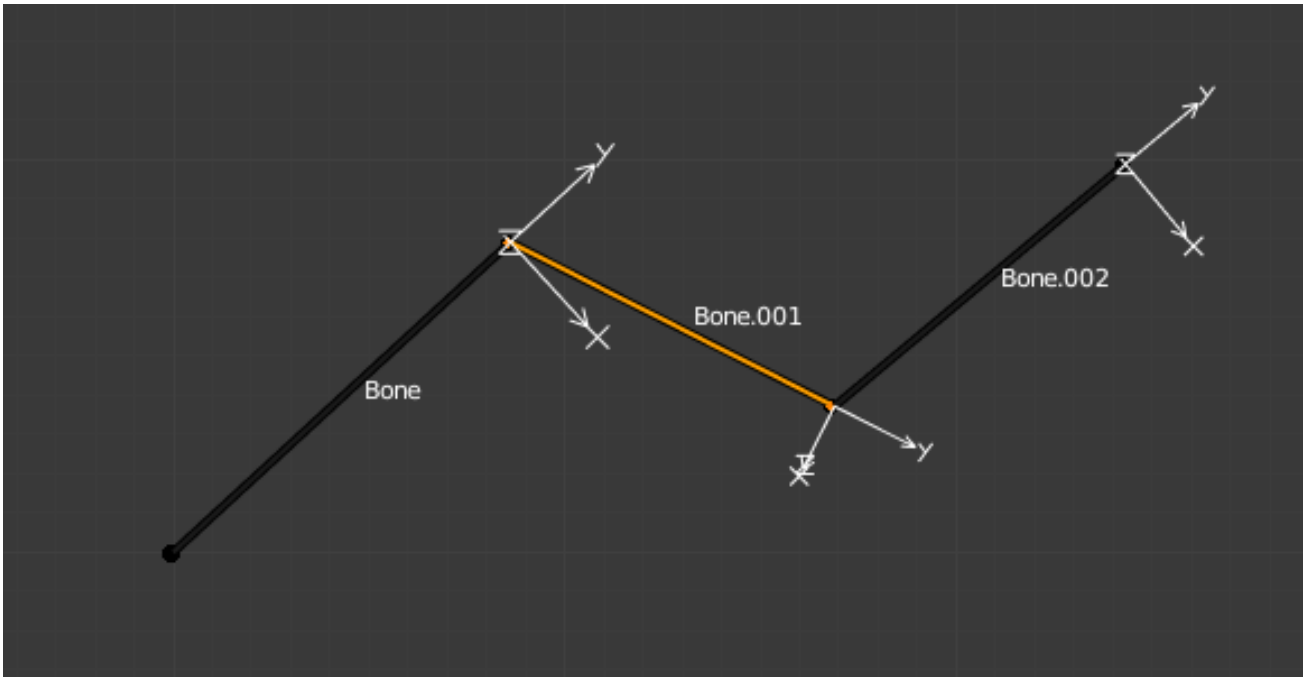


Fig. 2.1182: Note that Bone.001 roll angle is not visible (except by its XZ axes).

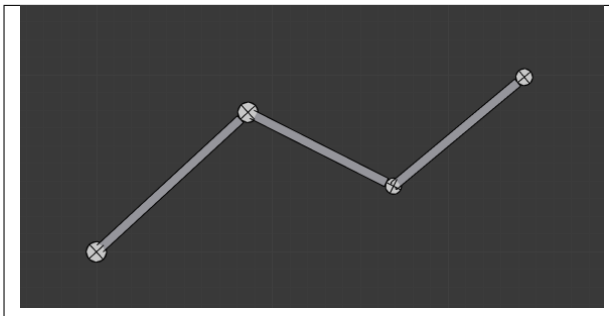


Fig. 2.1183: An armature of B-Bones, in Edit Mode.

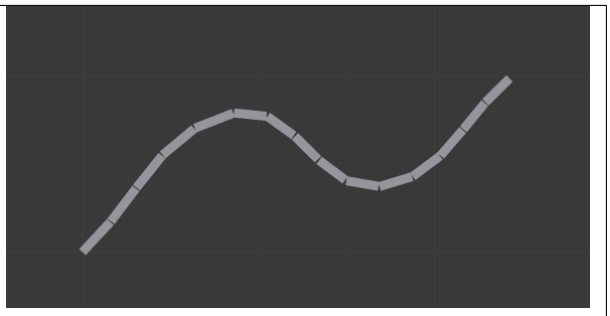


Fig. 2.1184: The same armature in Object Mode.

Envelope bone This visualization materializes the bone deformation influence. More on this in the *bone page*.

Wire bone This simplest visualization shows the curves of "smooth" multi-segmented bones.

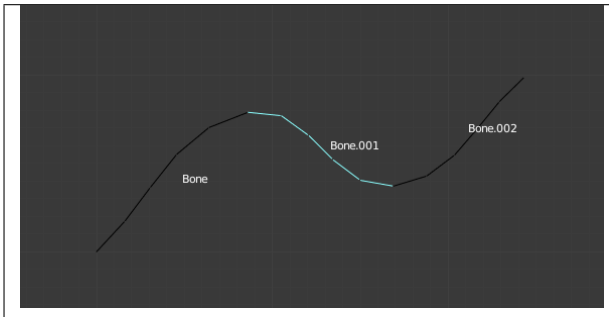


Fig. 2.1185: An armature of Wire, in Pose Mode.

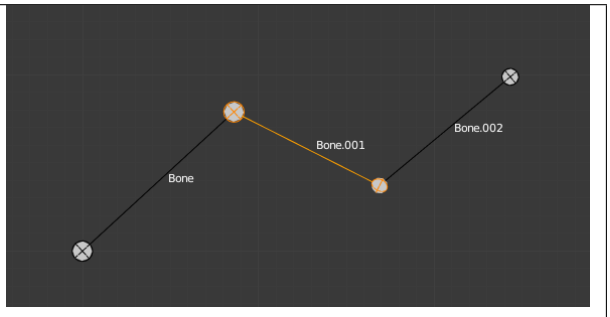
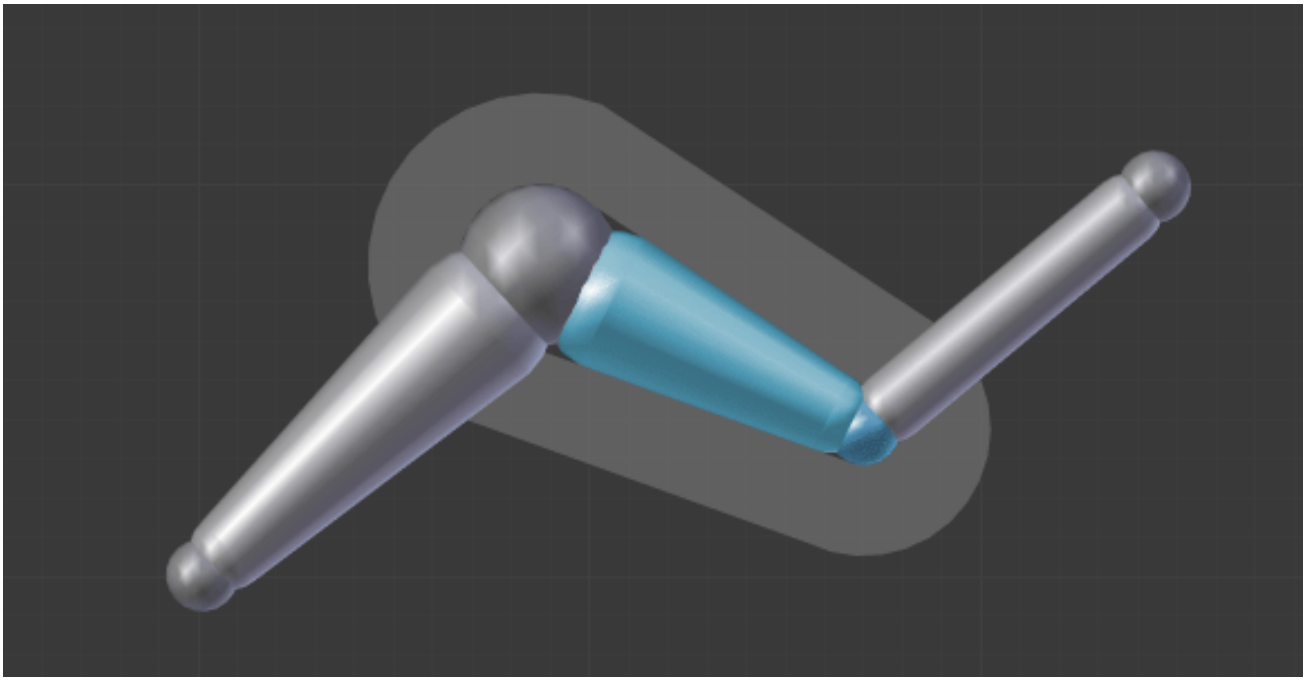


Fig. 2.1186: The same armature in Edit Mode.

Draw Options

Names When enabled, the name of each bone is drawn.

Colors This is only relevant for *Pose Mode*, and is described in detail *there*.



Axes When enabled, the (local) axes of each bone are drawn (only relevant for *Edit Mode* and *Pose Mode*).

X-Ray When enabled, the bones of the armature will always be drawn on top of the solid objects (meshes, surfaces, ...) - i.e. they will always be visible and selectable (this is the same option as the one found in the *Display* panel of the *Object data* tab. Very useful when not in *Wireframe* mode.

Shapes When enabled, the default standard bone shape is replaced, in *Object Mode* and *Pose Mode*, by the shape of a chosen object (see *Shaped Bones* for details).

Delay Refresh When enabled, the bone does not deform its children when manipulating the bone in pose mode.

Bone Groups

Reference

Mode: Pose Mode
Panel: *Armature tab* → *Motion Paths panel*
Menu: *Pose* → *Bone Groups* → ...

This panel allows the creation, deletion and editing of Bone Groups. The bone groups panel is available in the Armature tab of the Properties editor.

Bone Groups can be used for selection or to assign a color theme to a set of bones. In example to color the left parts of the rig as blue and right parts as red.

Active Bone Group The Bone Group *List view*.

Color Set

You can assign a "color theme" to a group (each bone will have these colors). Remember you have to enable the *Colors* checkbox (*Display* panel) to see these colors.

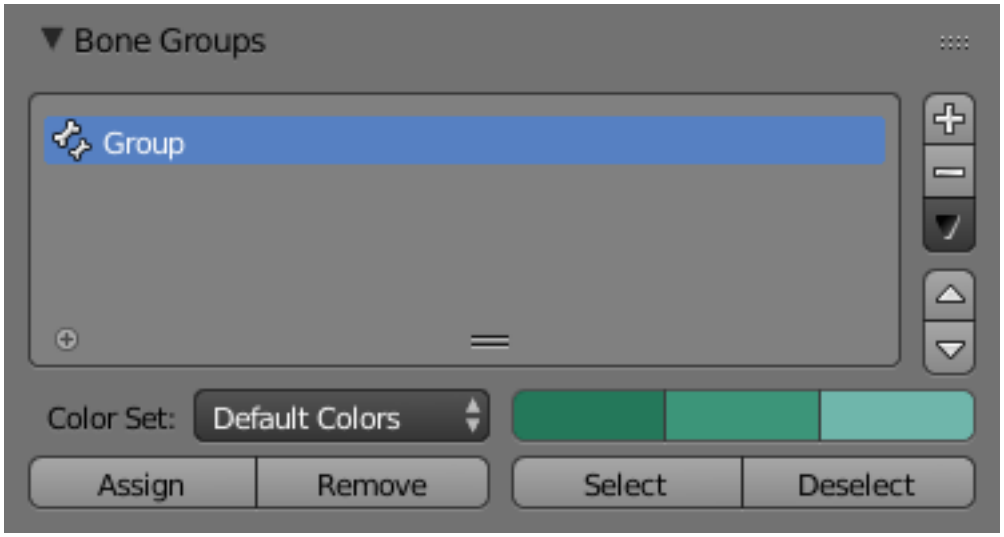


Fig. 2.1187: The Bone Groups panel.

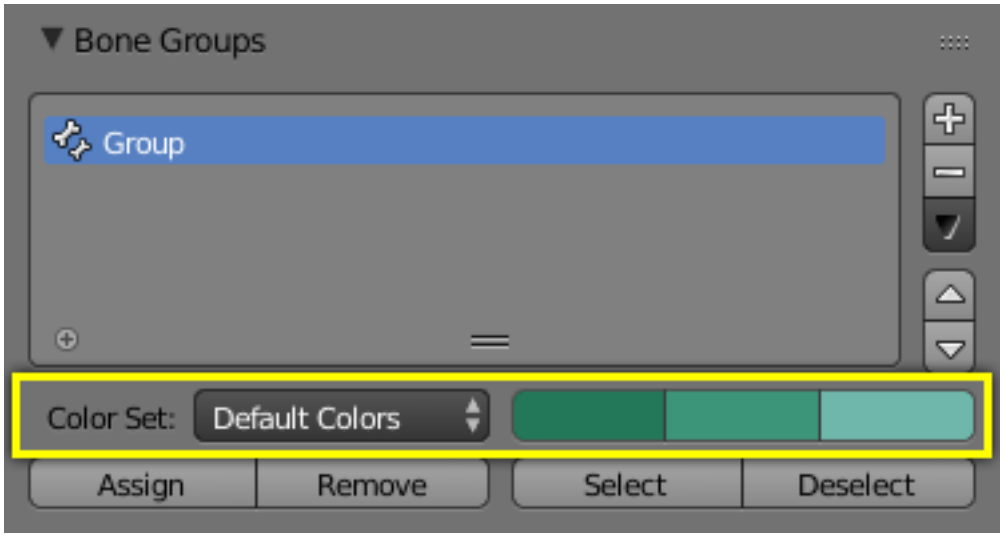


Fig. 2.1188: The Bone Color Set selector and the color buttons.

Bone Color Set A select menu.

- *Default Colors*: The default (gray) colors.
- *nn - Theme Color Set*: One of the twenty Blender presets by the theme.
- *Custom Set*: A custom set of colors, which is specific to each group.

Normal The first color button is the color of unselected bones.

Selected The second color button is the outline color of selected bones.

Active The third color button is the outline color of the active bone.

As soon as you alter one of the colors, it is switched to the *Custom Set* option.

Assign and Select

In the 3D Views, using the *Pose* → *Bone Groups* menu entries, and/or the *Bone Groups* pop-up menu `Ctrl-G`, you can:

Assign Assigns the selected bones to the active bone group. It is important to note that a bone can only belong to one group.

Remove Removes the selected bones from the active bone group.

Select Selects the bones in the active bone group.

Deselect Deselects the bones in the active bone group.

See also:

A single bone can be assign to a group in the *Relations panel*.

See also:

Bones belonging to multiple groups is possible with this add-on Selection Sets.

Pose Library Panel

The *Pose Library* panel is used to save, apply, and manage different armature poses. *Pose Libraries* are saved to *Actions*. They are not generally used as actions, but can be converted to and from.

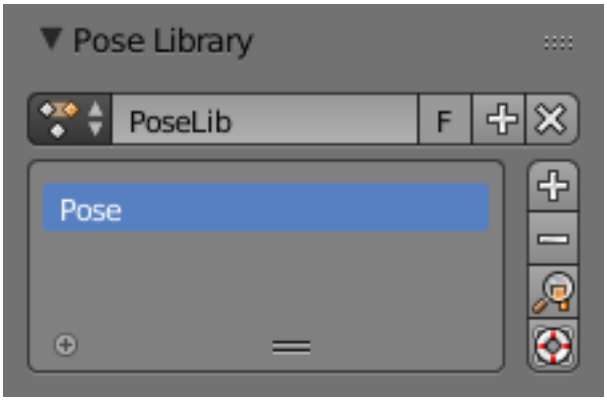


Fig. 2.1189: The Pose Library panel.

Action A *Data-Block Menu* for Actions or Pose Libraries.

Pose Libraries A *List Views & Presets* of poses for the active Pose Library.

Add + If a pose is added a *pose marker* is created.

Add New Adds a new pose to the active Pose Library with the current pose of the armature.

Add New (Current Frame). Will add a pose to the Pose Library based on the current frame selected in the Time line. In contrast to *Add New* and *Replace Existing* which automatically allocate a pose to an action frame.

Replace Existing Replace an existing pose in the active Pose Library with the current pose of the armature.

Apply Pose Apply the active pose to the selected pose bones (magnifying glass icon).

Sanitize Action Makes a action suitable for use as a Pose Library (livesaver icon). This is used to convert an Action to a Pose Library. A pose is added to the Pose Library for each frame with keyframes.

Shortcuts

3D View, Pose Mode. *Pose* → *Pose Library*

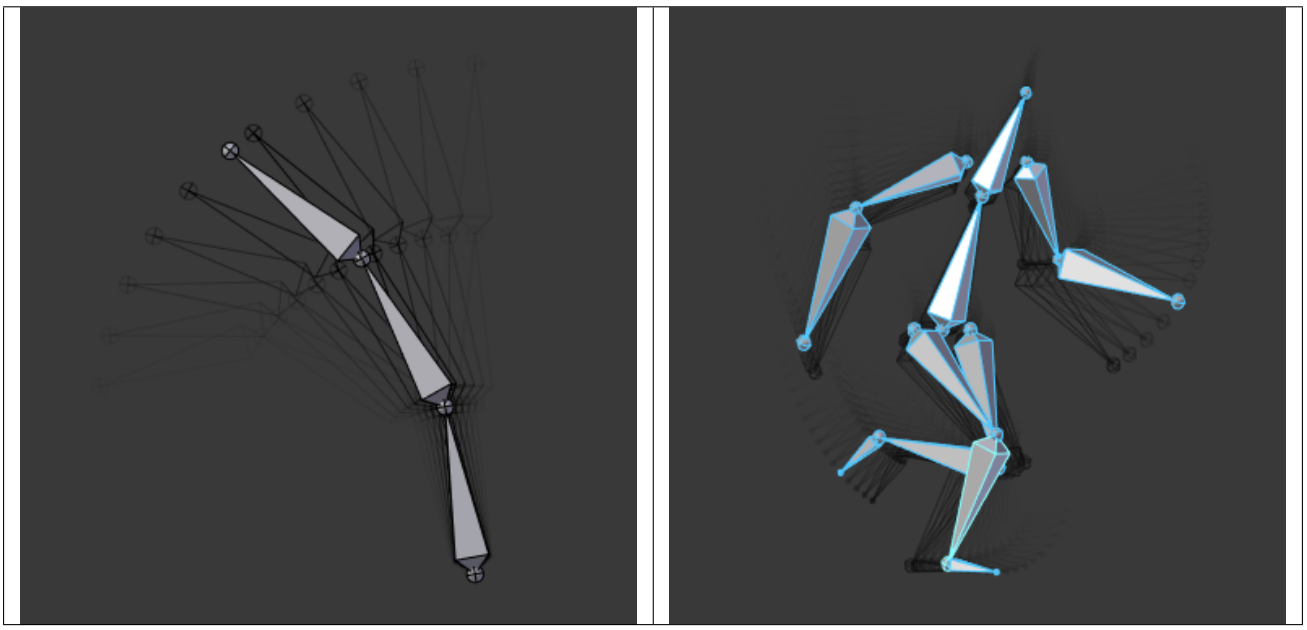
- Browse Poses. `Ctrl-L`.
- Add Pose. `Shift-L`.
- Rename Pose. `Shift-Ctrl-L`.
- Remove Pose. `Alt-L`.

Ghost

Reference

Mode: Pose Mode
Panel: *Armature tab* → *Ghost panel*

Table 2.70: Ghosts examples.



In traditional cartoon creation animators use tracing paper, to see several frames preceding the one they are working on. This allows them to visualize the overall movement of their character, without having to play it back.

Blender features something very similar for armatures in *Pose Mode*: the "ghosts". The ghosts are black outlines (more or less opaque) of the bones as they are at certain frames.

Options

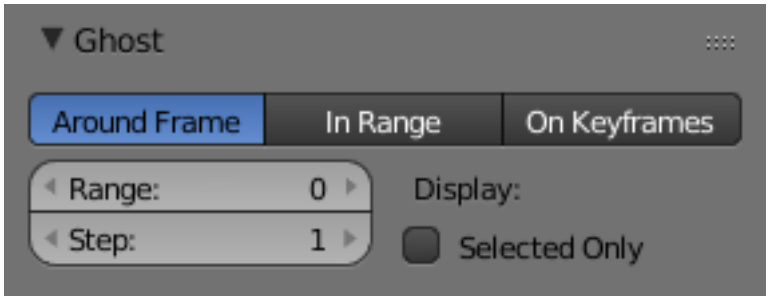


Fig. 2.1190: The Ghost panel.

The ghosts settings are found in the *Armature* tab, only active in *Pose Mode*.

Type

Around Current Frame This will display a given number of ghosts before and after the current frame. The ghosts are shaded from opaque at the current frame, to transparent at the most distant frames.

In Range This will display the ghosts of the armature's bones inside a given range of frames. The ghosts are shaded from transparent for the first frame, to opaque at the last frame. It has four options:

On Keyframes This is very similar to the *In Range* option, but there are ghosts only for keyframes in the armature animation (i.e. frames at which you keyed one or more of the bones). So it has the same options as above, except for the *Step* one (as only keyframes generate ghosts). Oddly, the shading of ghosts is reversed compared to *In Range* - from opaque for the first keyframe, to transparent for the last keyframe.

Range This number button specifies how many ghosts you will have on both "sides" (i.e. a value of 5 will give you ten ghosts, five before the current frame, and five after).

Start, End This number button specifies the start/end frame of the range (exclusive). Note that unfortunately, it cannot take a null or negative value, which means you can only see ghosts starting from frame 2 included...

Step This number button specifies whether you have a ghost for every frame (the default value of 1), or one each two frames, each three frames, etc.

Display

Selected Only When enabled, you will only see the ghosts of selected bones (otherwise, every bone in the armatures has ghosts...)

Finally, these ghosts are also active when playing the animation `Alt-A` – this is only useful with the *Around Current Frame* option, of course...

Note: There is no "global switch" to disable this display feature. To do so, you have to either set *Ghost* to 0 (for *Around Current Frame* option), or the same frame number in both *Start* and *End* (for the two other ghosts types).

Motion Paths

Reference

Mode: Pose Mode
Panel: *Armature tab* → *Motion Paths panel*

Menu: *Pose* → *Motion Paths* → ...
Hotkey: `W-3`, `W-4`

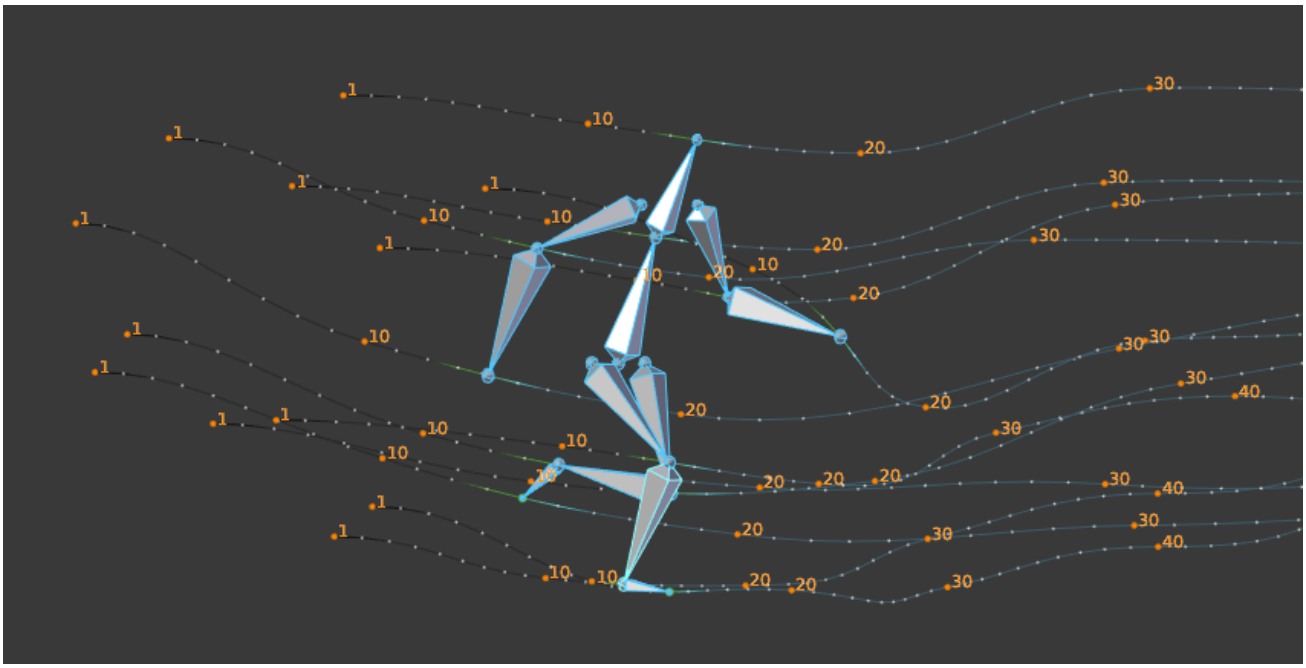


Fig. 2.1191: An example of motion paths.

This feature allows you to visualize as curves the paths of bones' ends (either their tips, by default, or their roots).

Before we look at its options, let us first see how to display/hide these paths. Unlike *Ghost*, you have to do it manually and you have to first select the bones you want to show/hide the motion paths. Then,

- To show the paths (or update them, if needed), click on the *Calculate Path* button in the panel, or, in the 3D Views, select the *Pose* → *Motion Paths* → *Calculate Paths* menu entry (or use the *Specials* pop-up menu, `W-3`).
- To hide the paths, click on the *Clear Paths* button, or, in the 3D Views, do *Pose* → *Motion Paths* → *Clear All Paths*, or `W-4`.

Note: Remember that only selected bones and their paths are affected by these actions!

The paths are drawn in a light shade of gray for unselected bones, and a slightly blueish gray for selected ones. Each frame is materialized by a small white dot on the paths.

As with ghosts, the paths are automatically updated when you edit your poses/keyframes, and they are also active during animation playback. `Alt-A` is only useful when the *Around Current Frame* option is enabled.

Options

Type

Around Frame Around Frame, Display Paths of poses within a fixed number of frames around the current frame. When you enable this button, you get paths for a given number of frames before and after the current one (again, as with ghosts).

In Range In Range, Display Paths of poses within specified range.

Display Range

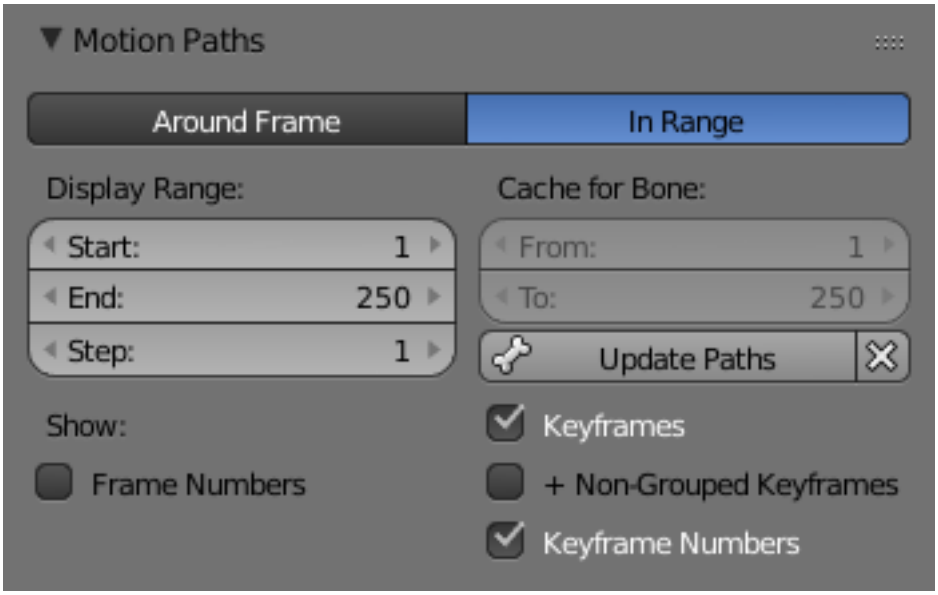


Fig. 2.1192: The Motion Paths Panel.

Before/After Number of frames to show before and after the current frame (only for 'Around Current Frame' Onion-skinning method).

Start/End Starting and Ending frame of range of paths to display/calculate (not for 'Around Current Frame' Onion-skinning method).

Step This is the same as the *Step* for ghosts. It allows you to only display on the path one frame for each *n* ones. Mostly useful when you enable the frame number display (see below), to avoid cluttering the 3D Views.

Cache for Bone From, To

Calculate

Start, End These are the start/end frames of the range in which motion paths are drawn. You have to *Calculate Paths* again if you modify this setting, to update the paths in the 3D Views. Note that unlike with ghosts, the start frame is *inclusive* (i.e. if you set *Start* to 1, you will really see the frame 1 as starting point of the paths...).

Bake Location By default, you get the tips' paths. By changing this setting to Tails, you will get the paths of the bone's roots (remember that in Blender UI, bones' roots are called "heads"...). You have to *Calculate Paths* again if you modify this setting, to update the paths in the 3D Views.

Show

Frame Numbers When enabled, a small number appears next to each frame dot on the path, which is of course the number of the corresponding frame.

Keyframes When enabled, big yellow square dots are drawn on motion paths, materializing the keyframes of their bones (i.e. only the paths of keyed bones at a given frame get a yellow dot at this frame).

+ Non-Grouped Keyframes For bone motion paths, search whole Action for keyframes instead of in group with matching name only (is slower).

Keyframe Numbers When enabled, you will see the numbers of the displayed keyframes, so this option is obviously only valid when *Show Keys* is enabled.

Structure

Armatures mimic real skeletons. They are made out of bones, which are (by default) rigid elements. But you have more possibilities than with real skeletons: In addition to the "natural" rotation of bones, you can also translate and even scale them! And your bones do not have to be connected to each other; they can be completely free if you want. However, the most natural and useful setups imply that some bones are related to others, forming so-called "chains of bones", which create some sort of "limbs" in your armature, as detailed in *Chains of Bones*.

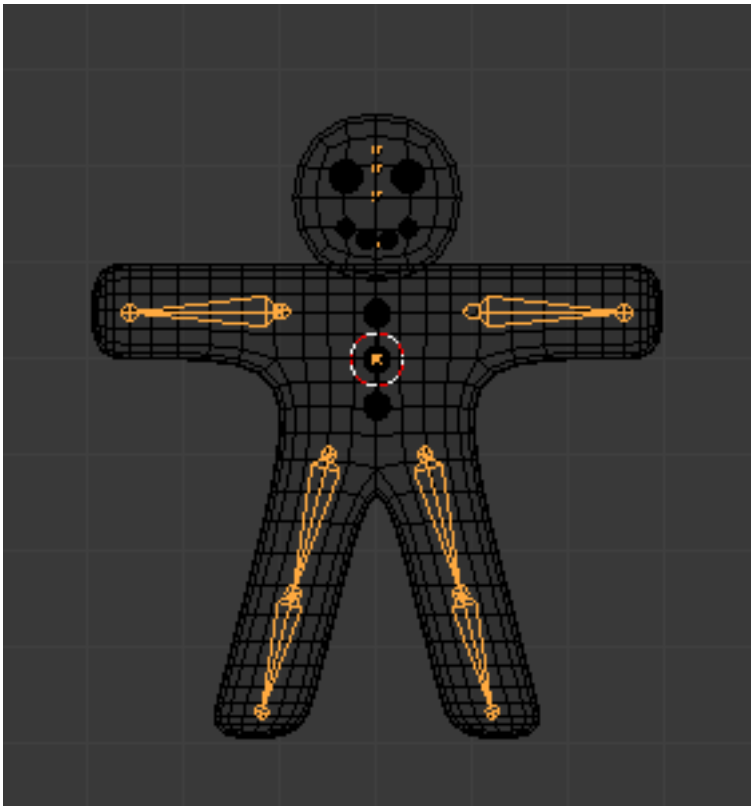


Fig. 2.1193: Example of a very basic armature.

Chains of Bones

The bones inside an armature can be completely independent from each other (i.e. the modification of one bone does not affect the others). But this is not often a useful set up: To create a leg, all bones "after" the thigh bone should move "with" it in a well-coordinated manner. This is exactly what happens in armatures by parenting a bone to the next one in the limb, you create a "chains of bones". These chains can be ramified. For example, five fingers attached to a single "hand" bone.

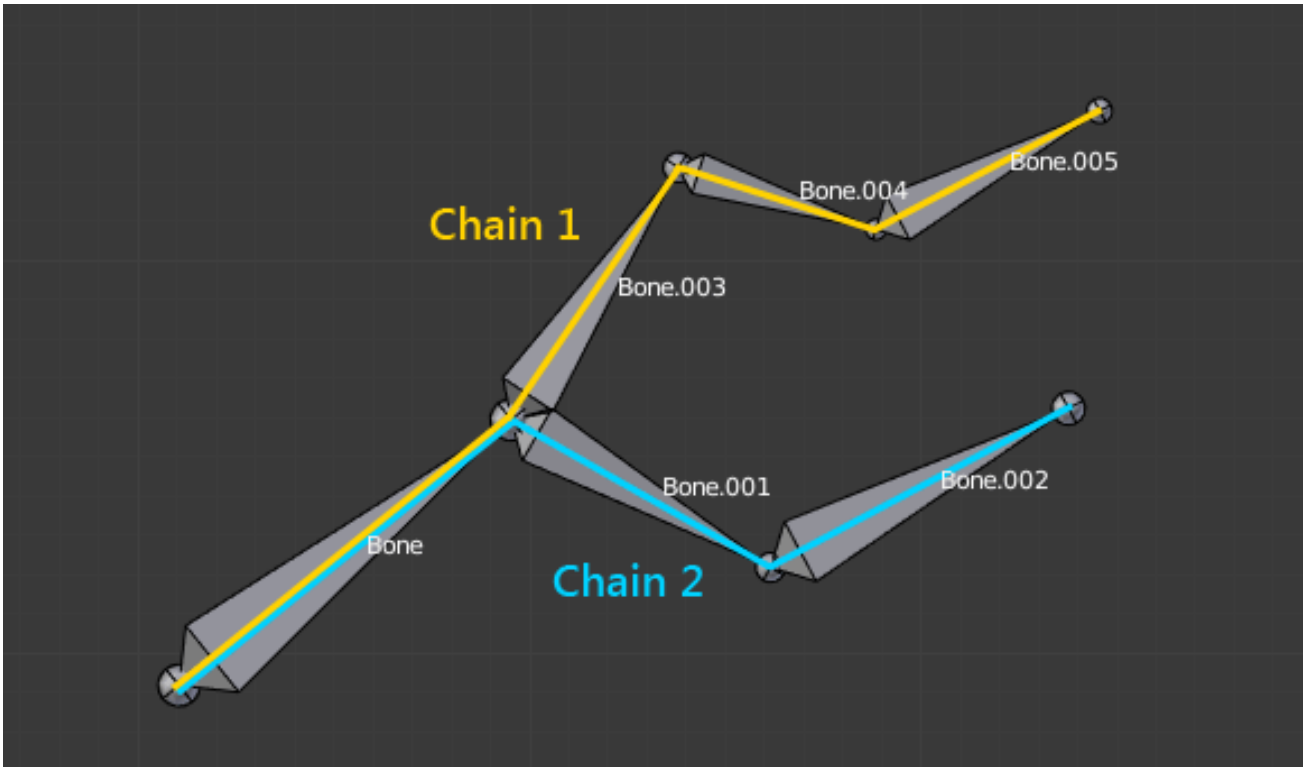


Fig. 2.1194: An armature with two chains of bones.

Bones are chained by linking the tip of the parent to the root of the child. Root and tip can be *connected*, i.e. they are always exactly at the

same point; or they can be *free*, like in a standard
parent-child object relationship.

A given bone can be the parent of several children, and hence be part of several chains at the same time.

The bone at the beginning of a chain is called its *root bone*, and the last bone of a chain is the *tip bone* (do not confuse them with similar names of bones' ends!).

Chains of bones are a particularly important topic in *posing* (especially with the standard *forward kinematics* versus "automatic" *inverse kinematics* posing techniques). You create/edit them in *Edit Mode*, but except in case of connected bones, their relationships have no effect on bone transformations in this mode (i.e. transforming a parent bone will not affect its children).

The easiest way to manage bones relationships is to use the *Relations panel* in the *Bone* tab.

Skinning

Introduction

We have seen in *previous pages* how to design an armature, create chains of bones, etc. Now, having a good rig is not the final goal, unless you want to produce a "Dance Macabre" animation, you will likely want to put some flesh on your skeletons! Surprisingly, "linking" an armature to the object(s) it should transform and/or deform is called the "skinning" process...

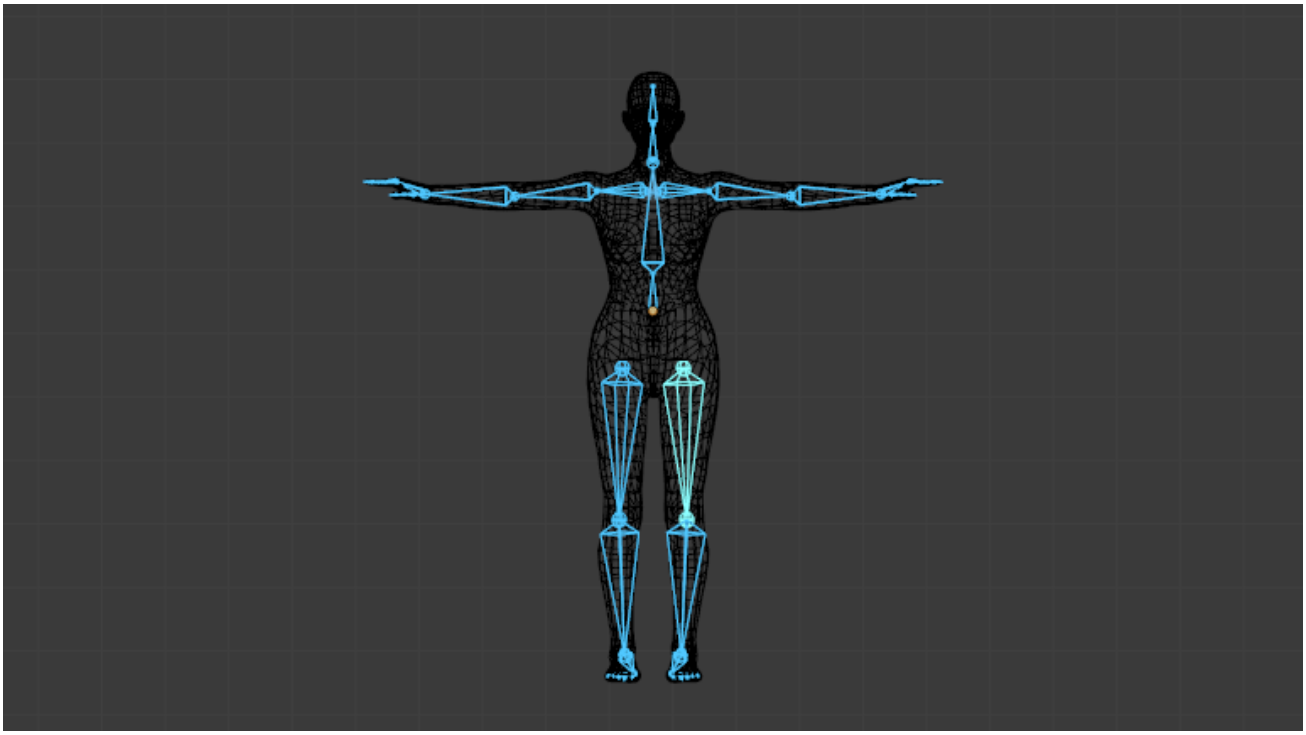


Fig. 2.1195: The human mesh skinned on its armature.

In Blender, you have two main skinning types:

- You can *Parent/Constrain Objects to Bones* - then, when you transform the bones in *Pose Mode*, their "children" objects are also transformed, exactly as with a standard parent/children relationship... The "children" are **never** deformed when using this method.
- You can *Using the Armature modifier on entire Mesh*, and then, some parts of this object to some bones inside this armature. This is the more complex and powerful method, and the only way to really deform the geometry of the object, i.e. to modify its vertices/control points relative positions.

Mesh Skinning

There are some settings necessary for an armature to deform a mesh:

- The mesh object needs to be assigned to the armature object or vice versa.
- The bones of the armature have to be assigned to parts of the mesh or vice versa.

Object Assignment

An object is assigned to an armature by either:

- Adding an *Armature Modifier* to the object and selecting the appropriate armature.
- Parenting the object to the armature (not as flexible but still working).

Bone Influences

The transformation of the mesh can either be done by:

- Using the *bone envelopes*.
- Making *vertex groups* for each bone.

The first option is default and should work automatically. The second option has to be set for each bone and vertex group individually, even though those setting can be mirrored in symmetrical meshes.

For the second option to work the vertex group must have the same name as the bone that controls it.

Skinning to Shapes

In the *previous page*, we saw how to link (parent) whole objects to armature bones – a way to control the transform properties of this object via a rig. However, armatures are much more powerful: they can deform the *shape* of an object (i.e. affect its Object Data data-block, which is its vertices or control points...).

In this case, the child object is parented (skinned) to the whole armature, so that each of its bones controls a part of the "skin" object's geometry. This type of skinning is available for meshes, lattices, curves, surfaces, and texts (with more options for the first two types).

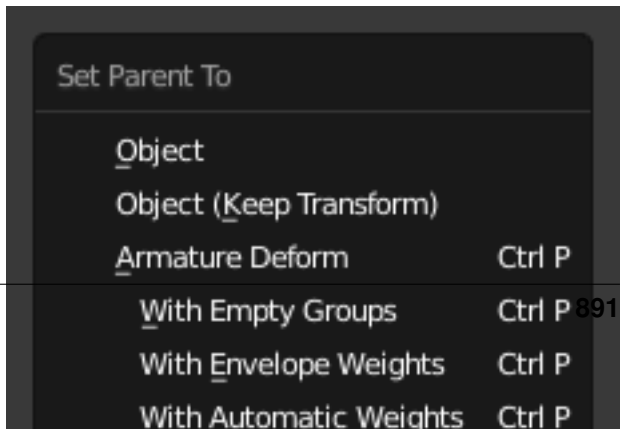
Bones can affect the object's shape in two ways:

- The *Envelope* process is available for all type of skinnable objects. It uses the "proximity" and "influence" of the bones to determine which part of the object they can deform.
- The *Vertex Groups* method is (obviously) reserved to meshes and lattices. One bone only affect the vertices in the *group* having the same name, using vertices' *weights* as influence value. A much more precise method, but also generally longer to set up.

Both methods have some *Common Options*, and can be mixed together.

Parenting to Whole Armatures

But before diving into this, let us talk about the different ways to skin (parent) an object to a whole armature as with *object skinning*, there is an "old parenting" method and a new, more flexible and powerful one, based on modifiers, which allows creation of very complex setups, with objects deformed by several armatures.



For meshes and lattices *only*, you can use the `Ctrl-P` parent shortcut in the 3D Views (after having selected first the "skin"

object, then the armature). The *Make Parent To* menu pops up, select the *Armature* entry. If the skinning object is a lattice, you are done; no more options are available. But with a child mesh, another *Create Vertex Groups?* menu appears, with the following options all regarding the "vertex groups" skinning method:

With Empty Groups will create, if they do not already exist, empty groups, one for each bone in the skinned armature, with these bones' names. Choose this option if you have already created (and weighted) all the vertex groups the mesh requires.

With Envelope Weights will create, as with *Name Groups* option, the needed vertex groups. However, it will also weight them according to the bones' envelope settings (i.e. it will assign to each groups the vertices that are inside its bone's influence area, weighted depending on their distance to this bone).

> **Warning:** This means that if you had defined vertex groups using same names as skinned bones, their content will be completely overridden. You will get the same behavior as if you used the envelopes skinning method, but with vertex groups?

With Automatic Weights Creates, as with *Envelope Weights* option, the needed vertex groups, with vertices assigned and weighted using the newer "bone heat" algorithm.

This "parenting" method will create an *Armature modifier* in the skinning object's modifiers stack. And so, of course, adding an *Armature modifier* to an object is the second, new skinning method (which also works for curves/surfaces/texts...). Follow the above link to read more about this modifier's specific options. Note that there is a way with new *Armature* modifiers to automatically create vertex groups and weight them; see the *Vertex Groups* description below.

> **Warning:** A single object can have several *Armature* modifiers (with e.g. different armatures, or different settings...), working on top of each other, **or** mixing their respective effects (depending whether their *MultiModifier* option is set, see *their description* for more details), and only one "virtual old parenting" one, which will always be at the top of the stack.

Note: Finally that for settings that are present in both the armature's Armature panel and in the objects' Armature modifier panel (namely, Vertex Groups , Envelopes, Quaternion and B-Bone Rest), the modifier ones always override the armature ones. This means that if, for example, you only enable the *Envelopes* deformation method of the armature, and then skin it with an object using an Armature modifier, where only *Vertex Groups* is enabled, the object will only be deformed based on its "bones" vertex groups, ignoring completely the bones' envelopes.

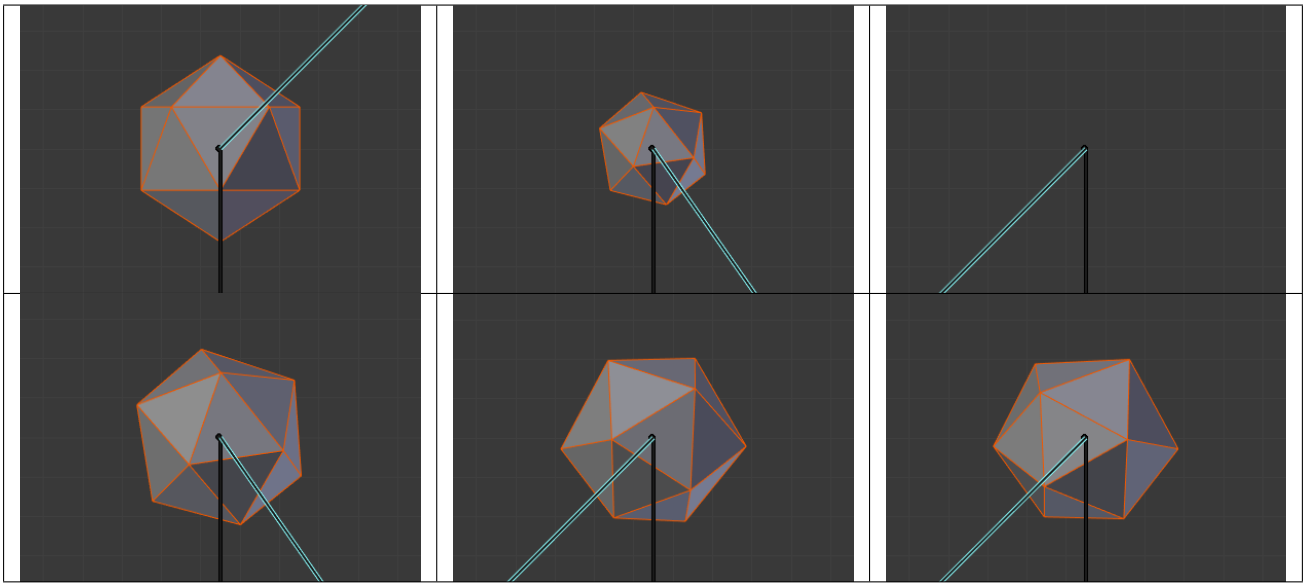
Common Options

There are two armature-global skinning options that are common to both envelopes and vertex groups methods:

Preserve Volume (Armature modifier) This affects the way geometry is deformed, especially at bones' joints, when rotating them.

Without *Preserve Volume*, rotations at joints tend to scale down the neighboring geometry, up to nearly zero at 180 degrees from rest position. With *Preserve Volume*, the geometry is no longer scaled down, but there is a "gap", a discontinuity when reaching 180 degrees from rest position.

Table 2.71: 180.1- rotation, Preserve Volume enabled.



Note: Note that the IcoSphere is deformed using the envelopes method.

Bone Deform Options

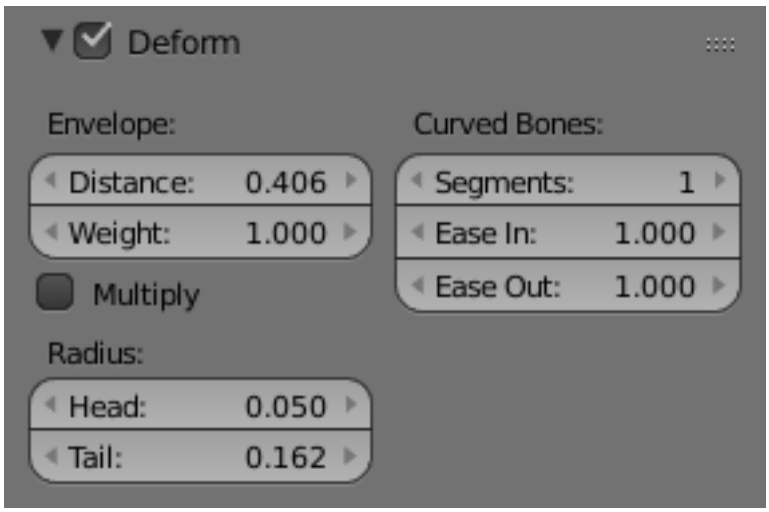


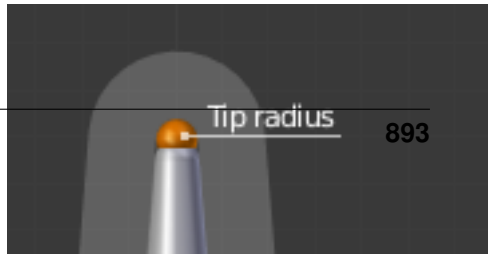
Fig. 2.1203: Bone Deform Options.

The bones also have some deforming options in the Deform panel (*Bone* tab), that you can therefore define independently for each of them.

Deform By disabling this setting (enabled by default), you can completely prevent a bone from deforming the geometry of the skin object.

Envelope

Envelopes is the most general skinning method. It works with all available object types for skinning (meshes, lattices, curves, surfaces and texts). It is



based on proximity between bones and their geometry, each bone having two different areas of influence, shown in the *Envelope* visualization:

- The inside area, materialized by the "solid" part of the bone, and controlled by both root and tip radius. Inside this zone, the geometry if fully affected by the bone.
- The outside area, materialized by the lighter part around the bone, and controlled by the *Distance* setting. Inside this zone, the geometry is less and less affected by the bone as it goes away by following a quadratic decay.

See also:

The *editing pages* for how to edit these properties.

There is also a bone property, *Weight* (Deform panel, in *Edit Mode* only, defaults is set to 1.0), that controls the global influence of the bone over the deformed object, when using the envelopes method. It is only useful for the parts of geometry that are "shared", influenced by more than one bone (generally, at the joints...) - a bone with a high weight will have more influence on the result than one with a low weight... Note that when set to 0.0, it has the same effect as disabling the *Deform* option.

Mult Short for 'Multiply'. This option controls how the two deforming methods interact, when they are both enabled. By default, when they are both active, all vertices belonging to at least one vertex group are only deformed through the vertex groups method. The other "orphan" vertices being handled by the envelopes one. When you enable this option, the "deformation influence" that this bone would have on a vertex (based from its envelope settings) is multiplied with this vertex's weight in the corresponding vertex group. In other words, the vertex groups method is further "weighted" by the envelopes method.

Radius Set the radius for the head and the tail of envelope bones.

Curved Bone

Curved Bones (previously known as B-bones) allow you make bones act like Bézier curve segments, which results in smoother deformations for longer bones.

See also:

The *editing pages* for how to edit these properties.

Vertex Groups

Vertex groups skinning method is only available for meshes and lattices. Which are the only objects having *vertex groups*. Its principle is very simple: each bone only affects vertices belonging to a vertex group having the same name as the bone. So if you have e.g. a `forearm` bone, it will only affect the `forearm` vertex group of its skin object(s).

The influence of one bone on a given vertex is controlled by the weight of this vertex in the relevant group. Thus, the *Weight Paint Mode*. `Ctrl-Tab`, if a mesh is selected is most useful here, to easily set/adjust the vertices' weights.

However, you have a few goodies when weight-painting a mesh already parented to (skinning) an armature. For these to work, you must:

1. Select the armature.
2. Switch to *Pose Mode* `Ctrl-Tab`.
3. Select the mesh to weight.
4. Hit again `Ctrl-Tab` to switch to *Weight Paint Mode*.

Now, when you select a bone of the armature (which remained in *Pose Mode*), you automatically activate the corresponding vertex group of the mesh – Very handy! Obviously, you can only select one bone at a time in this mode (so `Shift-LMB` clicking does not work).

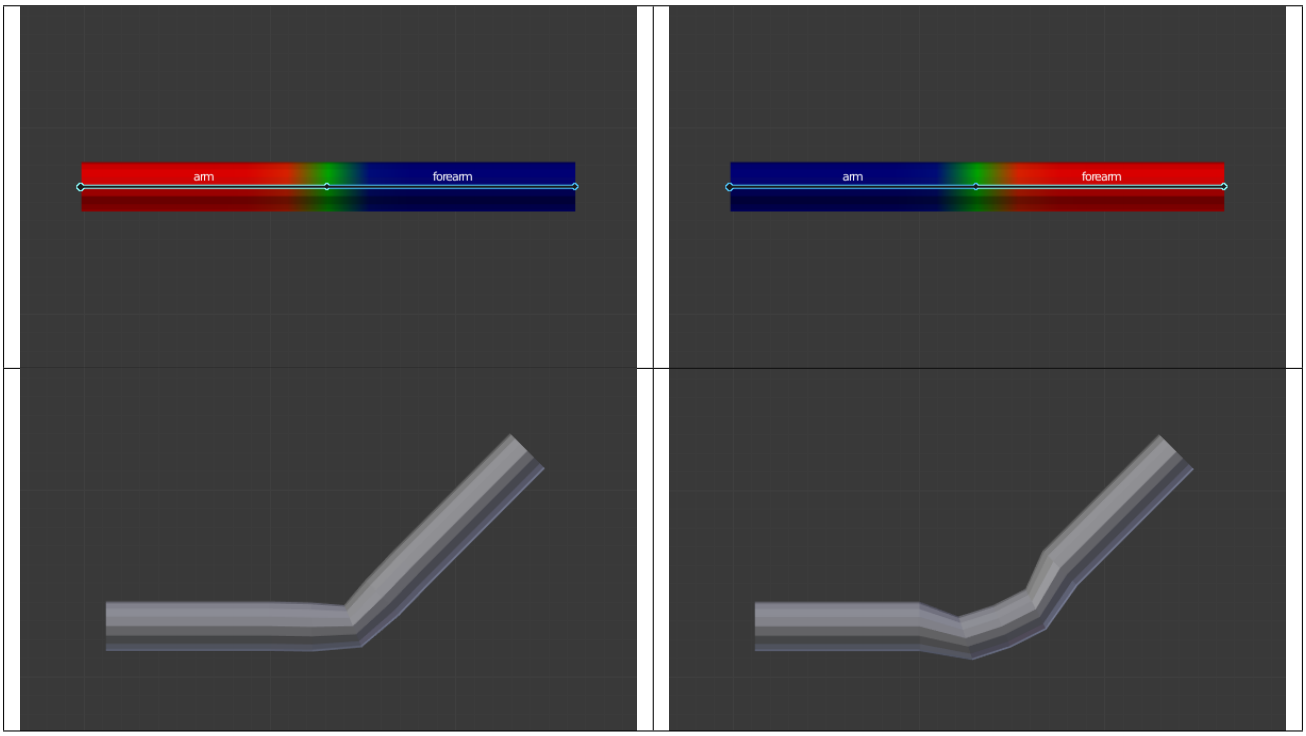
This way, you can also apply to the active bone/vertex group one of the same "auto-weighting" methods as available when doing an "old-parenting" to armature `Ctrl-P`:

- Select the bone (and hence the vertex group) you want.
- Hit `W`, and in the *Specials* menu that pops up, choose either *Apply Bone Envelopes to Vertex Groups* or *Apply Bone Heat Weights to Vertex Groups*. Once again, even though these names are plural, you can only affect *one* vertex group's weights at a time with these options.

To automatically weight multiple bones, you can simply:

- `Ctrl-Tab` out of Weight Paint Mode
- Select the Armature. It should be in Pose Mode. If it is not, go `Ctrl-Tab`
- Select multiple bones `Shift-LMB` or press `A` (once or twice).
- Select Mesh again
- If not in weight paint already, toggle back into `Ctrl-Tab`
- Use the `W` menu to automatic weight. This will weight all the bones you selected in Pose Mode.

Table 2.72: The same pose, but using envelopes method rather that vertex groups.



Obviously, the same vertex can belong to several groups, and hence be affected by several bones, with a fine tuning of each bone's influence using these vertex weights. Quite useful when you want to have a smooth joint. For example, when you skin an elbow, the upperarm vertex group contains the vertices of this part at full weight (*1.0*), and when reaching the elbow area, these weights decrease progressively to *0.0'* when reaching the forearm zone and vice versa for the forearm group weights... Of course, this is a very raw example skinning a realistic joint is a big job, as you have to carefully find good weights for each vertex, to have the most realistic behavior, when bending – and this is not an easy thing!

Armature Deform Parent

In Blender Armature Object Types are usually used to associate certain bones of an Armature to certain parts of a Mesh Object Types Mesh Geometry. You are then able to move the Armature Bones and the Mesh Object will deform.

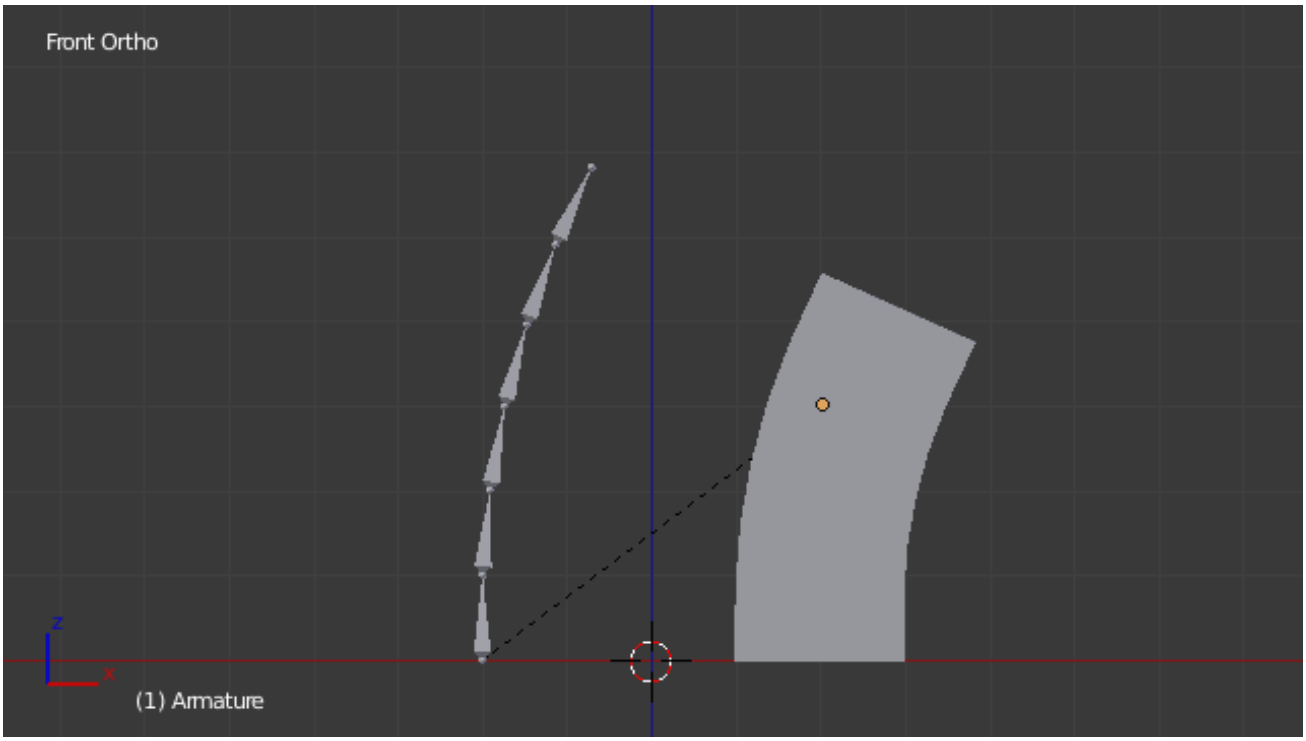


Fig. 2.1209: Bone associated with Mesh Object.

Armature Deform Parenting is one of the most flexible ways of associating Bones in an Armature to another Object, it gives a lot of freedom but that comes at the price of a little complexity, as there are multiple steps involved in setting up *Armature Deform Parenting* such that deformations are actually carried out.

Blender has several different ways of Parenting an Armature to an object, most of them can automate several of the steps involved, but all of them ultimately do all the steps we describe for Armature Deform Parenting.

Using the Armature Deform Parenting operator is the first step in setting up the relationship between an Armature Object and its Child Objects.

To use Armature Deform Parenting you must first select all the Child Objects that will be influenced by the Armature and then lastly, select the Armature Object itself. Once all the Child Objects and the Armature Object are selected press `Ctrl-P` and select Armature Deform in the Set Parent To pop-up menu.

Once this is done the Armature Object will be the Parent Object of all the other Child Objects, also we have informed Blender that the Bones of the Armature Object can be associated with specific parts of the Child Objects so that they can be directly manipulated by the Bones.

At this point however, all Blender knows is that the Bones of the Armature could be used to alter the Child Objects, we have not yet told Blender which Bones can alter which Child Objects or by how much.

To do that we must individually select each Child Object individually and toggle into Edit Mode on that Child Object. Once in Edit Mode we can then select the vertices we want to be influenced by the Bones in the Armature. Then with the vertices still selected navigate to *Properties Editor* → *Object Data* → *Vertex Groups* and create a new Vertex Group with the same name as the Bone that you want the selected vertices to be influenced by.

Once the Vertex Group has been created we then assign the selected vertices to the Vertex Group by clicking the Assign Button. By default when selected vertices are assigned to a Vertex Group they will have an Influence Weight of 1.0 This means that they are fully influenced when a Bone they are associated with is moved, if the Influence Weight had been 0.5 then when the bone moves the vertices would only move half as much.

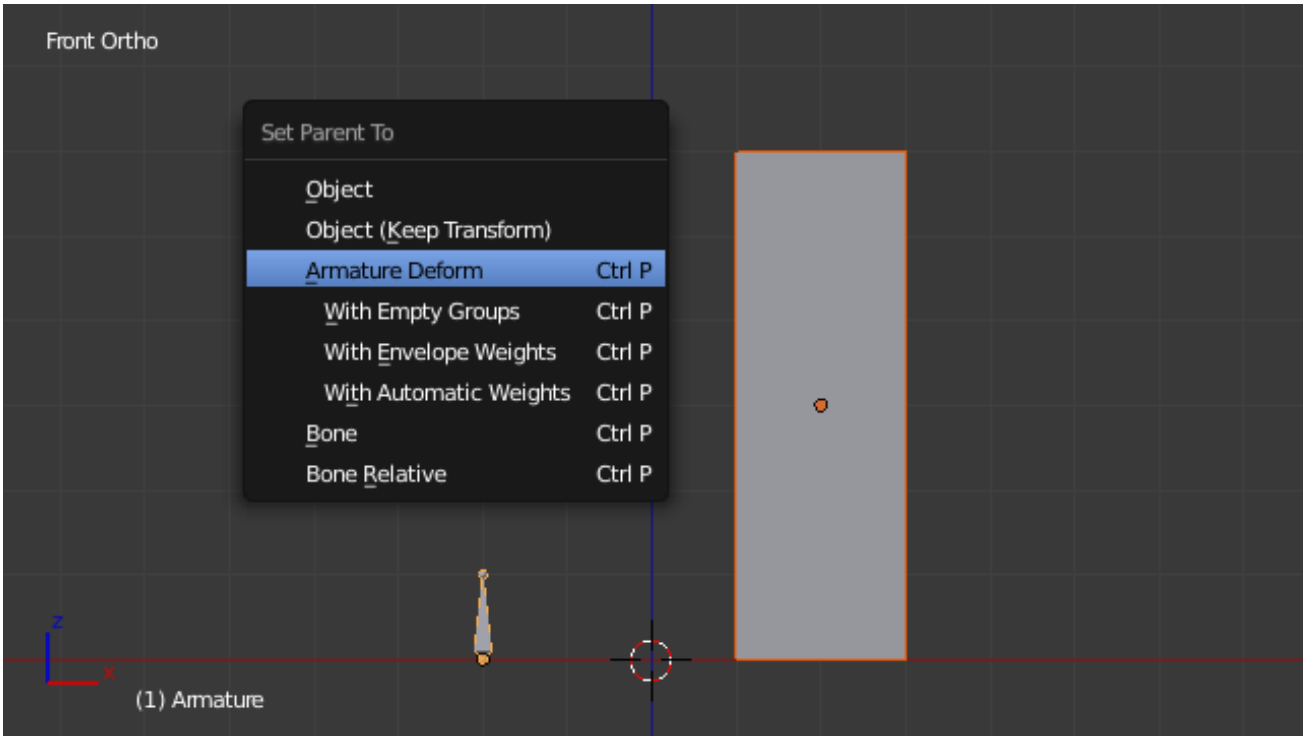


Fig. 2.1210: Set Parent To menu with Armature Deform Parenting option highlighted.

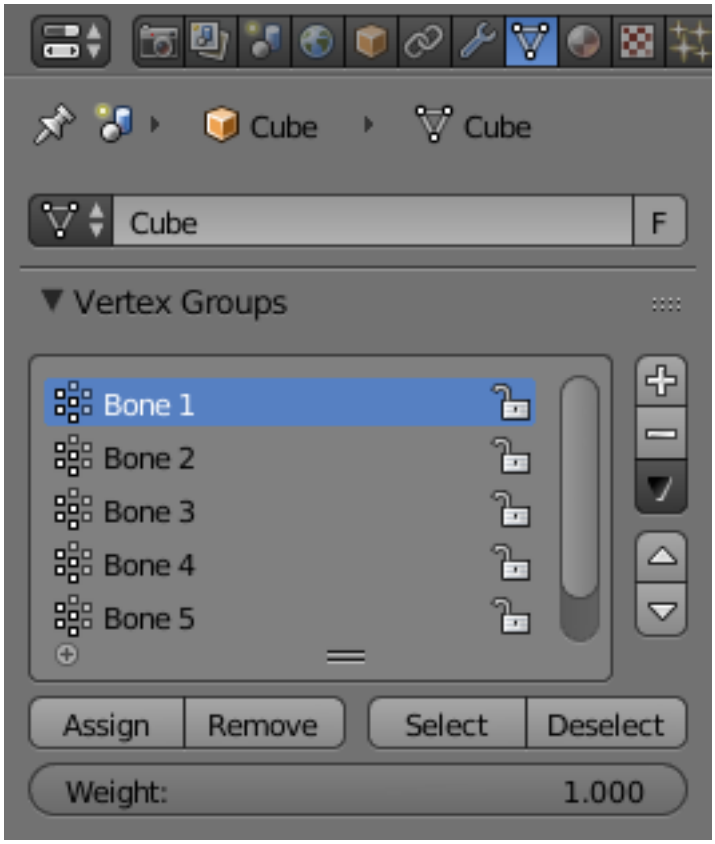


Fig. 2.1211: Vertex groups panel with Assign Button and influence Weight Slider highlighted.

Once all these steps have been carried out, the Bones of the Armature Object should be associated with the Vertex Groups with the same names as the Bones. You can then select the Armature Object and switch to Pose Mode in the *3D View Editor Header → Mode Select* menu.

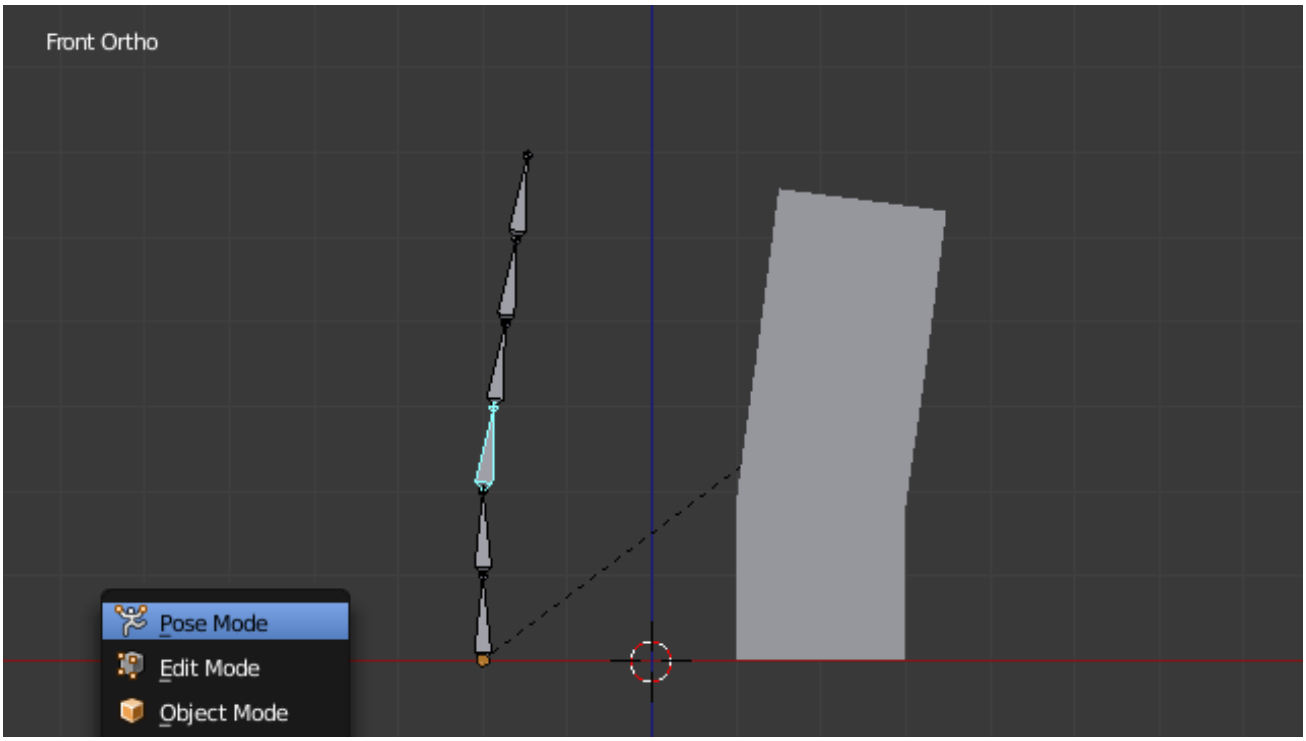


Fig. 2.1212: Armature Bone in Pose Mode affecting the Mesh Object.
The bone is highlighted in Cyan.

Once in Pose Mode transforming one of the Bones of the Armature that has been associated with vertices of an object will result in those vertices also being transformed.

Armature Deform Parent With Empty Groups

The Armature Deform With Empty Groups parenting method works in almost the same way as Armature Deform parenting with one difference. That difference is that when you parent a Child Object to an Armature Object the names of the bones in the armature are copied to the Child Objects in the form of newly created Vertex Groups, one for each different deforming armature bone name. The newly created Vertex Groups will be empty this means they will not have any vertices assigned to those Vertex Groups. You still must manually select the vertices and assign them to a particular Vertex Group of your choosing to have bones in the armature influence them.

For example, if you have an Armature Object which consists of three bones named "BoneA", "BoneB" and "BoneC" and Cube Mesh Object type called "Cube". If you parent the Cube Child Object to the Armature Parent Object the Cube will get three new Vertex Groups created on it called "BoneA", "BoneB" and "BoneC". Notice that each Vertex Group is empty.

Bones in an Armature can be generally classified into two different types:

- Deforming Bones
- Control Bones

Deforming Bones Are bones which when transformed will result in vertices associated with them also transforming in a similar way. Deforming Bones are directly involved in altering the positions of vertices associated with their bones.

Control Bones Are Bones which act in a similar way to switches, in that, they control how other bones or objects react when they are transformed. A Control Bone could for example act as a sliding switch control when the bone is in one position to the left it could indicate to other bones that they react in a particular way when transformed, when the

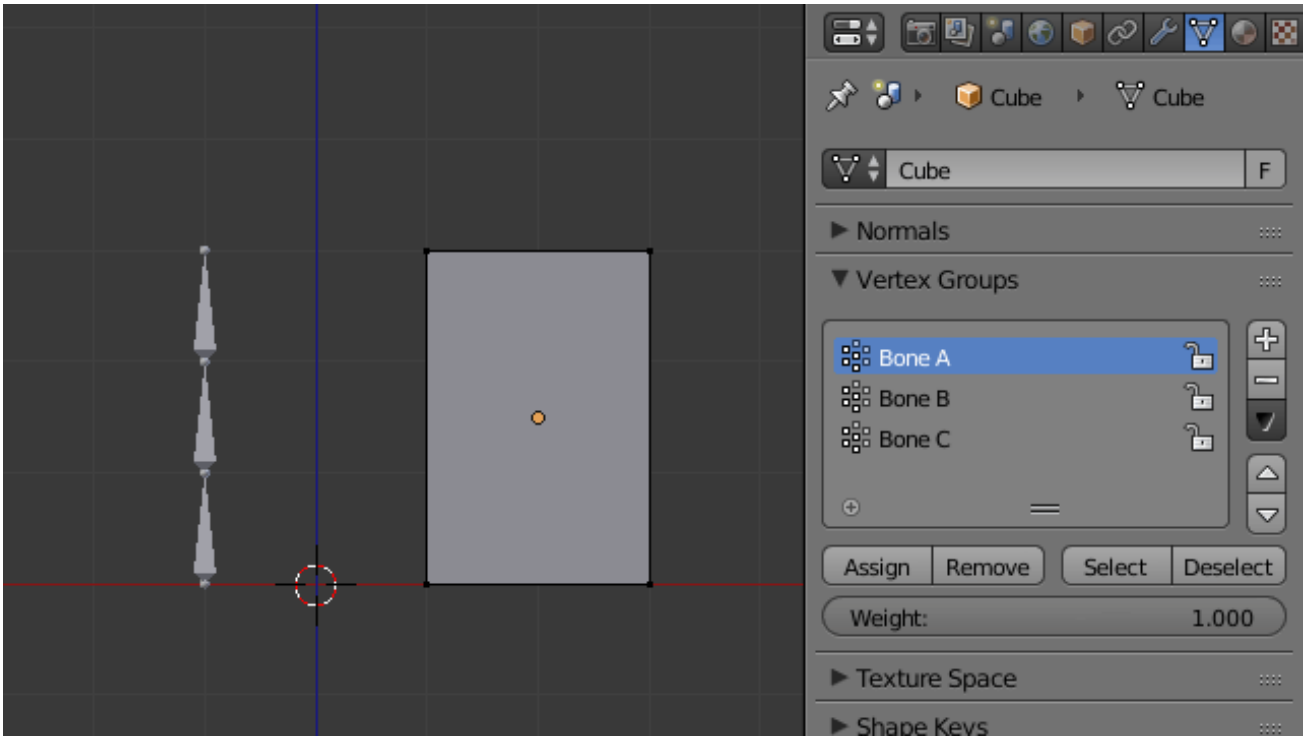


Fig. 2.1213: Cube in Edit Mode using Armature Deform with empty groups.

Control Bone is positioned to the right, transforming other bones or objects could do something completely different. Control Bones are not directly used to alter the positions of vertices, in fact, Control Bones often have no vertices directly associated with themselves.

When using the Armature Deform With Empty Groups parenting method Vertex Groups on the Child Object will only be created for Armature Bones which are setup as Deforming Bone types. If a Bone is a Control Bone no Vertex Group will be created on the Child Object for that bone.

To check whether a particular bone in an armature is a Deforming Bone simply switch to Pose or Edit Mode on the armature and select the bone you are interested in by RMB it. Once the bone of interest is selected navigate to *Properties Editor* → *Bone* → *Deform Panel* and check if the Deform tickable option is ticked or not. If it is the selected bone is a Deforming Bone, otherwise, it is a Control Bone.

Armature Deform With Automatic Weights

Armature Deform With Automatic Weights parenting feature does everything Armature Deform With Empty Groups does with one extra thing. That extra thing is that unlike Armature Deform With Empty Groups which leaves the automatically created Vertex Groups empty with no vertices assigned to them; Armature Deform With Automatic Weight will try to calculate how much Influence Weight a particular Armature Bone would have on a certain collection of vertices based on the distance from those vertices to a particular Armature Bone.

Once Blender has calculated the Influence Weight vertices should have it will assign that Influence Weight to the Vertex Groups that were previously created automatically by Blender on the Child Object when Armature Deform With Automatic Weights parenting command was carried out.

If all went well it should be possible to select the Armature Object switch it into Pose Mode and transform the bones of the Armature and the Child Object should deform in response. Unlike Armature Deform parenting you will not have to create Vertex Groups on the Child Object, neither will you have to assign Influences Weights to those Vertex Groups, Blender will try to do it for you.

To activate Armature Deform With Automatic Weights you must be in Object Mode or Pose Mode, then select all the Child Objects (usually Mesh Object Types) and lastly select the Armature Object; Once done press Ctrl-P and select the Armature Deform With Automatic Weights from the Set Parent To pop-up menu.

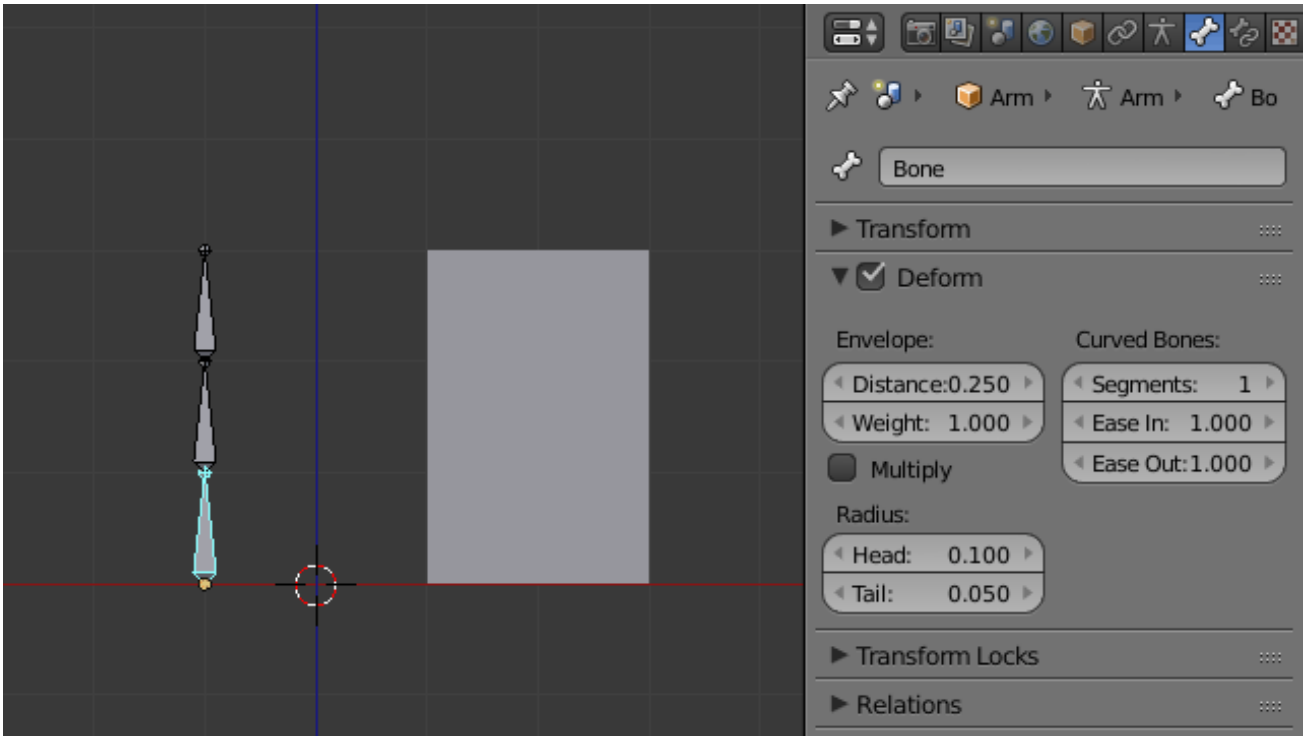


Fig. 2.1214: Three Bone Armature in *Pose* Mode with 1st bone selected.

This method of parenting is certainly easier setup but it can often lead to Armatures which do not deform Child Objects in ways you would want as Blender can get a little confused when it comes to determining which Bones should influence certain vertices when calculating Influence Weights for more complex armatures and Child Objects. Symptoms of this confusion are that when transforming the Armature Object in Pose Mode parts of the Child Objects do not deform as you expect; If Blender does not give you the results you require you will have to manually alter the Influence Weights of vertices in relation to the Vertex Groups they belong to and have influence in.

Armature Deform With Envelope Weights

Works in a similar way to Armature Deform With Automatic Weights in that it will create Vertex Groups on the Child Objects that have names matching those of the Parent Object Armature Bones. The created Vertex Groups will then be assigned Influence Weights. The major difference is in the way those Influence Weights are calculated.

Influence Weights that are calculated when using Armature Deform With Envelope Weights parenting are calculated entirely visually using Bone Envelopes.

Fig. *Single Armature Bone in Edit Mode with Envelope Weight display enabled.* shows a single Armature Bone in Edit Mode with Envelope Weight activated. The gray semi-transparent volume around the bone is the Bone Envelope.

Any Child Object that has vertices inside the volume of the Bone Envelope will be influenced by the Parent Object Armature when the Armature Deform With Envelope Weights operator is used. Any vertices outside the Bone Envelope volume will not be influenced. When the bones are transformed in Pose Mode the results are very different.

The default size of the Bone Envelope volume does not extend very far from the surface of a bone; You can alter the size of the Bone Envelope volume by clicking on the body of the bone you want to alter, switch to Edit Mode or Pose Mode and then pressing `Ctrl-Alt-S` then drag your mouse left or right and the Bone Envelope volume will alter accordingly.

You can also alter the Bone Envelope volume by selecting the Bone you wish to alter and switching to Edit Mode or Pose Mode, then navigate to *Properties Editor* → *Bone* → *Deform* → *Envelope* → *Distance* then enter a new value into it.

Altering the Bone Envelope volume does not alter the size of the Armature Bone just the range within which it can influence vertices of Child Objects.

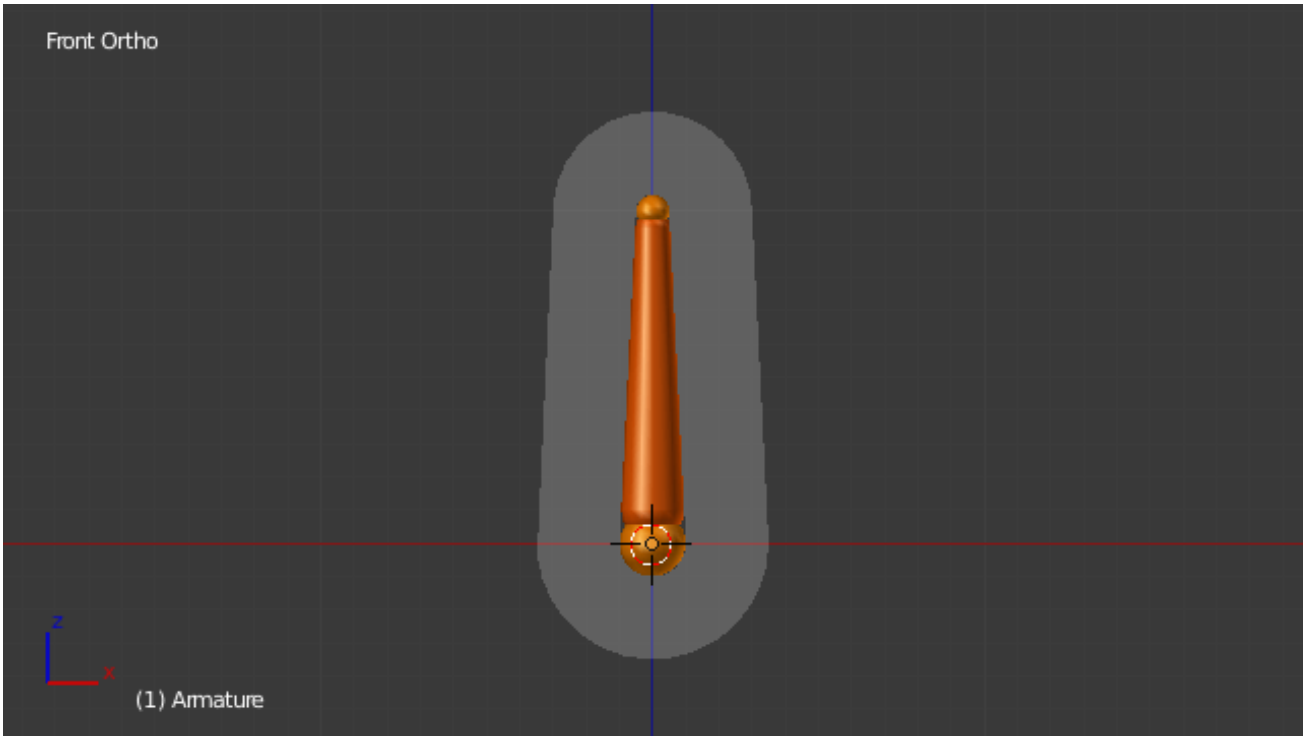


Fig. 2.1215: Single Armature Bone in Edit Mode with Envelope Weight display enabled.
The gray volume around the bone is the Bone Envelope.

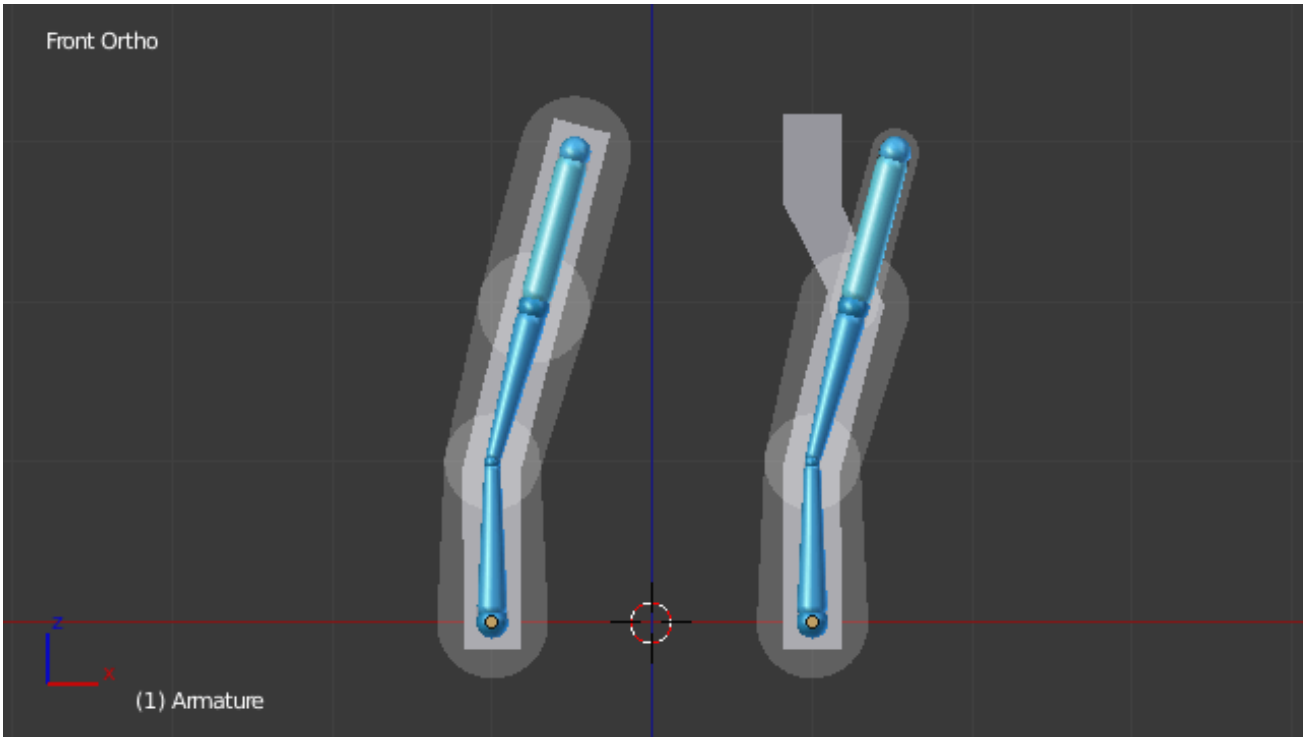


Fig. 2.1216: Two sets of Armatures each with three bones.

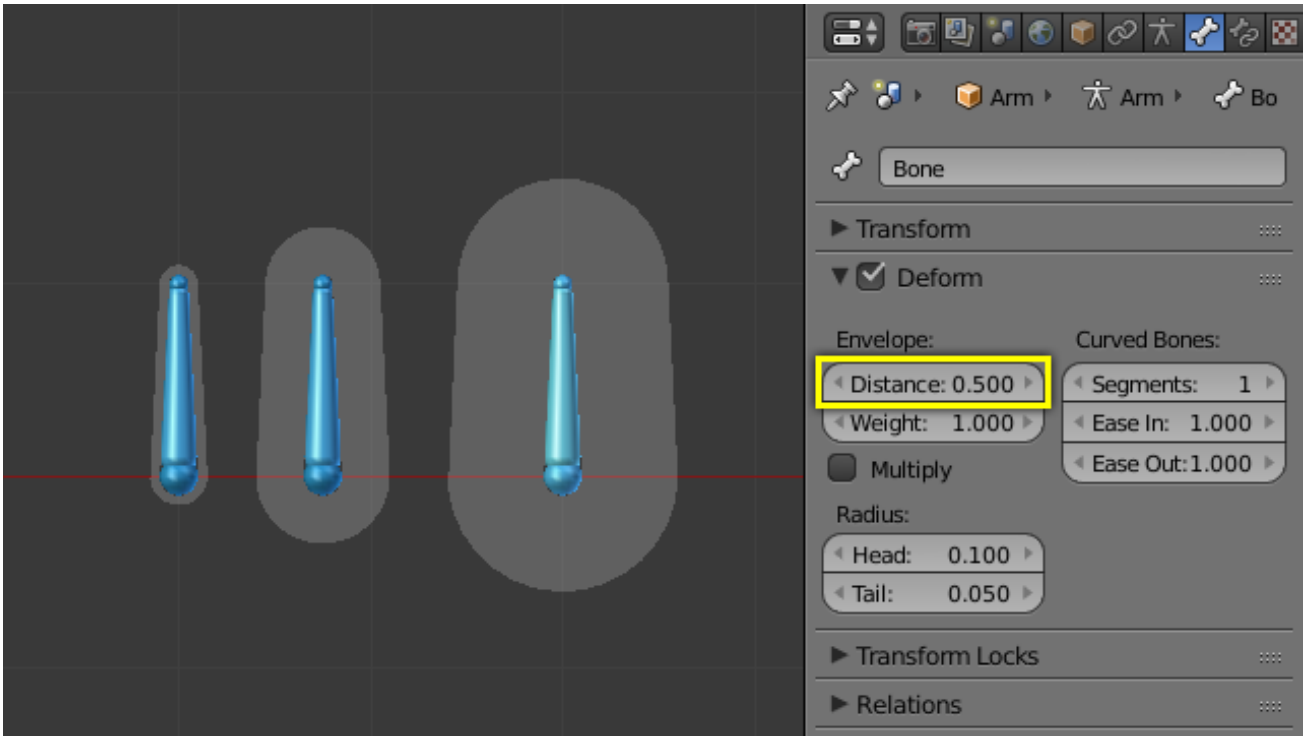


Fig. 2.1217: Single Armature Bone with various different Bone Envelope sizes.
Envelope distance fields highlighted.

You can alter the radius that a bone has by selecting the head, body or tail parts of a bone while in Edit Mode, and then press `Alt-S` and move the mouse left or right. This will make the selected bone fatter or thinner without altering the thickness of the Bone Envelope volume.

You can also alter the bone radius by selecting the tail or head of the bone you wish to alter and switching to Edit Mode, then navigate to *Properties Editor* → *Bone* → *Deform* → *Radius Section* and entering new values for the *Tail* and *Head* fields.

Note: If you alter the Bone Envelope volume of a bone so that you can have it include/exclude certain vertices after you have already used Armature Deform With Envelope Weights, by default, the newly included/excluded vertices will not be affected by the change. When using Armature Deform With Envelope Weights it only calculates which vertices will be affected by the Bone Envelope volume at the time of parenting, at which point it creates the required named Vertex Groups and assigns vertices to them as required. If you want any vertices to take account of the new Bone Envelope volume size you will have to carry out the Armature Deform With Envelope Weights parenting again; In fact, all parenting used in the Set Parent To pop-up menu which tries to automatically assign vertices to Vertex Groups works like this.

Retargeting

Posing

Introduction

Once your armature is *skinned* by the needed object(s), you can start to pose it. Basically, by transforming its bones, you deform or transform the skin object(s). But you do not do that in *Edit Mode* – remember that in this mode, you edit the default, base, "rest" position of your armature. You cannot use the *Object Mode* either, as here you can only transform whole objects...

So, armatures in Blender have a third mode, *Pose*, dedicated to this process. It is a sort of "object mode for bones". In rest position (as edited in *Edit Mode*), each bone has its own position/rotation/scale to neutral values (i.e. 0.0 for position and

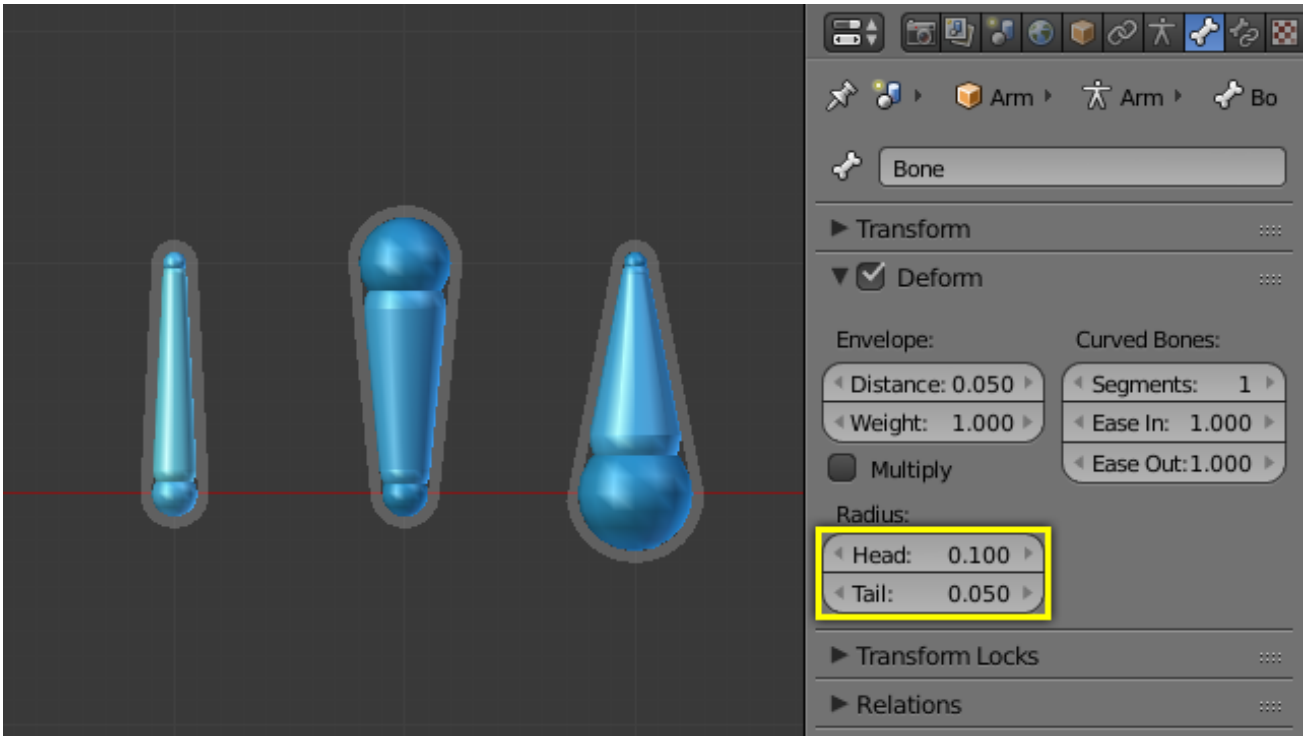


Fig. 2.1218: Three Armature Bones all using Envelope Weight.
The 1st with a default radius value, the two others with differing Tail and Head radius values.

rotation, and 1.0 for scale). Hence, when you edit a bone in *Pose Mode*, you create an offset in its transform properties, from its rest position – this is quite similar to *meshes relative shape keys*, in fact.

Posing Section Overview

In this section, we will see:

- How to *select and edit bones* in this mode.
- How to *use pose library*.
- How to *use constraints* to control your bones' DoF (degrees of freedom).
- How to *use inverse kinematics features*.
- How to *use the Spline inverse kinematics features*.

Even though it might be used for completely static purposes, posing is heavily connected with *animation features and techniques*.

In this part, we will try to focus on animation-independent posing, but this is not always possible. So if you know nothing about animation in Blender, it might be a good idea to read the *animation features and techniques* chapter first, and then come back here.

Visualization

Bone State Colors

The color of the bones are based on their state. There are six different color codes, ordered here by precedence (i.e. the bone will be of the color of the bottommost valid state):

- Gray: Default.

- Blue wireframe: in Pose Mode.
- Green: with Constraint.
- Yellow: with *IK Solver constraint*.
- Orange: with Targetless Solver constraint.

Note: When *Bone Groups* colors are enabled, the state colors will be overridden.

Selecting

Selection in *Pose Mode* is very similar to the one in *Edit Mode*, with a few specificities:

You can only select *whole bones* in *Pose Mode*, not roots/tips...

Grouped

You can select bones based on their group and/or layer, through the *Select Grouped* pop-up menu `Shift-G`:

- To select all bones belonging to the same group(s) as the selected ones, use the *In Same Group* entry `Shift-G-Numpad1`.
- To select all bones belonging to the same layer(s) as the selected ones, use the *In Same Layer* entry `Shift-G-Numpad2`.

Editing

In *Pose Mode*, bones behave like objects. So the transform actions (grab/rotate/scale, etc.) are very similar to the same ones in *Object* mode (all available ones are regrouped in the *Pose* → *Transform* sub-menu). However, there are some important specificities:

- Bones' relationships are crucial (see *Parenting*).
- The "transform center" of a given bone (i.e. its default pivot point, when it is the only selected one) is *its root*. Note by the way that some pivot point options seem to not work properly, In fact, except for the *3D Cursor* one, all others appear to always use the median point of the selection (and not e.g. the active bone's root when *Active Object* is selected, etc.).

Basic Posing

As previously noted, bones' transformations are performed based on the *Rest Position* of the armature, which is its state as defined in *Edit Mode*. This means that in rest position, in *Pose Mode*, each bone has a scale of 1.0, and null rotation and position (as you can see it in the *Transform* panel, in the 3D Views, `N`).

Moreover, the local space for these actions is the bone's own one (visible when you enable the *Axes* option of the *Armature* panel). This is especially important when using axis locking, for example, there is no specific "bone roll" tool in *Pose Mode*, as you can rotate around the bone's main axis just by locking on the local Y axis `R-Y-Y`... This also works with several bones selected; each one is locked to its own local axis!

When you pose your armature, you are supposed to have one or more objects skinned on it! And obviously, when you transform a bone in *Pose Mode*, its related objects or object's shape is moved/deformed accordingly, in real time.

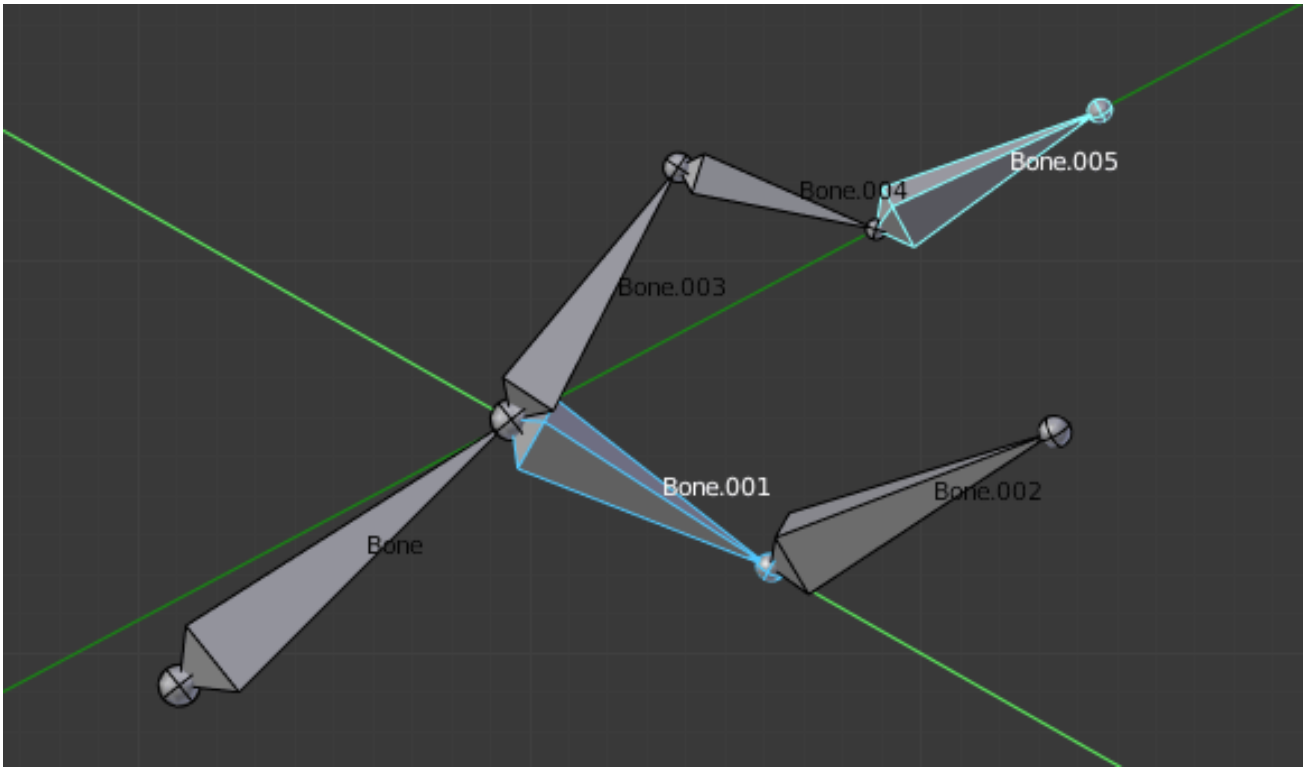


Fig. 2.1220: An example of locally-Y-axis locked rotation, with two bones selected. Note that the two green lines materializing the axes are centered on the armature's center, and not each bone's root...

Unfortunately, if you have a complex rig set-up and/or a heavy skin object, this might produce lag, and make interactive editing very painful. If you experience such troubles, try enabling the *Delay Deform* button of the *Armature* panel the skin objects will only be updated once you validate the transform operation.

Auto IK

The auto IK option in the tool shelf enables a temporary IK constraint when posing bones. The chain acts from the tip of the selected bone to root of the uppermost parent bone. Note that this mode lacks options, and only works by applying the resulting transform to the bones in the chain.

Clear Transform

Once you have transformed some bones, if you want to return to their rest position, just clear their transformations (usual `Alt-G`/`Alt-R`/`Alt-S` shortcuts, or *Pose* → *Clear Transform* → *Clear User Transform*, `W-5`, to clear everything at once... - commands also available in the *Pose* → *Clear Transform* sub-menu).

Note that in *Envelope* visualization, `Alt-S` does not clear the scale, but rather scales the *Distance* influence area of the selected bones (also available through the *Pose* → *Scale Envelope Distance* menu entry, which is only effective in *Envelope* visualization, even though it is always available...).

Apply

Conversely, you may define the current pose as the new rest position (i.e. "apply" current transformations to the *Edit Mode*), using the *Pose* → *Apply Pose as Restpose* menu entry (or `Ctrl-A` and confirm the pop-up menu). When you do

so, the skinned objects/geometry is **also** reset to its default, undeformed state, which generally means you will have to skin it again.

In-Betweens

There are several tools for editing poses in an animation.

Relax Pose

Reference

Mode: Pose Mode
Menu: *Pose* → *In-Betweens* → *Relax Pose*, `Alt-E`

Relax pose is somewhat related to the above topic, but it is only useful with keyframed bones (see the *animation chapter*). When you edit such a bone (and hence take it "away" from its "keyed position"), using this command will progressively "bring it back" to its "keyed position", with smaller and smaller steps as it comes near it.

Push Pose

Reference

Mode: Pose Mode
Menu: *Pose* → *In-Betweens* → *Relax Pose*, `Ctrl-E`

Push pose exaggerates the current pose.

Breakdowner

Reference

Mode: Pose Mode
Menu: *Pose* → *In-Betweens* → *Pose Breakdowner*, `Shift-E`

Creates a suitable breakdown pose on the current frame.

There are also in *Pose Mode* a bunch of armature-specific editing options/tools, like *auto-bones naming*, *properties switching/enabling/disabling*, etc., that we already described in the armature editing pages. See the links above...

Copy/Paste Pose

Reference

Mode: Pose Mode

Panel: *3D View* header ()
Menu: *Pose* → *Copy Current Pose*, *Pose* → *Paste Pose*, *Pose* → *Paste Flipped Pose*

Blender allows you to copy and paste a pose, either through the *Pose* menu, or directly using the three "copy/paste" buttons found at the right part of the 3D Views header:

Pose* → *Copy Current Pose to copy the current pose of selected bones into the pose buffer.

Pose* → *Paste Pose paste the buffered pose to the currently posed armature.

Pose* → *Paste Flipped Pose paste the *X axis mirrored* buffered pose to the currently posed armature.

Here are important points:

- This tool works at the Blender session level, which means you can use it across armatures, scenes, and even files. However, the pose buffer is not saved, so you lose it when you close Blender.
- There is only one pose buffer.
- Only the selected bones are taken into account during copying (i.e. you copy only selected bones' pose).
- During pasting, on the other hand, bone selection has no importance. The copied pose is applied on a per-name basis (i.e. if you had a `forearm` bone selected when you copied the pose, the `forearm` bone of the current posed armature will get its pose when you paste it – and if there is no such named bone, nothing will happen...).
- What is copied and pasted is in fact the position/rotation/scale of each bone, in its own space. This means that the resulting pasted pose might be very different from the originally copied one, depending on: - The rest position of the bones, and - The current pose of their parents.

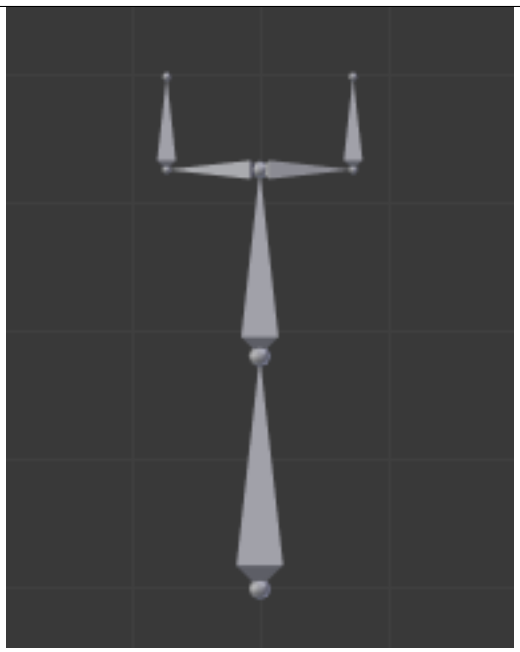


Fig. 2.1221: The rest position of our original armature.

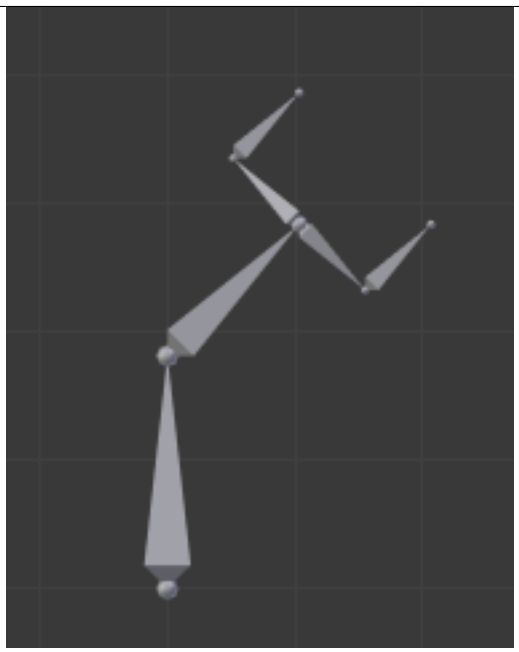


Fig. 2.1222: The rest position of our destination armature.

Table 2.73: ...and mirror-pasted on the destination armature.

Show/Hide

Reference

Mode: All Modes
Panel: *Bone* → *Display*
Menu: ... → *Show/Hide*

You do not have to use bone layers to show/hide some bones. As with objects, vertices or control points, you can use `H`:

- `H` will hide the selected bone(s).
- `Shift-H` will hide all bones *but the selected one(s)*.
- `Alt-H` will show all hidden bones.

You can also use the *Hide* checkbox of the *Bone tab* → *Display panel*.

Note that hidden bones are specific to a mode, i.e. you can hide some bones in *Edit Mode*, they will still be visible in *Pose Mode*, and vice-versa. Hidden bone in *Pose Mode* are also invisible in *Object Mode*. And in *Edit Mode*, the bone to hide

must be fully selected, not just his root or tip.

Bone Constraints

Introduction

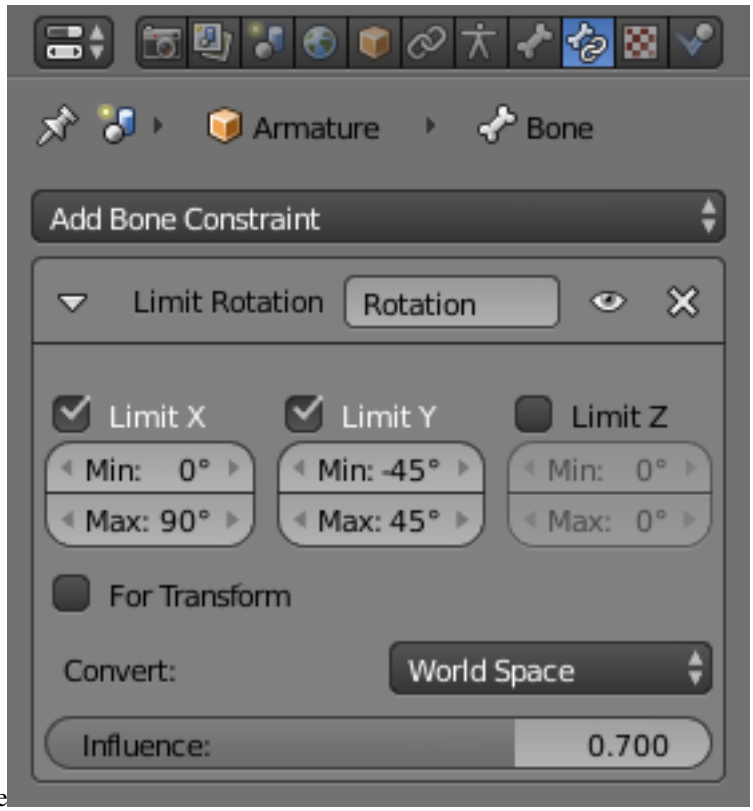
As bones behave like objects in *Pose Mode*, they can also be constrained. This is why the *Constraints* tab is shown in both *Object Mode* and *Edit Mode*. This panel contains the constraints *of the active bone* (its name is displayed at the top of the panel, in the *To Bone:...* static text field).

Constraining bones can be used to control their degree of freedom in their pose transformations, using e.g. the *Limit* constraints. You can also use constraints to make a bone track another object/bone (inside the same object, or in another armature), etc. And the *inverse kinematics feature* is also mainly available through the *IK Solver* constraint, which is specific to bones.

For example, a human elbow cannot rotate backward (unless the character has broken his hand), nor to the sides, and its forward and roll rotations are limited in a given range (for example, depending on the rest position of your elbow, it may be from (0 to 160) or from (-45 to 135).

So you should apply a *Limit Rotation* constraint to the forearm bone (as the elbow movement is the result of rotating the forearm bone around its root).

Using bones in constraints, either as owners or as targets, is discussed in detail in the *constraints pages*.



None

Fig. 2.1229: The Constraints panel in Pose Mode, with one Limit Rotation constraint applied to the active bone.

Inverse Kinematics

Introduction

IK simplifies the animation process, and makes it possible to make more advanced animations with lesser effort.

IK allows you to position the last bone in a bone chain and the other bones are positioned automatically. This is like how moving someone's finger would cause his arm to follow it. By normal posing techniques, you would have to start from the root bone, and set bones sequentially till you reach the tip bone: When each parent bone is moved, its child bone would inherit its location and rotation. Thus making tiny precise changes in poses becomes harder farther down the chain, as you may have to adjust all the parent bones first.

This effort is effectively avoided by use of IK.

Automatic IK

Automatic IK is a tool for quick posing, it can be enabled in the tool shelf in the 3D View, when in pose mode. When the Auto IK option is enabled, translating a bone will activate inverse kinematics and rotate the parent bone, and the parent's

parent, and so on, to follow the selected bone. The IK chain can only extend from a child to a parent bone if the child is *connected* to it.

The length of the chain is increased (if there is a connected parent available to add to it) with `Ctrl-PageUp` or `Ctrl-WheelDown`, and decreased with `Ctrl-PageDown` or `Ctrl-WheelUp`. However, the initial chain length is 0, which effectively means follow the connections to parent bones as far as possible, with no length limit. So pressing `Ctrl-PageUp` the first time sets the chain length to 1 (move only the selected bone), and pressing `Ctrl-PageDown` at this point sets it back to 0 (unlimited) again. Thus, you have to press `Ctrl-PageUp` *more than once* from the initial state to set a finite chain length greater than 1.

This is a more limited feature than using an IK constraint, which can be configured, but it can be useful for quick posing.

IK Constraints

IK is mostly done with bone constraints. They work by the same method but offer more choices and settings. Please refer to these pages for detail about the settings for the constraints:

- *IK Solver*
- *Spline IK*

Armature IK Panel

This panel is used to select the IK Solver type for the armature. *Standard* or *iTaSC*.

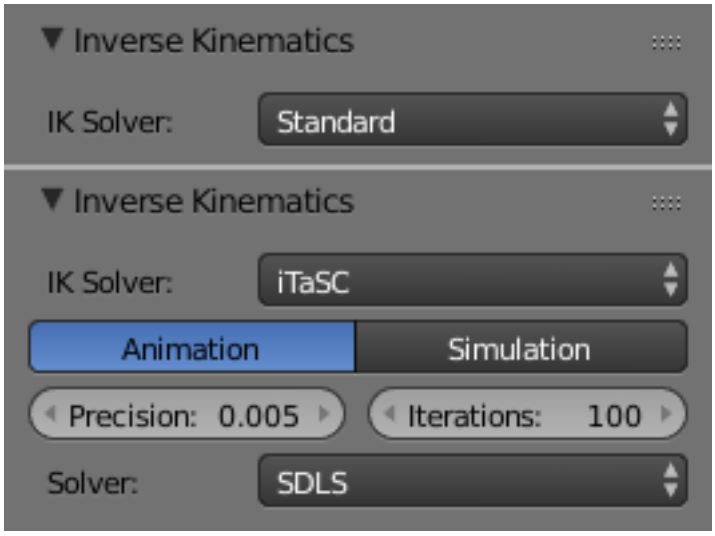


Fig. 2.1230: *Properties* → *Armature* → *Inverse Kinematics Panel*

Most the time people will use the *Standard* IK solver. There is some documentation for the *iTaSC* "instantaneous Task Specification using Constraints" IK solver here.

See also:

Robot IK Solver.

Bone IK Panel

This panel is used to control how the *Pose Bones* work in the IK chain.

Lock Disallow movement around the axis.

Stiffness Stiffness around the axis. Influence disabled if using *Lock*.

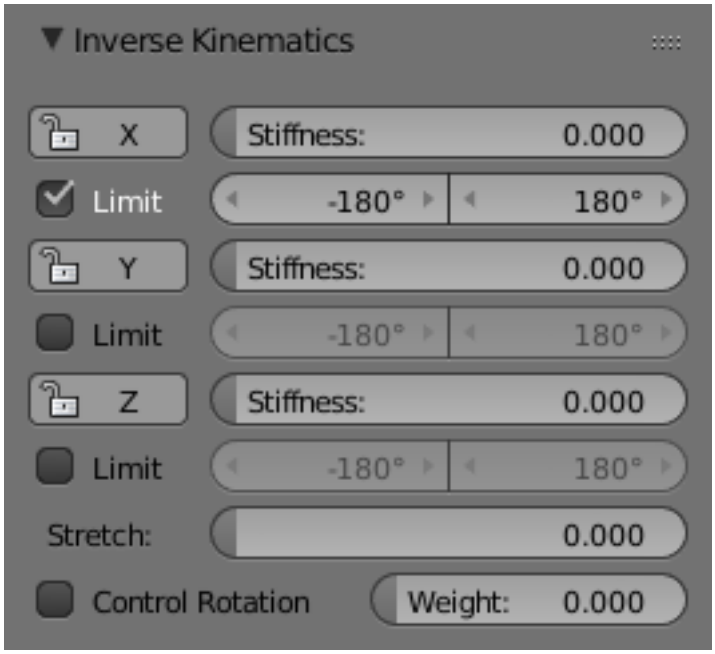


Fig. 2.1231: *Properties → Bone → Inverse Kinematics Panel*

Limit Limit movement around the axis.

Stretch Stretch influence to IK target.

Arm Rig Example

This arm uses two bones to overcome the twist problem for the forearm. IK locking is used to stop the forearm from bending, but the forearm can still be twisted manually by pressing `R-Y-Y` in *Pose Mode*, or by using other constraints.

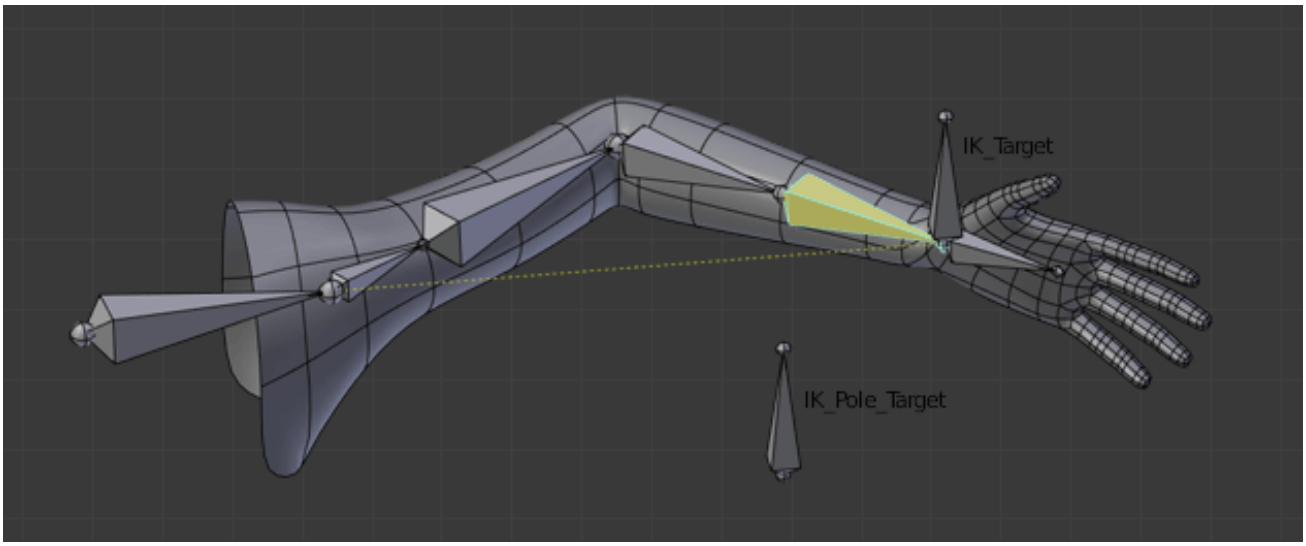


Fig. 2.1232: IK Arm Example..

Note that, if a *Pole Target* is used, IK locking will not work on the root boot.

Spline IK

Spline IK is a constraint which aligns a chain of bones along a curve. By leveraging the ease and flexibility of achieving aesthetically pleasing shapes offered by curves and the predictability and well integrated control offered by bones, Spline IK is an invaluable tool in the riggers' toolbox. It is particularly well suited for rigging flexible body parts such as tails, tentacles, and spines, as well as inorganic items such as ropes.

Full description of the settings for the spline IK are detailed on the *Spline IK* page.

Basic Setup

The Spline IK Constraint is not strictly an 'Inverse Kinematics' method (i.e. IK Constraint), but rather a 'Forward Kinematics' method (i.e. normal bone posing). However, it still shares some characteristics of the IK Constraint, such as operating on multiple bones not being usable for Objects, and being evaluated after all other constraints have been evaluated. It should be noted that if a Standard IK chain and a Spline IK chain both affect a bone at the same time the Standard IK chain takes priority. Such setups are best avoided though, since the results may be difficult to control.

To setup Spline IK, it is necessary to have a chain of connected bones and a curve to constrain these bones to:

- With the last bone in the chain selected, add a *Spline IK* Constraint from the Bone Constraints tab in the Properties Editor.
- Set the 'Chain Length' setting to the number of bones in the chain (starting from and including the selected bone) that should be influenced by the curve.
- Finally, set the 'Target' field to the curve that should control the curve.

Congratulations, the bone chain is now controlled by the curve.

Settings and Controls

Roll Control

To control the 'twist' or 'roll' of the Spline IK chain, the standard methods of rotating the bones in the chain along their y-axes still apply. For example, simply rotate the bones in the chain around their y-axes to adjust the roll of the chain from that point onwards. Applying copy rotation constraints on the bones should also work.

Offset Controls

The entire bone chain can be made to follow the shape of the curve while still being able to be placed at an arbitrary point in 3D-space when the 'Chain Offset' option is enabled. By default, this option is not enabled, and the bones will be made to follow the curve in its untransformed position.

Thickness Controls

The thickness of the bones in the chain is controlled using the constraint's 'XZ Scale Mode' setting. This setting determines the method used for determining the scaling on the X and Z axes of each bone in the chain.

The available modes are:

None this option keeps the X and Z scaling factors as 1.0 .

Volume Preserve the X and Z scaling factors are taken as the inverse of the Y scaling factor (length of the bone), maintaining the 'volume' of the bone

Bone Original this options just uses the X and Z scaling factors the bone would have after being evaluated in the standard way.

In addition to these modes, there is an option, *Use Curve Radius*. When this option is enabled, the average radius of the radii of the points on the curve where the endpoints of each bone are placed, are used to derive X and Z scaling factors. This allows the scaling effects, determined using the modes above, to be tweaked as necessary for artistic control.

Tips for Nice Setups

- For optimal deformations, it is recommended that the bones are roughly the same length, and that they are not too long, to facilitate a better fit to the curve. Also, bones should ideally be created in a way that follows the shape of the curve in its 'rest pose' shape, to minimize the problems in areas where the curve has sharp bends which may be especially noticeable when stretching is disabled.
- For control of the curve, it is recommended that hooks (in particular, Bone Hooks) are used to control the control-verts of the curve, with one hook per control-vert. In general, only a few control-verts should be needed for the curve (i.e. 1 for every 3-5 bones offers decent control).
- The type of curve used does not really matter, as long as a path can be extracted from it that could also be used by the Follow Path Constraint. This really depends on the level of control required from the hooks.
- When setting up the rigs, it is currently necessary to have the control bones (for controlling the curve) in a separate armature to those used for deforming the meshes (i.e. the deform rig containing the Spline IK chains). This is to avoid creating pseudo "Dependency Cycles", since Blender's Dependency Graph can only resolve the dependencies the control bones, curves, and Spline IK'ed bones on an object by object basis.

2.6.4 Lattice

Lattice – or commonly called deformation cage outside of Blender. A lattice consists of a three-dimensional non-renderable grid of vertices. Its main use is to apply a deformation to the object it controls with a *Lattice Modifier*. If the object is parented with *Lattice Deform* a lattice modifier is automatically applied.

Editing

Flip (Distortion Free) Mirrors the vertexes displacement from their base position.

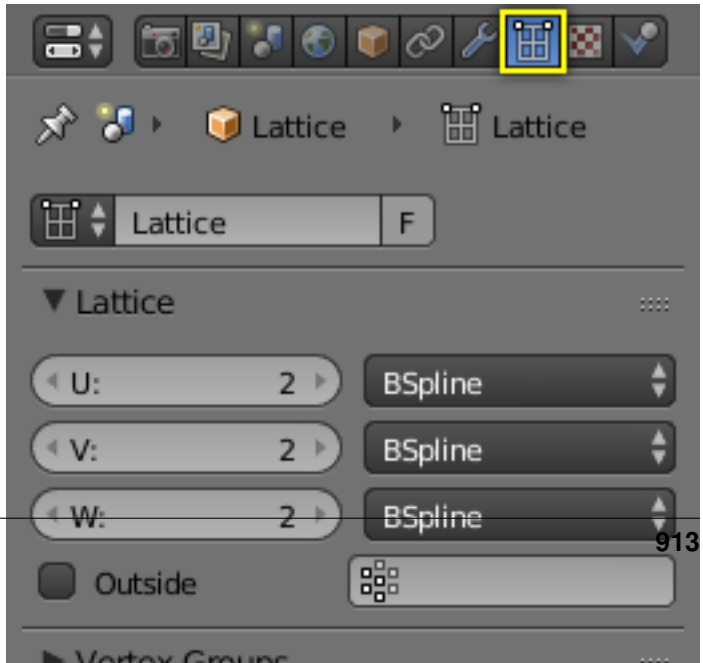
U, V, W

Make Regular Resets the whole lattice to a regular grid, where the cells are scaled to one cubic Blender Unit.

Properties

Lattice A *Data-Block Menu*.

Lattice



Points Rate of subdivision in the axes:

U, V, W

Interpolation Type Selector for each axis. See *Different types of interpolation.*.

Linear, Cardinal, Catmull-Rom, B-Spline

Outside Takes only the vertices on the surface of the lattice into account.

Vertex Group The strength of the influence assigned as a weight to the individual vertices in the selected vertex group.

Usage

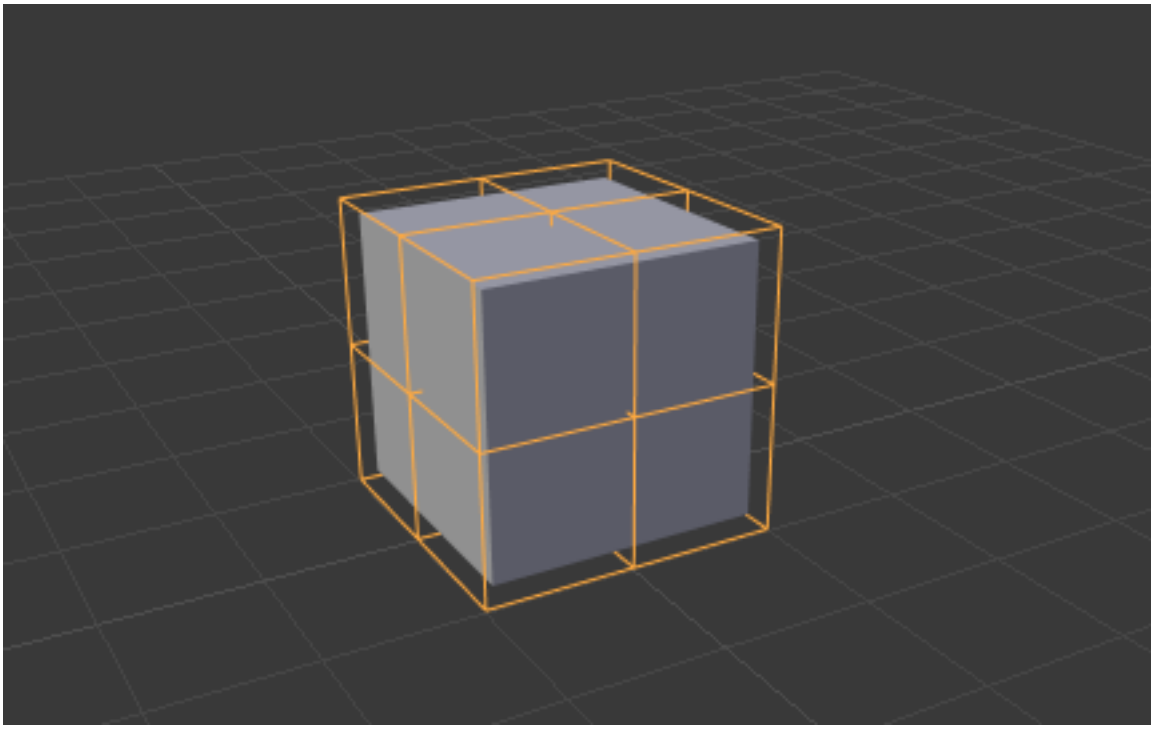


Fig. 2.1234: Lattice around the cube object in Object Mode.

The lattice should be scaled and moved to fit around your object in Object Mode. Any scaling applied to the object in Edit Mode will result in the object deforming. This includes applying scale with `Ctrl-A` as this will achieve the same result as scaling the lattice in Edit Mode, and therefore the object.

2.7 Animation

2.7.1 Introduction

Animation is making an object move or change shape over time. Objects can be animated in many ways:

Moving as a whole object Changing their position, orientation or size in time;

Deforming them Animating their vertices or control points;

Inherited animation Causing the object to move based on the movement of another object (e.g. its parent, hook, armature, etc...).

In this chapter, we will cover the first two, but the basics given here are actually vital for understanding the following chapters as well.

Animation is typically achieved with the use of *keyframes*.

Chapters

Animation Fundamentals

Actions Actions are used to record the animation of objects and properties.

Drivers Drivers are scripts used to control and animate properties.

Keying Sets Keying Sets are used to record a set of properties at the same time.

Markers Markers are used to mark key points/events within an animation.

Motion Paths Motion Paths are used to visualize an animation.

Shape Keys Shape Keys are used to deform objects into new shapes.

Animation Editors

Timeline The Timeline Editor is a quick editor to set and control the time frame. This also has some tools for animation.

Graph Editor The Graph Editor is mostly used to edit the F-Curves and Keyframes for Channels and Drivers.

Dope Sheet The Dopes Sheet contains a collection of animation editors.

NLA Editor The NLA Editor is used to edit and blend Actions together.

Related Sections

Rigging Rigging.

Constraints Constraints are a way of connecting transformation properties (position, rotation and scale) between objects.

Physical Simulation This category covers various advanced Blender effects, often used to simulate real physical phenomena. There is the Particle System for things like hair, grass, smoke, flocks. Soft Bodies are useful for everything that tends to bend, deform, in reaction to forces like gravity or wind. Cloth simulation, to simulate clothes or materials. Rigid Bodies can simulate dynamic objects that are fairly rigid. Fluids, which include liquids and gases, can be simulated, including Smoke. Force Fields can modify the behavior of simulations.

Motion Tracking Motion tracking is a technique available in Blender that supports basic operations for 2D motion tracking, 3D motion tracking, and camera solution.

State Colors

Fig. 2.1235: State colors of properties.

Properties have different colors and menu items for different states.

Gray	Default
Yellow	Keyframes
Green	Animated
Purple	Driver

2.7.2 Keyframes

Introduction

A *Keyframe* is simply a marker of time which stores the value of a property.

For example, a Keyframe might define that the horizontal position of a cube is at 3m on frame 1.

The purpose of a Keyframe is to allow for interpolated animation, meaning, for example, that the user could then add another key on frame 10, specifying the cube's horizontal position at 20m, and Blender will automatically determine the correct position of the cube for all the frames between frame 1 and 10 depending on the chosen interpolation method (e.g. Linear, Bézier, Quadratic, etc...).

Keyframe Types

For visually distinguish regular keyframes from different animation events or states (extremes, breakdowns, or other in betweens) there is the possibility the applying different colors on them for visualization.

Keyframe (yellow diamond) Normal keyframe.

Breakdown (cyan small diamond) Breakdown state. e.g. for transitions between key poses.

Moving Hold (slight orange diamond) A keyframe that adds a small amount of motion around a holding pose.

Extreme (red big diamond) An 'extreme' state, or some other purpose as needed.

Jitter (green tiny diamond) A filler or baked keyframe for keying on ones, or some other purpose as needed.

Editing

Adding Keyframes

There are several methods of adding new keys. Namely:

- In the 3D View, pressing `I` will bring up a menu to choose what to add a keyframe to.
- Hovering over a property and pressing `I` or with the context menu by `RMB` a property and choose *Insert Keyframe* from the menu.

Auto Keyframe

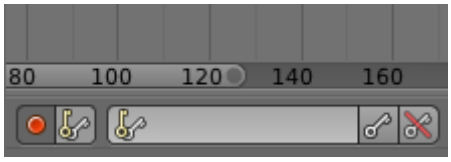


Fig. 2.1236: Timeline Auto Keyframe.

Auto Keyframe is the red record button in the *Timeline* header. Auto Keyframe adds keyframes automatically to the set frame if the value for transform type properties changes.

See *Timeline Keyframe Control* for more info.

Removing Keyframes

There are several methods of removing keyframes:

- In the 3D View press `Alt-I` to remove keys on the current frame for selected objects.
- When the mouse is over a value press `Alt-I`.
- `RMB` a value and choose *Delete Keyframe* from the menu.

Editing Keyframes

Keyframes can be edited in two editors. To do so go to either the *Graph Editor* or the *Dopesheet*.

Examples

Keyframe Animation

This example shows you how to animate a cubes location, rotation, and scale.

1. First, in the *Timeline*, or other animation editors, set the frame to 1.
2. With the *Cube* selected in *Object Mode*, press `I` in the 3D View.
3. From the *Insert Keyframe Menu* select *LocRotScale*. This will record the location, rotation, and scale, for the *Cube* on frame 1.
4. Set the frame to 100.
5. Use Grab/Move `G`, Rotate `R`, Scale `S`, to transform the cube.
6. Press `I` in the 3D View. From the *Insert Keyframe Menu* select *LocRotScale*.

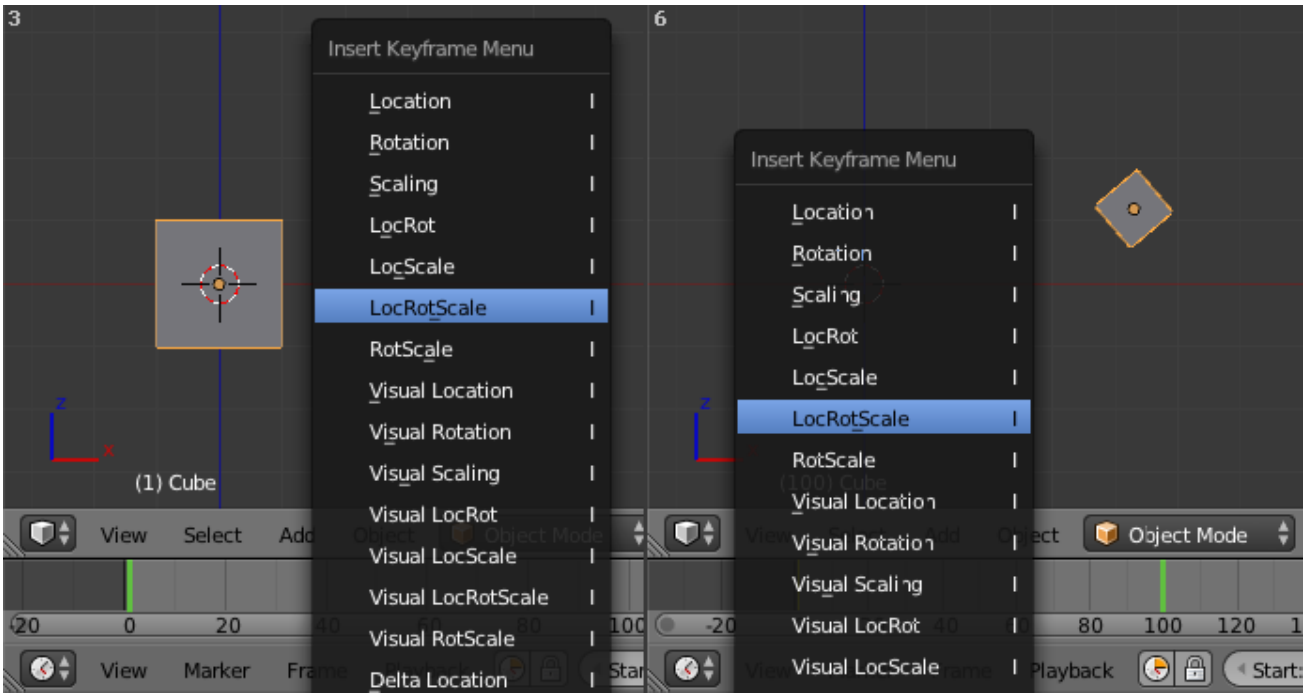


Fig. 2.1237: Insert Keyframes.

To test the animation, press `Alt-A` to play.

Visualization

There are some important visualization features in the 3D Views that can help animation.

When the current frame is a keyframe for the current active object, the name of this object (shown in the bottom left corner of the 3D Views) turns yellow.

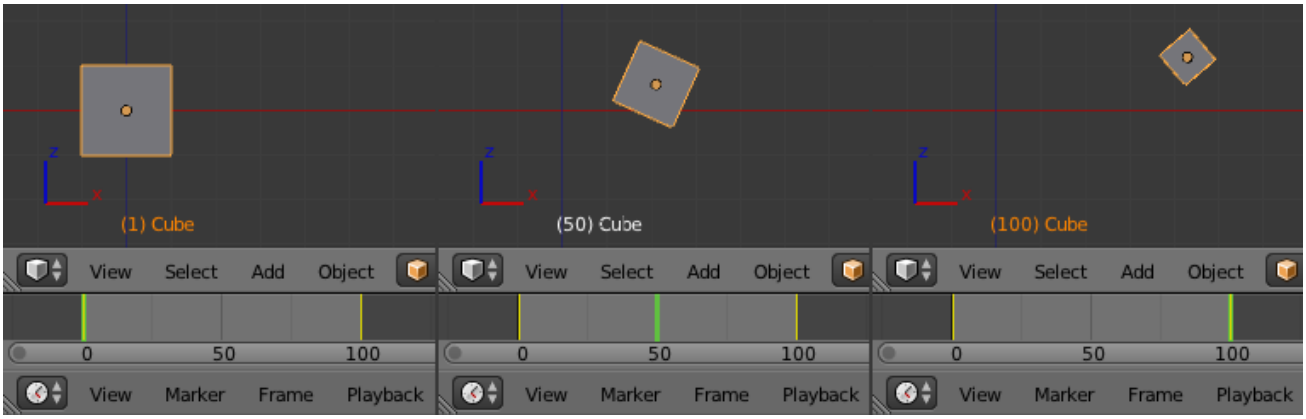


Fig. 2.1238: The animation on frames 1, 50, 100.

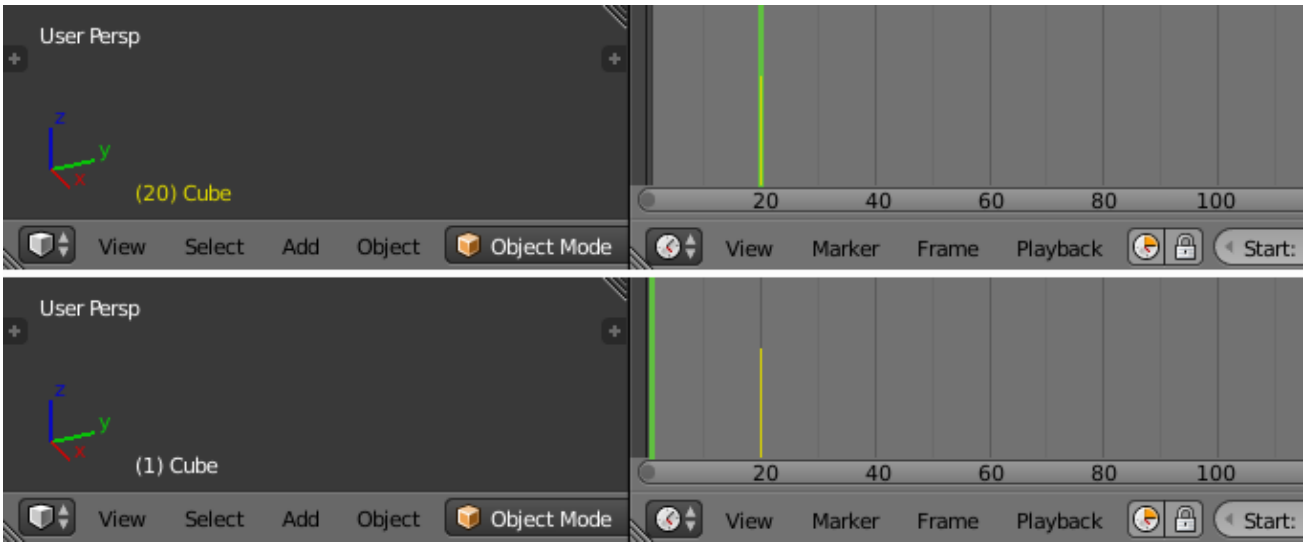


Fig. 2.1239: Buttom: Current frame at 0. Top: Current frame is a keyframe for Cube.

Motion Paths

Reference

Mode: Object Mode
Panel: *Object*

This feature allows you to visualize the animation of objects by displaying their position over a series of frames.

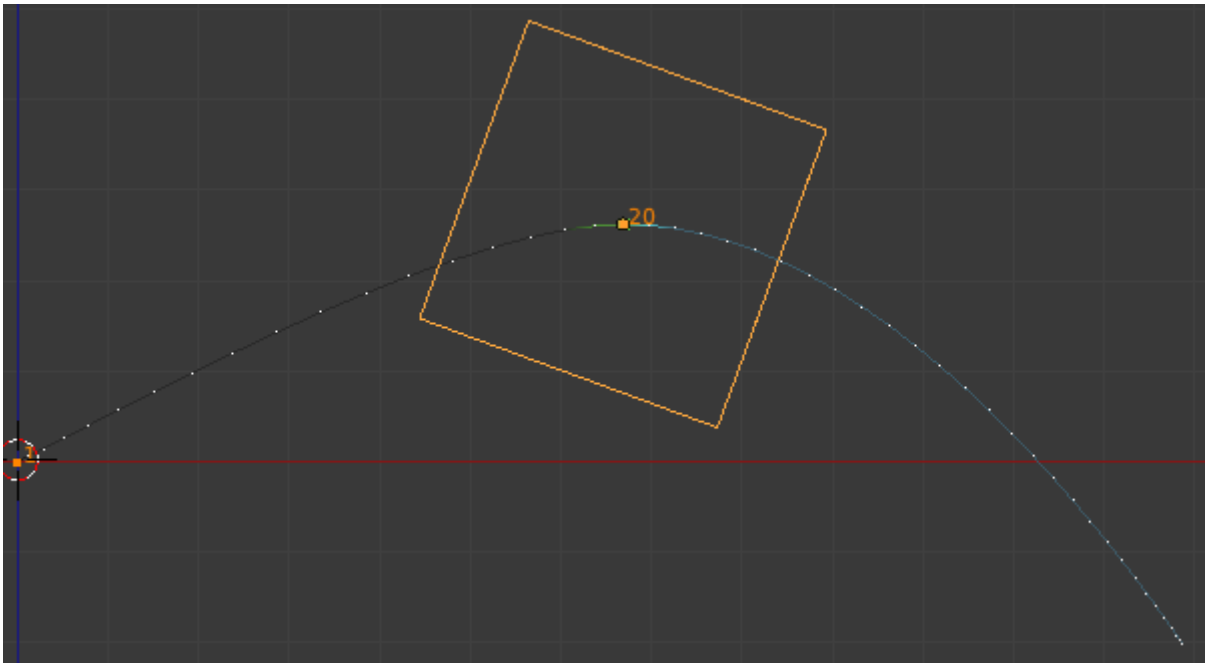


Fig. 2.1240: An animated cube with its motion path displayed.

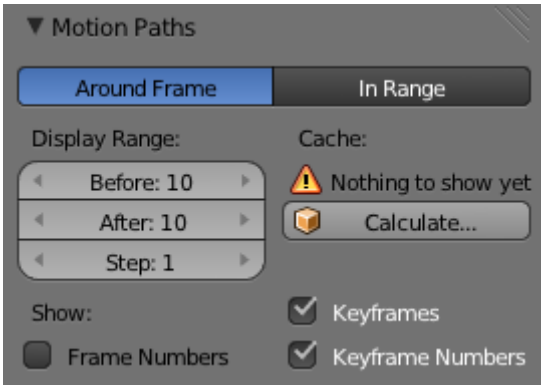


Fig. 2.1241: Motion paths panel.

Before we look at its options (all regrouped in the same *Visualizations* panel, in the Properties editor), let us first see how to display/hide these paths. You have to

do it manually and you have to first select the objects you want to show/hide the motion paths. Then:

1. To show the paths (or update them, if needed), click on the *Calculate Path* button.
2. To hide the paths, click on the *Clear Paths* button.

> **Warning:** Remember that only selected object and their paths are affected by these actions!

The paths are drawn in black with white dots indicating frames, and a blue glow around the current frame.

Options

Around Frame Around Frame, Display Paths of poses within a fixed number of frames around the current frame. When you enable this button, you rather get paths for a given number of frames before and after the current one (again, as with ghosts).

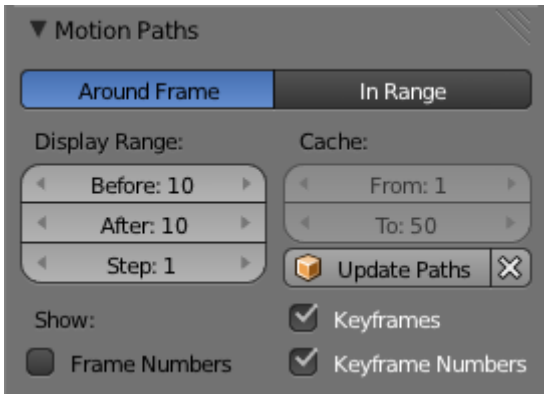


Fig. 2.1242: The Motion Paths Panel set to “Around Frame”

In Range In Range, Display Paths of poses within specified range.

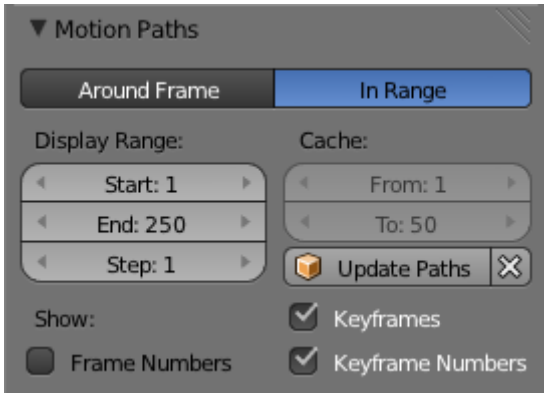


Fig. 2.1243: The Motion Paths Panel set to “In Range”

Display Range

Before/After Number of frames to show before and after the current frame (only for *Around Current Frame* Onion-skinning method)

Start/End Starting and Ending frame of range of paths to display/calculate (not for *Around Current Frame* Onion-skinning method)

Step This is the same thing as the *GStep* for ghosts – it allows you the only materialize on the path one frame each *n* ones. Mostly useful when you enable the frame number display (see below), to avoid cluttering the 3D Views.

Frame Numbers When enabled, a small number appears next to each frame dot on the path, which is, of course, the number of the corresponding frame...

Keyframes When enabled, big yellow square dots are drawn on motion paths, materializing the keyframes of their bones (i.e. only the paths of keyed bones at a given frame get a yellow dot at this frame).

Keyframe Numbers When enabled, you will see the numbers of the displayed keyframes – so this option is obviously only valid when *Show Keys* is enabled.

Cache

From/To These are the start/end frames of the range in which motion paths are drawn. You cannot modify this range without deleting the motion path first.

Calculate Paths/ Update Paths If no paths have been calculated, Calculate Paths will create a new motion path in cache. In the pop-up menu, select the frame range to calculate. If a path has already been calculated, Update Paths will update the path shape to the current animation. To change the frame range of the calculated path, you need to delete the path and calculate it again.

Keying Sets

Keying Sets are a collection of properties.

They are used to record multiple properties at the same time. Now when you press `I` in the 3D View, Blender will add keyframes for all the properties in the active keying set.

There are some built in Keying Sets, and also, custom Keying Sets called *Absolute Keying Sets*.

To select and use a Keying Set, set the *Active Keying Set* in the *Timeline Header*, or the Keying Set panel, or press `Ctrl-Alt-Shift-I` in the 3D View.

Keying Set Panel

This panel is used to add, select, manage *Absolute Keying Sets*.

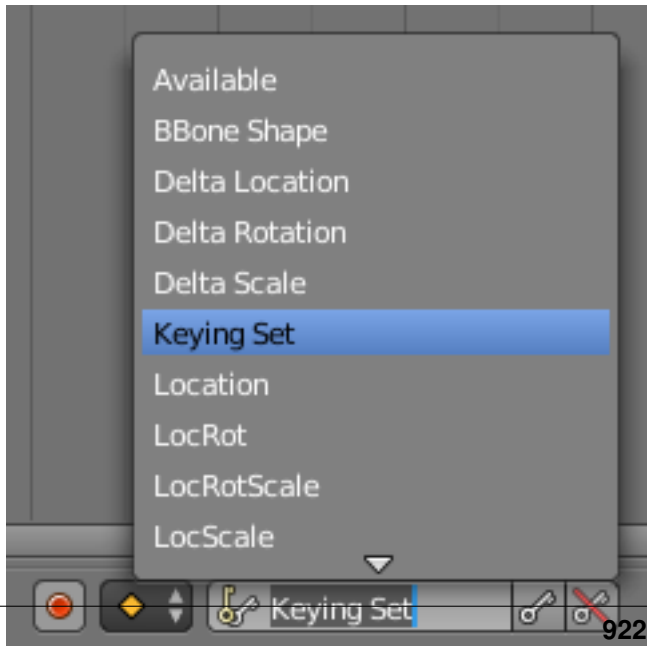


Fig. 2.1244: Timeline Keying Sets.

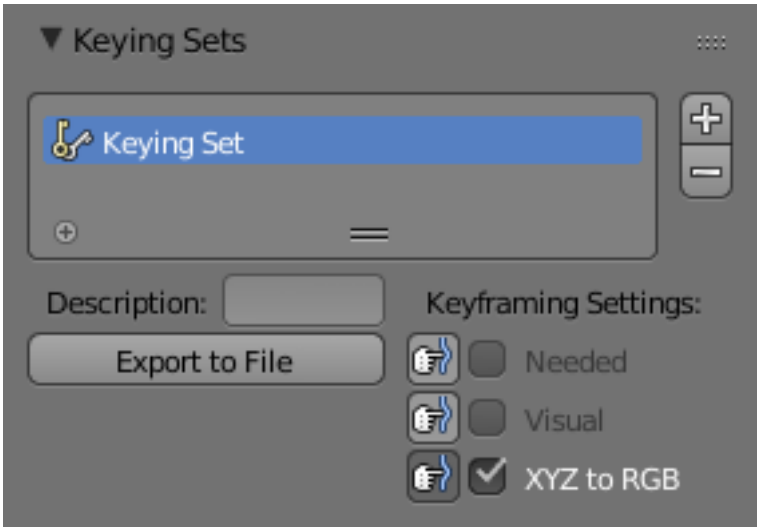


Fig. 2.1245: *Properties → Scene → Keying Set Panel*

Active Keying Set The *List View* of Keying Sets in the active Scene.

Add + Adds a empty Keying Set.

Properties

Description A short description of the keying set.

Export to File Export Keying Set to a Python script `File.py`. To re-add the keying set from the `File.py`, open then run the `File.py` from the Text Editor.

Keyframing Settings These options control all properties in the Keying Set. Note, the same settings in *User Preferences* override these settings if enabled.

Only Needed Only insert keyframes where they are needed in the relevant F-Curves.

Visual Keying Insert keyframes based on the visual transformation.

XYZ to RGB For new F-Curves, set the colors to RGB for the property set, Location XYZ for example.

Active Keying Set Panel

This panel is used to add properties to the active Keying Set.

Active Keying Set Paths A collection of paths in a *List View* each with a *Data Path* to a property to add to the active Keying Set.

Add + Adds a empty path.

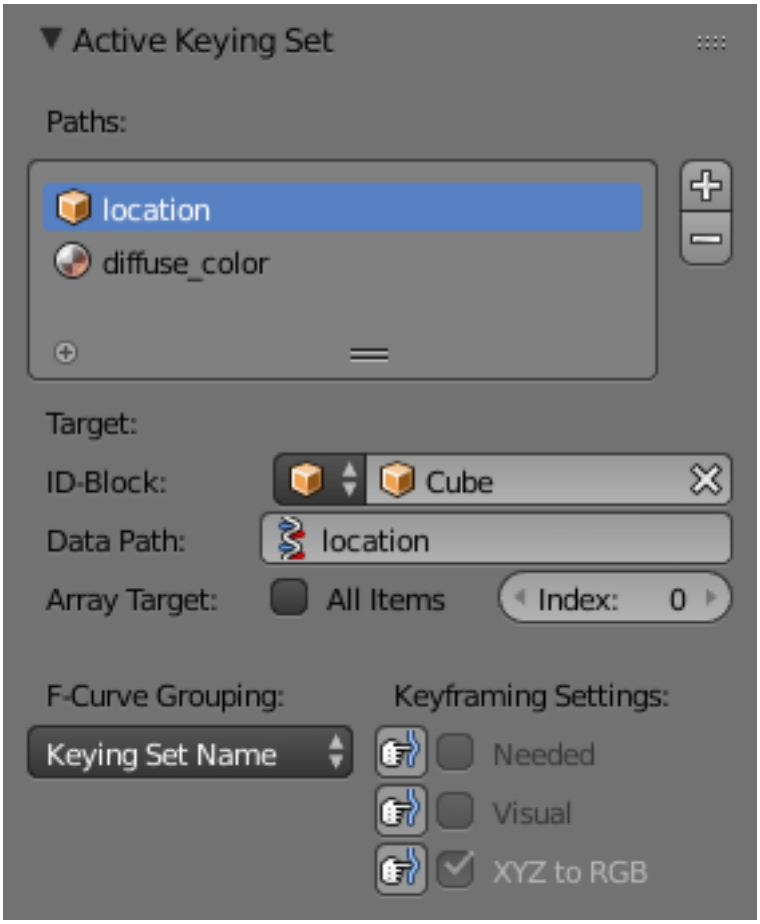


Fig. 2.1246: *Properties → Scene → Active Keying Set Panel*

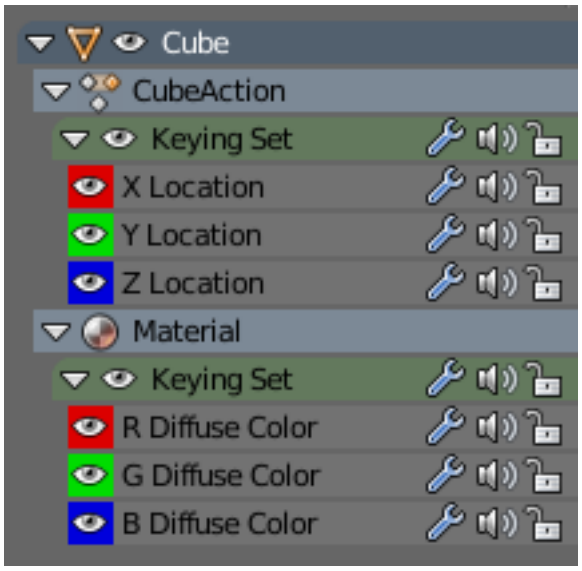


Fig. 2.1247: *Properties → Graph Editor → Channels, Named Group*

Properties

Target

ID-Block Set the ID-Type and the *Object IDs* Data Path for the property.

Data Path Set the rest of the Data Path for the property.

Array Target Use *All Items* from the Data Path or select the array index for a specific property.

F-Curve Grouping This controls what group to add the channels to.

Keying Set Name, None, Named Group

Keyframing Settings These options control individual properties in the Keying Set.

Only Needed Only insert keyframes where they are needed in the relevant F-Curves.

Visual Keying Insert keyframes based on the visual transformation.

XYZ to RGB For new F-Curves, set the colors to RGB for the property set, Location XYZ for example.

Adding Properties

Some ways to add properties to keying sets.

RMB the property in the *User Interface*, then select *Add Single to Keying Set* or *Add All to Keying Set*. This will add the properties to the active keying set, or to a new keying set if none exist.

Hover the mouse over the properties, then press K, to add *Add All to Keying Set*.

2.7.3 Actions

When animating objects and properties in Blender, Actions record and contain the data. As everything else in Blender, Actions are data-blocks.

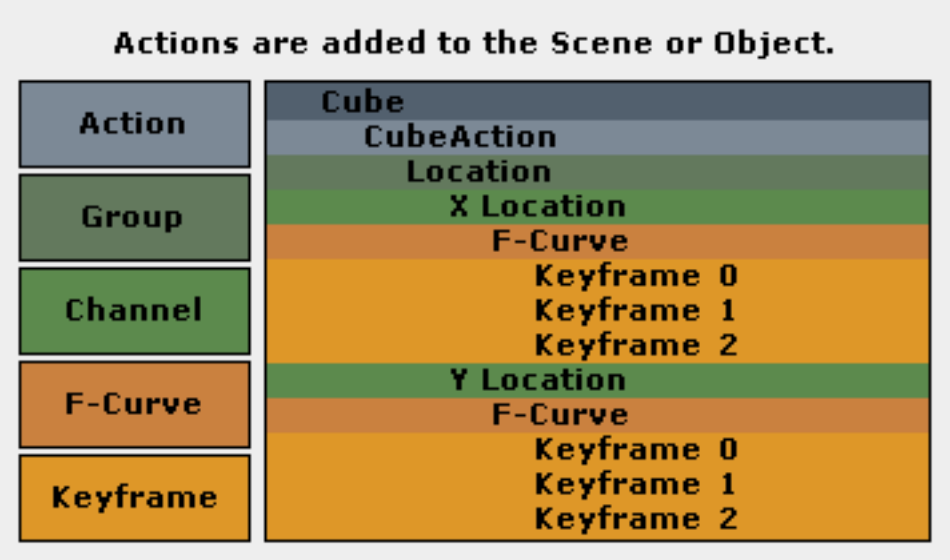


Fig. 2.1248: Actions.

So when you animate an object by changing its location with keyframes, the animation is saved to the Action.

Each property has a channel which it is recorded to, for example, Cube.location.x is recorded to Channel X Location. The *X location* and *Y location* properties can be shared across multiple objects, if both objects have *X location* and *Y location* properties beneath them.

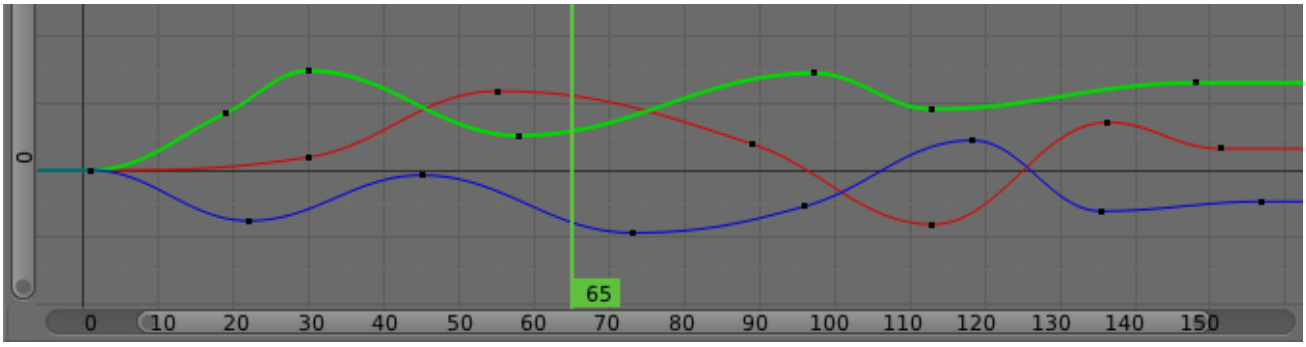


Fig. 2.1249: Graph Editor. Each Channel has an F-Curve represented by the lines between the keyframes.

Actions Record and contain animation data.

Groups Are groups of channels.

Channels Record properties.

F-Curves *F-Curve* are used to interpolate the difference between the keyframes.

Keyframes *Keyframes* are used to set the values of properties bind to a point in time.

Working with Actions

When you first animate an object by adding keyframes, Blender creates an *Action* to record the data.

Actions can be managed with the *Action data-block menu* in the Dope Sheet *Action Editor* header, or the properties region of the *NLA Editor*.

If you are making multiple actions for the same object, press the *F* button for each action, this will give the actions a *Fake User* and will make Blender save the unlinked actions.

Objects can only use one *Action* at a time for editing, the *NLA Editor* is used to blend multiple actions together.

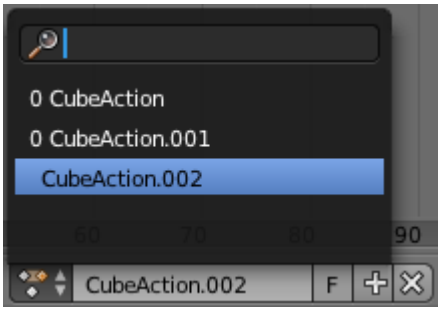


Fig. 2.1250: The Action data-block menu.

Bake Action

Reference

Mode: Object and Pose Modes
Menu: *3D View* → *Object/Pose* → *Animation* → *Bake action*

The *Bake Action* tool will apply interpolated frames into individual key frames.

This can be useful for adding deviation to a cyclic action like a *walk cycle*. This can also useful for keyframe animations created from drivers or constraints.

2.7.4 Drivers

Introduction

Drivers can use properties, numbers, transformations, and scripts, to control the values of properties.

Using a F-Curve, the driver reads the value of the Driver Value and sets the value of the selected property it was added to.

So from this example, if the Driver Value is 2.0 the property will be 0.5.

The Driver Value is determined by Driver Variables or a Scripted Expression.

Most the settings for the drivers *F-Curves* are found in the *Graph Editor*.

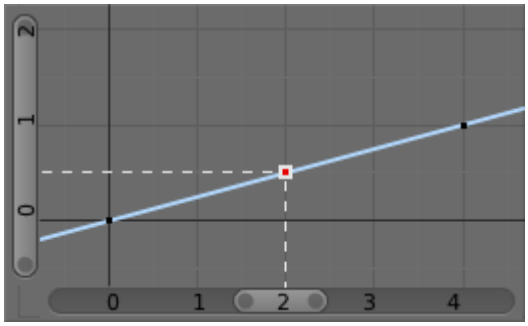


Fig. 2.1251: Graph Editor: Driver example.

Drivers Panel

This panel is located in the *Graph Editor* with the mode set to Drivers.

The drivers panel is for setting up *Driver Variables* or a *Scripted Expression* which will determine the value of the *Driver Value*.

Settings

Update Dependencies This will force an update for the Driver Value dependencies.

Remove Driver Removes the driver from the object.

Type The type of calculation to use on the set of Driver Variables. (If you only have one driver variable there is no real difference between average, sum, minimum and maximum)

Average Value Uses the average value of the referenced Driver Variables.

Sum Values Uses the sum of the referenced Driver Variables.

Scripted Expression Uses a Scripted Expression. See Expression. You must write a Python expression which performs your own calculations on the Driver Variables.

Minimum Value Uses the lowest value from the referenced Driver Variables.

Maximum Value Uses the highest value from the referenced Driver Variables.

Expression Scripted Expression. Here you can add real numbers, math operators, math functions, python properties, driver functions. See Driver Expression below for some examples.

Show Debug Info Shows the Driver Value. The current value of the variables or scripted expression.

Add Variable Adds a new Driver Variable.

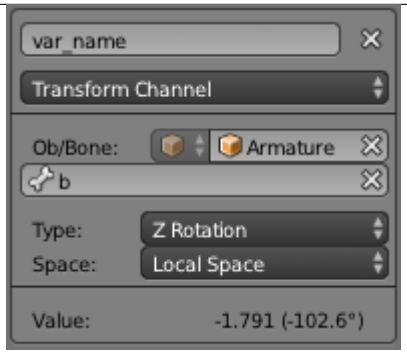


Fig. 2.1254: Transform Channel Setup.

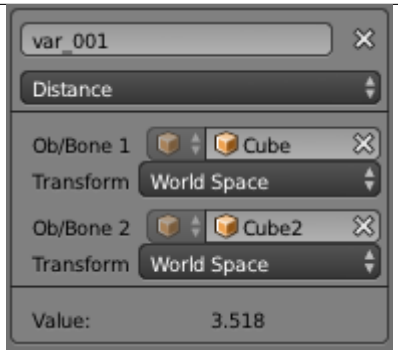


Fig. 2.1255: Distance Setup.

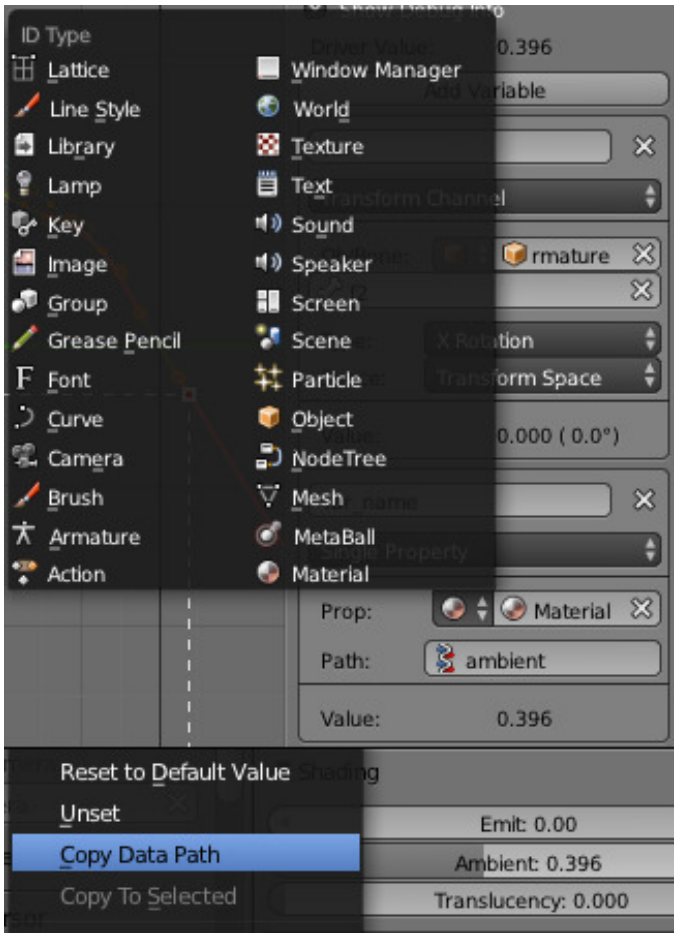


Fig. 2.1253: Setup of a Single Property.

Driver Variables

Name Name to use for scripted expressions/functions. No spaces or dots are allowed and must start with a letter.

Variable Type The type of variable to use.

Single Property Use the value from some RNA property. For example, the Ambient shading color from a material. First select the type of ID-block, then the ID of the ID-block, then copy and paste an RNA property `Ctrl-V`.

ID-Type The ID-Block type. Example: Key, Image, Object, Material.

ID The ID of the ID-Block type. Example: "Material.001".

RNA Path The RNA id name of the property. Example: 'ambient' from material shading.

Transform Channel Use one of the Transform channels from an object or bone.

ID ID of the object. Example: Cube, Armature, Camera.

Bone ID of the Armature bone. Example: "Bone", "Bone.002", "Arm.r". This option is for armatures.

Type Example, X Location, X Rotation, X Scale.

Space World Space, Transform Space, Local Space.

Rotational Difference Use the rotational difference between two objects or bones.

Distance Use the distance between two objects or bones.

Value Shows the value of the variable.

See also:

- *Animation*
- *Graph Editor*
- *F-Curves*
- *Extending Blender with Python*.
- Python and its documentation.
- functions.wolfram.com.

Workflow & Examples

Workflow

There are some different ways to add drivers in Blender. These are some driver examples and workflow. After adding drivers they are usually modified in the *Graph Editor* with the mode set the *Drivers*.

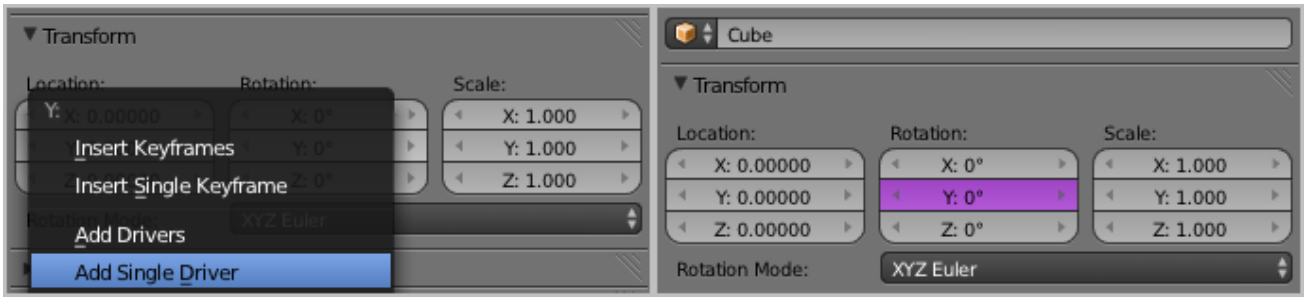
UI

The common way to add a driver to a property is to right click a property, then add a driver via the context menu.

Add Drivers This will add drivers to the set of properties related to the selected one. For example, it will add drivers to X, Y, and Z for Rotation.

Add Single Driver This will add a single driver to the selected property.

Drivers can also be added by pressing `CTRL-D` with the mouse over the property set.



Expression

This is a quick way to add drivers with a scripted expression. First click the property you want to add a driver to, then add a hash `#` and a scripted expression.

Some examples:

- `#frame`
- `#frame / 20.0`
- `#sin(frame)`
- `#cos(frame)`

Copy Paste

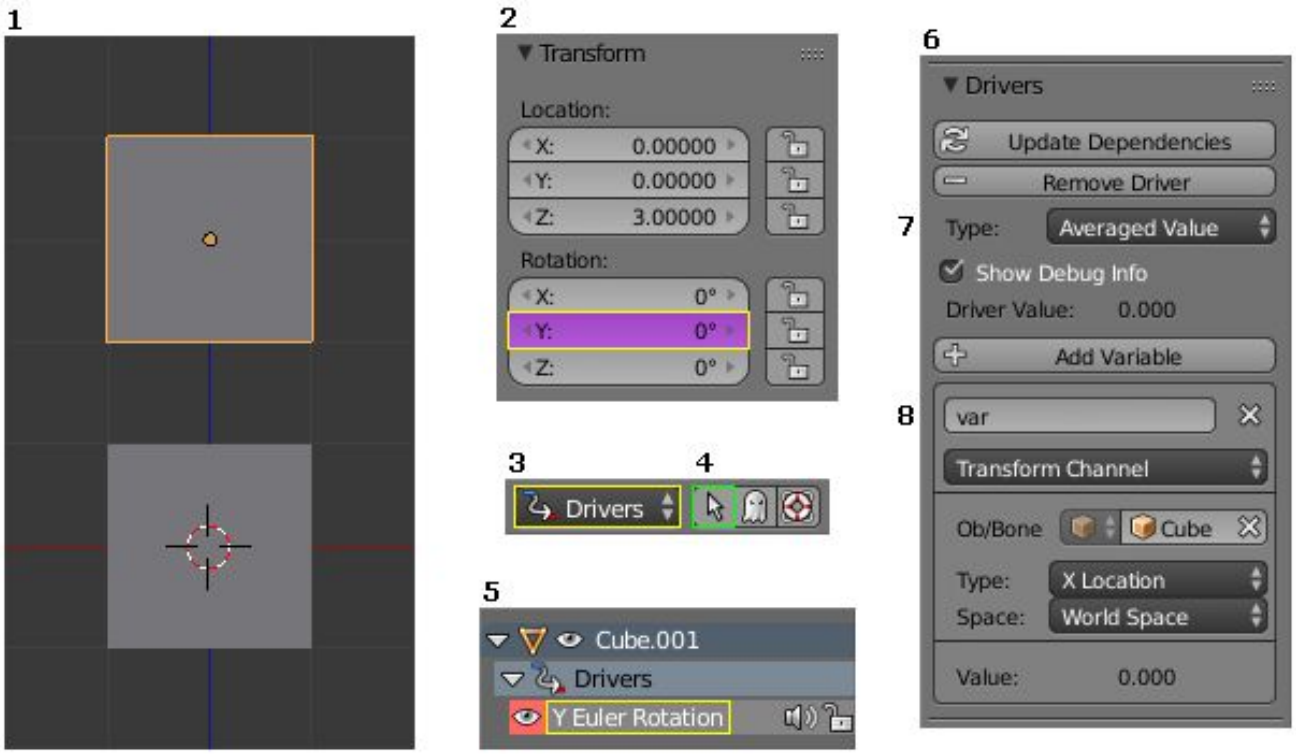
Drivers can be copied and pasted in the UI, via the context menu. When adding drivers with the same settings, this can save time modifying settings.

Transform Driver

This example shows you how setup a transform driver. First make sure you are in the Front Ortho view. `Numpad5`, `Numpad1`.

1. In object mode, select then duplicate the default Cube. `Shift-D`. Move "Cube.001" to a new location.
2. With "Cube.001" selected, add a single driver to the *Rotation Y* property.
3. Open the *Graph Editor*, set the Mode to *Drivers*.
4. *Show Only Selected* is useful disabled for drivers, marked green in the picture.
5. In the channels region, select the *Y Euler Rotation* property.
6. Press `N` to open the properties region, scroll down to Drivers panel.
7. Change the *Type* to *Averaged Value*, this will return the averaged value of the driver variables.
8. Modify the driver variable settings:
   - Type – Transform Channel
   - Ob/Bone – Cube
   - Transform Type – X Location
   - Transform Space – World Space

When finished, "Cube.001" should rotate on the Y axis when moving "Cube" left to right.



Examples

Driver Expression

Here are some examples using the scripted expression Expr to set the Driver Value.

Orbit a Point

Here two drivers have been added to the Cube, X Location and Y Location.

The scripted expressions are being used to set the object location.

X Location Expr

`0 + (sin(frame / 8) * 4)` `(frame/8)` : is the current frame of the animation, divided by 8 to slow the orbit down. `(sin( )*4)` : This returns the sine of (frame/8), then multiplies by 4 for a bigger circle. `0 +` : is used to control the X Location offset of the orbit.

Y Location Expr

`0 + (cos(frame / 8) * 4)` `(frame / 8)` : is the current frame of the animation, divided by 8 to slow the orbit down. `(cos( ) * 4)` : This returns the cosine of (frame/8), then multiplies by 4 for a bigger circle. `0 +` : is used to control the Y Location offset of the orbit.

`frame` is the same as bpy.context.scene.frame_current.

Driver Namespace

There is a list of built-in driver functions and properties. These can be displayed via the Python Console:

```
>>> bpy.app.driver_namespace['
                             __builtins__']
                             __doc__']
```

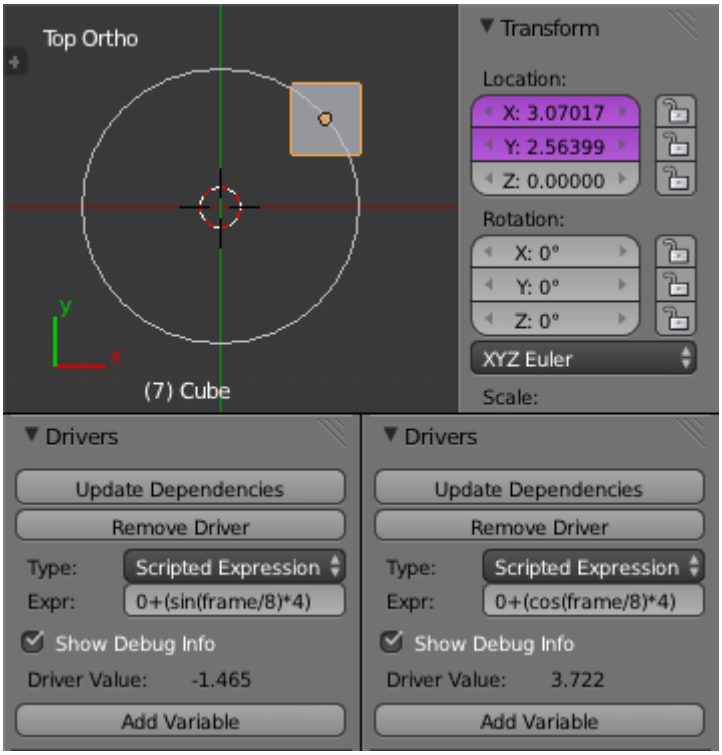


Fig. 2.1256: Object Rotation.

```
__loader__']
__name__']
__package__']
acos']
acosh']
asin']
asinh']
atan']
atan2']
atanh']
bpy']
ceil']
copysign']
cos']
cosh']
..
```

This script will add a function to the driver namespace, which can then be used in the expression `driver_func(frame)`:

```
import bpy

def driver_func(val):
    return val * val    # return val squared

# add function to driver_namespace
bpy.app.driver_namespace['driver_func'] = driver_func
```

Shape Key Driver

This example is a Shape Key Driver. The driver was added to the shape key Value.

This example uses the Armature Bone “b” ‘s Z Rotation to control the Value of a Shape Key. The bone rotation mode is set

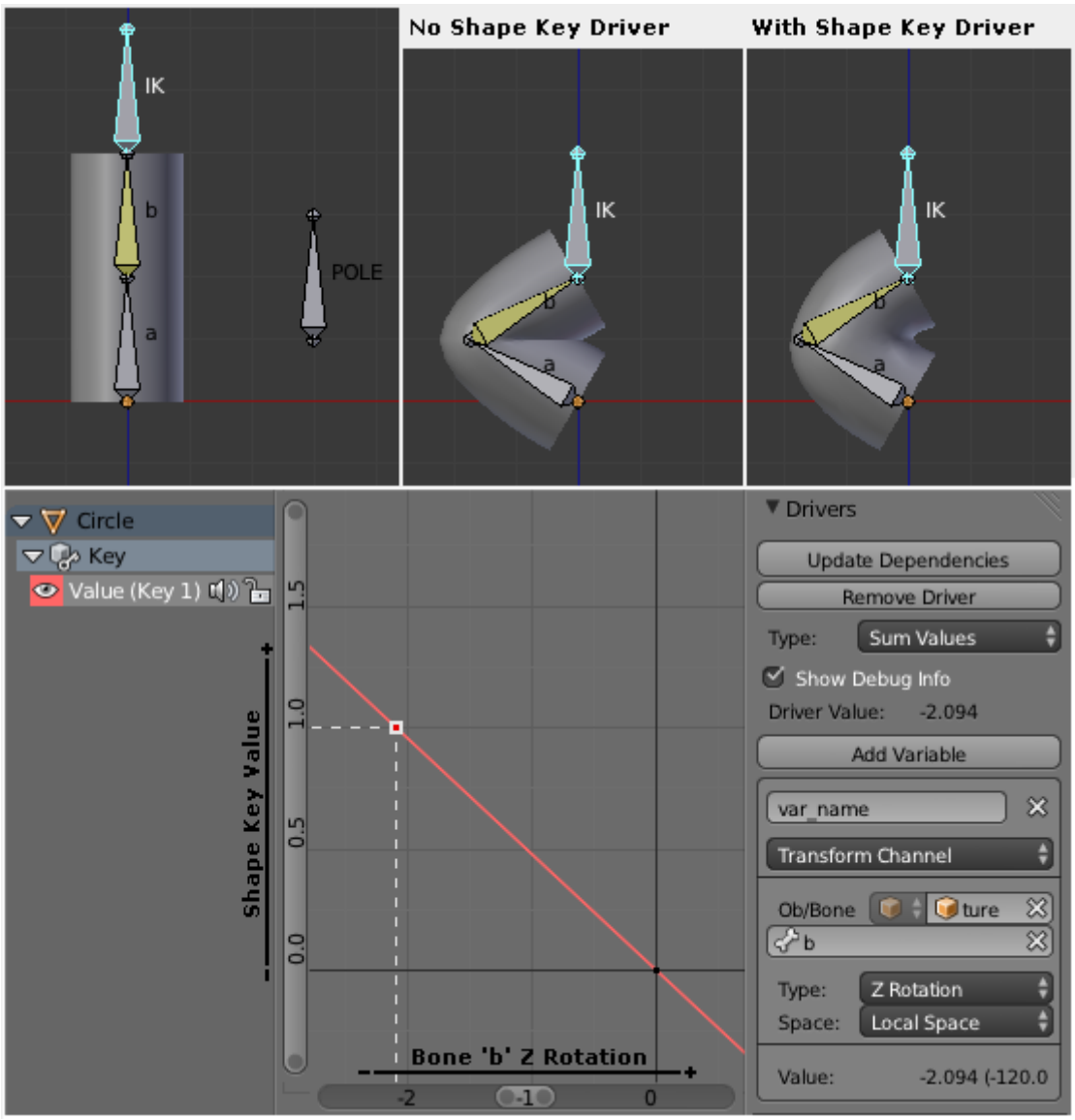


Fig. 2.1257: Shape Key Driver. Click to enlarge.

to XYZ Euler.

The Driver F-Curve is mapped like so:

- Bone Z Rotation 0.0 (0.0): Shape Key value 0.0
- Bone Z Rotation -2.09 (-120.0): Shape Key value 1.0

This kind of driver can also be setup with the Variable Type Rotational Difference.

See *Shape Keys* for more info.

Drivers And Multiple Relative Shape Keys

The following screenshots illustrate combining shape keys, bones, and drivers to make multiple chained relative shape keys sharing a single root. While it lacks the convenience of the single Evaluation Time of an absolute shape key, it allows you to have more complex relationships between your shape keys.

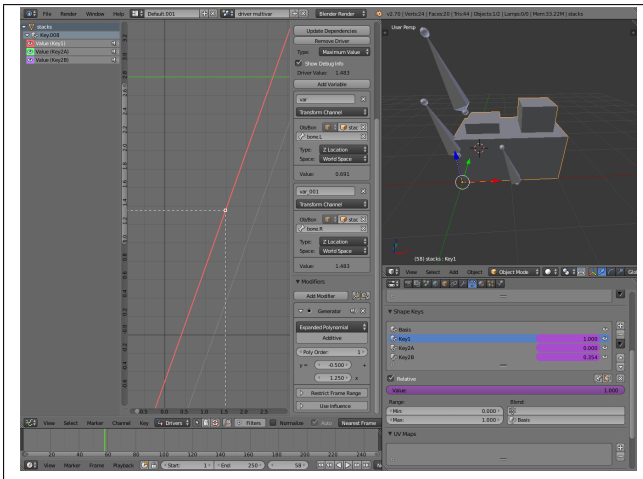


Fig. 2.1258: Key1 must handle conflicting values from the two bones.

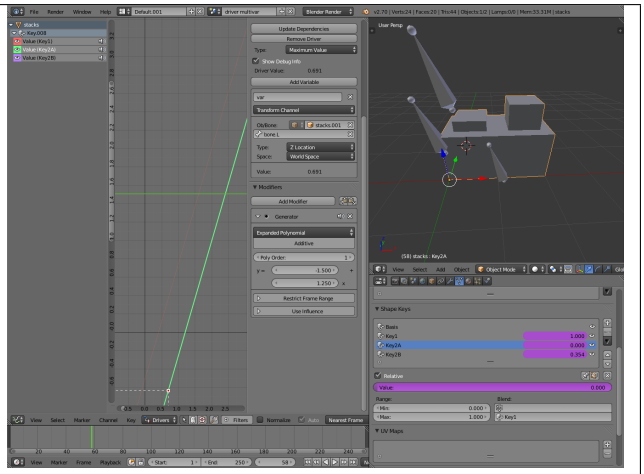


Fig. 2.1259: Key2A has different generator coefficients so it is activated in a different range of the bone's position.

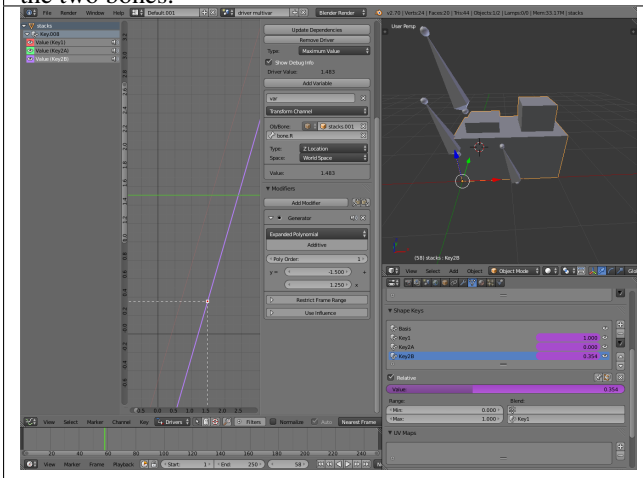


Fig. 2.1260: Key2B is the same as Key2A, but is controlled by the second bone.

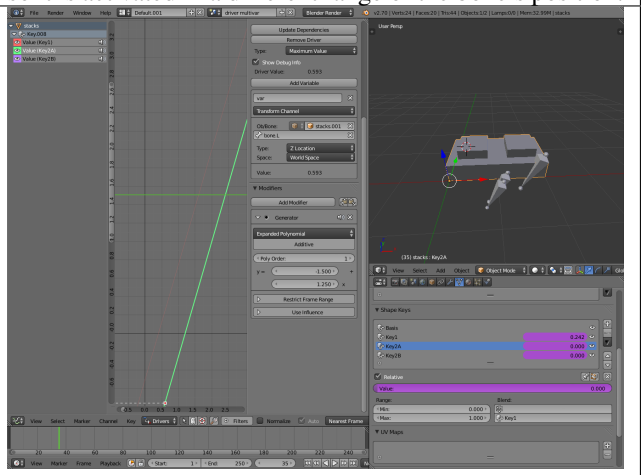
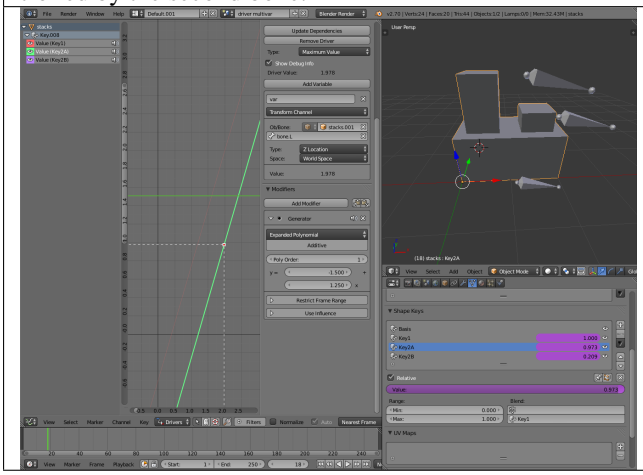


Fig. 2.1261: When both bones are low, Key2B and Key2A are deactivated and Key1 is at low influence.



The Basis shape key has the stacks fully retracted. Key1 has the base fully extended. Key2A has the left stack fully extended. Key2B has the right stack fully extended. Key2A and Key2B are both relative to Key1 (as you can see in the field in the bottom right of the Shape Keys panel.

The value of Key1 is bound to the position of bones by a driver with two variables. Each variable uses the world Z coordinate of a bone and uses the maximum value to determine how much the base should be extended. The generator polynomial is crafted such that the top of the dominant stack should line up with the bone for that stack.

The value of Key2A is bound to the position of "Bone.L". Its generator parameters are crafted such that when Key1's value

reaches 1, the value of Key2A starts increasing beyond zero. In this way, the top of the left stack will move with bone.L (mostly).

The value of Key2B is bound to the position of "Bone.R". Its generator parameters are similar to Key2A so that the top of the right stack will move with bone.R (mostly).

Since it is quite easy for bone.L and bone.R to be in positions that indicate conflicting values for Key1 there will be times when the bones do not line up with the tops of their respective stacks. If the driver for Key1 was to use Average or Minimum instead of Maximum to determine the value of the shape key then "conflicts" between bone.L and bone.R would be resolved differently. You will choose according to the needs of your animation.

Troubleshooting

Some common problems people may run into when using drivers.

Scripted Expression

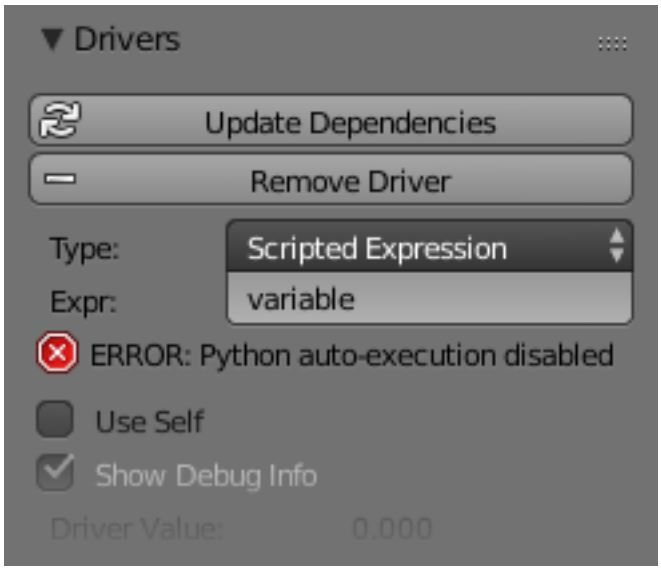


Fig. 2.1262: *Graph Editor* → *Properties* → *Drivers*

Fig. 2.1263: Info Header.

By default Blender will not autorun Python scripts.

If using a *Scripted Expression* Driver Type, you will have to open the file as *Trusted Source*, or set *Auto Run Python Scripts* in *User Preferences* → *File* → *Auto Execution*.

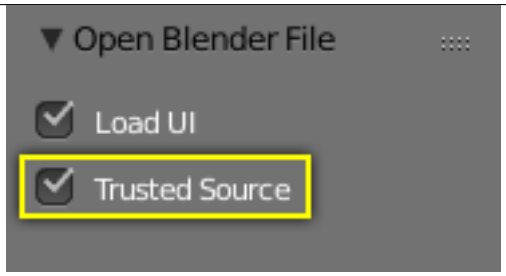


Fig. 2.1264: File Browser.

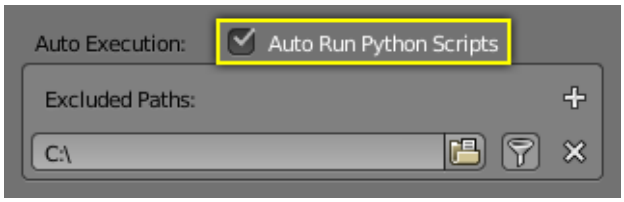


Fig. 2.1265: *User Preference* → *File* → *Auto Execution*

Rotational Properties are Radians

Parts of the User Interface may use different units of measurements for angles, rotation. In the Graph Editor, while working with Drivers, all angles are Radians.

Intra-armature Bone Drivers Can Misbehave

There is a well-known limitation with drivers on bones that refer to another bone in the same armature. Their values can be incorrectly calculated based on the position of the other bone as it was *before* you adjust the current_frame. This can lead to obvious shape glitches when the rendering of frames has a jump in the frame number (either because the blend-file is currently on a different frame number or because you are skipping already rendered frames).

2.7.5 Markers

Markers are used to denote frames at which something significant happens, i.e. it could be that a character's animation starts, the camera changes position, or a door opens. Markers can be given names to make them more meaningful at a quick glance. They are available in many of Blender's editors.

Note: Unlike keyframes, markers are always placed at a whole frame number, you cannot set a marker at frame 2.5.

Markers can be created and edited in the following editors:

- *Graph Editor*
- *Dope Sheet*
- *NLA Editor*
- *Video Sequence Editor*
- *Timeline*

Note: A marker created in one of these editors will also appear in all others that support them.

Types

Next to the standard markers *Pose markers* are another type of markers, which are specific to armatures and shape keys. They are used to denote poses in the Action Editor mode and Shape Keys Editor of Dope Sheet.

Visualization

Standard

Most of the editors visualize markers the same way: as small triangles at their bottom, white if unselected or yellow if selected.

If they have a name, this is shown to their right, in white when the marker is selected.

Sequencer

The *Video Sequence Editor* just adds a vertical dashed line to each marker (gray if the marker is unselected, or white if it is selected).

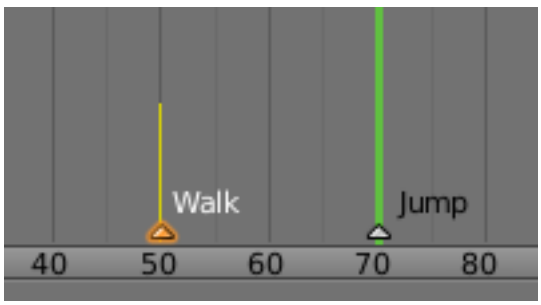


Fig. 2.1266: Markers: small but useful.

3D View

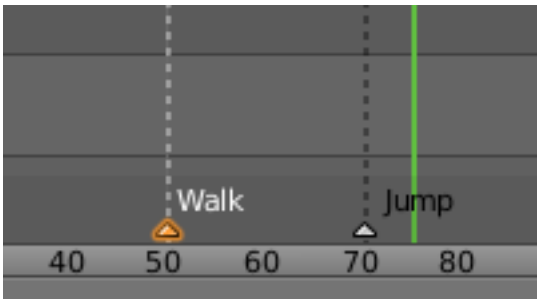


Fig. 2.1267: Markers in the Sequencer.

The 3D View does not allow you to create, edit, and remove markers, it just show their name in the Object Info in the bottom left corner, when on their frame (see Marker in a 3D View).

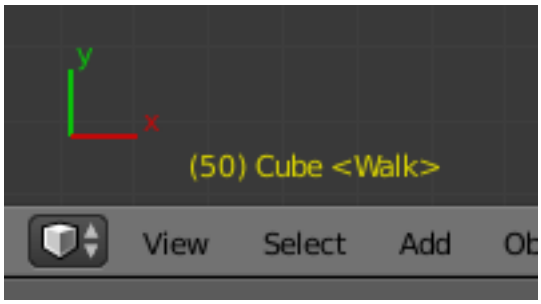


Fig. 2.1268: Marker in a 3D View.

Pose Markers

Pose markers show a diamond-shaped icon in the Dope Sheet. In the NLA editor the pose markers are shown as a red dashed line.

Add Marker

Reference

Mode: All modes
Menu: *Marker* → *Add Marker*
Hotkey: `M` or `Ctrl-Alt-M` in the VSE Editor

The simplest way to add a marker is to move to the frame where you would like it to appear, and press `M`.

Hint: Markers can also be added while playback.

Pose Markers

If *Show Pose Markers* is checked a pose marker and a new pose in the *Pose Library* are added.

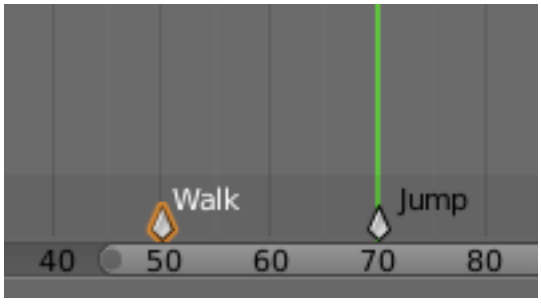


Fig. 2.1269: Pose markers in the Action Editor.

Selecting

Reference

Mode: All modes

Hotkey: RMB

Click RMB on the marker's triangle to select it. Use Shift−RMB to select multiple markers.

In the Graph Editor, Dope Sheet, NLA Editor, and Video Sequence Editor, you can also select all markers with Ctrl−A, and border-select them with Ctrl−B (as usual, LMB to select, RMB to deselect). The corresponding options are found in the Select menu of these editors.

In the Timeline, you can select all markers with A, and border select with B.

Editing

Duplicate Marker

Reference

Mode: All modes
Menu: *Marker* → *Duplicate Marker*
Hotkey: Shift−D

You can duplicate the selected markers by pressing Shift−D. Once duplicated, the new ones are automatically placed in grab mode, so you can move them to the desired location.

Note: Note that unlike most other duplications in Blender, the names of the duplicated markers are not altered at all (no .001 numeric counter append).

Deleting Markers

Reference

Mode: All modes
Menu: *Marker* → *Delete Marker*
Hotkey: X

To delete the selected markers simply press X, and confirm the pop-up message with LMB.

Rename Marker

Reference

Mode: All modes
Menu: *Marker* → *Rename Marker*
Hotkey: `Ctrl-M`

Having dozens of markers scattered throughout your scene's time will not help you much unless you know what they stand for. You can name a marker by selecting it, pressing `Ctrl-M`, typing the name, and pressing the OK button.

Grab/Move Marker

Reference

Mode: All modes
Menu: *Marker* → *Grab/Move Marker*
Hotkey: `G`

Once you have one or more markers selected, press `G`, while hovering with the mouse over the marker bar, to move them, and confirm the move with `LMB` or `Return` (as usual, cancel the move with `RMB`, or `Esc`). Or drag them with the `RMB`.

By default, you grab the markers in one-frame steps, but if you hold `Ctrl`, the markers will move in steps corresponding to one second (according to the scenes *FPS*).

Show Pose Markers

Reference

Mode: Action Editor and Shape Keys Editor
Menu: *Marker* → *Show Pose Markers*

Only Pose markers are shown and editable in Action editor or Shape Keys editor by enabling the *Marker* → *Show Pose Markers* checkbox.

Make Markers Local

Reference

Mode: All modes
Menu: *Marker* → *Make Markers Local*

It is possible to convert standard markers into Pose markers with *Marker* → *Make Markers Local*. Note that the original marker will be gone. If you want to keep it, make a duplicate before you convert.

2.7.6 Shape Keys

Introduction

Shape Keys are used on Objects like *Mesh*, *Curve*, *Surface*, *Lattice*. They are used to animate deform the object vertices into a new shape.

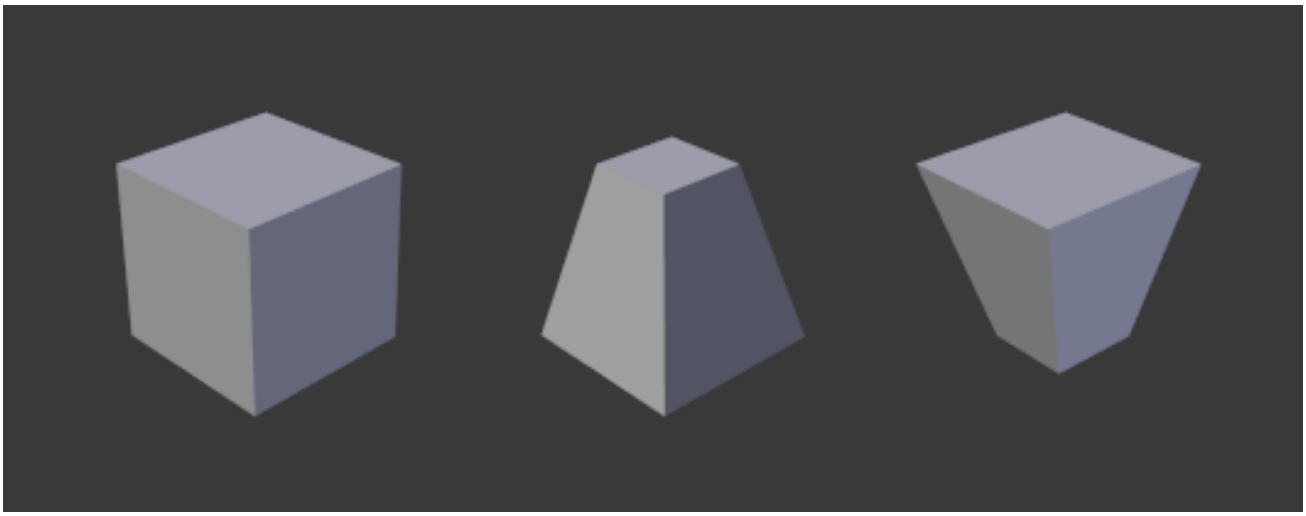


Fig. 2.1270: Example a mesh with different shape keys applied.

There are two types of Shape Keys:

Relative Which are relative to the Basis or selected shape key. They are mainly used as, for limb joints, muscles, or Facial Animation.

Absolute Which are relative to the previous and next shape key. They are mainly used to deform the objects into different shapes over time.

The shape key data, the deformation of the objects vertices, is usually modified in the 3D View by selecting a shape key, then moving the object vertices to a new position.

Shape Keys Panel

Reference

Mode: All modes
Panel: *Properties editor* → *Object Data* → *Shape Keys*

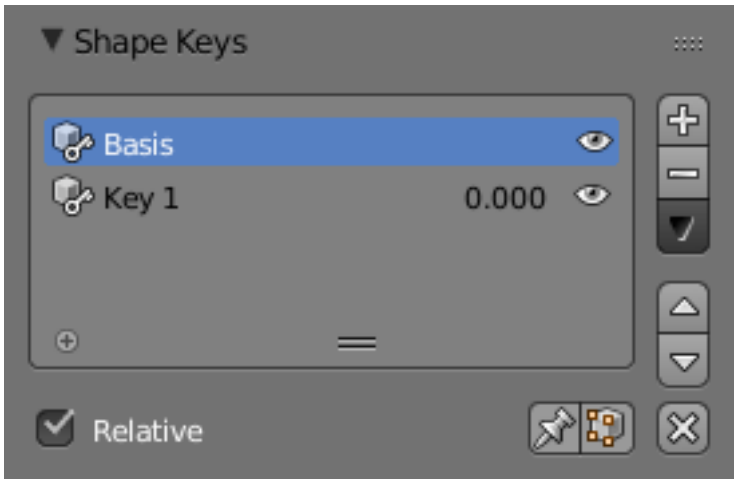


Fig. 2.1271: Shape Keys panel.

Active Shape Key Index A *List Views & Presets*.

Value Current Value of the Shape Key (0.0 to 1.0).

Mute (eye icon) This visually disables the shape key in the 3D View.

Specials

Transfer Shape Key Transfer the active *Shape Key* from a different object. Select two objects, the active Shape Key is copied to the active object.

Join as Shapes Transfer the *Current Shape* from a different object. Select two objects, the Shape is copied to the active object.

Mirror Shape Key If your mesh is nice and symmetrical, in *Object Mode*, you can mirror the shape keys on the X axis. This will not work unless the mesh vertices are perfectly symmetrical. Use the *Mesh* → *Symmetrize* function in *Edit Mode*.

Mirror Shape Key (Topology) This is the same as *Mirror Shape Key* though it detects the mirrored vertices based on the topology of the mesh. The mesh vertices do not have to be perfectly symmetrical for this one to work.

New Shape From Mix Add a new shape key with the current deformed shape of the object.

Delete All Shapes Delete all shape keys.

Relative Set the shape keys to *Relative* or *Absolute*.

Show Only (pin icon) Show the shape of the active shape key without interpolation in the 3D View. *Show Only* is enabled while the object is in *Edit Mode*, unless the setting below is enabled.

Edit Mode Modify the shape key while the object is in *Edit Mode*.

Relative Shape Keys

Relative shape keys deform from a selected shape key. By default, all relative shape keys deform from the first shape key called the Basis shape key.

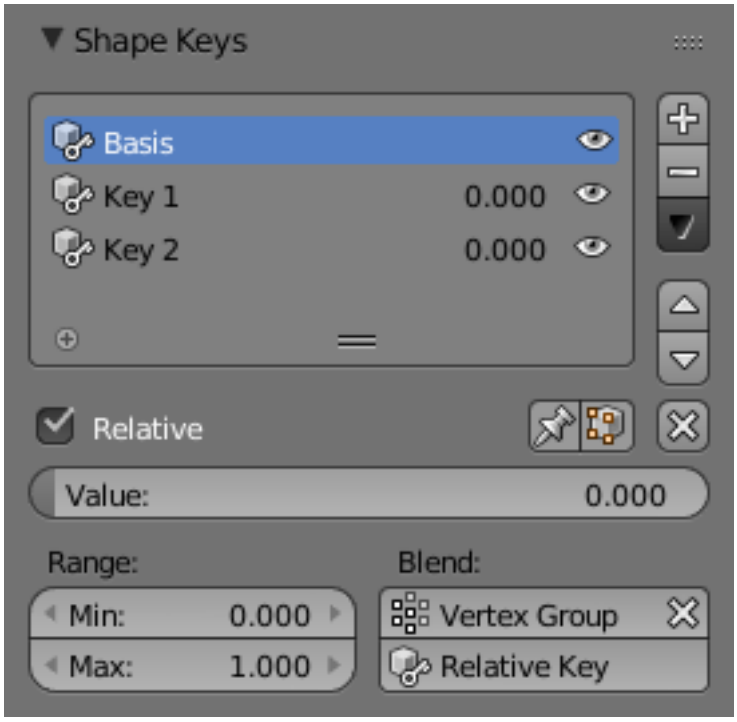


Fig. 2.1272: Relative Shape Keys options.

Clear Weights x Set all values to zero.

Value The value of the active shape key.

Range Min and Max range of the active shape key value.

Blend

Vertex Group Limit the active shape key deformation to a vertex group.

Relative Select the shape key to deform from.

Absolute Shape Keys

Absolute shape keys deform from the previous and to the next shape key. They are mainly used to deform the object into different shapes over time.

Reset Timing (clock icon) Reset the timing for absolute shape keys.

Interpolation This controls the interpolation between shape keys.

Linear, Cardinal, Catmull-Rom, B-Spline

Evaluation Time This is used to control the shape key influence.

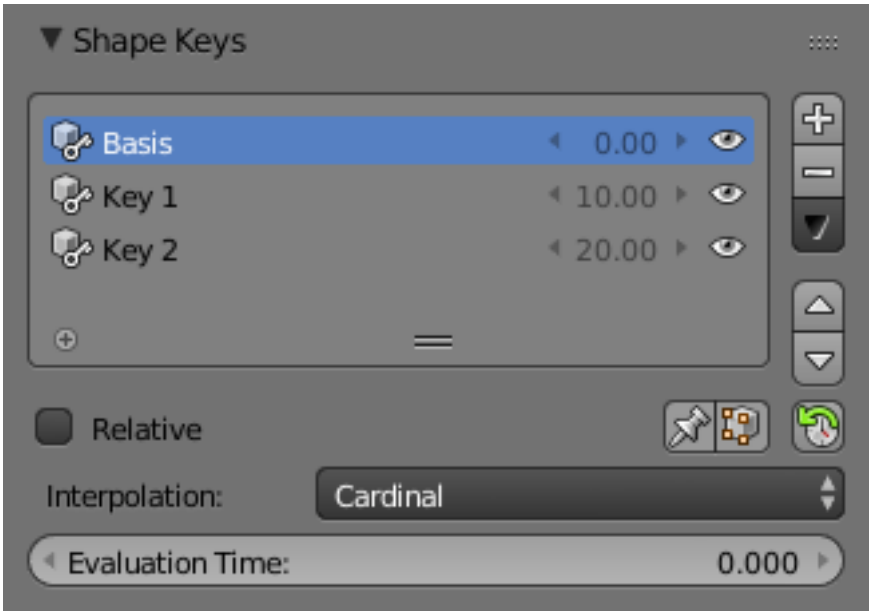


Fig. 2.1273: Absolute Shape Keys options.

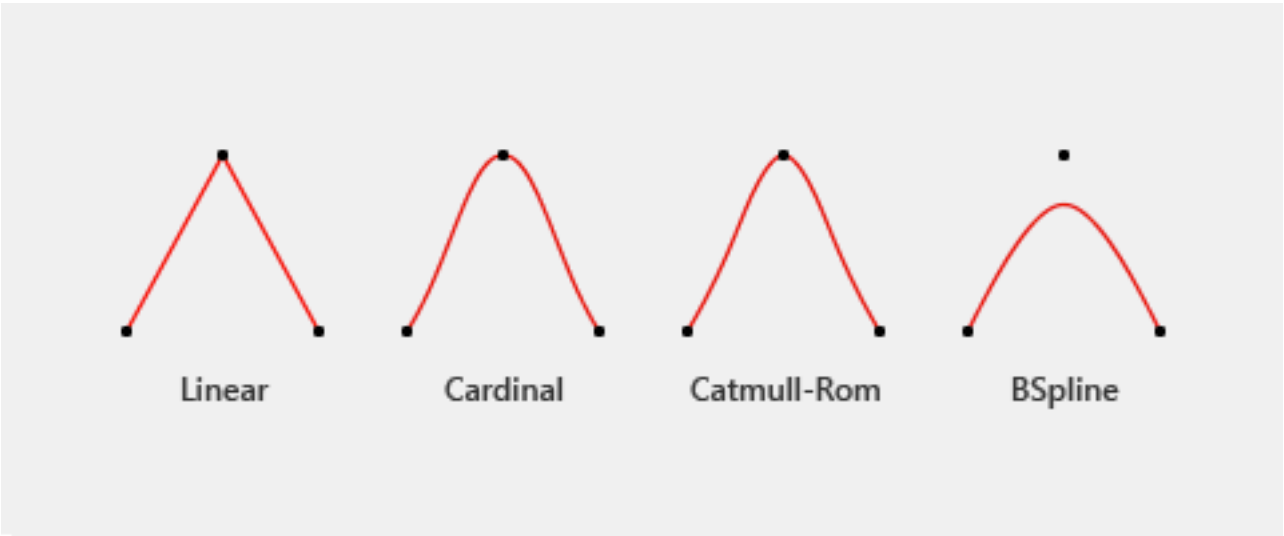


Fig. 2.1274: Different types of interpolation.
The red line represents interpolated values between keys (black dots).

Examples

Reset Timing

For example, if you have the shape keys, Basis, Key_1, Key_2, in that order.

Reset Timing will loop the shape keys, and set the shape keyframes to +0.1:

- Basis 0.1
- Key_1 0.2
- Key_2 0.3

Evaluation Time will show this as frame 100:

- Basis 10.0
- Key_1 20.0
- Key_2 30.0

Evaluation Time

For example, if you have the shape keys, Basis, Key_1, Key_2, in that order, and you reset timing:

- Basis 10.0
- Key_1 20.0
- Key_2 30.0

Workflow

Relative Shape Keys

1. In *Object Mode*, add a new shape keys via the *Shape Key* panel with the + button.
2. "Basis" is the rest shape. "Key 1", "Key 2", etc. will be the new shapes.
3. Switch to *Edit Mode*, select "Key 1" in the *Shape Key* panel.
4. Deform mesh as you want (do not remove or add vertices).
5. Select "Key 2", the mesh will be changed to the rest shape.
6. Transform "Key 2" and keep going for other shape keys.
7. Switch back to *Object Mode*.
8. Set the *Value* for "Key 1", "Key 2", etc. to see the transformation between the shape keys.

   In the figure below, from left to right shows: "Basis", "Key 1", "Key 2" and mix ("Key 1" `1.0` and "Key 2" `0.8`) shape keys in Object Mode.

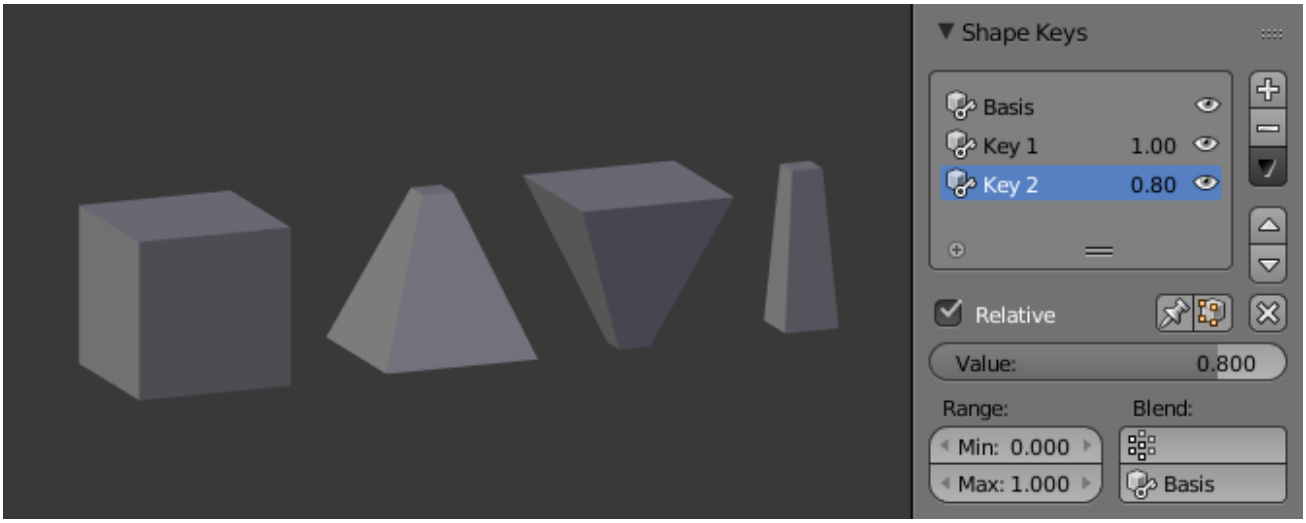


Fig. 2.1275: Relative Shape Keys example.

Absolute Shape Keys

1. Add sequence of shape keys as described above for relative shape keys.
2. Uncheck the *Relative* checkbox.
3. Click the *Reset Timing* button.
4. Switch to *Object Mode*.
5. Drag *Evaluation Time* to see how the shapes succeed one to the next.

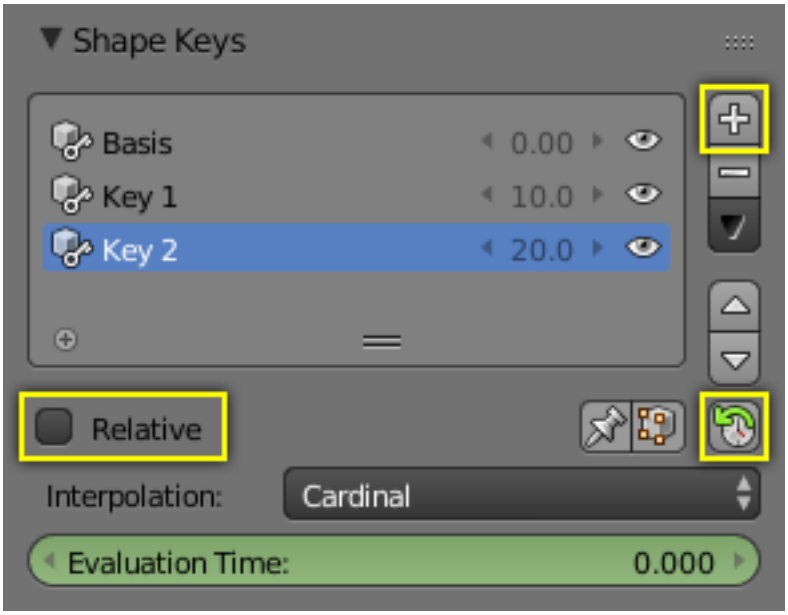


Fig. 2.1276: Absolute Shape Keys workflow.

By adding a *driver* or setting *keyframes* to *Evaluation Time* you can create an animation.

See also:

Shape Key Operators

There are two modeling tools used to control Shape Keys and are found in *Edit Mode*.

2.7.7 Animation Techniques

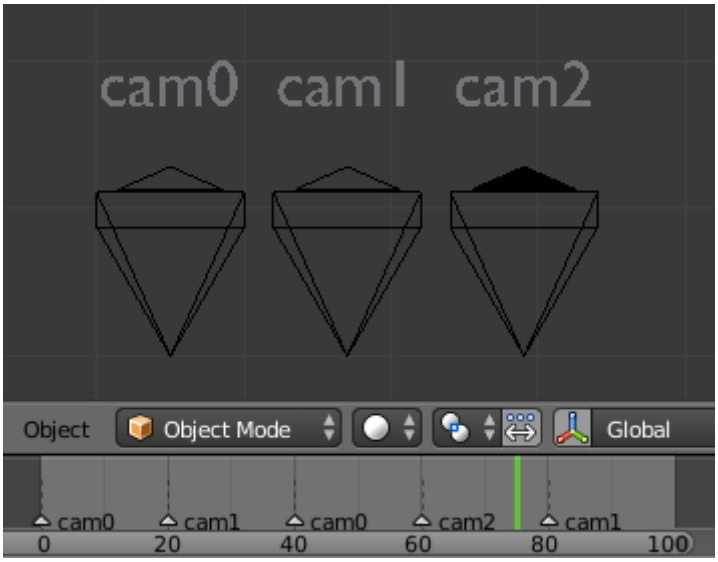
Animating Cameras

These are some basic tools and properties animators may use for the camera.

Switching Cameras

Switching cameras can be done with the *Timeline* operator *Bind Camera to Markers*.

The triangle above the camera will become shaded when active.



First in the Timeline, add a set of markers used to switch cameras. Press `M` to add marker, then `Ctrl-M` to rename, duplicated markers should retain the same name.

1. In the 3D View, select the Camera the Markers will switch to.
2. In the Timeline, select the Marker(s) to switch to the Camera.
3. In the Timeline, press `Ctrl-B` to Bind Cameras to Markers.

Moving Cameras

Move Along a Path

Sometimes it is easier to move objects on path, see *Moving Objects on a Path* for more info.

Fly/Walk Modes

Fly/Walk Mode can be used in conjunction with the timeline record option.

To record your flight path as animation curves.

Lock Camera to View

Lock Camera to View can be used in conjunction with the timeline record option.

To record your viewport navigation as animation curves.

Dolly Zoom

The camera has a set of properties and tools via the *Properties Editor*.

While the camera is moving towards an object the *Focal Length* property can be decreased to produce a *Dolly Zoom* camera effect, or vice versa.

The video below demos the *Dolly Zoom* camera effect.

Moving Objects on a Path

To make objects move along a path is a very common animation need. Think of a complex camera traveling, a train on his rails and most other vehicles can also use “invisible” tracks, the links of a bicycle chain, etc. All these movements could obviously be done with standard F-Curves, but this would be a nightmare! It is much easier and intuitive to define a path materializing the desired movement, and make your object(s) follow it.

Blender features you two different constraints to make an object follow a path, which have different ways to determine/animate the position of their owner along their path.

In Blender, any *curve object* can become a path. A curve becomes a path when its *Path Animation* button is enabled in the *Curve* data panel, but you do not even have to bother about this: once a curve is selected as the target for a "path" constraint, it automatically is enabled.

You can also directly add a "path" from the *Add* → *Curve* → *Path* menu entry (in a 3D View). This will insert in your scene a *three-dimensional* NURBS curve. This is an important point: by default, Blender's curves are *2D* and will not move on the Z axis. To turn a standard curve three-dimensional, enable its *3D* button, in the same *Curve and Surface* editing panel.

One last curve property that is important for a path is its *direction*, which is, for three-dimensional ones, materialized by its small arrows. You can switch it with the *Curve* → *Segments* → *Switch Direction*, `W 2`.

For more on editing path/curves, see the *modeling chapter*.

Note: Shapes on Curves

If you would rather like to have your object's *shape* follow a path (like e.g. a sheet of paper inside a printer), you should use the *Curve Modifier*

Parenting Method

Older versions of Blender did not have constraints to make an object follow a path. They used a different method (deprecated, but still available), based on parenting.

To use this method, select the object that will follow the path, then `Shift` select the curve, and use `Ctrl-P` to bring up the parenting menu. Choose *Follow Path*. The object will now be animated along the path.

The settings for the path animation are in the *Path Animation* panel of the Curve tab in the Properties editor.

Frames Defines the number of frames it takes for the object to travel the path.

Evaluation Time Defines current frame of the animation. By default, it is linked to the global frame number, but could be keyframed to give more control over the path animation.

Follow Causes the curve path children to rotate along the curvature of the path.

Radius Causes the curve path child to be scaled by the set curve radius. See *Curve Extruding*

Offset Children Causes the animation to be offset by the curve path child's time offset value, which can be found in its *Relations Extras* section of the *Object Panel*.

The Follow Path Constraint

The *Follow Path* constraint implements the most "classical" technique. By default, the owner object will walk the whole path only once, starting at frame one, and over 100 frames. You can set a different starting frame in the *Offset* field of the constraint panel, and change the length (in frames) of the path using its *Frames* property (*Curve and Surface* panel).

But you can have a much more precise control over your object's movement along its path by keyframing or defining a *Speed* animation curve for the path's *Evaluation Time* attribute. This curve maps the current frame to a position along the path, from (0.0 to 1.0) (start point to end point).

For more details and examples, see the *Follow Path constraint page*.

The Clamp To Constraint

Another method of keeping objects on a path is to use the *Clamp To* constraint, which implements a more advanced technique. To determine where along the path should lay its owner, its uses the *location of this owner* along a given axis. So to animate the movement of your owner along its target path, you have to animate some way (F-Curves or other indirect animation) its location.

This implies that here, the length of the path have no more any effect – and that by default, the object is static somewhere on the path!

For more details and examples, see the *Clamp To constraint page*.

Using Constraints in Animation

Constraints are a way to control an object's properties (its location/rotation/scale), using either plain static values (like the *"limit" ones*), or (an)other object(s), called "targets" (like e.g. the *"copy" ones*).

Even though these constraints might be useful in static projects, their main usage is obviously in animation. There are two different aspects in constraints' animation:

- You can control an object's animation through the targets used by its constraints (this is a form of indirect animation).
- You can animate constraints' settings.

Controlling Animation with Constraints

This applies only to constraints using target(s). Indeed, these targets can then control the constraint's owner's properties, and hence, animating the targets will indirectly animate the owner.

This indirect "constraint" animation can be very simple, like for example with the *Copy Location constraint*, where the owner object will simply copy the location of its target (with an optional constant offset). But you can also have very complex behaviors, like when using the *Action constraint*, which is a sort of *Animation Driver* for actions!

We should also mention the classical *Child of Constraint*, which creates parent/child relationship. These relationships indeed imply indirect animation (as transforming the parent affects by default all its

children). But the *Child Of* constraint is also very important, as it allows you to parent your objects to bones, and hence use *Armatures* to animate them!

Back to our simple *Copy Location* example, you can have two different behaviors of this constraint:

- When it is *Offset* button is disabled (the default), the location of the owner is "absolutely" controlled by the constraint's target, which means nothing (except other constraints below in the stack...) will be able to control the owner's position. Not even the object's animation curves.
- However, when the *Offset* button is enabled, the location of the owner is "relatively" controlled by the constraint's target. This means that location's properties of the owner are offset from the location of the target. And these owner's location properties can be controlled e.g. by its *Loc...* curves (or actions, or NLA...)!

Example

Let us use the *Copy Location* constraint and its *Offset* button. For example, you can make your owner (let us call it "moon") describe perfect circles centered on the (0.0, 0.0, 0.0) point (using e.g. py-driven *LocX*/*LocY* animation curves, see *Drivers*), and then make it copy the location of a target (called it "earth", for example) with the *Offset* button enabled. Congratulation, you just modeled a satellite in a (simplified) orbit around its planet. Just do the same thing with its planet around its star (which you might call "sun", what do you think?), and why not, for the star around its galaxy.

Here is a small animation of a "solar" system created using (among a few others) the technique described above:

Note that this "solar" system is not realistic at all (the wrong scale, the "earth" is rotating in the wrong direction around the "sun", ...).

You can download the blend-file (download here) used to create this animation.

Note: Animating Constraints Influence

More "classically", you can also animate a few properties of each constraint using animation curves:

- You can animate the *Influence* of a constraint. For example, in the *Example* above, it is used to first stick the camera to the "moon", then to the "earth", and finally to nothing, using two *Copy Location* constraints with *Offset* set, and their *Influence* cross-fading together.
- More anecdotal, you can also, for some constraints using an armature's bone as target, animate where along this bone (between root and tip) lays the real target point (0.0 to 1.0) means influence from the (root or tip).

2.8 Physics

2.8.1 Introduction

Physics allows you to simulate real world physical phenomena. Blender offers a variety for different physics, for example you can use Blender to simulate to following kinds of simulation:

- Smoke
- Rain
- Dust
- Cloth
- Water
- Jello

Particle Systems can be used to simulate many things: hair, grass, smoke, flocks.

Hair is a subset of the particle system, and can be used for strand-like objects, such as hair, fur, grass, quills, etc.

Soft Bodies are useful for everything that tends to bend, deform, in reaction to forces like gravity or wind, or when colliding with other objects... It can be used for skin, rubber, and even clothes, even though there is separate *Cloth Simulation* specific for cloth-like objects.

Rigid Bodies can simulate dynamic objects that are fairly rigid.

Fluids, which include liquids and gases, can be simulated, including *Smoke*.

Force Fields can modify the behavior of simulations.

Gravity

Gravity is a global setting that is applied the same to all physics systems in a scene, which can be found in the scene tab. This value is generally fine left at its default value, at -9.810 in the Z-Axis, which is the force of gravity in the real world. Lowering this value would simulate a lower or higher force of gravity. Gravity denoted g, measurement $m \times s^{-2}$).

Gravity is practically same around whole *Earth*. For rendering scenes from *Moon* use value six times smaller, e.g. 1.622 $m \times s^{-2}$. The *Mars* has g = 3.69.

Note: The gravity value per physics system can be scale down in the *Field Weights tab*.

2.8.2 Baking Physics Simulations

Baking refers to the act of storing or caching the results of a calculation.

It is generally recommended to bake your physics simulations before rendering. Aside from no longer needing to go through the time-consuming process of simulating again, baking can help prevent potential glitches and ensure that the outcome of the simulation remains exactly the same every time.

Note: Most physics simulators in Blender use a similar system, but not all have exactly the same settings available. All the settings are covered here, but individual physics types may not provide all these options.

Compression Compression level for cache files. Some physics caches can be very large (such as smoke). Blender can compress these caches in order to save space.

None Do not compress the cache.

Light Compression optimizes speed of compressing/decompressing operations over file size.

Heavy Compression will result in smaller cache files more than *Light*, however, requires more CPU time to compress/decompress.

External Read and write the cache to disk using a user-specified file path.

Index Number This number specifies which cache should be used when the specified cache directory contains *multiple caches*. `0` refers to the top-most cache, `1` to the second from the top, `2` to the third, and so on.

Use Lib Path Share the disk cache when the physics object is *linked* into another blend-file.

When this option is enabled, linked versions of the object will reference the same disk cache. When disabled, linked versions of the object will use independent caches.

Start Frame on which to start the simulation.

End Frame on which to stop the simulation.

Cache Step Interval for storing simulation data.

Note: Some physics systems (such as particles) allow for positions to be stored only on every nth frame, letting the positions for in-between frames be interpolated. Using a cache step greater than one will result in a smaller cache, but the result may differ from the original simulation.

Bake Start baking. Blender will become unresponsive during most baking operations. The cursor will display as a number representing the bakes' progress.

Free Bake Mark the baked cache as temporary. The data will still exist, but will be removed with the next object modification and frame change. This button is only available when the physics system has been baked.

Calculate To Frame Bake only up to the current frame. Limited by *End* frame set in the cache settings.

Current Cache to Bake Store any temporarily cached simulation data as a bake. Note that playing the animation will try to simulate any visible physics simulations. Depending on the physics type, this data may be temporarily cached. Normally such temporary caches are cleared when an object or setting is modified, but converting it to a bake will "save" it.

Bake All Dynamics Bake all physics systems in the scene, even those of different types. Useful for baking complex setups involving interactions between different physics types.

See *Bake*

Free All Bakes Free bakes of all physics systems in the scene, even those of different types.

See *Free Bake*.

Update All To Frame Bake all physics systems in the scene to the current frame.

See *Calculate To Frame*

Multiple Caches

Blender allows for storing and managing multiple caches at once for the same physics object.

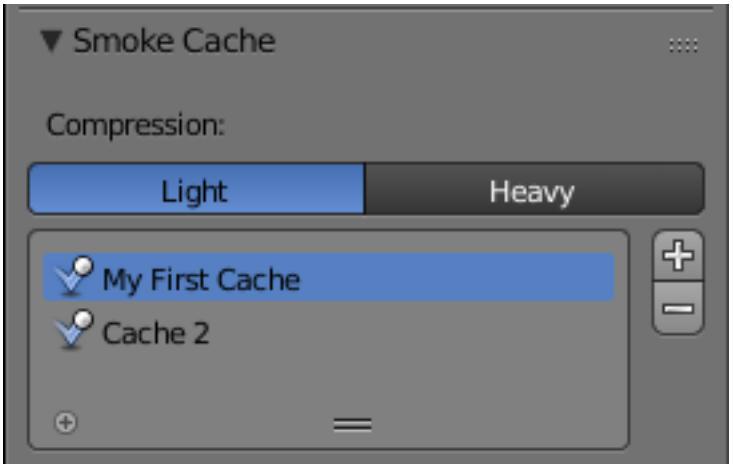


Fig. 2.1277: Two different caches stored simultaneously.

Caches can be added and removed with the `Plus` and `Minus` buttons. Renaming a cache can be done by either double clicking or pressing `Ctrl-LMB` on the desired cache.

2.8.3 Physic Types

Force Fields

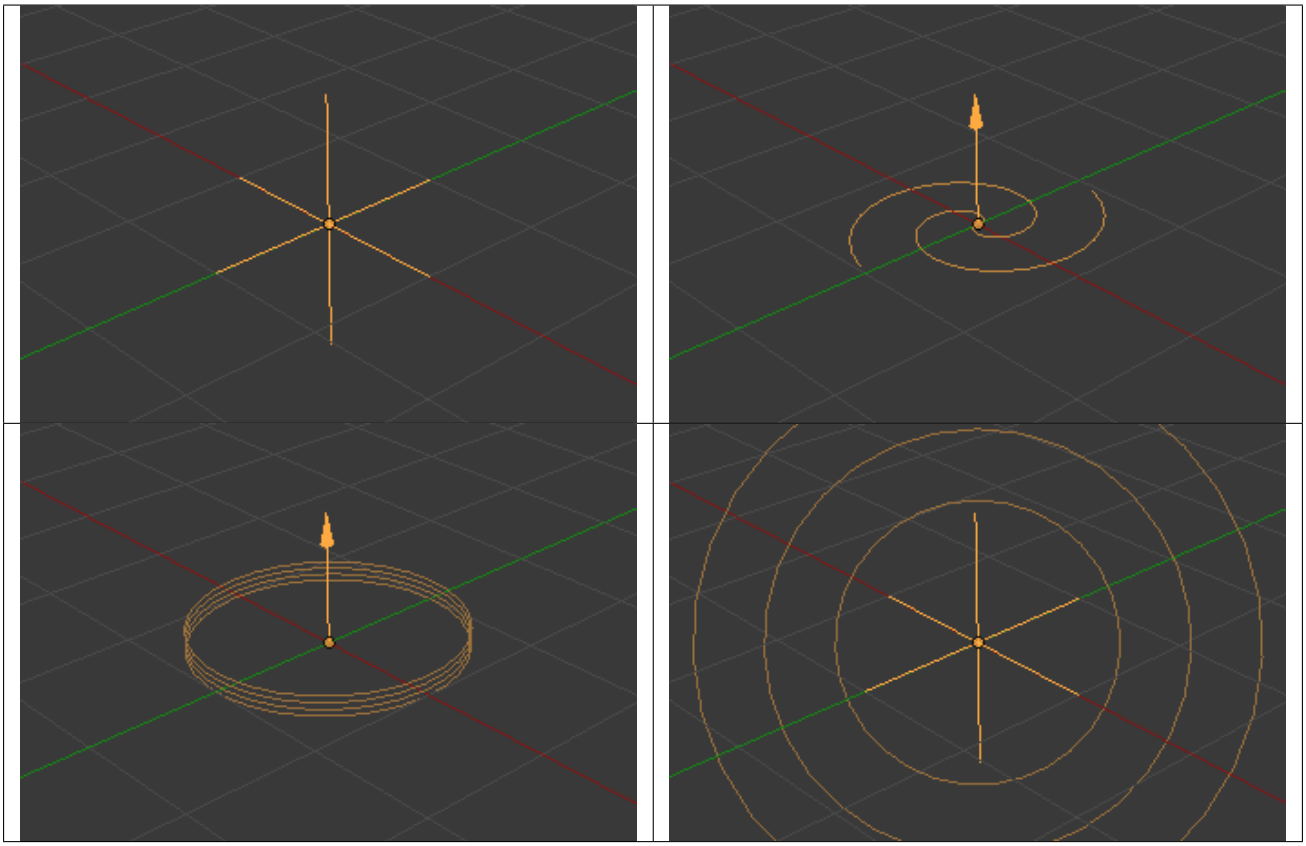
Introduction

Force Fields offer a way to add extra movement to dynamic systems. *Particles*, *Soft Bodies*, Rigid Bodies and *Cloth objects* can all be affected by forces fields. Force Fields automatically affect everything. To remove a simulation or particle system from their influence, simply turn down the influence of that type of Force Field in its Field Weights panel.

- All types of objects and particles can generate fields, but only curve object can bear *Curve Guides* fields.
- Force Fields can also be generated from particles. See *Particle Physics*
- The objects need to share at least one common layer to have effect.

  You may limit the effect on particles to a group of objects (see the *Particle Physics* page).

Table 2.74: Force field types



Creating a Force Field

Reference

Mode: Object Mode
Panel: *Physics* → *Fields*

To create a single Force Field, you can select *Add* → *Force Field* and select the desired force field. This method creates an Empty with the force field attached.

To create a field from an existing object you have to select the object and change to the *Physics* tab. Select the field type in the *Fields* menu.

The fields have many options in common, these common options are explained for the *Spherical* field.

Note: After changing the fields *Fields* panel or deflection *Collision* panel settings, you have to recalculate the particle, softbody or cloth system by *Free Cache*, this is not done automatically. You can clear the cache for all selected objects with `Ctrl-B` → *Free cache selected*.

Particles react to all kind of *Force Fields*, Soft Bodies only to *Spherical*, *Wind*, *Vortex* (they react on *Harmonic* fields but not in a useful

way).

Common Field Settings

Most Fields have the same settings, even though they act very differently. Settings unique to a field type are described below. Curve Guide and Texture Fields have very different options.

Shape The field is either a *Point*, with omni-directional influence, or a *Plane*, constant in the XY-plane, changes only in Z direction.

Strength The strength of the field effect. This can be positive or negative to change the direction that the force operates in. A force field's strength is scaled with the force object's scale, allowing you to scale up and down scene, keeping the same effects.

Flow Convert effector force into air flow velocity.

Noise Adds noise to the strength of the force.

Seed Changes the seed of the random noise.

Effect Point You can toggle the field's effect on particle *Location* and *Rotation*

Collision Absorption Force gets absorbed by collision objects.

Falloff

Here you can specify the shape of the force field (if the *Fall-off* Power is greater than 0).

Sphere Falloff is uniform in all directions, as in a sphere.

Tube Fall off results in a tube shaped force field. The Field's *Radial falloff* can be adjusted, as well as the *Minimum* and *Maximum* distances of the field.

Cone Fall off results in a cone shaped force field. Additional options are the same as those of *Tube* options.

Z Direction *Fall-off* can be set to apply only in the direction of the positive Z Axis, negative Z Axis, or both.

Power (Power) How the power of the force field changes with the distance from the force field. If *r* is the distance from the center of the object, the force changes with $1/ r^{power}$. A *Fall-off* of 2 changes the force field with $1/ r^2$, which is the falloff of gravitational pull.

Max Distance Makes the force field only take effect within a specified maximum radius (shown by an additional circle around the object).

Min Distance The distance from the object center, up to where the force field is effective with full strength. If you have a *Fall-off* of 0 this parameter does nothing, because the field is effective with full strength up to *Max Distance* (or the infinity). Shown by an additional circle around the object.

Types

Force

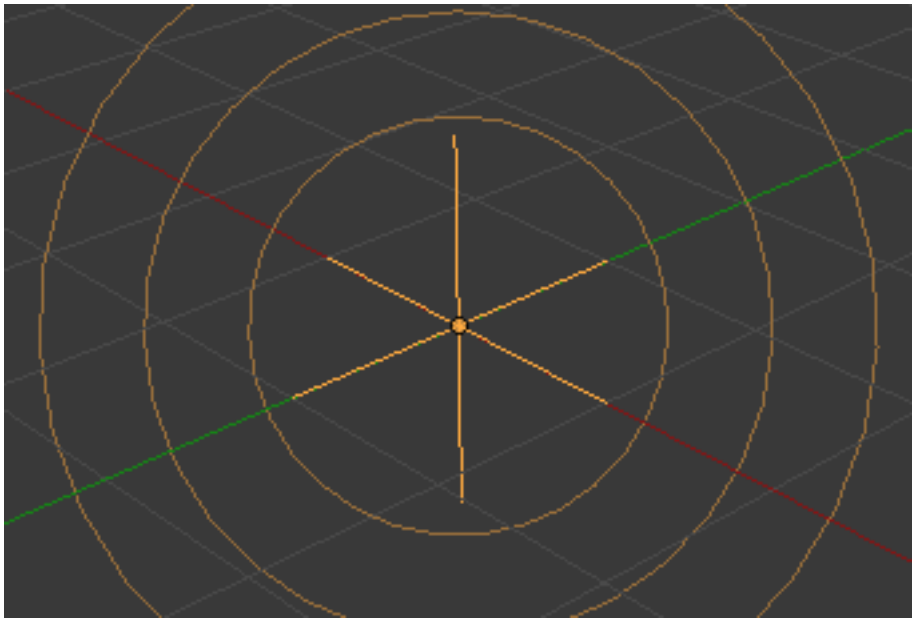


Fig. 2.1278: Force force field.

The *Force* field is the simplest of the fields. It gives a constant force towards (positive strength) or away from (negative strength) the object's center. Newtonian particles are attracted to a field with negative strength, and are blown away from a field with positive strength.

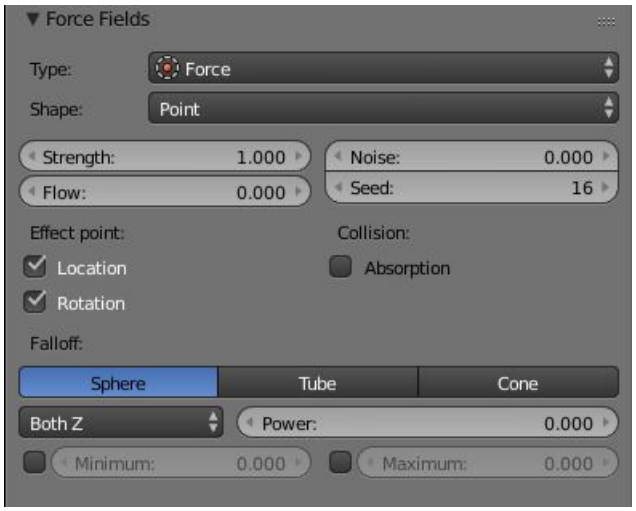


Fig. 2.1279: UI for a Force force field.

For *Boids Particles* a field with positive strength can be used as a *Goal*, a field with negative strength can be used as *Predator*. Whether *Boids* seek or fly goals/predators depends on the *Physics* settings of the Boids.

Wind

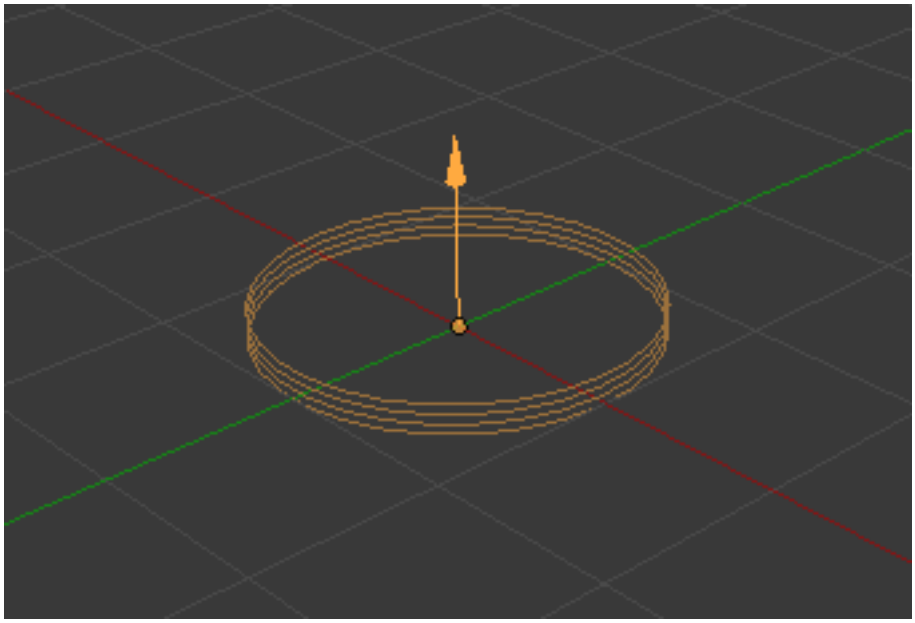


Fig. 2.1280: Wind force field.

The *Wind* force field gives a constant force in a single direction, along the force object's local Z axis. The strength of the force is visualized by the spacing of the circles shown.

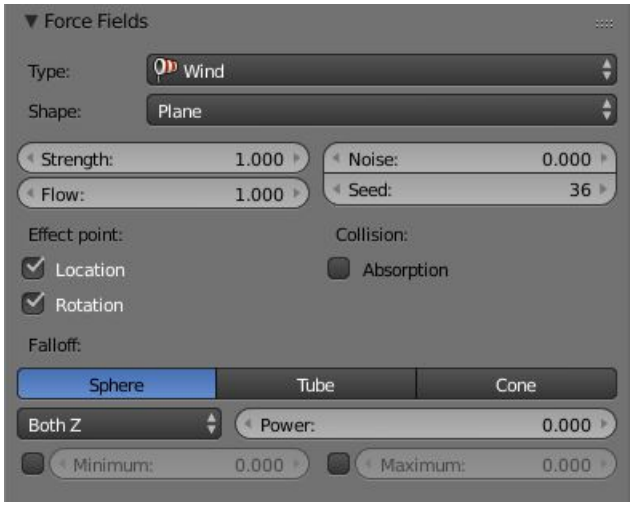


Fig. 2.1281: UI for a Wind force field.

Vortex

The *Vortex* force field gives a spiraling force that twists the direction of points around the force object's local Z axis. This can be useful for making a swirling sink, or tornado, or kinks in particle hair.

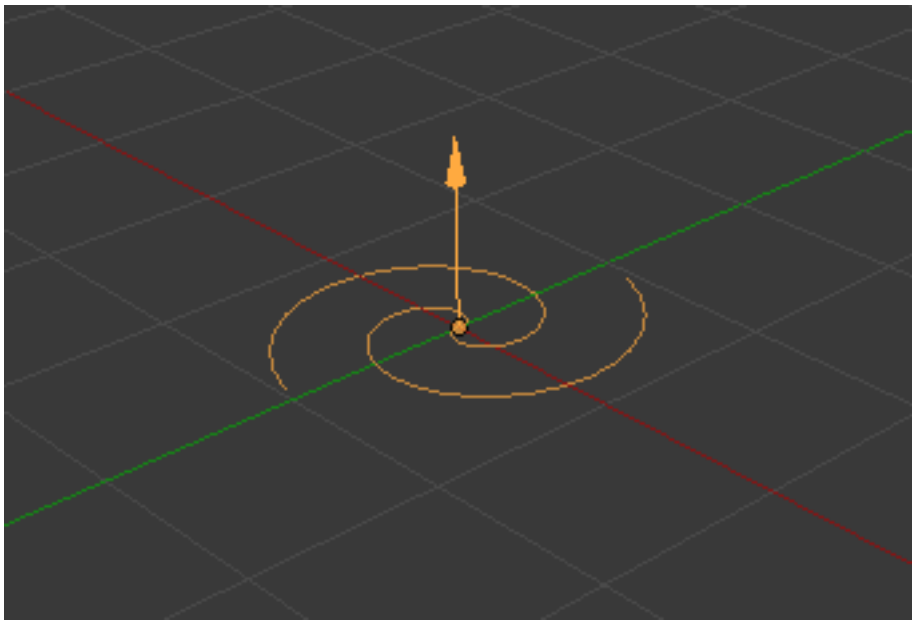


Fig. 2.1282: Vortex force field.

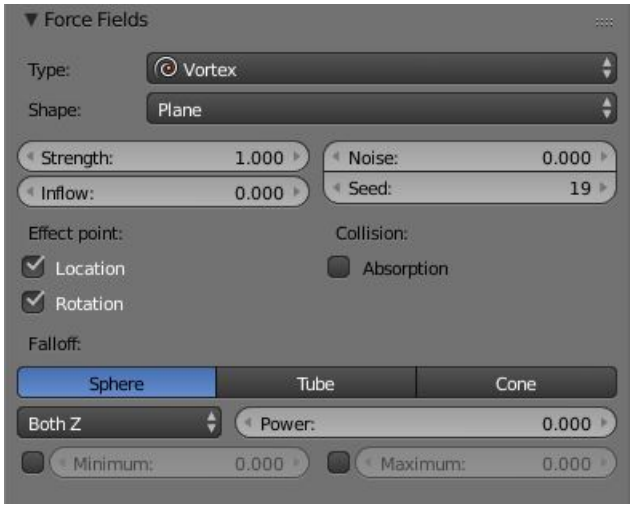


Fig. 2.1283: UI for a Vortex force field.

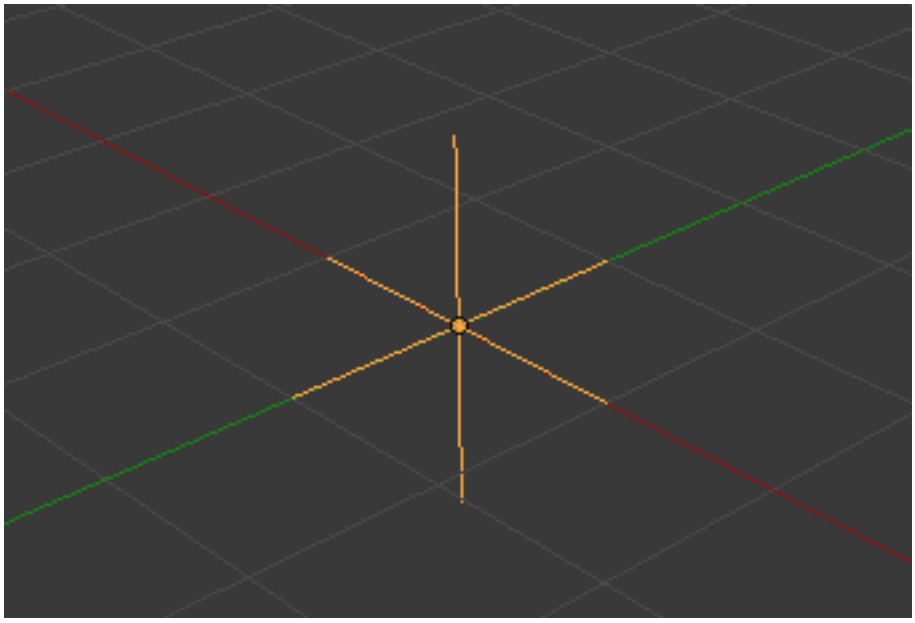


Fig. 2.1284: Magnetic force field.

Magnetic

This field depends on the speed of the particles. It simulates the force of magnetism on magnetized objects.

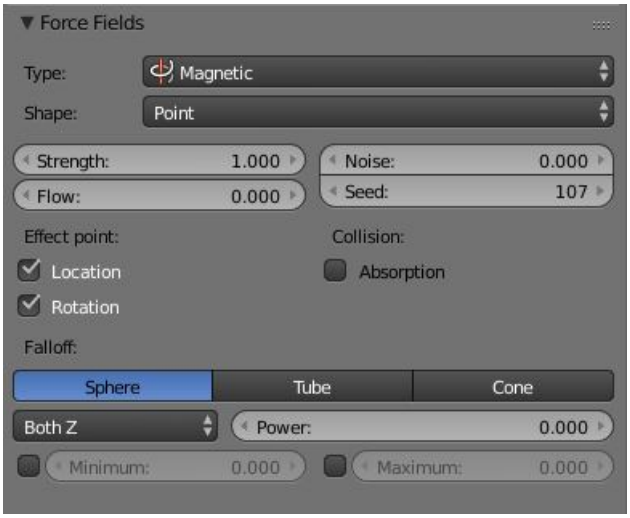


Fig. 2.1285: UI for a Magnetic force field.

Harmonic

In a *Harmonic* force field, the source of the force field is the zero point of a harmonic oscillator (spring, pendulum). If you set the *Damping* parameter to 1, the movement is stopped in the moment the object is reached. This force field is really special if you assign it to particles.

Rest Length Controls the rest length of the harmonic force.

Multiple Springs Causes every point to be affected by multiple springs.

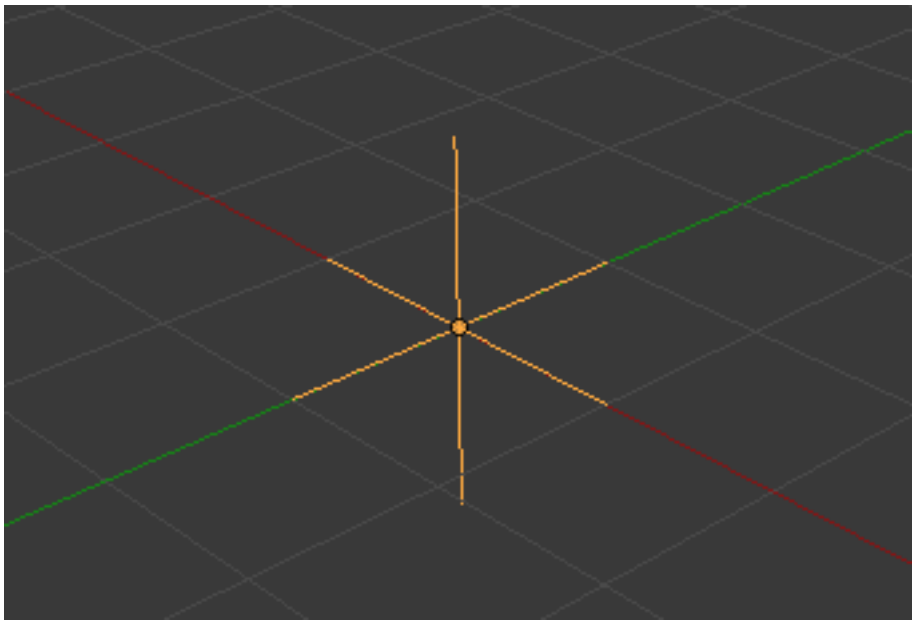


Fig. 2.1286: Harmonic force field.

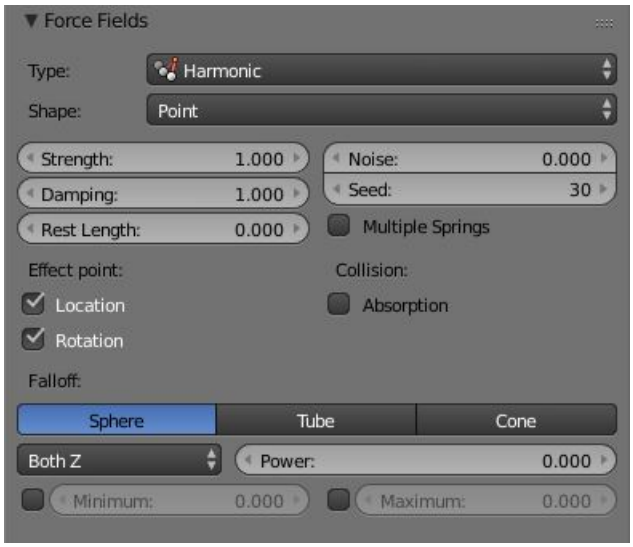


Fig. 2.1287: UI for a Harmonic force field.

Normally every particle of the field system influences every particle of the target system. Not with *Harmonic* ! Here every target particle is assigned to a field particle. So particles will move to the place of other particles, thus forming shapes.

Tutorial: Particles forming Shapes.

Charge

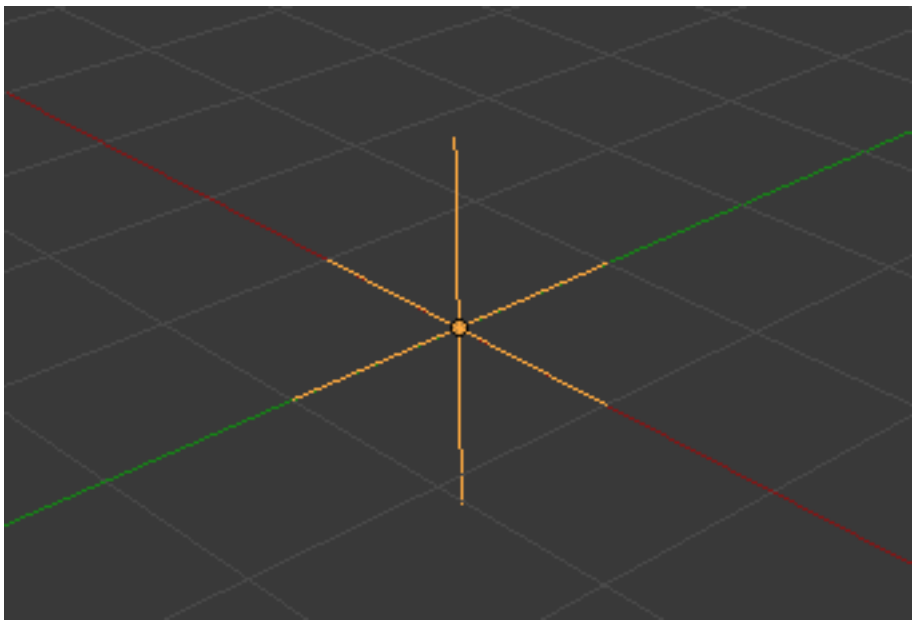


Fig. 2.1288: Charge force field.

A *Charge* force field is similar to spherical field except it changes behavior (attract/repulse) based on the effected particles charge field (negative/positive), like real particles with a charge. This mean this field has only effect on particles that have also a *Charge* field (else, they have no "charge", and hence are unaffected)!

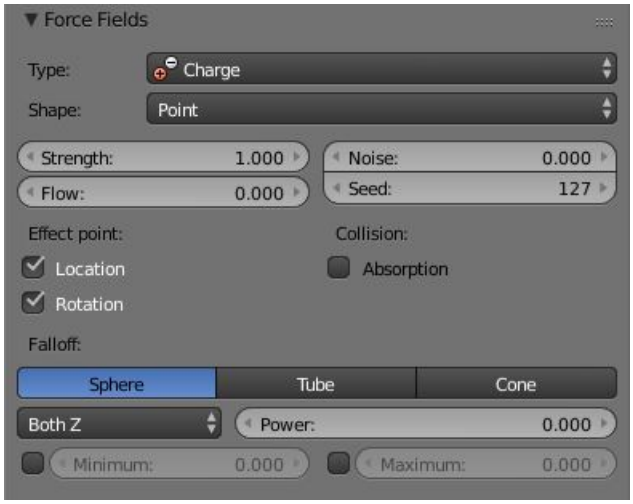


Fig. 2.1289: UI for a Charge force field.

Lennard Jones

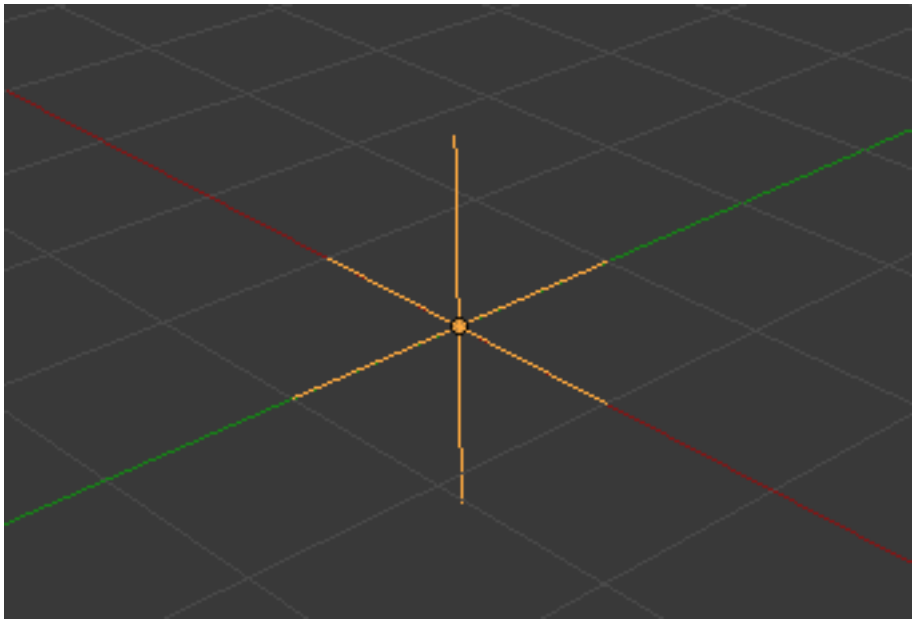


Fig. 2.1290: Lennard Jones force field.

The *Lennard Jones* force field is a very short range force with a behavior determined by the sizes of the effector and effected particle. At a distance smaller than the combined sizes the field is very repulsive and after that distance it is attractive. It tries to keep the particles at an equilibrium distance from each other. Particles need to be at a close proximity to each other to be effected by this field at all.

Particles can have for example both a charge and a Lennard-Jones potential, which is probably something for the nuclear physicists amongst us.

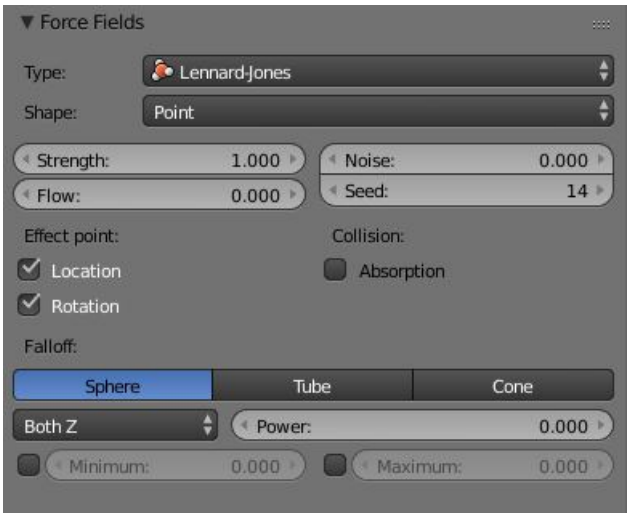


Fig. 2.1291: UI for a Lennard Jones force field.

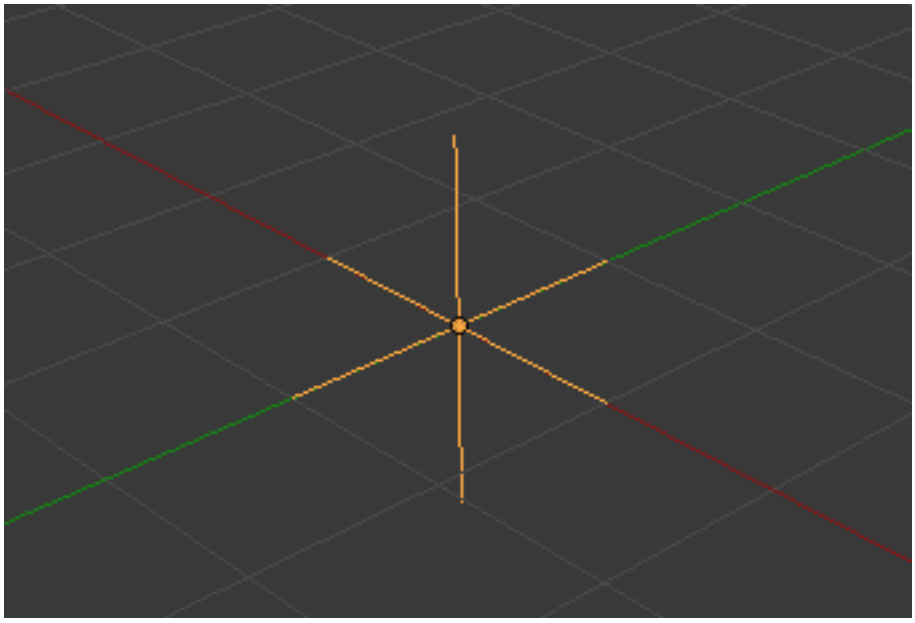


Fig. 2.1292: Texture force field.

Texture

You can use a *Texture* force field to create an arbitrarily complicated force field, which force in the three directions is color coded. Red is coding for the X-axis, green for the Y-axis and blue for the Z-axis (like the color of the coordinate axes in the 3D View). A value of 0.5 means no force, a value larger than 0.5 acceleration in negative axis direction (like -Z), a value smaller than 0.5 acceleration in positive axis direction (like +Z).

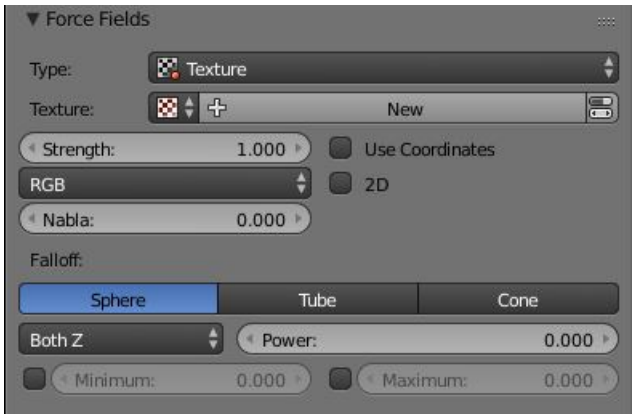


Fig. 2.1293: UI for a Texture force field.

Texture mode This sets the way a force vector is derived from the texture.

RGB Uses the color components directly as the force vector components in the color encoded directions. You need an RGB texture for this, e.g. an image or a colorband. So a *Blend* texture without a colorband would not suffice.

Gradient Calculates the force vector as the 3D-gradient of the intensity (grayscale) of the texture. The gradient vector always points to the direction of increasing brightness.

Curl Calculates the force vector from the curl of the 3D-RGB texture (rotation of RGB vectors). This also works only with a color texture. It can be used for example to create a nice looking turbulence force with a color clouds texture with Perlin noise.

Nabla It is the offset used to calculate the partial derivatives needed for *Gradient* and *Curl* texture modes.

Use Object Coordinates Uses the emitter object coordinates (and rotation & scale) as the texture space the particles use. Allows for moving force fields, that have their coordinates bound to the location coordinates of an object.

Root Texture Coordinates This is useful for hair as it uses the texture force calculated for the particle root position for all parts of the hair strand.

2D The *2D* button disregards the particles z-coordinate and only uses particles x&y as the texture coordinates.

Remember that only procedural texture are truly 3D.

Examples

- A single colored texture (0.5, 0.0, 0.5) creates a force in the direction of the positive Y-axis, e.g. hair is orientated to the Y-axis.
- A blend texture with colorband can be used to created a force "plane". E.g. on the left side (0.5, 0.5, 0.5), on the right side (1.0, 0.5, 0.5) you have a force plane perpendicular to XY (i.e. parallel to Z). If you use an object for the coordinates, you can use the object to push particles around.
- An animated wood texture can be used to create a wave like motion.

Curve Guide

The *Curve Guide* is used to force particles to follow a certain path defined by a *Curve Object*. A typical scenario would be to move a red blood cell inside a vein, or to animate the particle flow in a motor. You can use *Curve Guide* s also to shape certain hair strands.

Note: You can also use the *Particle Edit Mode* to define a path.

Since you can animate curves as Softbody or any other usual way, you may build very complex animations while keeping great control and keeping the simulation time to a minimum.

The option *Curve Follow* does not work for particles. Instead you have to set *Angular Velocity* (*Particle system* tab) to *Spin* and leave the rotation constant (i.e. do not turn on *Dynamic*).

Curve Guide s affect all particles on the same layer, independently from their distance to the curve. If you have several guides in a layer, their fields add up to each other (the way you may have learned it in your physics course). But you can limit their influence radius by changing there *Minimum Distance* (see below).

Note: The Curve Guide does not effect *Softbodys*.

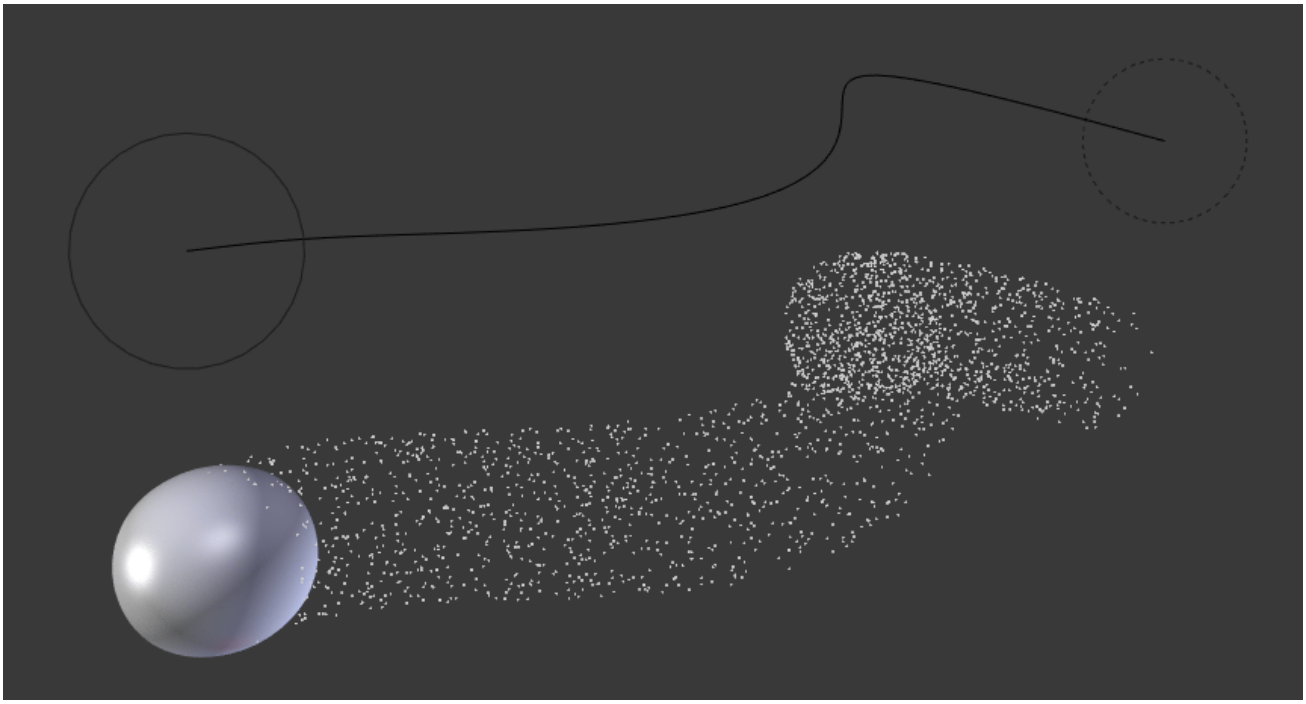


Fig. 2.1294: Curve Guide force field.

Options

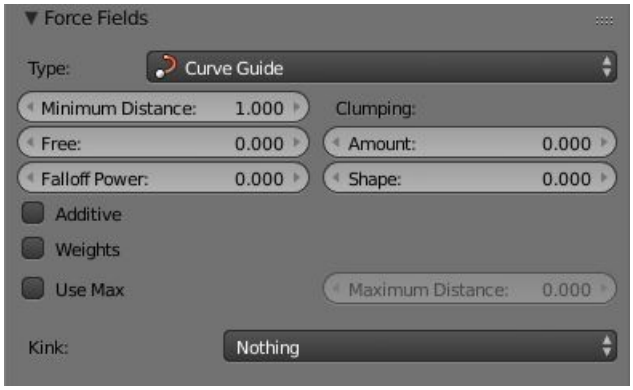


Fig. 2.1295: UI for a Curve Guide force field.

Minimum Distance The distance from the curve, up to where the force field is effective with full strength. If you have a *Fall-off* of 0 this parameter does nothing, because the field is effective with full strength up to *Max Distance* (or the infinity). *Min Distance* is shown with a circle at the endpoints of the curve in the 3D View.

Free Fraction of particle life time, that is not used for the curve.

Fall-off This setting governs the strength of the guide between *Min Distance* and *Max Distance*. A *Fall-off* of 1 means a linear progression.

A particle follows a *Curve Guide* during its lifetime, the velocity depends on its lifetime and the length of the path.

Additive If you use *Additive*, the speed of the particles is also evaluated depending on the *Fall-off*.

Weights Use Curve weights to influence the particle influence along the curve.

Maximum Distance / Use Max The maximum influence radius. Shown by an additional circle around the curve object.

The other settings govern the form of the force field along the curve.

Clumping Amount The particles come together at the end of the curve (1) or they drift apart (-1).

Shape Defines the form in which the particles come together. +0.99: the particles meet at the end of the curve. 0: linear progression along the curve. -0.99: the particles meet at the beginning of the curve.

Kink Changes the shape that the particles can take:

Curl The radius of the influence depends on the distance of the curve to the emitter.

Radial A three dimensional, standing wave.

Wave A two dimensional, standing wave.

Braid Braid.

Roll An one dimensional, standing wave.

It is not so easy to describe the resulting shapes, so have a look at the example below.

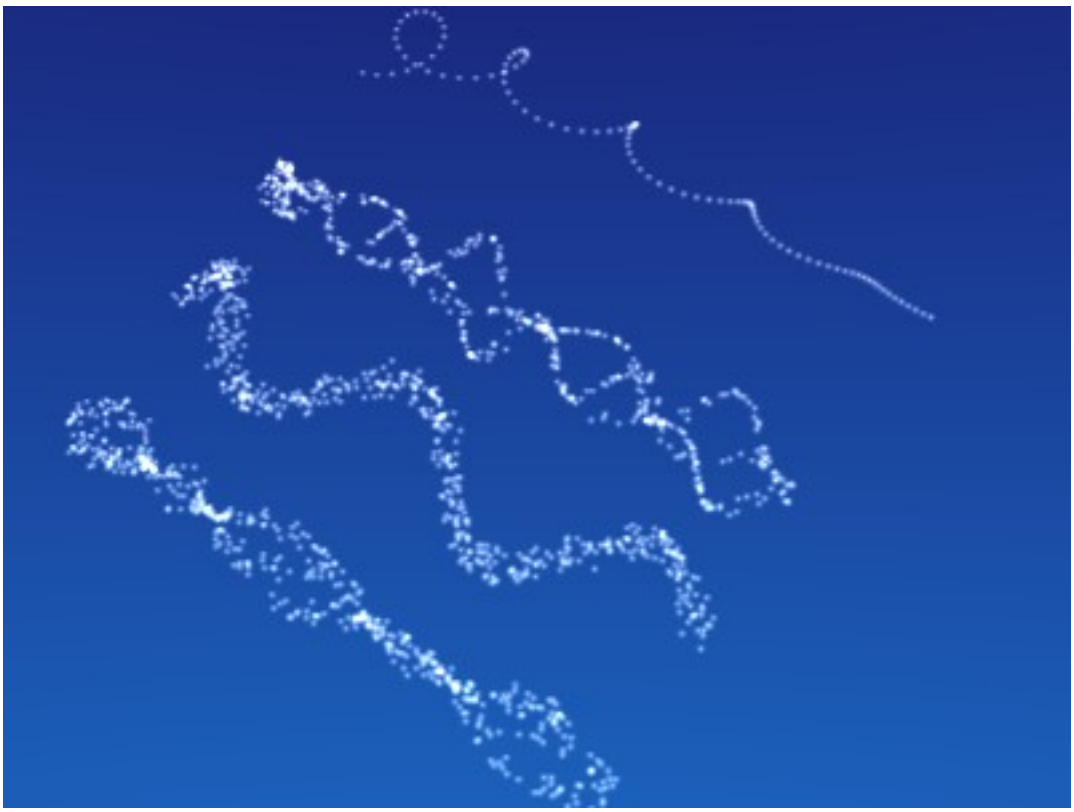


Fig. 2.1296: Kink options of a curve guide. From left to right: Radial, Wave, Braid, Roll. Animation.

Frequency The frequency of the offset.

Shape Adjust the offset to the beginning/end.

Amplitude The Amplitude of the offset.

Boid

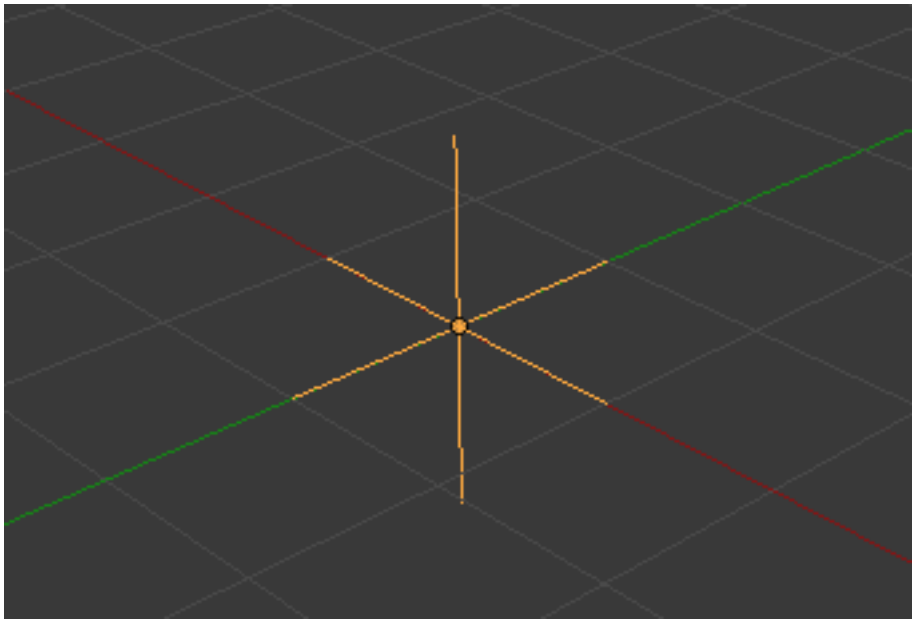


Fig. 2.1297: Boid force field.

Boid probably comes from theoretical works. *Boids* is an artificial life program, developed by Craig Reynolds in 1986, which simulates the flocking behavior of birds. His paper on this topic was published in 1987 in the proceedings of the ACM SIGGRAPH conference. The name refers to a "bird-like object", but its pronunciation evokes that of "bird" in a stereotypical New York accent. As with most artificial life simulations, Boids is an example of emergent behavior; that is, the complexity of Boids arises from the interaction of individual agents (the boids, in this case) adhering to a set of simple rules.

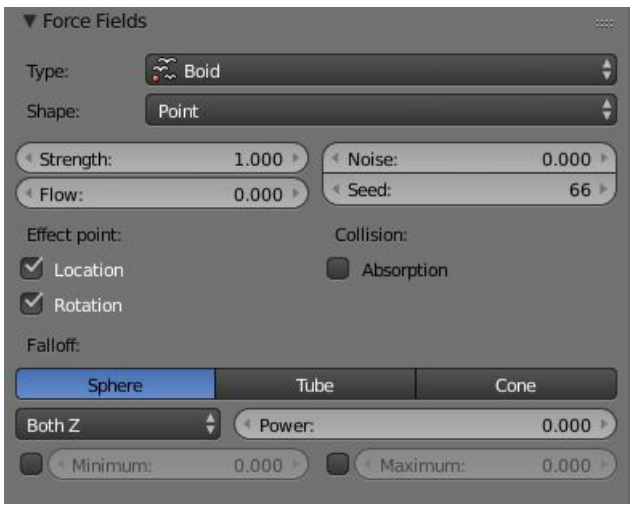


Fig. 2.1298: UI for a Boid force field.

The rules applied in the simplest Boids world are as follows: separation: steer to avoid crowding local flock mates alignment: steer towards the average heading of local flock mates cohesion: steer to move toward the average position (center of mass) of local flock mates

More complex rules can be added, such as obstacle avoidance and goal seeking.

Turbulence

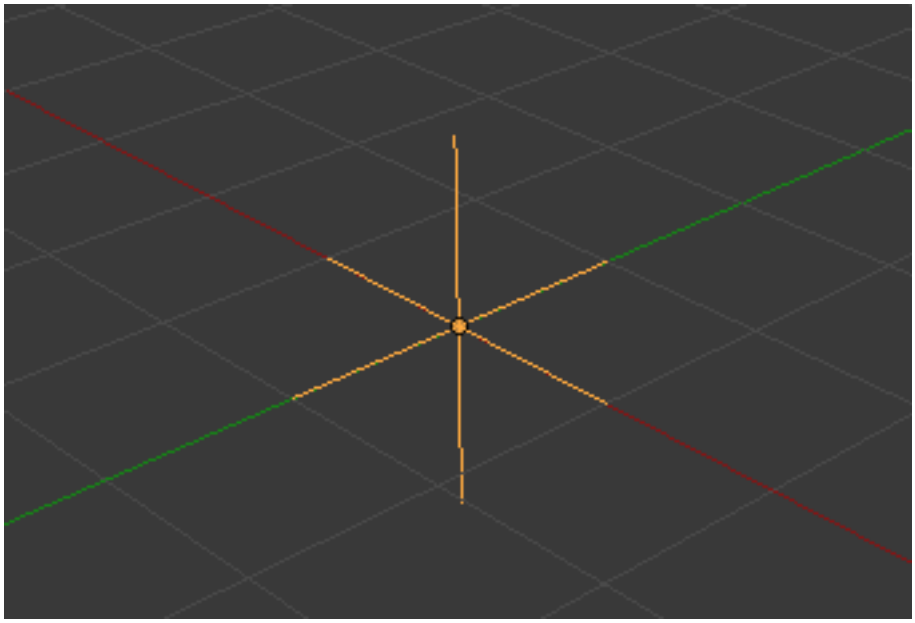


Fig. 2.1299: Turbulence force field.

A *Turbulence* force field creates a random & chaotic 3D noise effect, similar to jets of water or geysers under the ocean.

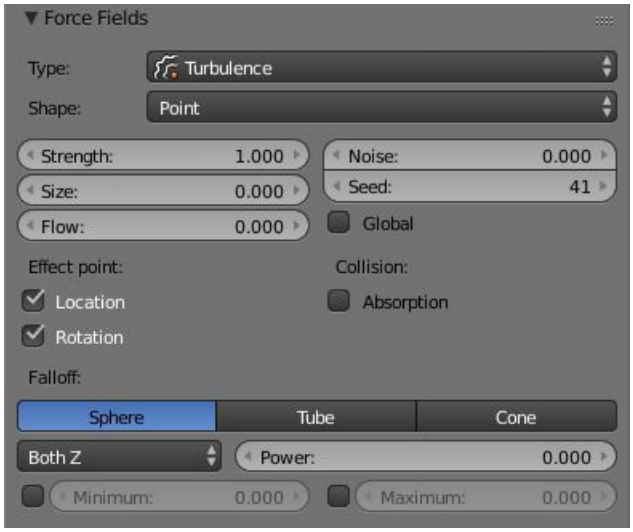


Fig. 2.1300: UI for a Turbulence force field.

Size Indicates the scale of the noise.

Global Makes the size and strength of the noise relative to the world, instead of the object it is attached to.

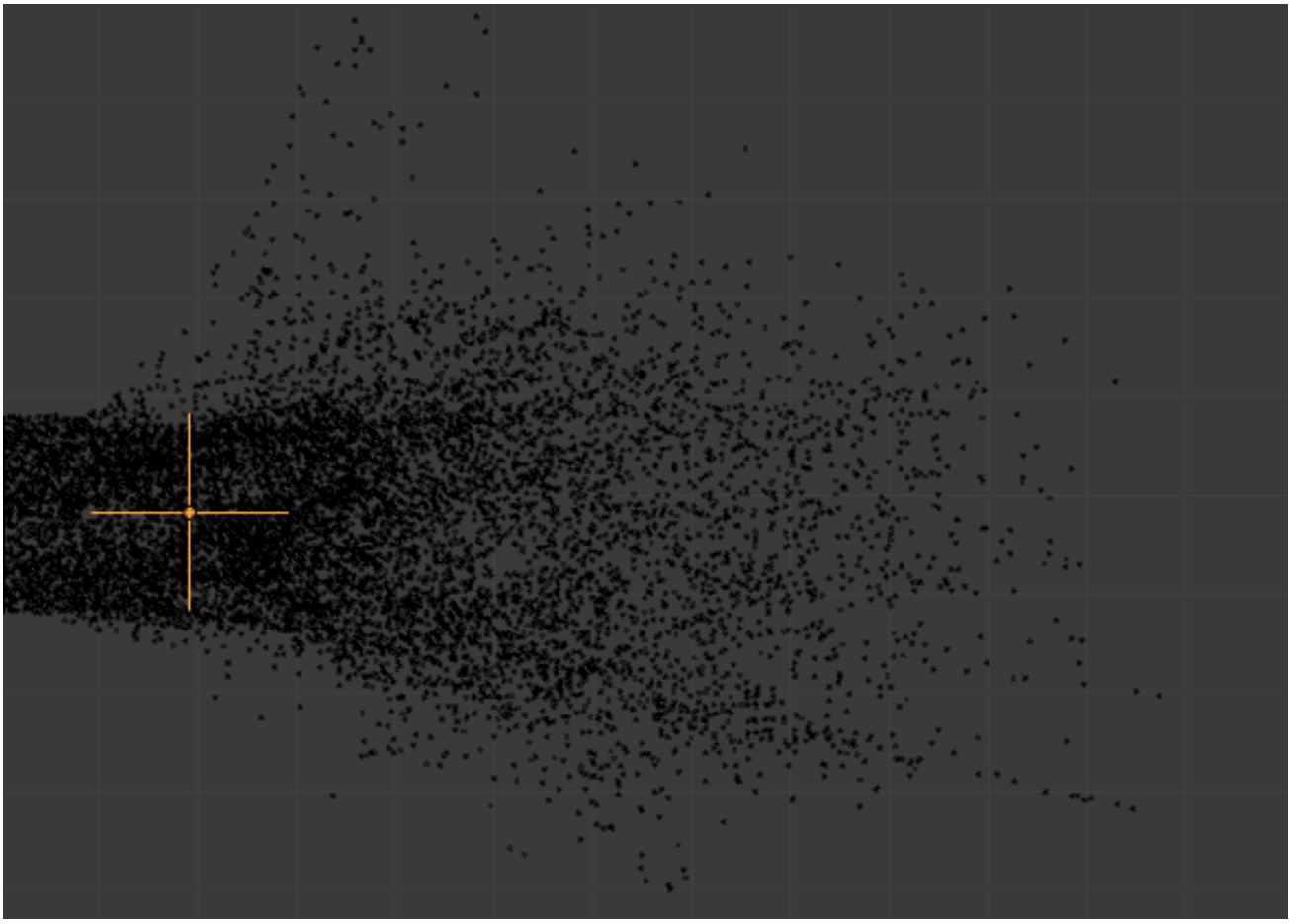


Fig. 2.1301: Turbulence force field affecting a particle system.

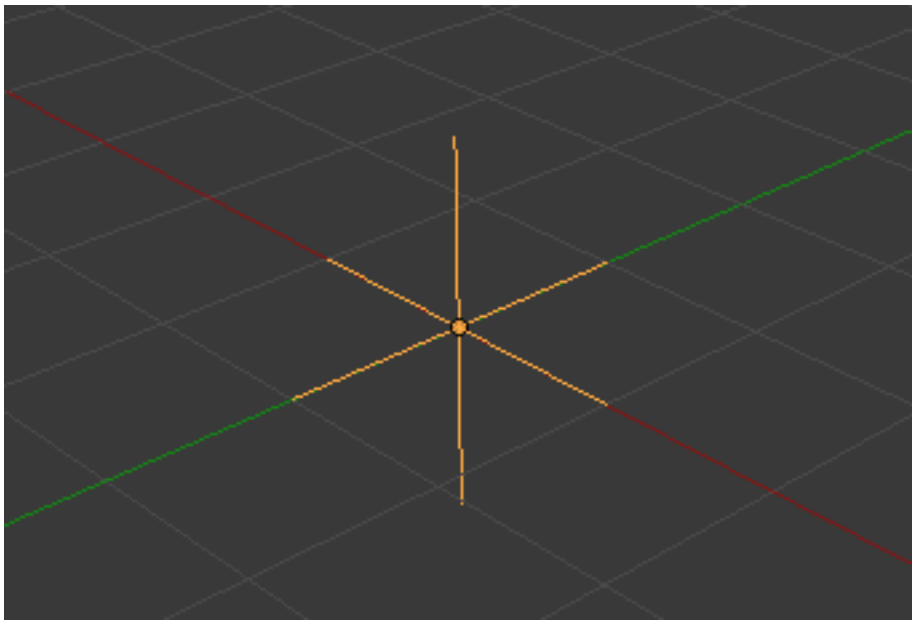


Fig. 2.1302: Drag force field.

Drag

A *Drag* force field resists particle motion by slowing it down.

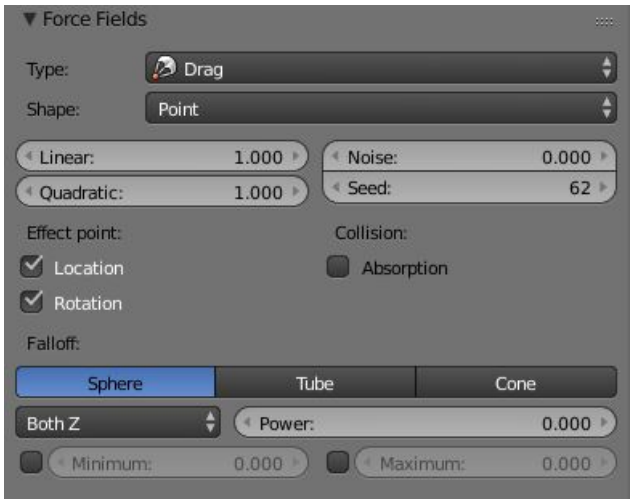


Fig. 2.1303: UI for a Drag force field.

Linear Drag component proportional to velocity.

Quadratic Drag component proportional to the square of the velocity.

Smoke Flow

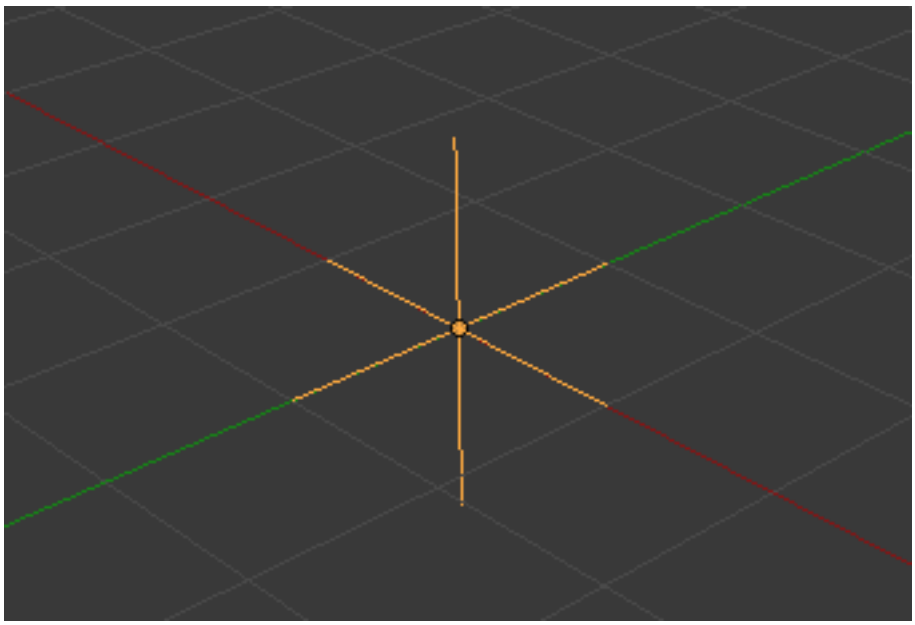


Fig. 2.1304: Smoke Flow force field.

A *Smoke Flow* force field directs the smoke within a smoke simulation.

Collisions

Particles, *Soft Bodies* and *Cloth objects* may collide with mesh objects. *Boids* try to avoid *Collision* objects.

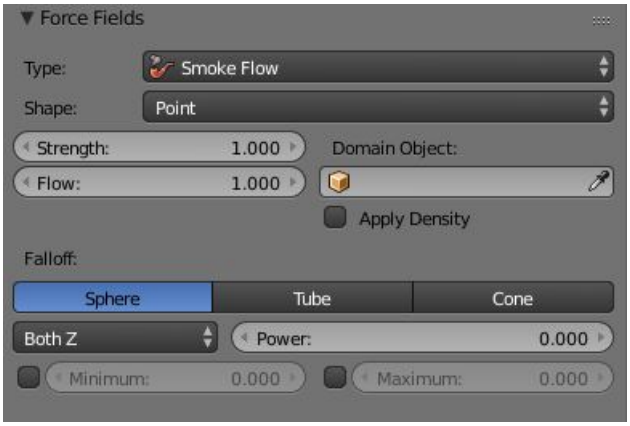


Fig. 2.1305: UI for a Smoke Flow force field.

- The objects need to share at least one common layer to have effect.
- You may limit the effect on particles to a group of objects (in the *Field Weights panel*).
- *Deflection* for softbody objects is difficult, they often penetrate the colliding objects.
- Hair particles ignore deflecting objects (but you can animate them as softbodies which take deflection into account).

  If you change the deflection settings for an object you have to recalculate the particle, softbody or cloth system by *Free Cache*, this is not done automatically. You can clear the cache for all selected objects with `Ctrl-B` → *Free cache selected*.

Reference

Mode: Object Mode
Panel: *Physics* → *Collision*

Options

Permeability Fraction of particles passing through the mesh.

Stickiness How much particles stick to the object.

Kill Particles Deletes Particles upon impact.

Damping Factor Damping during a collision (independent of the velocity of the particles).

Random damping Random variation of damping.

Friction Factor Friction during movements along the surface.

Random friction Random variation of friction.

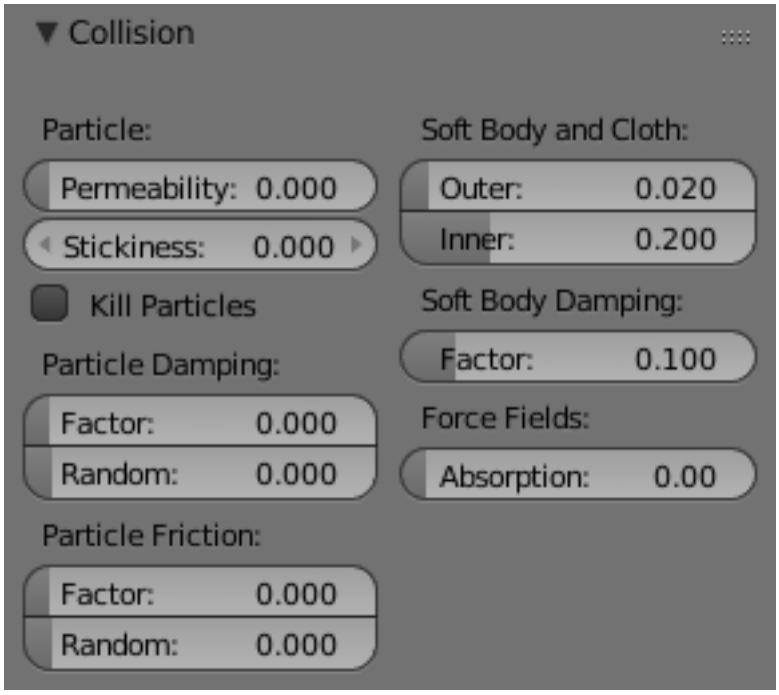


Fig. 2.1306: Collision Panel.

Fig. 2.1307: A softbody vertex colliding with a plane.

Soft Body and Cloth Interaction

Outer Size of the outer collision zone.

Inner Size of the inner collision zone (padding distance).

Outside and inside is defined by the face normal, depicted as blue arrow in Fig. *A softbody vertex colliding with a plane.*.

Damping Factor Damping during a collision.

Softbody collisions are difficult to get perfect. If one of the objects move too fast, the soft body will penetrate the mesh. See also the section about *Soft Bodies*.

Force Field Interaction

Absorption A deflector can also deflect effectors. You can specify some collision/deflector objects which deflect a specific portion of the effector force using the *Absorption* value. 100% absorption results in no force getting through the collision/deflector object at all. If you have three collision object behind each other with e.g. 10%, 43% and 3%, the absorption ends up at around 50% $100(1-0.1)(1-0.43)(1-0.03)$.

Examples

Here is a *Meta* object, dupliverted to a particle system emitting downwards, and deflected by a mesh cube:

Fig. 2.1308: Deflected Particles.

Hints

- Make sure that the normals of the mesh surface are facing towards the particles/points for correct deflection.
- Hair particles react directly to force fields, so if you use a force field with a short range you do not need necessarily collision.
- Hair particles avoid their emitting mesh if you edit them in *Particle Edit Mode*. So you can at least model the hair with collision.

Cloth Simulations

Introduction

Cloth simulation is one of the hardest aspects of CG, because it is a deceptively simple real-world item that is taken for granted, yet actually has very complex internal and environmental interactions. After years of development, Blender has a very robust cloth simulator that is used to make clothing, flags, banners, and so on. Cloth interacts with and is affected by other moving objects, the wind and other forces, as well as a general aerodynamic model, all of which is under your control.

Fig. 2.1309: Cloth Example.

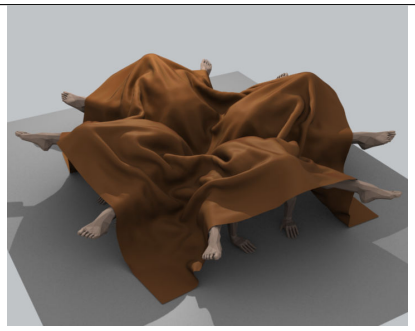


Fig. 2.1310: Cloth on carved wooden men (made by motorsep).

Fig. 2.1311: Cloth Example.

A piece of cloth is any mesh, open or enclosed, that has been designated as cloth. The *Cloth* panels are located in the *Physics* tab and consist of three panels of options. Cloth is either an open or closed mesh and is mass-less, in that all cloth is assumed to have the same density, or mass per square unit.

Cloth is commonly modeled as a mesh grid primitive, or a cube, but can also be, for example, a teddy bear. However, Blender's *Softbody system* provides better simulation of closed meshes; Cloth is a specialized simulation of fabrics.

Once the object is designated as Cloth, a Cloth *modifier* will be added to the object's modifier stack automatically. As a *modifier* then, it can interact with other modifiers, such as *Armature* and *Smooth*. In these cases, the ultimate shape of the mesh is computed in accordance with the order of the modifier stack. For example, you should smooth the cloth *after* the modifier computes the shape of the cloth.

So you edit the Cloth settings in two places: use the Physics buttons to edit the properties of the cloth and use the Modifier stack to edit the Modifier properties related to display and interaction with other modifiers.

You can *Apply* the cloth modifier to freeze, or lock in, the shape of the mesh at that frame, which removes the modifier. For example, you can drape a flat cloth over a table, let the simulation run, and then apply the modifier. In this sense, you are using the simulator to save yourself a lot of modeling time.

Results of the simulation are saved in a cache, so that the shape of the mesh, once calculated for a frame in an animation, does not have to be recomputed again. If changes to the simulation are made, you have full control over clearing the cache and re-running the simulation. Running the simulation for the first time is fully automatic and no baking or separate step interrupts the workflow.

Computation of the shape of the cloth at every frame is automatic and done in the background; thus you can continue working while the simulation is computed. However, it is CPU-intensive and depending on the power of your PC and the complexity of the simulation, the amount of CPU needed to compute the mesh varies, as does the lag you might notice.

Note: Do not jump ahead

If you set up a cloth simulation but Blender has not computed the shapes for the duration of the simulation, and if you jump ahead a lot of frames forward in your animation, the cloth simulator may not be

able to compute or show you an accurate mesh shape for that frame, if it has not previously computed the shape for the previous frame(s).

Workflow

A general process for working with cloth is to:

- Model the cloth object as a general starting shape.
- Designate the object as a "cloth" in the *Physics* tab of the Properties editor.
- Model other deflection objects that will interact with the cloth. Ensure the Deflection modifier is last on the modifier stack, after any other mesh deforming modifiers.
- Light the cloth and assign materials and textures, UV-unwrapping if desired.
- If desired, give the object particles, such as steam coming off the surface.
- Run the simulation and adjust Options to obtain satisfactory results. The Timeline editors VCR controls are great for this step.
- Optionally age the mesh to some point in the simulation to obtain a new default starting shape.
- Make minor edits to the mesh on a frame-by-frame basis to correct minor tears.

Tip: To avoid unstable simulation, ensure that the cloth object does not penetrate any of the deflection objects or an unstable simulation will result.

Settings

Settings

Cloth

Presets Contains a number of preset cloth examples.

Quality Set the number of simulation steps per frame. Higher values result in better quality, but is slower.

Speed Adjust how fast time flows in the cloth simulation.

Material

Mass The mass of the cloth material.

Structural Overall stiffness of the cloth.

Bending Wrinkle coefficient. Higher creates more large folds.

Damping

Spring Damping of cloth velocity. Higher values give a more smooth result (less jiggling).

Air Air normally has some thickness which slows falling things down.

Velocity Damps the velocity to help the cloth reach the final resting position faster.

Pinning

The first thing you need when pinning cloth is a *Vertex Group*. There are several ways of doing this including using the Weight Paint tool to paint the areas you want to pin (see the *Weight Paint* section of the manual). The weight of each vertex in the group controls how strongly it is pinned.

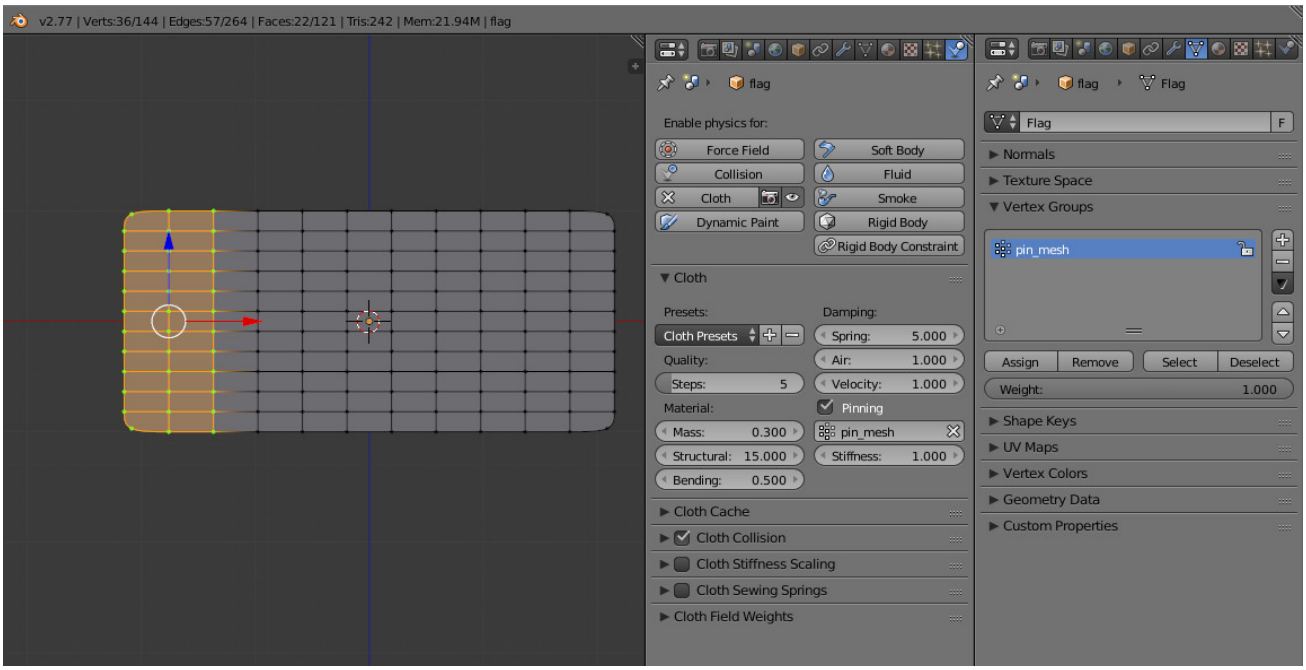


Fig. 2.1312: Cloth Pinning.

Once you have a vertex group set, things are pretty straightforward; all you have to do is press the *Pinning of cloth* button in the *Cloth* panel and select which vertex group you want to use, and the stiffness you want it at.

Stiffness Target position stiffness. You can leave the stiffness as it is; the default value of 1 is fine.

Pinning Clothing To An Armature

Clothing can be simulated and pinned to an armature. For example, a character could have a baggy tunic pinned to the character's waist with a belt.

The typical workflow for pinning:

- Set the armature to its bind pose.
- Model clothing that encloses but does not penetrate the character's mesh.
- Parent the clothing objects to the armature. The armature will now have several child meshes bound to it.
- Create a new vertex group on each cloth object for its pinned vertices
- Add vertexes to be pinned to this vertex group and give these vertices non-zero weights (you probably want weight = 1). For example the belt area of the tunic would be in the vertex group and have weight one.
- Designate the clothing objects as "cloth" in the Physics tab of the Properties editor. Make sure the *Cloth* modifier is below the *Armature* modifier in the modifier stack.
- press the *Pinning of Cloth* button in the *Cloth* panel and select the vertex group.
- Designate the character's mesh as "collision" object in the Physics tab of the Properties editor.
- The clothing is now ready. Non-pinned vertices will be under control of the Cloth modifier. Pinned vertices will be under control of the Armature modifier.

Note: When animating or posing the character you must begin from the bind pose. Move the character to its initial pose over several frames so the physics engine can simulate the clothing moving. Very fast movements and teleport jumps can break the physics simulation.

Dynamic Mesh

Normally cloth uses the state of the object in the first frame to compute the natural rest shape of the cloth, and keeps that constant throughout the simulation. This is reasonable for fully realistic scenes, but does not quite work for clothing on cartoon style characters that use a lot of squash and stretch.

When *Dynamic Mesh* is enabled, the rest shape is recalculated every frame, allowing unpinned cloth to squash and stretch following the character with the help of an Armature modifier, but otherwise move freely under control of the physics simulation.

Dynamic Mesh is incompatible with using a shape key to specify the rest shape.

Cloth Sewing Springs

Another method of restraining cloth similar to pinning is sewing springs. Sewing springs are virtual springs that pull vertices in one part of a cloth mesh toward vertices in another part of the cloth mesh. This is different from pinning which binds vertices of the cloth mesh in place or to another object. A clasp on a cloak could be created with a sewing spring. The spring could pull two corners of a cloak about a character's neck. This could result in a more realistic simulation than pinning the cloak to the character's neck since the cloak would be free to slide about the character's neck and shoulders.

Sewing springs are created by adding extra edges to a cloth mesh that are not included in any faces. They should connect vertices in the mesh that should be pulled together. For example the corners of a cloak.

To activate the springs, enable the *Cloth Sewing Springs* panel.

Sewing Force Maximum force that can be applied by sewing springs. Zero means unbounded, but it is not recommended to leave the field at zero in most cases, as it can cause instability due to extreme forces in the initial frames where the ends of the sewing springs are far apart.

The *Cloth Sewing Springs* panel also contains controls for shrinking the actual cloth faces.

Shrinking Group Vertex group that is used to vary the intensity of the shrinking effect over the cloth.

Min Fraction of the size to shrink the cloth by around vertices with weight 0 (or those not in vertex group.) The value 0.01 means shrink by 1% etc.

Max Fraction of the size to shrink the cloth by around vertices with weight 1.

Like unbounded sewing forces, immediately applying a large amount of shrink can cause instability, so it is advisable to keyframe these fields and ease in from 0 during draping.

Collisions

In most cases, a piece of cloth does not just hang there in 3D space, it collides with other objects in the environment. To ensure proper simulation, there are several items that have to be set up and working together:

- The *Cloth* object must be told to participate in collisions.
- Optionally (but recommended) tell the cloth to collide with itself.
- Other objects must be visible to the *Cloth* object *via* shared layers.
- The other objects must be mesh objects.
- The other objects may move or be themselves deformed by other objects (like an armature or shape key).
- The other mesh objects must be told to deflect the cloth object.
- The blend-file must be saved in a directory so that simulation results can be saved.
- You then *Bake* the simulation. The simulator computes the shape of the cloth for a frame range.
- You can then edit the simulation results, or make adjustments to the cloth mesh, at specific frames.
- You can make adjustments to the environment or deforming objects, and then re-run the cloth simulation from the current frame forward.

Collision Settings

Now you must tell the *Cloth* object that you want it to participate in collisions. For the cloth object, locate the *Cloth Collision* panel,

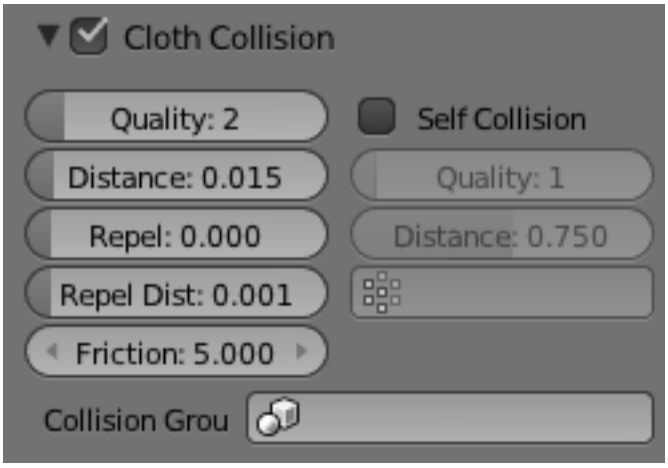


Fig. 2.1313: Cloth Collisions panel.

shown to the right:

Quality A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the cloth.

Distance As another object gets this close to it (in Blender Units), the simulation will start to push the cloth out of the way.

Repel Repulsion force to apply when cloth is close to colliding.

Repel Distance Maximum distance to apply repulsion force. Must be greater than minimum distance.

Friction A coefficient for how slippery the cloth is when it collides with the mesh object. For example, silk has a lower coefficient of friction than cotton.

Self-collisions

Real cloth cannot permeate itself, so you normally want the cloth to self-collide.

Enable Self Collisions Click this to tell the cloth object that it should not penetrate itself. This adds to simulation compute time, but provides more realistic results. A flag, viewed from a distance does not need this enabled, but a close-up of a cape or blouse on a character should have this enabled.

Quality For higher self-collision quality just increase the *Quality* and more self collision layers can be solved. Just keep in mind that you need to have at least the same *Collision Quality* value as the *Quality* value.

Distance If you encounter problems, you could also change the *Min Distance* value for the self-collisions. The best value is 0.75; for fast things you better take 1.0. The value 0.5 is quite risky (most likely many penetrations) but also gives some speedup.

Regression blend-file: Cloth selfcollisions.

Shared Layers

Suppose you have two objects: a pair of Pants on layers 2 and 3, and your Character mesh on layers 1 and 2. You have enabled the Pants as cloth as described above. You must now make the Character "visible" to the Cloth object, so that as your character bends its leg, it will push the cloth. This principle is the same for all simulations; simulations only interact with objects on a shared layer. In this example, both objects share layer 2.

To view/change an object's layers, `RMB` click to select the object in *Object Mode* in the 3D View. `M` to bring up the "Move Layers" pop-up, which shows you all the layers that the object is on. To put the object on a single layer, `LMB` click the layer button. To put the object on multiple layers, `Shift-LMB` the layer buttons. To remove an object from a selected layer, simply `Shift-LMB` the layer button again to toggle it.

Mesh Objects Collide

If your colliding object is not a mesh object, such as a NURBS surface, or text object, you must convert it to a mesh object. To do so, select the object in object mode, and in the 3D View header, select *Object* → *Convert Object Type* `Alt-C`, and select *Mesh* from the pop-up menu.

Cloth - Object collisions

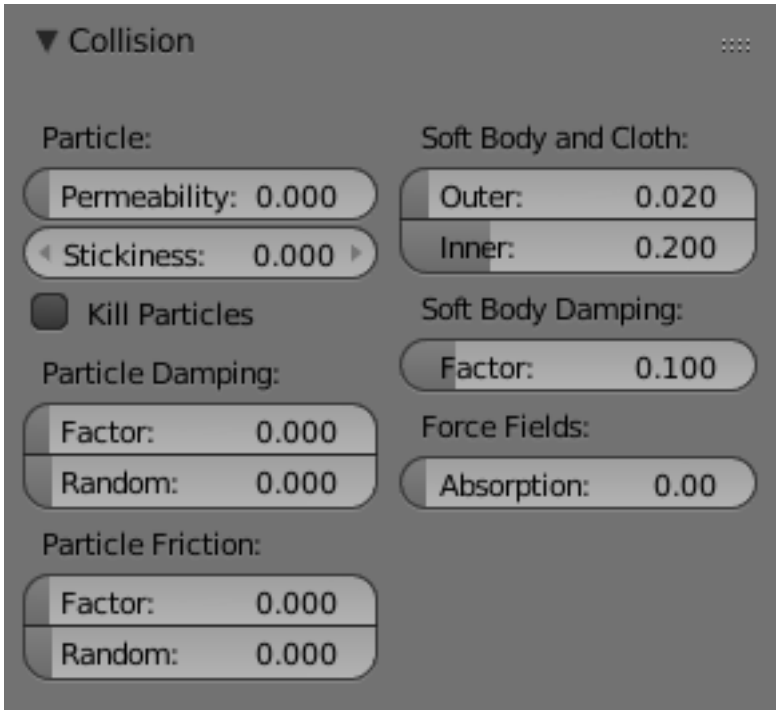


Fig. 2.1314: Collision settings.

The cloth object needs to be deflected by some other object. To deflect a cloth, the object must be enabled as an object that collides with the cloth object. To enable Cloth - Object collisions, you have to enable deflections on the collision object (not on the cloth object).

In the Properties editor, *Object* tab and *Physics* tab, locate the *Collision* panel shown to the right. It is also important to note that this collision panel is used to tell all simulations that this object is to participate in colliding/deflecting other objects on a shared layer (particles, soft bodies, and cloth).

Warning: There are three different *Collision* panels, all found in the *Physics* tab. The first (by default), a tab beside the *Fields* panel, is the one needed here. The second panel, a tab in the *Soft Body* group, concern softbodies (and so has nothing to do with cloth). And we have already seen the last one, by default a tab beside the *Cloth* panel.

Mesh Object Modifier Stack

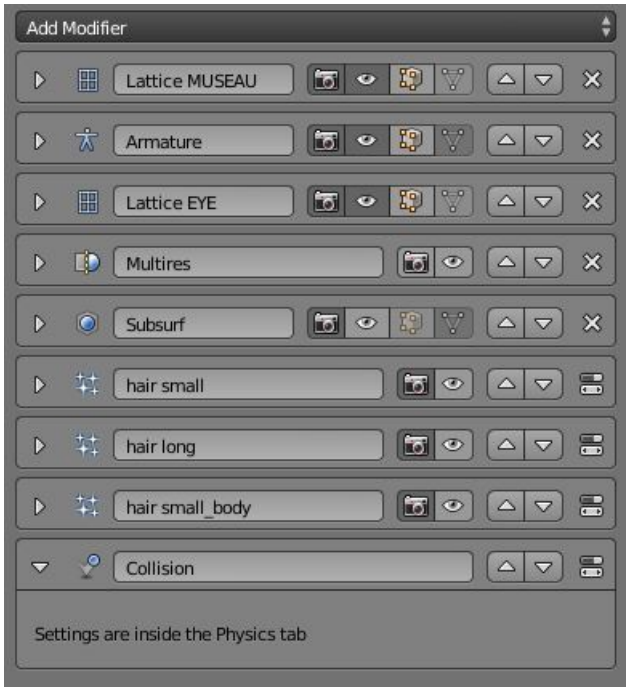


Fig. 2.1315: Collision stack.

The object's shape deforms the cloth, so the cloth simulation must know the "true" shape of that mesh object at that frame. This true shape is the basis shape as modified by shape keys or armatures. Therefore, the *Collision* modifier must be **after** any of those. The image to the right shows the *Modifiers* panel for the Character mesh object (not the cloth object).

Cloth Cache

Cache settings for cloth are the same as with other dynamic systems. See *Particle Cache* for details.

Bake Collision

After you have set up the deflection mesh for the frame range you intend to run the simulation (including animating that mesh *via* armatures), you can now tell the cloth simulation to compute (and avoid)

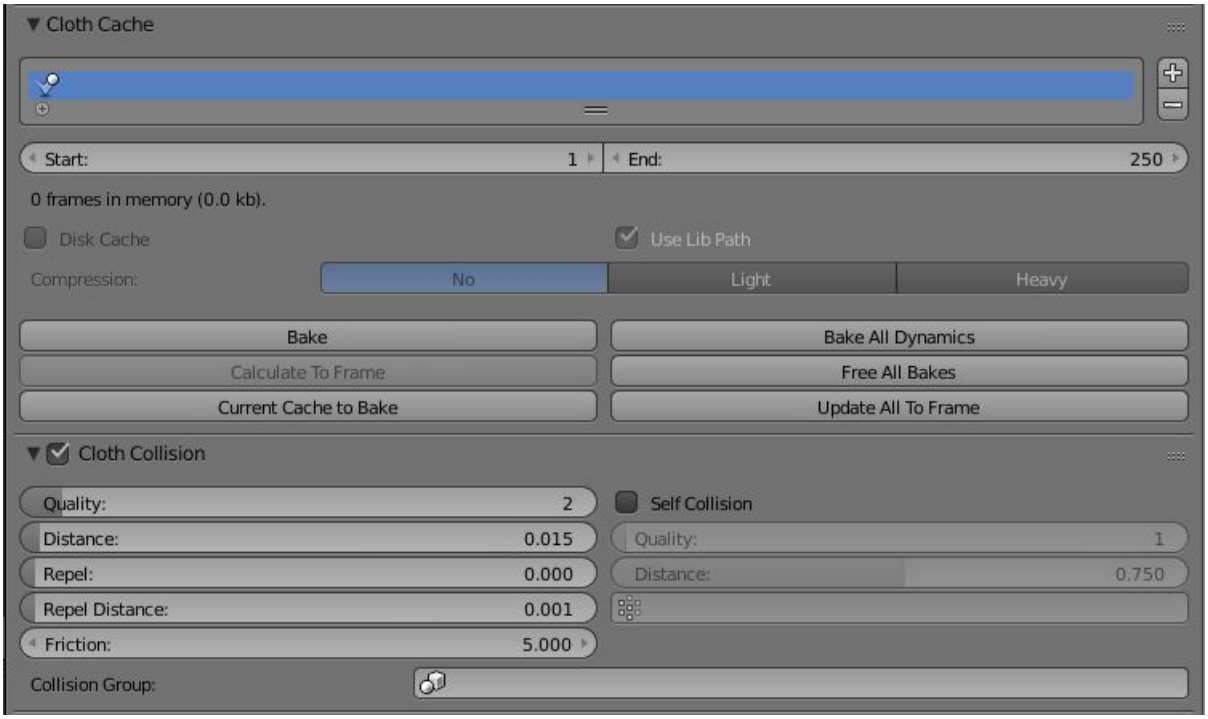


Fig. 2.1316: After Baking.

collisions. Select the cloth object and in the *Object* tab, *Physics* tab, set the *Start* and *End* settings for the simulation frames you wish to compute, and click the *Bake* button.

You cannot change *Start* or *End* without clearing the bake simulation. When the simulation has finished, you will notice you have the option to free the bake, edit the bake and re-bake:

There are a few things you will probably notice right away. First, it will bake significantly slower than before, and it will probably clip through the box pretty badly as in the picture on the right.

Editing the cached simulation

The cache contains the shape of the mesh at each frame. You can edit the cached simulation, after you have baked the simulation and pressed the *Bake Editing* button. Just go to the frame you want to fix and `Tab` into *Edit Mode*. There you can move your vertices using all of Blender's mesh shaping tools. When you exit, the shape of the mesh will be recorded for that frame of the animation. If you want Blender to resume the simulation using the new shape going forward, `LMB` click *Rebake from next Frame* and play the animation. Blender will then pick up with that shape and resume the simulation.

Edit the mesh to correct minor tears and places where the colliding object has punctured the cloth.

If you add, delete, extrude, or remove vertices in the mesh, Blender will take the new mesh as the starting shape of the mesh back to the *first frame* of the animation, replacing the original shape you started with, up to the frame you were on when you edited the mesh. Therefore, if you change the content of a mesh, when you `Tab` out of *Edit*

Mode, you should unprotect and clear the cache so that Blender will make a consistent simulation.

Troubleshooting

If you encounter some problems with collision detection, there are two ways to fix them:

- The fastest solution is to increase the *Min Distance* setting under the *Cloth Collision* panel. This will be the fastest way to fix the clipping; however, it will be less accurate and will not look as good. Using this method tends to make it look like the cloth is resting on air, and gives it a very rounded look.
- A second method is to increase the *Quality* (in the first *Cloth* panel). This results in smaller steps for the simulator and therefore to a higher probability that fast-moving collisions get caught. You can also increase the *Collision Quality* to perform more iterations to get collisions solved.
- If none of the methods help, you can easily edit the cached/baked result in *Edit Mode* afterwards.
- The Cloth is torn by the deforming mesh – he "Hulks Out": Increase its structural stiffness (*Structure Stiffness* setting, *Cloth* panel), very high, like 1000.

Note: Subdivision Surface Modifier

A bake/cache is done for every subdivision level so please use **the equal** subdivision level for render and preview.

Examples

To start with cloth, the first thing you need, of course, is some fabric. So, let us delete the default cube and add a plane. In order to get some good floppy and flexible fabric, you will need to subdivide it several times, about eight is a good number. So `Tab` into *Edit Mode*, and press `W` → *Subdivide multi*, and set it to 8.

Now, we will make this cloth by going to the Physics tab. Scroll down until you see the *Cloth* panel, and press the *Cloth* button. Now, a lot of settings will appear, most of which we will ignore for now.

That is all you need to do to set your cloth up for animating, but if you press `Alt-A`, your lovely fabric will just drop very un-spectacularly. That is what we will cover in the next two sections about pinning and colliding.

Using Simulation to Shape/Sculpt a Mesh

You can *Apply* the *Cloth* modifier at any point to freeze the mesh in position at that frame. You can then re-enable the cloth, setting the start and end frames from which to run the simulation forward.

Another example of aging is a flag. Define the flag as a simple grid shape and pin the edge against the flagpole. Simulate for 50 frames or so, and the flag will drop to its "rest" position. Apply the *Cloth*

modifier. If you want the flag to flap or otherwise move in the scene, re-enable it for the frame range when it is in camera view.

Smoothing of Cloth

Now, if you followed this from the previous section, your cloth is probably looking a little blocky. In order to make it look nice and smooth like the picture you need to apply a *Smooth* and/or *Subdivision Surface* modifier in the *Modifiers* tab. Then, in the same editor, find the *Links and Materials* panel (the same one you used for vertex groups) and press *Set Smooth*.

Now, if you press `Alt-A`, things are starting to look pretty nice, do not you think?

Cloth on armature

Cloth deformed by armature and also respecting an additional collision object: Regression blend-file.

Cloth with animated vertex groups

Cloth with animated pinned vertices: Regression blend-file. UNSUPPORTED: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will not work (e.g. from goal = 0 to goal = 0.5).

Cloth with Dynamic Paint

Cloth with Dynamic Paint using animated vertex groups: Regression blend-file. UNSUPPORTED: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will not work (e.g. from goal = 0 to goal = 0.5) because the necessary "goal springs" cannot be generated on the fly.

Using Cloth for Softbodies

Cloth can also be used to simulate softbodies. It is for sure not its main purpose but it works nonetheless. The example image uses standard *Rubber* material, no fancy settings, just `Alt-A`.

Blend file for the example image: Using Cloth for softbodies.

Cloth with Wind

Regression blend-file for Cloth with wind and self collisions (also the blend for the image above): Cloth flag with wind and selfcollisions.

Fig. 2.1317: Using cloth for softbodies.

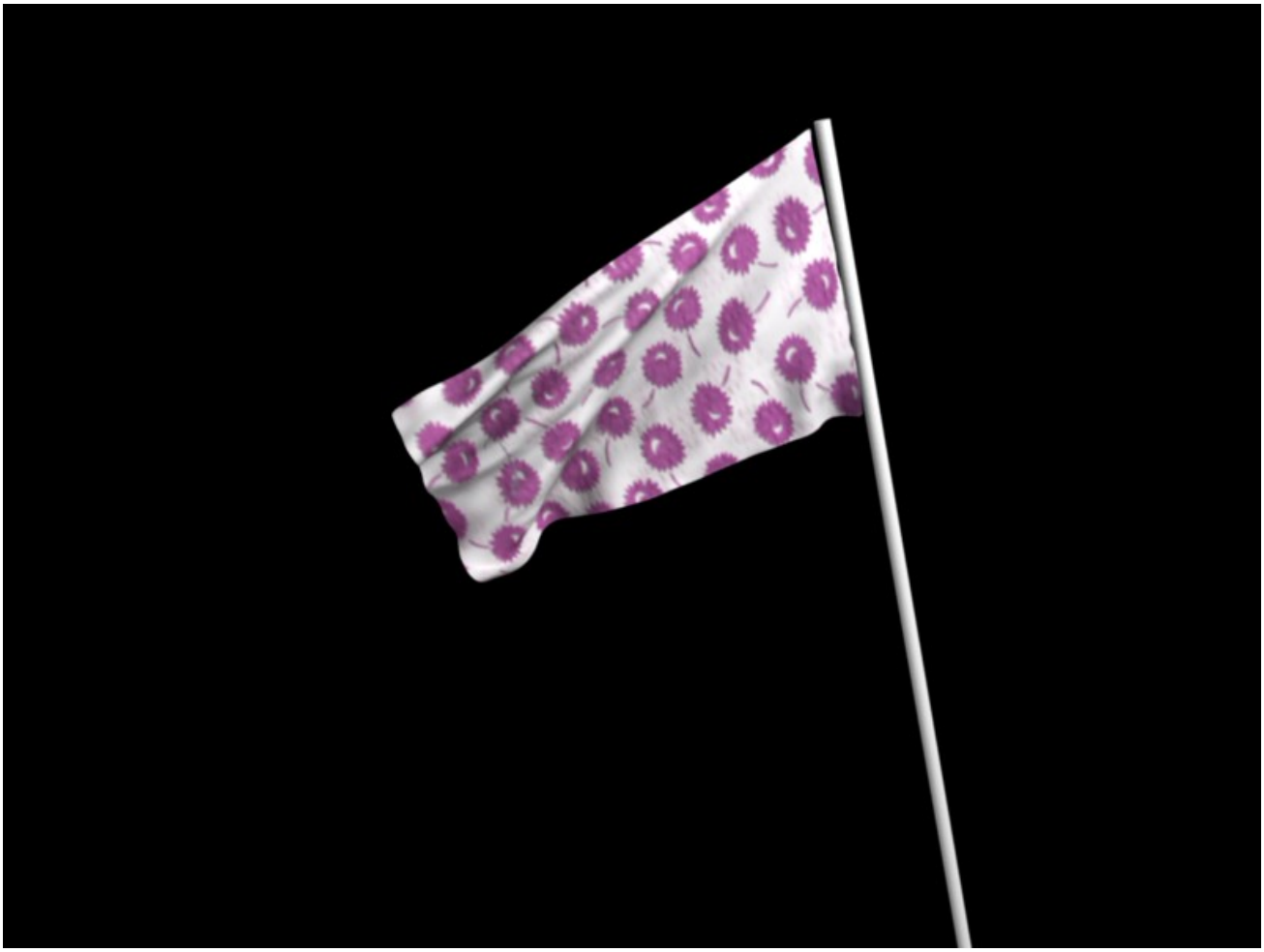


Fig. 2.1318: Flag with wind applied.

Dynamic Paint

Introduction

Dynamic paint is a new modifier and physics system that can turn objects into paint canvases and brushes, creating vertex colors, image sequences or displacement. This makes many effects possible that were previously difficult to achieve, for example footsteps in the snow, raindrops that make the ground wet, paint that sticks to walls, or objects that gradually freeze.

This guide explains the very basics of Dynamic Paint user interface and general features.

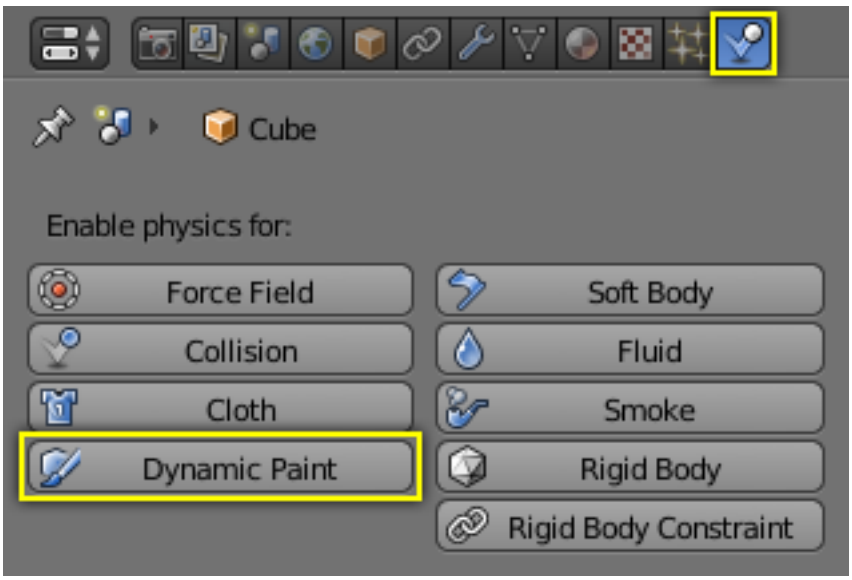


Fig. 2.1319: How to activate the Dynamic Paint.

Activating the modifier

Dynamic Paint can be activated from the "Physics" tab of the "Properties" editor.

Types

Modifier itself has two different types:

Canvas Makes object receive paint from Dynamic Paint brushes.

Brush Makes object apply paint on the canvas.

Note: You can also enable brush and canvas simultaneously. In that case same object's "brush" does not influence its "canvas", but can still interact with other objects in the scene.

See also:

- A step-by step introduction
- A detailed guide that covers every setting with images and examples (Currently not up-to-date)

Dynamic Paint Brush

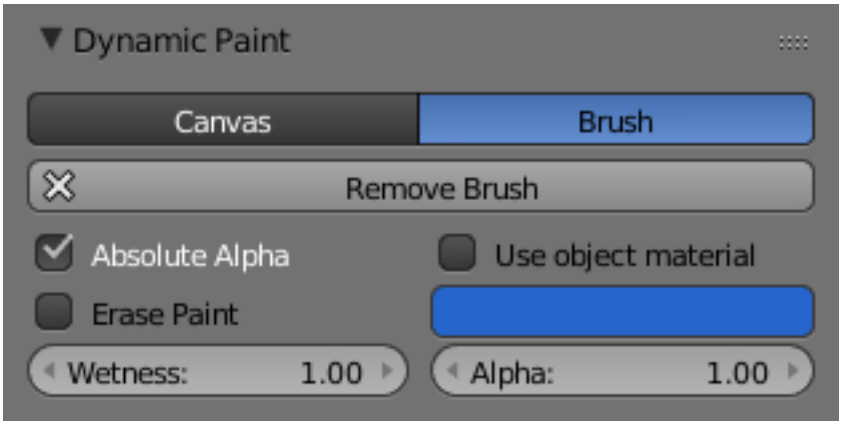


Fig. 2.1320: Brush main panel.

From the first brush panel you can define how brush affects canvas color surfaces.

Absolute Alpha This setting limits brush alpha influence. Without it, brush is "added" on surface over and over again each frame, increasing alpha and therefore influence of brush on canvas. In many cases however, it is preferred to not increase brush alpha if it already is on brushes level.

Erase Paint Makes brush dissolve exiting paint instead of adding it.

Wetness Defines how "wet" new paint is. Wetness is visible on "Paint" surface "wetmap". Speed of "Drip" and "Spread" effects also depends on how wet the paint is.

Use object material When enabled, you can define a material to be used as brush color. This includes material's base color and all textures linked to it, eventually matching the rendered diffuse color. This setting is only available when using "Blender Internal" renderer at the moment.

Otherwise you can define a color for the brush from the color box below.

Alpha Defines brush alpha or visibility. Final wetness is also affected by alpha.

Source Panel

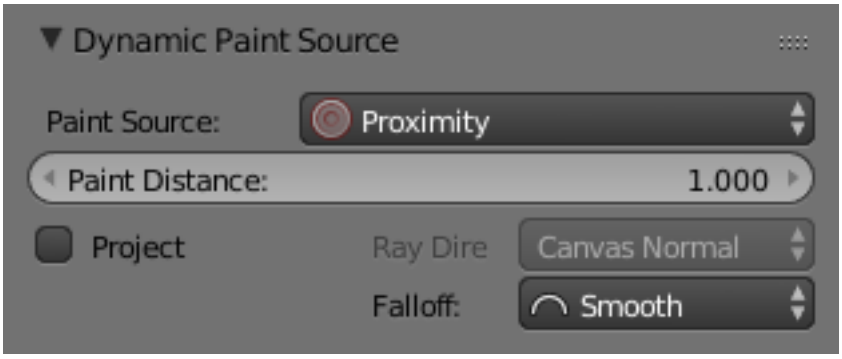


Fig. 2.1321: Brush source panel.

Brush "Source" setting lets you define how brush influence/intersection is defined.

There are currently five brush behavior types to choose from, each having individual settings for further tweaking:

Mesh Volume This the default option. Brush affects all surface point inside the mesh volume.

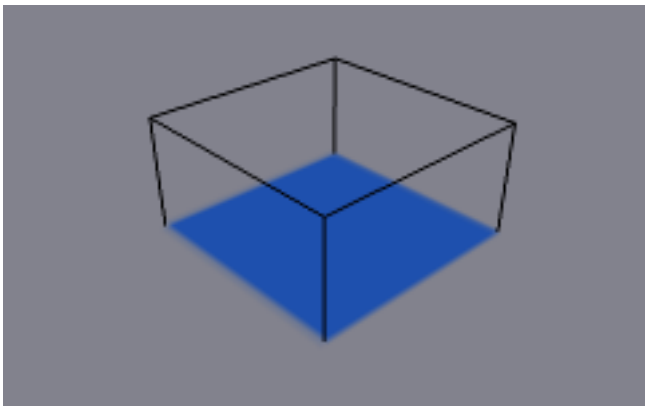


Fig. 2.1322: Source: Mesh Volume.

Proximity Only uses defined distance to the closest point on brush mesh surface. Note that inside of the volume is not necessarily affected because it is not close to the surface.

Proximity falloff type can be "Smooth", "Sharp" or tweaked with a color ramp.

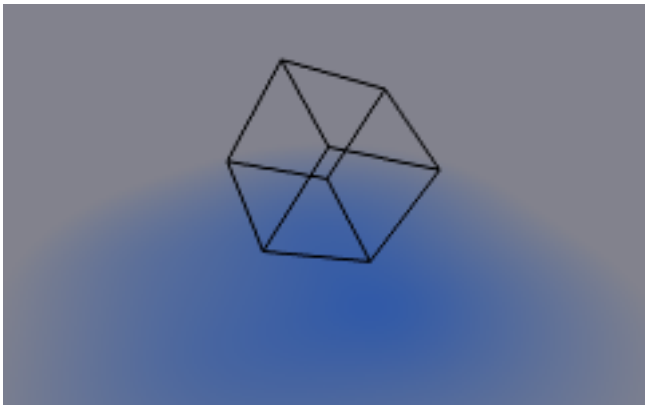


Fig. 2.1323: Source: Proximity. Brush affects all canvas pixels around it.

Project Projects brush to the canvas from a defined direction. Basically this can be considered as "direction aligned" proximity.

Mesh Volume + Proximity Same as volume type, but also has influence over defined distance. Same falloff types as for "Proximity" type are available.

Inner Proximity Applies proximity inside the mesh volume.

Negate Volume Negates brush alpha within mesh volume.

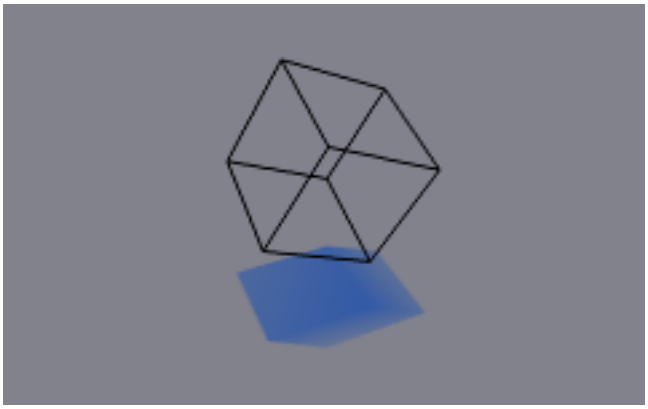


Fig. 2.1324: "Project" setting enabled. See how brush only affects canvas in normal direction.

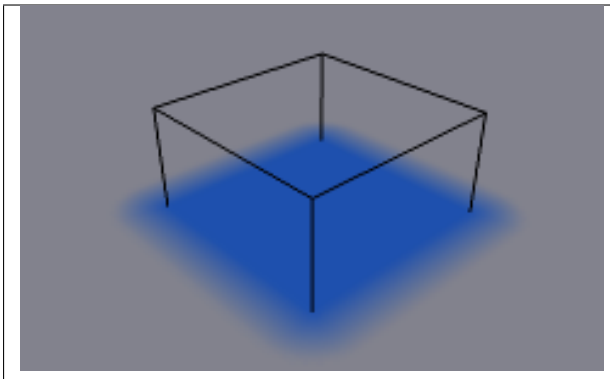


Fig. 2.1325: "Volume + Proximity" brush with no additional settings.

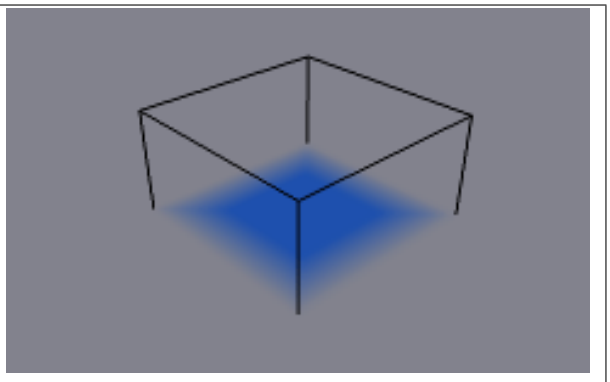


Fig. 2.1326: Inner Proximity. Proximity falloff is now visible inside the volume.

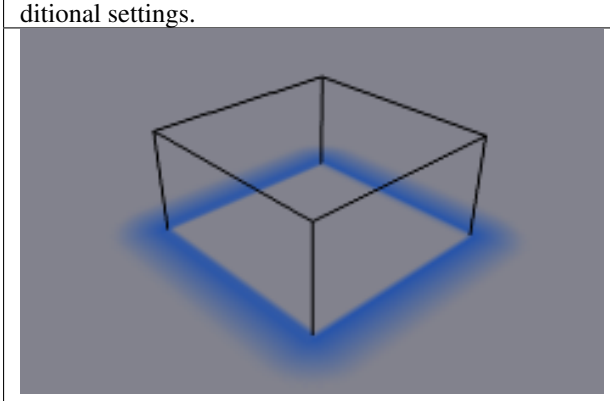


Fig. 2.1327: Negate Volume. Inner side of the volume has become completely transparent.

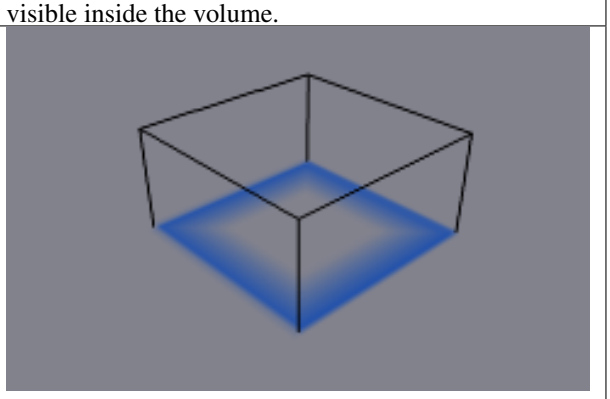


Fig. 2.1328: Inner Proximity and Negate Volume enabled together.

Object Center Instead of calculating proximity to the brush object mesh, which can be quite slow in some cases, only distance to only center is calculated. This is much faster and often good enough.

Particle System Brush influence is defined by particles from a selected particle system.

Velocity Panel

This panel shows brush options that are based on object velocity.

On top you have a color ramp and several related settings. Basically the color ramp represents brush velocity values: left side being zero

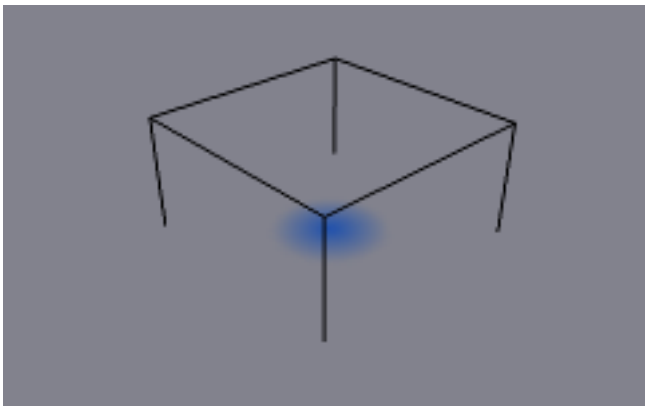


Fig. 2.1329: Source: Object Center.

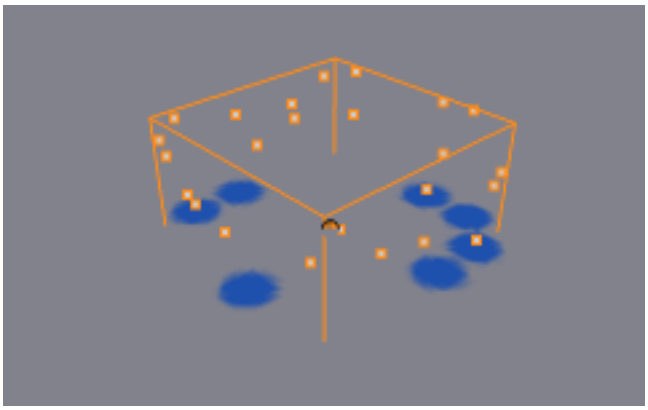


Fig. 2.1330: Source: Particle System.

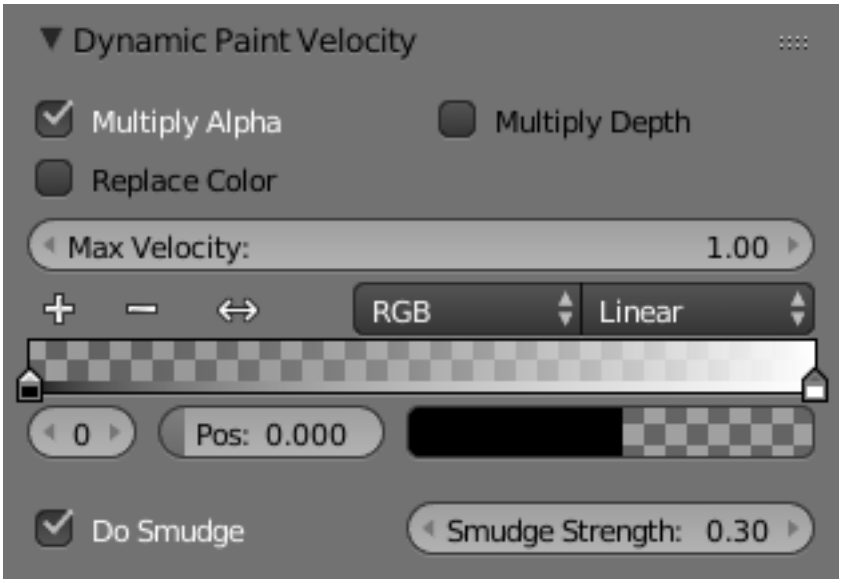


Fig. 2.1331: Velocity panel.

velocity and right side being the "Max velocity". Speed is measured in "Blender units per frame".

Checkboxes above can be used to define color ramp influence.

Multiply Alpha Uses color ramp's alpha value depending on current velocity and multiplies brush alpha with it.

Replace Color Replaces the brush color with the values from the *Color Ramp Widget*.

Multiply Depth Multiplies brushes "depth intersection" effect. Basically you can adjust displace and wave strength depending on brush speed.

Smudge settings Enabling Smudge makes the brush "smudge" (or "smear") existing colors on the surface as it moves. The strength of this effect can be defined from the "Smudge Strength" property.

Even when smudge is enabled brush still does its normal paint effect. If you want a purely smudging brush use zero alpha. It is also possible to have "Erase" option enabled together with smudge.

Waves Panel

Fig. 2.1332: Brush Waves panel.

This panel is used to adjust brush influence to "Wave" surfaces.

Wave Type Select what effect the brush has on the wave simulation.

Depth Change This option makes brush create waves when the intersection depth with the surface is *changed* on that point. If the brush remains still it will not have influence.

Using a negative "Factor" with this type can create a nice looking "wake" for moving objects like ships.

Obstacle Constantly affects surface whenever intersecting. Waves are also reflected off this brush type. However, due the nature of wave simulation algorithm this type creates an unnatural "dent" in the surface if brush remains still.

Force Directly affects the velocity of wave motion. Therefore the effect is not one to one with brush intersection depth, yet the force strength depends on it.

Reflect Only This type has no visible effect on the surface alone but reflects waves that are already on the surface.

Factor Adjusts how strongly brush "depth" affects the simulation. You can also use negative values to make brush pull water up instead of down.

Clamp Waves In some cases the brush goes very deep inside the surface messing whole simulation up. You can use this setting to "limit" influence to only certain depth.

Dynamic Paint Canvas

The first panel of canvas contains the list of Dynamic Paint surfaces. These surfaces are basically layers of paint, that work independently from each other. You can define individual settings for them and bake them separately.

If surface type/format allows previewing results in 3D View, an eye icon is visible to toggle preview.

The checkbox toggles whether surface is active at all. If not selected, no calculations or previews are done.

You can also give each surface a unique name to easily identify them.

Below you can set surface type and adjust quality and timing settings.

Each surface has a certain format and type. Format determines how data is stored and outputted. Currently there are two formats available:

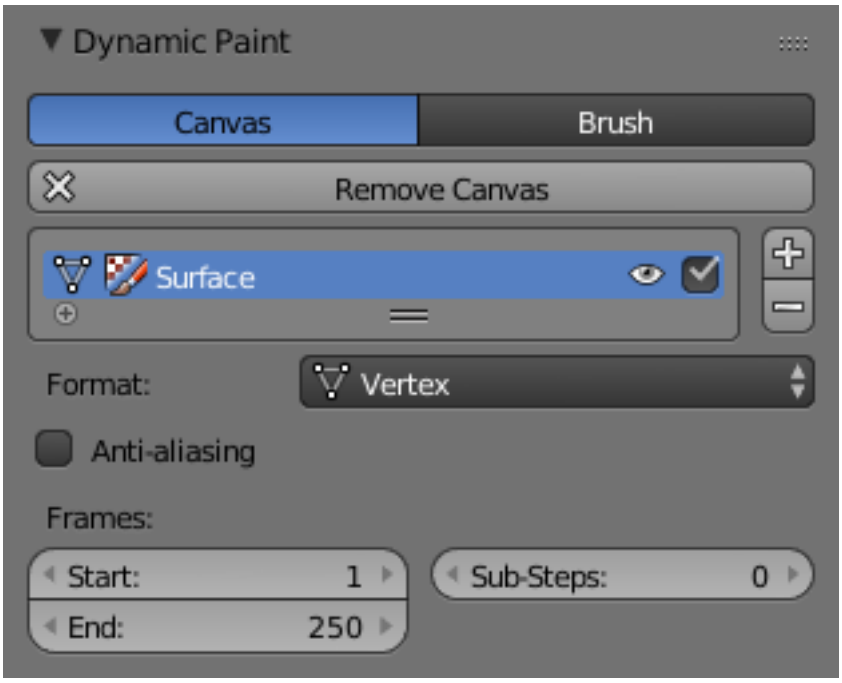


Fig. 2.1333: Canvas main panel.

- Image Sequences. Dynamic Paint generates UV wrapped image files of defined resolution as output.
- Vertex. Dynamic Paint operates directly on mesh vertex data. Results are stored by point cache and can be displayed in viewports. However, using vertex level also requires a highly subdivided mesh to work.

From quality settings you can adjust image resolution (for image sequences) and anti-aliasing.

Then you can define surface processing start and end frame, and number of used sub-steps. Sub-steps are extra samples between frames, usually required when there is a very fast brush.

Advanced Panel

From "Advanced" panel you can adjust surface type and related settings.

Each surface has a "type" that defines what surface is used for. Available types are:

- Paint
- Displace
- Waves
- Weight

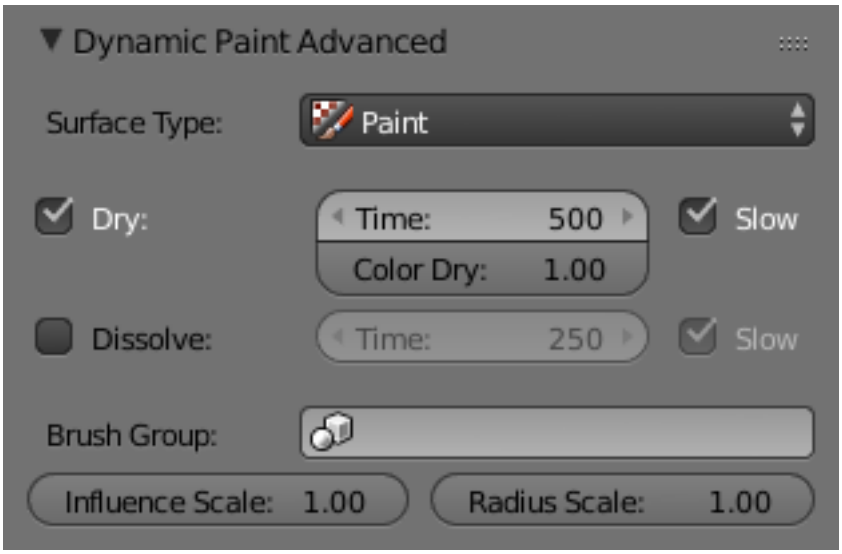


Fig. 2.1334: Canvas advanced panel.

Common Options

For each surface type there are special settings to adjust. Most types have the

settings *Dissolve* and *Brush* :

Dissolve used to make the surface smoothly return to its original state during a defined time period.

Brush Group used to define a specific object group to pick brush objects from.

Paint

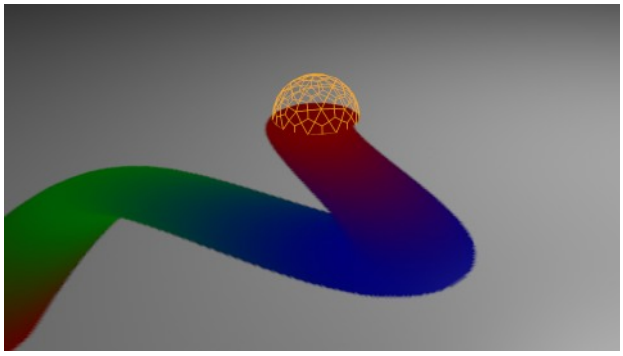


Fig. 2.1335: Paint Surface.

"Paint" is the basic surface type that outputs color and wetness values. In case of vertex surfaces results are outputted as vertex colors.

Wetmap is a black-and-white output that visualizes paint wetness. White being maximum wetness, black being completely dry. It is usually used as mask for rendering. Some "paint effects" affect wet paint only.

Displace

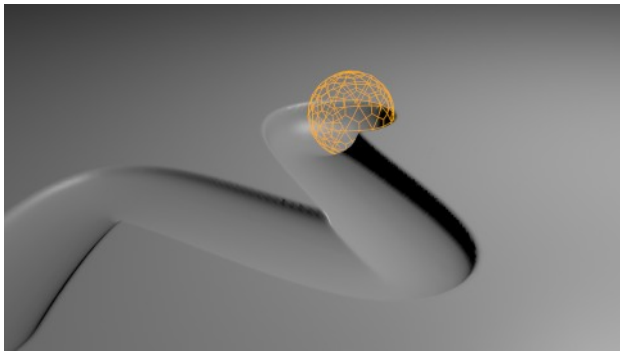


Fig. 2.1336: Displace Surface.

This type of surface outputs intersection depth from brush objects.

Tip: If the displace output seems too rough it usually helps to add a "Smooth" modifier after Dynamic Paint in the modifier stack.

Waves

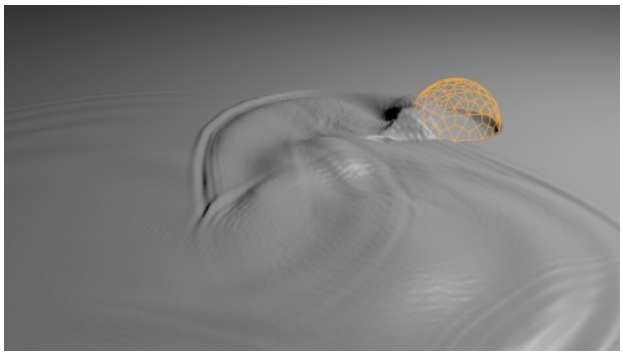


Fig. 2.1337: Waves Surface.

This surface type produces simulated wave motion. Like displace, wave surface also uses brush intersection depth to define brush strength.

You can use following settings to adjust the motion:

Open Borders Allows waves to pass through mesh "edges" instead of reflecting from them.

Timescale Directly adjusts simulation speed without affecting simulation outcome. Lower values make simulation go slower and otherwise.

Speed Affects how fast waves travel on the surface. This setting is also corresponds to the size of the simulation. Half the speed equals surface double as large.

Damping Reduces the wave strength over time. Basically adjusts how fast wave disappears.

Spring Adjusts the force that pulls water back to "zero level".

Tip: In some cases the wave motion gets very unstable around brush. It usually helps to reduce wave speed, brush "wave factor" or even the resolution of mesh/surface.

Weight

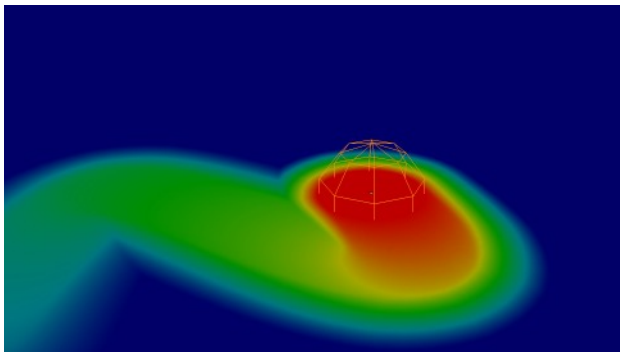


Fig. 2.1338: Weight Surface.

This is a special surface type only available for vertex format. It outputs vertex weight groups that can be used by other Blender modifiers and tools.

Tip: It is usually preferred to use "proximity" based brushes for weight surfaces to allow smooth falloff between weight values.

Output Panel

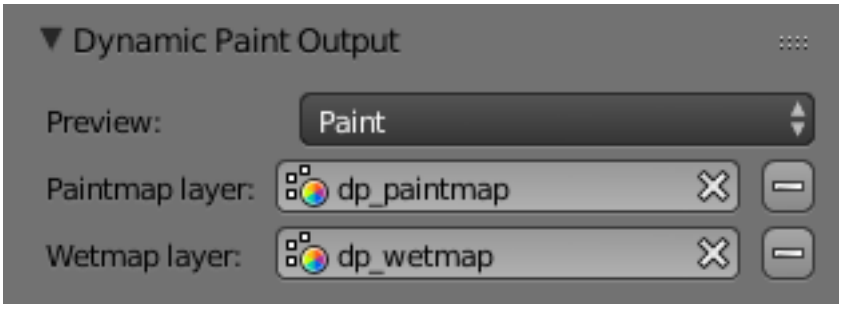


Fig. 2.1339: Canvas output panel.

From "Output" panel you can adjust how surface outputs its results.

For "Vertex" format surfaces, you can select a mesh data layer (color / weight depending on surface type) to generate results to. You can use the "+"/"-" icons to add/remove a data layers of given name. If layer with given name is not found, it is shown as red.

For "Image Sequence" surfaces, you can define used "UV Layer" and output file saving directory, filenames and image format.

Effects Panel

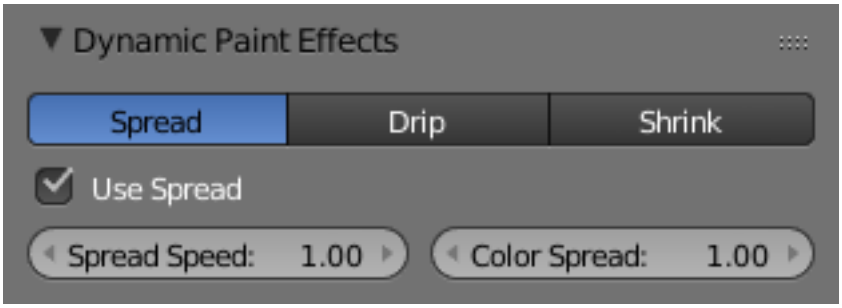


Fig. 2.1340: Canvas effects panel.

This is a special feature for "Paint" type surface. It generates animated movement on canvas surface.

Effects

Spread Paint slowly spreads to surrounding points eventually filling all connected areas.

Drip Paint moves in specific direction specified by Blender force fields, gravity and velocity with user defined influences.

Shrink Painted area slowly shrinks until disappears completely.

For spread and drip effects, only "wet paint" is affected, so as the paint dries, movement becomes slower until it stops.

Cache Panel

This panel is currently only visible for "vertex" format surfaces. You can use it to adjust and bake point cache.

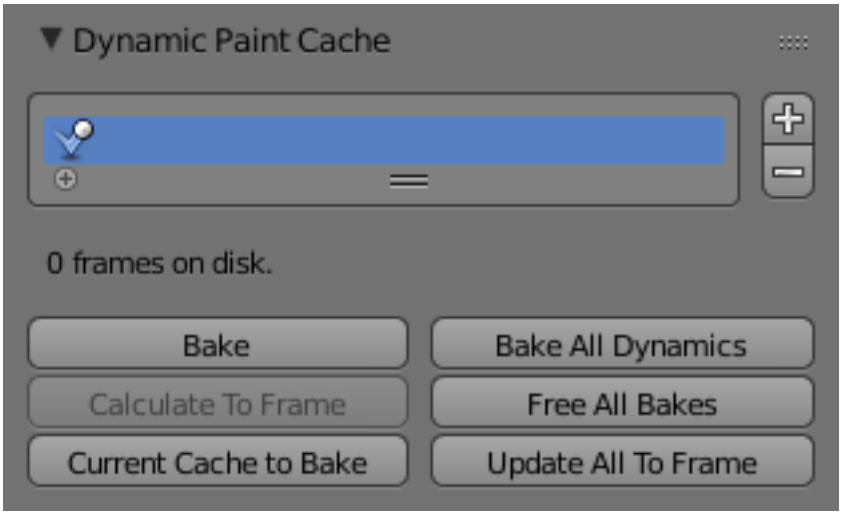


Fig. 2.1341: Canvas cache panel.

Soft Body

Introduction

Fig. 2.1342: A softbody cloth uncovering a text.
Animation video and Blend file.

A Soft Body in general, is a simulation of a soft or rigid deformable object. In Blender, this system is best for simple cloth objects and closed meshes. There is dedicated *Cloth Simulation* physics that use a different solver, and is better for cloth.

This simulation is done by applying forces to the vertices or controlpoints of the object. There are exterior forces like gravity or forcefields and interior forces that hold the vertices together. This way you can simulate the shapes that an object would take on in reality if it had volume, was filled with something, and was acted on by real forces.

Soft Bodies can interact with other objects trough *Collision*. They can interact with themselves trough *Self Collision*.

The result of the Soft Body simulation can be converted to a static object. You can also *bake edit* the simulation, i.e. edit intermediate results and run the simulation from there.

Typical scenarios for using Soft Bodies

Fig. 2.1343: A wind cone. The cone is a Soft Body, as the suspension.
Animation - Blend file

Soft Bodies are well suited for:

- Elastic objects with or without collision.
- Flags, fabric reacting to forces.
- Certain modeling tasks, like a cushion or a table cloth over an object.
- Blender has another simulation system for clothing (see *Clothes*). But you can sometimes use Soft Bodies for certain parts of clothing, like wide sleeves.
- Hair (as long as you minimize collision).
- Animation of swinging ropes, chains and the like.

The following videos may give you some more ideas:

- https://www.youtube.com/watch?v=qdusMZlBbQ4
- https://www.youtube.com/watch?v=3du8ksOm9Fo&hl

Creating a Soft Body

Soft Body simulation works for all objects that have vertices or control points:

- Meshes.
- Curves.
- Surfaces.
- Lattices.

To activate the Soft Body simulation for an object:

- In the Properties editor, go to the *Physics* tab (it is all the way on the right, and looks like a bouncing ball).
- Activate the *Soft Body* button.

A lot of options appear. For a reference of all the settings see *this page*.

- You start a Soft Body simulation with `Alt-A`.
- You pause the simulation with `Spacebar`, continue with `Alt-A`.
- You stop the simulation with `Esc`.

Simulation Quality

The settings in the *Soft Body Solver* panel determine the accuracy of the simulation.

Min Step Minimum simulation steps per frame. Increase this value, if the Soft Body misses fast moving collision objects.

Max Step Maximum simulation steps per frame. Normally the number of simulation steps is set dynamically (with the *Error Limit*) but you have probably a good reason to change it.

Auto-Step Use Velocities for automatic step sizes.

Error Limit Rules the overall quality of the solution delivered. Default 0.1. The most critical setting that says how precise the solver should check for collisions. Start with a value that is 1/2 the average edge length. If there are visible errors, jitter, or over-exaggerated responses, decrease the value. The solver keeps track of how "bad" it is doing and the *Error Limit* causes the solver to do some "adaptive step sizing".

Fuzzy Simulation is faster, but less accurate.

Choke Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Diagnostics

Print Performance to Console Prints on the console how the solver is doing.

Estimate Matrix Estimate matrix. Split to `COM`, `ROT`, `SCALE`

Cache and Bake

Soft Bodies and other physic simulations use a unified system for caching and baking. See *Particle Cache* for reference.

The results of the simulation are automatically cached to disk when the animation is played, so that the next time it runs, it can play again quickly by reading in the results from the disk. If you *Bake* the simulation the cache is protected and you will be asked when you are trying to change a setting that will make a recalculating necessary.

Tip: Beware of the *Start* and *End* settings

The simulation is only calculated for the frames in-between the *Start* and *End* frames (*Bake* panel), even if you do not actually bake the simulation! So if you want a simulation longer than the default setting of 250 frames you have the change the *End* frame.

Caching

- As animation is played, each physics system writes each frame to disk, between the simulation start and end frames. These files are stored in folders with prefix `blendcache`, next to the blend-file.
- The cache is cleared automatically on changes. But not on all changes, so it may be necessary to free it manually, e.g. if you change a force field. Note that for the cache to fill up, one has to start playback before or on the frame that the simulation starts.
- If you are not allowed to write to the required sub-directory caching will not take place.
- The cache can be freed per physics system with a button in the panels, or with the `Ctrl-B` shortcut key to free it for all selected objects.
- You may run into trouble if your blend-file path is very long and your operating system has a limit on the path length that is supported.

Baking

- The system is protected against changes after baking.
- The *Bake* result is cleared also with `Ctrl-B` for all selected objects or click on *Free Bake* for the current Soft Body system.
- If the mesh changes the simulation is not calculated anew.

For renderfarms, it is best to bake all the physics systems, and then copy the blendcache to the renderfarm as well.

Interaction in real time

To work with a Soft Body simulation you will find it handy to use the Timeline editor. You can change between frames and the simulation will always be shown in the actual state. The option *Continue Physics* in the *Playback* menu of the *Timeline* editor lets you interact in real time with the simulation, e.g. by moving collision objects or shake a Soft Body object.

Tip: *Continue Physics* does not work while playing the animation with `Alt-A`

Right. This works only if you start the animation with the *Play* button of the Timeline editor.

You can than select the Soft Body object while running the simulation and *Apply* the modifier in the *Modifiers* tab of the Properties editor. This makes the deformation permanent.

Tips

- Soft Bodies work especially well if the objects have an even vertex distribution. You need enough vertices for good collisions. You change the deformation (the stiffness) if you add more vertices in a certain region (see the animation of Fig. *A wind cone. The cone is a Soft Body, as the suspension.*).
- The calculation of collisions may take a long time. If something is not visible, why calculate it?
- To speed up the collision calculation it is often useful to collide with an additional, simpler, invisible, somewhat larger object (see the example to Fig. *A softbody cloth uncovering a text.*).
- Use Soft Bodies only where it makes sense. If you try to cover a body mesh with a tight piece of cloth and animate solely with Soft Body, you will have no success. Self collision of Soft Body hair may be activated, but that is a path that you have to wander alone. We will deal with *Collisions* in detail later.
- Try and use a *Lattice* or a *Curve Guide* Soft Body instead of the object itself. This may be magnitudes faster.

Settings

Soft Body This creates the soft body modifier on the selected object

Render Enable soft body during render

Display Display soft body in real time.

Soft Body

Friction The friction of the surrounding medium. Generally friction dampens a movement.

Mass Mass value for vertices. Larger mass slows down acceleration, except for gravity where the motion is constant regardless of mass. Larger mass means larger inertia, so also braking a Soft Body is more difficult.

Vertex Group Mass Use a specified vertex group for mass values.

Speed You can control the internal timing of the Softbody system with this value.

Collision Group If set, Soft Body collides with objects from the group, instead of using objects that are on the same layer.

Soft Body Cache

Note: Caching and cach options are documented *Here*.

Soft Body Goal

Use Goal Soft Body Goal acts like a pin on a chosen set of vertices; controlling how much of an effect soft body has on them. Enabling this tells Blender to use the position / animated position of a vertex in the simulation. Animating the vertices can be done in all the usual ways before the Soft Body simulation is applied. The *goal* is the desired end-position for vertices. How a softbody tries to achieve this goal can be defined using stiffness forces and damping.

Default A *Goal* value of 1.0 means no Soft Body simulation, the vertex stays at its original (animated) position. When setting *Goal* to 0.0, the object is only influenced by physical laws. By setting goal values between 0.0 and 1.0, you can blend between having the object affected only by the animation system, and having the object affected only by the soft body effect.

Minimum / Maximum When you paint the values in vertex-groups (using *Weight Paint Mode*), you can use the *G Min* and *G Max* to fine-tune (clamp) the weight values. The lowest vertex-weight (blue) will become *G Min*, the highest value (red) becomes *G Max* (please note that the blue-red color scale may be altered by User Preferences).

Stiffness The spring stiffness for *Goal*. A low value creates very weak springs (more flexible "attachment" to the goal), a high value creates a strong spring (a stiffer "attachment" to the goal).

Damping The friction for *Goal*. Higher values dampen the effect of the goal on the soft body.

Vertex Group Use a vertex group to allow per-vertex goal weights (multiplied by the *Default* goal).

Soft Body Edges

Use Edges The edges in a Mesh Object can act as springs as well, like threads in fabric.

Pull The spring stiffness for edges (how much the edges are stretched). A low value means very weak springs (a very elastic material), a high value is a strong spring (a stiffer material) that resists being pulled apart. 0.5 is latex, 0.9 is like a sweater, 0.999 is a highly-starched napkin or leather.

Push How much the Softbody resist being scrunched together, like a compression spring. Low values for fabric, high values for inflated objects and stiff material.

Damp The friction for edge springs. High values (max of 50) dampen the edge stiffness effect and calm down the cloth.

Plastic Plasticity, permanent deformation of the object.

Bending This option creates virtual connections between a vertex and the one after the next. This includes diagonal edges. Damping applies also to these connections.

Length The edges can shrink or been blown up. This value is given in percent, 0 disables this function. 100% means no change, the body keeps 100% of his size.

Stiff Quads For quad faces, the diagonal edges are used as springs. This stops quad faces to collapse completely on collisions (what they would do otherwise).

Shear Stiffness of the virtual springs for quad faces.

Aerodynamics

Simple If you turn on *Aero* the force is not confined to the vertices, but has an effect also on the edges. The angle and the relative speed between medium and edge is used to calculate the force on the edge. This force results that vertices with little connecting edges (front of a plane) fall faster than vertices with more connecting edges (middle of a plane). If all vertices have the same amount of edges in a direction they fall with equal speed. An edge moving in its own direction feels no force, and an edge moving perpendicular to its own direction feels maximum force (think of a straw moving through air). Try it with an *Factor* of 30 at first.

Lift Force Use an aerodynamic model that is closer to physical laws and looks more interesting. Disable for a more muted simulation.

Factor How much aerodynamic effect to use

Edge Checks for edges of the softbody mesh colliding.

Face Checks for any portion of the face of the softbody mesh colliding (compute intensive!). While *CFace* enabled is great, and solves lots of collision errors, there does not seem to be any dampening settings for it, so parts of the softbody object near a collision mesh tend to "jitter" as they bounce off and fall back, even when there is no motion of any meshes. Edge collision has dampening, so that can be controlled, but Deflection dampening value on a collision object does not seem to affect the face collision.

Soft Body Self Collision

Note: *Self Collision* is working only if you have activated *Use Edges*.

Self Collision When enabled, allows you to control how Blender will prevent the Soft Body from intersecting with itself. Every vertex is surrounded with an elastic virtual ball. Vertices may not penetrate the balls of other vertices. If you want a good result you may have to adjust the size of these balls. Normally it works pretty well with the default options.

Manual The *Ball Size* directly sets the ball size (in BU).

Average The average length of all edges attached to the vertex is calculated and then multiplied with the *Ball Size* setting. Works well with evenly distributed vertices.

Minimal / Maximal The ball size is as large as the smallest/largest spring length of the vertex multiplied with the *Ball Size*.

Average Min Max Size = ((Min + Max)/2) × *Ball Size*.

Size Default 0.49 BU or fraction of the length of attached edges. The edge length is computed based on the algorithm you choose. You know how when someone stands too close to you, and feel uncomfortable? We call that our "personal space", and this setting is the factor that is multiplied by the spring length. It is a spherical distance (radius) within which, if another vertex of the same mesh enters, the vertex starts to deflect in order to avoid a self-collision. Set this value to the fractional distance between vertices that you want them to have their own "space". Too high of a value will include too many vertices all the time and slow down the calculation. Too low of a level will let other vertices get too close and thus possibly intersect because there will not be enough time to slow them down.

Stiffness Default 1.0. How elastic that ball of personal space is.

Dampening Default 0.5. How the vertex reacts. A low value just slows down the vertex as it gets too close. A high value repulses it.

Collisions with other objects are set in the (other) *Collision panel*. To collide with another object they have to share at least one common layer.

Soft Body Solver

These settings determine the accurateness of the simulation.

Min Step Minimum simulation steps per frame. Increase this value, if the Soft Body misses fast moving collision objects.

Max Step Maximum simulation steps per frame. Normally the number of simulation steps is set dynamically (with the *Error Limit*) but you have probably a good reason to change it.

Auto-Step helps the Solver figure out how much work it needs to do based on how fast things are moving.

Error Limit Rules the overall quality of the solution delivered. Default 0.1. The most critical setting that says how precise the solver should check for collisions. Start with a value that is 1/2 the average edge length. If there are visible errors, jitter, or over-exaggerated responses, decrease the value. The solver keeps track of how "bad" it is doing and the *Error Limit* causes the solver to do some "adaptive step sizing".

Fuzzy Fuzziness while on collision, high values make collision handling faster but less stable.

Choke Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Print Performance to Console Prints on the console how the solver is doing.

Estimate Matrix Estimate matrix... split to COM, ROT, SCALE

Forces

Exterior

Exterior forces are applied to the vertices (and nearly exclusively to the vertices) of Soft Body objects. This is done using Newtons Laws of Physics:

- If there is no force on a vertex, it stays either unmoved or moves with constant speed in a straight line.
- The acceleration of a vertex depends on its mass and the force. The heavier the mass of a vertex the slower the acceleration. The larger the force the greater the acceleration.
- For every action there is an equal and opposite reaction.

Well, this is done only in the range of computing accurateness, there is always a little damping to avoid overshoot of the calculation.

Example

We will begin with a very simple example: the default cube.

- To judge the effect of the external forces you should at first turn off the *Goal*, so that the vertices are not retracted to their original position.
- Press `Alt-A`.

What happens? The cube moves in negative Z-direction. Each of its eight vertices is affected by a global, constant force – the gravitation. Gravitation without friction is independent from the weight of an object, so each object you would use as a Soft Body here would fall with the same acceleration. The object does not deform, because every vertex moves with the same speed in the same direction.

Settings

Soft Body Panel

Friction The friction of the surrounding medium. The larger the friction, the more viscous is the medium. Friction always appears when a vertex moves relative to its surround medium.

Mass Mass value for vertices. Larger mass slows down acceleration, except for gravity where the motion is constant regardless of mass. Larger mass means larger inertia, so also braking a Soft Body is more difficult.

Mass Vertex Group You can paint weight values for an mesh's mass, and select that vertex group here.

Speed You can control the internal timing of the Softbody system with this value. It sets the correlation between frame rate and tempo of the simulation. A free falling body should cover a distance of about five meters after one second. You can adjust the scale of your scene and your simulation with this correlation. If you render with 25 frames per second and 1 meter shall be 1 BU, you have to set *Speed* to 1.3.

Force Fields

To create other forces you have to use another object, often *Empty* objects are used for that. You can use some of the forces on Soft Body vertices as on *Particles*. Soft Bodies react only to:

- Spherical
- Wind
- Vortex

Soft bodies do react on *Harmonic* fields, but not in a useful way. So if you use a *Harmonic* field for particles move the Soft body to another layer.

See the section *Force Fields* for details. The force fields are quite strong, a *Spherical* field with a strength of -1.0 has the same effect that gravity has – approximately – a force of 10 Newton.

Aerodynamics

This special exterior force is not applied to the vertices but to the connecting edges. Technically, a force perpendicular to the edge is applied. The force scales with the projection of the relative speed on the edge (dot product). Note that the force is the same if wind is blowing or if you drag the edge through the air with the same speed. That means that an edge moving in its own direction feels no force, and an edge moving perpendicular to its own direction feels maximum force.

Simple Edges receive a drag force from surrounding media

Lift Force Edges receive a lift force when passing through surrounding media.

Factor How much aerodynamic force to use. Try a value of 30 at first.

Using a Goal

A goal is a shape that a soft body object tries to conform to.

You have to confine the movement of vertices in certain parts of the mesh, e.g. to attach a Soft Body object at other objects. This is done with the *Vertex Group* (target). The target position is the original position of the vertex, like it would result from the "normal" animation of an object including *Shape Keys*, *Hooks* and *Armatures*. The vertex tries to reach its target position with a certain, adjustable intensity.

Imagine the vertex is connected with its target through a spring Fig. *Shock absorber description.*.

Default This parameter defines how strong the influence of this spring is. A strength of 1 means, that the vertex will not move as Soft Body at all, instead keep its original position. 0 *Goal* (or no *Goal*) means, that the vertex moves only according to Soft Body simulation. If no vertex group is used/assigned, this number button is the default goal weight

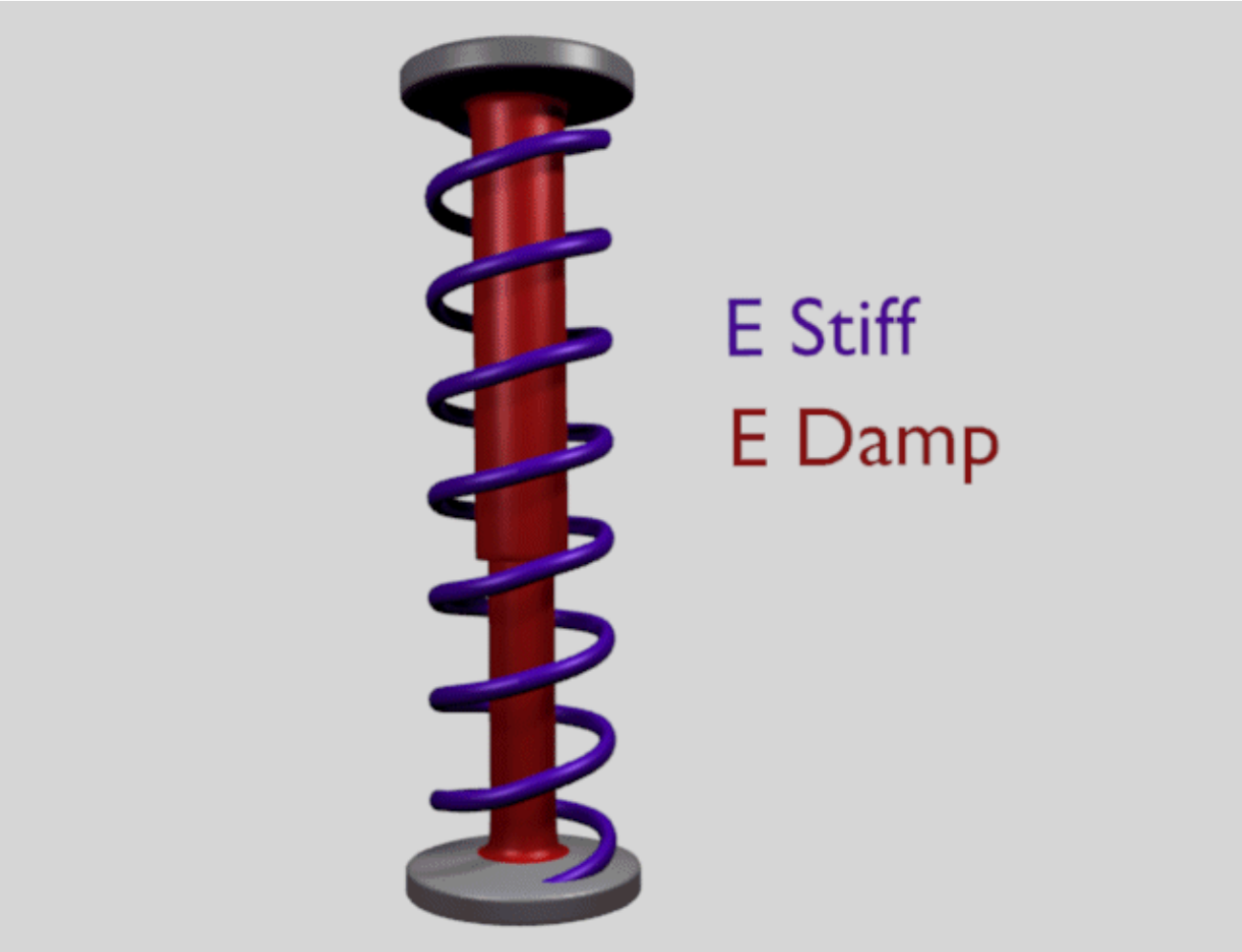


Fig. 2.1344: Shock absorber description.

for all vertices. If a vertex group is present and assigned, this button instead shows an list field, that allows you to choose the name of the goal vertex group. If you use a vertex group the weight of a vertex defines its goal.

Often *Weight Paint* is used to adjust the weight comfortably. For non-mesh objects the *Weight* parameter of their vertices/controlpoints is used instead (`W` in *Edit Mode*) or use the *Transform* panel. The weight of *Hair* particles can also be painted in *Particle Edit Mode*.

Minimum / Maximum When you paint the values in vertex-groups (using *WeightPaint Mode*), you can use the *G Min* and *G Max* to fine-tune (clamp) the weight values. The lowest vertex-weight (blue) will become *G Min*, the highest value (red) becomes *G Max* (please note that the blue-red color scale may be altered by User Preferences).

Tip: For now all is applied to single vertices

For now we have discussed vertex movement independent of each other, similar to particles. Every object without *Goal* would collapse completely if a non uniform force is applied. Now we will move to the next step, the forces that keep the structure of the object and make the Soft Body to a real Body.

Stiffness The spring stiffness for Goal. A low value creates very weak springs (more flexible "attachment" to the goal), a high value creates a strong spring (a stiffer "attachment" to the goal).

Dampimg The friction of the spring. With a high value the movement will soon come to an end (little jiggle).

Interior

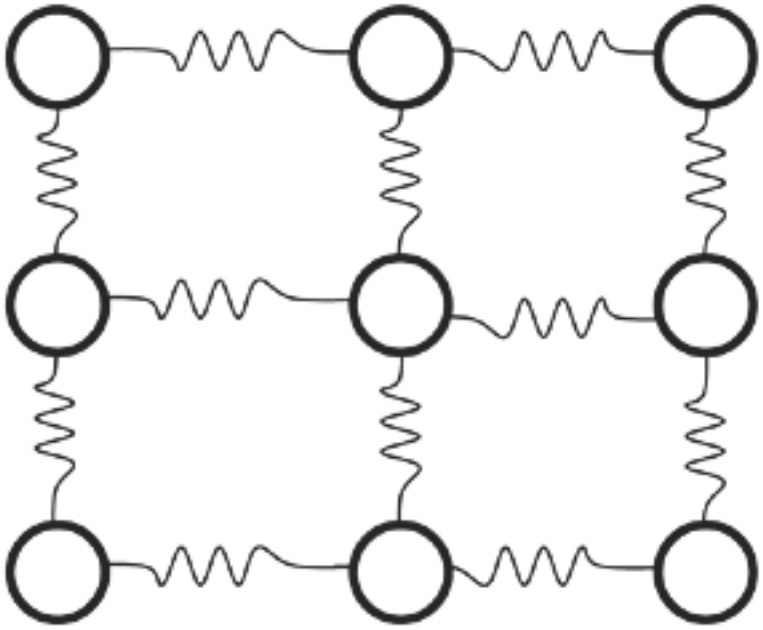


Fig. 2.1345: Vertices and forces along their connection edges.

To create a connection between the vertices of a Soft Body object there have to be forces that hold the vertices together. These forces are effective along the edges in a mesh, the connections between the vertices. The forces act like a spring. Fig. *Vertices and forces along their connection edges.* illustrates how a 3×3 grid of vertices (a mesh plane in Blender) are connected in a Soft Body simulation.

But two vertices could freely rotate if you do not create additional edges between them. Have you ever tried building a storage shelf out of four planks alone? Well, do not do it, it will not be stable. The logical method to keep a body from collapsing would be to create additional edges between the vertices. This works pretty well, but would change your mesh topology drastically.

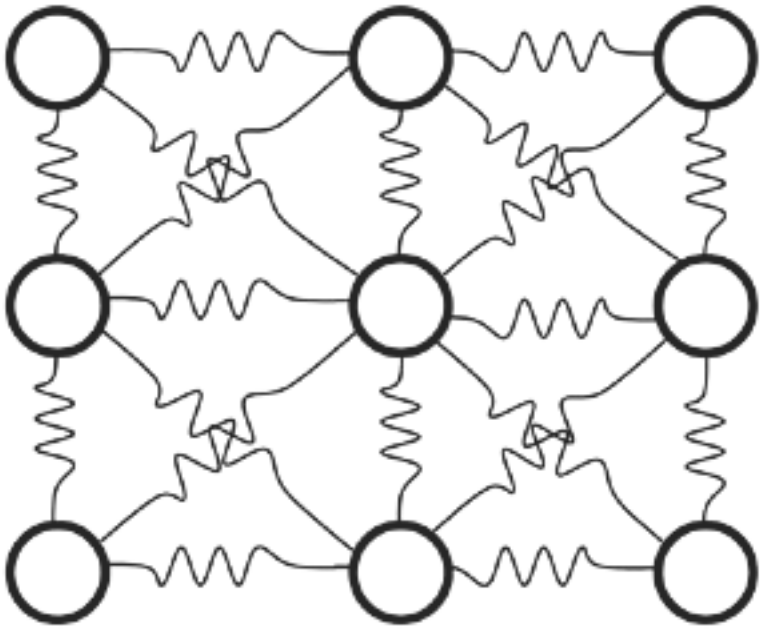


Fig. 2.1346: Additional forces with Stiff Quads enabled.

Luckily, Blender allows to define additional *virtual* connections. On one hand we can define virtual connections between the diagonal edges of a quad face (*Stiff Quads* Fig. *Additional forces with Stiff Quads enabled.*), on the other hand we can define virtual connections between a vertex and any vertices connected to its neighbors *Bending Stiffness*. In other words, the amount of bend that is allowed between a vertex and any other vertex that is separated by two edge connections.

Edges Settings

The characteristics of edges are set with the *Soft Body Edge* properties.

Use Edges Allow the edges in a Mesh Object to act like springs.

Pull The spring stiffness for edges (how much the edges are allowed to stretch). A low value means very weak springs (a very elastic material), a high value is a strong spring (a stiffer material) that resists being pulled apart. 0.5 is latex, 0.9 is like a sweater, 0.999 is a highly-starched napkin or leather. The Soft Body simulation tends to get unstable if you use a value of 0.999, so you should lower this value a bit if that happens.

Push How much the Softbody resist being scrunched together, like a compression spring. Low values for fabric, high values for inflated objects and stiff material.

Damp The friction for edge springs. High values (max of 50) dampen the *Push* / *Pull* effect and calm down the cloth.

Plastic Permanent deformation of the object after a collision. The vertices take a new position without applying the modifier.

Bending This option creates virtual connections between a vertex and the vertices connected to its neighbors. This includes diagonal edges. Damping also applies to these connections.

Length The edges can shrink or been blown up. This value is given in percent, 0 disables this function. 100% means no change, the body keeps 100% of his size.

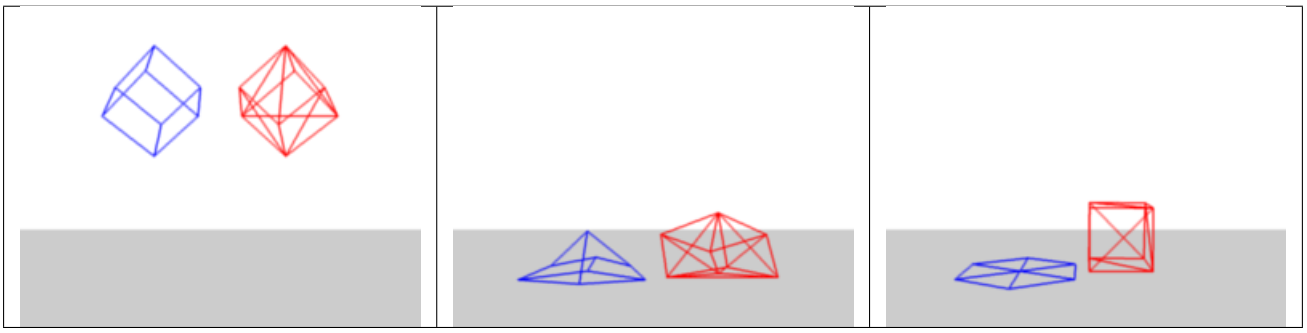
Stiff Quads For quad faces, the diagonal edges are used as springs. This stops quad faces to collapse completely on collisions (what they would do otherwise).

Shear Stiffness of the virtual springs created for quad faces.

Preventing Collapse

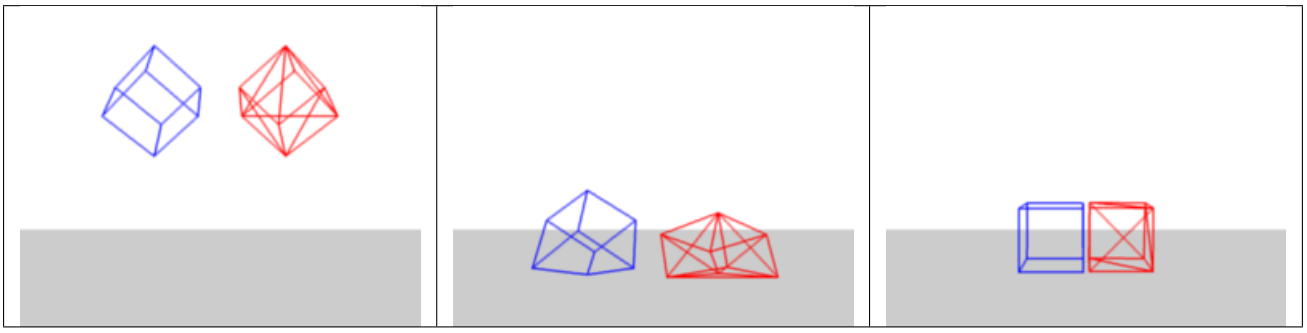
To show the effect of the different edge settings we will use two cubes (blue: only quads, red: only tris) and let them fall without any goal onto a plane (how to set up collision is shown on the page *Collisions*).

Table 2.75: Frame 401.



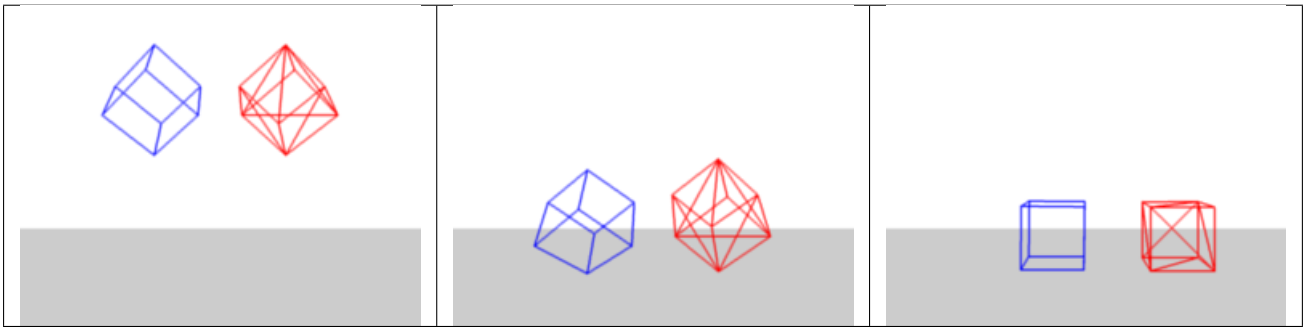
In Fig. *Without Stiff Quads.*, the default settings are used (without *Stiff Quads*). The "quad only" cube will collapse completely, the cube composed of tris keeps its shape, though it will deform temporarily because of the forces created during collision.

Table 2.76: Frame 401.



In Fig. *With Stiff Quads.*, *Stiff Quads* is activated (for both cubes). Both cubes keep their shape, there is no difference for the red cube, because it has no quads anyway.

Table 2.77: Frame 401.



The second method to stop an object from collapsing is to change its *Bending Stiffness*. This includes the diagonal edges (Damping also applies to these connections).

In Fig. *Bending Stiffness. Blend file.*, *Be* is activated with a strength setting of 1. Now both cubes are more rigid.

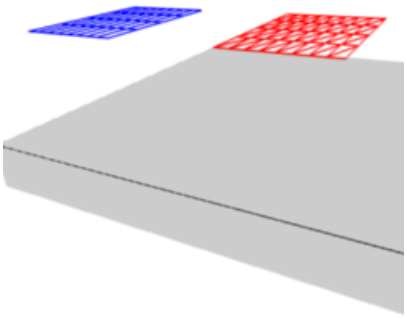


Fig. 2.1356: Two planes going to collide.

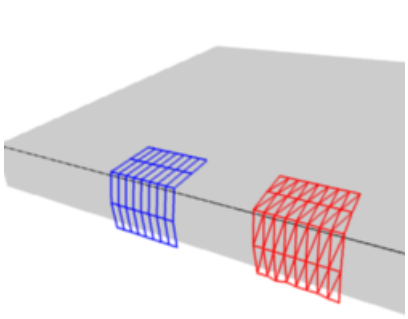


Fig. 2.1357: No bending stiffness, Frame 101.

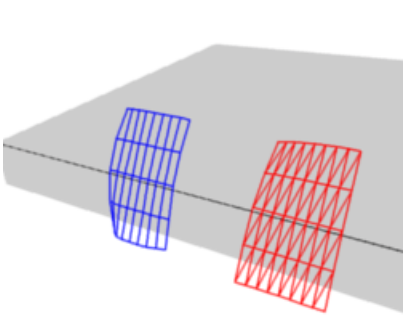


Fig. 2.1358: High bending stiffness (10), Frame 101.

Bending stiffness can also be used if you want to make a subdivided plane more plank like. Without *Be* the faces can freely rotate against each other like hinges Fig. *No bending stiffness, Frame 101.*. There would be no change in the simulation if you activated *Stiff Quads*, because the faces are not deformed at all in this example.

Bending stiffness is the strength needed for the plane to be deformed.

Collisions

There are two different collision types that you may use: collision between different objects and internal collision. We should set one thing straight from the start: the primary targets of the collision calculation are the vertices of a Soft Body. So if you have too few vertices too few collision takes place. Secondarily, you can use edges and faces to improve the collision calculation.

Collisions with other objects

For a *Soft Body* to collide with another object there are a few prerequisites:

- If *Collision Group* is set, the object must belong to the group. Otherwise, both objects have to share a layer, but the layer does not necessarily have to be visible.
- The collision object has to be a mesh object.
- You have to activate the option *Collision* in the *Collision* panel of the *Physics* tab for the collision object. The collision object may also be a Soft Body.
- If you use modifiers such as *Array* and *Mirror* you have to activate *EV.M.Stack* to ensure that collision calculation is based on the modified object. The sequence of *Modifiers* is not important.

Examples

Fig. 2.1359: A Soft Body cube colliding with a plane.

Fig. 2.1360: A Soft Body plane colliding with a cube, so no interaction at all.

Fig. 2.1361: Collision with Face activated.

A cube colliding with a plane works pretty well Fig. *A Soft Body cube colliding with a plane.*, but a plane falls right through a cube that it is supposed to collide with Fig. *A Soft Body plane colliding with a cube, so no interaction at all.*. Why is that? Because the default method of calculation only checks to see if the four vertices of the plane collides with the cube as the plane is pulled down by gravity. You can activate *CFace* to enable collision between the face of the plane and the object instead Fig. *Collision with Face activated.*, but this type of calculation takes much longer.

Let us have a closer look at the collision calculation, so you can get an idea of how we might optimize it.

Calculating Collisions

Fig. 2.1362: Visualization of the collision of a Soft Body vertex with a plane.	Fig. 2.1363: Six Soft Body vertices with different speed. Blend file.

Soft Body simulation is by default done on a per vertex basis. If the vertices of the Soft Body do not collide with the collision object there will be no interaction between the two objects.

In Fig. *Visualization of the collision of a Soft Body vertex with a plane.* you can see a vertex colliding with a plane. If a vertex penetrates the zone between *Outer* and *Inner*, it is repulsed by a force in the direction of the face normal. The position that a vertex finally ends up in is dependent on the forces that act upon it. In the example gravity and the repulsion force of the face balance out. The speed at which the vertex is pulled out of the collision zone is influenced by the *Choke* parameter Fig. *Parameters for Soft Body calculation.*.

Now lets see what happens if we make vertices heavier and let them travel at a faster speed. In Fig. *Six Soft Body vertices with different speed.* you can see vertices traveling at different speeds. The two on the far right (5 and 6) are traveling so fast that they pass right through the collision zone (this is because of the default solver precision, which we can fix later). You will notice that the fourth vertex also travels quite fast and because it is heavier it breaches the inner zone. The first three vertices collide correctly.

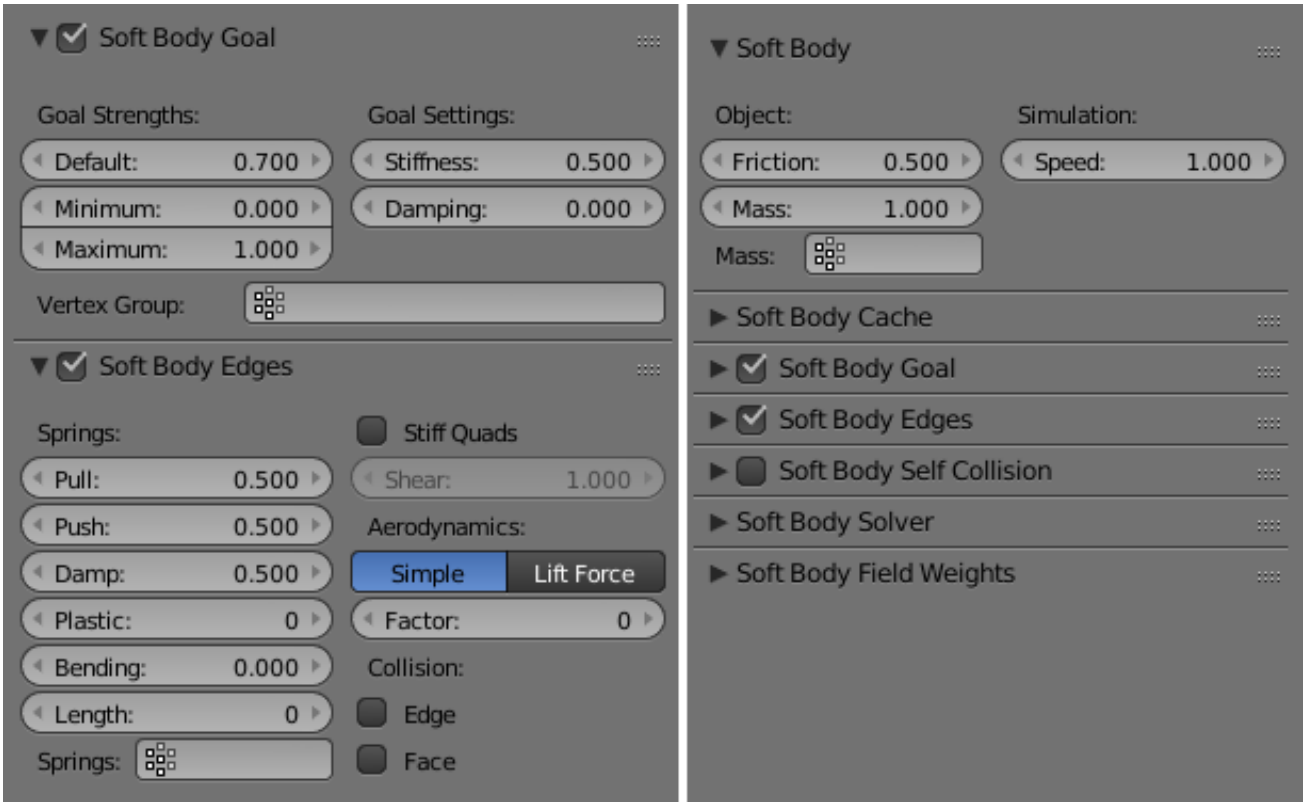


Fig. 2.1364: Also Edges and Faces can be used for the collision calculation.

You can set up your collision so that edges and even faces are included in the collision calculation Fig. *Also Edges and Faces can be used for the collision calculation.*. The collision is then calculated differently. It is checked whether the edge or face intersects with the collision object, the collision zones are not used.

Good collisions

If the collision you have set up is not behaving properly, you can try the following:

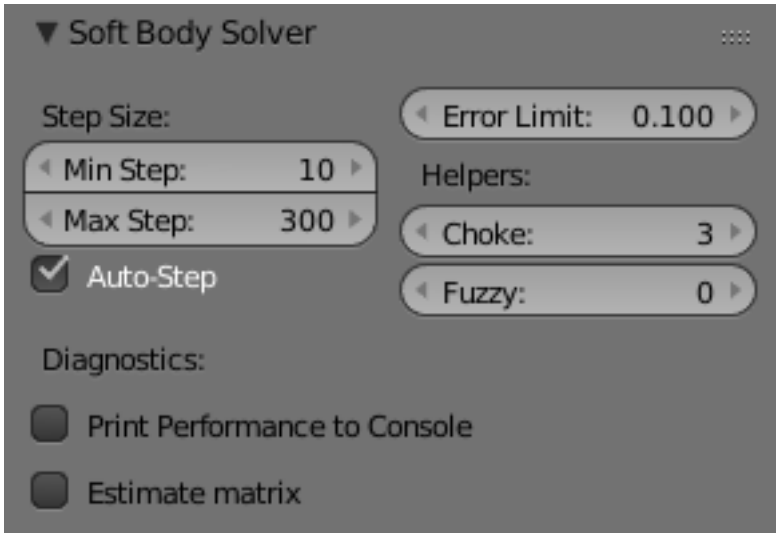


Fig. 2.1365: Parameters for Soft Body calculation.

Tip: The best way

Add *Loop Cuts* to your Soft Body object in strategic areas that you know are most likely to be involved in a collision.

- The Soft Body object must have more subdivisions than the collision object.
- Check the direction of the face normals.
- If the collision object has sharp spikes they might penetrate the Soft Body.
- The resolution of the solver must match the speed at which Soft Body vertices are traveling. Lower the parameter *Error Lim* and carefully increase *Min S*.
- *Outer* and *Inner* should be large enough, but zones of opposite faces should not overlap, or you have forces in opposite directions.
- If you use strong forces you should use large zones.
- Set *Choke* to a high enough value (all the way up if necessary) if you have difficulties with repelled vertices.
- Colliding faces are difficult to control and need long calculation times. Try not to use them.

Often it is better to create a simplified mesh to use as your collision object, however, this may be difficult if you are using an animated mesh.

Self Collision

Self Collision is working only if you have activated *Use Edges*.

When enabled, allows you to control how Blender will prevent the Soft Body from intersecting with itself. Every vertex is surrounded with an elastic virtual ball. Vertices may not penetrate the balls of other vertices. If you want a good result you may have to adjust the size of these balls. Normally it works pretty well with the default options.

Ball Size Calculation

Man ("manual") The *Ball Size* directly sets the ball size (in BU).

Av ("average") The average length of all edges attached to the vertex is calculated and then multiplied with the *Ball Size* setting. Works well with evenly distributed vertices.

Min / Max The ball size is as large as the smallest/largest spring length of the vertex multiplied with the *Ball Size*.

AvMiMax ("average min/max") Size = ((Min + Max)/2) × *Ball Size*.

Ball Size Default 0.49 BU or fraction of the length of attached edges. The edge length is computed based on the algorithm you choose. You know how when someone stands too close to you, and feel uncomfortable? We call that our "personal space", and this setting is the factor that is multiplied by the spring length. It is a spherical distance (radius) within which, if another vertex of the same mesh enters, the vertex starts to deflect in order to avoid a self-collision.

Set this value to the fractional distance between vertices that you want them to have their own "space". Too high of a value will include too many vertices all the time and slow down the calculation. Too low of a level will let other vertices get too close and thus possibly intersect because there will not be enough time to slow them down.

Stiffness Default 1.0. How elastic that ball of personal space is.

Damping Default 0.5. How the vertex reacts. A low value just slows down the vertex as it gets too close. A high value repulses it.

Collisions with other objects are set in the (other) *Collision panel*. To collide with another object they have to share at least one common layer.

Simple Examples

Here are some simple examples showing the power of softbody physics.

Bouncing Cube

The Process

First, change your start and end frames to 1 and 150.

Fig. 2.1366: The timeline.

Then, add a plane, and scale it five times. Next, go to the physics tab, and add a collision. The default settings are fine for this example.

Now add a cube, or use the default cube. Tab into edit mode and subdivide it three times. Add a bevel modifier to it to smoothen the edges and then to add a little more, press r twice, and move your cursor a bit.

When finished, your scene should look like this:

Everything is ready to add the softbody physics. Go to *Properties* → *Physics* and choose *Softbody*. Uncheck the soft body goal, and check softbody self collision. Also, under soft body edges, increase the bending to 10.

Playing tha animation with `Alt-A` will now give a slow animation of a bouncing cube. To speed things up, we need to bake the softbody physics.

Under *Soft Body Cache* change the start and end values to your start and end frames. In this case 1 and 150. Now, to test if everything is working, you can take a cache step of 5 or 10, but for the final animation it is better to reduce it to 1, to cache everything.

When finished, your physics panel should look like this:

You can now bake the simulation, give the cube materials and textures and render the animation.

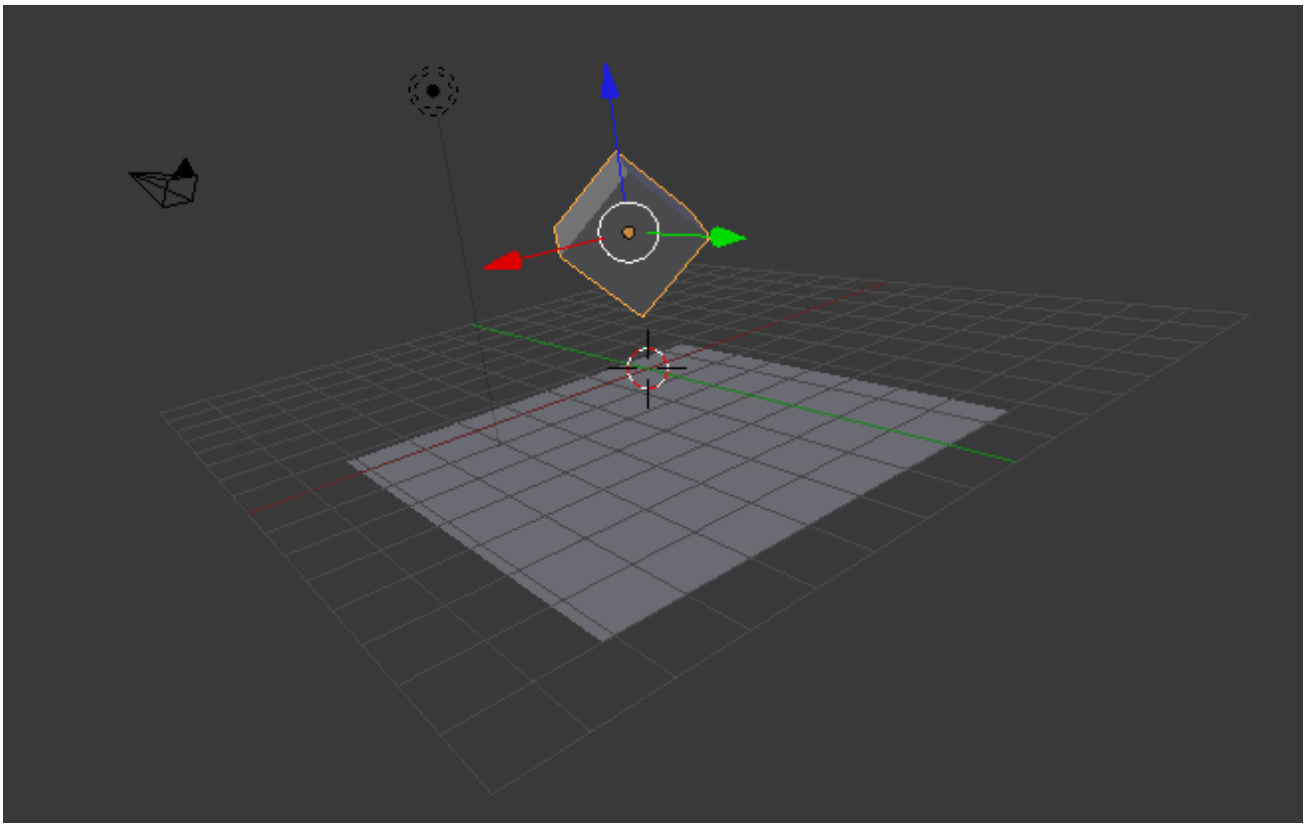


Fig. 2.1367: The scene, ready for softbody physics.

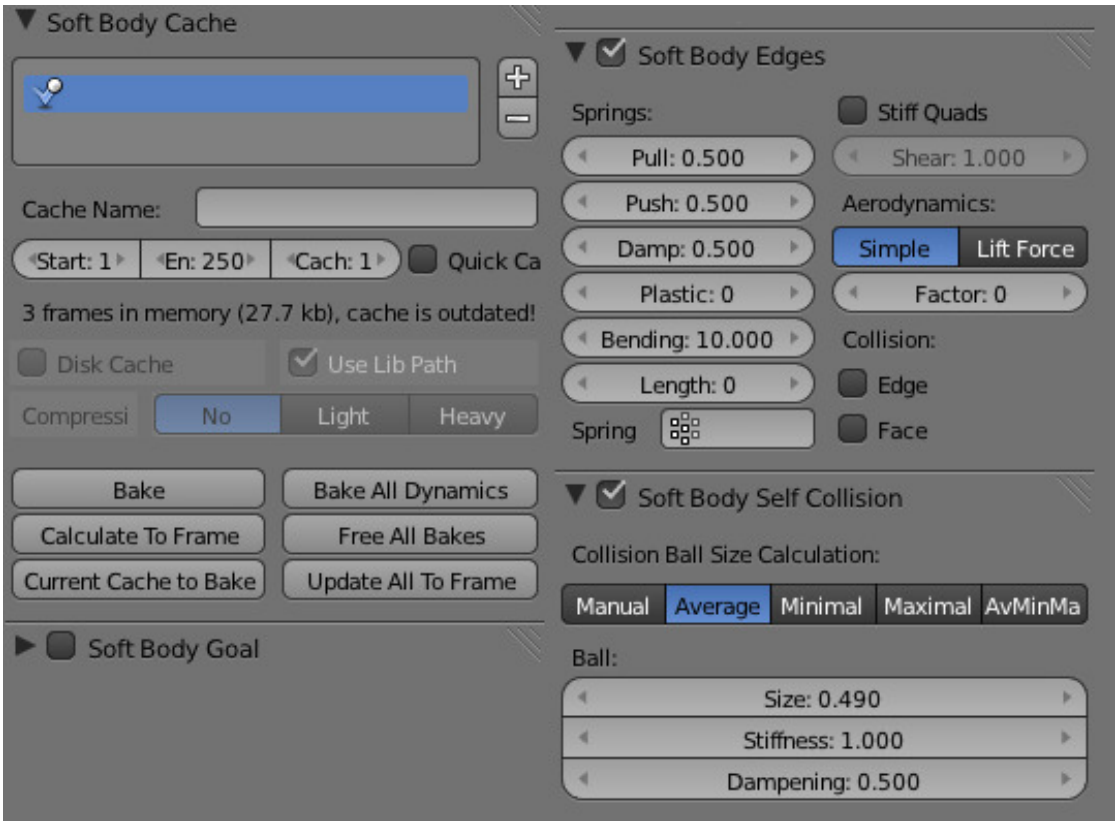


Fig. 2.1368: The physics settings.

The Result

The rendered bouncing cube:

Fluid Simulation

Introduction

Fluid physics are used to simulate physical properties of liquids especially water. While creating a scene in Blender, certain objects can be marked to participate in the fluid simulation. These can include but not limited to, being a fluid or as an obstacle. For a fluid simulation you have to have a domain to define the space that the simulation takes up. In the domain settings you will be able to define the global simulation parameters (such as viscosity and gravity).

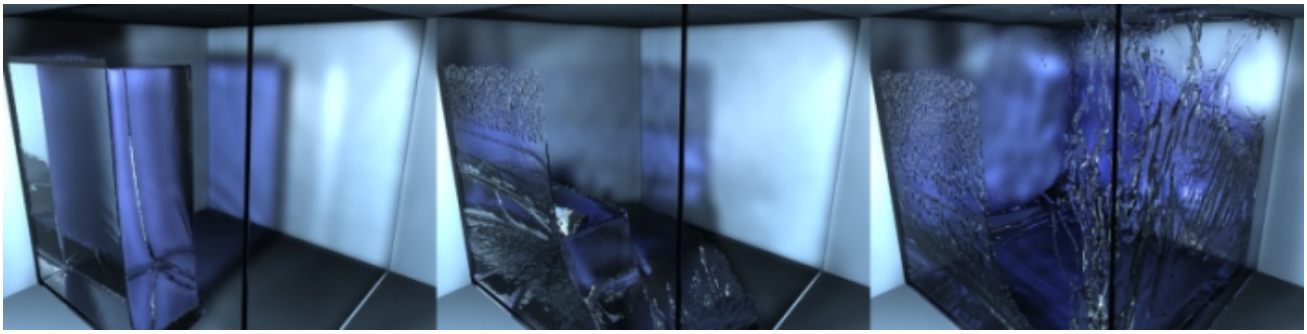


Fig. 2.1369: Example of Fluid Simulation.

Workflow

In general, you follow these steps:

1. First you want to set the *simulation domain*,
2. Next set the *fluid source(s)*, and specify there physical properties.
3. In some cases you may want to set other objects to *Control the Flow* of the fluid.
4. You can also depending on your scene add other objects related to the fluid, like: *Obstacles*, *Particles* floating on the fluid.
5. And lastly you must *Bake the Simulation*.

Tip: Baking is done on the Domain object!

When you calculate the fluid simulation, you bake the simulation on the domain object.

For this reason:

- All the baking options are visible only when selecting the Domain Object,
- Baking options are explained in the *the baking section* of the Domain manual page.

See also:

To know more about simulating fluids in Blender you can read the *fluids appendix*. There you can find the limitations and workarounds, and some additional links.

Fluid Types

Introduction

Common Options

Animated Mesh/Export

Click this button if the network is animated (eg . Deformed by an armature, shape keys, or lattice). It can become very slow and is not necessary if the network's position and rotation are animated. (ie only object transformations).

Volume Initialization Type

A common option amoung the different fluid types is *Volume Initialization*.

Volume The inside of the object is initialized as fluid all . This works only if the closed mesh .

Shell It is initialized as a thin fluid layer of the surface of the mesh . This can also be used in the mesh open.

Both It is a state , such as the sum of the Volume and Shell. This also must be a closed mesh.

Fig. 2.1370: Example of different types of initiation of volume

Fluid Domain

The Domain Object

The bounding box of the object serves as the boundary of the simulation. All fluid objects **must** be in the domain. Fluid objects outside the domain will not bake. No tiny droplets can move outside this domain; it's as if the fluid is contained within the 3D space by invisible force fields. There can be only a single fluid simulation domain object in the scene.

The shape of the object does **not** matter because it will *always* be treated like a box (The lengths of the bounding box sides can be different). So, usually there will not be any reason to use another shape than a box. If you need obstacles or other boundaries than a box to interfere with the fluid flow, you need to insert additional obstacle objects *inside* the domain boundary.

This object will be *replaced* by the fluid during the simulation.

Tip: Baking is done on the Domain object

When you calculate the fluid simulation, you bake the simulation on the domain object. For this reason all the baking options are visible only when selecting the Domain Object.

For baking options, see *Baking*.

Options

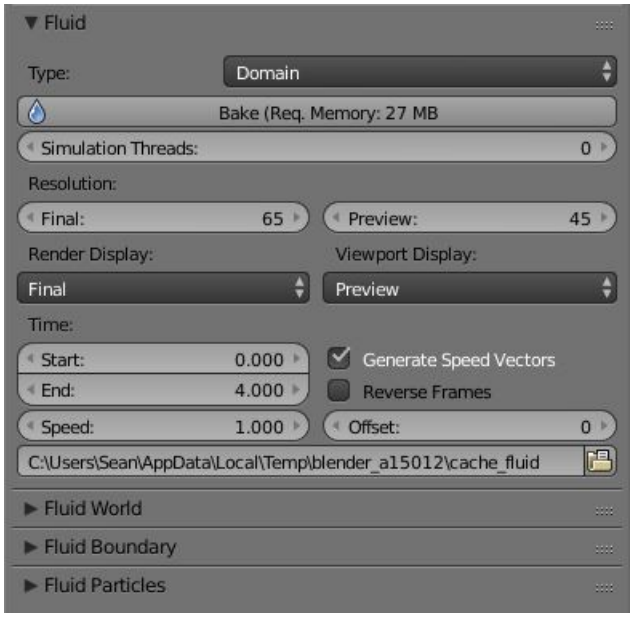


Fig. 2.1371: Fluid Domain Settings.

Bake button For baking options, see *Baking*.

Resolution

Render resolution The granularity at which the actual fluid simulation is performed. This is probably the most important setting for the simulation as it determines the amount of details in the fluid, the memory and disk usage as well as computational time.

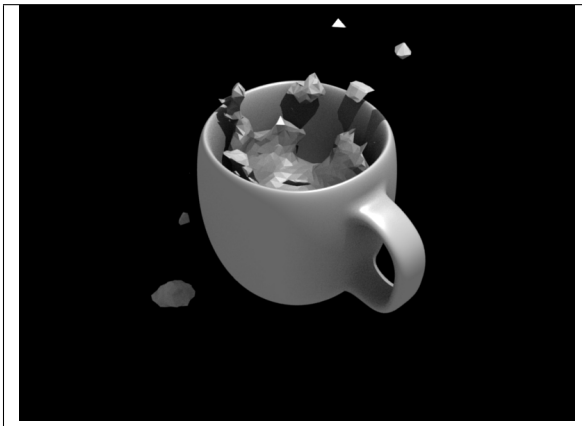


Fig. 2.1372: 10cm mug at Resolution 70.

Fig. 2.1373: 10cm mug at Resolution 200.

Note: The amount of required memory quickly increases: a resolution of 32 requires ca. 4MB, 64 requires ca. 30MB, while 128 already needs more than 230MB. Make sure to set the resolution low enough, depending on how much memory you have, to prevent Blender from crashing or freezing. Remember also that many operating systems limit the amount of memory that can be allocated by a single *process*, such as Blender, even if the *machine* contains much more than this. Find out what limitations apply to your machine.

Note: Resolution and Real-size of the Domain

Be sure to set the resolution appropriate to the real-world size of the domain (see the *Realworld-size* in the *Domain World*). If the domain is not cubic, the resolution will be taken for the longest side. The resolutions along the other sides will be reduced according to their lengths (therefore, a non-cubic domain will need less memory than a cubic one, resolutions being the same).

Preview resolution This is the resolution at which the preview surface meshes will be generated. So it does not influence the actual simulation. Even if "there is nothing to see" in the preview, there might be a thin fluid surface that cannot be resolved in the preview.

Display quality How to display a baked simulation in the 3D View (menu *Viewport Display*) and for rendering (menu *Render Display*):

Geometry Use the original geometry (before simulation).

Preview Use the preview mesh.

Final Use the final high definition mesh.

When no baked data is found, the original mesh will be displayed by default.

After you have baked a domain, it is displayed (usually) in the Blender window as the preview mesh. To see the size and scope of the original domain box, select *Geometry* in the left selector.

Time

Start It is the simulation start time (in seconds).

This option makes the simulation computation in Blender start later in the simulation. The domain deformations and fluid flow prior to the start time are not saved.

For example, if you wanted the fluid to appear to already have been flowing for 4 seconds before the actual first frame of data, you would enter 4.0 here.

End It is the simulation ending time (in seconds).

Tip: Start and end times have nothing to do with how many frames are baked

If you set *Start* time to 3.0, and *End* time to 4.0, you will simulate 1 second of fluid motion. That one second of fluid motion will be spread across however-many frames are set in *Render* → *Dimensions*.

This means, for example, that if you have Blender set to make 250 frames at 25 fps, the fluid will look like it had already been flowing for 3 seconds at the start of the simulation, *but* will play in slow motion (one-tenth normal speed), since the 1 second fluid sim plays out over the course of 10 video seconds. To correct this, change the end time to 13.0 (3.0 + 10.0) to match the 250 frames at 25 fps. Now, the simulation will be real-time, since you set 10 seconds of fluid motion to simulate over 10 seconds of animation. Having these controls in effect gives you a "speed control" over the simulation.

Generate Speed Vector If this button is clicked, no speed vectors will be exported. So by default, speed vectors are generated and stored on disk. They can be used to compute image based motion blur with the compositing nodes.

Reverse fluid frames The simulation is calculated backward

***Bake* directory** For baking options see *Baking*.

Domain World

Viscosity Presets The "thickness" of the fluid and actually the force needed to move an object of a certain surface area through it at a certain speed.

For manual entry, please note that the normal real-world viscosity (the so-called dynamic viscosity) is measured in Pascal-seconds (Pa.s), or in Poise units (P, equal to 0.1 Pa.s, pronounced *pwaz*, from the Frenchman Jean-Louis

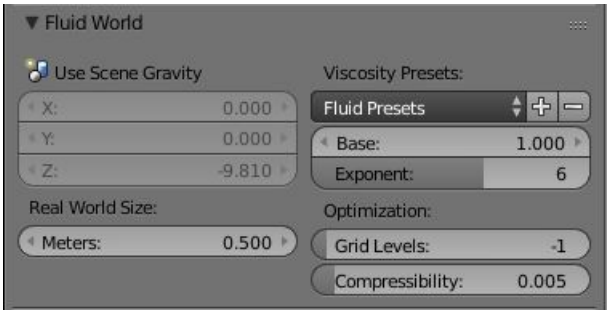


Fig. 2.1374: The Domain World options.

Poiseuille, who discovered the laws on "the laminar flow of viscous fluids"), and commonly centiPoise units (cP, equal to 0.001 Pa.s, *sentipwaz*). Blender, on the other hand, uses the kinematic viscosity (which is dynamic viscosity in Pa.s, divided by the density in $kg.m^{-3}$, unit $m^2.s^{-1}$). The table below gives some examples of fluids together with their dynamic and kinematic viscosities.

Table 2.78: Blender Viscosity Unit Conversion.

Fluid	dynamic viscosity (in cP)	kinematic viscosity (Blender, in $m^2.s^{-1}$)
Water (20- C)	1.002×10^{0} (1.002)	1.002×10^{-6} (0.000001002)
Oil SAE 50	5.0×10^{2} (500)	5.0×10^{-5} (0.00005)
Honey (20- C)	1.0×10^{4} (10,000)	2.0×10^{-3} (0.002)
Chocolate Syrup	3.0×10^{4} (30,000)	3.0×10^{-3} (0.003)
Ketchup	1.0×10^{5} (100,000)	1.0×10^{-1} (0.1)
Melting Glass	1.0×10^{15}	1.0×10^{0} (1.0)

Manual entries are specified by a floating point number and an exponent. These floating point and exponent entry fields (scientific notation) simplify entering very small or large numbers. The viscosity of water at room temperature is 1.002 cP, ou 0.001002 Pa.s; the density of water is about 1000 $kg.m^{-3}$, which gives a kinematic viscosity of 0.000001002 $m^2.s^{-1}$ – so the entry would be 1.002 times 10 to the minus six (1.002×10^{-6} in scientific notation). Hot Glass and melting iron is a fluid, but very thick; you should enter something like 1.0×10^{0} (= 1.0) as its kinematic viscosity (indicating a value of 1.0×10^{6}cP).

Note that the simulator is not suitable for non-fluids, such as materials that do not "flow". Simply setting the viscosity to very large values will not result in rigid body behavior, but might cause instabilities.

Note: Viscosity varies

The default values in Blender are considered typical for those types of fluids and "look right" when animated. However, actual viscosity of some fluids, especially sugar-laden fluids like chocolate syrup and honey, depend highly on temperature and concentration. Oil viscosity varies by SAE rating. Glass at room temperature is basically a solid, but glass at 1500 degrees Celsius flows (nearly) like water.

Real World Size Size of the domain object in the real world in meters. If you want to create a mug of coffee, this might be 10 cm (0.1 meters), while a swimming pool might be 10m. The size set here is for the longest side of the domain bounding box.

Optimization

Gridlevel How many adaptive grid levels to be used during simulation. Setting this to -1 will perform automatic selection.

Compressibility If you have problems with large standing fluid regions at high resolution, it might help to reduce this number (note that this will increase computation times).

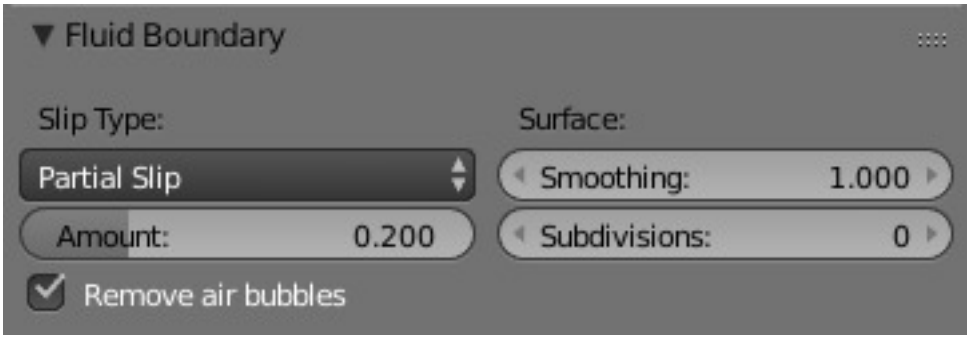


Fig. 2.1375: The Domain Boundary panel.

Fluid Boundary

This box has all the slip and surface options.

Boundary type The stickiness of the surface of the obstacle, to determine the "tacky surface (Surface Adhesion)." In the real world, and the tackiness and fluid, the granularity of the object surface, tack, determined by the elasticity.

No Slip Fluid will stick To snugly (speed 0).

Free Slip Fluid will move on the object (0 normal direction of speed).

Part Slip It is a two intermediate. It is almost No slip, 1 in the Free exactly the same in 0.

Surface

Surface Smoothing Amount of smoothing to be applied to the fluid surface. 1.0 is standard, 0 is off, while larger values increase the amount of smoothing.

Subdivisions Allows the creation of high-res surface meshes directly during the simulation (as opposed to doing it afterwards like a subdivision modifier). A value of 1 means no subdivision, and each increase results in one further subdivision of each fluid voxel. The resulting meshes thus quickly become large, and can require large amounts of disk space. Be careful in combination with large smoothing values – this can lead to long computation times due to the surface mesh generation.

Fluid Particles

Fig. 2.1376: The Domain Particles Panel.

Here you can add particles to the fluid simulated, to enhance the visual effect.

Tracer Particles Number of tracer particles to be put into the fluid at the beginning of the simulation. To display them create another object with the *Particle* fluid type, explained below, that uses the same bake directory as the domain.

Generate Particles Controls the amount of fluid particles to create (0=off, 1=normal, >1=more). To use it, you have to have a surface subdivision value of at least 2.

Fluid Object

All regions of this object that are inside the domain bounding box will be used as actual fluid in the simulation. If you place more than one fluid object inside the domain, they should currently not intersect. Also make sure the surface normals are

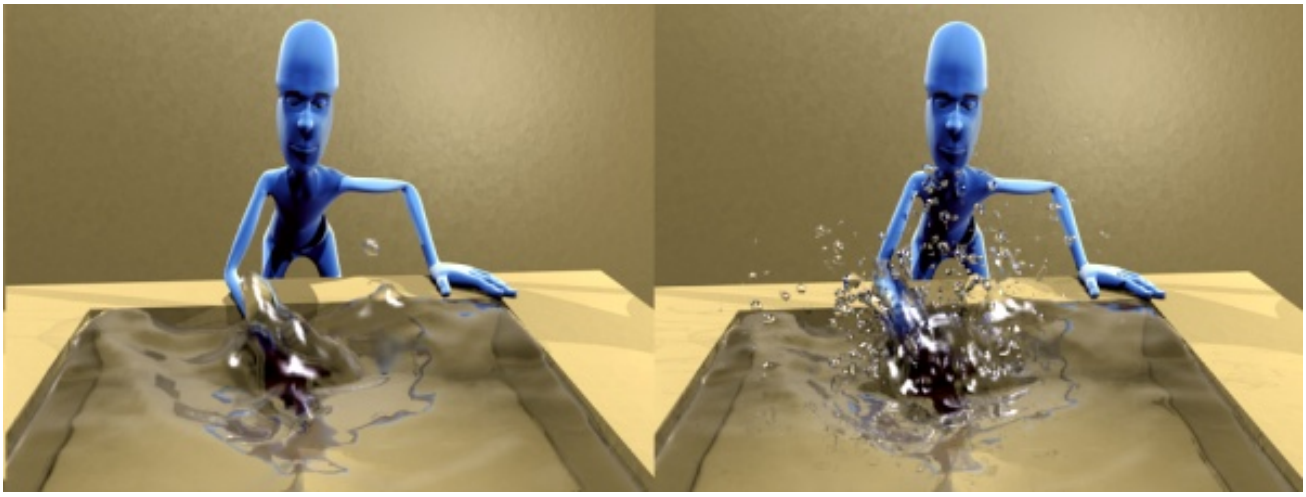


Fig. 2.1377: An example of Particles effects.
Left: Simulated without; Right: With particles and subdivision enabled.

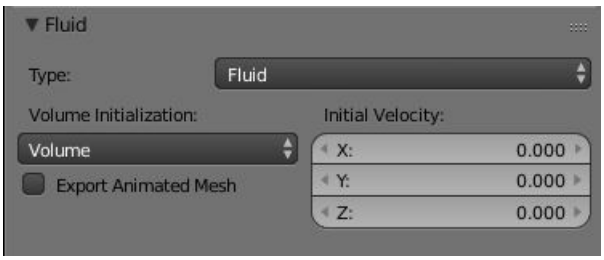


Fig. 2.1378: Fluid object settings.

pointing outwards. In contrast to domain objects, the actual mesh geometry is used for fluid objects.

Volume Initialization Type See *Volume Initialization Type*

Animated Mesh/Export See *Animated Mesh/Export*

Initial velocity Speed of the fluid at the beginning of the simulation, in meters per second.

Tip: The direction of Surface Normals makes a big difference!

Blender uses the orientation of the Surface Normals to determine what is "inside of" the Fluid object and what is "outside". You want all of the normals to face *outside* (in *Edit Mode*, use `Ctrl-N` or press `Spacebar` and choose *Edit*?? *Normals*?? *Calculate Outside*). If the normals face the wrong way, you will be rewarded with a "gigantic flood of water" because Blender will think that the volume of the object is outside of its mesh! This applies regardless of the *Volume init* type setting.

Fluid Obstacle

This object will be used as an obstacle in the simulation. As with a fluid object, obstacle objects currently should not intersect. As for fluid objects, the actual mesh geometry is used for obstacles. For objects with a volume, make sure that the normals of the obstacle are calculated correctly, and radiating properly (use the *Flip Normal* button, in *Edit Mode*, *Mesh Tools* panel, in the Tool shelf), particularly when using a spinned container. Applying a *Subdivision Surface Modifier* before baking the simulation could also be a good idea if the mesh is not animated.

Volume Initialization Type See *Volume Initialization Type*

Boundary type Determines the stickiness of the obstacle surface, called "Surface Adhesion". Surface Adhesion depends in real-world on the fluid and the graininess or friction/adhesion/absorption qualities of the surface.

No Slip Causes the fluid to stick to the obstacle (zero velocity).

Free Slip Allows movement along the obstacle (only zero normal velocity).

Part Slip Mixes both types, with 0 being mostly no slip, and 1 being identical to free slip.

Note that if the mesh is moving, it will be treated as no slip automatically.

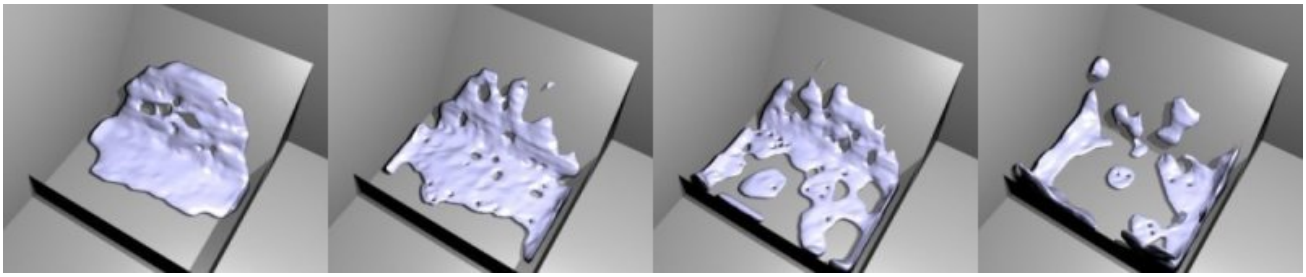


Fig. 2.1379: Example of the different boundary types for a drop falling onto the slanted wall. From left to right: no-slip, part-slip 0.3, part-slip 0.7 and free-slip.

Animated Mesh/Export See *Animated Mesh/Export*

Part Slip Amount Amount of mixing between no- and free-slip, described above.

Impact Factor Amount of fluid volume correction for gain/loss from impacting with moving objects. If this object is not moving, this setting has no effect. However, it if is and the fluid collides with it, a negative value takes volume away from the Domain, and a positive number adds to it. Ranges from -2.0 to 10.0.

Fluid Inflow / Outflow

To control the volume of the fluid simulation, you can set objects in the scene to add or absorb fluid within the *Fluid Domain*.

Inflow

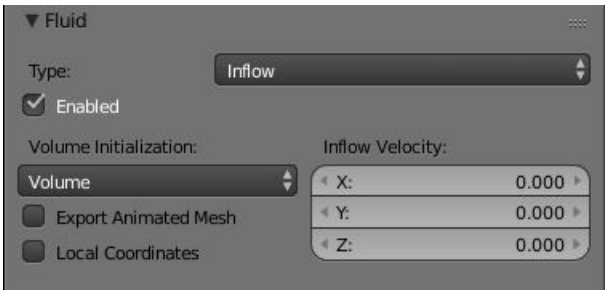


Fig. 2.1380: Fluid Inflow Settings.

Volume Initialization Type See *Volume Initialization Type*

This object will put fluid into the simulation, like a water tap.

Inflow Velocity Speed of the fluid that is created inside of the object.

Local Coords/Enable Use local coordinates for the inflow. This is useful if the inflow object is moving or rotating, as the inflow stream will follow/copy that motion. If disabled, the inflow location and direction do not change.

Animated Mesh/Export See *Animated Mesh/Export*

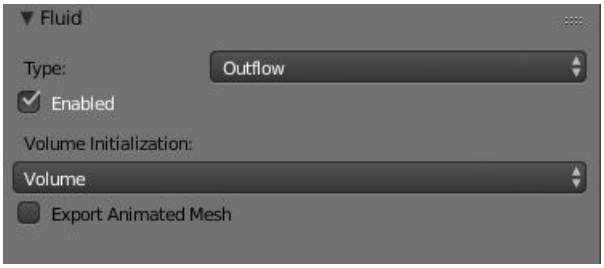


Fig. 2.1381: Fluid Outflow Settings.

Outflow

Any fluid that enters the region of this object will be deleted (think of a drain or a black hole). This can be useful in combination with an inflow to prevent the whole domain from filling up. When enabled, this is like a tornado (waterspout) or "wet vac" vacuum cleaner, and the part where the fluid disappears will follow the object as it moves around.

Volume Initialization Type See *Volume Initialization Type*

Animated Mesh/Export See *Animated Mesh/Export*

Fluid Particle

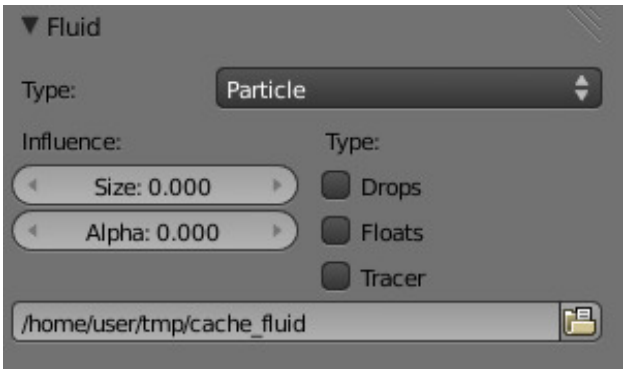


Fig. 2.1382: Fluid particle settings.

This type can be used to display particles created during the simulation. For now only tracers swimming along with the fluid are supported.

Note that the object can have any shape, position or type. Once the particle button is pressed, a particle system with the fluid simulation particles will be created for it at the correct position. When moving the original object, it might be necessary to delete the particle system, disable the fluidsim particles, and enable them again. The fluidsim particles are currently also unaffected by any other particle forces or settings.

Influence

Size Influence The particles can have different sizes, if this value is 0 all are forced to be the same size.

Alpha Influence If this value is >0, the alpha values of the particles are changed according to their size.

Particle type

Drops Surface splashes of the fluid result in droplets being strewn about, like fresh water, with low Surface Tension.

Floats The surface tension of the fluid is higher and the fluid heavier, like cold seawater and soup. Breakaways are clumpier and fall back to the surface faster than *Drops*, as with high Surface Tension.

Tracer Droplets follow the surface of the water where it existed, like a fog suspended above previous fluid levels. Use this to see where the fluid level has been.

Path (bake directory) The simulation run from which to load the particles. This should usually have the same value as the fluid domain object (e.g. copy by `Ctrl-C`, `Ctrl-V`).

Fluid Control

Using the Lattice-boltzman method, the fluid is controlled using particles which define local force fields and are generated automatically from either a physical simulation or a sequence of target shapes. At the same time, as much as possible of the natural fluid motion is preserved.

Options

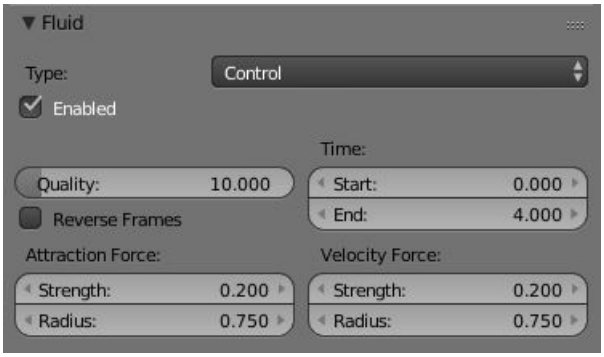


Fig. 2.1383: Fluid control options.

Quality Higher quality result in more control particles for the fluid control object.

Reverse Frames The control particle movement gets reversed.

Time You specify the start and end time during which time the fluid control object is active.

Attraction force The attraction force specifies the force which gets emitted by the fluid control object. Positive force results in attraction of the fluid, negative force in avoidance.

Velocity force If the fluid control object moves, the resulting velocity can also introduce a force to the fluid.

Examples

In this examples, we use the Fluid Control option to control part of the fluid so that it has a certain shape (the sphere drop or the teapot drop) before it falls in the rest of the fluid:

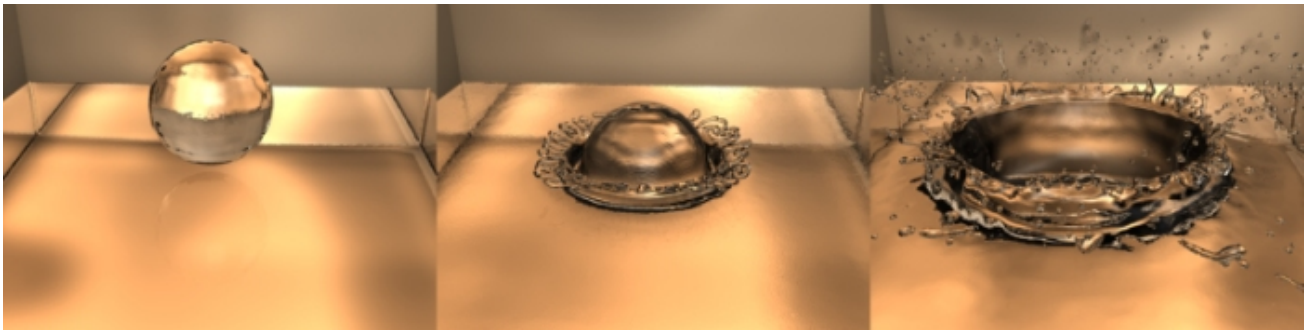


Fig. 2.1384: Falling drop.

Fig. 2.1385: "Magic Fluid Control."

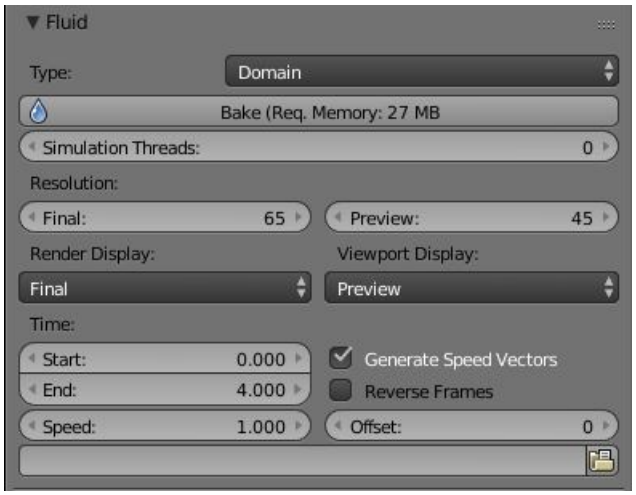


Fig. 2.1386: The fluid simulation options with Domain selected.

Baking

Bake Button

Perform the actual fluid simulation. Blender will continue to work normally, except there will be a progress bar in the header of the Info Editor, next to the render pulldown. Pressing `Esc` or the "x" next to the status bar will abort the simulation. Afterwards two `.bobj.gz` (one for the *Final* quality, one for the *Preview* quality), plus one `.bvel.gz` (for the *Final* quality) will be in the selected output directory for each frame.

Bake directory

Directory and file prefix to store baked surface meshes.

This is similar to the animation output settings, only selecting a file is a bit special: when you select any of the previously generated surface meshes (e.g. `test1_fluidsurface_final_0132.bobj.gz`), the prefix will be automatically set (`test1_` in this example). This way the simulation can be done several times with different settings, and allows quick changes between the different sets of surface data.

Notes

Unique domain Because of the possibility of spanning and linking between scenes, there can only be one domain in an entire blend-file.

Selecting a Baked Domain After a domain has been baked, it changes to the fluid mesh. To re-select the domain so that you can bake it again after you have made changes, go to any frame and select `RMB` the fluid mesh. Then you can click the *bake* button again to recompute the fluid flows inside that domain.

Baking always starts at Frame #1 The fluid simulator disregards the *Start* setting in the *Animation* panel, it will always bake from frame 1. If you wish the simulation to start later than frame 1, you must key the fluid objects in your domain to be inactive until the frame you desire to start the simulation.

Baking always ends at the *End* Frame set in the *Animation* panel If your frame-rate is 25 frames per second, and ending time is 4.0 seconds, then you should (if your start time is 0) set your animation to end at frame `4.0 × 25 = 100`

Freeing the previous baked solutions Deleting the content of the "Bake" directory is a destructive way to achieve this. Be careful if more than one simulation uses the same bake directory (be sure they use different filenames, or they will overwrite one another)!

Reusing Bakes Manually entering (or searching for) a previously saved (baked) computational directory and filename mask will switch the fluid flow and mesh deformation to use that which existed during the old bake. Thus, you can re-use baked flows by simply pointing to them in this field.

Baking processing time Baking takes a **lot** of compute power (hence time). Depending on the scene, it might be preferable to bake overnight.

If the mesh has modifiers, the rendering settings are used for exporting the mesh to the fluid solver. Depending on the setting, calculation times and memory use might exponentially increase. For example, when using a moving mesh with *Subdivision Surface* as an obstacle, it might help to decrease simulation time by switching it off, or to a low subdivision level. When the setup/rig is correct, you can always increase settings to yield a more realistic result.

Fluid Appendix

Hints

Some useful hints about fluid simulation in Blender:

- Do not be surprised, but you will get whole bunch of mesh (.bobj.gz) files after a simulation. One set for preview, and another for final. Each set has a `.gz` file for each frame of the animation. Each file contains the simulation result – so you will need them.
- Currently these files will not be automatically deleted, so it is a good idea to e.g. create a dedicated directory to keep simulation results. Doing a fluid simulation is similar to clicking the *animation* button. Currently you have to take care of organizing the fluid surface meshes in some directory by yourself. If you want to stop using the fluid simulation, you can simply delete all the `*fluid*.bobj.gz` files.
- Before running a high resolution simulation that might take hours, check the overall timing first by doing lower resolution runs.
- Fluid objects must be completely inside the bounding box of the domain object. If not, baking may not work correctly or at all. Fluid and obstacle objects can be meshes with complex geometries. Very thin objects might not appear in the simulation, if the chosen resolution is too coarse to resolve them (increasing it might solve this problem).
- Do not try to do a complicated scene all at once. Blender has a powerful compositor that you can use to combine multiple animations.

  For example, to produce an animation showing two separate fluid flows while keeping your domain small, render one `.avi` using the one flow. Then move the domain and render another `.avi` with the other flow using an alpha channel (in a separate B&W .avi?). Then, composite both `.avi`'s using the compositor's add function. A third `.avi` is usually the smoke and mist and it is laid on top of everything as well. Add a rain sheet on top of the mist and spray and you will have quite a storm brewing! And then lightning flashes, trash blowing around, all as separate animations, compositing the total for a truly spectacular result.

Limitations & Workarounds

- If the setup seems to go wrong, make sure all the normals are correct (hence, enter *Edit Mode*, select all, and recalculate normals once in a while).

- Currently there is a problem with zero gravity simulation. It could be avoided by simply selecting a very small gravity until this is fixed.
- If an object is initialized as *Volume*, it has to be closed and have an inner side (a plane will not work). To use planes, switch to *Shell*, or extrude the plane.
- Blender freezes after clicking *bake*. Pressing `Esc` makes it work again after a while – this can happen if the resolution is too high and memory is swapped to hard disk, making everything horribly slow. Reducing the resolution should help in this case.
- Blender crashes after clicking *bake* – this can happen if the resolution is really high and more than 2GB are allocated, causing Blender to crash. Reduce the resolution. Many operating systems limit the total amount of memory that can be allocated by a *process*, such as Blender, even if the *machine* has more memory installed.
- The meshes should be closed, so if some parts of e.g. a fluid object are not initialized as fluid in the simulation, check that all parts of connected vertices are closed meshes. Unfortunately, the Suzanne (monkey) mesh in Blender is not a closed mesh (the eyes are separate).
- If the fluid simulation exits with an error message (stating e.g. that the "init has failed"), make sure you have valid settings for the domain object, e.g. by resetting them to the defaults.
- Note that first frame may well take only a few hundred MBs of RAM memory, but latter ones go over one GB, which may be why your bake fails after awhile. If so, try to bake one frame at the middle or end at full res so you will see if it works.
- Memory used doubles when you set surface subdivision from 1 to 2.
- Using "generate particles" will also add memory requirements, as they increase surface area and complexity. Ordinary fluid-sim generated particles probably eat less memory.

Smoke Simulation

Introduction

Smoke simulation is used to simulate the fluid movement of air and generate animated *voxel* textures representing the density, heat, and velocity of other fluids or suspended particles (i.e. smoke) which can be used for rendering.

Smoke and fire are emitted into a *Domain* from a mesh object or particle system. Smoke movement is controlled by airflow inside the domain, which can be influenced by *smoke collision objects*. Smoke will also be affected by scene gravity and *force fields*. Airflow inside the domain can affect other physics simulations via the smoke flow force field.

Workflow

At least a *Domain Object* object and one *Flow object* are required to create a smoke simulation. A basic workflow looks like this:

1. Create a *Domain Object* that defines the bounds of the simulation volume.
2. Define a *Flow object* or objects which will emit smoke and fire.
3. Set *Collision objects* to make the smoke interact with objects in the scene.
4. Assign a *Volumetric material* to the domain object.
5. Save the blend-file.
6. *Bake* the simulation.

Note: There is a *Quick Smoke* operator which will automatically create a domain object with a basic smoke/fire material. It can be found in *3D View* → *Object* → *Quick Effects* → *Quick Smoke*, or in the `Spacebar` search box.

Fig. 2.1387: Example of smoke simulation.

Note: Blender's smoke simulation is based on the paper Wavelet Turbulence for Fluid Simulation and associated sample code.

Smoke Types

Smoke Domain

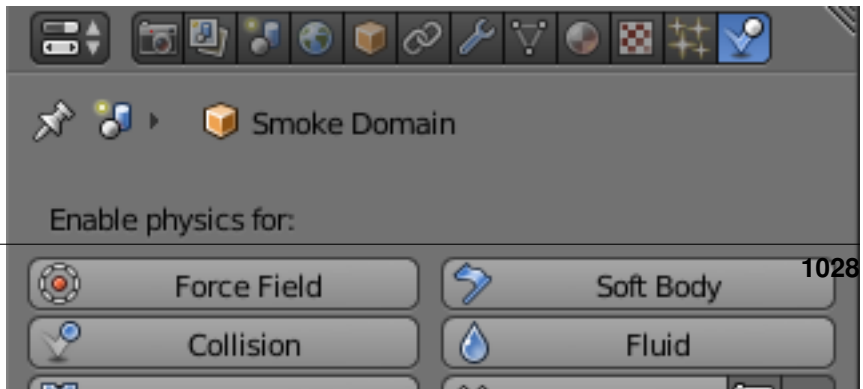
The domain object contains the entire simulation. Smoke and fire cannot leave the domain, it will either collide with the edge or disappear, depending on the domain's settings.

Keep in mind that large domains require higher resolutions and longer bake times. You will want to make it just large enough that the simulation will fit inside it, but not so large that it takes too long to compute the simulation.

To create a domain, add a cube *Add* → *Mesh* → *Cube*, `Shift-A` and transform it until it encloses the area where you want smoke. Translation, rotation, and scaling are all allowed. To turn it into a smoke domain, click *Smoke* in *Properties* → *Physics*, then select *Domain* as the *Smoke Type*.

Note: You *can* use other shapes of mesh objects as domain objects, but the smoke simulator will use the shape's *bounding box* as the domain bounds. In other words, the actual shape of the domain will still be rectangular.

Settings



Resolution The smoke domain is subdivided into many "cells" called voxels (see *voxel*) which make up "pixels" of smoke. This setting controls

the number of subdivisions in the domain. Higher numbers of subdivisions are one way of creating higher resolution smoke (See *Smoke High Resolution*)

Since the resolution is defined in terms of *subdivisions*, larger domains will need more divisions to get an equivalent resolution to a small domain.

Also see *Note on Divisions and High Resolution*.

Time Scale Controls the speed of the simulation. Low values result in a "slow motion" simulation, while higher values can be used to advance the simulation faster (useful for generating smoke for use in still renders).

Border Collisions Controls which sides of the domain will allow smoke "through" the domain, making it disappear without influencing the rest of the simulation, and which sides will deflect smoke as if colliding with a *Collision Object*.

Vertically Open Smoke disappears when it hits the top or bottom of the domain, but collides with the walls.

Open Smoke disappears when it hits any side of the domain.

Collide All Smoke collides with all sides of the domain.

Density Controls how much smoke is affected by density.

- Values above 0 will cause the smoke to rise (simulating smoke which is lighter than ambient air).
- Values below 0 will cause smoke to sink (simulating smoke which is heavier than ambient air).

Temp. Diff. The *Temperature Difference* setting controls how much smoke is affected by temperature.

The effect this setting has on smoke depends on the *per flow object *Temp. Diff.* setting*:

- Values above 0 will result in the smoke rising when the flow object *Temp. Diff.* is set to a positive value, and smoke sinking when the flow object *Temp. Diff.* is set to a negative value.
- Values below 0 will result in the opposite of positive values, i.e. smoke emit-

Vortic

Fig. 2.1389: Comparison of different amounts of vorticity. The domain on the left has a vorticity of 3, while the domain on the right has a vorticity of 0.01.

Dissolve Allow smoke to dissipate over time.

Time Speed of smoke's dissipation in frames.

Slow Dissolve smoke in a logarithmic fashion. Dissolves quickly at first, but lingers longer.

Smoke Flames

Speed How fast fuel burns. Larger values result in smaller flames (fuel burns before it can go very far), smaller values result in larger flames (fuel has time to flow farther before being fully consumed).

Smoke Amount of extra smoke created automatically to simulate burnt fuel.

Vorticity Additional vorticity for flames.

Ignition Minimum temperature of flames.

Maximum Maximum temperature of flames.

Smoke Color Color of smoke created by burnt fuel.

Smoke Adaptive Domain

When enabled, the domain will adaptively shrink to best fit the smoke, saving computation time by leaving voxels without smoke out of the simulation. Unless the *Additional* option is used, the adaptive domain will not exceed the bounds of the original domain.

Additional Number of voxels to add around the outside of the domain.

Margin Amount of extra space to leave around smoke, measured in voxels. With very fast moving smoke larger margins may be required to prevent the smoke from being cut off by the adaptive boundary, but note this will increase the number of voxels which need to be computed.

Threshold Smallest amount of smoke a voxel can contain before it is considered empty and the adaptive domain is allowed to cut it out of the simulation.

Smoke High Resolution

The High Resolution option lets you simulate at low resolution and then uses

noise techniques to enhance the resolution without actually computing it. This allows animators to set up a low resolution simulation quickly and later add details without changing the overall fluid motion. Also see *Note on Divisions and High Resolution*.

Resolution/Divisions Factor by which to enhance the resolution of smoke using the specified noise method.

Show High Resolution Show high resolution in the viewport (may cause viewport responsiveness to suffer).

Noise Method The two options, *Wavelet* and *FFT*, are very similar.

Note: *Wavelet* is an implementation of Turbulence for Fluid Simulation.

Strength Strength of noise.

Smoke Groups

Flow Group If set, only objects in the specified *Group* will be allowed to act as flow objects in this domain.

Collision Group If set, only objects in the specified *Group* will be allowed to act as collision objects in this domain.

Smoke Cache

See *Baking*.

Smoke Field Weights

These settings determine how much gravity and *Force Fields* affect the smoke.

Effector Group When set, smoke can only be influenced by force fields in the specified group.

Gravity How much the smoke is affected by Gravity.

All Overall influence of all force fields.

The other settings determine how much influence individual force field types have.

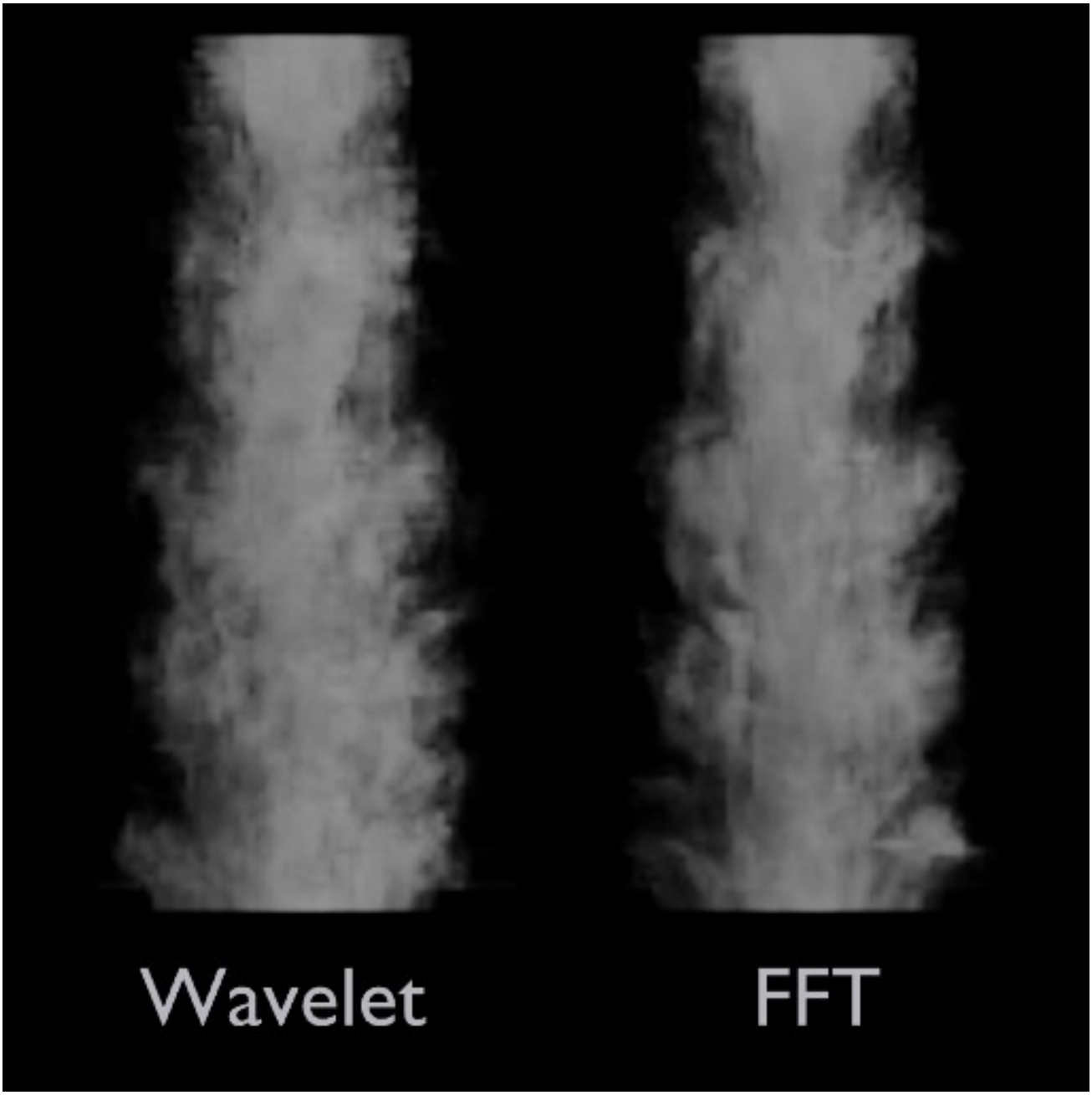


Fig. 2.1390: Comparison of noise methods. *Wavelet* on the left, *FFT* on the right.

Fig. 2.1391: From left to right, the domains' high resolution strengths are set to 0, 2, and 6.

Fig. 2.1393: Comparison between a domain with 24 divisions and 4 *High Resolution* divisions (left), and a domain with 100 divisions and 1 *High Resolution* division (right).

Low division simulations with lots of *High Resolution* divisions generally appear smaller in real-world scale (larger flames etc.) and can be used to achieve pyroclastic plumes such as this:

High *Domain Division* simulations tend

to appear larger in real-world scale, with many smaller details.

Smoke Flow Object

Smoke Flow objects are used to add or remove smoke and fire to a *Smoke Domain* object.

To define any mesh object as a *Smoke Flow* object, add smoke physics by clicking *Smoke* in *Properties* → *Physics*. Then select *Flow* as the *Smoke Type*. Now you should have a default smoke flow source object. You can test this by playing the animation `Alt-A` from the first frame. If your source object is inside your domain, you should see smoke.

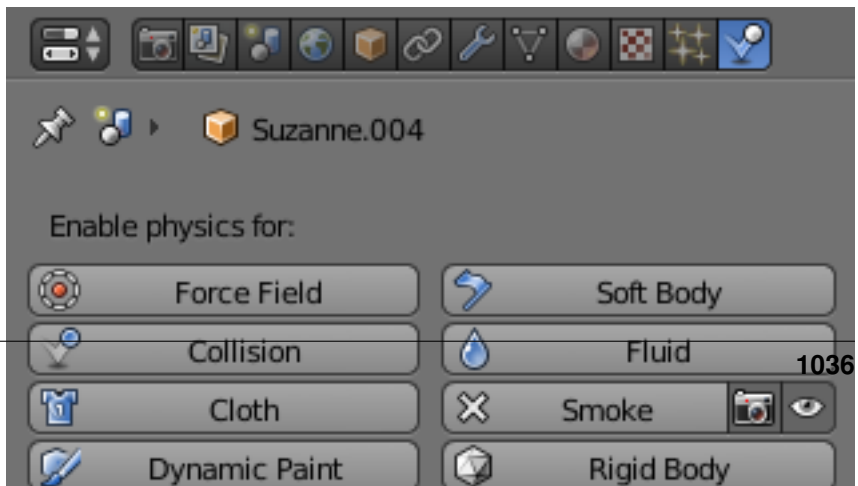
Settings

Flow Type

Fire Emit only fire. Note that the domain will automatically create some smoke to simulate smoke left by burnt fuel.

Smoke Emit only smoke.

Fire + Smoke Emit both fire and smoke.



Outflow Remove smoke and fire. Note that the shape of the outflow will use the object's *bounding box*.

Flow Source

Source This setting defines the method used to emit smoke and fire.

Mesh Create smoke/fire directly from the object's mesh. With this option selected there two additional settings, *Surface* and *Volume*.

Surface Maximum distance in voxels from the surface of the mesh in which smoke is created (see *voxel*). Since this setting uses voxels to determine distance, results will vary depending on the domain's resolution.

Volume Amount of smoke to emit inside the emitter mesh, where 0 is none and 1 is Note that emitting smoke based on volume may have unpredictable results if your mesh is *non-manifold*.

Particle System Create smoke/fire from a particle system on the flow object. Note that only *Emitter* type particle systems can add smoke. See *Particles* for information on how to create a particle system.

With this option selected there is a box to select a particle system and one addition setting, *Set Size*.

Set Size When this setting is enabled it allows the *Size* setting to define the maximum distance in voxels at which particles can emit smoke, similar to the **Surface** setting for mesh sources.

When disabled, particles will fill the nearest *voxel* with smoke.

Initial Velocity When enabled, smoke will inherit the momentum of the flow source.

Source Multiplier for inherited velocity. A value of 1 will emit smoke moving at the same speed as the source.

Normal When using a mesh source, this option controls how much velocity smoke is given along the source's *normal*.

Initial Values

Smoke Color Color of emitted smoke. When smoke of different colors are mixed they will blend together, eventually settling into a new combined color.

Flame Rate: Amount of "fuel" being burned per second. Larger values result in larger flames, smaller values result in smaller flames:

Absolute Density Maximum density of smoke allowed within range of the source.

Density Amount of smoke to emit at once.

Temp. Diff. Difference between the temperature of emitted smoke and the domain's ambient temperature. This setting's effect on smoke depends on the *domain's *Temp. Diff.* setting*.

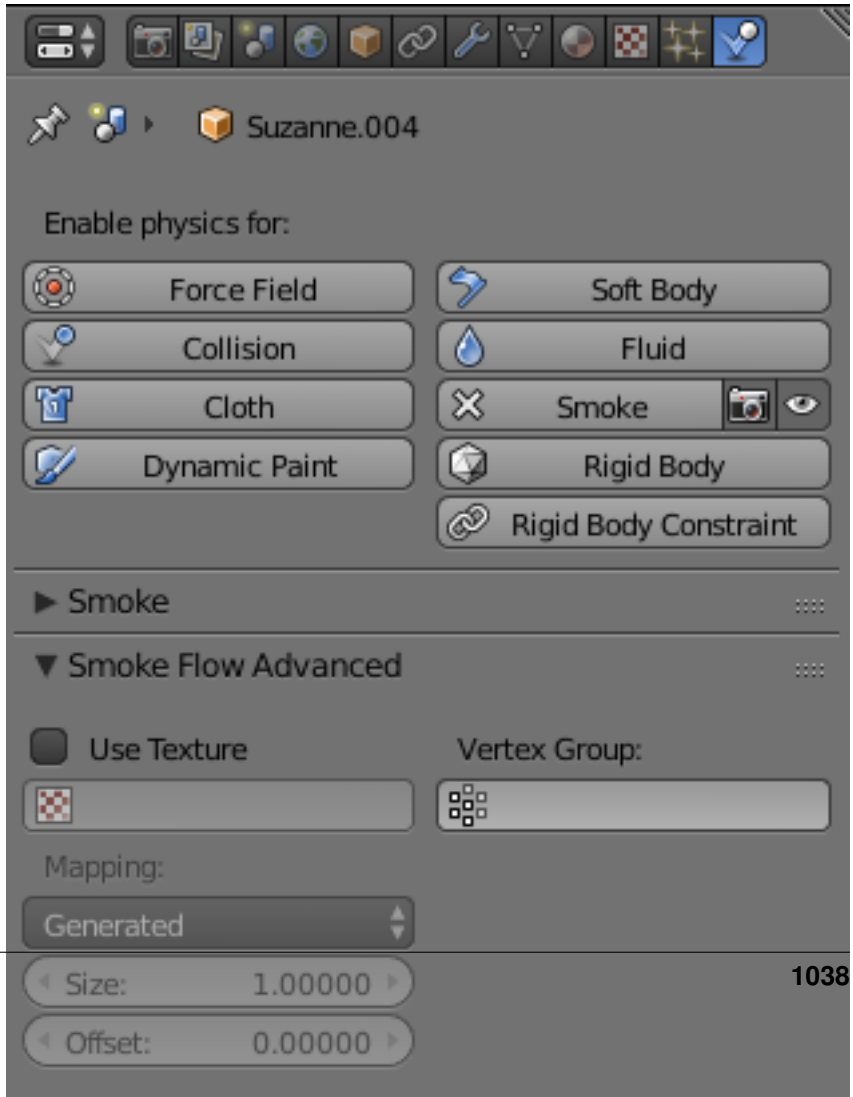
Sampling Number of sub-frames used to reduce gaps in emission of smoke from fast-moving sources.

Smoke Flow Advanced

When using a mesh as the *Flow Source*, you can use these settings to control where on the mesh smoke can be emitted from. These settings have no effect on outflow objects.

Use Texture When enabled, use the specified texture to control where smoke is emitted.

Vertex Group When set, use the specified *Vertex Group* to control where smoke is emitted.



Example

These settings are useful for effects like this:

Collisions

Smoke Collision objects are used to deflect smoke and influence airflow.

To define any mesh object as a *Smoke Collision* object, add smoke physics by clicking *Smoke* in *Properties* → *Physics*. Then select *Collision* as the *Smoke Type*.
Collision type

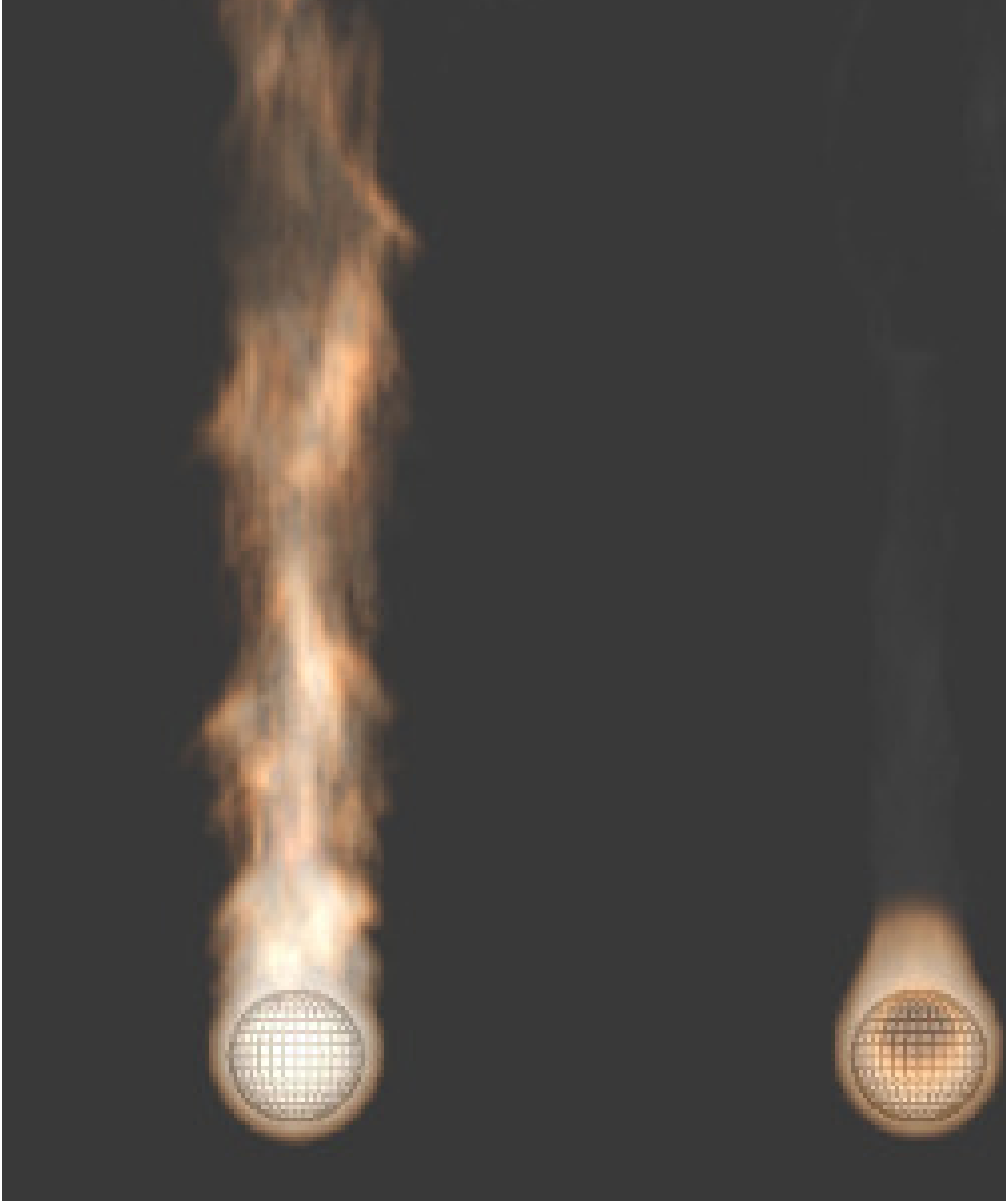


Fig. 2.1395: Example showing two fire sources. The object on the left has a *Flame Rate* of 5, while the one on the right has 0.3.

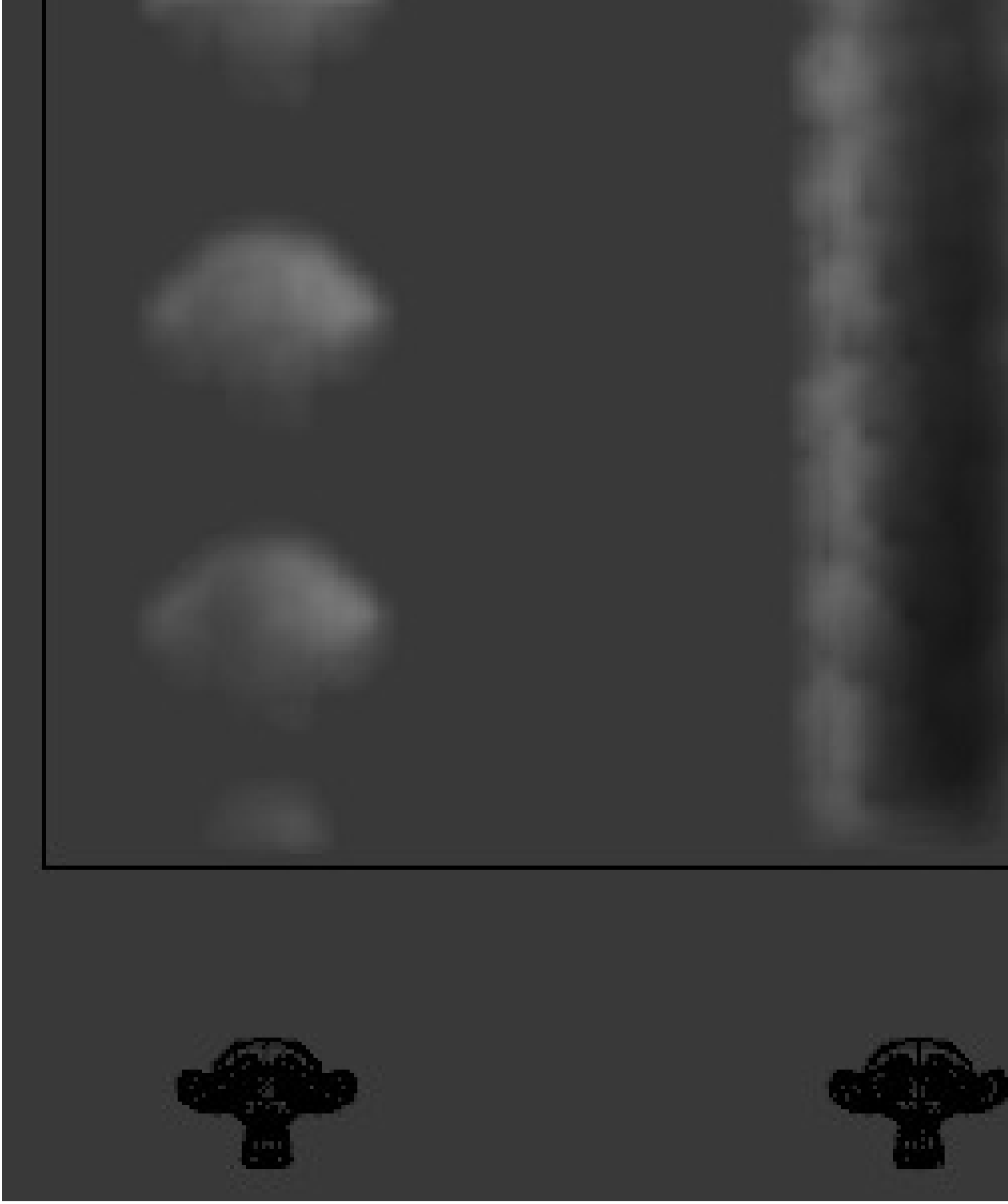


Fig. 2.1396: Example showing two fast moving sources. The object on the left uses 0 subframes, while the one on the right uses 6.

Static Simple collision model which can be calculated quickly, but may be inaccurate for moving objects.

Animated More complex collision model which takes into account impulse imparted to smoke when the collider is moving. Calculations are slower, but more accurate for moving objects.

Rigid Identical to *Static* (unfinished code).

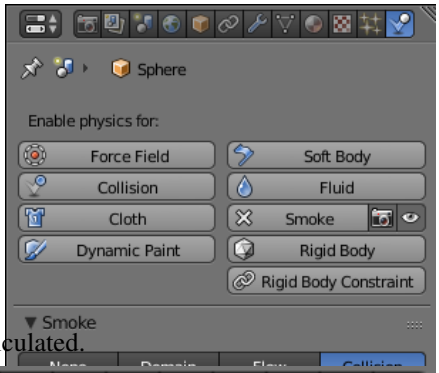
Forces

Force Fields (such as wind or vortex) are supported, like most physics systems. The influence individual force types have *can be controlled per domain object*.

Smoke Material

Blender has multiple render engines each with its own method of rendering smoke-data:

- *Blender Internal*
- *Cycles Render*



Baking Smoke Simulations

Baking is used to store the outcome of a simulation so it does not need to be recalculated.

Smoke baking settings are in *Properties* → *Physics* → *Smoke* → *Smoke Cache*. Unlike most physics simulations smoke physics has some settings that are specific to smoke.

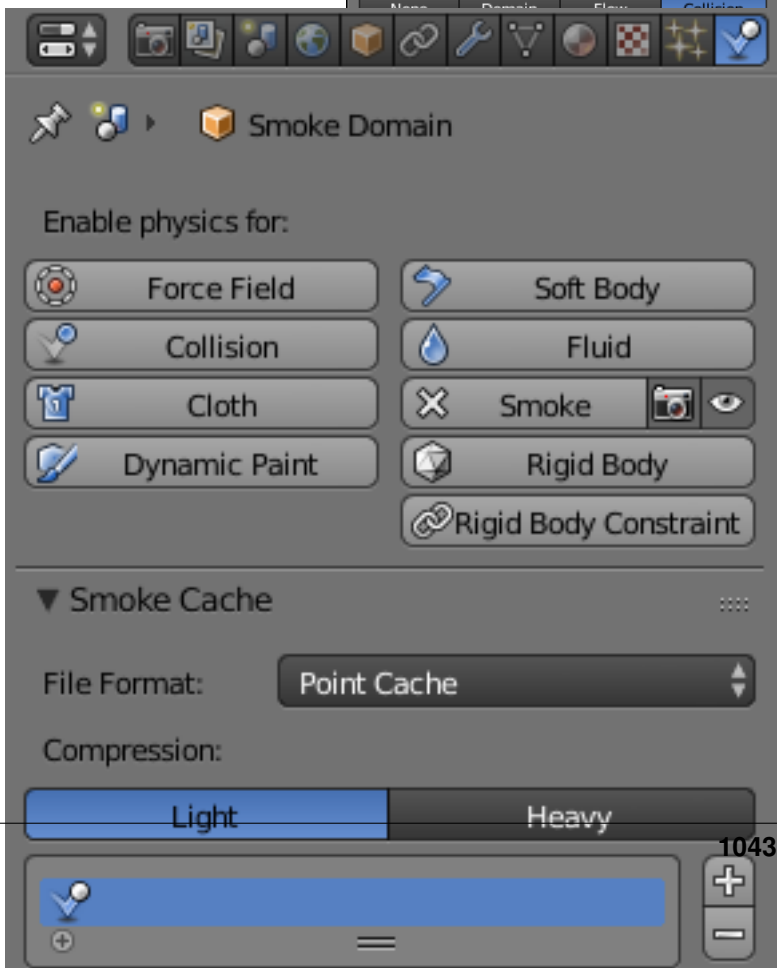
File Format File format that the cache data is to be stored.

Point Cache Blender's own caching format.

OpenVDB Advanced and efficient storage method developed by DreamWorks Animation.

The simulation fields can be found in the `.vdb` files under the following names:

- "color"
- "density"
- "heat"
- "heat old" (the temperate at the previous fame)
- "flame"
- "fuel"
- "react" (reaction coordinates, used for fire)



- "velocity"
- "shadow" (the shadows of the volume computed for viewport rendering)
- "texture coordinates" (used for turbulence)

Compression Method of data compression.

Zip Efficient but slower compression method.

Blosc Multi-threaded compression with about the same quality and size as `Zip`.

None Do not compress the data.

Data Depth Bit depth for writing all scalar (including vectors), lower values reduce the file size of the cache.

Float (Half) Half float (16 bit data). Gives less data with the benefit of smaller file sizes.

Float (Full) Full float (32 bit data). Gives more data at the cost of larger file sizes.

See also:

For other options see the *General Baking* docs for more information.

Note: Baking can only been done once your blend-file is saved. If your blend-file has not been saved, the *Smoke Cache* panel will be disabled.

Rigid Body

Introduction

The rigid body simulation can be used to simulate the motion of solid objects. It affects the position and orientation of objects and does not deform them.

Unlike the other simulations in Blender, the rigid body sim works closer with the animation system. This means that rigid bodies can be used like regular objects and be part of parent-child relationships, animation constraints and drivers.

Creating a Rigid Body

Creating the Rigid Body.

Right now only mesh objects can participate in the rigid body simulation. To create rigid bodies, either click on *Rigid Body* button in the *Physics* tab of the Properties editor or use the *Add Active*/*Add Passive* buttons in the *Physics* tab of the *Tool Shelf*.

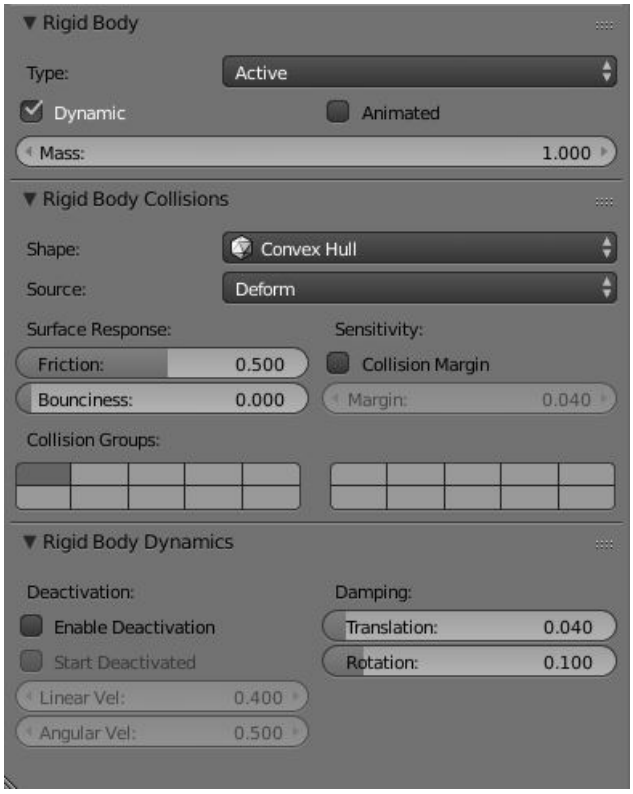


Fig. 2.1399: Default rigid body panel.

There are two types of rigid body: active and passive. *Active* bodies are dynamically simulated, while *passive* bodies remain static. Both types can be driven by the animation system when using the *Animated* option.

During the simulation, the rigid body system will override the position and orientation of dynamic rigid body objects. Note however, that the location and rotation of the objects is not changed, so the rigid body simulation acts similar to a constraint. To apply the rigid body transformations you can use the *Apply Transformation* button in the *Physics* tab of the *Tool Shelf*.

The scale of the rigid body object also influences the simulation, but is always controlled by the animation system.

Rigid Body physics on the object can be *removed* with the *Rigid Body* button in the *Physics* tab or *Remove* button in the *Physics* tab of the *Tool Shelf*.

Rigid Body Properties

Type Role of the rigid body in the simulation. Active objects can be simulated dynamically, passive object remain static.

Active Object is directly controlled by simulation results. The possibility to select this type also available with *Add Active* button in the *Physics* tab of the *Tool Shelf*.

Passive Object is directly controlled by animation system. Thus, this type is not available for *Rigid Body Dynamics*. The possibility to select this type also available with *Add Passive* button in the *Physics* tab of the *Tool Shelf*.

Dynamic Enables/disables rigid body simulation for object.

Animated Allows the rigid body additionally to be controlled by the animation system.

Mass Specifies how heavy the object is and "weights" irrespective of gravity. There are predefined mass preset available with the *Calculate Mass* button in the *Physics* tab of the *Tool Shelf*.

Calculate Mass Automatically calculate mass values for Rigid Body Objects based on volume. There are many useful presets available from the menu, patching real-world objects.

Note: Also you can have *Custom* mass material type, which is achieved by setting a custom density value (kg/m^3).

Rigid Body Collisions

Rigid Body Collisions panel.

General settings

Surface Response

Friction Resistance of object to movement. Specifies how much velocity is lost when objects collide with each other.

Bounciness Tendency of object to bounce after colliding with another (0 to 1) (rigid to perfectly elastic). Specifies how much objects can bounce after collisions.

Collision Groups Allows rigid body collisions allocate on different groups (maximum 20).

Collision shapes

The *Shape* option determines the collision shape of the object. The following Collision Shapes are available:

Primitive shapes these are best in terms of memory/performance but do not necessarily reflect the actual shape of the object. They are calculated based on the object's bounding box. The center of gravity is always in the middle for now.

Box Box-like shapes (i.e. cubes), including planes (i.e. ground planes). The size per axis is calculated from the bounding box.

Sphere Sphere-like shapes. The radius is the largest axis of the bounding box.

Capsule This points up the Z-Axis.

Cylinder This points up the Z-Axis. The height is taken from the z-axis, while the radius is the larger of the x/y-axes.

Cone This points up the Z-Axis. The height is taken from the z-axis, while the radius is the larger of the x/y-axes.

Mesh based shapes these are calculated based on the geometry of the object so they are a better representation of the object. The center of gravity for these shapes is the object origin.

Convex Hull A mesh-like surface encompassing (i.e. shrinkwrap over) all vertices (best results with fewer vertices). Convex approximation of the object, has good performance and stability.

Mesh *Mesh* consisting of triangles only, allowing for more detailed interactions than convex hulls. Allows to simulate concave objects, but is rather slow and unstable.

The changing collision shape is available also with *Change Shape* button in the *Physics* tab of the *Tool Shelf*.

Mesh source

Users can now specify the mesh *Source* for *Mesh* bases collision shapes:

Base The base mesh of the object.

Deform Includes any deformations added to the mesh (shape keys, deform modifiers).

Deforming Rigid body deforms during simulation.

Final Includes all modifiers.

Collision Margin

Margin Threshold of distance near surface where collisions are still considered (best results when non-zero).

The collision margin is used to improve performance and stability of rigid bodies. Depending on the shape, it behaves differently: some shapes embed it, while others have a visible gap around them.

The margin is *embedded* for these shapes:

- Sphere
- Box
- Capsule
- Cylinder
- Convex Hull: Only allows for uniform scale when embedded.

The margin is *not embedded* for these shapes:

- Cone
- Active Triangle Mesh
- Passive Triangle Mesh: Can be set to 0 most of the time.

Rigid Body Dynamics

Rigid Body Dynamics panel.

This panel is available only for *Active* type of rigid bodies.

Deactivation

Enable Deactivation Enable deactivation of resting rigid bodies. Allows object to be deactivated during the simulation (improves performance and stability, but can cause glitches).

Start Deactivated Starts objects deactivated. They are activated on collision with other objects.

Linear Vel Specifies the linear deactivation velocity below which the rigid body is deactivated and simulation stops simulating object.

Angular Vel Specifies the angular deactivation velocity below which the rigid body is deactivated and simulation stops simulating object.

Damping

Translation Amount of linear velocity that is lost over time.

Rotation Amount of angular velocity that is lost over time.

Rigid Body World

The *Rigid Body World* is a group of Rigid Body objects, which holds settings that apply to all rigid bodies in this simulation. These setting

can be found in *Properties Editor* → *Scene* → *Rigid Body World*.

When you add Rigid Body physics on an object, primary there is created a group of objects with default "RigidBodyWorld" name. Rigid body objects automatically are added to this group when you add Rigid Body physics for them.

You can be create several Rigid Body World groups and allocate there yours Rigid Body objects with *Groups* panel in *Object* tab.

Rigid body objects and constraints are only taken into account by the simulation if they are in the groups specified in *Group* field of the *Rigid Body World* panel in the *Scene* tab.

Rigid Body World Enable/disable evaluation of the Rigid Body simulation based on the rigid body objects participating in the specified group of Rigid Body World.

Remove Rigid Body World Remove Rigid Body simulation from the current scene.

Group Containing rigid body objects participating in this simulation.

Constraints Containing rigid body object constraints participating in the simulation.

Simulation quality and timing settings:

Speed Can be used to speed up/slow down the simulation.

Split Impulse Enable/disable reducing extra velocity that can build up when objects collide (lowers simulation stability a little so use only when necessary). Limits the force with which objects are separated on collision, generally produces nicer results, but makes the simulation less stable (especially when stacking many objects).

Steps Per Second Number of simulation steps made per second (higher values are more accurate but slower). This only influences the accuracy and not the speed of the simulation.

Solver Iterations Amount of constraint solver iterations made per simulation step (higher values are more accurate but slower). Increasing this makes constraints and object stacking more stable.

Rigid Body Cache

The *Rigid Body Cache* panel specifies the frame range in which the simulation is active. Can be used to bake the simulation.

Start/End First and last frame of the simulation.

Bake Calculates the simulation and protects the cache. You need to be in *Object Mode* to bake.

Free Bake Active after the baking of simulation. Clears the baked cache.

Calculate to Frame Bake physics to current frame.

Current Cache to Bake Bake from Cache.

Bake All Dynamics Bake all physics.

Free All Bakes Free all baked caches of all objects in the current scene.

Update All To Frame Update cache to current frame.

If you haven't saved the blend-file, the cache is created in memory, so save your file first or the cache may be lost.

Rigid Body Field Weights

As other physics dynamics systems, Rigid Body simulation are also influenced by external force effectors.

Rigid Body Constraints

Introduction

Constraints (also known as joints) for rigid bodies connect two rigid bodies.

The physics constraints available in the non-game modes are meant to be attached to an *Empty* object. The constraint then has fields which can be pointed at the two physics-enabled object which will be bound by the constraint. The *Empty* object provides a location and axis for the constraint distinct from the two constrained objects. The location of the entity hosting the physics constraint marks a location and set of axes on each of the two constrained objects. These two anchor points are calculated at the beginning of the animation and their position and orientation remain fixed in the *local* coordinate system of the object for the duration of the animation. The objects can move far from the constraint object, but the constraint anchor moves with the object. If this feature seems limiting, consider using multiple objects with a non-physics *Child-of* constraint and animate the relative location of the child.

The quickest way to constrain two objects is to select both and click the *Connect* button in the *Physics* tab of the *Tool Shelf*. This creates a new *Empty* object (named "Constraint") with

a physics constraint already attached and pointing at the two selected objects.

Also you can create *Rigid Body Constaint* on of the two constrained objects with *Rigid Body Constaint* button of the *Physics* tab in the Properties editor. This constraint is dependent on the object location and rotation on which it was created. This way, there are no *Empty* object created for the constraint. The role of the *Empty* object is put on this object. The constrained object can be then set as *Passive* type for better driving the constrain.

Additional parameters appear in the *Rigid Body Constaint* panel of the *Physics* tab in the Properties editor for the selected *Empty* object or the one of the two constrained objects with the created constraint.

Common Options

Rigid Body Constraint panel.

Enabled Specifies whether the constraint is active during the simulation.

Disable Collisions Allows constrained objects to pass through one another.

Object 1 First object to be constrained.

Object 2 Second object to be constrained.

Breakable Allows constraint to break during simulation. Disabled for the *Motor* constraint.

Threshold Impulse strength that needs to be reached before constraint breaks.

Override Iterations Allows to make constraints stronger (more iterations) or weaker (less iterations) than specified in the rigid body world.

Iterations Number of constraint solver iterations made per simulation step for this constraint.

Limits By using limits you can constrain objects even more by specifying a translation/rotation range on/around respectively axis (see below for each one individually). To lock one axis, set both limits to 0.

Types

Fixed

This constraint cause the two objects to move as one. Since the physics system does have a tiny bit of slop in it, the objects do not move

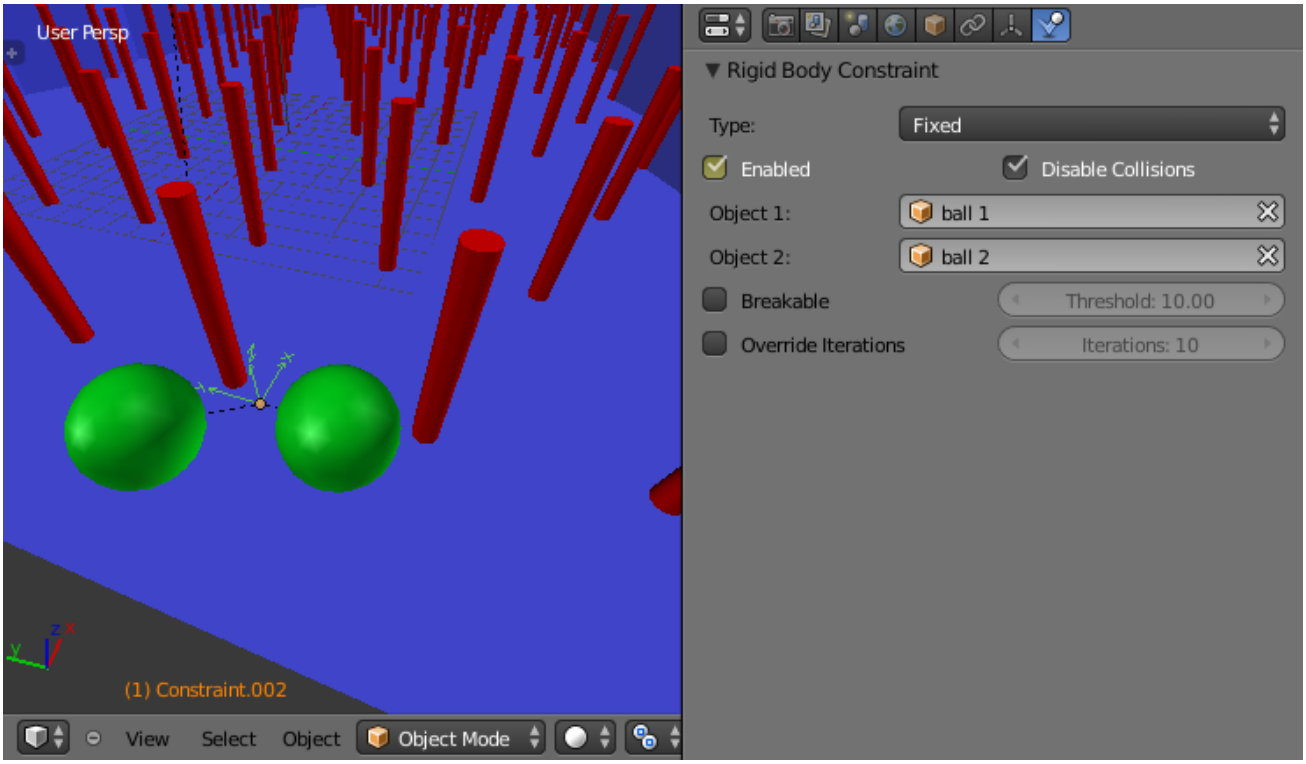


Fig. 2.1400: Options available to a *Fixed* constraint.

as rigidly as they would if they were part of the same mesh.

Point

The objects are linked by a point bearing allowing any kind of rotation around the location of the constraint object, but no relative translation is permitted. The physics engine will do its best to make sure that the two points designated by the constraint object on the two constrained objects are coincident.

Hinge

The hinge permits 1 degree of freedom between two objects. Translation is completely constrained. Rotation is permitted about the Z axis of the object hosting the Physics constraint (usually an *Empty*, distinct from the two objects that are being linked). Adjusting the position and rotation of the object hosting the constraint allows you to control the anchor and axis of the hinge.

The Hinge is the only 1-axis rotational constraint that uses the Z axis instead of the X axis. If something is wrong with your hinge, check

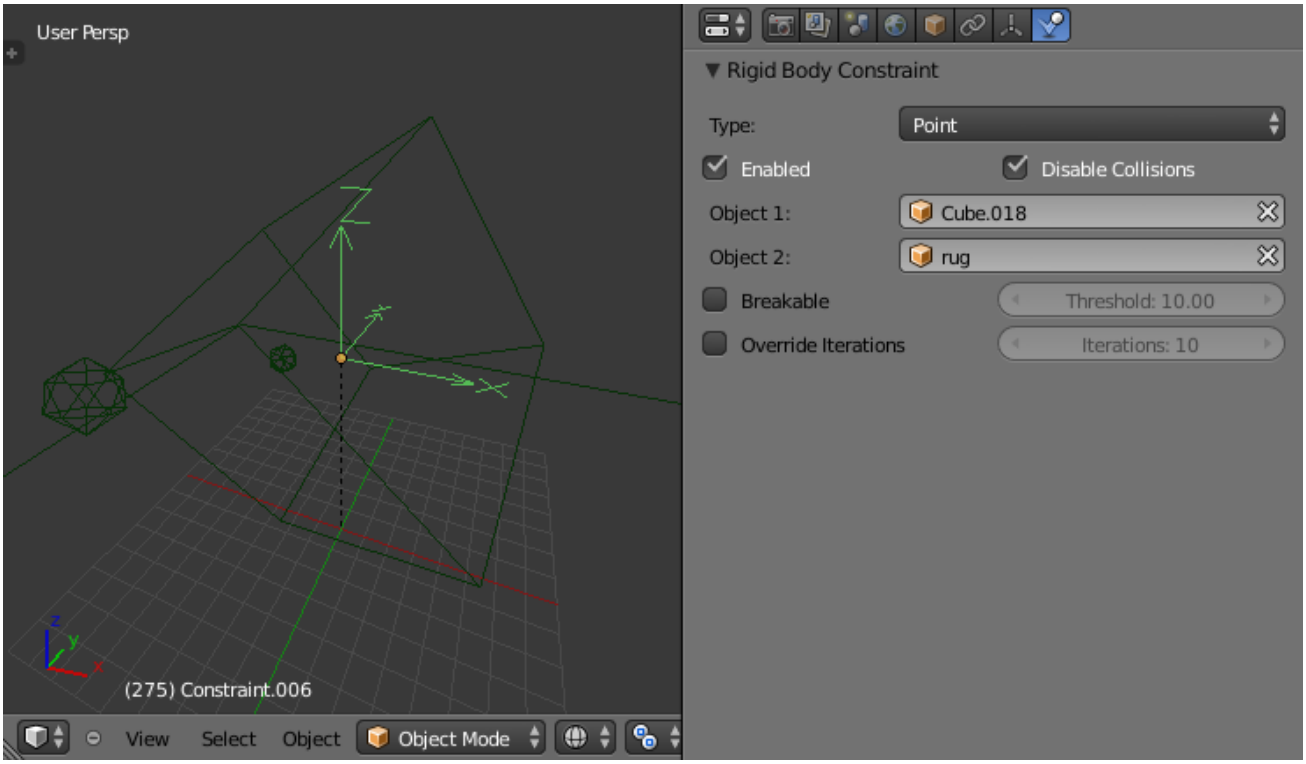


Fig. 2.1401: Options available to a *Point* constraint.

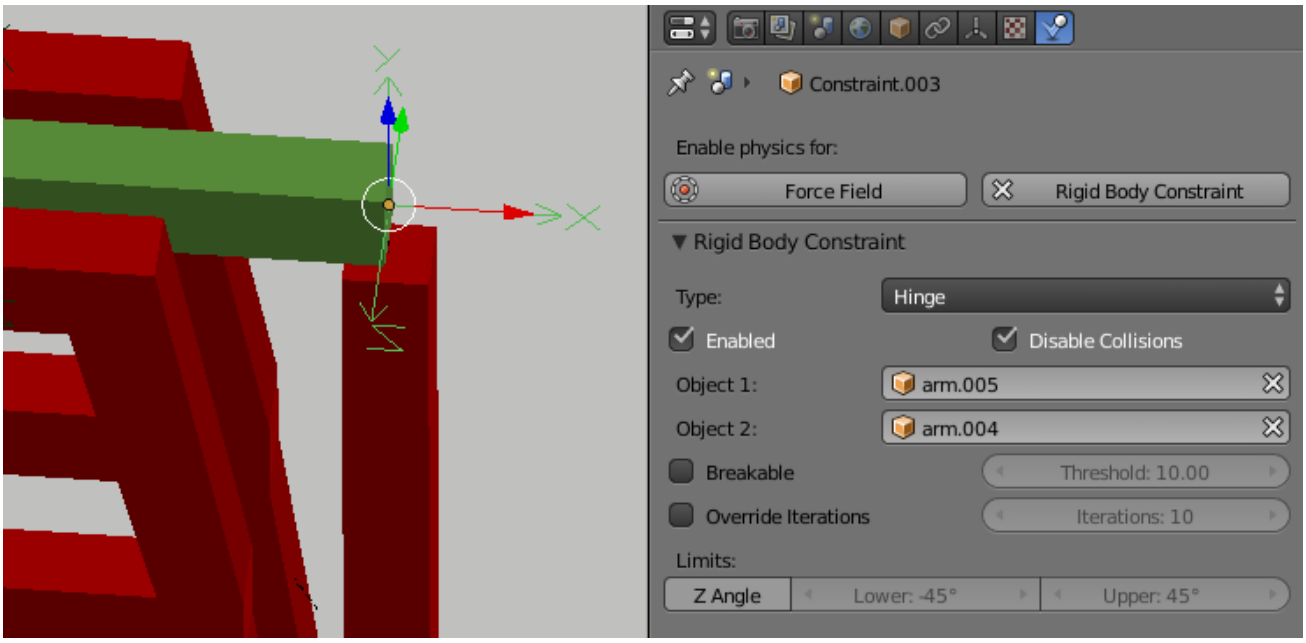


Fig. 2.1402: Options available to a *Hinge* constraint.

your other constraints to see if this might be the problem.

Limits

Z Angle Enables/disables limit rotation around Z axis.

Lower Lower limit of Z axis rotation.

Upper Upper limit of Z axis rotation.

Slider

The Slider constraint allows relative translation along the X axis of the constraint object, but permits no relative rotation, or relative translation along other axes.

Limits

X Axis Enables/disables limit translation around X axis.

Lower Lower limit of X axis translation.

Upper Upper limit of X axis translation.

Piston

A piston permits translation along the X axis of the constraint object. It also allows rotation around the X axis of the constraint object. It is like a combination of the freedoms of a slider with the freedoms of a hinge (neither of which is very free alone).

Limits

X Axis Enables/disables limit translation around X axis.

Lower Lower limit of X axis translation.

Upper Upper limit of X axis translation.

X Angle Enables/disables limit rotation around X axis.

Lower Lower limit of X axis rotation.

Upper Upper limit of X axis rotation.

Generic

The generic constraint has a lot of available parameters.

The X, Y, and Z axis constraints can be used to limit the amount of translation between the objects. Clamping the min/max to zero has the same effect as the Point constraint.

Clamping the relative rotation to zero keeps the objects in alignment. Combining an absolute rotation and translation clamp would behave much like the Fixed constraint.

Using a non-zero spread on any parameter allows it to rattle around in that range throughout the course of the simulation.

Limits

X Axis/Y Axis/Z axis Enables/disables limit translation on X, Y or Z axis respectively.

Lower Lower limit of translation for X, Y or Z axis respectively.

Upper Upper limit of translation for X, Y or Z axis respectively.

X Angle/Y Angle/Z Angle Enables/disables limit rotation around X, Y or Z axis respectively.

Lower Lower limit of rotation for X, Y or Z axis respectively.

Upper Upper limit of rotation for X, Y or Z axis respectively.

Generic Spring

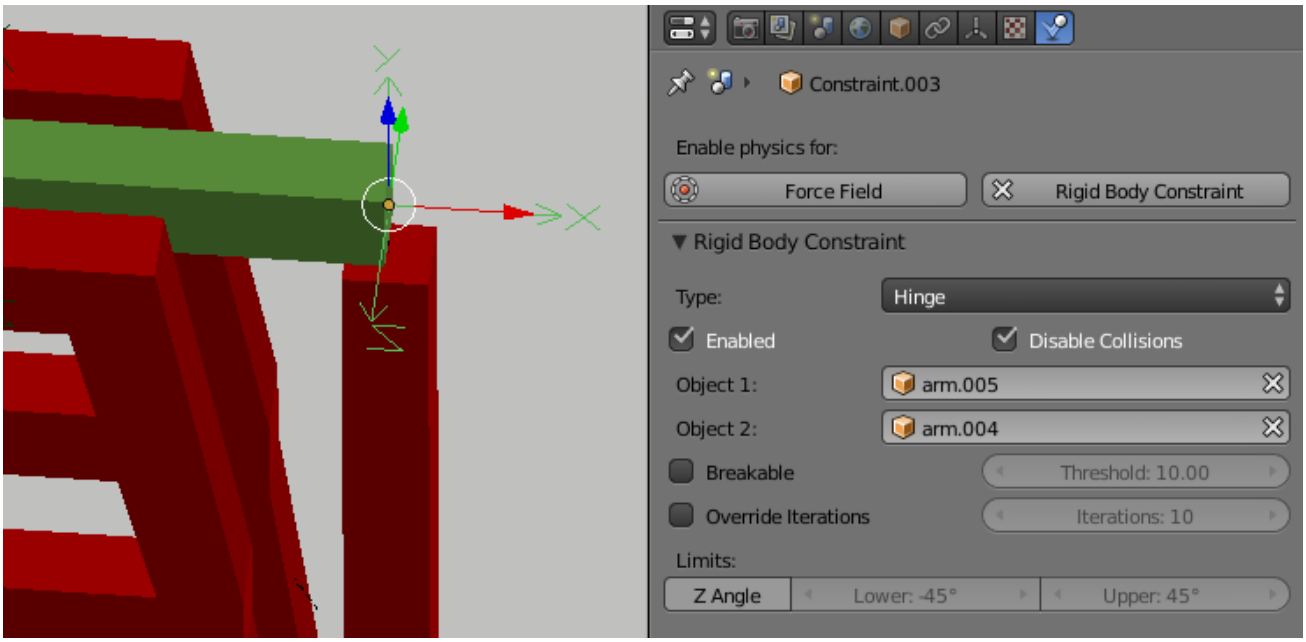


Fig. 2.1403: Options available to a *Generic Spring* constraint.

The generic spring constraint adds some spring parameters for the X/Y/Z axes to all the options available on the Generic constraint. Using the spring alone allows the objects to bounce around as if attached with a spring anchored at the constraint object. This is usually a little too

much freedom, so most applications will benefit from enabling translation or rotation constraints.

If the damping on the springs is set to 1, then the spring forces are prevented from realigning the anchor points, leading to strange behavior. If your springs are acting weird, check the damping.

Limits

X Axis/Y Axis/Z axis Enables/disables limit translation on X, Y or Z axis respectively.

Lower Lower limit of translation for X, Y or Z axis respectively.

Upper Upper limit of translation for X, Y or Z axis respectively.

X Angle/Y Angle/Z Angle Enables/disables limit rotation around X, Y or Z axis respectively.

Lower Lower limit of rotation for X, Y or Z axis respectively.

Upper Upper limit of rotation for X, Y or Z axis respectively.

Springs

X/Y/Z Enables/disables springs on X, Y or Z axis respectively.

Stiffness Spring Stiffness on X, Y or Z axis respectively. Specifies how "bendy" the spring is.

Damping Spring Damping on X, Y or Z axis respectively. Amount of damping the spring has.

Motor

The motor constraint causes translation and/or rotation between two entities. It can drive two objects apart or together. It can drive simple rotation, or rotation and translation (although it will not be constrained like a screw since the translation can be blocked by other physics without preventing rotation).

The rotation axis is the X axis of the object hosting the constraint. This is in contrast with the Hinge which uses the Z axis. Since the Motor is vulnerable to confusing perturbations without a matching Hinge constraint, special care must be taken to align the axes. Without proper alignment, the motor will appear to have no effect (because the hinge is preventing the motion of the motor).

Linear motor/Angular motor

Enable Enable linear or angular motor respectively.

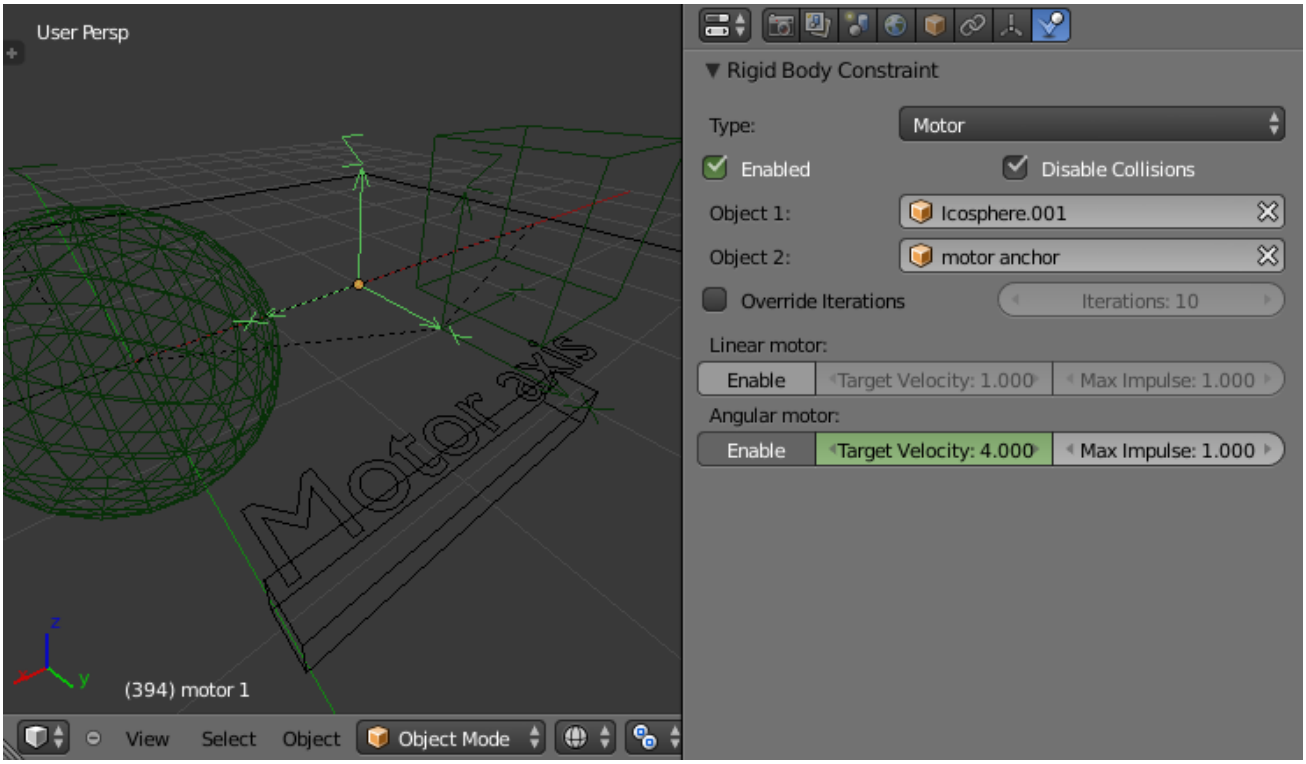


Fig. 2.1404: Options available to a *Motor* constraint.

Target Velocity Target linear or angular motor velocity respectively.

Max Impulse Maximum linear or angular motor impulse respectively.

Tips

As with all physics-enabled objects, pay close attention to the *Animated* check box in the *Rigid Body* panel of the *Physics* tab in the Properties editor. A common mistake is to use keyframe animation on a *Passive* physics object without checking the *Animated* box. The object will move, but the physics engine will behave as if the *Passive* is still in its starting place, leading to disappointment.

Animation

The most common trick is to *keyframe* animate the location or rotation of an *Active* physics object as well as the *Animated* checkbox. When the curve on the *Animated* property switches to disabled, the physics engine takes over using the object's last known location, rotation and velocities.

Animating the strengths of various other parameters (a *Motor's* Target Velocity, a *Hinge's* lim-

its, etc) can be used to accomplish a wide variety of interesting results.

Enabling a constraint during the physics simulation often has dramatic results as the physics engine tries to bring into alignment two objects which are often dramatically out of alignment. It is very common for the affected objects to build up enough kinetic energy to bounce themselves out of camera (and into orbit, although the physics engine is not yet capable of simulating a planet's gravity well, so scratch that).

Rigid Body dynamics can be baking to normal keyframes with *Bake To Keyframes* button in the *Physics* tab of the *Tool Shelf*.

Simulation Stability

The simplest way of improving simulation stability is to increase the steps per second. However, care has to be taken since making too many steps can cause problems and make the simulation even less stable (if you need more than 1000 steps, you should look at other ways to improve stability).

Increasing the number of solver iterations helps making constraints stronger and also improves object stacking stability.

It is best to avoid small objects, as they are currently unstable. Ideally, objects should be at least 20 cm in diameter. If it is still necessary, setting the collision margin to 0, while generally not recommended, can help making small object behave more naturally.

When objects are small and/or move very fast, they can pass through each other. Besides what is mentioned above it's also good to avoid using mesh shapes in this case. Mesh shapes consist of individual triangles and therefore do not really have any thickness, so objects can pass through more easily. You can give them some thickness by increasing the collision margin.

Combining Rigid Bodies with Other Simulations

Since the rigid body simulation is part of the animation system, it can influence other simulations just like the animation system can.

In order for this to work, the rigid body object needs to have a collision modifier. Simply click on *Collision* in the *Physics* tab.

Scaling Rigid Bodies

Rigid body objects can be scaled, also during the simulation. This work well in most cases, but can sometimes cause problems.

If dynamic scaling is not needed, rigid body objects should have the scale applied by using the *Apply Scale* command `Ctrl-A`.

Particles

Introduction

Particles are lots of items emitted from mesh objects, typically in the thousands. Each particle can be a point of light or a mesh, and be joined or dynamic. They may react to many different influences and forces, and have the notion of a lifespan. Dynamic particles can represent fire, smoke, mist, and other things such as dust or magic spells.

Hair type particles are a subset of regular particles. Hair systems form strands that can represent hair, fur, grass and bristles.

You see particles as a *Particle* modifier, but all settings are done in the *Particle tab*.

Particles generally flow out from their mesh into space. Their movement can be affected by many things, including:

- Initial velocity out from the mesh.
- Movement of the emitter (vertex, face or object) itself.
- Movement according to "gravity" or "air resistance".
- Influence of force fields like wind, vortexes or guided along a curve.
- Interaction with other objects like collisions.
- Partially intelligent members of a flock (herd, school, ...), that react to other members of their flock, while trying to reach a target or avoid predators.
- Smooth motion with softbody physics (only *Hair* particle systems).
- Or even manual transformation with *Lattices*.

  Particles may be rendered as:

  - *Halos* (for Flames, Smoke, Clouds).
- Meshes which in turn may be animated (e.g. fish, bees, ...). In these cases, each particle "carries" another object.

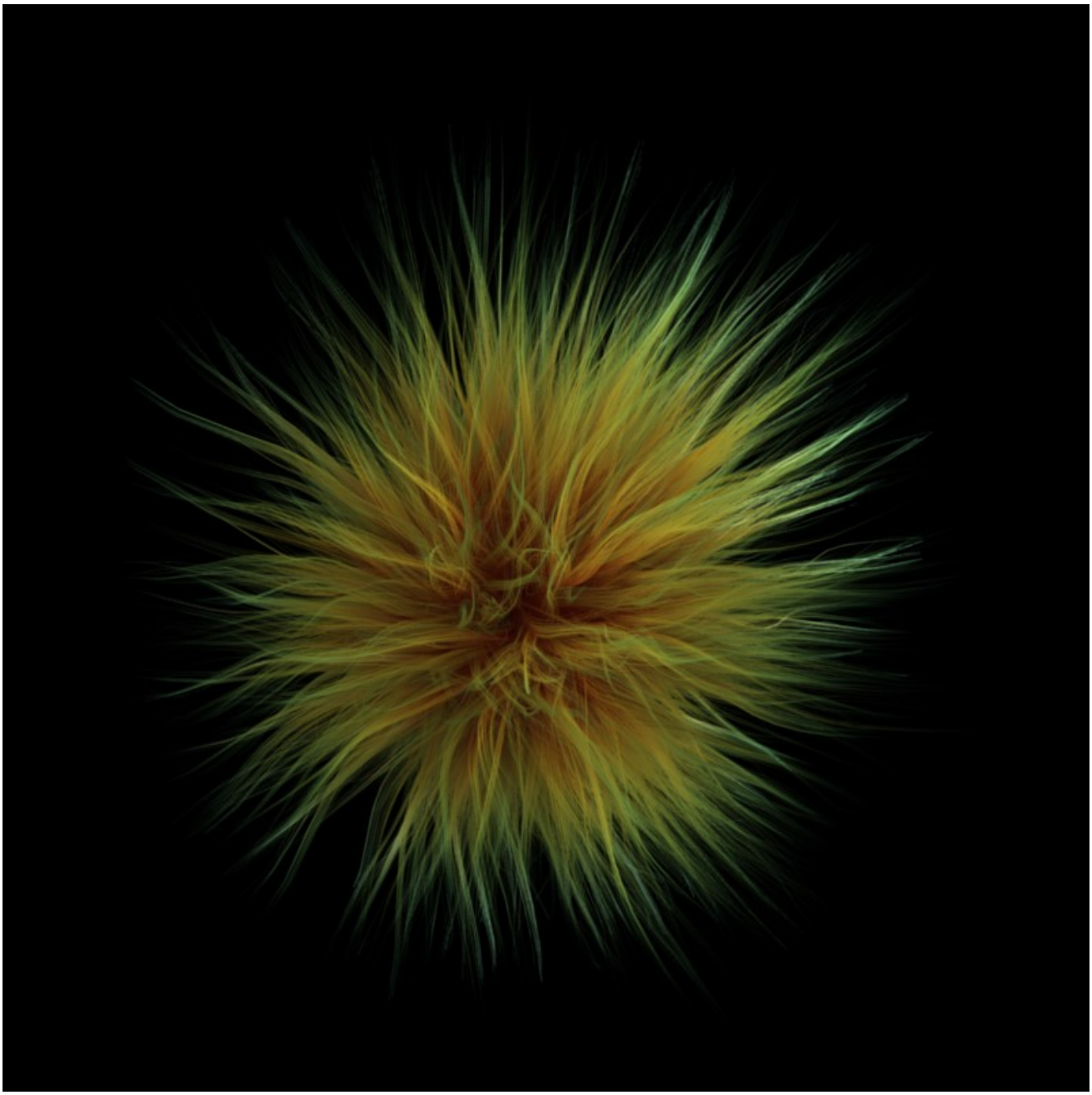


Fig. 2.1405: Some fur made from particles.

- *Strands* (for *Hair, Fur, Grass*); the complete way of a particle will be shown as a strand. These strands can be manipulated in the 3D View (combing, adding, cutting, moving, etc).

  Every object may carry many particle systems. Each particle system may contain up to 100.000 particles. Certain particle types (*Hair* and *Keyed*) may have up to 10.000 children for each particle (children move and emit more or less like their respective parents). The size of your memory and your patience are your practical boundaries.

Options

Each system has the same basic sets of controls, but options within those sets vary based on the system employed. These sets of controls are:

Particle System Panel Basic Settings.

Emission Settings for the initial distribution of particles on the emitter and the way they are born into the scene.

Cache In order to increase realtime response and avoid unnecessary recalculation of particles, the particle data can be cached in memory or stored on disk.

Velocity Initial speed of particles.

Rotation Rotational behavior of particles.

Physics How the movement of the particles behaves.

Render Rendering options.

Display Realtime display in the 3D View.

Children Control the creation of additional child particles.

Field Weights Factors for external forces.

Force Field Settings Makes particles force fields.

Vertex Groups Influencing various settings with vertex groups.

Particle System Panel

Active Particle System The *List View* of the objects Particle Modifier(s).

Particle Settings The *Data-Block menu* for settings.

Type Main selector of the system type.

Emitter In such a system particles are *emitted* from the object.

Hair The *Hair* type is rendered as strands.

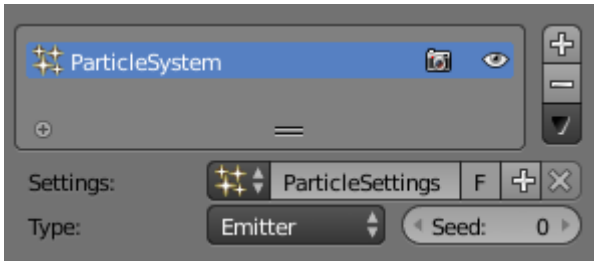


Fig. 2.1406: Particle System Panel.

Seed This initial value for random properties can be used to create a look, which is slightly different, even when using the same settings.

Workflow

The process for working with standard particles is:

- Create the mesh which will emit the particles.
- Create one or more Particle Systems to emit from the mesh. Many times, multiple particle systems interact or merge with each other to achieve the overall desired effect.
- Tailor each Particle System's settings to achieve the desired effect.
- Animate the base mesh and other particle meshes involved in the scene.
- Define and shape the path and flow of the particles.
- For *Hair* particle systems: Sculpt the emitter's flow (cut the hair to length and comb it for example).
- Make final render and do physics simulation(s), and tweak as needed.

Creating a Particle System

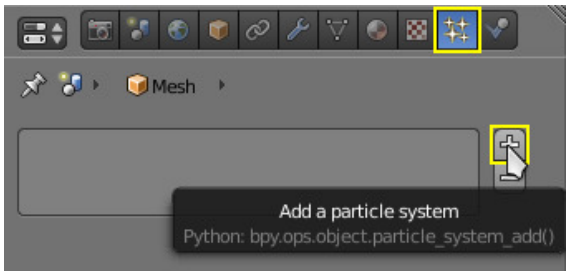


Fig. 2.1407: Adding a particle system.

To add a new particle system to an object, go to the *Particles* tab of the Properties editor and

click the small + button. An object can have many Particle Systems.

Each particle system has separate settings attached to it. These settings can be shared among different particle systems, so one does not have to copy every setting manually and can use the same effect on multiple objects.

Types of Particle systems

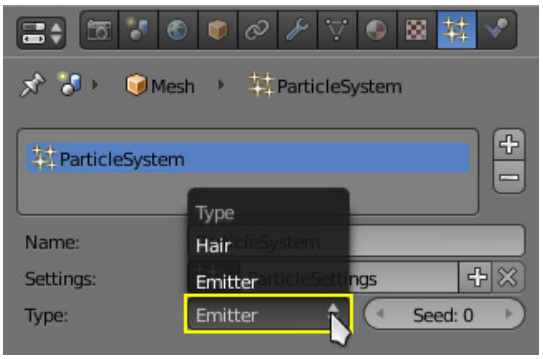


Fig. 2.1408: Particle System Types.

After you have created a particle system, the Properties editor fills with many panels and buttons. But do not panic! There are two different types of particle systems, and you can change between these two with the *Type* selector: Emitter and Hair.

The settings in the *Particle System* tab are partially different for each system type. For example, in *Particle System Types.* they are shown for only system type *Emitter*.

Properties

Emission

The *Emitter* system works just like its name says: it emits/produces particles for a certain amount of time. In such a system, particles are emitted from the selected object from the *Start* frame to the *End* frame and have a certain lifespan. These particles are rendered default as *Halos*, but you may also render these kind of particles as objects (depending on the particle system's render settings, see *Visualization*).

Options

The buttons in the *Emission* panel control the way particles are emitted over time:

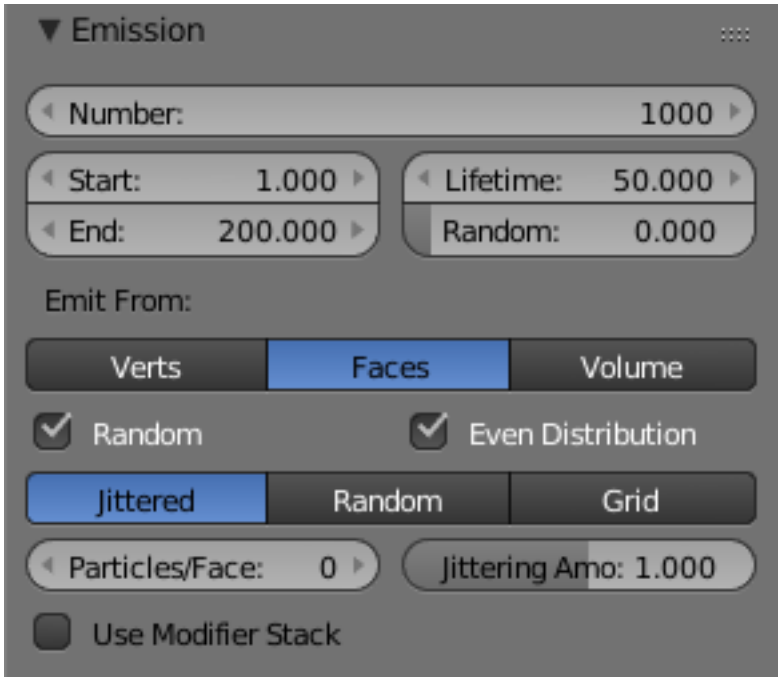


Fig. 2.1409: Particle Emission Settings.

Amount The maximum amount of parent particles used in the simulation.

Start The start frame of particle emission. You may set negative values, which enables you to start the simulation before the actual rendering.

End The end frame of particle emission.

Lifetime The lifespan (in frames) of the particles.

Random A random variation of the lifetime of a given particle. The shortest possible lifetime is *Lifetime* × (1 - *Rand*). Values above 1.0 are not allowed. For example with the default *Lifetime* value of 50 a *Random* setting of 0.5 will give you particles with lives ranging from 50 frames to $50(1.0 - 0.5) = 25$ frames, and with a *Random* setting of 0.75 you will get particles with lives ranging from 50 frames to $50(1.0 - 0.75) = 12.5$ frames.

Emission Location

Emit From parameters define how and where the particles are emitted, giving precise control over their distribution. You may use vertex groups to confine the emission, that is done in the *Vertex Groups* panel.

Vertices Emit particles from the vertices of a mesh.

Faces Emit particles from the surface of a mesh's faces.

Volume Emit particles from the volume of an enclosed mesh.

Distribution Settings

These settings control how the emissions of particles are distributed throughout the emission locations.

Random The emitter element indices are gone through in a random order instead of linearly (one after the other).

For Faces and Volume, additional options appear:

Even Distribution Particle distribution is made even based on surface area of the elements, i.e. small elements emit less particles than large elements, so that the particle density is even.

Jittered Particles are placed at jittered intervals on the emitter elements.

Particles/Face Number of emissions per face (0 = automatic).

Jittering Amount Amount of jitter applied to the sampling.

Random Particles are emitted from random locations in the emitter's elements.

Grid Particles are set in a 3D grid and particles near/in the elements are kept.

Invert Grid Invert what is considered the object and what is not.

Hexagonal Uses a hexagonal shaped grid instead of a rectangular one.

Resolution Resolution of the grid.

Random Add a random offset to grid locations.

Tip: Your mesh must be *manifold* to emit particles from the volume.

Some modifiers like *Edge Split* break up the surface, in which case volume emission will not work correctly!

Use Modifier Stack Take any *Modifiers* above the particle modifier in the *Modifier Stack* into account when emitting particles.

Note: Note that particles may differ in the final render if these modifiers generate different geometry between the viewport and render.

Cache

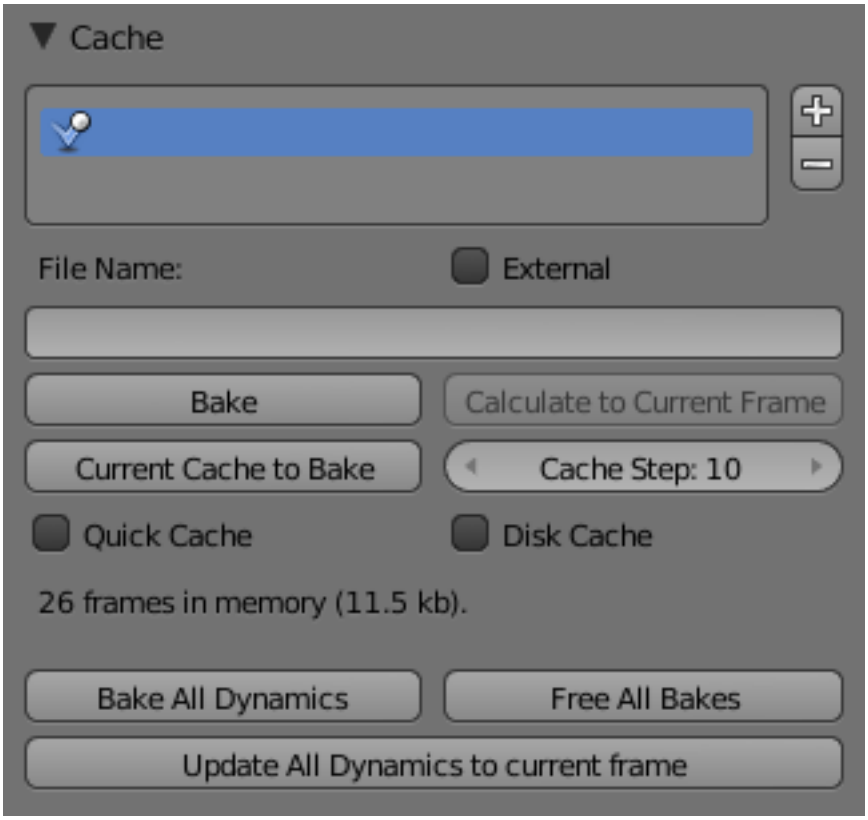


Fig. 2.1410: Particles Cache Settings.

Emitter systems use a unified system for caching and baking (together with softbody and cloth). The results of the simulation are automatically cached to disk when the animation is played, so that the next time it runs, it can play again quickly by reading in the results from the disk. If you *Bake* the simulation the cache is protected and you will be asked when you are trying to change a setting that will make a recalculating necessary.

Tip: Beware of the *Start* and *End* Settings

The simulation is only calculated for the positive frames in-between the *Start* and *End* frames of the *Bake* panel, whether you bake or not. So if you want a simulation longer than 250 frames you have to change the *End* frame!

Caching

- As animation is played, each physics system writes each frame to disk, between the simulation start and end frames. These files are stored in folders with prefix `blendcache`, next to

the blend-file. Note that for the cache to fill up, one has to start playback before or on the frame that the simulation starts.

- The cache is cleared automatically on changes. But not on all changes, so it may be necessary to free it manually e.g. if you change a force field.
- If it is impossible to write in the subdirectory there will be no caching.
- The cache can be freed per physics system with a button in the panels, or with the `Ctrl-B` shortcut key to free it for all selected objects.
- If the file path to the cache is longer than what is possible with your operating system (more than 250 characters for example), strange things might happen.

Baking

- The system is protected against changes after baking.
- The *Bake* result is cleared also with `Ctrl-B` for all selected objects or click on *Free Bake* for a singular particle system.
- If the mesh changes the simulation is not calculated anew.
- Sorry: no bake editing for particles like for soft-bodies and clothes.

Two notes at the end:

- For renderfarms, it is best to bake all the physics systems, and then copy the blendcache to the renderfarm as well.
- Be careful with the sequence of modifiers in the modifier stack (as always). You may have a different number of faces in the 3D View and for rendering (e.g. when using subdivision surface), if so, the rendered result may be very different from what you see in the 3D View.

Velocity

The initial velocity of particles can be set through different parameters, based on the type of the particle system (see Particle System tab). If the particle system type is Emitter or Hair, then the following parameters give the particle an initial velocity in the direction of...

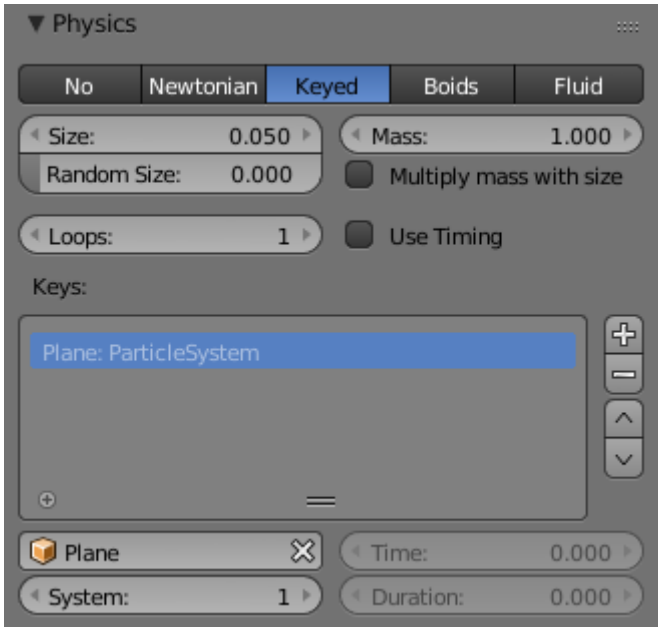


Fig. 2.1411: Initial velocity.

Emitter Geometry

Normal The emitter's surface normals (i.e. let the surface normal give the particle a starting speed).

Tangent Let the tangent speed give the particle a starting speed.

Rot Rotates the surface tangent.

Emitter Object

Align Give an initial velocity in the X, Y, and Z axes.

X, Y, Z

Object The emitter objects movement (i.e. let the object give the particle a starting speed).

Random Gives the starting speed a random variation. You can use a texture to only change the value, see Controlling Emission, Interaction and Time).

Rotation

These parameters specify how the individual particles are rotated during their travel. To visualize the rotation of a particle you should choose visualization type Axis in the Visualization panel and increase the Draw Size.

Initial Rotation Mode Sets the initial rotation of the particle by aligning the x-axis in the direction of:

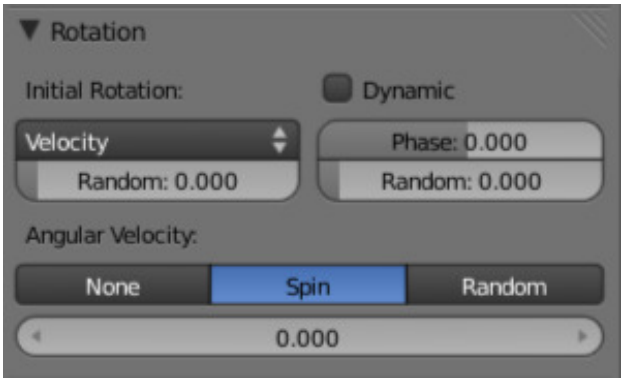


Fig. 2.1412: Particles rotation settings.

None The global X-axis.

Normal Orient to the emitter's surface normal, the objects Y axis points outwards.

Normal-Tangent As with normal, orient the Y axis to the surface normal. Also orient the X axis to the tangent for control over the objects rotation about the normal. requires UV coordinates, the UV rotation effects the objects orientation, currently uses the active UV layer. This allow deformation without the objects rotating in relation to their surface.

Velocity The particle's initial velocity.

Global X/Global Y/Global Z One of the global axes.

Object X/Object Y/Object Z One of the emitter object axes.

Random Randomizes rotation.

Dynamic If enabled, only initializes particles to the wanted rotation and angular velocity and let us physics handle the rest. Particles then change their angular velocity if they collide with other objects (like in the real world due to friction between the colliding surfaces). Otherwise the angular velocity is predetermined at all times (i.e. set rotation to dynamic/constant).

Phase Initial rotation phase.

Random Rand allows a random variation of the Phase.

Angular Velocity The magnitude of angular velocity, the selector specifies the axis of angular velocity to be.

None a zero vector (no rotation).

Spin the particles velocity vector.

Random a random vector.

If you use a Curve Guide and want the particles to follow the curve, you have to set Angular Ve-

locity to Spin and leave the rotation on Constant (i.e. do not turn on Dynamic). Curve Follow does not work for particles.

Physics

Introduction

The movement of particles may be controlled in a multitude of ways. With particles physics: there are five different systems:

None (*No Physics*) It does not give the particles any motion, which makes them belong to no physics system.

Newtonian Movement according to physical laws.

Keyed Dynamic or static particles where the (animated) targets are other particle systems.

Boids Particles with limited artificial intelligence, including behavior and rules programming, ideal for flocks of birds or schools of fishes, or predators vs preys simulations.

Fluid Movement according to fluid laws (based on Smoothed Particle Hydrodynamics technique).

Additional ways of moving particles:

- By softbody animation (only for Hair particle systems).
- By forcefields and along curves.
- By lattices.

Here we will discuss only the particle physics in the narrower sense, i.e. the settings in the Physics panel.

Common Physics Settings

Size Sets the size of the particles.

Random Size Give the particles a random size variation.

Mass Specify the mass of the particles.

Multiply mass with particle size Causes larger particles to have larger masses.

No Physics

At first a Physics type that makes the particles do nothing could seem a bit strange, but it can be very useful at times. None physics make the particles stick to their emitter their whole life time. The initial velocities here are for example

used to give a velocity to particles that are affected by a harmonic effector with this physics type when the effect of the effector ends.

Moreover, it can be very convenient to have particles at disposal (whose both Unborn and Died are visible on render) to groom vegetation and/or ecosystems using Object, Group or Billboard types of visualization.

Newtonian

These are the "normal" particle physics. Particles start their life with the specified initial velocities and angular velocities, and move according to Newtonian forces. The response to environment and to forces is computed differently, according to any given integrator chosen by the animator.

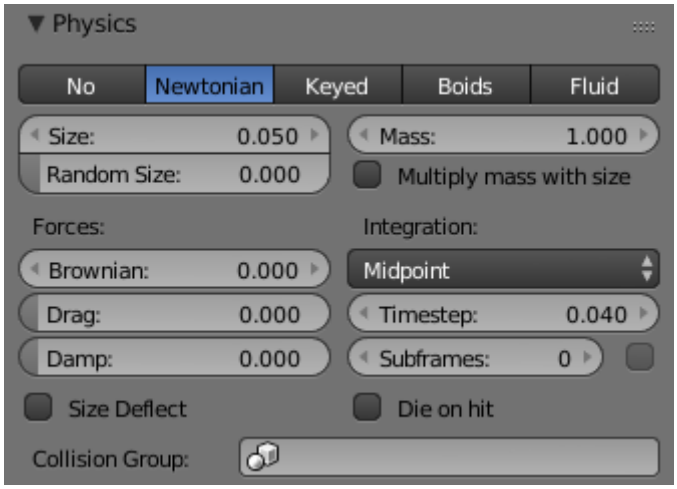


Fig. 2.1413: Newtonian Physics Settings.

Forces

Brownian Specify the amount of Brownian motion. Brownian motion adds random motion to the particles based on a Brownian noise field. This is nice to simulate small, random wind forces.

Drag A force that reduces particle velocity in relation to its speed and size (useful in order to simulate Air-Drag or Water-Drag).

Damp Reduces particle velocity (deceleration, friction, dampening).

Collision

Size Deflect Use the particle size in deflections.

Die on Hit Kill particle when it hits a deflector object.

Collision Group If set, particles collide with objects from the group, instead of using objects that are on the same layer.

Integration

Integrators are a set of mathematical methods available to calculate the movement of particles. The following guidelines will help to choose a proper integrator, according to the behavior aimed at by the animator.

Euler Also known as "Forward Euler". Simplest integrator. Very fast but also with less exact results. If no dampening is used, particles get more and more energy over time. For example, bouncing particles will bounce higher and higher each time. Should not be confused with "Backward Euler" (not implemented) which has the opposite feature, energies decrease over time, even with no dampening. Use this integrator for short simulations or simulations with a lot of dampening where speedy calculations is more important than accuracy.

Varlet Very fast and stable integrator, energy is conserved over time with very little numerical dissipation.

Midpoint Also known as "2nd order Runge-Kutta". Slower than Euler but much more stable. If the acceleration is constant (no drag for example), it is energy conservative. It should be noted that in example of the bouncing particles, the particles might bounce higher than they started once in a while, but this is not a trend. This integrator is a generally good integrator for use in most cases.

RK4 Short for "4th order Runge-Kutta". Similar to Midpoint but slower and in most cases more accurate. It is energy conservative even if the acceleration is not constant. Only needed in complex simulations where Midpoint is found not to be accurate enough.

Frame Settings

Timestep The simulation time step per frame.

Subframes Subframes to simulate for improved stability and finer granularity in simulations. Use higher values for faster moving particles.

Keyed

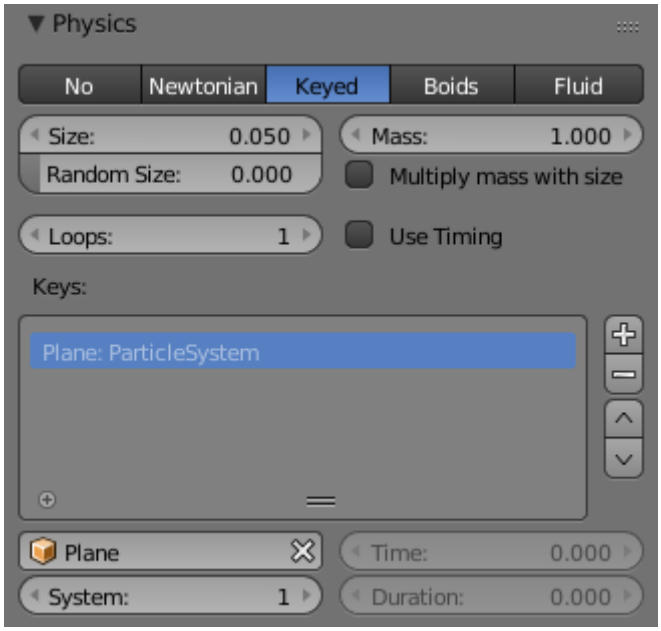


Fig. 2.1414: Keyed Physics Settings.

The particle paths of keyed particles are determined from the emitter to another particle system's particles. This allows creation of chains of systems with keyed physics to create long strands or groovy moving particles. Basically the particles have no dynamics but are interpolated from one system to the next at drawtime.

Setup

To setup Keyed particles you need at least two particle systems.

The first system has keyed physics, and it needs the option First activated. This will be the system thats is visible.

The second system may be another keyed system but without the option First, or a normal particle system. This second system is the target of the keyed system.

Loops Sets the number of times the keys are looped. Disabled if *Use Timing* is enabled.

Keys

Key Targets You have to enter the name of the object which bears the target system and if there are multiple particle systems the number of the system.

Click the `Plus` to add a key, then select the object.

If you use only one keyed system the particles will travel in their lifetime from the emitter to the target. A shorter lifetime means faster movement. If you have more than one keyed system in a chain, the lifetime will be split equally. This may lead to varying particle speeds between the targets.

Timing

Use Timing Timing works together with the Time slider for the other keyed systems in a chain. The Time slider allows to define a fraction of particle lifetime for particle movement.

An example: let us assume that you have two keyed systems in a chain and a third system as target. The particle lifetime of the first system shall be 50 keys. The particles will travel in 25 frames from the first keyed system to the second, and in further 25 frames from the second system to the target. If you use the Timed button for the first system, the Time slider appears in the second systems panel. Its default value is 0.5, so the time is equally split between the systems. If you set Time to 1, the movement from the first system to the second will get all the lifetime (the particles will die at the second system).

If you set Time to 0 the particles will start at the second system and travel to the target.

Boids

Boids particle systems can be set to follow basic rules and behaviors. They are useful for simulating flocks, swarms, herds and schools of various kind of animals, insects and fishes. They can react on the presence of other objects and on the members of their own system. Boids can handle only a certain amount of information, therefore the sequence of the Behavior settings is very important. In certain situations only the first three parameter are evaluated.

To view the panel to the right, add a *Particle System* of type *Emitter* and look in the middle area of the *Particle System* tab.

Physics

Boids try to avoid objects with activated Deflection. They try to reach objects with positive

Fig. 2.1415: Boid Physics Settings.

Spherical fields, and fly from objects with negative Spherical fields. The objects have to share one common layer to have effect. It is not necessary to render this common layer, so you may use invisible influences.

Boids can different physics depending on whether they are in the air, or on land (on collision object)

Allow Flight Allow boids to move in the air.

Allow Land Allow boids to move on land.

Allow Climbing Allow boids to climb goal objects.

Max Air Speed Set the Maximum velocity in the air.

Min Air Speed Set the Minimum velocity in the air.

Max Air Acceleration Lateral acceleration in air, percent of max velocity (turn). Defines how fast a boid is able to change direction.

Max Air Angular Velocity Tangential acceleration in air, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Air Personal Space Radius of boids personal space in air. Percentage of particle size.

Landing Smoothness How smoothly the boids land.

Max Land Speed Set the Maximum velocity on land.

Jump Speed Maximum speed for jumping

Max Land Acceleration Lateral acceleration on land, percent of max velocity (turn). Defines how fast a boid is able to change direction.

Max Land Angular Velocity Tangential acceleration on land, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Land Personal Space Radius of boids personal space on land. Percentage of particle size.

Land Stick Force How strong a force must be to start effecting a boid on land.

Banking Amount of rotation around velocity vector on turns. Banking of 1.0 gives a natural banking effect.

Pitch Amount of rotation around side vector.

Height Boid height relative to particle size.

Battle

Health Initial boid health when born.

Strength Maximum caused damage per second on attack.

Aggression Boid will fight this times stronger than enemy.

Accuracy Accuracy of attack.

Range Maximum distance of which a boid can attack.

Alliance

The relations box allows you to set up other particle systems to react with the boids. Setting the type to *Enemy* will cause the systems to fight with each other. *Friend* will make the systems work together. *Neutral* will not cause them to align or fight with each other.

Deflectors and Effectors

As mentioned before, very much like Newtonian particles, Boids will react to the surrounding deflectors and fields, according to the needs of the animator:

Deflection: Boids will try to avoid deflector objects according to the Collision rule's weight. It works best for convex surfaces (some work needed for concave surfaces). For boid physics, Spherical fields define the way the objects having the field are seen by others. So a negative Spherical field (on an object or a particle system) will be a predator to all other boids particle systems, and a positive field will be a goal to all other boids particle systems.

When you select an object with a particle system set on, you have in the Fields tab a little menu stating if the field should apply to the emitter object or to the particle system. You have to select the particle system name if you want prey particles to flew away from predator particles.

Spherical fields: These effectors could be predators (negative Strength) that boids try to avoid or targets (positive Strength) that boids try to reach according to the (respectively) Avoid and Goal rules' weights. Spherical's effective Strength is multiplied by the actual relevant weight (e.g. if either Strength or Goal is null, then a flock of boids will not track a positive Spherical field). You can also activate Die on hit (Extras panel) so that a prey particle simply disappears when "attacked" by a predator particle which reaches it. To make this work, the predator particles have to have a spherical field with negative force, it is not sufficient just to set a positive goal for the prey particles (but you may set the predators force strength to -0.01). The size of the predators and the prey can be set with the Size button in the Extras panel.

Boid Brain

The Boid Brain panel controls how the boids particles will react with each other. The boids' behavior is controlled by a list of rules. Only a certain amount of information in the list can be evaluated. If the memory capacity is exceeded, the remaining rules are ignored.

The rules are by default parsed from top-list to bottom-list (thus giving explicit priorities), and the order can be modified using the little arrows buttons on the right side.

The list of rules available are:

Goal Seek goal (objects with Spherical fields and positive Strength).

Predict Predict target's movements.

Avoid Avoid "predators" (objects with Spherical fields and negative Strength).

Predict Predict target's movements.

Fear Factor Avoid object if danger from it is above this threshold.

Avoid Collision Avoid objects with activated Deflection.

Boids Avoid collision with other boids.

Deflectors Avoid collision with deflector objects.

Look Ahead Time to look ahead in seconds.

Separate Boids move away from each other.

Flock Copy movements of neighboring boids, but avoid each other.

Follow Leader Follows a leader object instead of a boid.

Distance Distance behind leader to follow.

Line Follow the leader in a line.

Average Speed Maintain average velocity.

Speed Percentage of maximum speed.

Wander How fast velocity's direction is randomized.

Level How much velocity's Z component is kept constant.

Fight Move toward nearby boids.

Fight Distance Attack boids at a maximum of this distance.

Flee Distance Flee to this distance.

Rule Evaluation

There are three ways control how rules are evaluated.

Average All rules are averaged.

Random A random rule is selected for each boid.

Fuzzy Uses fuzzy logic to evaluate rules. Rules are gone through top to bottom. Only the first rule that effect above fuzziness threshold is evaluated. The value should be considered how hard the boid will try to respect a given rule (a value of 1.000 means the Boid will always stick to it, a value of 0.000 means it will never). If the boid meets more than one conflicting condition at the same time, it will try to fulfill all the rules according to the respective weight of each.

Please note that a given boid will try as much as it can to comply to each of the rules he is given, but it is more than likely that some rule will take precedence on other in some cases. For example, in order to avoid a predator, a boid could probably "forget" about Collision, Crowd and Center rules, meaning that "while panicked" it could well run into obstacles, for example, even if instructed not to, most of the time.

As a final note, the Collision algorithm is still not perfect and in research progress, so you can expect wrong behaviors at some occasion. It is worked on.

Fluid

Fluid simulations are widely used in CG, and a very desired feature of any particle system, fluid particles are similar to newtonian ones but this time particles are influenced by internal forces like pressure, surface tension, viscosity, springs, etc. Blender particle fluids use the SPH techniques to solve the particles fluid equations.

Smoothed-particle hydrodynamics (SPH) is a computational method used for simulating fluid flows. It has been used in many fields of research, including astrophysics, ballistics, vulcanology, and oceanography. It is a mesh-free Lagrangian method (where the co-ordinates move with the fluid), and the resolution of the method can easily be adjusted with respect to variables such as the density.

From liquids to slime, goo to sand and wispy smoke the possibilities are endless.

Settings

Fluid physics share options with *Newtonian Physics*. These are covered on that page.

Fluid Properties

Stiffness How incompressible the fluid is.

Viscosity Linear viscosity. Use lower viscosity for thicker fluids.

Buoyancy Artificial buoyancy force in negative gravity direction based on pressure differences inside the fluid.

Advanced

Repulsion Factor How strongly the fluid tries to keep from clustering (factor of stiffness). Check box sets repulsion as a factor of stiffness.

Stiff Viscosity Creates viscosity for expanding fluid. Check box sets this to be a factor of normal viscosity.

Interaction Radius Fluid's interaction radius. Check box sets this to be a factor of 4 × *particle size*.

Rest Density Density of fluid when at rest. Check box sets this to be a factor of default density.

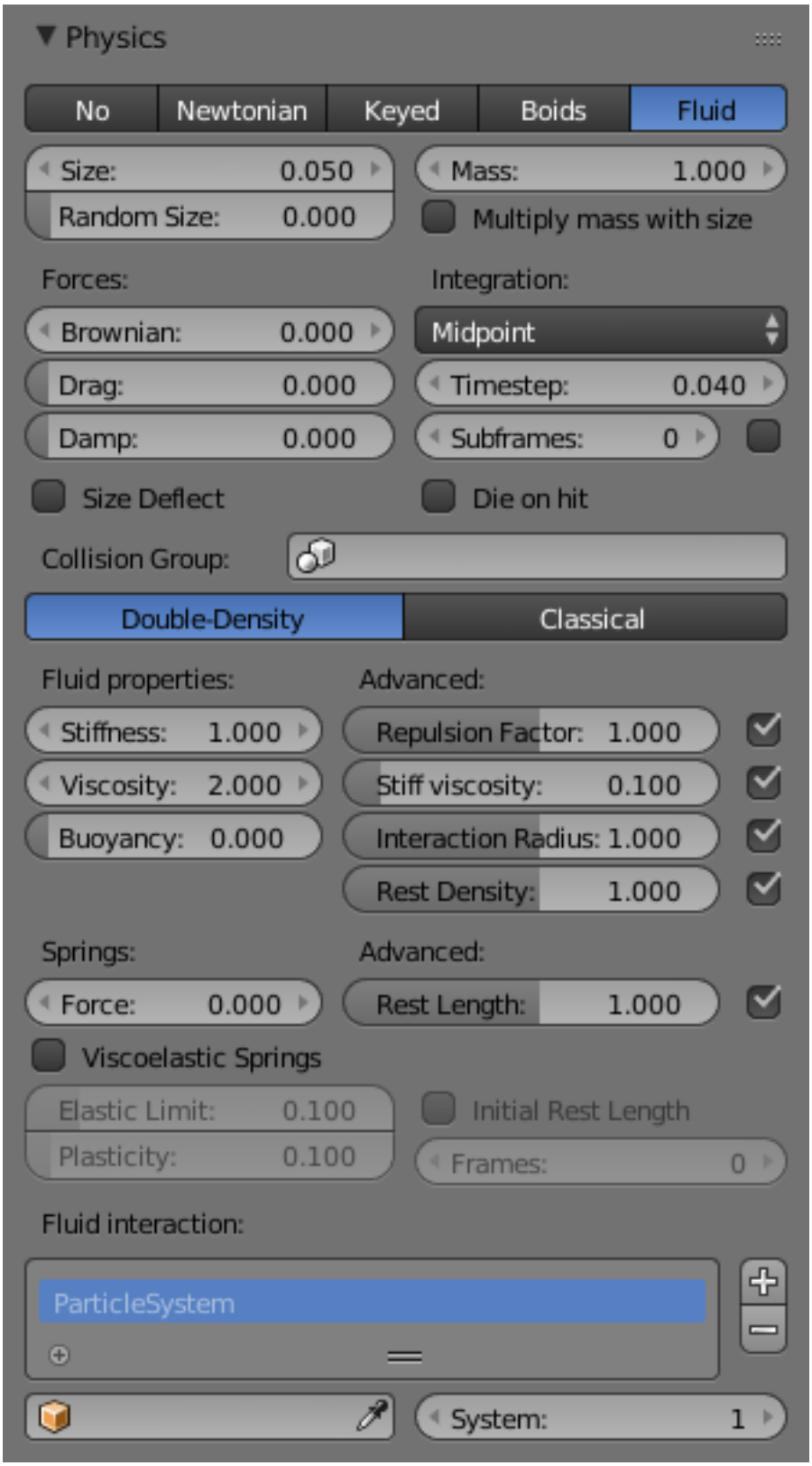


Fig. 2.1416: Fluid Physics Settings.

Springs

Force Spring force

Rest Length Rest length of springs. Factor of particle radius. Check box sets this to be a factor of $2 \times$ *particle size*.

Viscoelastic Springs Use viscoelastic springs instead of Hooke's springs.

Elastic Limit How much the spring has to be stretched/compressed in order to change its rest length

Plasticity How much the spring rest length can change after the elastic limit is crossed.

Initial Rest Length Use initial length as spring rest length instead of $2 \times$ *particle size*.

Frames Create springs for this number of frames since particle's birth (0 is always).

Render

The Render Panel controls how particles appear when they are rendered.

Material Index Set which of the object's material is used to shade the particles.

Parent Use a different object's coordinates to determine the birth of particles.

Emitter When disabled, the emitter is no longer rendered. Activate the button *Emitter* to also render the mesh.

Parents Render also parent particles if child particles are used. Children have a lot of different deformation options, so the straight parents would stand between their curly children. So by default *Parents* are not rendered if you activate *Children*.. See *Children*

Unborn Render particles before they are born.

Died Render particles after they have died. This is very useful if particles die in a collision *Die on hit*, so you can cover objects with particles.

None

When set to *None* particles are not rendered. This is useful if you are using the particles to duplicate objects.

Halo

Halo particles are rendered as *Halo Type Materials*.

Trail Count Set the number of trail particles. When greater than 1, additional options appear.

Length in Frames Path timing is in absolute frames.

Length End time of drawn path.

Random Give path lengths a random variation.

Line

The Line visualization mode creates (more or less thin) polygon lines with the strand renderer in the direction of particles velocities. The thickness of the line is set with the parameter *Start* of the *Strands* shader (*Material* tab, *Links and Pipeline* panel).

Tail Set the length of the particle's tail.

Head Set the length of the particle's head.

Speed Multiply the line length by particles' speed. The faster, the longer the line.

Trail Count See description in *Halo*.

Path

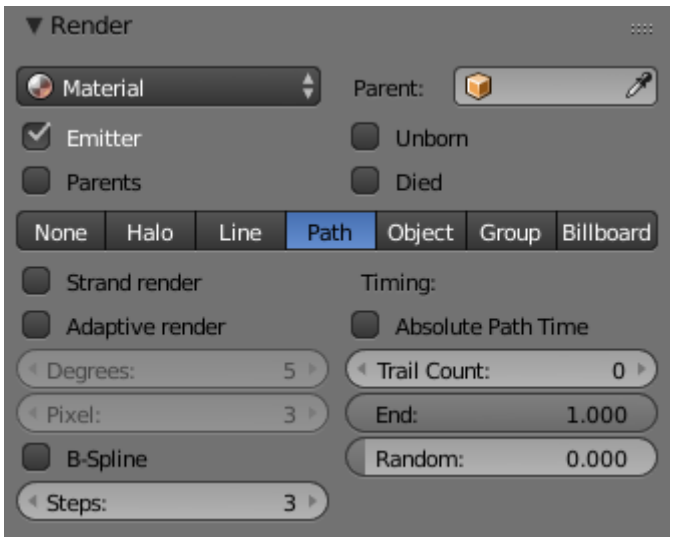


Fig. 2.1417: The Visualization panel for Path visualization.

The *Path* visualization needs a *Hair* particle system or *Keyed* particles.

Strand render [Keypointstrands] Use the strand primitive for rendering. Very fast and effective renderer.

Adaptive render Tries to remove unnecessary geometry from the paths before rendering particle strands in order to make the render faster and easier on memory.

Angle How many degrees path has to curve to produce another render segment (straight parts of paths need fewer segments).

Pixel How many pixels path has to cover to produce another render segment (very short hair or long hair viewed from far away need fewer parts). (only for Adaptive render).

B-Spline Interpolate hair using B-Splines. This may be an option for you if you want to use low *Render* values. You loose a bit of control but gain smoother paths.

Steps Set the number of subdivisions of the rendered paths (the value is a power of 2). You should set this value carefully, because if you increase the render value by two you need four times more memory to render. Also the rendering is faster if you use low render values (sometimes drastically). But how low you can go with this value depends on the waviness of the hair.(the value is a power of 2). This means 0 steps give 1 subdivision, 1 give 2 subdivisions, 2 –> 4, 3 –> 8, 4 –> 16, ... n –> n^2.

Timing

Absolute Path Time Path timing is in absolute frames.

Start Start time of the drawn path.

End End time of the drawn path.

Random Give the path length a random variation.

Please see also the manual page about *Strands* for an in depth description.

Object

Dupli Object The specified object is duplicated in place of each particle.

Global Use object's global coordinates for duplication.

Rotation Use the rotation of the object.

Scale Use the size of the object.

Group

Dupli Group The objects that belong to a group are duplicated sequentially in the place of the par-

ticles.

Whole Group Use the whole group at once, instead of one of its elements, the group being displayed in place of each particle.

Pick Random The objects in the group are selected in a random order, and only one object is displayed in place of a particle. Please note that this mechanism fully replaces old Blender particles system using parentage and DupliVerts to replace particles with actual geometry. This method is fully deprecated and does not work anymore.

Use Count Use objects multiple times in the same groups. Specify the order and number of times to repeat each object with the list box that appears. You can duplicate an object in the list with the `Plus` button, or remove a duplicate with the `Minus` button.

Use Global Use object's global coordinates for duplication.

Rotation Use the rotation of the objects.

Scale Use the size of the objects.

Billboard

Billboards are aligned square planes. They are aligned to the camera by default, but you can choose another object that they should be aligned to.

If you move a billboard around its target, it always faces the center of its target. The size of a billboard is set with the parameter *Size* of the particle (in Blender Units). You can use them e.g. for Sprites, or to replace *Halo* visualization. Everything that can be done with a halo can also be done with a billboard. But billboards are real objects, they are seen by raytracing, they appear behind transparent objects, they may have an arbitrary form and receive light and shadows. They are a bit more difficult to set up and take more render time and resources.

Texturing billboards (including animated textures with alpha) is done by using uv coordinates that are generated automatically for them so they can take an arbitrary shape. This works well for animations, because the alignment of the billboards can be dynamic. The textures can be animated in several ways:

- Depending on the particle lifetime (relative time).
- Depending on the particle starting time.

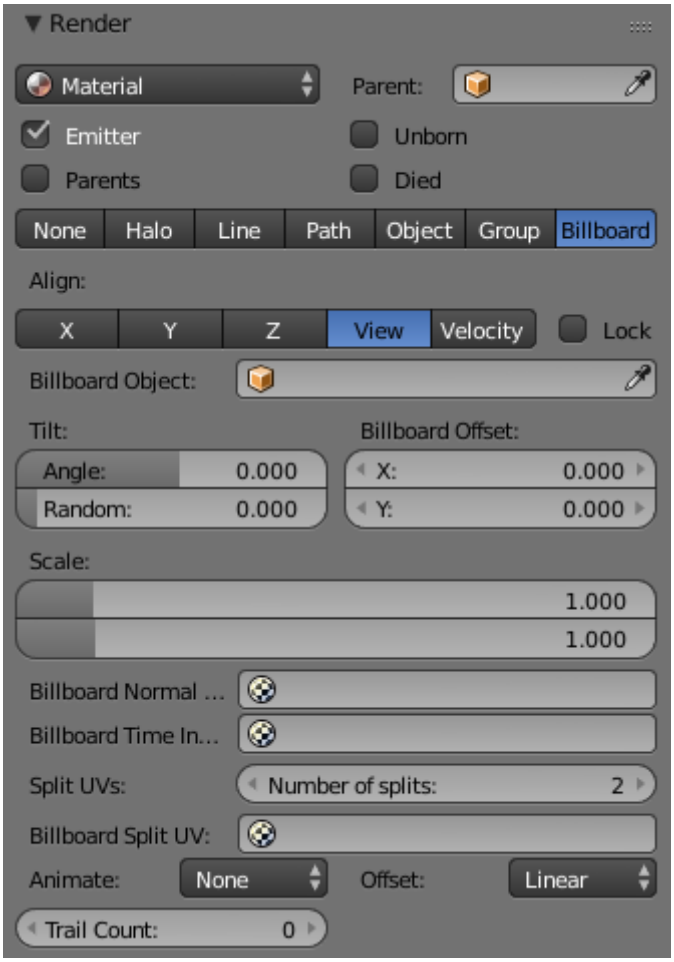


Fig. 2.1418: Billboard visualization for particles.

- Depending on the frame (absolute time).

  You can use different sections of an image texture:

- Depending on the lifetime of the billboard.
- Depending on the emission time.
- Depending on align or tilt.

  Since you use normal materials for the billboard you have all freedoms in mixing textures to your liking. The material itself is animated in absolute time.

  The main thing to understand is that if the object does not have any *UV Layers*, you need to create at least one in the *Objects Data* tab, for any of these to work. Moreover, the texture has to be set to UV coordinates in the *Map Input* panel. If you want to see examples for some of the animation possibilities, see the Billboard Animation Tutorial.

  An interesting alternative to billboards are in certain cases strands, because you can animate the shape of the strands. Because this visualization type has so much options it is explained in greater detail below.

Align You can limit the movement with these options. How the axis is prealigned at emission time.

X, Y , Z Along the global X/Y/Z-axis respectively.

View No prealignement, normal orientation to the target.

Velocity Along the speed vector of the particle.

Lock Locks the align axis, keeps this orientation, the billboard aligns only along one axis to its target.

Billboard Object The target object that the billboards are facing. By default, the active camera is used.

Tilt Angle Rotation angle of the billboards planes. A tilt of 1 rotates by 180 degrees (turns the billboard upside down).

Random Random variation of tilt.

Offset X Offset the billboard horizontally in relation to the particle center, this does not move the texture.

Offset Y Offset the billboard vertically in relation to the particle center.

UV Channels Billboards are just square polygons. To texture them in different ways we have to have a way to set what textures we want for the billboards and how we want them to be mapped

to the squares. These can then be set in the texture mapping buttons to set wanted textures for different coordinates. You may use three different UV layers and get three different sets of UV coordinates, which can then be applied to different (or the same) textures.

Billboard Normal UV Coordinates are the same for every billboard, and just place the image straight on the square.

Billboard Time-Index (X-Y) Coordinates actually define single points in the texture plane with the x-axis as time and y-axis as the particle index. For example using a horizontal blend texture mapped to color from white to black will give particles that start off as white and gradually change to black during their lifetime. On the other hand a vertical blend texture mapped to color from white to black will make the first particle to be white and the last particle to be black with the particles in between a shade of gray.

The animation of the UV textures is a bit tricky. The UV texture is split into rows and columns (N times N). The texture should be square. You have to use *UV Split* in the UV channel and fill in the name of the UV layer. This generated UV coordinates for this layer.

Split UV's The amount of rows/columns in the texture to be used. Coordinates are a single part of the *UV Split* grid, which is a n?n grid over the whole texture. What the part is used for each particle and at what time is determined by the *Offset* and *Animate* controls. These can be used to make each billboard unique or to use an "animated" texture for them by having each frame of the animation in a grid in a big image.

Billboard Split UV Set the name of the *UV layer* to use with billboards (you can use a different one for each *UV Channel*). By default, it is the active UV layer (check the *Object Data* tab in the Properties editor).

Animate Select menu, indicating how the split UVs could be animated (changing from particle to particle with time):

None No animation occurs on the particle itself, the billboard uses one section of the texture in its lifetime.

Age The sections of the texture are gone through sequentially in particles' lifetimes.

Angle Change the section based on the angle of rotation around the *Align to* axis, if *View* is used the change is based on the amount of tilt.

Frame The section is changes according to the frame.

Offset Specifies how to choose the first part (of all the parts in the n×n grid in the texture defined by the *UV Split* number) for all particles.

None All particles start from the first part.

Linear First particle will start from the first part and the last particle will start from the last part, the particles in between will get a part assigned linearly from the first to the last part.

Random Give a random starting part for every particle.

Trail Count See the description in *Halo*.

Display

The Display Panel controls how particles are displayed in the 3D View. This does not necessarily determine how they will appear when rendered.

Draw Method

None The particles are not shown in the 3D View and are not rendered. The emitter may be rendered though.

Point Particles are displayed as square points. Their size is independent of the distance from the camera.

Circle Particles are displayed as circles that face the view. Their size is independent of the distance from the camera.

Cross Particles are displayed as 6-point crosses that align to the rotation of the particles. Their size is independent of the distance from the camera.

Axis Particles are displayed as 3-point axes. This useful if you want to see the orientation and rotation of particles in the view port. Increase the *Draw Size* until you can clearly distinguish the axis.

Particles visualized like Point, Circle, Cross and Axis do not have any special options, but can be very useful when you have multiple particle systems at play, if you do not want to confuse particles of one system from another (e.g. in simulations using *Boids* physics).

Display Specifies the percentage of all particles to show in the viewport (all particles are still rendered).

Draw Size Specifies how large (in pixels) the particles are drawn in the viewport.

Size Draw the size of the particles with a circle.

Velocity Draw the velocity of the particles with a line that points in the direction of motion, and length relative to speed.

Number Draw the id-numbers of the particles in the order of emission.

Color

The Color Menu allows you to draw particles according to certain particle properties.

None Particles are black.

Material Particles are colored according to the material they are given.

Velocity Color particles according to their speed. The color is a ramp from blue to green to red, Blue being the slowest, and Red being velocities approaching the value of *Max* or above. Increasing *Max* allows for a wider range of particle velocities.

Acceleration Color particles according to their acceleration.

Children

Children are *Hair* and *Keyed* particles assigned subparticles. They make it possible to work primarily with a relatively low amount of Parent particles, for whom the physics are calculated. The children are then aligned to their parents. Without recalculating the physics the number and visualization of the children can be changed.

Children can be emitted from particles or from faces (with some different options). Emission from *Faces* has some advantages, especially the distribution is more even on each face (which makes it better suitable for fur and the like). However, children from particles follow their parents better, e.g. if you have a softbody animation and do not want the hair to penetrate the emitting mesh. But see also our manual page about *Hair*.

If you turn on children the parents are no longer rendered (which makes sense because the shape of the children may be quite different from that of their parents). If you want to see the parents additionally turn on the *Parents* button in the *Visualization* panel.

Children carry the same material as their parents and are colored according to the exact

place from where they are emitted (so all children may have different color or other attributes).

The possible options depend from the type of particle system, and if you work with *Children from faces* or *Children from particles*. We do not show every possible combination, only the settings for a *Hair* particle system.

Settings

Child Type

None No Children are generated.

Simple Children are emitted from the parent position.

Interpolated Children are emitted between the *Parent* particles on the faces of a mesh. They interpolate between adjacent parents. This is especially useful for fur, because you can achieve an even distribution. Some of the children can become virtual parents, which are influencing other particles nearby.

Display The number of children in the 3D View.

Render The number of children to be rendered.

Simple

Size Only for *Emitter*. A multiplier for children size.

Random Random variation to the size of child particles.

Interpolated

Seed Offset the random number table for child particles, to get a different result.

Virtual Relative amount of virtual parents.

Long Hair Calculate children that suit long hair well.

Effects

Use Clump Curve Todo.

Clump Clumping. The children may meet at their tip (1.0) or start together at their root (-1.0).

Shape Form of *Clump*. Either inverse parabolic (0.99) or exponentially (-0.99).

Use Clump Noise Todo.

Length Length of child paths.

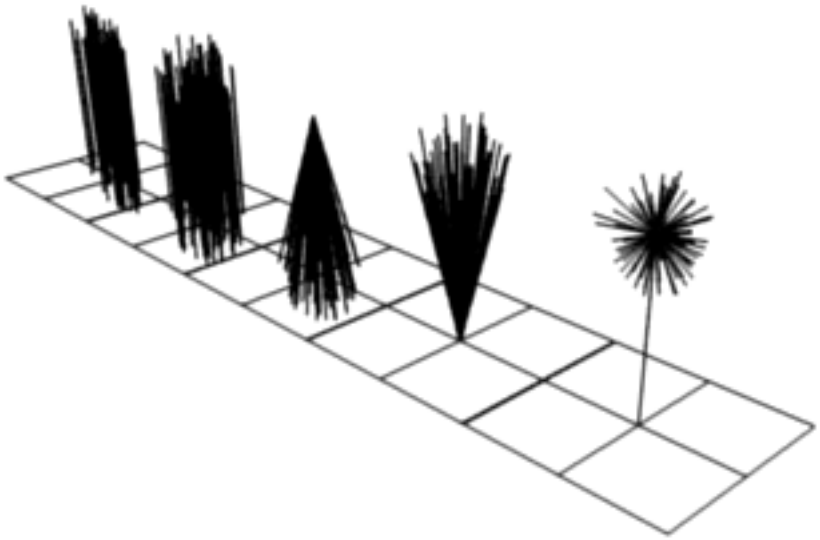


Fig. 2.1419: From left to right: Round: 0.0, Round: 1.0, Clump: 1.0, Clump: -1.0, Shape: -0.99.

Threshold Amount of particles left untouched by child path length.

Radius The radius in which the children are distributed around their parents. This is 3D, so children may be emitted higher or lower than their parents.

Roundness The roundness of the children around their parents. Either in a sphere (1.0) or in-plane (0.0).

Seed Offset in the random number table for child particles, to get a different randomized result.

Roughness

Use Roughness Curve Todo.

Uniform, Size It is based on children location so it varies the paths in a similar way when the children are near.

Endpoint, Shape "Rough End" randomizes path ends (a bit like random negative clumping). Shape may be varied from <1 (parabolic) to 10.0 (hyperbolic).

Random, Size, Threshold It is based on a random vector so it is not the same for nearby children. The threshold can be specified to apply this to only a part of children. This is useful for creating a few stray children that will not do what

others do.

Kink

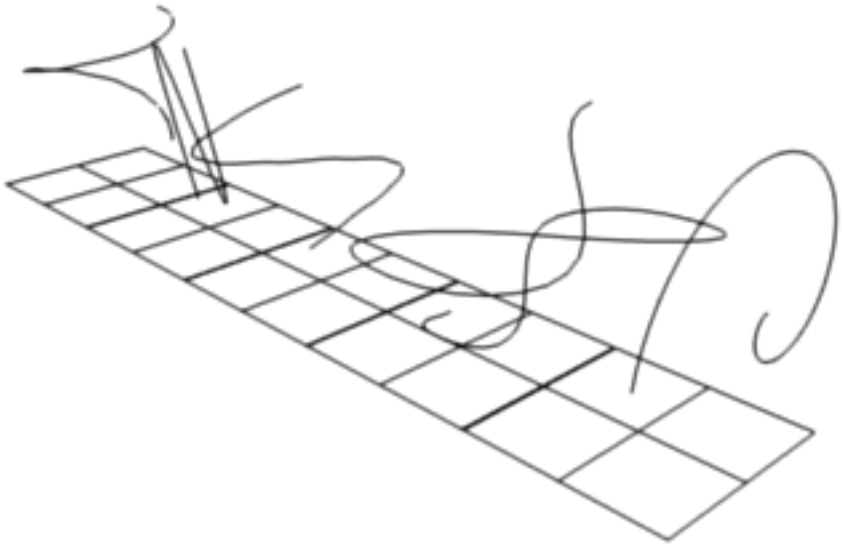


Fig. 2.1420: Child particles with Kink. From left to right: Curl, Radial, Wave, Braid, Roll.

With *Kink* you can rotate the children around the parent. See Fig. *Child particles with Kink. From left to right: Curl, Radial, Wave, Braid, Roll.* above picture for the different types of *Kink*.

Kink

Nothing Deactivated.

Curl Children grow in a spiral around the parent hairs.

Radial Children form around the parent a wave shape that passes through the parent hair.

Wave Children form a wave, all in the same direction.

Braid Children braid themselves around the parent hair.

Spiral Todo.

Amplitude The amplitude of the offset.

Clump How much clump effects kink amplitude.

Flatness How flat the hairs are.

Frequency The frequency of the offset (1/total length). The higher the frequency the more rotations are done.

Shape Where the rotation starts (offset of rotation).

Force Fields

Field Weights

The Field Weight Panel allows you to control how much influence each type of external force field, or effector, has on the particle system. Force fields are external forces that give dynamic systems motion. The force fields types are detailed on the *Force Field Page*.

Effector Group Limit effectors to a specified group. Only effectors in this group will have an effect on the current system.

Gravity Control how much the Global Gravity has an effect on the system.

All Scale all of the effector weights.

Force Fields Settings

The Force Field Settings Panel allows you to make each individual act as a force field, allowing them to affect other dynamic systems, or even, each other.

Self Effect Causes the particle force fields to have an effect on other particles within the same system.

Amount Set how many of the particles act as force fields. 0 means all of them are effectors.

You can give particle systems up to two force fields. By default they do not have any. Choose an effector type from the selector to enable them. Settings are described on the *Force Field Page*.

Vertex Groups

The Vertex groups panel allows you to specify vertex groups to use for several child particle settings. You can also negate the effect of each vertex group with the check boxes. You can affect the following attributes:

Density Defines the "density" of the particle distribution.

Length Defines the "length" of the particle distribution.

Clump Todo.

Kink Todo.

Roughness 1 Uniform.

Roughness 2 Random.

Roughness End Endpoint.

Examples

Todo.

Hair

Introduction

When set to hair type, particle system creates only static particles, which may be used for hair, fur, grass and the like.

Fig. 2.1421: Particle hair systems example. Used for the grass and fur.

Growing

The first step is to create the hair, specifying the amount of hair strands and their lengths.

The complete path of the particles is calculated in advance. So everything a particle does a hair may do also. A hair is as long as the particle path would be for a particle with a lifetime of 100 frames. Instead of rendering every frame of the particle animation point by point there are calculated control points with an interpolation, the segments.

Styling

The next step is to style the hair. You can change the look of base hairs by changing the *Physics Settings*.

A more advanced way of changing the hair appearance is to use *Children*. This adds child hairs to the original ones, and has settings for giving them different types of shapes.

You can also interactively style hairs in *Particle Edit Mode*. In this mode, the particle settings become disabled, and you can comb, trim, lengthen, etc. the hair curves.

Animating

Hair can now be made dynamic using the cloth solver. This is covered in the *Hair Dynamics* page.

Rendering

Blender can render hairs in several different ways. Materials have a Strand section, which is covered in the materials section in the *Strands Page*.

Hair can also be used as a basis for the *Particle Instance modifier*, which allows you to have a mesh be deformed along the curves, which is useful for thicker strands, or things like grass, or feathers, which may have a more specific look.

Options

Regrow Regrow Hair for each frame.

Advanced Enables advanced settings which reflect the same ones as working in Emitter mode.

Note: This manual assumes that this option is enabled.

Segments

Emission

Amount Set the amount of hair strands. Use as little particles as possible, especially if you plan to use softbody animation later. But you need enough particles to have good control. For a "normal" haircut I found some thousand (very

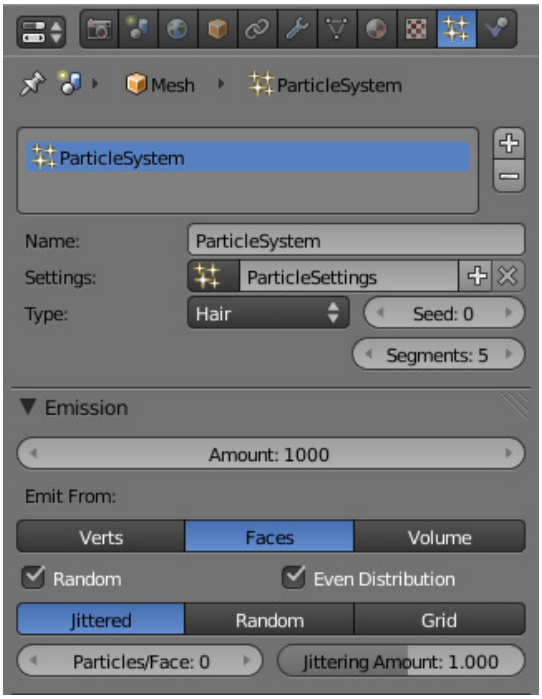


Fig. 2.1422: Hair particle system settings.

roughly 2000) particles to give enough control. You may need a lot more particles if you plan to cover a body with fur. Volume will be produced later with *Children*.

Hair Dynamics

Hair particles can have dynamic properties using physics.

To enable hair physics, click the check box beside *Hair Dynamics*.

Structure

Mass Value for the mass of the hair.

Stiffness Controls how stiff the root of the hair strands are.

Random Random stiffness of hair.

Damping Damping of bending motion.

Volume

Air Drag Controls how thick the hair is around the hair causing the hair to flow slower.

Internal Friction Amount of friction between individual hairs.

Density Target Maximum density if the hair.

Strength The influence that the *Density Target* has on the simulation.

Voxel Grid Cell Size Size of the voxel grid cells for interaction effects.

Pinning

Goal Strength Spring stiffness of the vertex target position.

Quality

Steps Quality of the simulation in steps per frame. (higher is better quality but slower).

Hair Grid Show hair simulation grid.

> **Warning:** If you use motion blur in your animation, you will need to bake one extra frame past the last frame which you will be rendering.

Display

Rendered Draw hair as curves.

Path Draw just the end points if the hairs.

Steps The number of segments (control points minus 1) of the hair strand. In between the control points the segments are interpolated. The number of control points is important:

- For the softbody animation, because the control points are animated like vertices, so more control points mean longer calculation times.
- For the interactive editing, because you can only move the control points (but you may recalculate the number of control points in *Particle Edit Mode*).

Hint: Segments

Ten Segments should be sufficient even for very long hair, five Segments are enough for shorter hair, and two or three segments should be enough for short fur.

Children

See *Children*.

Render

Hair can be rendered as a Path, Object, or Group. See *Particle Visualization* for descriptions.

See also:

Blender Hair Basics, a thorough overview of all of the hair particle settings.

Texture Influence

Defines the settings of a Particle system spatial with a texture.

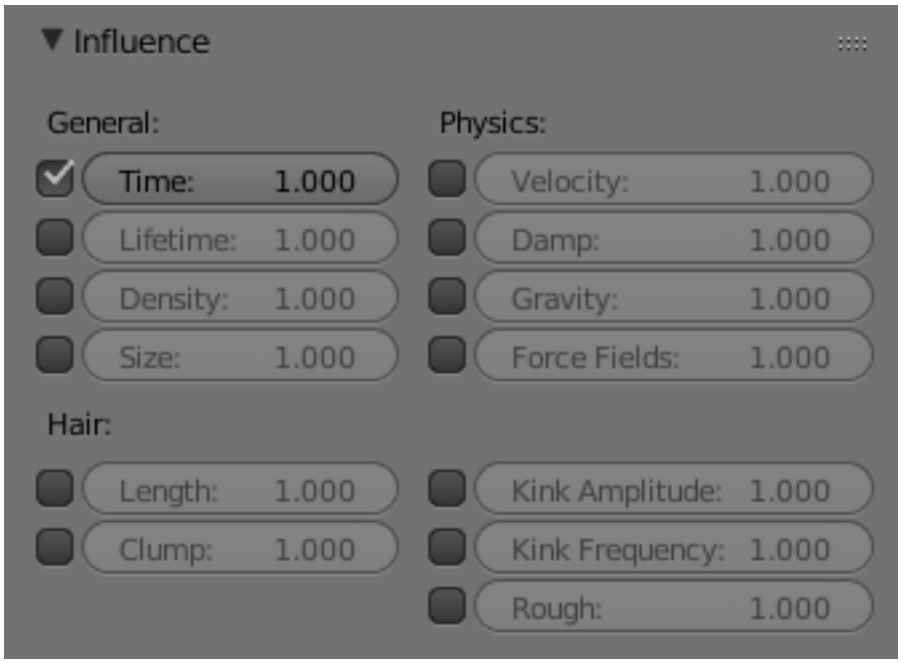


Fig. 2.1423: Texture influence settings.

General

Time Affect the emission time of the particles.

Lifetime Affect the life time of the particles.

Density Affect the density of the particles.

Size Affect the particles size.

Physics

Velocity Affect the particles initial velocity.

Damp Affect the particles velocity damping.

Gravity Affect the particles gravity.

Force Fiels Affect the particles force fields.

Hair

Length Affect the child hair length.

Clump Affect the child clumping.

Kink Affect the child kink.

Rough Affect the child roughness.

Particle Edit Mode

Using *Particle Edit Mode* you can edit the key-points (Keyframes) and paths of *Baked Hair*, *Particle*, *Cloth*, and *Soft Body* simulations. (You can also edit and style hair before baking).

Since working in Particle Edit Mode is pretty easy and very similar to working with vertices in the 3D View, we will show how to set up a particle system and then give a reference of the various functions.

Usage

Tip: Only Frames Baked to Memory are Editable!

If you cannot edit the particles, check that you are not baking to a *Disk Cache*.

Setup for Hair Particles

1. Create a *Hair* particle system.
2. Give it an initial velocity in the *Normal* direction.
3. Create a simulation
4. Check the *Hair Dynamics* box.

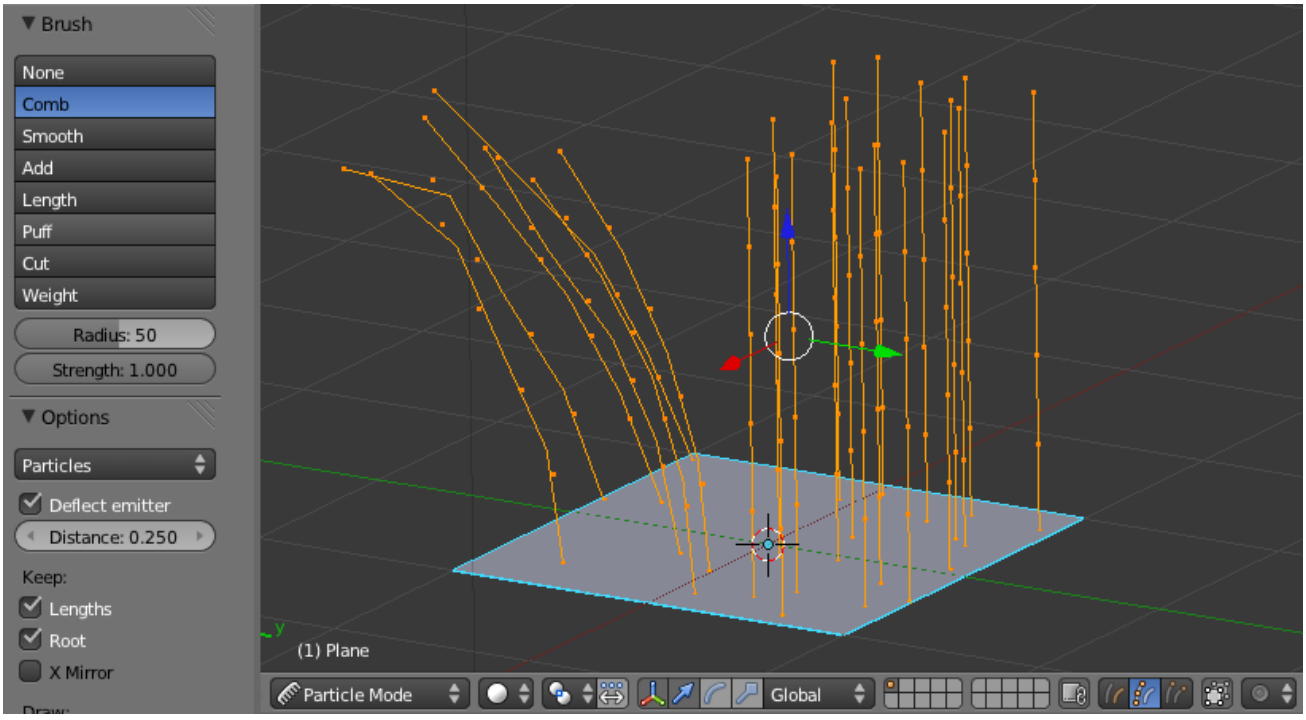


Fig. 2.1424: Editing hair strands in Particle Edit Mode.

Fig. 2.1425: Editing a baked particle simulation's particle paths in Particle Edit Mode.

Setup for Particle, Cloth, and Soft Body Simulations

1. Use *Emitter* particles, or a cloth/soft-body simulation
2. Create a simulation by setting up objects and or emitters, set your time range (use a small range if you are just starting out and experimenting), set up the simulation how you want it, using `Alt-A` to preview it.

Bake the Simulation

Once you are happy with the general simulation, *bake* the simulation from object mode. The simulation must be baked to enable editing.

Edit the Simulation

Switch to *Particle Edit* from the *Mode select menu* in the header of the *3D View* to edit the particle's paths/Keyframes. You may need to press `T` from within the 3D View to see the *Particle Edit* panel. Move to the frame you want to edit and use the various *Particle Edit* tools to edit your simulation. Work slowly, previewing your changes with `Alt-A`, and save often so that you can go back to the previous version should something happen, or that you do not like the latest changes you have made.

Tip: To be able to clearly see what you are working on:

1. Open the Options panel in the Tool shelf.
2. Select *Point select mode* (see below) in the header of the 3D View. This will display key points along the particle path.

Selecting

- Single: `RMB`.
- All: `A`.
- Linked: Move the mouse over a keypoint and press `L`.
- Border select: `B`.
- First/last: `W` → *Select First / Select Last*.

You may also use the *Select* Menu.

Tip: Selections

Selections are extremely useful for modifying only the particles that you want. Hover over a particle path and press `L` to link-select it, hover over the next and press `L` to add that path to the selection. To remove a path, hold `Shift` and press `L`. To Deselect all press `A`.

The method to select individual points is the same as in edit mode. `RMB` to select, `Shift-RMB` to add/remove a point from the selection.

Select Modes

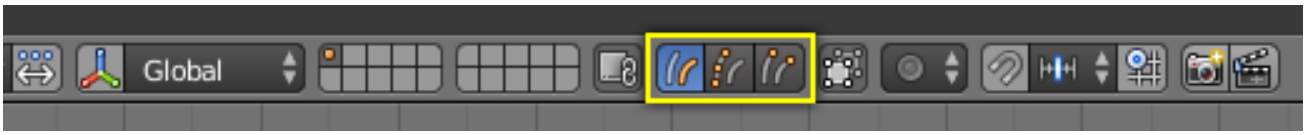


Fig. 2.1426: Select Modes.

Path No keypoints are visible, you can select/deselect only all particles.

Point You see all of the keypoints.

Tip You can see and edit (including the brushes) only the tip of the particles, i.e. the last keypoint.

Brush

With the buttons you can select the type of "Comb" utility you want to use. Below the brush types, their settings appear:

None No special tool, just edit the keypoints as "normal" vertices.

Comb Moves the keypoints (similar to "proportional editing").

Smooth Parallels visually adjacent segments.

Add Adds new particles.

Count The number of new particles per step.

Interpolate Interpolate the shape of new hairs from existing ones.

Steps Amount of brush steps

Keys How many keys to make new particles with.

Length Scales the segments, so it makes the hair longer with *Grow* or shorter with *Shrink*.

Puff Rotates the hair around its first keypoint (root). So it makes the hair stand up with *Add* or lay down with *Sub*.

Puff Volume Apply puff to unselected end-points, (helps maintain hair volume when puffing root)

Cut Scales the segments until the last keypoint reaches the brush.

Weight This is especially useful for softbody animations, because the weight defines the softbody *Goal*. A keypoint with a weight of 1 will not move at all, a keypoint with a weight of 0 subjects fully to softbody animation. This value is scaled by the *GMin* to *GMax* range of softbody goals...

Options

Common Options:

Radius Set the radius if the brush.

Tip: Brush Size

Press `F` to resize the brush while working.

Strength Set the strength of the brush effect (not for Add brush).

Add/Sub Grow/Shrink Sets the brush to add the effect or reverse it.

Deflect Emitter, Do not move keypoints through the emitting mesh.

Distance The distance to keep from the Emitter.

Keep

Length Keep the length of the segments between the keypoints when combing or smoothing the hair. This is done by moving all the other keypoints.

Root Keep first key unmodified, so you cannot transplant hair.

X Mirror Enable mirror editing across the local x axis.

Draw

Path Steps Drawing steps, sets the smoothness of the drawn path.

Show Children Draws the children of the particles too. This allows to fine tune the particles and see their effects on the result, but it may slow down your system if you have many children.

Editing

Warning: Beware of Undo!

Using *Undo* in *Particle Edit Mode* can have strange results. Remember to save often!

Moving keypoints or particles

- To move selected keypoints press G, or use one of the various other methods to grab vertices.
- To move a particle root you have to turn off Keep *Root* in the Tool Shelf.
- You can do many of the things like with vertices, including scaling, rotating and removing (complete particles or single keys).
- You may not duplicate or extrude keys or particles, but you can subdivide particles which adds new keypoints W → *Subdivide'or :kbd:'Numpad2*.
- Alternatively you can rekey a particle W → *Rekey* or Numpad1 and choose the number of keys.

How smoothly the hair and particle paths are displayed depends on the *Path Steps* setting in the Tool Shelf. Low settings produce blocky interpolation between points, while high settings produce a smooth curve.

Mirroring Particles

If you want to create an X-Axis symmetrical haircut you have to do following steps:

- Select all particles with A.
- Mirror the particles with Ctrl-M, or use the *Particle* → *Mirror* menu.
- Turn on *X-Axis Mirror Editing* in the *Particle* menu.

It may happen that after mirroring two particles occupy nearly the same place. Since this would be a waste of memory and rendertime, you can *Remove doubles* either from the *Specials* W or the *Particle* menu.

Hiding/Unhiding

Hiding and unhiding of particles works similar as with vertices in the 3D View. Select one or more keypoints of the particle you want to hide and press H. The particle in fact does not vanish, only the key points.

Hidden particles (i.e. particles whose keypoints are hidden) do not react on the various brushes. But:

If you use *Mirror Editing* even particles with hidden keypoints may be moved, if their mirrored counterpart is moved.

To un-hide all hidden particles press Alt-H.

Unify Length

This tool is used to make all selected hair uniform length by finding the average length.

2.9 Render

2.9.1 Introduction

Rendering is the process of creating a 2D image (or video) from your 3D scene. What that image looks like is based on four factors which the user can control:

- A *Camera*
- The *Lighting* in your scene
- The *Material* of each object
- Various render settings (quality, image size, layers etc)

Your computer will perform various complex calculations based on those factors in order to give you your rendered image. This process may take some time depending on the complexity of the scene and your hardware.

Once the render is complete, it is possible to do additional manipulation of the image, called *Post Processing*.

Finally, the output can be saved to an image or video file using one of the *Output Formats*.

Workflow

In general, the process for rendering is:

1. Position the camera
2. Light the scene
3. Setup materials
4. Render a test image using lower quality settings
5. Change or fix anything you noticed in the render
6. Repeat the above two steps until you are satisfied
7. Render a high quality image, change or fix any issues and repeat until satisfied
8. Save your image to a file, or render the animation to a video or image sequence.

Render Engines

The Render Engine is the set of code which controls how your materials and lighting are used, and ultimately what the rendered image looks like.

Some engines may be better at certain things than others due to the math they use or core principles around which they were written.

Blender includes two render engines by default:

- *Blender Render*
- *Cycles*

More render engines from third-party developers can also be added using *Add-ons*

2.9.2 Blender Render Engine

Materials

Introduction

A material defines the artistic qualities of the substance that an object is made of. In its simplest form, you can use materials to show the substance an object is made of, or to "paint" the object with different colors. Usually, the substance is represented by its surface qualities (color, shininess, reflectance, etc.) but it can also exhibit more complicated effects such as transparency, diffraction and sub-surface scattering. Typical materials might be brass, skin, glass, or linen.

The basic (un-textured) Blender material is uniform across each face of an object (although the various pixels of each face of the object may appear differently because of lighting effects). However, different faces of the object may use different materials (see *Multiple Materials*).

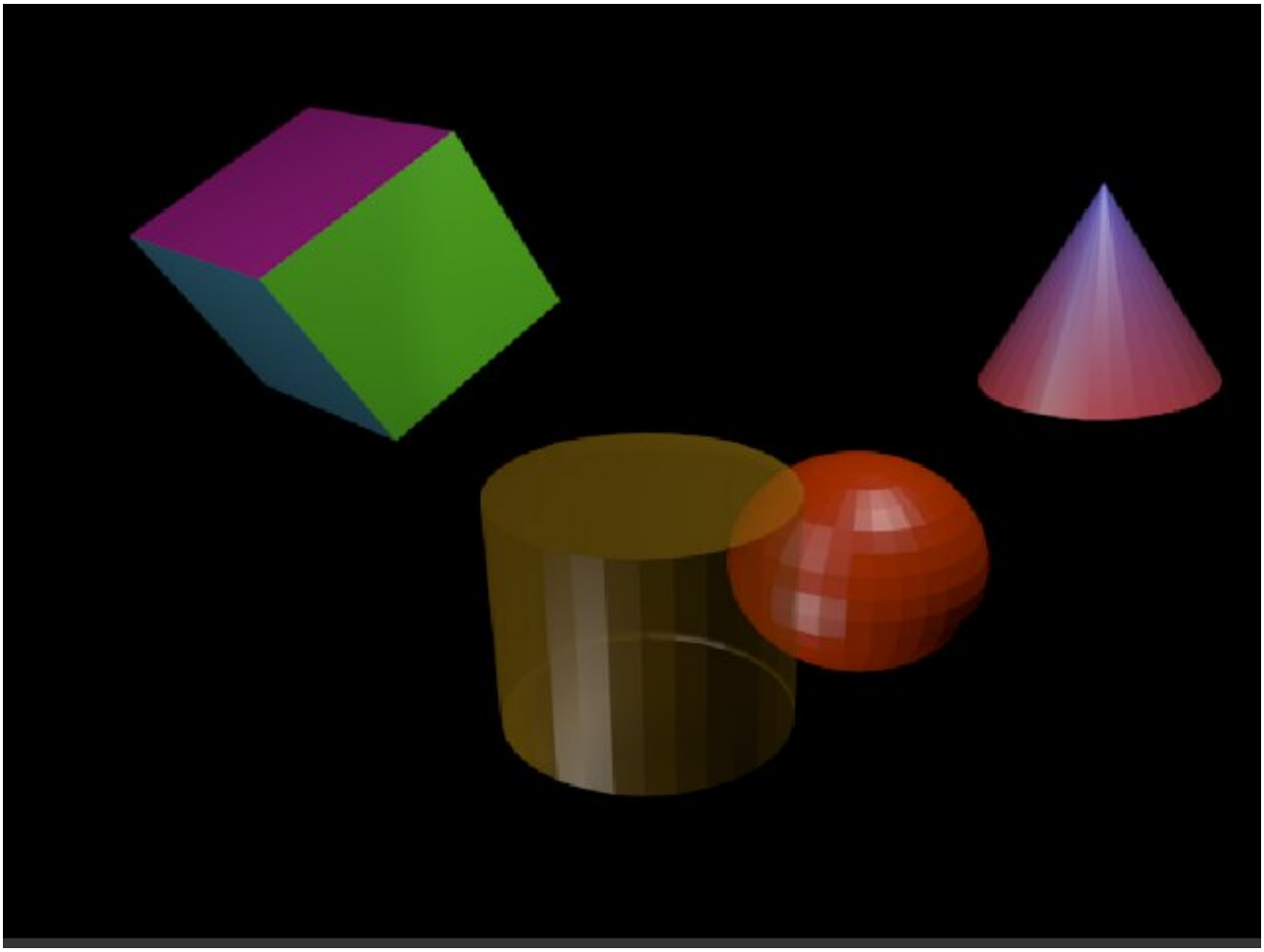


Fig. 2.1427: Various basic materials (single, multiple material, transparency, vertex paint).

In Blender, materials can (optionally) have associated textures. Textures *describe* the substance: e.g. *polished* brass, *dirty* glass or *embroidered* linen. The *Textures* chapter describes how to add textures to materials.

How Materials Work

Before you can understand how to design effectively with materials, you must understand how simulated light and surfaces interact in Blender's rendering engine and how material settings control those interactions. A deep understanding of the engine will help you to get the most from it.

The rendered image you create with Blender is a projection of the scene onto an imaginary surface called the *viewing plane*. The viewing plane is analogous to the film in a traditional camera, or the rods and cones in the human eye, except that it receives simulated light, not real light.

To render an image of a scene we must first determine what light from the scene is arriving at each point on the viewing plane. The best way to answer this question is to follow a straight line (the simulated light ray) backwards through that point on the viewing plane and the focal point (the location of the camera) until it hits a renderable surface in the scene, at which point we can determine what light would strike that point.

The surface properties and incident light angle tells how much of that light would be reflected back along the incident viewing angle *(Rendering engine basic principle)*.

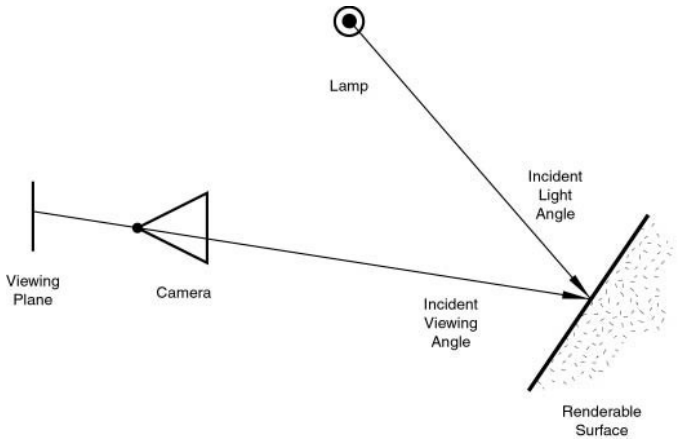


Fig. 2.1428: Rendering engine basic principle.

Two basic types of phenomena take place at any point on a surface when a light ray strikes it: diffusion and specular reflection. Diffusion and specular reflection are distinguished from each other mainly by the relationship between the incident light angle and the reflected light angle.

The shading (or coloring) of the object during render will then take into account the base color (as modified by the diffusion and specular reflection phenomenon) and the light intensity.

Using the internal ray tracer, other (more advanced) phenomena could occur. In ray-traced reflections, the point of a surface struck by a light ray will return the color of its surrounding environment, according to the rate of reflection of the material (mixing the base color and the surrounding environment's) and the viewing angle.

On the other hand, in ray-traced refractions, the point of a surface struck by a light ray will return the color of its background environment, according to the rate of transparency (mixing the base color and the background environment's along with its optional filtering value) of the material and the optional index of refraction of the material, which will distort the viewing angle.

Of course, shading of the object hit by a light ray will be about mixing all these phenomena at the same time during the rendering. The appearance of the object, when rendered, depends on many inter-related settings:

- World (Ambient color, Radiosity, Ambient Occlusion)
- Lights
- Material settings (including ambient, emission, and every other setting on every panel in that tab)

- Texture(s) and how they are mixed
- Material Nodes
- Camera
- Viewing angle
- Obstructions and transparent occlusions
- Shadows from other opaque/transparent objects
- Render settings
- Object dimensions (SS settings are relevant to dimensions)
- Object shape (refractions, Fresnel effects)

Using Materials

Tip: Check your Render

When designing materials (and textures and lighting), frequently check the rendered appearance of your scene, using your chosen render engine/shader settings. The appearance might be quite different from that shown in the texture display in the 3D panel.

As stated above, the material settings usually determine the surface properties of the object. There are several ways in which materials can be set up in Blender. Generally speaking, these are not compatible. You must choose which method you are going to use for each particular object in your scene:

1. First, you can set the *Properties* in the various Material panels.
2. Second, you can use *Nodes*; a graphical nodes editor is available.
3. Last, you can directly set the color of object surfaces using various special effects. Strictly speaking, these are not materials at all, but they are included here because they affect the appearance of your objects. These include *Vertex Painting*, *Wire Rendering*, *Volume Rendering*, and *Halo Rendering*.

The exact effect of Material settings can be affected by a number of system settings. First and foremost is the Render Engine used: Cycles and the Blender Render Engine (aka Blender Internal or BI) require quite different illumination levels to achieve similar results, and even then the appearance of objects can be quite different. Also, the material properties settings can be affected by the texture method used (Single Texture, Multitexture or GLSL). So it is recommended to always select the appropriate system settings before starting the design of materials.

Material Panel

Materials can be linked to objects and Object's data in the *materials tab* → *materials panel*. Here is where you can manage how materials are linked to objects, meshes, etc. and activate a material for editing in the rest of the panels.

Material slots

Active Material The objects material slots displayed in a *List View*.

Specials Copy and paste the selected material slot.

Multiple materials

Meshes can handle having more than one material. Materials can be mapped on a per-face basis, as detailed on the *Multiple Materials* page. In edit mode, the following tools appear:

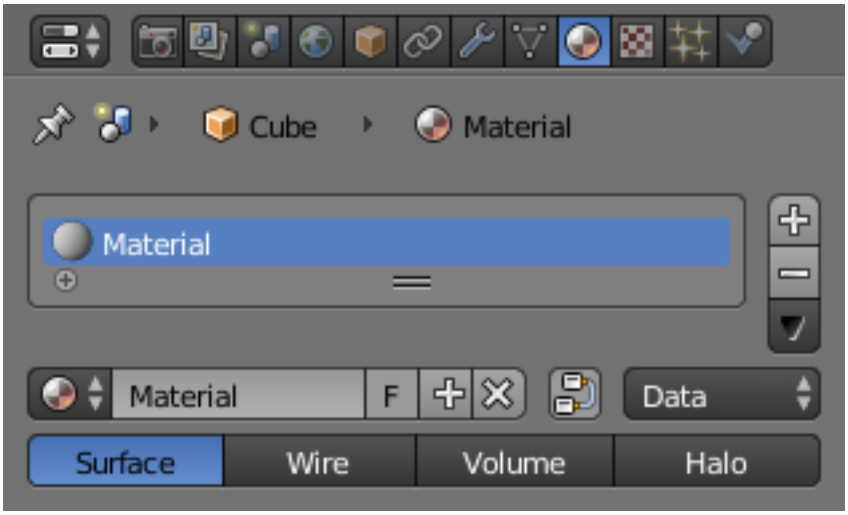


Fig. 2.1429: Material panel.

Assign Assign the material in the selected material slot to selected vertices.

Select Select vertices assigned to the selected material slot.

Deselect Deselect vertices assigned to the selected material slot.

Material naming and linking

Material The Material *Data-Block Menu* for the selected material slot.

Nodes Toogle  that designates this material to be a material node setup, and not from the Material/Ramps/Shaders settings.

Data-block Links Specifies whether the material is to be linked to the Object or to the Object Data.

The Link selector has two choices, Data and Object. These two menu choices determine whether the material is linked to the object or to the data, (i.e. a mesh or curve). The Data menu item determines that this material will be linked to the mesh's data-block which is linked to the object's data-block. The Object menu item determines that the material will be linked to the object's data-block directly.

This has consequences of course. For example, different objects may share the same mesh data-block. Since this data-block defines the shape of the object any change in edit mode will be reflected on all of those objects. Moreover, anything linked to that mesh data-block will be shared by every object that shares that mesh. So, if the material is linked to the mesh, every object will share it.

On the other hand, if the material is linked directly to the object data-block, the objects can have different materials and still share the same mesh.

Short explanation: If connected to the object, you can have several instances of the same Object Data using different materials. If linked to mesh data, you cannot. See *Data System* for more information.

Material type

Material added in edit mode These toggles tell Blender where this material fits into the Render Pipeline, and what aspects of the material are to be rendered.

Surface Render object as a surface.

Wire Render the edges of faces as wires (not supported in ray tracing).

Volume Render object as a volume. See *Volume*.

Halo Render object as halo particles. See *Halo*.

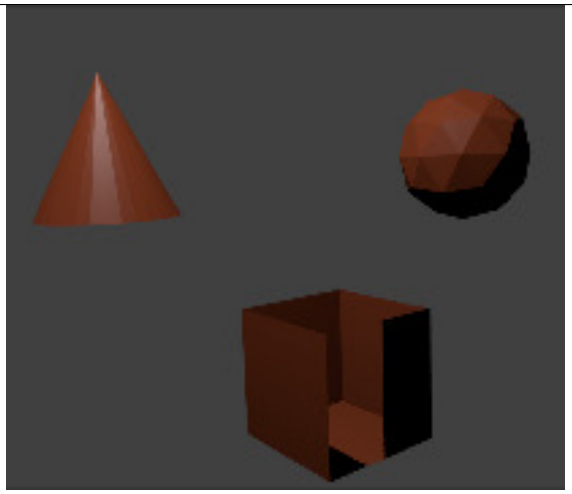


Fig. 2.1430: Surface.

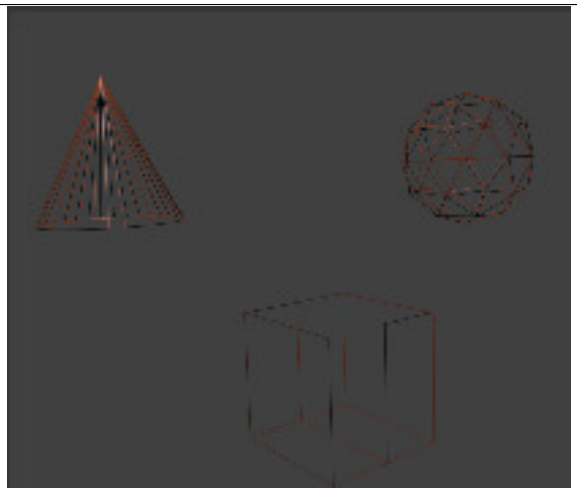


Fig. 2.1431: Wire.

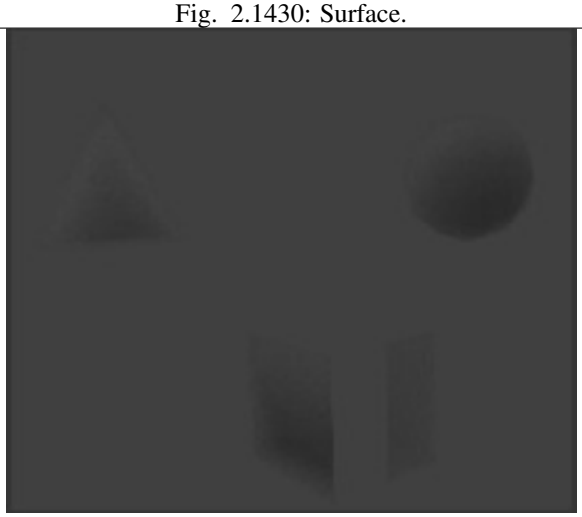


Fig. 2.1432: Volume.

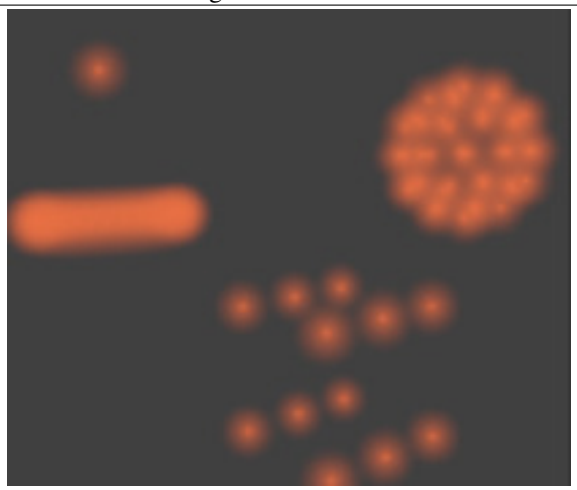


Fig. 2.1433: Halo.

Assigning a Material

Materials available in the currently-open blend-file can be investigated by clicking on the Materials icon in the Properties editor Header. In this section we look at how to assign or remove a material to/from the Active Object in Blender, either by:

- Creating a new material,
- re-using an existing material, or
- deleting a material.

We also give hints about practical material usage.

Creating a new Material

Every time a new Object is created it has no material linked to it. You can create a new material for the object by:

- Selecting the object.
- In the Properties editor, click on the object button.

- Click on the Materials button in the Properties editor header (1).

The Shading panel then appears. This contains the following elements:

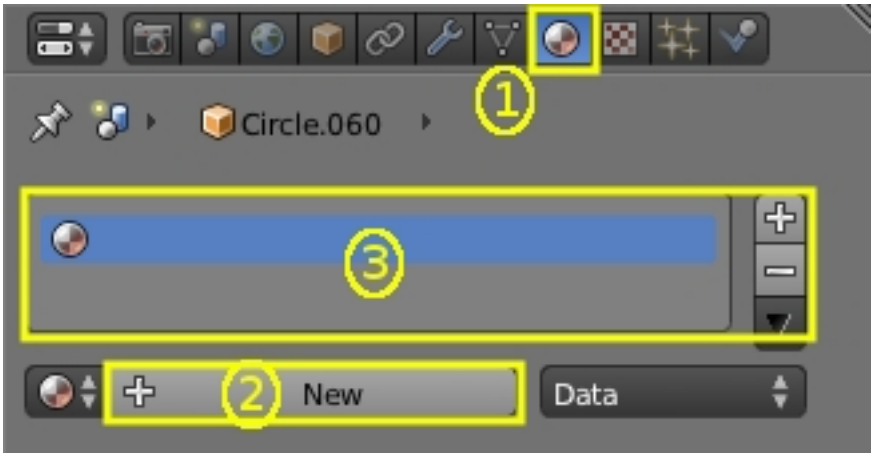


Fig. 2.1434: Add new material.

- Context: The currently-selected scene and object
- Object Material Slots (3): this panel shows the "slots" for the material (or materials) that this object data contains.
- Active Material (2): Initially empty, asking for "New".

To add a new material, click "+" in the Active Material box. This action has a series of effects:

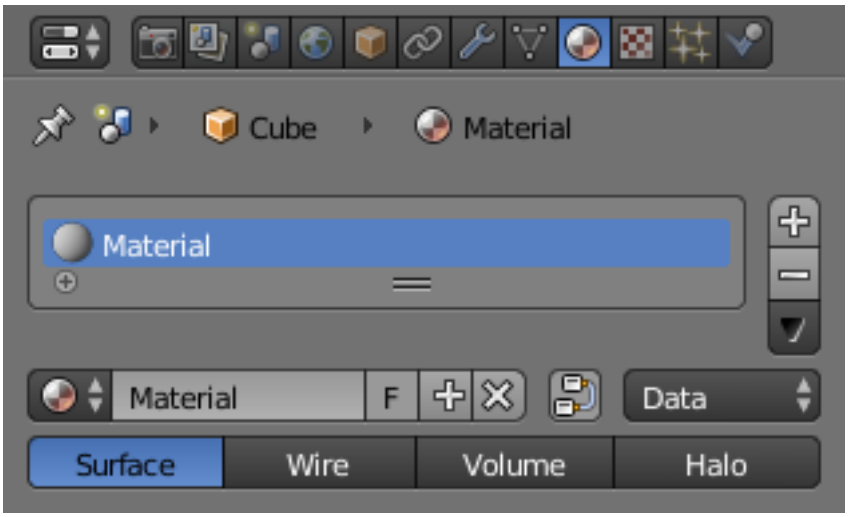


Fig. 2.1435: Materials Panel with New Entry.

1. Opens the new material in the Active Material box.
2. Brings up additional buttons in the immediate panel.
3. Adds the new material to the Available Materials list.
4. Adds the new material to the Object Material Slots list for the active object (or its object data – see below).
5. Brings up a *preview* of the new material.
6. Provides you with a range of panels allowing you to select the *properties* of the new material.

New Material Panel Buttons

Details of the additional buttons which appear in the Material panel for a new Active Material are as follows:

Active Material List View.

Material The Material *Data-Block Menu* for the selected Material slot.

Tip: Naming materials

It is a very good idea to give your materials clear names so you can keep track of them, especially when they are linked to multiple objects. Try to make your names descriptive of the material, not its function (e.g. "Yellow Painted" rather than "Kitchen Table Color").

Node Use Nodes.

Data Specifies whether the material is to be linked to the Object or to the Object Data.

Material type This menu has four options which define how the object is to be rendered.

Reusing Existing Materials

Blender is built to allow you to reuse *anything*, including material settings, between many objects. Instead of creating duplicate materials, you can simply re-use an existing material. There are several ways to do this using the Materials data-block menu:

Single Object – With the object selected, click the sphere located to the left of the Material name. A pop-up appears showing all the materials available in the current blend-file. To use one, just click on it.

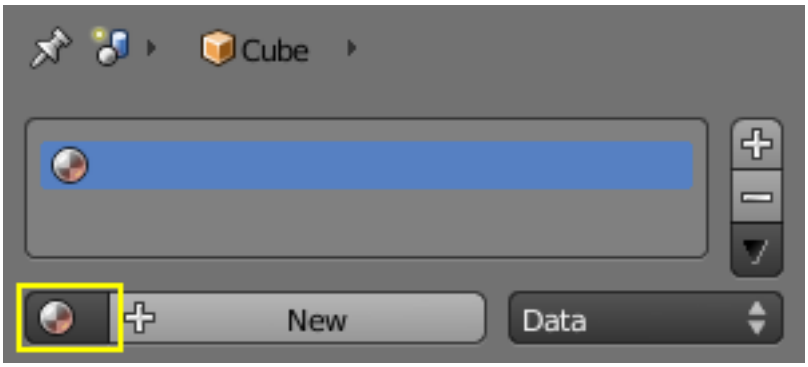


Fig. 2.1436: Select an existing material.

Tip: Searching for Materials

The search field at the bottom of the material list allows you to search the names in the list. For example, by entering "wood" all existent materials are filtered so that only materials containing "wood" are displayed in the list.

Multiple Objects – In the 3D View, with `Ctrl-L` you can quickly link all selected objects to the material (and other aspects) of the *active object*. Very useful if you need to set a large number of objects to the same material; just select all of them, then the object that has the desired material, and `Ctrl-L` link them to that "parent". (See Tip on Linking Data in Creating about data linking.)

Deleting a Material

To delete a material, select the material and click X in the Available Materials List entry.

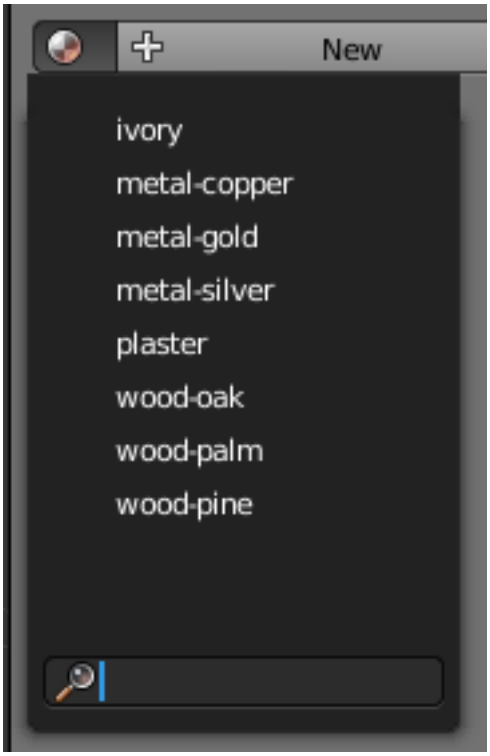


Fig. 2.1437: List of available materials.

Although the material will seem to disappear immediately, the Delete action can depend on how the material is used elsewhere.

If the material is linked to the Object and there are other objects which use this material, then the material will be removed from that object (but remain on all its other objects).

If the "Fake User" button (F) has been lit in the Available Materials list, then the material will be retained when the file is saved, even if it has no users.

Only if it has 0 "real" users, and no "Fake" user, will the material be permanently deleted. Note that it will still remain in the Materials list until the blend-file is saved, but will have disappeared when the file is reloaded.

Multiple Materials

Normally, different colors or patterns on an object are achieved by adding textures to your materials. However, in some applications you can obtain multiple colors on an object by assigning different materials to the individual faces of the object.

To apply several materials to different faces of the same object, you use the Material Slots options (3) in the Materials header panel.

The workflow for applying a second material to some faces of an object covered by a base material is as follows:

1. In Object Mode, apply the base material to the whole object (as shown in *Assigning a material*)
2. Create/select the second material (the whole object will change to this new material).
3. In the Active Material box (2), re-select the base material.
4. Go to Edit Mode and Face Select (a new box appears above the Active Material box with Assign/Select/Deselect).
5. Select the face/faces to be colored with the second material.
6. In the Object Material Slots box (3), click the `Plus` to create a new slot, and while this is still active, click on the second material in the Available Materials list.

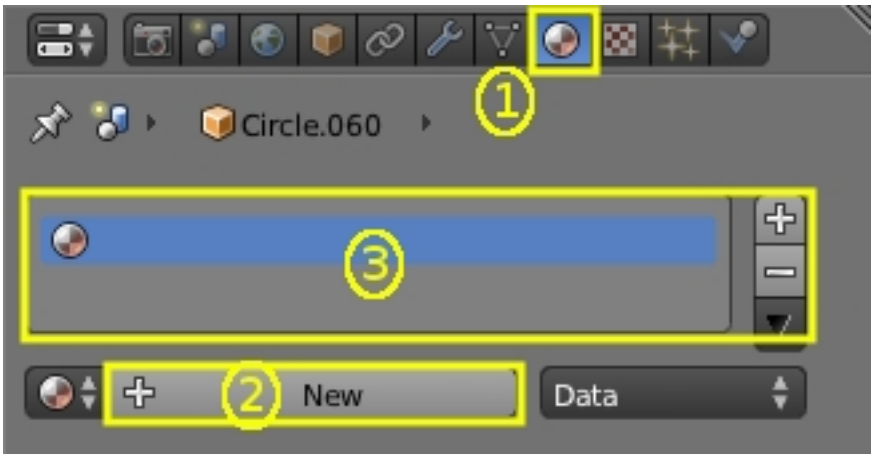


Fig. 2.1438: Add new material.

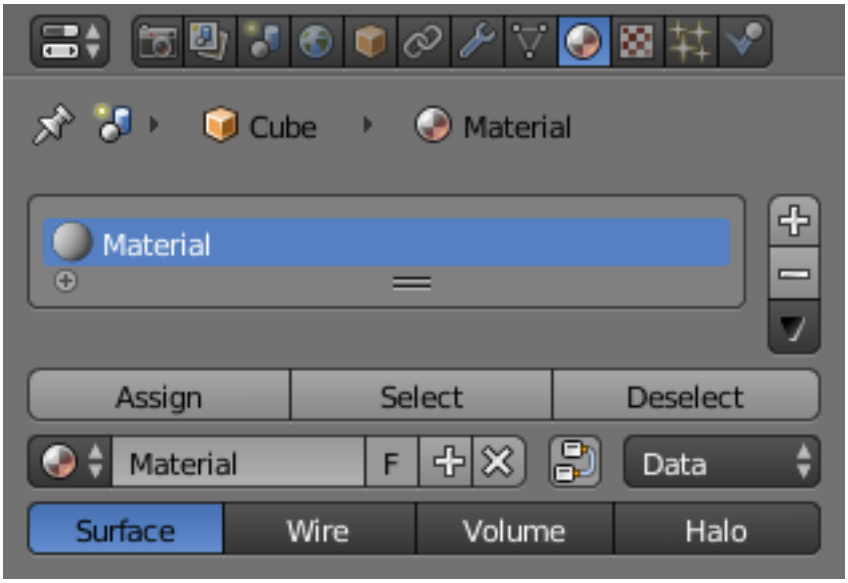


Fig. 2.1439: Material menu in edit mode.

7. Click the Assign button, and the second material will appear on the selected object faces.

You can also make this new material a copy of an existing material by adding the data-block:

Select the object, get the material, (R Click) and Copy data to clipboard. When you have renamed the material, click "Link: Data" to link to the existing material. Proceed to assign faces as required. NB: If you change the material on the original object, the new object color changes too.

Material Properties

Introduction

Materials can have a wide array of properties. It is the combination of all of these things that define the way a material looks, and how objects using that material will appear when rendered. These properties are set using the various panels in the material tab.

Remember that the appearance of your materials are affected by the way that they are rendered (surface, wire, volume or halo), and by the rendering engine (Blender, Cycles, or Game) used. Most properties for images rendered using Cycles can only be controlled using the Node system.

The list below sets out the various panels available in Blender Render and Game Engine, and brief details of their scope. Details of their controls and settings are given on the relevant pages.

Preview A preview of the current material mapped on to one of several basic objects.

Diffuse Shaders The basic color of the material, together with different models for dispersion.

Specular Shaders The reflected highlights: color, strength and different models for dispersion.

Color Ramps How to vary the base color over a surface in both Diffuse ans Specular shaders.

Shading Properties of various characteristics of the shading model for the material.

Transparency Sets options for objects in which light can pass through.

Mirror (Only Blender Render): Reflective properties of the material.

Subsurface Scattering (Only Blender Render): Simulates semi-translucent objects in which light enters, bounces around, then exits in a different place.

Strand (Only Blender Render): For use when surfaces are covered with hair, fur, etc.

Options Various options for shading and coloring the object.

Shadow Controls how objects using this material cast and receive shadows.

Game Settings (Only Blender Render): Controls settings for real-time rendering of Game Engine objects.

Preview

The Preview panel gives a quick visualization of the active material and its properties. Including its *Shaders*, *Ramps*, *Mirror Transp* properties and *Textures*. It provides several shapes that are very useful for designing new shaders: For some shaders (like those based on *Ramp* colors, or a Diffuse shader like *Minnaert*), one needs fairly complex or specific previewing shapes to decide if the shader being designed achieves its goal.

Options

Flat XY plane Useful for previewing textures and materials of flat objects, like walls, paper and such.

Sphere Useful for previewing textures and materials of sphere-like objects, but also to design metals and other reflective/transparent materials, thanks to the checkered background.

Cube Useful for previewing textures and materials of cube-like objects, but also to design procedural textures. Features a checkered background.

Monkey Useful for previewing textures and materials of organic or complex non-primitive shapes. Features a checkered background.

Hair strands Useful for previewing textures and materials of strand-like objects, like grass, fur, feathers and hair. Features a checkered background.

Large Sphere with Sky Useful for previewing textures and materials of sphere-like objects, but also to design metals and other reflective materials, thanks to the gradient Sky background.

Examples

Fig. 2.1440: Plane preview.

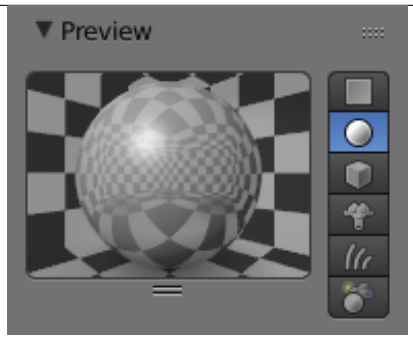


Fig. 2.1441: Sphere preview.

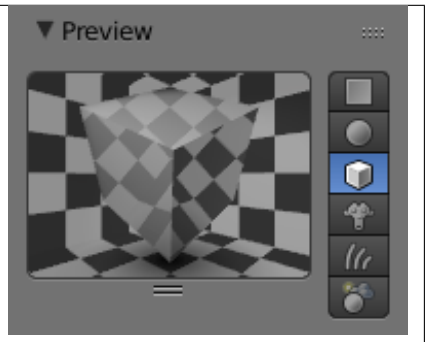


Fig. 2.1442: Cube preview.

Fig. 2.1443: Monkey preview.

Fig. 2.1444: Hair Strands preview.

Fig. 2.1445: Sky Sphere preview.

Diffuse Shaders

Reference

Mode: All Modes
Panel: *Shading/Material* → *Diffuse*

A diffuse shader determines, simply speaking, the general color of a material when light shines on it. Most shaders that are designed to mimic reality give a smooth falloff from bright to dark from the point of the strongest illumination to the shadowed areas, but Blender also has other shaders for various special effects.

Common Options

All diffuse shaders have the following options:

Color Select the base *diffuse color* of the material.

Intensity The shader's brightness, or more accurately, the amount of incident light energy that is actually diffusely reflected towards the camera.

Ramp Allows you to set a range of colors for the *Material*, and define how the range will vary over a surface. See *Color Ramps* for details.

Technical Details

Light striking a surface and then re-irradiated via a Diffusion phenomenon will be scattered, i.e., re-irradiated in all directions isotropically. This means that the camera will see the same amount of light from that surface point no matter what the *incident viewing angle* is. It is this quality that makes diffuse light *viewpoint independent*. Of course, the amount of light that strikes the surface depends on the incident light angle. If most of the light striking a surface is reflected diffusely, the surface will have a matte appearance (Light re-irradiated in the diffusion phenomenon).

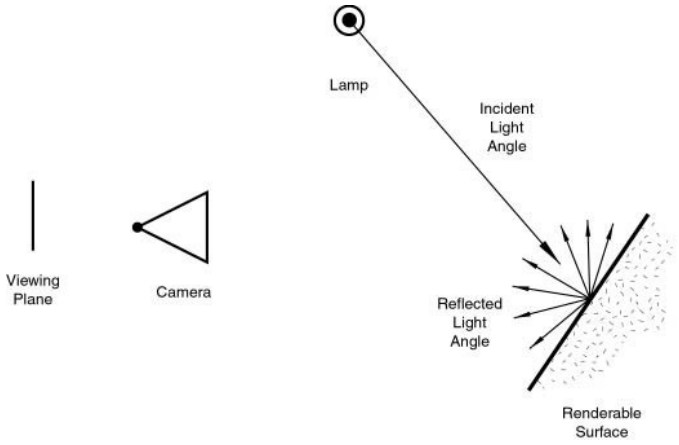


Fig. 2.1446: Light re-irradiated in the diffusion phenomenon.

Tip: Shader Names

Some shaders are – traditionally – named after the people, who first introduced the models on which they are based.

Lambert

Reference

Mode: All Modes
Panel: *Shading/Material* → *Shaders*

This is Blender's default diffuse shader, and is a good general all-around workhorse for materials showing low levels of specular reflection.

Johann Heinrich Lambert (1728-1777) was a Swiss mathematician, physicist and astronomer who published works on the reflection of light, most notably the Beer-Lambert Law which formulates the law of light absorption.

This shader has only the default option, determining how much of available light is reflected. Default is 0.8, to allow other objects to be brighter.

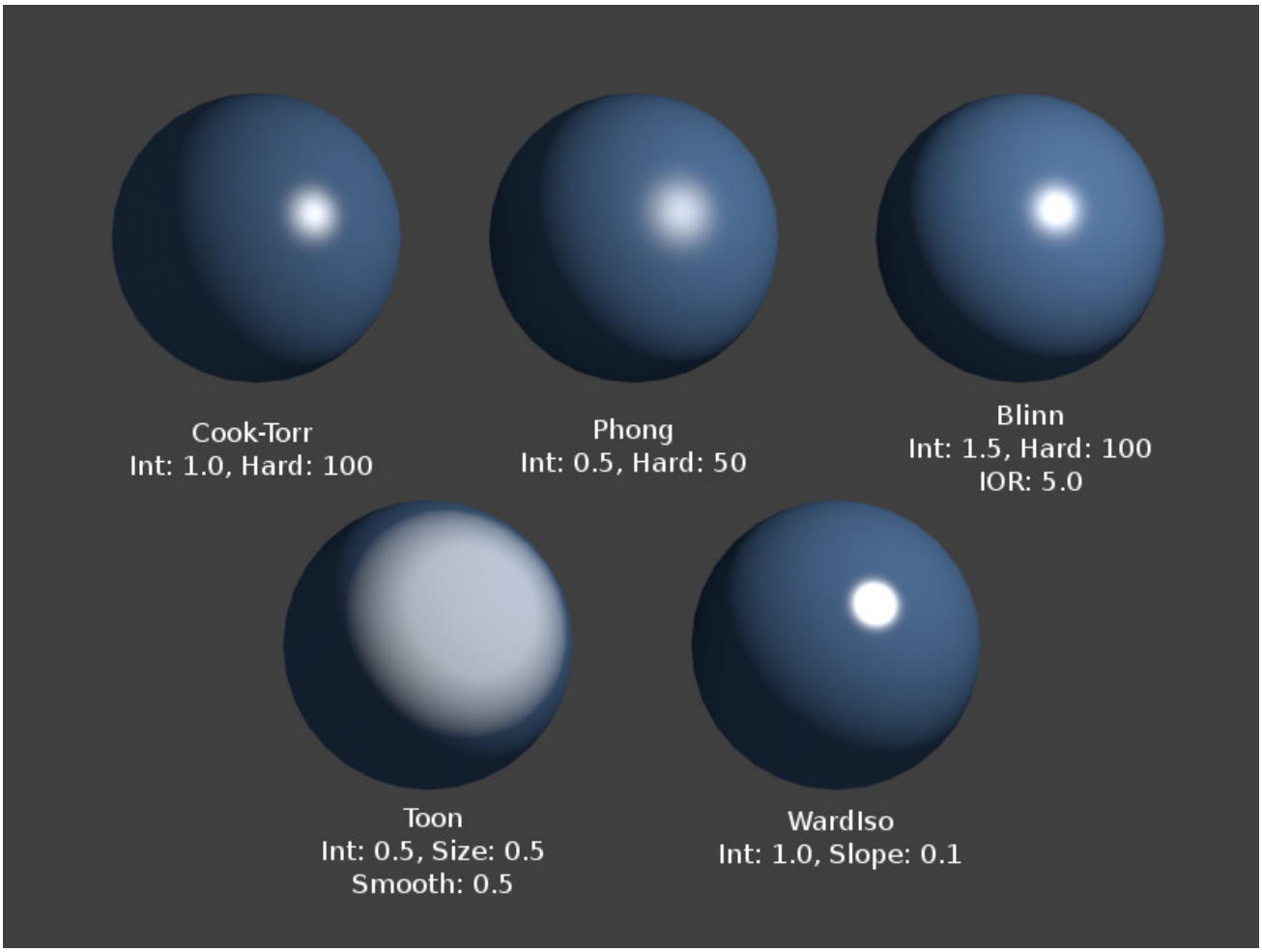


Fig. 2.1447: Lambert Shader.

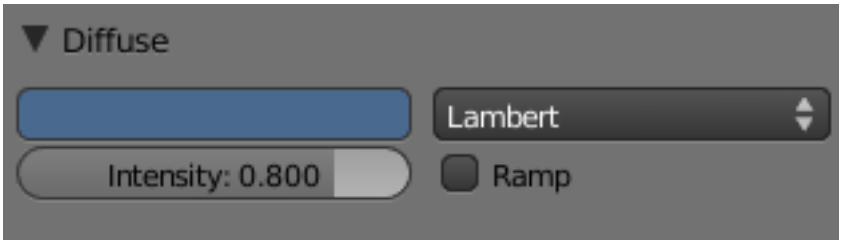


Fig. 2.1448: The Lambert diffuse shader settings.

Oren-Nayar

Reference

Mode: All Modes
Panel: *Shading/Material* → *Shaders*

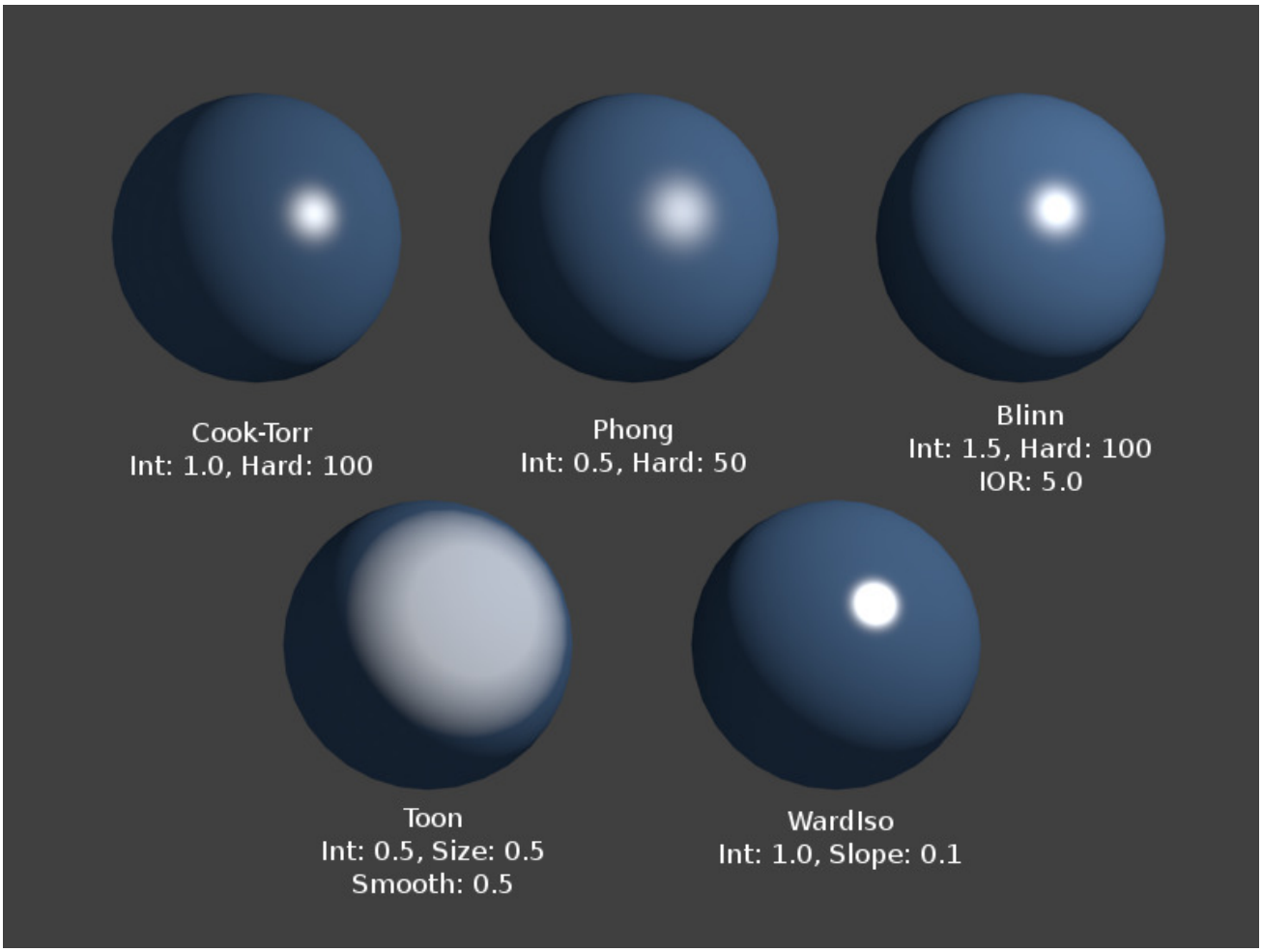


Fig. 2.1449: Oren-Nayar Shader.

Oren-Nayar takes a somewhat more 'physical' approach to the diffusion phenomena as it takes into account the amount of microscopic roughness of the surface. Michael Oren and Shree K. Nayar Their reflectance model, developed in the early 1990s, is a generalization of Lambert's law now widely used in computer graphics.

Options

Roughness The roughness of the surface, and hence, the amount of diffuse scattering.

Toon

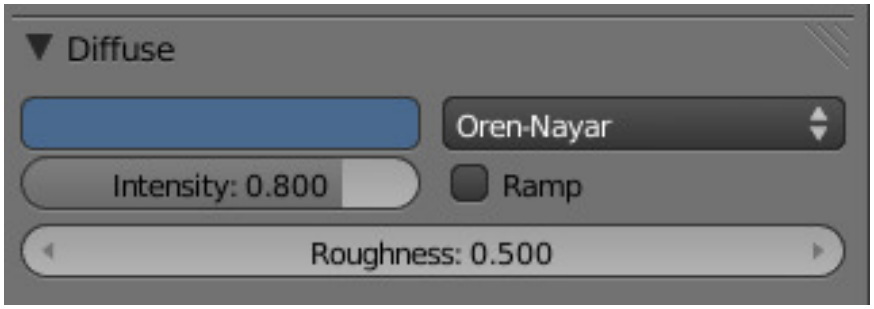


Fig. 2.1450: The Oren-Nayar diffuse shader settings.

Reference

Mode: All Modes
Panel: *Shading/Material* → *Shaders*

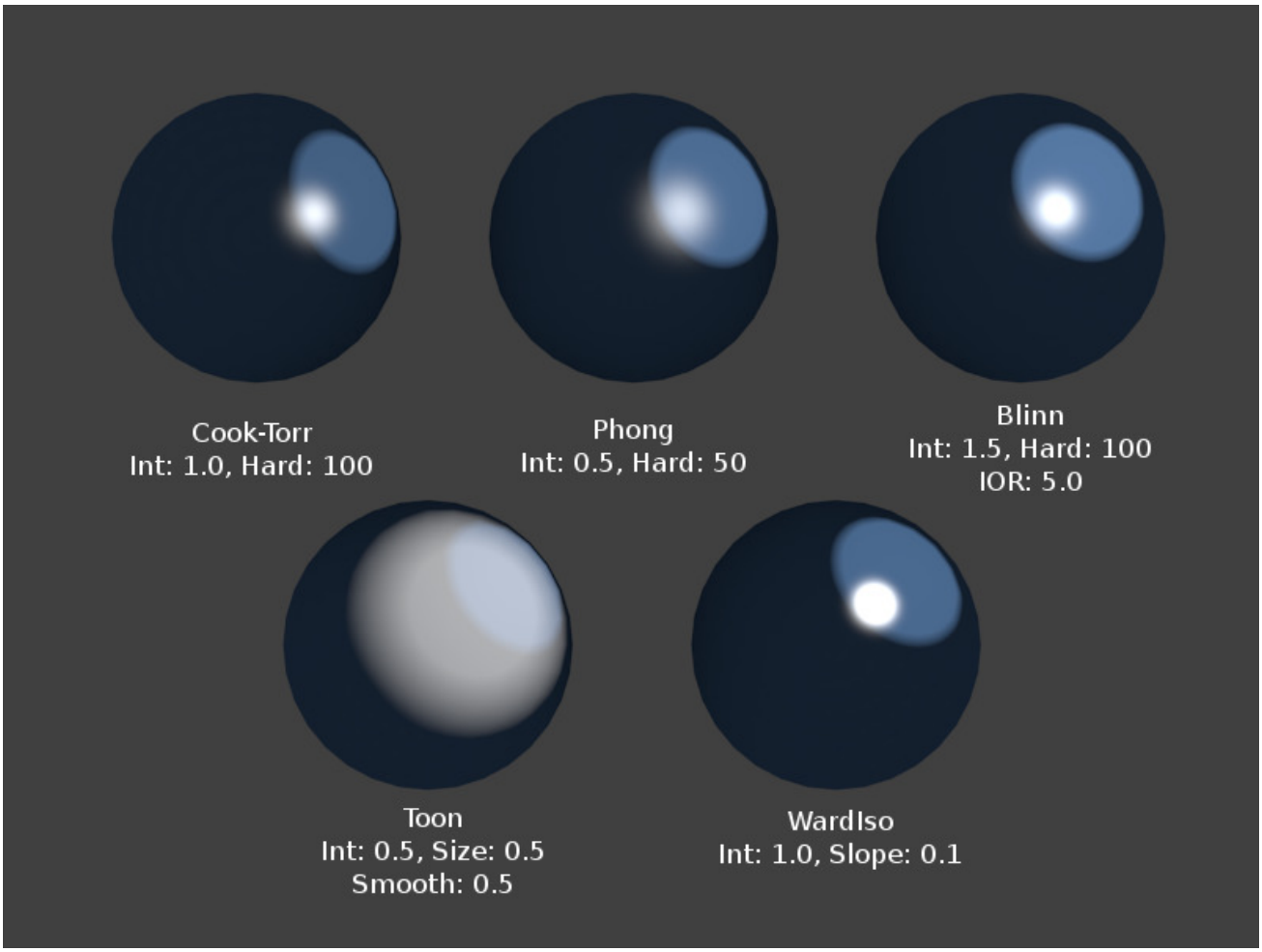


Fig. 2.1451: Toon Shader, Different Spec.

The Toon shader is a very 'un-physical' shader in that it is not meant to fake reality, but to produce cartoon cel styled rendering, with clear boundaries between light and shadow and uniformly lit/shadowed regions.

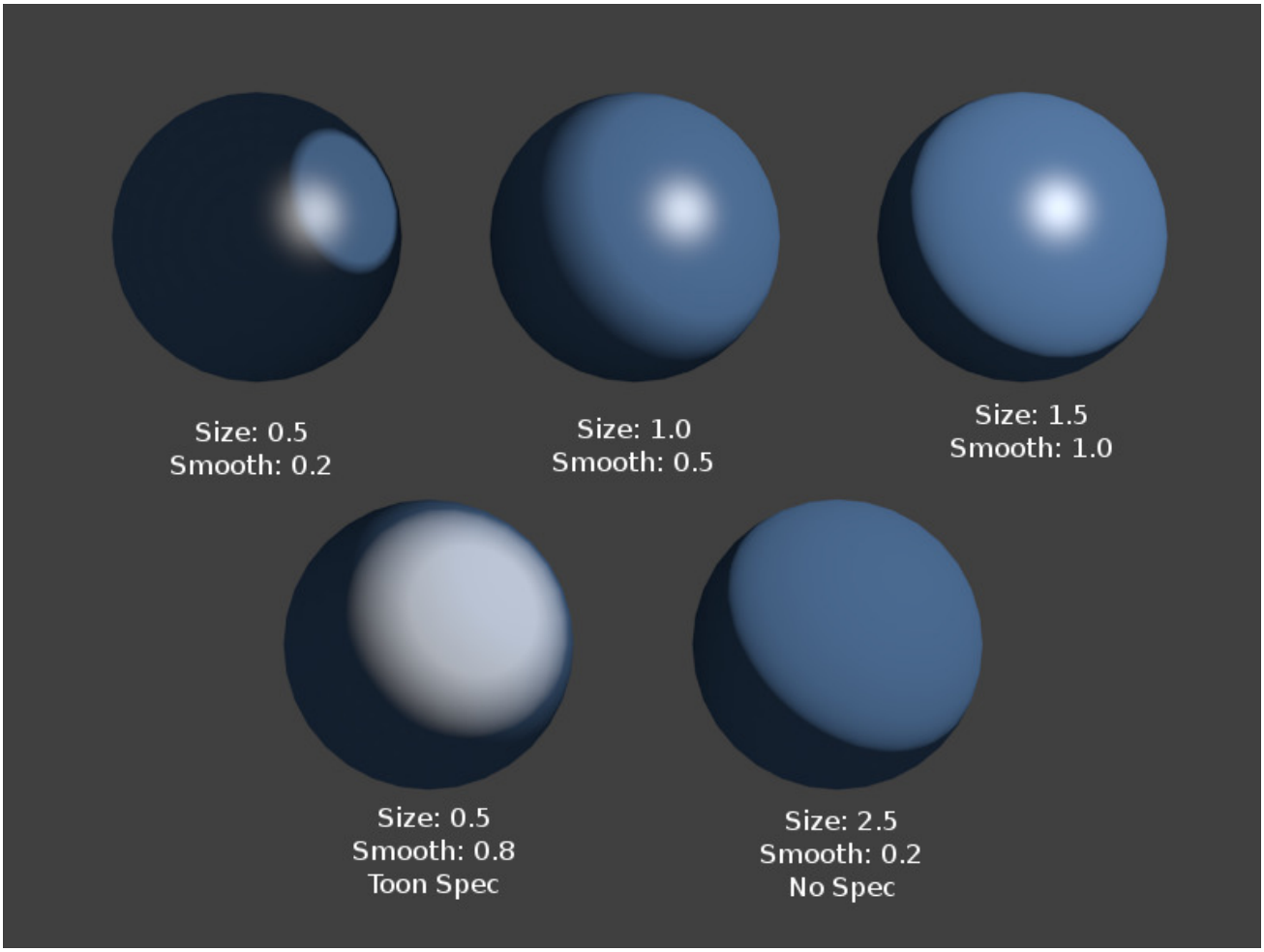


Fig. 2.1452: Toon Shader Variations.

Options

Size The size of the lit area.

Smooth The softness of the boundary between lit and shadowed areas.

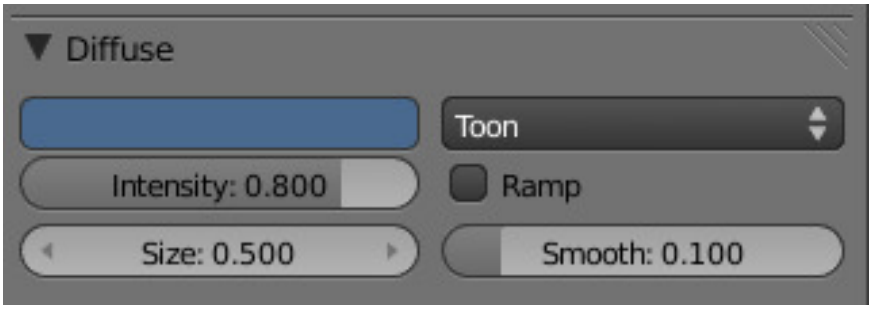


Fig. 2.1453: The Toon diffuse shader settings.

Minnaert

Reference

Mode: All Modes
Panel: *Shading/Material* → *Shaders*

Minnaert works by darkening parts of the standard Lambertian shader, so if *Dark* is 1 you get exactly the Lambertian result. Higher darkness values will darken the center of an object (where it points towards the viewer). Lower darkness values will lighten the edges of the object, making it look somewhat velvet. Marcel Minnaert (1893-1970) was a Belgian astronomer interested in the effects of the atmosphere on light and images who in 1954 published a book entitled "The Nature of Light and Color in the Open Air".

Options

Dark The darkness of the 'lit' areas (higher) or the darkness of the edges pointing away from the light source (lower).

Fresnel

Reference

Mode: All Modes
Panel: *Shading/Material* → *Shaders*

With a Fresnel shader the amount of diffuse reflected light depends on the incidence angle, i.e. from the direction of the light source. Areas pointing directly towards the light source appear darker; areas perpendicular to the incoming light become brighter. Augustin-Jean Fresnel (1788-1827) was a French physicist who contributed significantly to the establishment of the theory of wave optics.

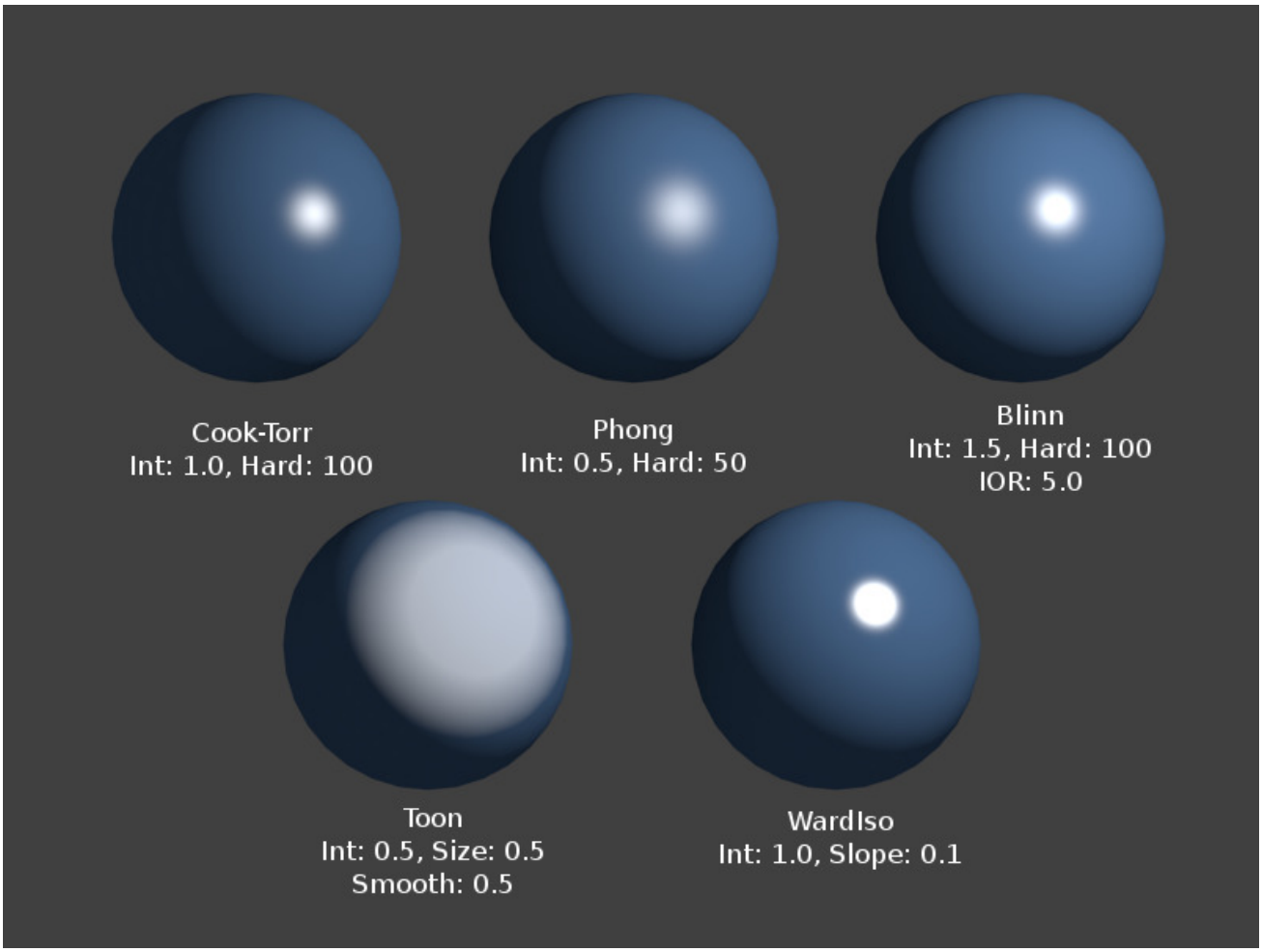


Fig. 2.1454: Minnaert Shader.

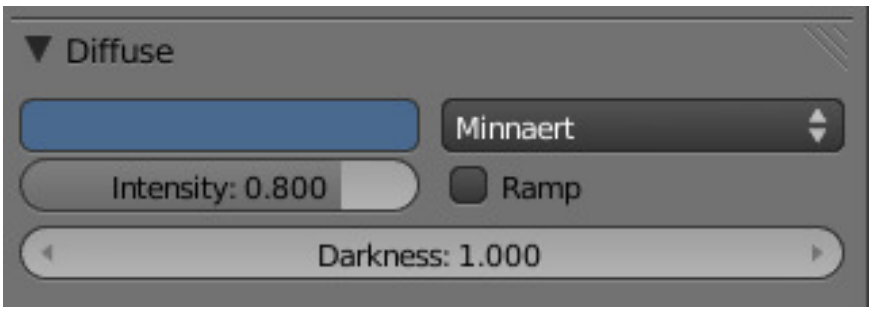


Fig. 2.1455: The Minnaert diffuse shader settings.

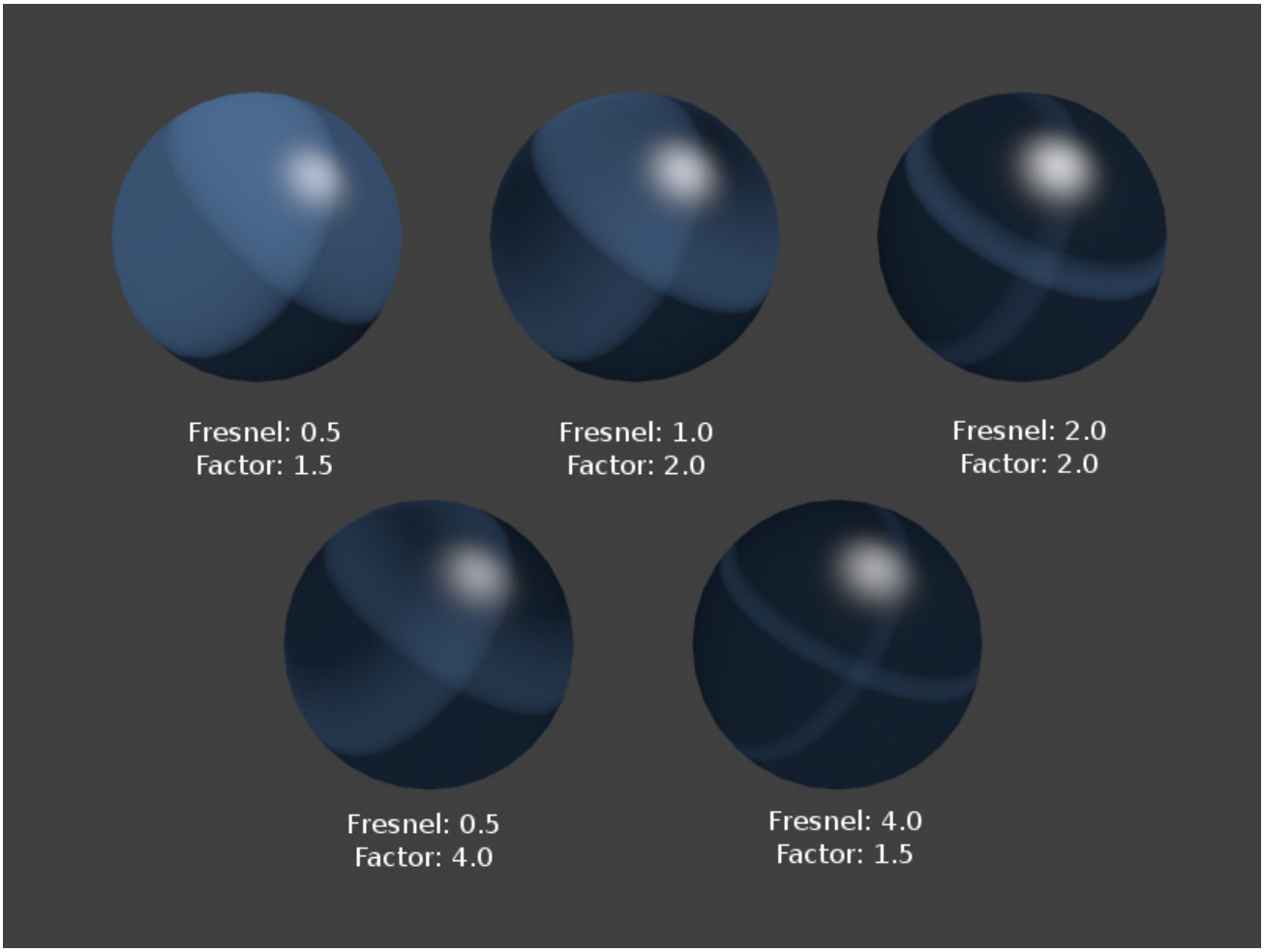


Fig. 2.1456: Various settings for the Fresnel shader, Cook-Torr Specular shader kept at Intensity 0.5, Hardness: 50.

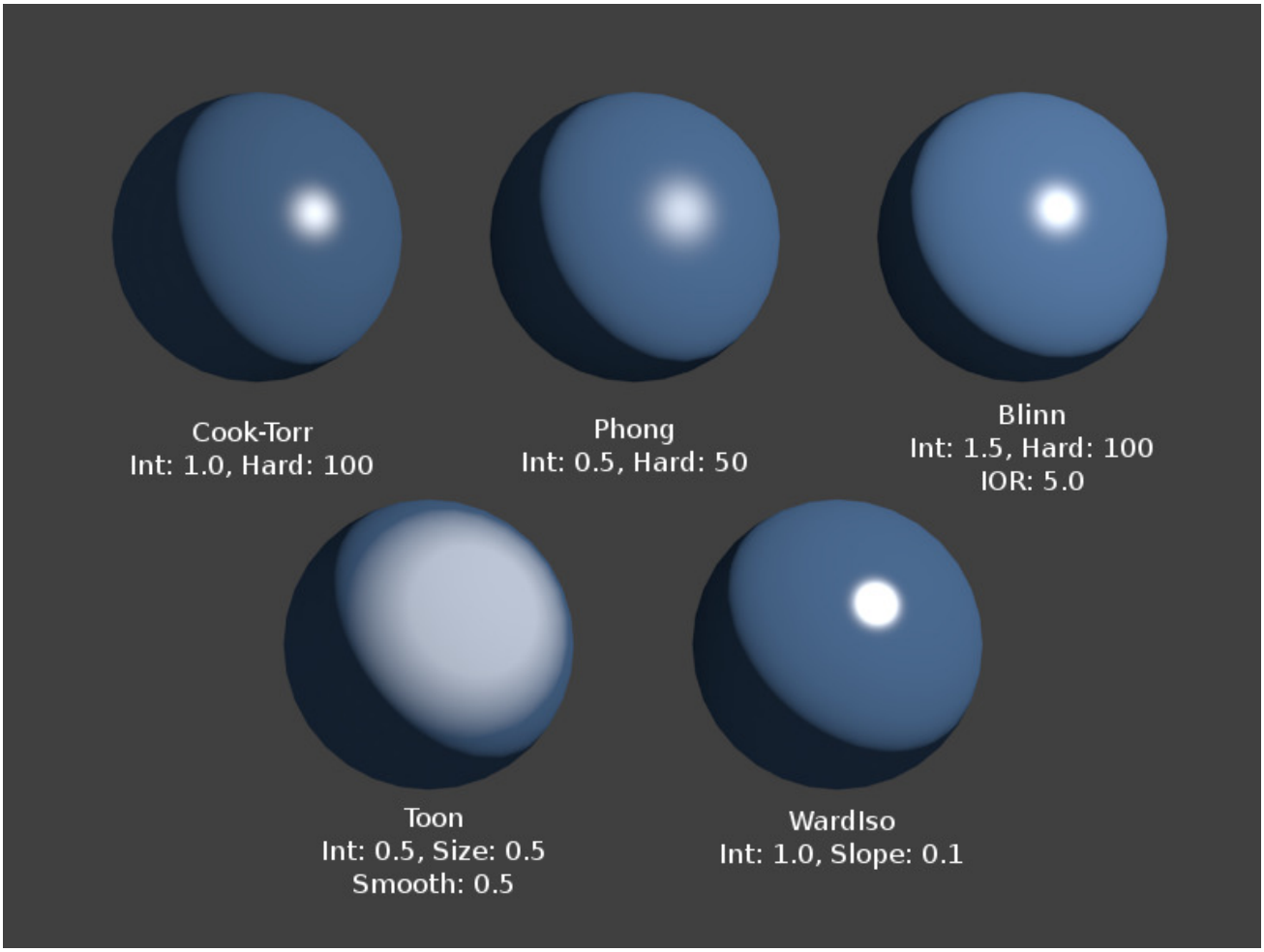


Fig. 2.1457: Fresnel Shader, Different Spec.

Options

Fresnel Power of the Fresnel effect, 5.0 is max.

Factor Blending factor of the Fresnel factor to blend in, 5.0 is max.

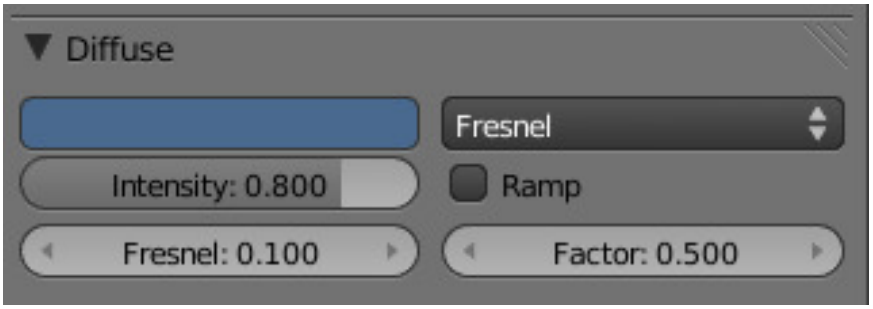


Fig. 2.1458: The Fresnel diffuse shader settings.

Emit Amount of light to emit

Ambient Amount of global ambient color the material receives

Translucency Amount of diffuse shading on the back side

Shadeless Make this material insensitive to light or shadow

Tangent Shading Use the material's tangent vector instead of the normal for shading – for anisotropic shading effects (e.g. soft hair and brushed metal).

See also:

Settings for strand rendering in the menu further down and in the Particle System menu.

Cubic Interpolation Use cubic interpolation for diffuse values, for smoother transitions between light areas and dark areas.

Specular Shaders

Reference

Mode: All Modes
Panel: *Material* → *Specular*

Specular shaders create the bright highlights that one would see on a glossy surface, mimicking the reflection of light sources. Unlike *diffuse shading*, specular reflection is *viewpoint dependent*. According to Snell's Law, light striking a specular surface will be reflected at an angle which mirrors the incident light angle (with regard to the surface's normal), which makes the viewing angle very important.

Tip: Not a Mirror!

It is important to stress that the *specular reflection* phenomenon discussed here is not the reflection we would see in a mirror, but rather the light highlights we would see on a glossy surface. To obtain true mirror-like reflections you would need to use the internal raytracer. Please refer to section *rendering* of this manual.

Common Options

Each specular shader share the following common options:

Specular Color The color of the specular highlight

Intensity The intensity, or brightness of the specular highlight. This has a range of [0-1].

Ramp Allows you to set a range of specular colors for *Material*, and define how the range will vary over a surface. See *Ramps* for details.

As a result, a material has at least two different colors, a diffuse, and a specular one. The specular color is normally set to pure white (the same "pure white" as the reflected light source), but it can be set to different values for various effects (e.g. metals tend to have colored highlights).

Technical Details

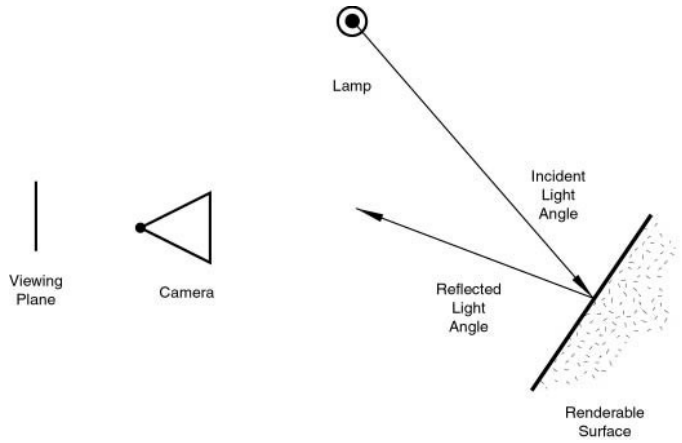


Fig. 2.1459: Specular Reflection.

In reality, the quality of Diffuse and Specular reflection are generated during the same process of light scattering, but are not the same. Diffusion is actually subsurface scattering at a very small scale.

Imagine that a surface is made up of extremely microscopic semi-transparent, reflective facets. The sharpness of Specular reflection is determined by the distribution of the angle of these microfacets on the surface of an object. The more deep and jagged these facets are, the more the light spreads when it hits the surface. When these facets are flatter against the "macrosurface", the surface will have a tighter reflection, closer to a mirror. This is a condensed explanation of the generally accepted microfacet theory of reflectance, which is the basis of all modern BRDFs (Bi-directional Reflectance Distribution Functions), or shading models.

Because these microfacets are transparent, some light that hits them travels into the surface and diffuses. The light that makes it back out is roughly Lambertian most of the time, meaning that it spreads evenly in all directions. It is also attenuated by the pigmentation in the surface, hence creating what we perceive as diffuse, and the color of an object.

Note that at glancing angles, the reflectivity of a surface will always go to 1.

If it is difficult for you to understand this relationship, try to imagine a ball (say, of centimeter scale): if you throw it against a wall of raw stones (with a scale of roughness of a decimeter), it will bounce in a different direction each time, and you will likely quickly lose it! On the other hand, if you throw it against a smooth concrete wall (with a roughness of, say, a millimeter scale), you can quite easily anticipate its bounce, which follow (more or less!) the same law as the light reflection.

Cook-Torrance

Reference

Mode: All Modes
Panel: *Material* → *Shaders*

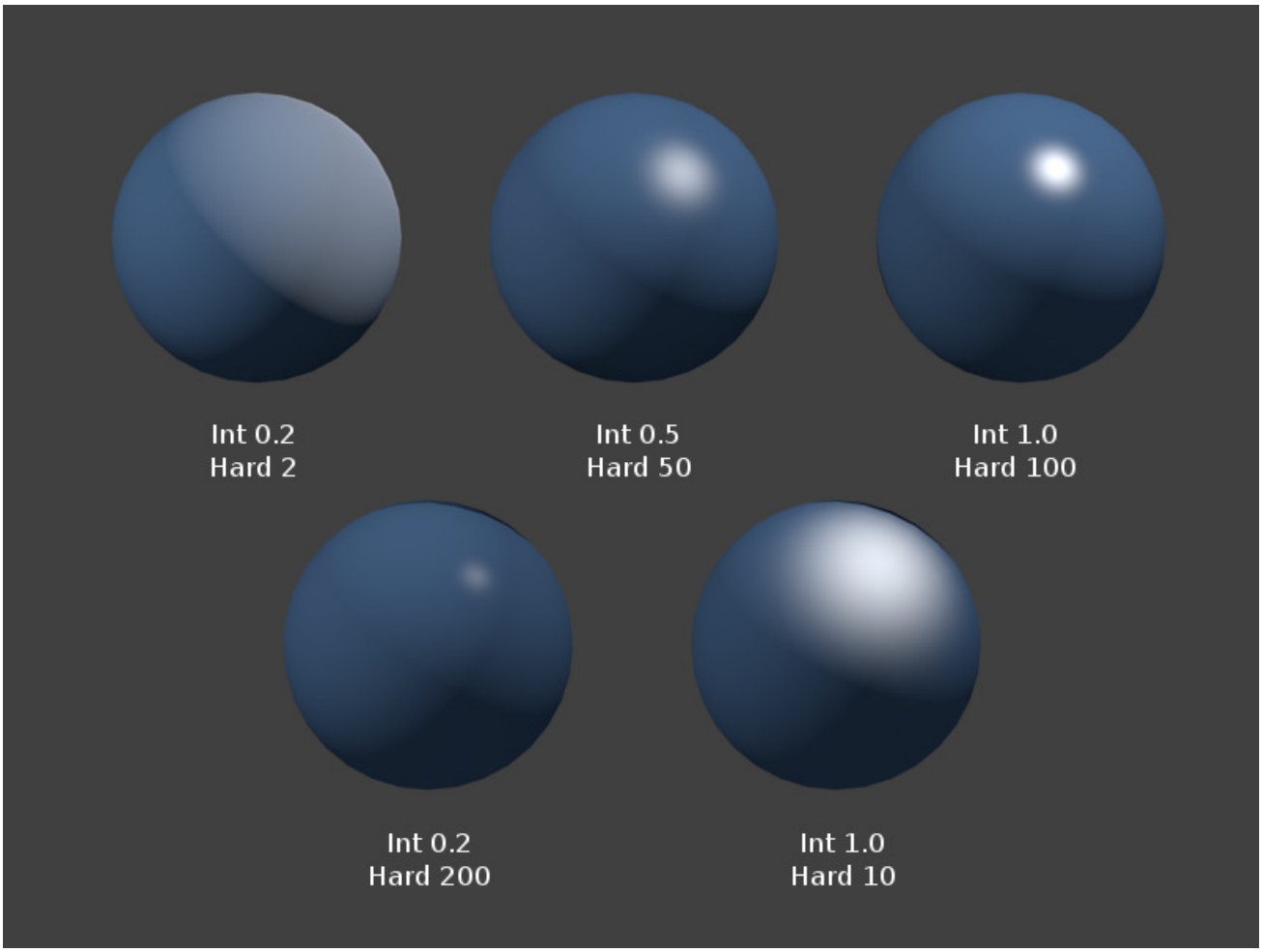


Fig. 2.1460: Cook-Torrance Shader (Lambert 0.8).

Cook-Torrance is a basic specular shader that is most useful for creating shiny plastic surfaces. It is a slightly optimized version of Phong. Robert L. Cook (LucasFilm) and Kenneth E. Torrance (Cornell University) In their 1982 paper A Reflectance Model for Computer Graphics (PDF), they described "a new reflectance model for rendering computer synthesized images" and applied it to the simulation of metal and plastic.

Options

Hardness Size of the specular highlight

Phong

Reference

Mode: All Modes
Panel: *Material* → *Shaders*

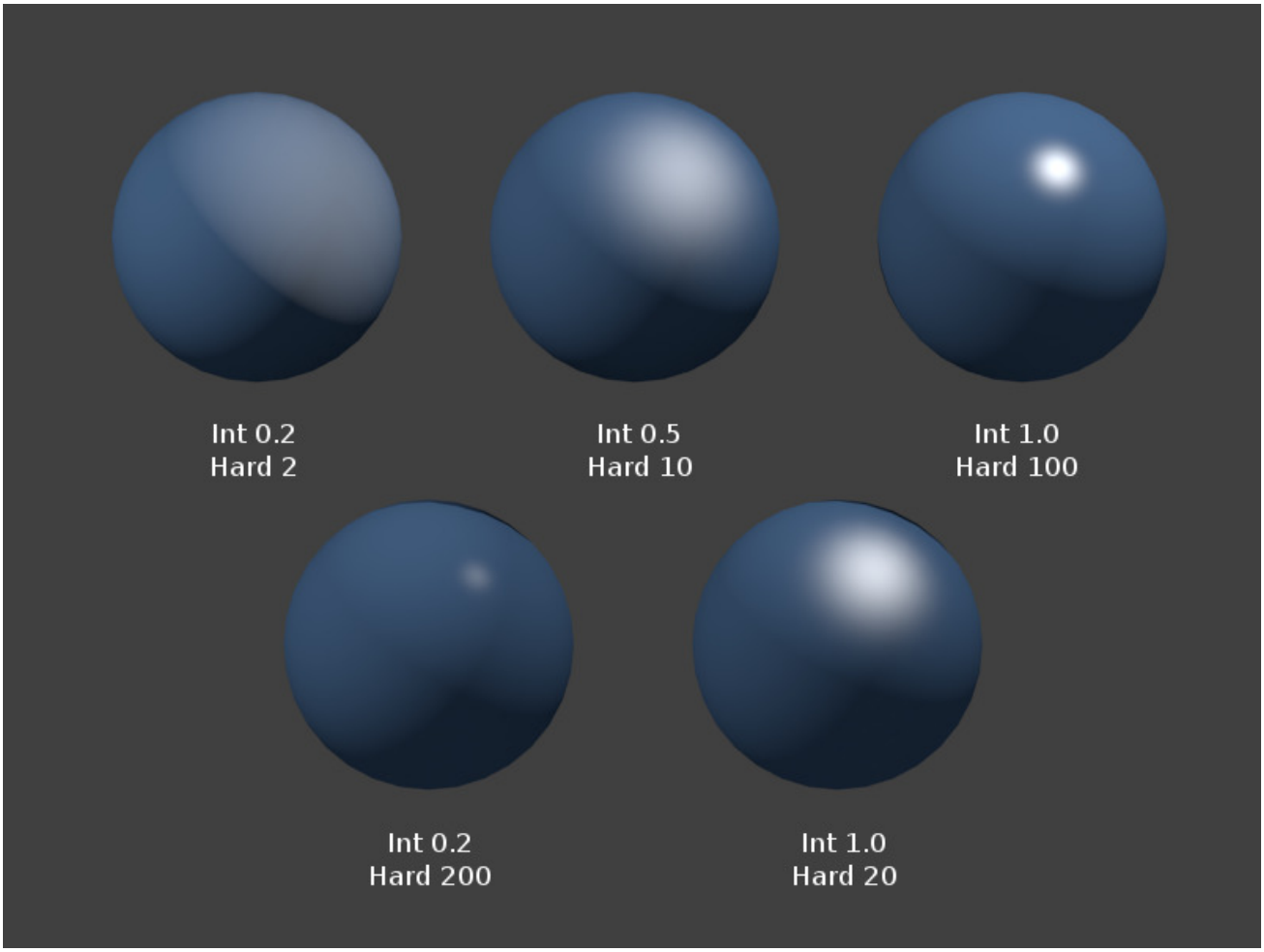


Fig. 2.1461: Phong Shader (Lambert 0.8).

Phong is a basic shader that is very similar to CookTorr, but is better for skin and organic surfaces. Bui Tuong Phong (1942-1975) was a Vietnamese-born computer graphics pioneer that developed the first algorithm for simulating specular phenomenon. His model included components not only for specular lighting, but also diffuse and ambient lighting.

Options

Hardness Size of the specular highlight.

Tip: Planet Atmosphere

Because of its fuzziness, this shader is good for atmosphere around a planet. Add a sphere around the planet, slightly larger than the planet. For its material, use a phong specular shader. Set it to a very low alpha (.05), zero diffuse, low hardness (5) but high specularity (1).

Blinn

Reference

Mode: All Modes
Panel: *Material* → *Shaders*

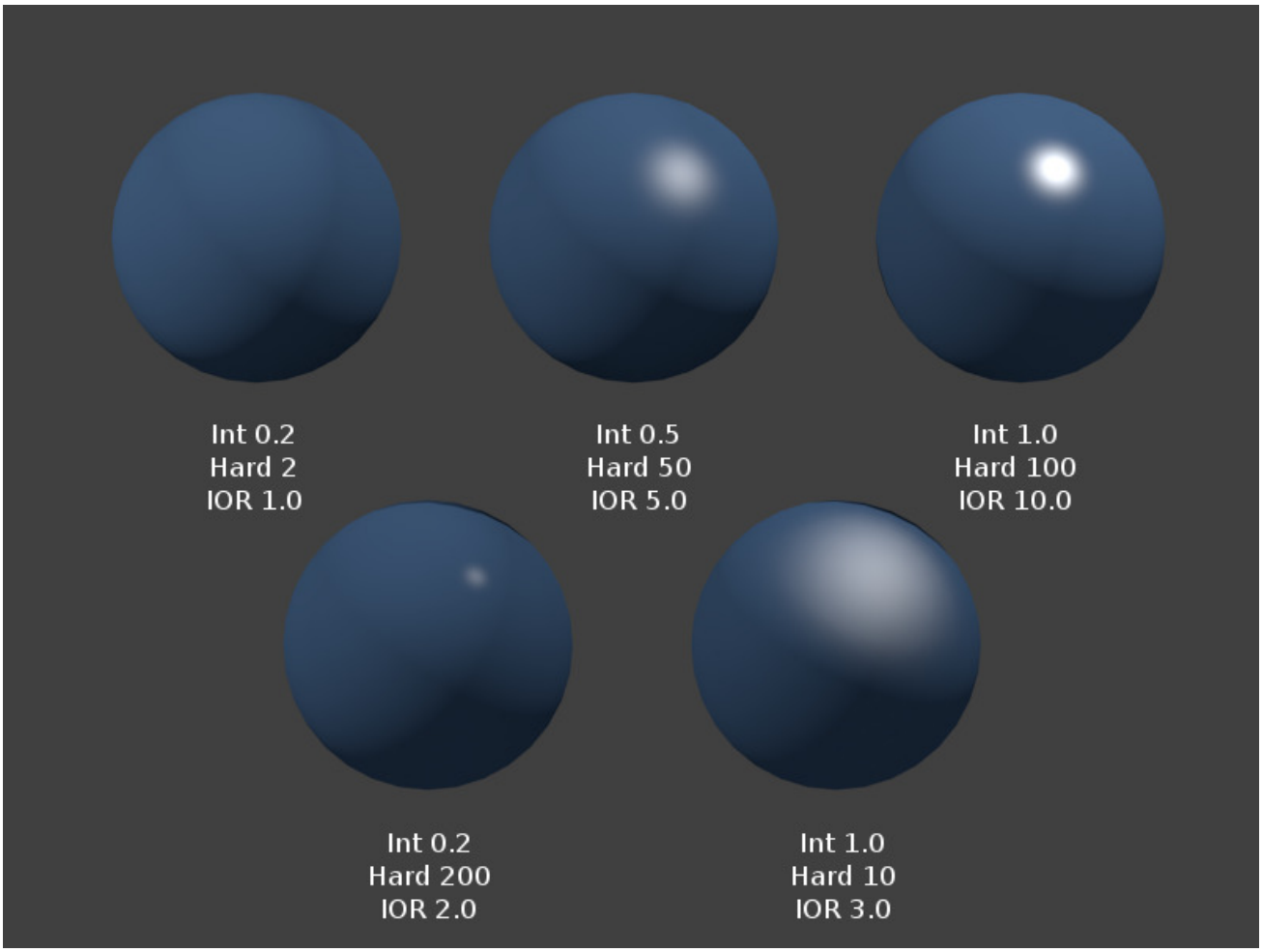


Fig. 2.1462: Blinn Shader (Oren-Nayar Int 0.8, Rough 0.5).

Blinn is a more 'physical' specular shader, often used with the Oren-Nayar diffuse shader. It can be more controllable because it adds a fourth option, an *index of refraction*, to the aforementioned three. James F. Blinn worked at NASA's Jet Propulsion Laboratory and became widely known for his work on Carl Sagan's TV documentary *Cosmos*. The model he described in his 1977 paper Models of Light Reflection for Computer Synthesized Pictures (PDF) included changes in specular intensity with light direction and more accurately positioned highlights on a surface.

Options

Hardness Size of the specular highlight. The Blinn shader is capable of much tighter specular highlights than Phong or CookTorr.

IOR 'Index of Refraction'. This parameter is not actually used to compute refraction of light rays through the material (a ray tracer is needed for that), but to correctly compute specular reflection intensity and extension via Snell's Law.

Toon

Reference

Mode: All Modes
Panel: *Material* → *Shaders*

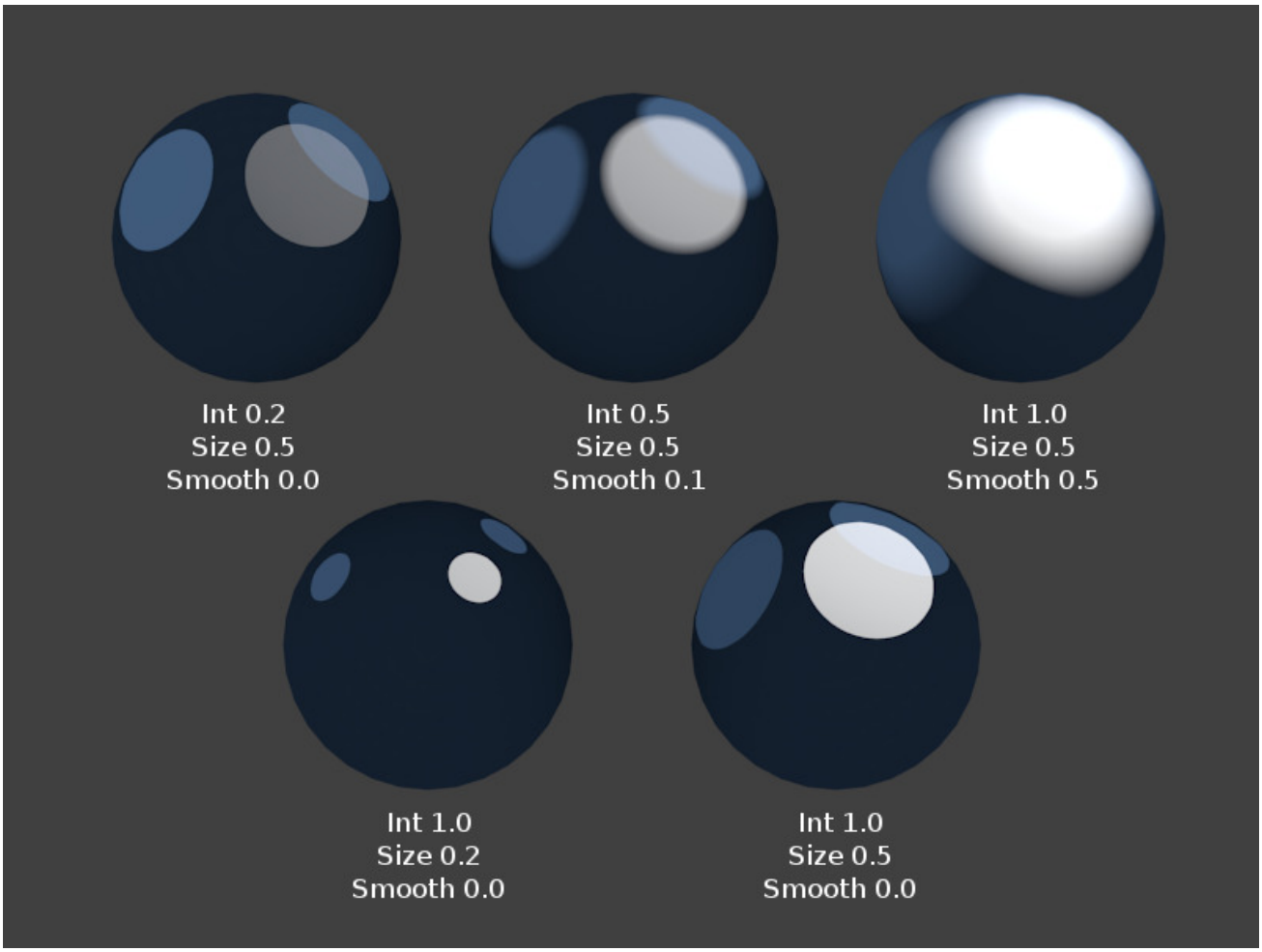


Fig. 2.1463: Toon Specular Shader (Toon Diffuse, Int 0.8, Size & Smooth match).

The Toon specular shader matches the Toon diffuse shader. It is designed to produce the sharp, uniform highlights of cartoon cels.

Options

Size Size of the specular highlight.

Smooth Softness of the highlight's edge.

Tip: Alternative Method

The Toon shader effect can also be accomplished in a more controllable way using color ramps.

Ward Isotropic

Reference

Mode: All Modes
Panel: *Material* → *Shaders*

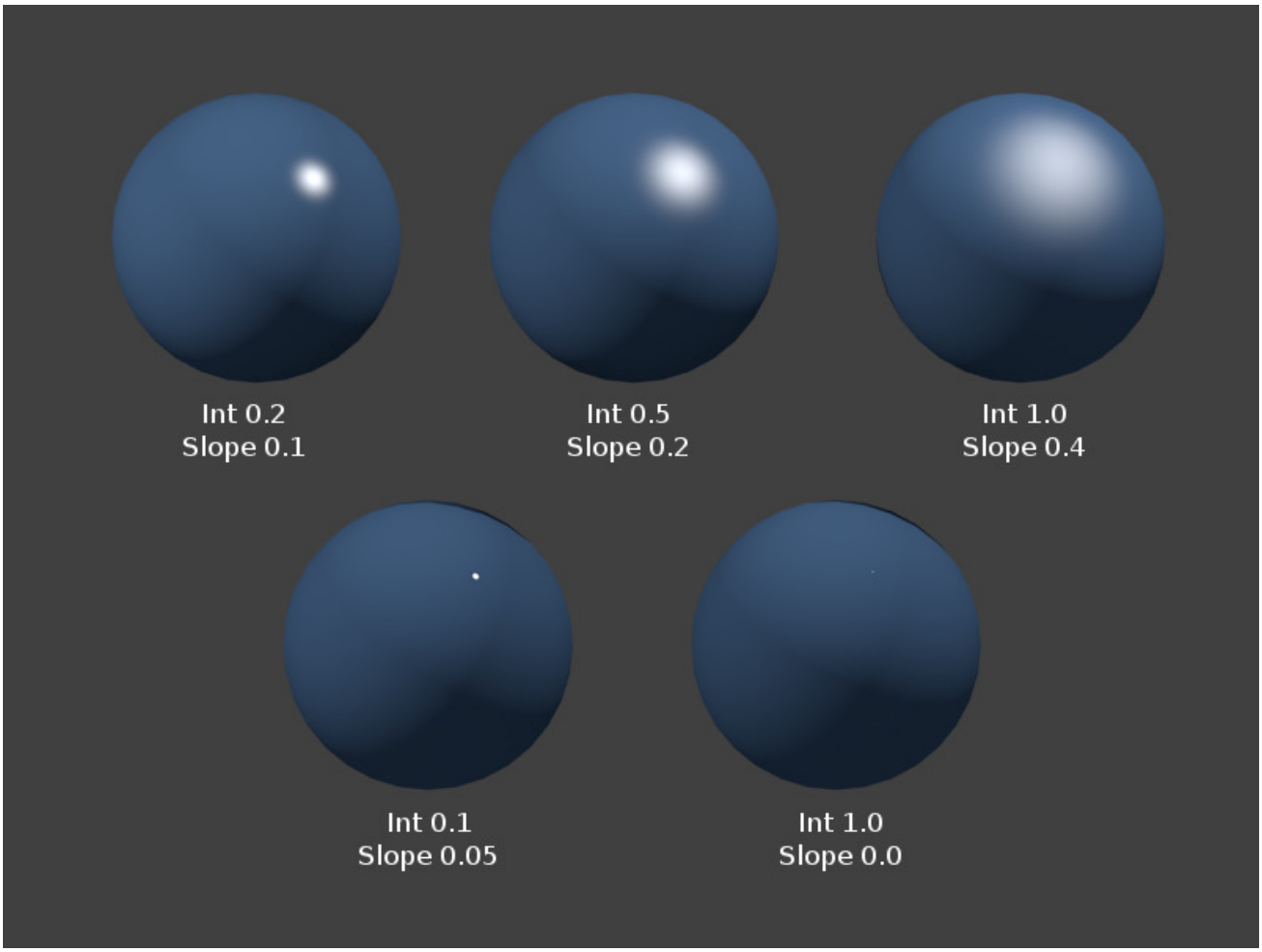


Fig. 2.1464: Ward isotropic Shader.

Ward isotropic is a flexible specular shader that can be useful for metal or plastic.

Gregory J. Ward developed a relatively simple model that obeyed the most basic laws of physics. In his 1992 paper, "Measuring and modeling anisotropic reaction", Ward introduced a Bidirectional Reflectance Distribution Function (BRDF) since then widely used in computer graphics because the few parameters it uses are simple to control. His model could represent both isotropic surfaces (independent of light direction) and anisotropic surfaces (direction dependent). In Blender, the Ward specular shader is still called "Ward Isotropic" but is actually anisotropic. (PDF)

Options

Slope Standard deviation for of surface slope. Previously known as the root-mean-square or rms value, this parameter in effect controls the size of the specular highlight, though using a different method to that of the other specular shaders. It is capable of extremely sharp highlights.

Color Ramps

Reference

Mode: All Modes
Panel: *Material* → *Ramps*

In many real life situations, like skin or metals, the color of diffuse and specular reflections can differ slightly, based on the amount of energy a surface receives or on the light angle of incidence. The *Ramp Shader* options in Blender allow you to set a range of colors for a *Material*, and define how the range will vary over a surface, and how it blends with the 'actual color' (typically from a material or as output of a texture).

Ramps allow you to precisely control the color gradient across a material, rather than just a simple blend from a brightened color to a darkened color, from the most strongly lit area to the darkest lit area. As well as several options for controlling the gradient from lit to shadowed, ramps also provide 'normal' input, to define a gradient from surfaces facing the camera to surfaces facing away from the camera. This is often used for materials like some types of metallic car paint that change color based on viewing angle.

Since texture calculations in Blender happen before shading, the *Ramp Shader* can completely replace texture or material color. But by use of the mixing options and Alpha values it is possible to create an additional layer of shading in Blender materials.

Options

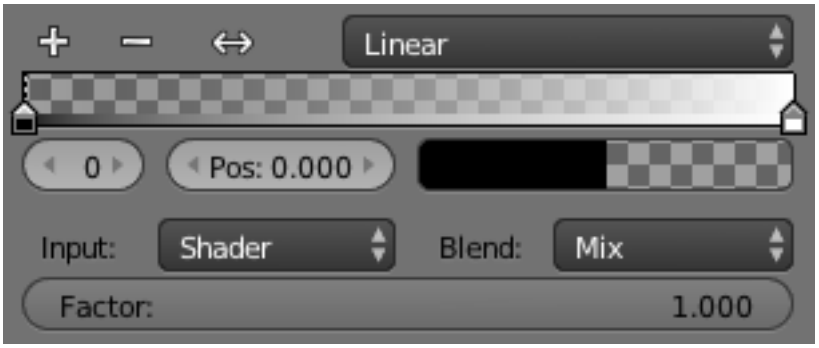


Fig. 2.1465: Ramps Panel.

For the first part of the color ramp option see *Color Ramp Widget*.

Input The input menu contains the following options for defining the gradient:

Shader The value as delivered by the material's shader (*Lambert*, *Cook Torrance*) defines the color. Here the amount of light does not matter for color, only the direction of the light.

Energy As *Shader*, now also lamp energy, color, and distance are taken into account. This makes the material change color when more light shines on it.

Normal The surface normal, relative to the camera, is used for the *Ramp Shader*. This is possible with a texture as well, but added for convenience.

Result While all three previous options work per lamp, this option only works after shading calculations. This allows full control over the entire shading, including 'Toon' style results. Using alpha values here is most useful for tweaking a finishing touch to a material.

Blend A list of the various *Color Blend Modes* are available for blending the ramp shader with the color from *Input*.

Factor This slider denotes the overall factor of the ramp shader with the color from *Input*.

Shading

In the separate *Shading* panel six more options are available:

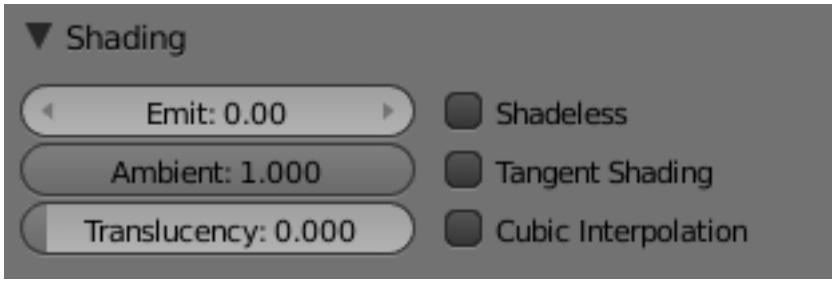


Fig. 2.1466: Shading menu, default settings.

Emit Amount of light to emit.

Ambient Amount of global ambient color the material receives. Each material has an *Ambient* slider that lets you choose how much ambient light that object receives. Set to 1.0 by default.

You should set this slider depending on the amount of ambient light you think the object will receive. Something deep in the cave will not get any ambient light, whereas something close to the entrance will get more. Note that you can animate this effect, to change it as the object comes out of the shadows and into the light.

See also:

Settings for Ambient Occlusion and Environment Lighting can be found in the World tab, with parameters affecting both these lighting components found in the *World* → *Gather* panel.

Translucency Amount of light from the back side that shows through.

Shadeless Disables the calculation of any shading. This makes material insensitive to light or shadow, resulting in a solid, uniform color for the whole object.

Tangent Shading Use the material's tangent vector instead of the normal for shading, i.e. for anisotropic shading effects (like soft hair and brushed metal). This shading was introduced in 2.42, see also settings for strand rendering in the menu further down and in the Particle System menu.

Cubic Interpolation Use cubic interpolation for diffuse values, for smoother transitions between light areas and shadowed areas. Enhances the perceived contrast.

Fig. 2.1467: Without Cubic enabled.

Fig. 2.1468: With Cubic enabled.

Transparency

Reference

Mode: All Modes
Panel: *Material* → *Transparency*

Materials in Blender can be set to be transparent, so that light can pass through any objects using the material. Transparency is controlled using an "alpha" channel, where each pixel has an additional value, range 0-1, in addition to its RGB color values. If alpha=0, then the pixel is transparent, and the RGB values for the surface contribute nothing to the pixel's appearance; for alpha=1, the surface is fully opaque, and the color of the surface determines the final color of the pixel.

Fig. 2.1469: Transparency Panel.

In Blender, there are three ways in which the transparency of a material can be set: Mask, Z-Buffer and Ray-trace. Each of these is explained in more detail below. The *Material Preview* option with a sphere object gives a good demonstration of the capabilities of these three options.

Common Options

The following property controls are available for all transparency options:

Alpha Sets the transparency of the material by setting all pixels in the alpha channel to the given value.

Fresnel Sets the power of the Fresnel effect. The Fresnel effect controls how transparent the material is, depending on the angle between the surface normal and the viewing direction. Typically, the larger the angle, the more opaque a material becomes (this generally occurs on the outline of the object).

Specular Controls the alpha/falloff for the specular color.

Blend Controls the blending between transparent and non-transparent areas. Only used if Fresnel is greater than 0.

Mask

This option simply masks the Background. It uses the alpha channel to mix the color of each pixel on the active object plane with the color of the corresponding background pixel, according to the alpha channel of the pixel. Thus for alpha = 1, the object color is seen – the object is completely opaque; but if alpha = 0, only the background is seen – the object is transparent (but note that any other object behind the active object disappears).

This is useful for making textures of solid or semi-transparent objects from photographic reference material, i.e. a mask is made with alpha opaque for pixels within the object, and transparent for pixels outside the object.

See also:

Mask Transparency.

Z Buffer

This uses the alpha buffer for transparent faces. The alpha value of each pixel determines the mix of the basic color of the material, and the color of the pixel is determined from the objects/background behind it. Only basic settings are available with this option; it does not calculate refractions.

Raytraced Transparency

Uses ray tracing to calculate refractions. Ray tracing allows for complex refractions, falloff, and blurring, and is used for simulating the refraction of light rays through a transparent material, like a lens.

Note: The Raytrace option is only available in the Blender Render and Cycles render engines, but not in the Game Engine.

A ray is sent from the camera and travels through the scene until it encounters an object. If the first object hit by the ray is non-transparent, then the ray takes the color of the object.

If the object is transparent, then the ray continues its path through it to the next object, and so on, until a non-transparent object is finally encountered which gives the whole chain of rays its color. Eventually, the first transparent object inherits the colors of its background, proportional to its *Alpha* value (and the Alpha value of each transparent Material hit in between).

But while the ray travels through the transparent object, it can be deflected from its course according to the Index of Refraction (IOR) of the material. When you actually look through a plain sphere of glass, you will notice that the background is upside-down and distorted: this is all because of the Index of Refraction of glass.

Note: Enable Raytracing

To get ray-traced transparency, you need to:

- Enable ray tracing in your Render settings. This is done in the *Render* → *Shading* panel. Ray tracing is enabled by default.
- Set your Alpha value to something other than 1.0.
- In order for the background material to receive light passing through your transparent object, *Receive Transparent* must be turned on for that material in the *Material* → *Shadow* panel.

Options

In addition to the common options given above, the following property controls are available:

IOR Index of Refraction. Sets how much a ray traveling through the material will be refracted, hence producing a distorted image of its background. See *IOR values for Common Materials* below.

Filter Amount of filtering for transparent ray trace. The higher this value, the more the base color of the material will show. The material will still be transparent but it will start to take on the color of the material. Disabled (0.0) by default.

Falloff How fast light is absorbed as it passes through the material. Gives 'depth' and 'thickness' to glass.

Limit Materials thicker than this are not transparent. This is used to control the threshold after which the filter color starts to come into play.

Depth Sets the maximum number of transparent surfaces a single ray can travel through. There is no typical value. Transparent objects outside the *Depth* range will be rendered pitch black if viewed through the transparent object that the *Depth* is set for. In other words, if you notice black areas on the surface of a transparent object, the solution is probably to increase its *Depth* value (this is a common issue with ray tracing transparent objects). You may also need to turn on transparent shadows on the background object.

Gloss Settings for the glossiness of the material.

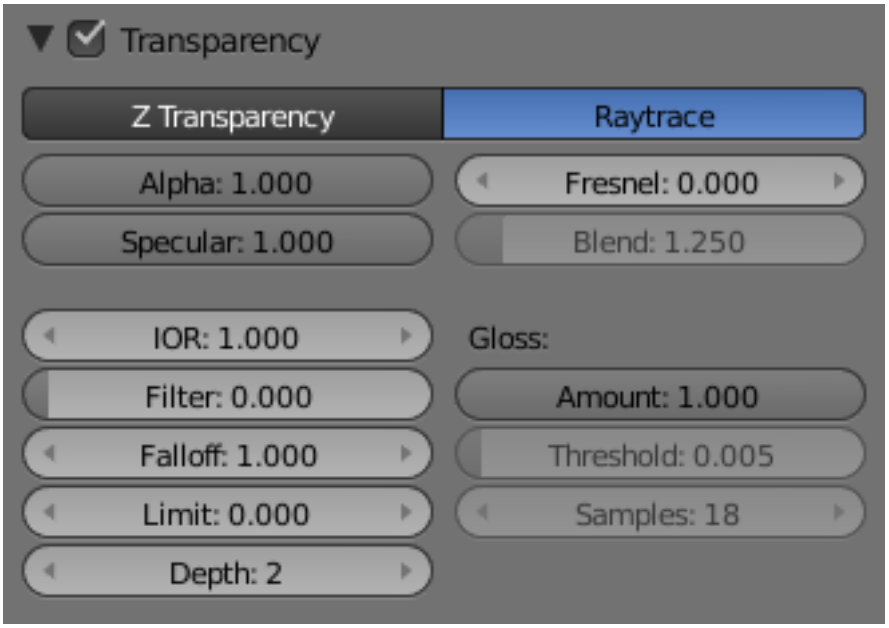


Fig. 2.1470: The Transparency Panel.

Amount The clarity of the refraction. Set this to something lower than zero to get a blurry refraction.

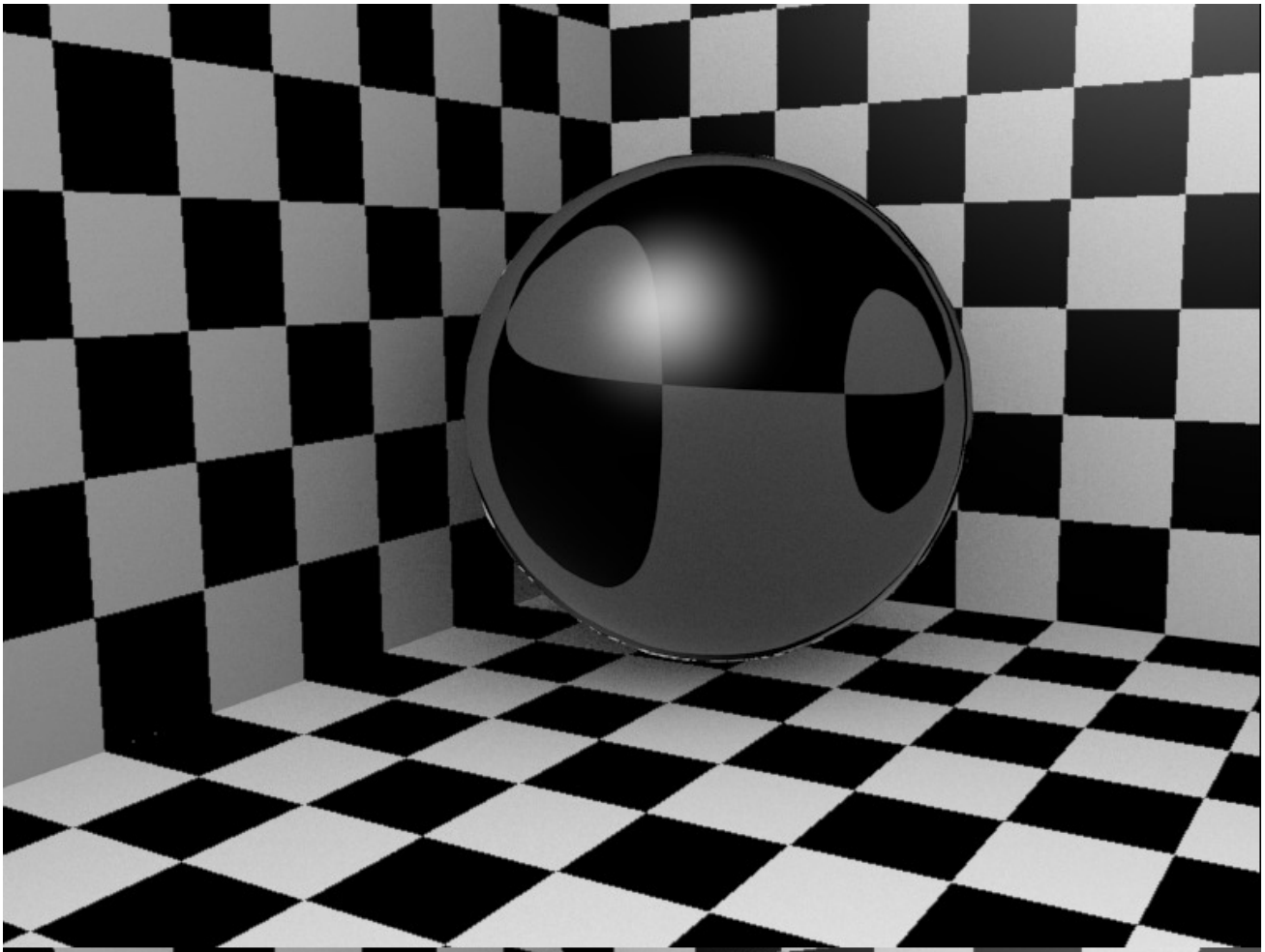
Threshold Threshold for adaptive sampling. If a sample contributes less than this amount (as a percentage), sampling is stopped.

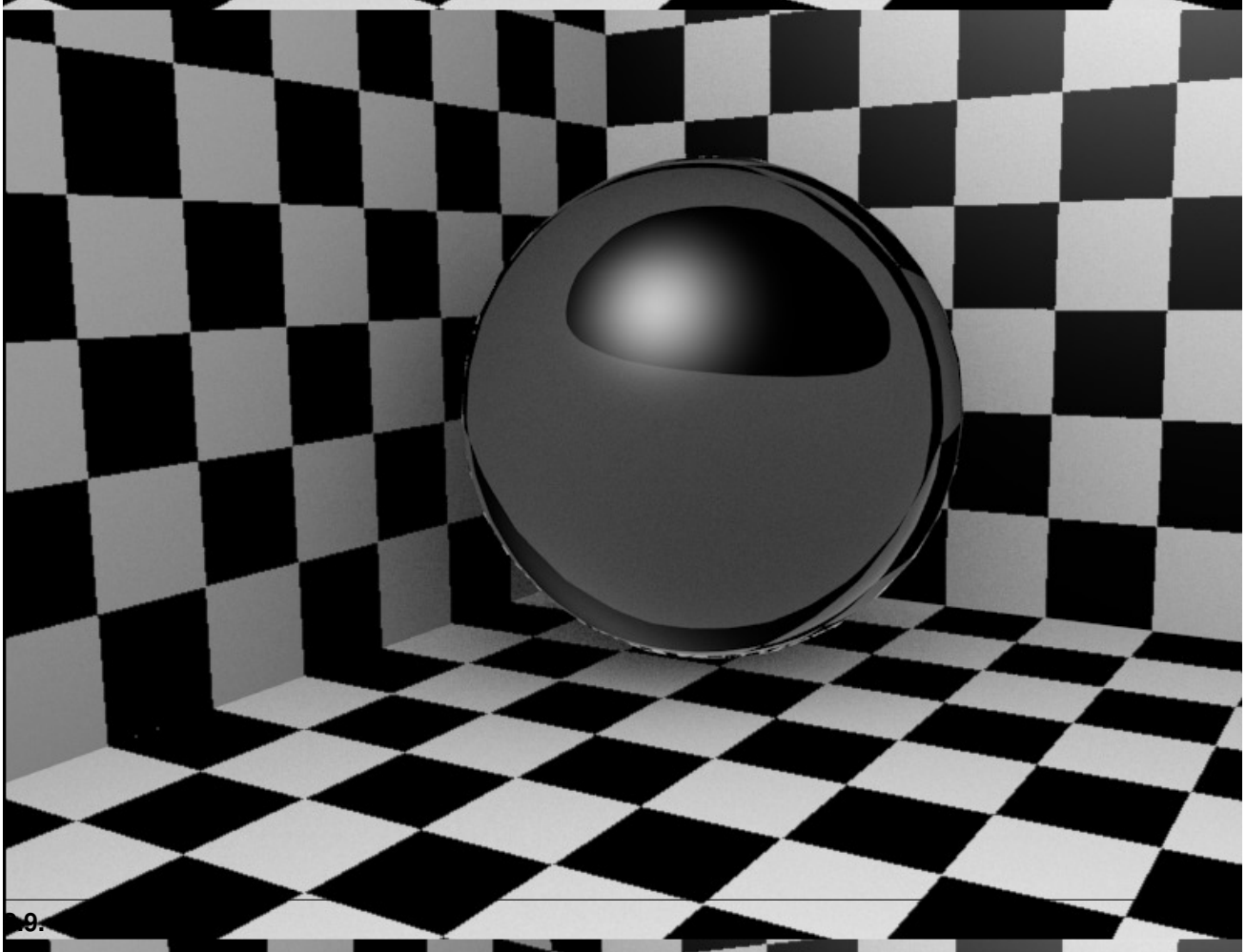
Samples Number of cone samples averaged for blurry refraction.

Examples

Index of Refraction

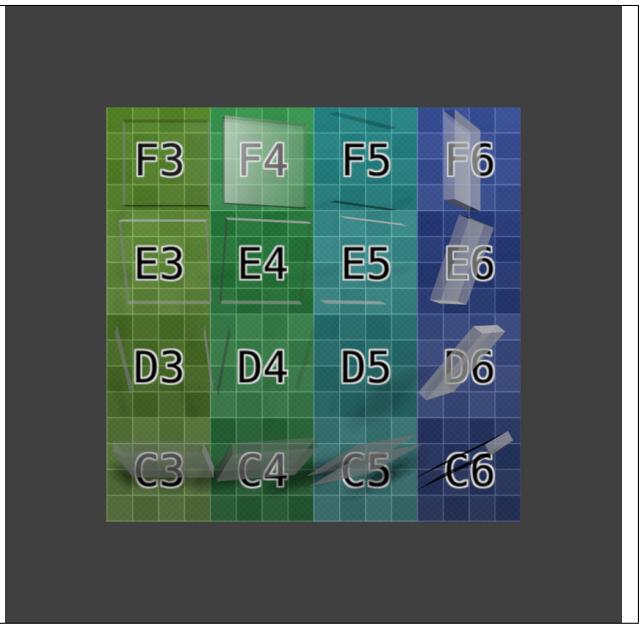
(Influence of the IOR of an Object on the distortion of the background: spheres of Water, Glass and Diamond (top to bottom)). There are different values for typical materials: Air: 1.000 (no refraction), Alcohol: 1.329, Glass: 1.517, Plastic: 1.460, Water: 1.333 and Diamond: 2.417.





Fresnel

Table 2.79: With alpha buffered transparency.



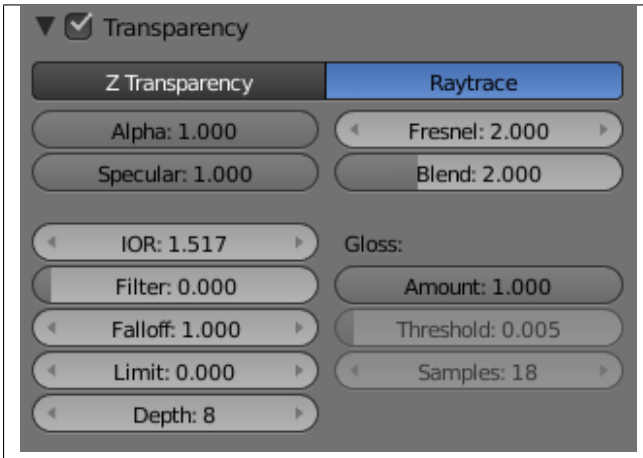


Fig. 2.1474: Settings for Fresnel using ray-traced.

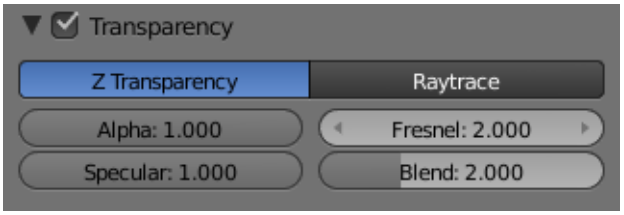


Fig. 2.1475: Settings for Fresnel using Z transparency.

Note: The specular highlight in the F4 glass tile (which is facing midway between the light and the camera); the Fresnel effect can be seen in row C and column 6 where the faces are turned away from the camera.

The amount of Fresnel effect can be controlled by either increasing the *Blend* value or decreasing the *Alpha* value.

Depth

Increasing *Depth* also considerably increases render time. Each time a light ray passes through a surface, the ray-tracing algorithm is called recursively. In the example above, each side of each glass has an exterior and an interior surface. Light rays thus have to pass through four surfaces for each glass.

But not only that, at every point on a surface, some of the light can be reflected, or mirrored off the surface in various directions. This results in multiple rays needing to be calculated for each point (often referred to as a tree of rays). In each of the rendered images above there are 640×400=256 000 pixels. By increasing *Depth*, at least one tree of rays is added to each pixel.

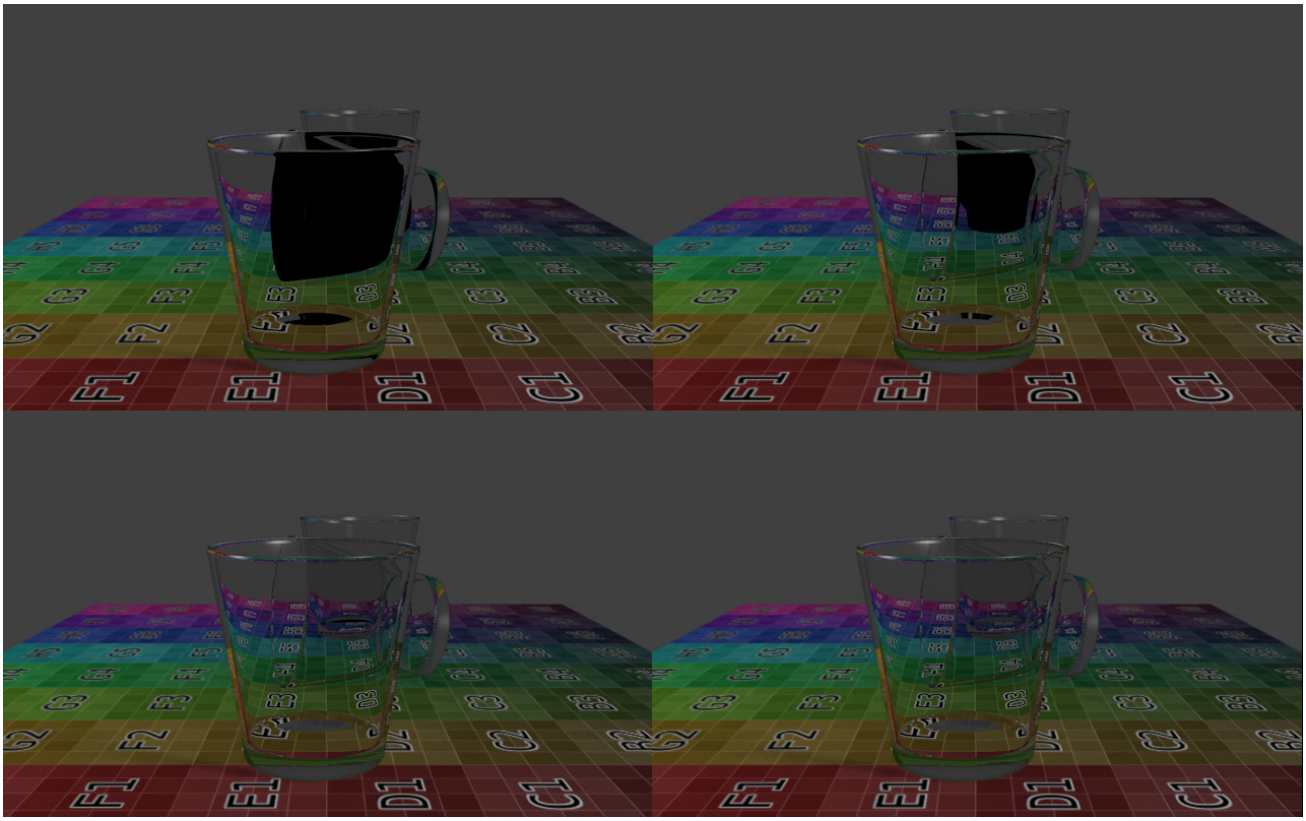


Fig. 2.1476: A simple scene with three glasses on a surface, and three lamps. Depth was set to 4, 8, 12, and 14, resulting in render times of 24 sec, 34 sec, 6 min, and 11 min respectively. (Download blend-file.)

Be kind to your computer. Carefully placing objects in a scene to avoid overlapping transparent objects is often an interesting alternative.

Hints

Transparent shadows

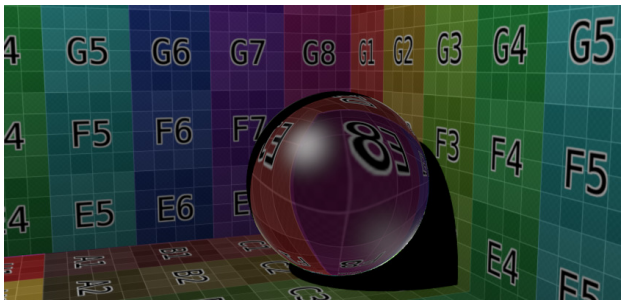


Fig. 2.1477: No transparent shadows.

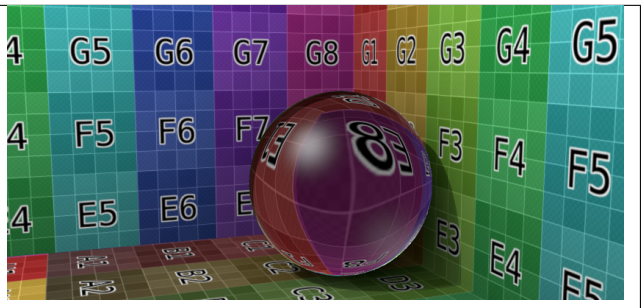


Fig. 2.1478: No transparent shadows, environment lighting enabled.

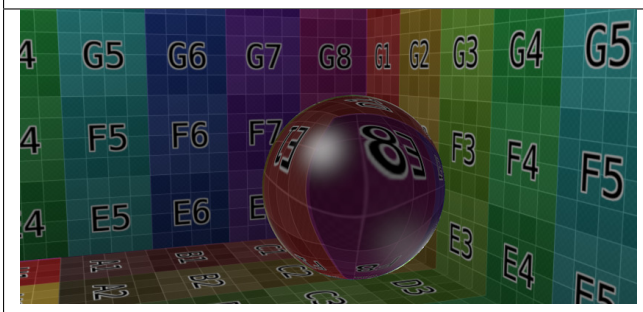


Fig. 2.1479: Transparent shadows enabled, alpha set to 0.0.

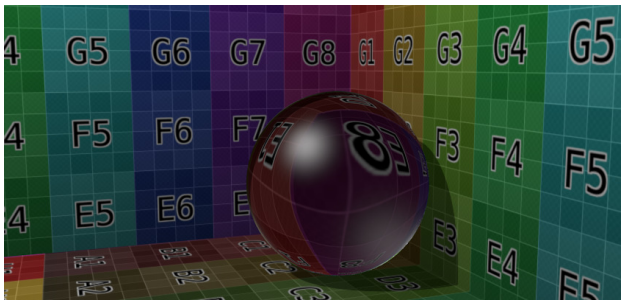


Fig. 2.1480: As previous, alpha set to 0.25.

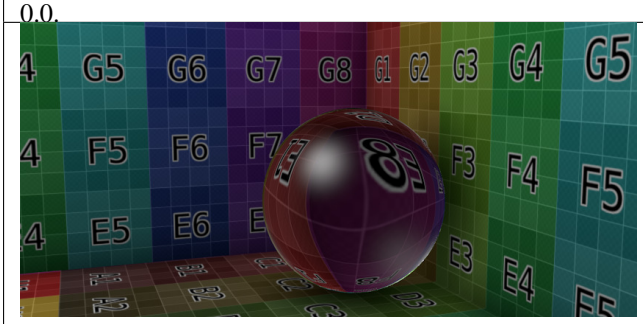


Fig. 2.1481: Transparent shadows with ambient occlusion set to multiply, distance 1 (radius of sphere).

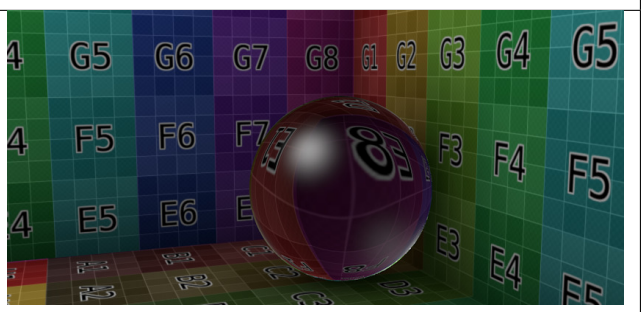


Fig. 2.1482: As previous, distance increased to 2 (diameter of sphere).

By default, the shadows of transparent objects are rendered solid black, as if the object was not transparent at all. But in reality, the more transparent an object is, the lighter its shadow will be.

In Blender, transparent shadows are set on the materials that receive the shadows from the transparent object. This is enabled and disabled with the *Receive Transparent* button, in the *Material* → *Shadow* panel. The shadow's brightness is dependent on the *Alpha* value of the shadow casting material.

Alternatives to transparent ray-traced shadows can be found in the *World* tab, namely the *Ambient Occlusion*, *Environment Lighting*, and *Gather* panels. Alternatively, a texture can be used to control the *Intensity* value of the shadow-receiving material.

IOR values for Common Materials

The following list provides some index of refraction values to use when ray-traced transparency is used for various liquids, solids (gems), and gases:

Gasses

- Air 1.000
- Carbon Dioxide 1.000449
- Oxygen 1.000276

Common Liquids

- Alcohol 1.329
- Milk 1.35
- Oil, vegetable (50- C) 1.47
- Shampoo 1.362
- Water (0- C) 1.33346
- Water (100- C) 1.31766
- Water (20- C) 1.33283
- Water (gas) 1.000261
- Water (35- C, room temp) 1.33157
- Vodka 1.363

Common Transparent Materials

- Glass 1.51714
- Ice 1.309
- Rock Salt 1.544

Common Opaque Materials

- Asphalt 1.635
- Chalk 1.510
- Plastic 1.46
- Rubber, Natural 1.5191
- Silicon 4.24

Gemstones

- Diamond 2.417
- Jade, Nephrite 1.61
- Opal 1.45
- Ruby 1.757 -1.779

Metals

- Aluminum 1.44
- Bronze 1.18
- Copper 1.10
- Gold 0.47
- Iron 1.51
- Lead 2.01
- Platinum 2.33
- Silver 0.18
- Steel 2.50
- Titanium 2.16

Mirror Reflections

Mirror reflections are computed in the Blender Render and Cycles render engines using ray tracing. (NB: Reflections are not available in the Game Engine.) Ray tracing can be used to make a material reflect its surroundings, like a mirror. The principle of ray-traced reflections is very simple: a ray is fired from the camera and travels through the scene until it encounters an object. If the first object hit by the ray is not reflective, then the ray takes the color of the object. If the object is reflective, then the ray bounces from its current location and travels up to another object, and so on, until a non-reflective object is finally met and gives the whole chain of rays its color.

Eventually, the first reflective object inherits the colors of its environment, proportional to its *Reflectivity* value. Obviously, if there are only reflective objects in the scene, then the render could last forever. This is why a mechanism for limiting the travel of a single ray is set through the *Depth* value: this parameter sets the maximum number of bounces allowed for a single ray.

Note: You need to enable ray tracing in your scene settings if you want to use ray-traced reflections. This is done in the *Render* → *Shading* panel.

The *Mirror Color* in the mirror panel is the color of the light reflected back. Usually, for normal mirrors, use white. However, some mirrors color the reflection (e.g. metals), so you can change the color by clicking on the color button. The amount of mirrored reflection is determined by the *Reflectivity* value. If set to something greater than 0, mirrored reflectivity will be activated and the reflection will be tinted the color set in Mirror Color.

Options

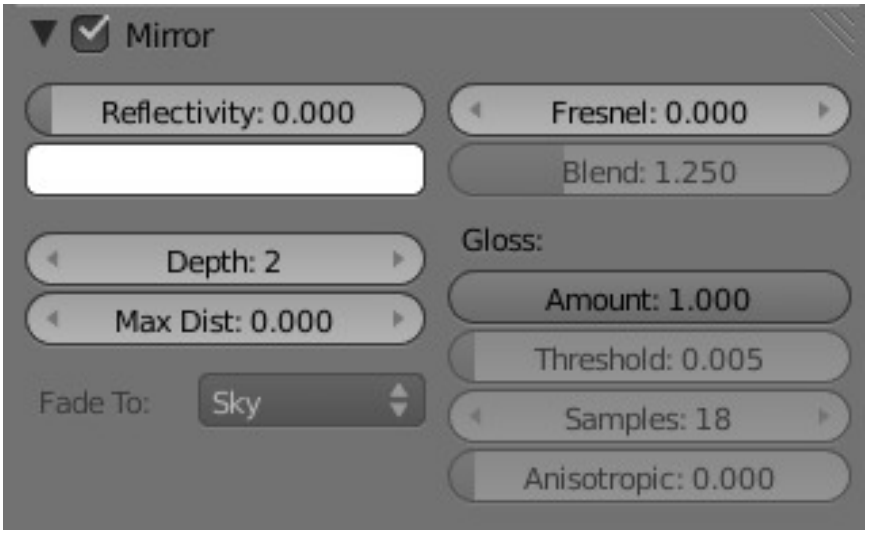


Fig. 2.1483: The Mirror Panel.

Enable ray-traced reflections Enable or disable ray-traced reflections

Reflectivity Sets the amount of reflectiveness of the object. Use a value of 1.0 if you need a perfect mirror, or set it to 0.0 if you do not want any reflection.

Mirror Color Color of mirrored reflection By default, an almost perfectly reflective material like chrome, or a mirror object, will reflect the exact colors of its surrounding. But some other equally reflective materials tint the reflections with their own color. This is the case for well-polished copper and gold, for example. In order to replicate this within Blender, you have to set the Mirror Color accordingly. To set a mirror color, simply click the color button in the mirror panel and select a color.

Fresnel Sets the power of the Fresnel effect. The Fresnel effect controls how reflective the material is, depending on the angle between the surface normal and the viewing direction. Typically, the larger the angle, the more reflective a material becomes (this generally occurs on the outline of objects).

Fig. 2.1484: Picking a mirror color.

Blend A controlling factor to adjust how the blending happens between the reflective and non-reflective areas.

Depth Maximum allowed number of light inter-reflections. If your scene contains many reflective objects and/or if the camera zooms in on such a reflective object, you will need to increase this value if you want to see surrounding reflections in the reflection of the reflected object (!). In this case, a Depth of 4 or 5 is typically a good value.

Max Distance Maximum distance of reflected rays away from camera (Z-Depth) in Blender units. Reflections further than this range fade out to reduce compute time.

Fade to The color that rays with no intersection within the *Max Distance* take. *Material* color can be best for indoor scenes, *Sky* color (World settings) for outdoor scenes.

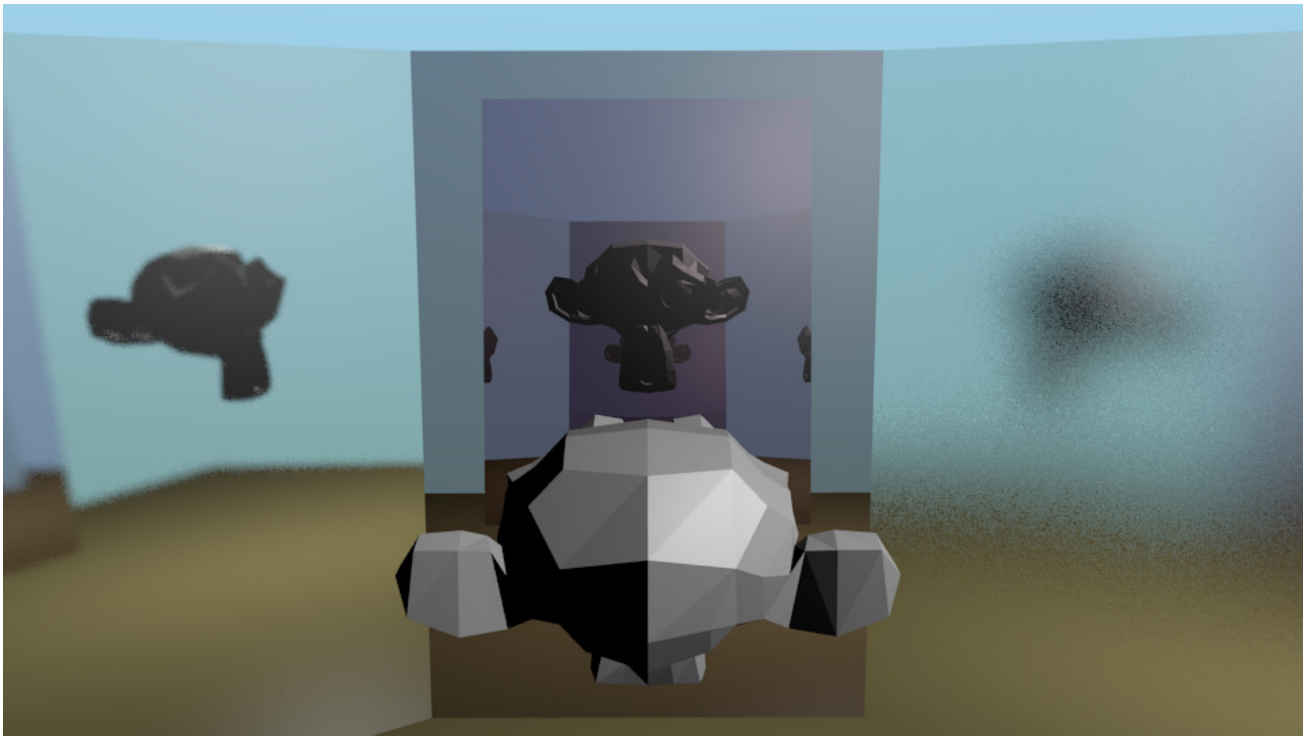


Fig. 2.1485: Suzanne in the Fun House (blend-file)

Gloss In paint, a high-gloss finish is very smooth and shiny. A flat, or low gloss, finish disperses the light and gives a very blurry reflection. Also, uneven or waxed-but-grainy surfaces (such as car paint) are not perfect and therefore slightly need a Gloss greater than 1.0. In the example to the right, the left mirror has a Gloss of 0.98, the middle is Gloss = 1.0, and the right one has Gloss of 0.90. Use this setting to make a realistic reflection, all the way up to a completely foggy mirror. You can also use this value to mimic depth of field in mirrors.

Amount The shininess of the reflection. Values < 1.0 give diffuse, blurry reflections and activate the settings below.

Threshold Threshold for adaptive sampling. If a sampling contributes less than this amount (as percentage), sampling is stopped. Raising the threshold will make the adaptive sampler skip more often, however, the reflections could become noisier.

Samples Number of cone samples averaged for blurry reflection. More samples will give a smoother result, but will also increase render time.

Examples

Fresnel

Let us undertake a small experiment in order to understand what Fresnel is really about. After a rainy day, go out and stand over a puddle of water. You can see the ground through the puddle. If you kneel just in front of the puddle, your face close to the ground, and look again at a distant point on the puddle of water, the liquid surface part which is closer to you lets

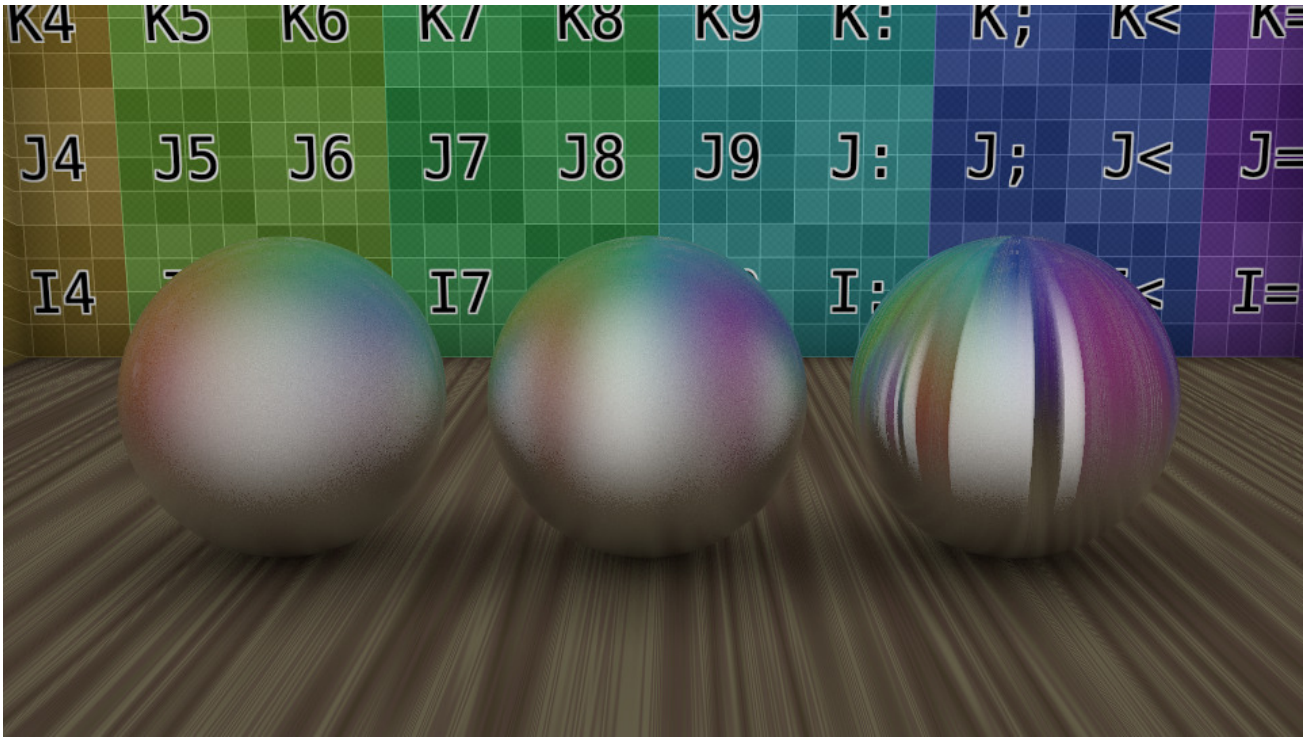
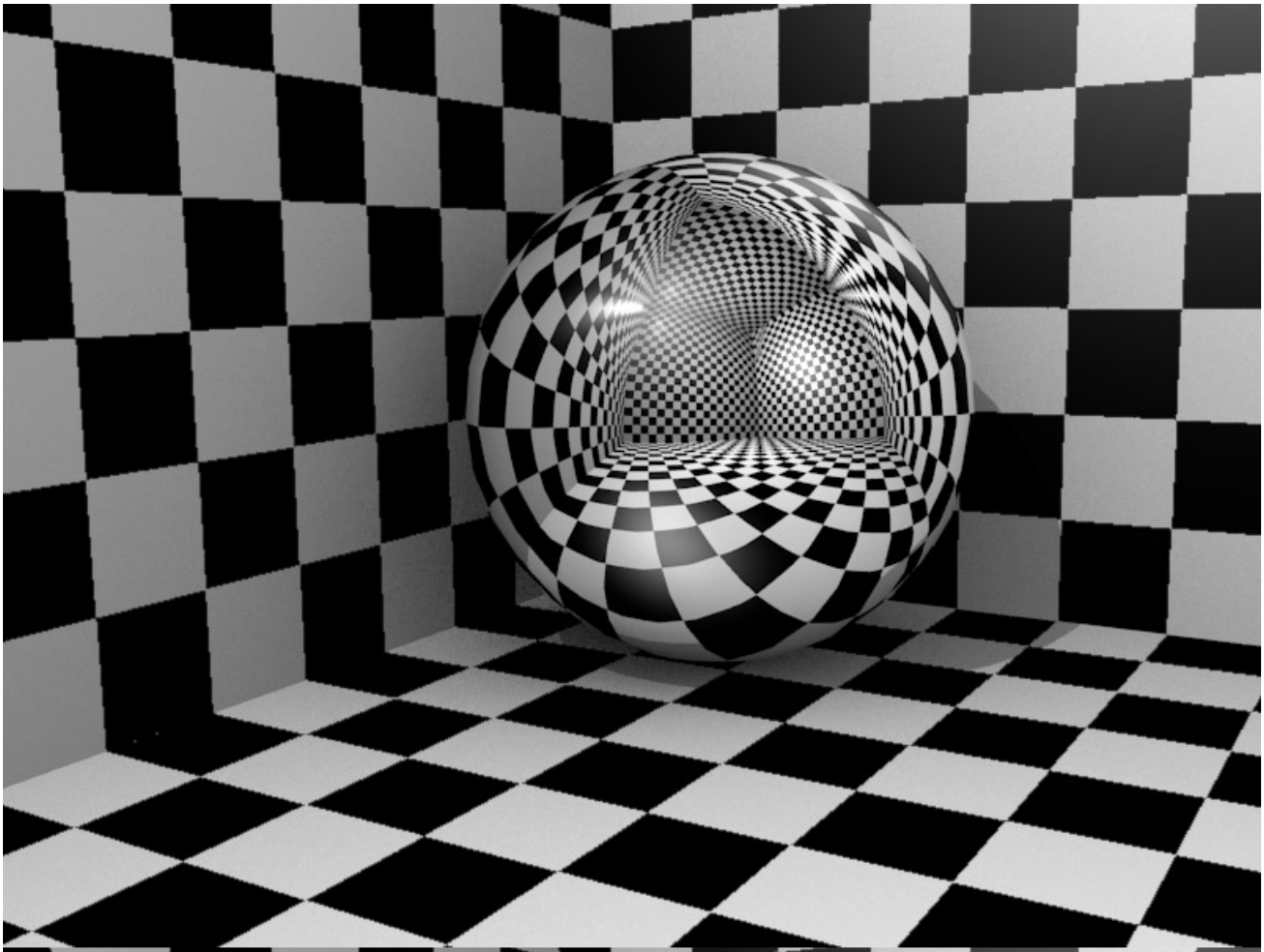
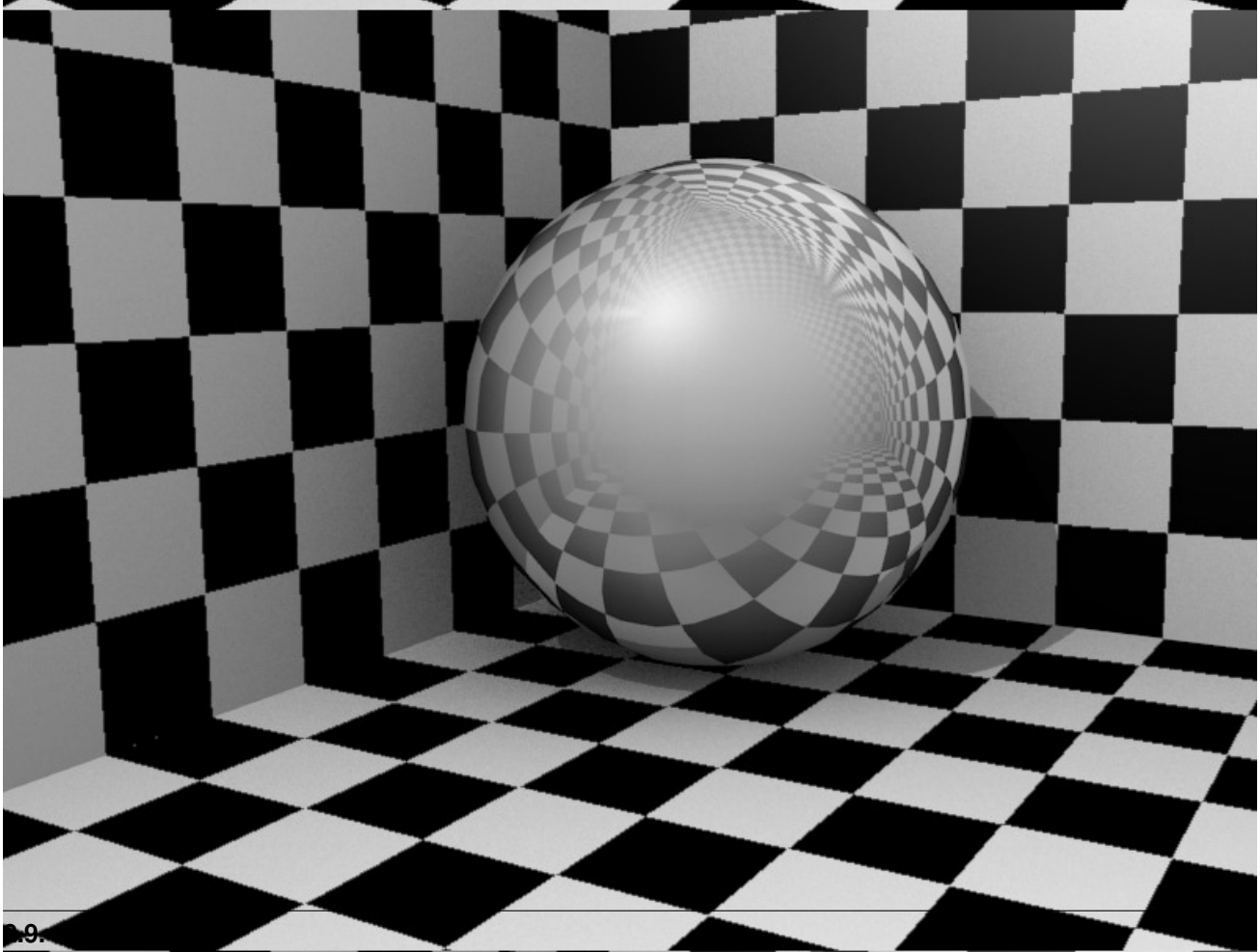


Fig. 2.1486: Anisotropic tangent reflecting spheres with anisotropic set to 0.0, 0.75, 1.0. (.blend)

Anisotropic The shape of the reflection, from 0.0 (circular) to 1.0 (fully stretched along the tangent). If the *Tangent Shading* is on, Blender automatically renders blurry reflections as anisotropic reflections. When Tangent is switched on, the *Anisotropic* slider controls the strength of this anisotropic reflection, with a range of 1.0 (default) being fully anisotropic and 0.0 being fully circular, as is when tangent shading on the material is switched off. Anisotropic ray-traced reflection uses the same tangent vectors as for tangent shading, so you can modify the angle and layout the same way, with the auto-generated tangents, or based on the mesh's UV co-ordinates.





you see the ground, but if you move your gaze towards the other end of the puddle, then the ground is gradually masked until all you see is the reflection of the sky. This is the Fresnel effect: having a surface sharing reflective and non-reflective properties according to the viewing angle and the surface normal.

In *Demonstration of Fresnel effect with values equal to (from top to bottom) 0.0, 2.5 and 5.0*, this behavior is demonstrated for a perfectly reflective Material (Mirror Reflectivity 1.0).

Fresnel 0.0 stands for a perfect mirror Material, while Fresnel 5. 0 could stand for a glossy Material. It is barely noticeable but in the lower picture, the Material is perfectly reflective around the edges.

The smoothness of the Fresnel limit can be further controlled using the *Blend* slider.

Subsurface Scattering

Many organic and inorganic materials are not totally opaque right at the surface, so light does not just bounce off the top surface. Instead, some light also penetrates the skin surface deeply, and scatters around inside, taking on the color of the insides and emerging back out at a different location. Human/animal skin, the skin of grapes, tomatoes, fruits, wax, gels (like honey, or Jello) and so on all have subsurface scattering (SSS), and photo-realism really cannot be achieved without it.

It is important to understand that subsurface scattering and diffuse are one and the same. The difference is in how far light can diffuse beneath the surface before it is absorbed or transmitted back out.

How it works

Actually calculating the light path beneath the surface of an object is not practical. But it has been shown that it is not necessary to do this, and that one can use a different approach.

Blender calculates SSS in two steps:

- At first the irradiance, or brightness, of the surface is calculated, from the front side of the object as well as from its back side. This is pretty much the same as in a normal render. Ambient Occlusion, Radiosity, the type of diffuse Shader, the light color, etc. are taken into account.
- In the second step, the final image is rendered, but now the SSS shader replaces the diffuse shader. Instead of the lamps, the calculated lightmap is used. The brightness of a surface point is the calculated "Average" of the brightness of its surrounding points. Depending on your settings the whole surface may be taken into account, and it is a bit more complicated than simply calculating the average, but do not bother too much with the math behind it.

Instead let us see what SSS does to a distinct light point.

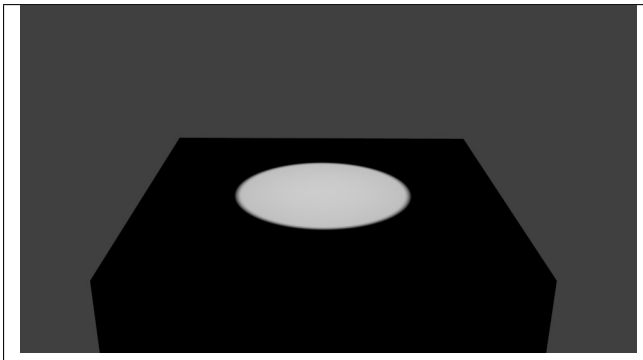


Fig. 2.1488: No SSS.

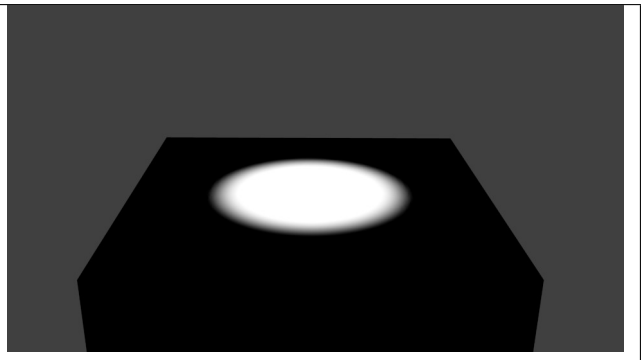


Fig. 2.1489: Small SSS radius.

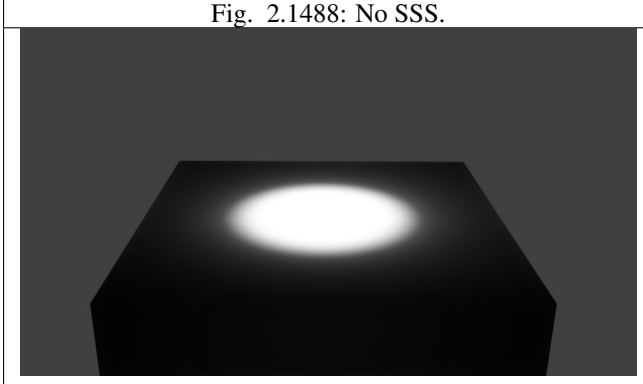


Fig. 2.1490: SSS radius enlarged.

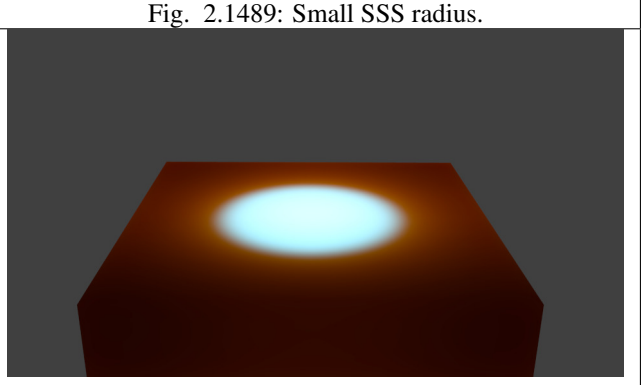


Fig. 2.1491: SSS with very large red radius value.

If you turn on SSS, the light is distributed over a larger area. The size of this area depends on the radius values. Instead of distributing all colors with the same amount, you may choose different radius values for each of the RGB colors.

If you use a very large radius value for a color, its light is evenly distributed over the whole object.

Note: Note about scatter radius

Because of the way this scattering is calculated, when using large radius values, you will notice fringing artifacts that appear as the complementary color to the predominant color of the scattering. Above, you see in the last image a bluish band in the illuminated area. This is an unfortunate limitation. A way to lessen this effect is use multiple passes with different scatter radii, and average them.

Enabling Subsurface Scattering

- Enable SSS by clicking on the *Subsurface Scattering* button.
- Accessible at the top are various presets. When you select a preset, the *Radius* values, the *RGB Radius* and the *IOR* are set for you. The remaining options are not set (because they are mostly dependent on the size of your object).

Subsurface Scattering does not need ray tracing. But since it is dependent on the incident light and shadows, you need proper shadow calculation (which may need ray tracing).

Options

The numeric sliders control how the light is scattered:

IOR The *Index Of Refraction* value determines the falloff of incident light. Higher values means that light falls off faster. The effect is quite subtle and changes the distribution function only a little bit. By the examination of many different materials, values of (1.3 to 1.5) have been found to work well for most materials. If you know the exact material you are trying to simulate, see *IOR values for Common Materials*.

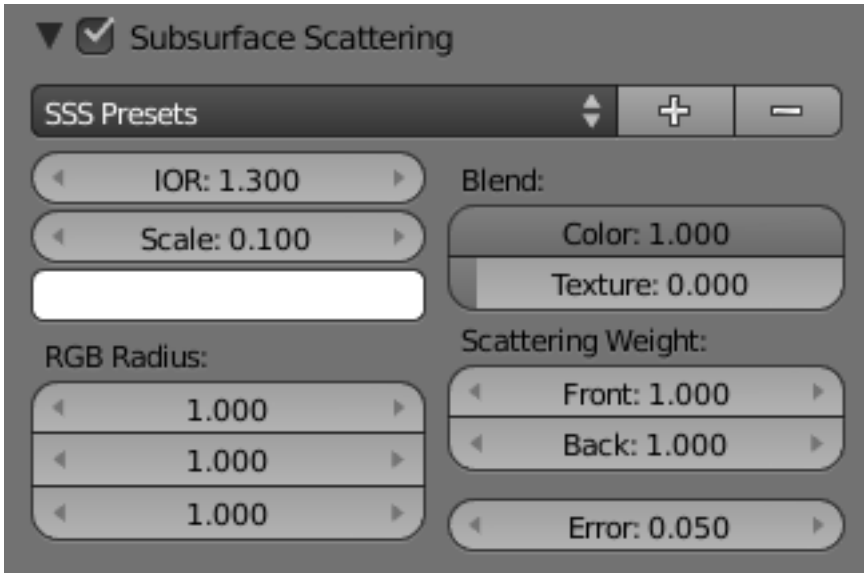


Fig. 2.1492: The SSS Panel. SSS is already enabled.

Scale The scale of your object, in Blender units, across which you want the scattering effect to take place. Scale of 1.0 means 1 Blender unit equals 1 millimeter, scale of 0.001 means 1 Blender unit equals 1 meter. If you want to work out what scale value to use in your scene, just use the formula: (size in Blender units)/(real world size in millimeters)=scale.

Scattering Color (Albedo) Albedo is the probability that light will survive a scattering event. If you think of scattering as a filter, this is the height of the filter. It is multiplied by the surface color. In practice, this is unintuitive. It should be the same as the surface color, however, changing this value has unintuitive results on the scattering effect:

The darker the color the more light is scattered. A value of 1 will produce no scattering effect.

So if you set it to green, the lit areas of the object will appear as green, and green is scattered only a little. Therefore the darker areas will appear in red and blue. You can compensate the different scattering by setting a larger radius for the color.

RGB Radius This is not in fact the radius of the subsurface scattering, but the average path length between scattering events. As the light travels through the object it bounces around then emerges from the surface at some other point. This value corresponds to the average length the light travels between each bounce. The longer the path length is, the further the light is allowed to scatter. This is the main source of a material's perceived "scatter color." A material like skin will have a higher red radius than green and blue. Subsurface scattering is the diffusion of light beneath the surface. You control how far the light spreads to achieve a specific result.

Blend

Color This controls how much the RGB option modulates the diffuse color and textures. Note that even with this option set to 0.0, the RGB option still influences the scattering behavior.

Texture How much the surface texture is blurred along with the shading.

Scattering Weight

Front Factor to increase or decrease the front scattering. When light enters through the front of the object, how much is absorbed or added? (Normally 1.0 or 100%).

Back Factor to increase or decrease the back scattering. Light hitting an object from behind can go all the way through the object and come out on the front of the object. This happens mostly on thin objects, like hands and ears.

Error This parameter controls how precisely the algorithm samples the surrounding points. Leaving it at 0.05 should give images without artifacts. It can be set higher to speed up rendering, potentially with errors.

Setting it at 1.0 is a good way to quickly get a preview of the look, with errors.

Developing your own SSS material

The Traditional Approach

A more common but less intuitive approach is to use "layering". This is a simplified version of the layering approach. See the external links for more information:

- Set the SSS color on a value of your choice, normally the predominant color of the object. If you want to use different radii for the colors, do not make it too dark.
- Set the scale factor. If you want to see much translucency you need small objects or large scale values.
- Set the radius values.
- Adjust the brightness with the *Front* and *Back* values.

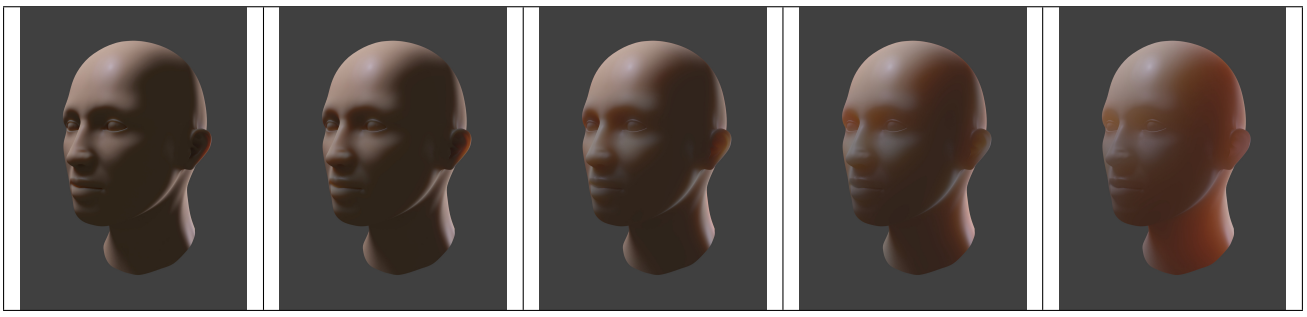
A more intuitive approach

- Set the Scattering color to 0.5
- Set the Front weight to 2.0
- Set the scale factor based on the size of your object relative to the scene. If you want to see much translucency you need small objects or large scale values.
- Set the radius values appropriately.

Examples

Skin

Table 2.80: Scale: 5.



See also:

- Development Release Log: Subsurface Scattering.
- Ben Simonds: Three Layer SSS in Blender Demystified.

Strands

The Strand section of the Material tab is specific to the rendering of Hair particles. There are two different strand methods available:

Polygon strands This is the default (old) method. The strands are rendered as flat polygons. The number of polygons depend on the *Steps* settings in the *Particles system* tab.

Strand Primitive You activate Strand Primitive with the button *Strand render* in the *Render* panel of the particle system. The hair curves are not stored as polygons; only the key points are stored, which are then converted to polygons on the fly. A second difference is the way transparency works. Rather than rendering using the existing system, all strand segments in a part are sorted front to back and rendered in that order.

Strand Primitives

- Are more memory efficient and faster, to make rendering of large amounts of fur and grass possible. For good performance, the render steps button should be lowered (e.g. 2 should be good enough fur), since the result will be a smoothed curve anyway. You need 1 to 2 render steps less than steps in the 3D View. Also, using more render parts helps to reduce memory usage.
- Have a distance of vision reduction (in the *Render* panel under *Child Simplification*) for children from faces.
- May be faded out towards the tip without an additional texture.
- Are not ray traced. So they are not visible through ray-transparent materials or in a ray mirror (you can use *Environment Mapping* for that).
- Have shape problems if they are rendered with a greater width.
- Cannot carry a UV-Texture along the strand.

Polygon strands

- Work well with greater width, so you can use them as an alternative to billboards because the strands may have an animated shape.
- Can be textured with a UV-Texture along the strands.
- Are seen by ray tracing.

Strands Shading

Strands are rendered with the material of the underlying face/vertex, including shading with a UV-Texture. Since you can assign more than one material to each face, each particle system may have its own material and the material of the underlying face can be different from the material of the strands.

Additionally strands can be shaded along the strand (from root to tip) with a mono-dimensional texture; only polygon strands can carry a two-dimensional UV-Texture.

The options for strand shading are in the *Strands* section of the *Material* tab.

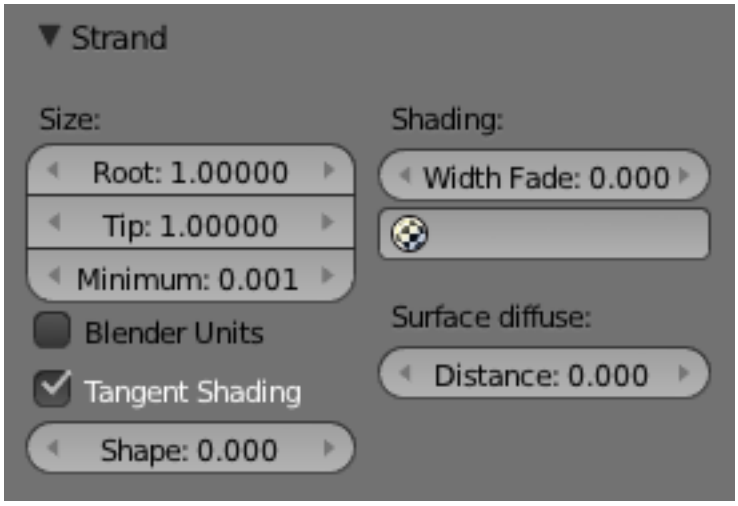


Fig. 2.1498: Strands Panel.

Root Width of the hair at the root.

Tip Width of the hair at the tip.

Minimum This is the minimum thickness (in pixels) of the strands. Strands below that size are not rendered smaller, but are faded to alpha (well, the fading works only for strand primitives). This gives a much better rendering result for thin hair.

Blender Units Normally strands are quite thin; the thickness is given in screenpixels. If you use Blender units (BU) you may set the root value up to 2 BU, and the tip value up to 1 BU. You have to consider the overall object size, because the smallest possible size is 0.001 BU. So if you use 1 BU for 1 meter the smallest possible size would be 1 mm (too thick for thin hair).

Use Tangent Shading Calculates the light as if the strands were very thin and round. This makes the hair appear brighter and shinier. Disabling the "Tangent Shading" option will still render nicely, but resembles more solid strands, as though made of metal or wood.

Shape This slider allows you to control the interpolation. Default (0.0) is a linear interpolation between *Root* and *Tip*. A negative value will make the strand narrower (spiky), a positive value will make it fatter.

Fig. 2.1499: Various Shape settings. From left to right, 0 (root and tip are equal in the first), 0, -0.4, -0.9, 0.4, 0.9.

Width Fade To fade out along the width of the strand. This works only for Strand Primitives. 0.0 is no fading at all, 1.0 linear fading out.

UV Layer You can texture polygon strands with a UV-Texture. Fill in the name of the UV-Set (not the texture) here. You also have to load the texture in the *Texture* tab and *Material* tab (*Mapping*: *UV*; you may use every *Influence* setting you like – especially the alpha value; see Fig. *Various Shape settings. From left to right, 0 (root and tip are equal in the first), 0, -0.4, -0.9, 0.4, 0.9.*).

Surface Diffuse Computes the strand normal, taking the normal at the surface into account. This eases the coloring and lighting of hair a lot, especially for Strand Primitives. Essentially hair reacts similar to ordinary surfaces and do not show exaggerated strong and large specular highlights.

Distance The distance in Blender units over which to blend in the normal at the surface (if you want to use *Surface Diffuse* only for Grass/Fur at greater distances).

Texturing along the Strand

Strands can be textured along the strand, i.e. from root to tip. To do that you have to select *Strand/Particle* in the *Coordinates* select menu in the *Mapping* panel of the *Material* tab.

Pretty much the most important setting is shown in Fig. *Fading a strand to alpha.*, how to fade the tip of a strand to alpha to make nice, fuzzy-looking hair. Normally you would use a linear blend texture for this.

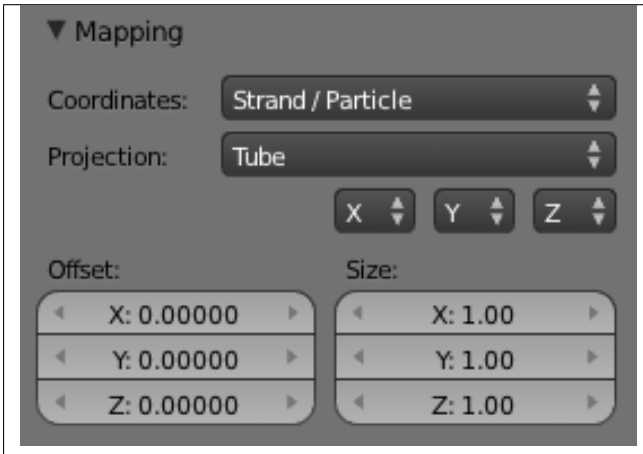


Fig. 2.1500: Fading a strand to alpha.

Fig. 2.1501: The render result.

You may of course set any attribute you like, especially color. Be careful with specularity; hairs tend to get too shiny.

Strand Render Simplification

If you use Strand Primitives (*Strand render* button) and have activated *Interpolated Children*, the *Child Simplification* option appears. The strand render has options to remove child strands as the object's faces become smaller.

Fig. 2.1502: Strand render child simplification.

Reference Size This is the approximate size of the object on screen (in pixels), after which simplification starts.

Rate How fast strands are removed.

Transition The transition period for fading out strands as they are removed.

Viewport This removes strands on faces that are outside of the viewport.

Rate Controls how fast these are removed.

Options

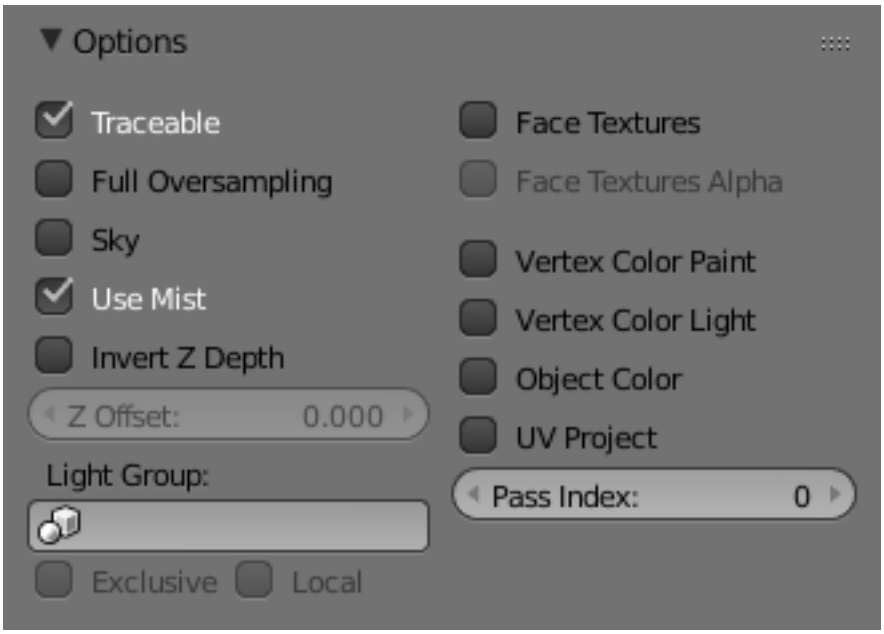


Fig. 2.1503: Material Options panel.

This panel provides a series of control options concerning how objects using this material will appear in the rendered image. All controls are default "Off" unless otherwise stated.

Traceable Include this material and the geometry that uses it in ray-tracing calculations. See *Transparency* for details of ray-tracing.

Full Oversampling Force this material to render full shading and textures for all *anti-aliasing* samples.

Sky Render this material with zero alpha, but with *sky background* in place (scanline only)

Use Mist Use *mist* on this material (see "World Settings" for more details)

Invert Z depth Render material's faces with an inverted Z buffer (scanline only)

Z Offset Give faces an artificial Z offset for Z transparency.

Light Group Limit lighting to lamps in this *light group*.

Exclusive Uses the *light group* exclusively. These lamps will be excluded from other scene lighting.

Local When linked in, uses local *light group* with the same name.

Face Textures Replace object's base color with color from *UV map* image textures.

Face Textures Alpha Replace object's base alpha with alpha from *UV map* image textures.

Vertex Color Paint Replace object's base color with *vertex paint* colors (multiply with 'texture face' face assigned textures)

Vertex Color Light Add *vertex paint* colors as additional lighting. (This can be used to produce good incandescence effects).

Object Color Modulate the result with a per object color. See *Object Display panel*.

UV Project Use to ensure UV interpolation is correct for camera projections (use with *UV project* modifier).

Pass Index Index number for the Material Index render pass.

Shadows

The shadows that appear in a scene are affected by a combination of the layout of objects, the shape of the objects, the materials of the objects, and the lighting. In Blender, the Display Mode (Single Texture, Multitexture, or GLSL) also affects the appearance of shadows. See *Shadows* for a more complete description of this subject.

Tip: Shadows in 3D mode

To see shadows in 3D (textured) mode, you must have switched to GLSL mode before making any materials. In MultiTexture mode, shadows only appear in the rendered image: none of these can be seen in the preview image.

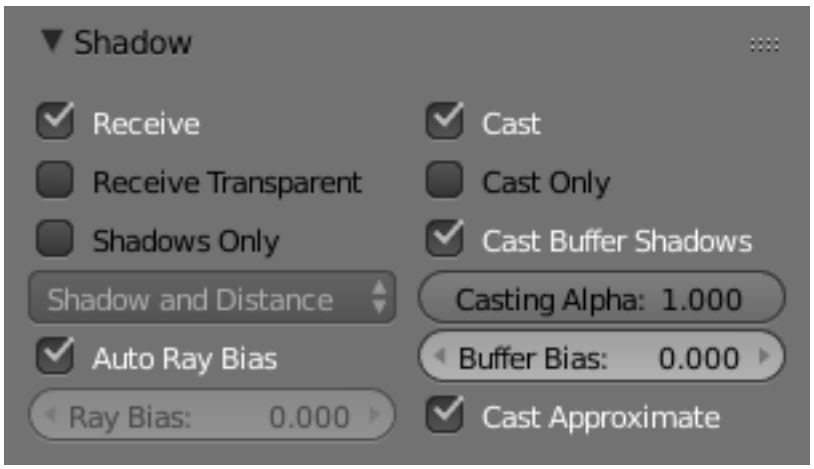


Fig. 2.1504: Shadow Panel.

The Shadow panel in the Materials Properties editor (see Fig. *Shadow Panel.*) controls the effects that the material can have on the shadows that appear in the scene. The various properties are described in the sections below.

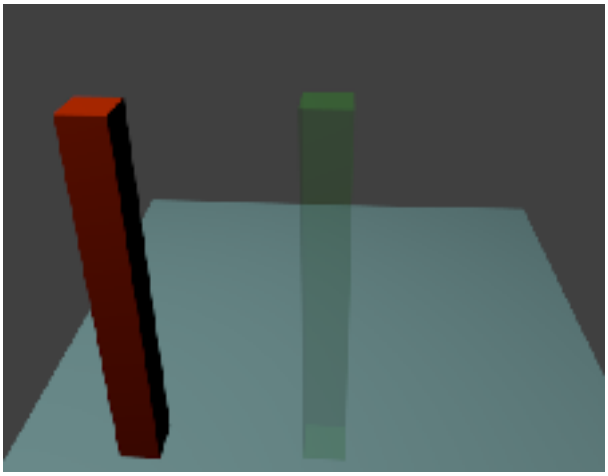


Fig. 2.1505: Scene with all shadow properties off.

Options

The following properties can be set for each individual material with which objects in the scene are shaded. The effects are illustrated with rendered images for a simple scene (Fig. *Scene with all shadow properties off.*) consisting of two "posts", one with a red (totally non-transparent) material; one green (partially transparent) material, set up on a light blue plane to receive the shadows. The illustrations were all taken in Blender Render engine, with Multitexture mode.

Shadow Receiving Object Material

The following options affect the material that receives shadows:

Receive Allows this material to receive full-intensity shadows (Fig. *Plane with Receive.*).

Receive Transparent Allows this material to receive shadows whose intensity is modified by the transparency and color of the shadow-casting object (Fig. *Plane with Receive and Receive Transparency.*).

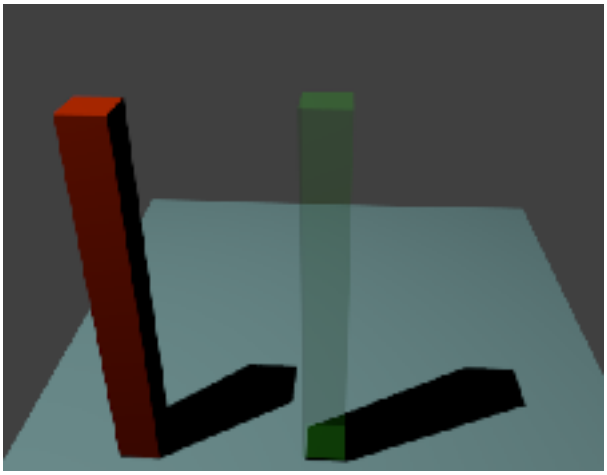


Fig. 2.1506: Plane with Receive.

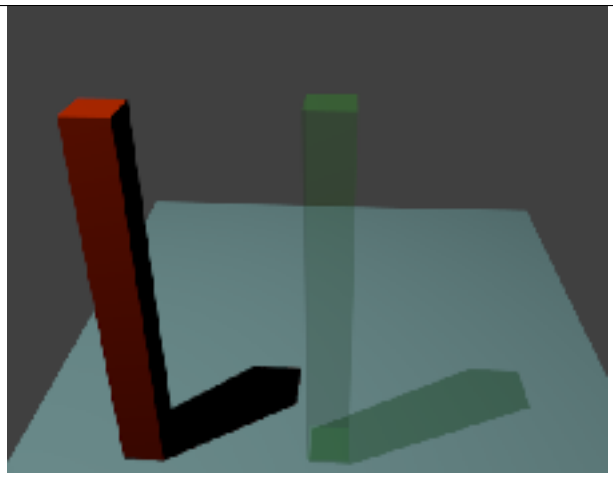


Fig. 2.1507: Plane with Receive and Receive Transparency.

Shadow Casting Object Material

The following options affect the material that casts shadows:

Cast Only Causes objects with the material to only cast a shadow, and not appear in renders. (Fig. *Posts with Cast Only.*).

Casting Alpha Sets the Alpha of shadow casting. Used for irregular and deep shadow buffering.

Shadows Only Renders shadows as materials alpha value, making materials transparent, except for areas where it receives shadows from other objects, and also it retains its own transparency (Fig. *Posts with Shadows Only.*). Note the faint image of the partly-transparent post.

Shadow Only Type Set the type of shadows used when Shadows Only is enabled.

- Shadow and Distance
- Shadow Only
- Shadows and Shading

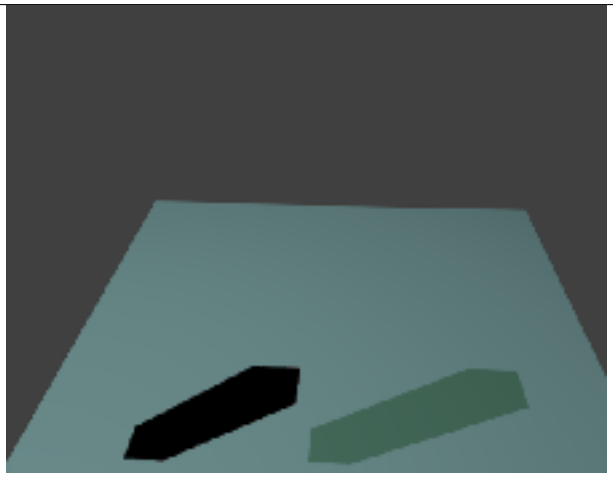


Fig. 2.1508: Posts with Cast Only.

Fig. 2.1509: Posts with Shadows Only.

Buffered Shadow Options

In addition to the shadow options described above, there are further material properties which control buffered shadow features. See section on *Spot Buffered Shadows* for further discussion of these techniques.

Cast Buffer Shadow Casts shadows from shadow buffer lamps.

Buffer Bias Multiplication factor for Buffer shadows (0 = ignore).

Auto Ray Bias Prevent raytraced shadow errors on surfaces with smooth shaded normals.

Ray Bias Shadow raytracing bias value to prevent terminator artifacts on shadow boundary.

Cast Approximate Allow this material to cast shadows when using approximate ambient occlusion.

Material Nodes

Introduction to Nodes

In addition to creating materials as just described using all the settings on all the materials panels, Blender allows you to create a material by routing basic materials through a set of nodes. Each node performs some operation on the material, changing how it will appear when applied to the mesh, and passes it on to the next node. In this way, very complex material appearances can be achieved.

You should already be familiar with general material concepts and how to create materials/textures using the material menu. You should also have a general understanding of the texture coordinate systems available in Blender (e.g. Generated, UV, etc.). Also, many aspects of a node will be skipped here because in later sections you will see the function expanded upon. Each section builds off the previous.

To start, the node system does not make the material menu obsolete. Many features and material settings are still only accessible through the material panel (e.g. Ray Mirror). However, with the advent of nodes, more complex and fantastic materials can be created since we now have greater control.

Just in case you are not (yet) familiar with the concepts: when you create a system of nodes, you are describing a data-processing pipeline of sorts, where data "flows from" nodes which describe various *sources,* "flows through" nodes which represent various processing and filtering stages, and finally "flows into" nodes which represent outputs or destinations. You can connect the nodes to one another in many different ways, and you can adjust "knobs, " or parameters, that control the behavior of each node. This gives you a tremendous amount of creative control. And, it will very quickly become intuitive.

Having said all that, let us begin with a normal material.

Here we have the standard material we have added to a cube mesh. We could, as we have in the past, add color and other settings to this material and it would certainly look nice. But let us say we are just not getting what we are looking for? What if we want to control the creation more tightly or add more complexity? Here is where nodes come in.

Making this node map is accomplished by working in a *Node Editor*. This section covers:

- Enabling Material Nodes.
- The Node Editor, its basic controls, and working with nodes.
- The specific types of nodes available for materials.

Accessing The Node Editor

First lets enter the *node editor* and make sure that the node editor has the material node button (the sphere icon) pressed, not the composite or texture node buttons.

Enabling Node Materials in the Material Buttons

Let us take the base material and hit the Nodes button next to the material name in the material panel or the node editor. You will see a change in the material panel.

Fig. 2.1510: Use material nodes button.

What you have just done is told Blender to make the material you were on to become the node tree. Most of the panels we normally find in the material menu are now gone.

If you switch to the *Compositing screen* with `Ctrl-Left`, if you are on the default screen, you will find a *Node Editor* on the top half of the screen. When you enabled material nodes, a material node and an output node were automatically added to the node editor.

You can also split the 3D View in the default screen in two and change one into a *Node Editor*.

It is important to note that you can add a new material (which you can edit and change like any other material in the material panel), add an already created material or append a material from another blend-file, and also use the material that you used to create the node tree.

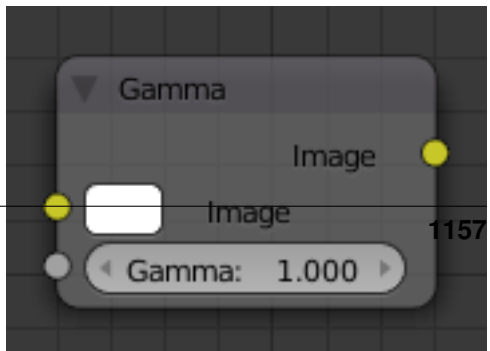
Here, we added a new material in the *Node editor* "Material.001", and as we did, we can access the properties of this material in the material's menu.

Node Types

Color Nodes

Gamma Node

Use this node to apply a gamma correction.



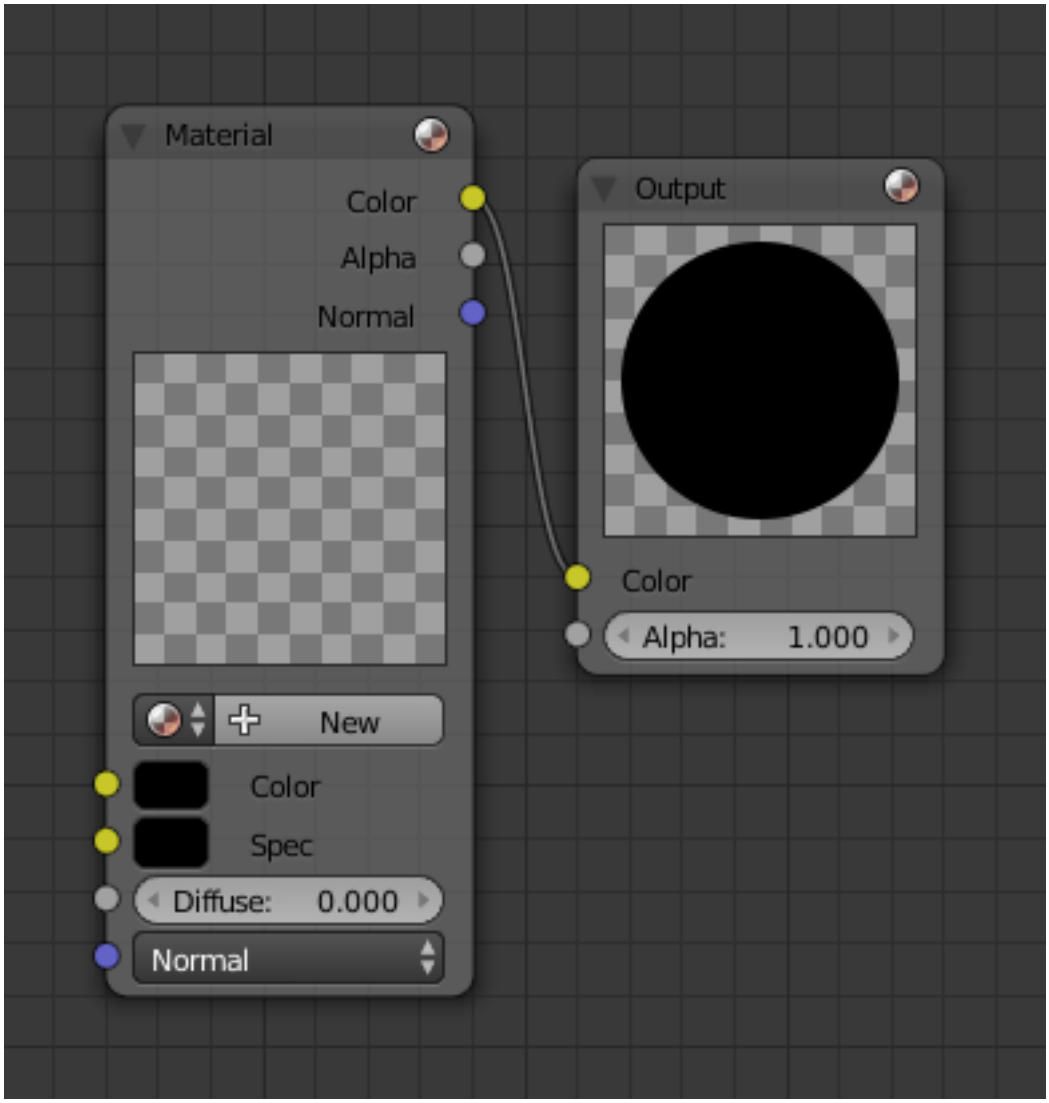


Fig. 2.1511: Default nodes.

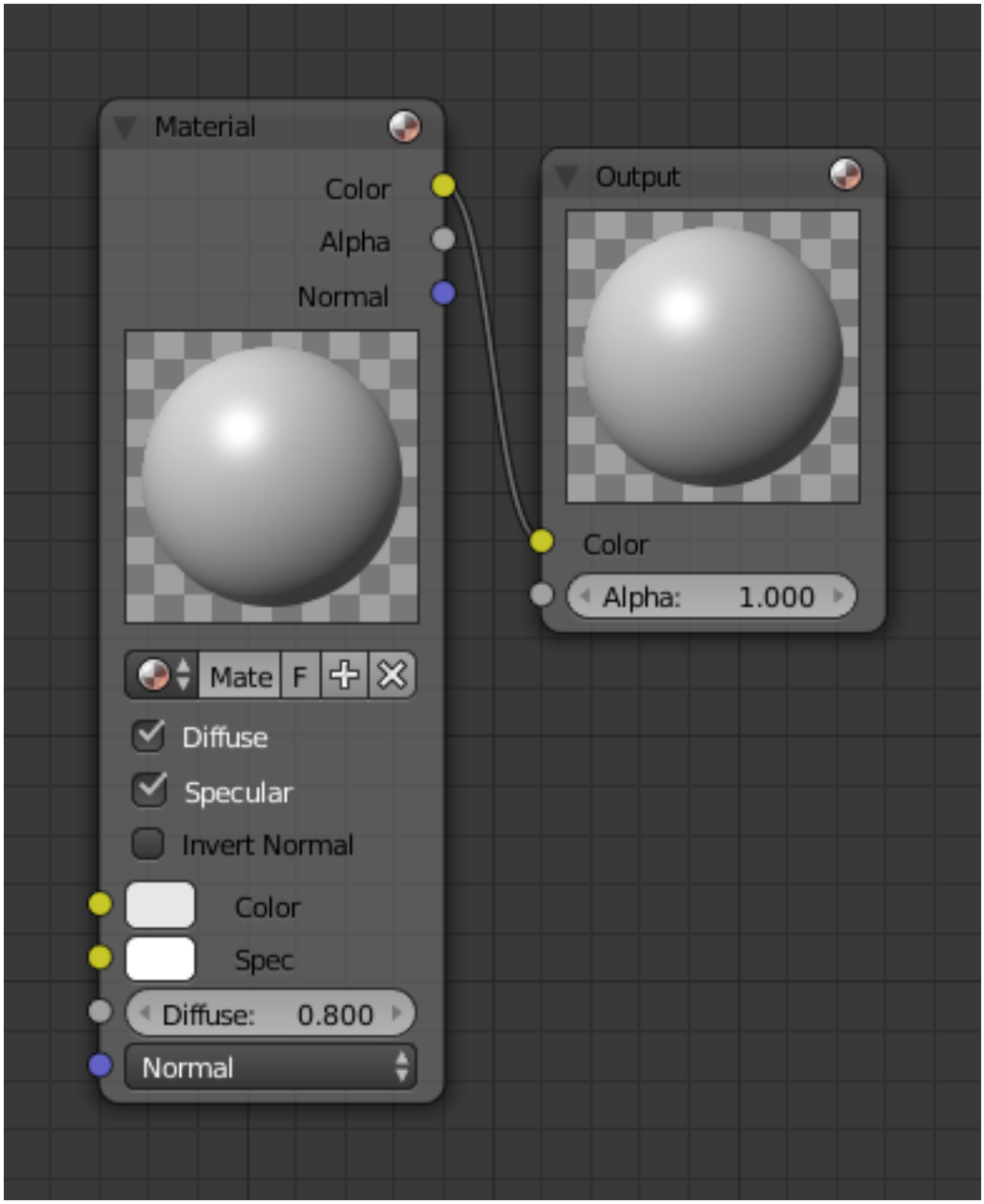


Fig. 2.1512: A first material added to the node setup.

Inputs

Image Standard image input.

Gamma An exponential brightness factor.

Properties

This node has no properties.

Outputs

Image Standard image output.

Examples

Fig. 2.1514: Example of Gamma node.

Hue Saturation Value Node

This node applies a color transformation in the HSV color space. Called "Hue Saturation Value" in shader and texture context.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Properties

The transformations are relative shifts. In the shader and texture context the following properties are available as input sockets.

Hue Specifies how the hue rotation of the image. 360° are mapped to (0 to 1). The hue shift of 0 (-180°) and 1 (+180°) have the same result.

Saturation A saturation of 0 removes hues from the image, resulting in a grayscale image. A shift greater 1.0 increases saturation.

Value Value is the overall brightness of the image. De/Increasing values shift an image darker/lighter.

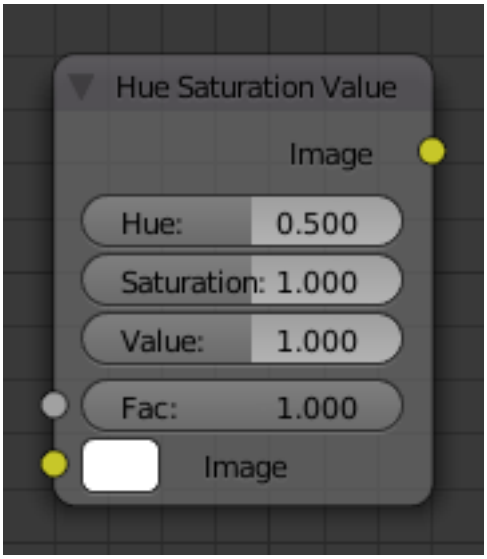


Fig. 2.1515: Hue Saturation Node.

Outputs

Image Standard image output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are vice versa A blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (recall that white, minus blue, equals yellow). A yellow image, with a Hue setting at 0 or 1, is blue.

Hue and Saturation work together. So, a Hue of 0.5 keeps the blues the same shade of blue, but *Saturation* can deepen or lighten the intensity of that color.

Gray & White are neutral hues A gray image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with *Value*. This applies to all shades of gray, from black to white; wherever the values are equal.

Changing the effect over time The Hue and Saturation values can be animated with a *Time Node* or by animating the property.

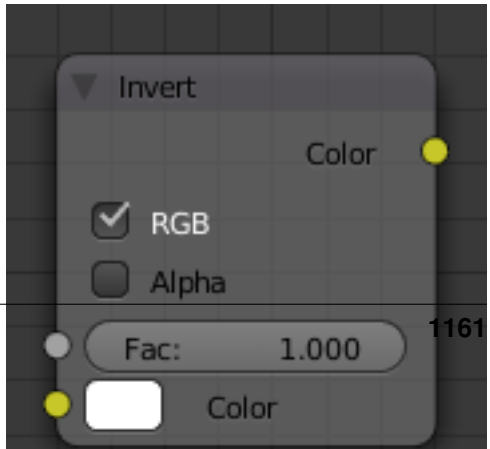
Note: Tinge

This HSV node simply shifts hues that are already there. To colorize a gray image, or to add a tint to an image, use a mix node to add in a static color from an RGB input node with your image.

HSV Example

Invert Node

This node inverts the colors in the input image, producing a negative.



Inputs

Factor Controls the amount of influence the node exerts on the output image.

Color Standard image input.

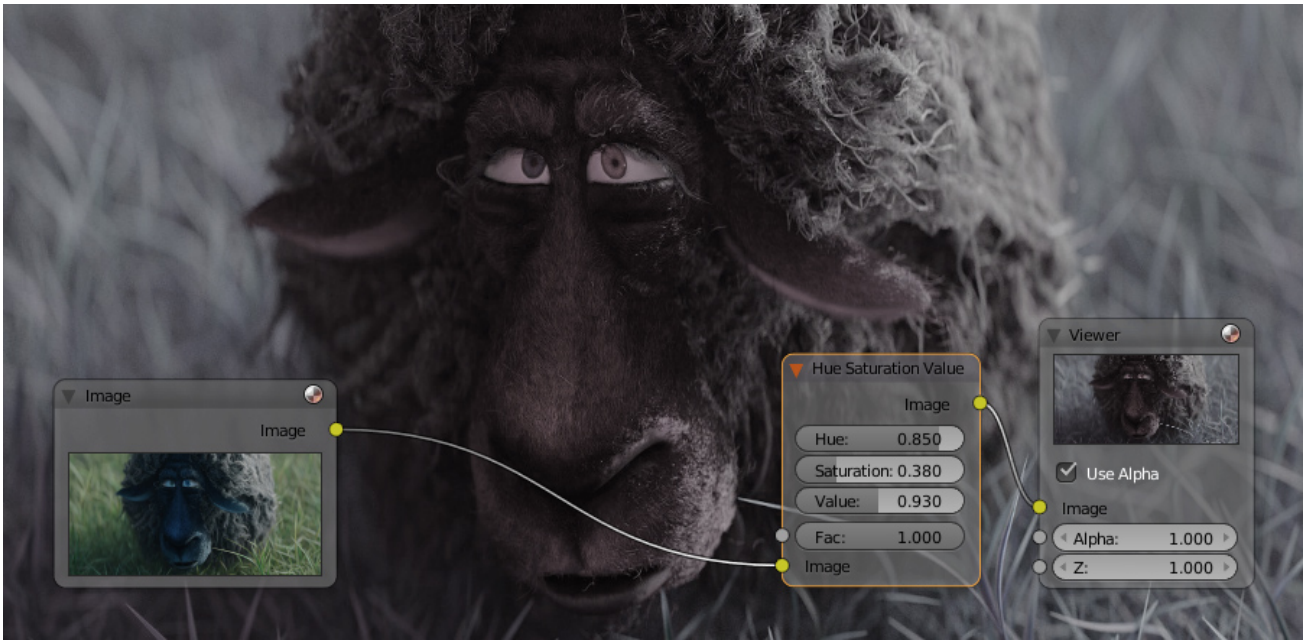


Fig. 2.1516: A basic example.

Properties

In the compositing context this node has the following properties.

RGB De/activation of the color channel inversion.

Alpha De/activation of the alpha channel inversion.

Outputs

Color Standard image output.

Mix Node

This node mixes images by working on the individual and corresponding pixels of the two input images. Called "MixRGB" in the shader and texture context.

Inputs

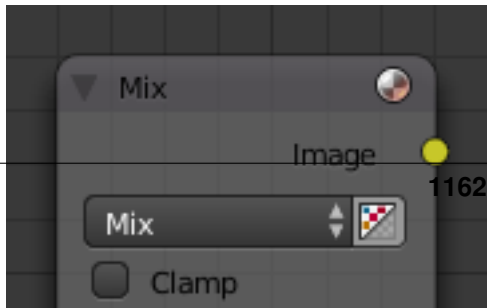
Factor Controls the amount of influence the node exerts on the output image.

Image The background image. The image size and resolution sets the dimensions of the output image.

Image The foreground image.

Properties

Mix The Blend types could be selected in the select menu. See *Color Blend Modes* for details on each blending mode.



Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Dodge, Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Use Alpha If activated, by clicking on the *Color and Alpha* icon, the Alpha channel of the second image is used for mixing. When deactivated, the default, the icon background is a light gray. The alpha channel of the base image is always used.

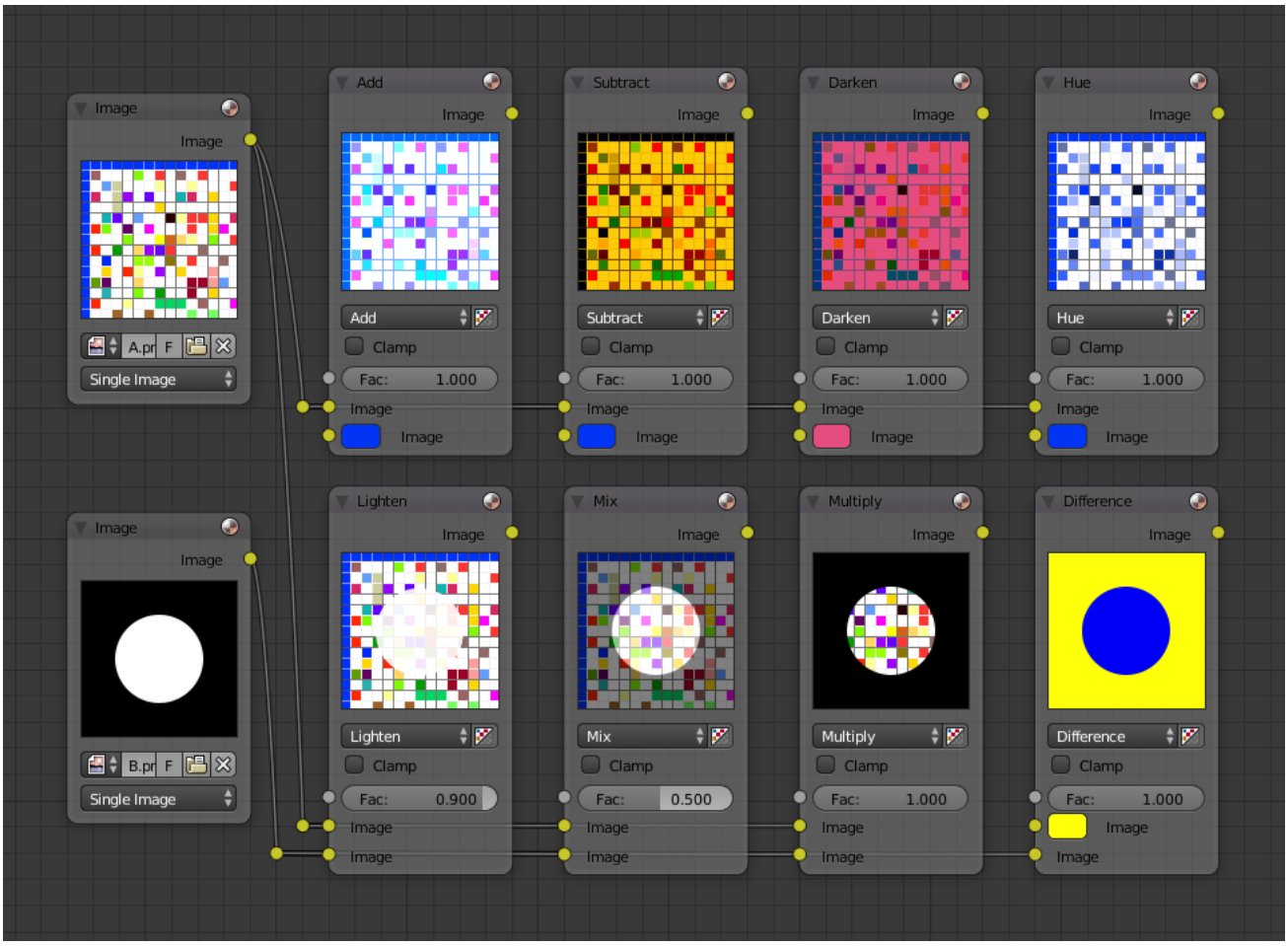
Clamp Limit the highest color value to not exceed 1.

Outputs

Image Standard image output.

Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.



Some explanation of the mixing methods above might help you use the Mix node effectively:

Add adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.

Subtract Taking Blue away from white leaves Red and Green, which combined make Yellow. Taking Blue away from Purple leaves Red. Use this to desaturate an image. Taking away yellow makes an image bluer and more depressing.

Multiply Black (0.00) times anything leaves black. Anything times White (1.00) is itself. Use this to mask out garbage, or to colorize a black-and-white image.

Hue Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').

Mix Combines the two images, averaging the two.

Lighten Like bleach makes your whites whiter. Use with a mask to lighten up a little.

Difference Kinda cute in that it takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two verrry similar images to see what had been done to one to make it the other; sorta like a change log for images. You can use this to see a watermark (see *Watermark images*) you have placed in an image for theft detection.

Darken with the colors set here, is like looking at the world through rose-colored glasses.

Contrast Enhancement

Here is a small map showing the effects of two other common uses for the RGB Curve: *Darken* and *Contrast Enhancement*. You can see the effect each curve has independently, and the combined effect when they are *mixed* equally.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast.

In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB (S shaped) curve scales the output such that middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus, whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better.

Watermark images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising. Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting service, or simply to track their image's proliferation throughout the web. Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

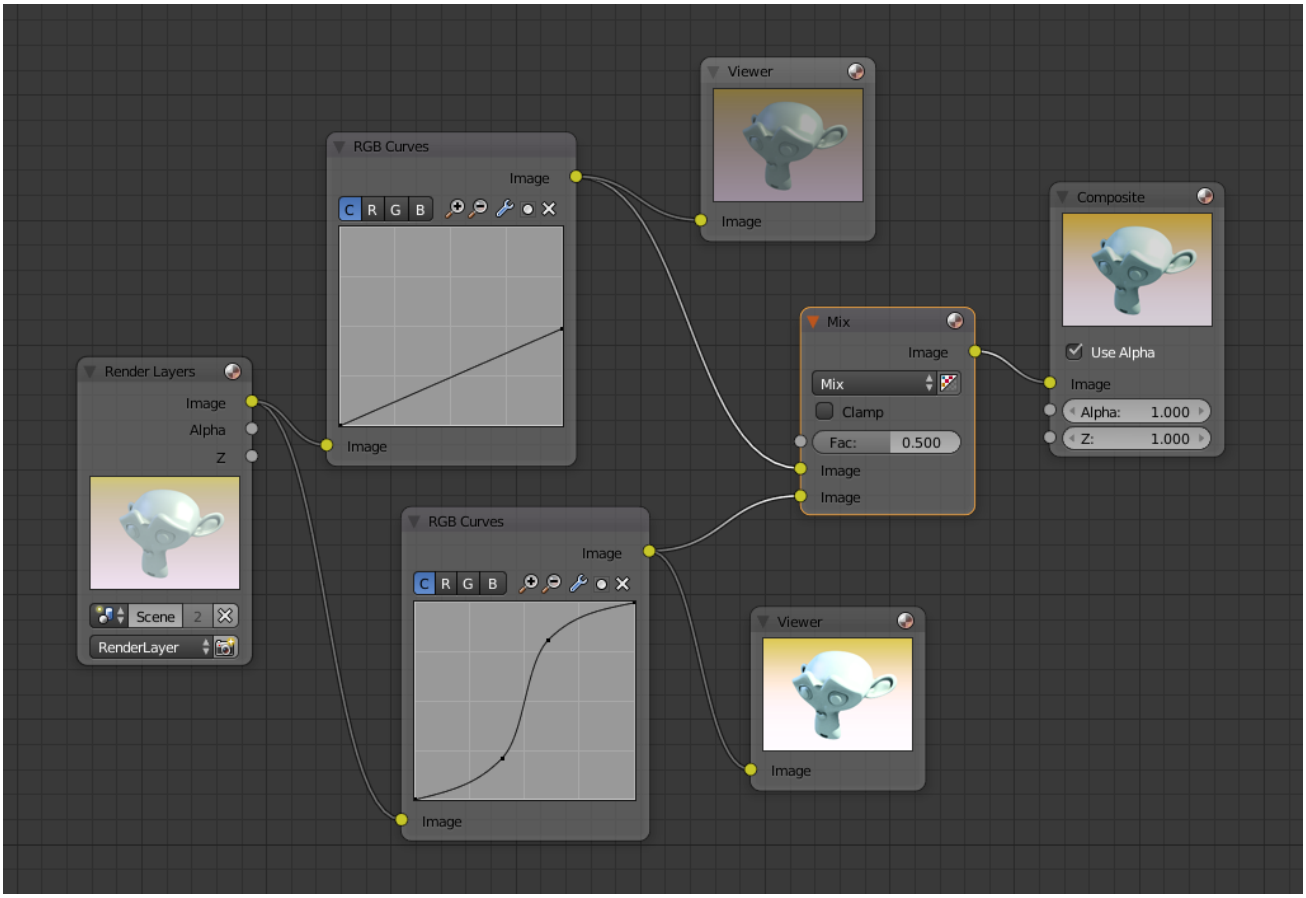


Fig. 2.1519: Example node setup showing “Darken”, “Enhance Contrast” and “Mix” nodes for composition.

Encoding Your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it is up to you. In the example below, we are encoding the watermark in a specific location in the image using the *Translate* node; this helps later because we only have to look at a specific location for the mark. We then use the RGB to BW node to convert the image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one-tenth of its original intensity. The Add node adds the corresponding pixels, make the ones containing the mark ever-so-slightly brighter.

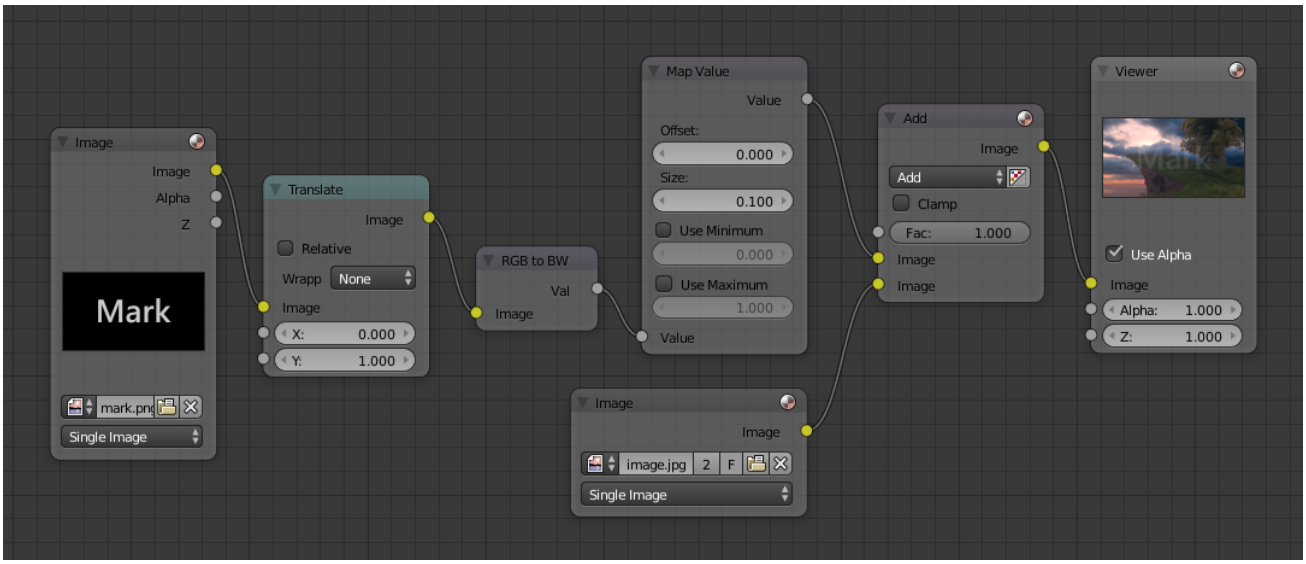


Fig. 2.1520: Embedding your mark in an Image using a Mark and Specific Position.

Of course, if you *want* people to notice your mark, do not scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Note: Additional uses

You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface, or as a special effect to make words appear on the possessed girl's forearm. yuk.

Decoding an Image for your Watermark

When you see an image that you think might be yours, use the node map below to compare it to your stock image (pre-watermarked original). In this map, the Mix node is set to Difference, and the Map Value node amplifies any difference. The result is routed to a viewer, and you can see how the original mark stands out, clear as a bell:

Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding map

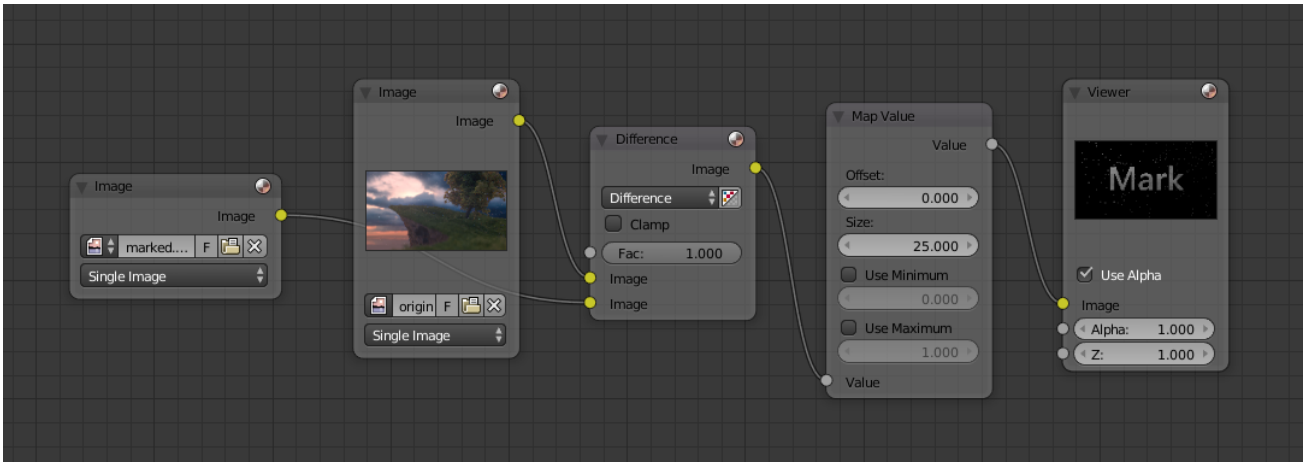


Fig. 2.1521: Checking an image for your watermark.

in one scene, and the decoding map in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to JPEG compression of 50%.

RGB Curves Node

This node allows color corrections for each color channel and levels adjustments in the compositing context.

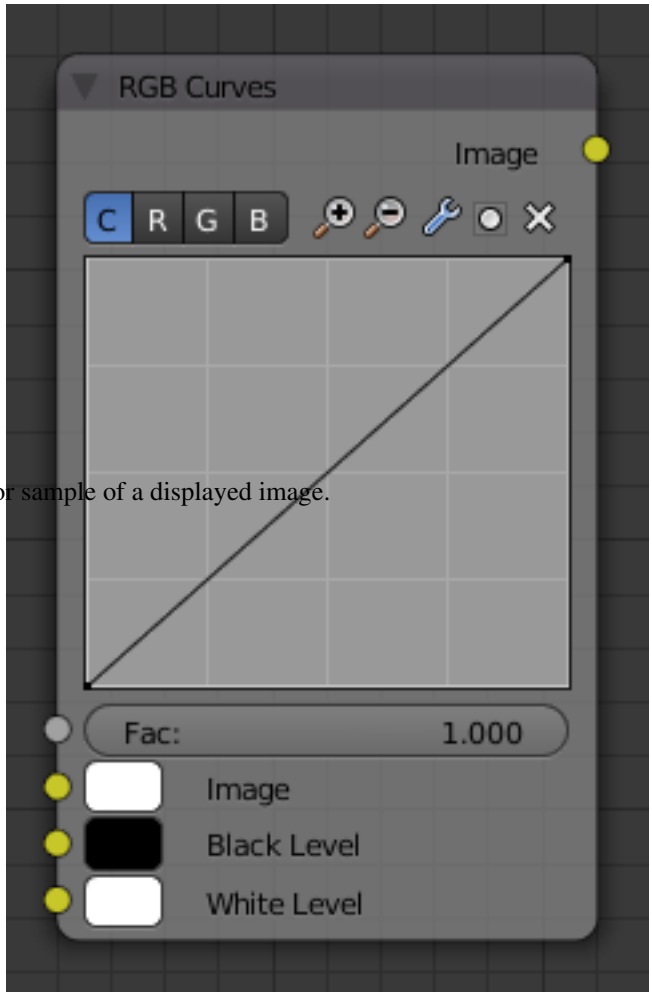


Fig. 2.1522: RGB Curves Node.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Black Level Defines the input color that is (linear) mapped to black.

White Level Defines the input color that is (linear) mapped to white.

Tip: To define the levels, use the *eye dropper* to select a color sample of a displayed image.

Properties

Channel Clicking on one of the channels displays the curve for each.

C (Combined RGB), R (Red), G (Green), B (Blue), L (Luminance)

Curve A Bézier curve that varies the input levels (x-axis) to produce an output level (y-axis). For the curve controls see: *Curve widget*.

Outputs

Image Standard image output.

Examples

Here are some common curves you can use to achieve desired effects:

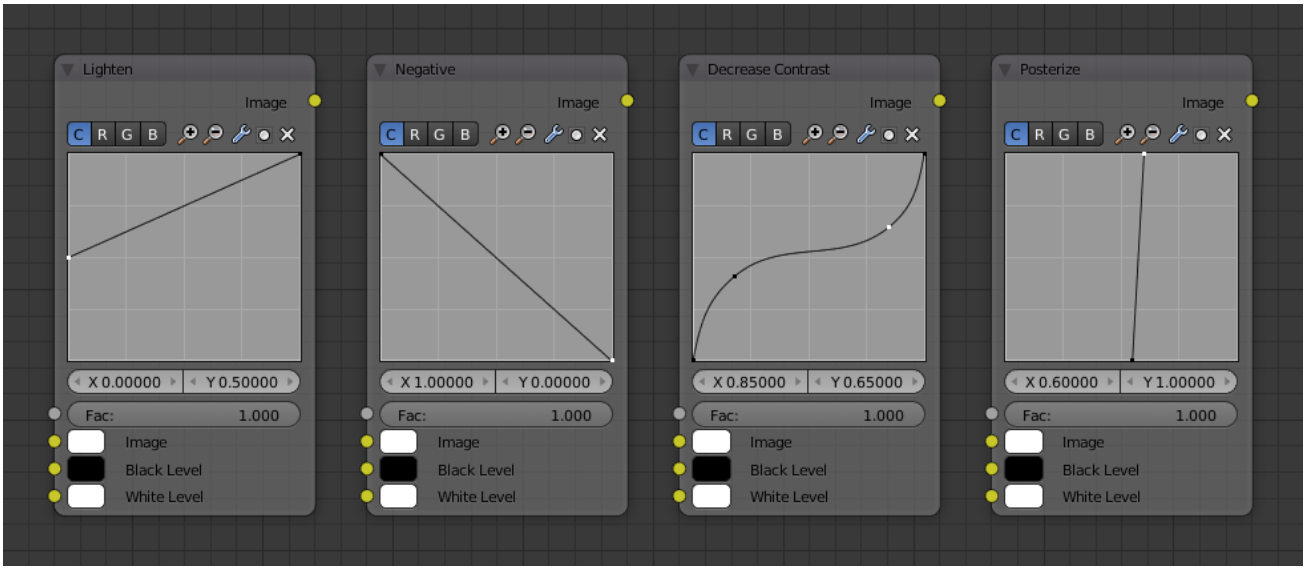


Fig. 2.1523: From left to right: 1. Lighten 2. Negative 3. Decrease Contrast 4. Posterize.

Color correction using Curves

In this example, the image has way too much red in it, so we run it through an RGB node and reduce the Red channel by about half.

We added a middle dot so we could make the line into a sideways exponential curve. This kind of curve evens out the amount of a color in an image as it reaches saturation. Also, read on for examples of the Darken and Contrast Enhancement curves.

Color correction using Black/White Levels

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the UV/Image editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side-effect of setting Black and White Levels.

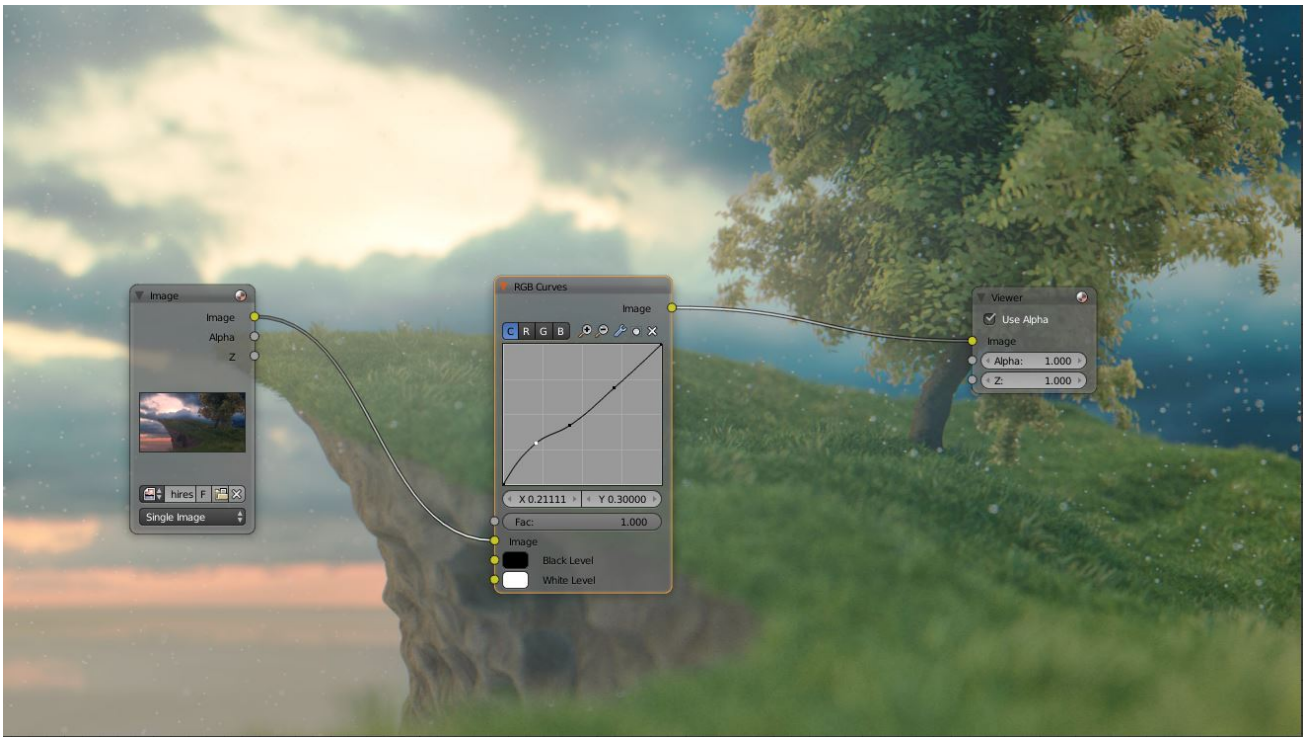


Fig. 2.1524: Color correction with curves.

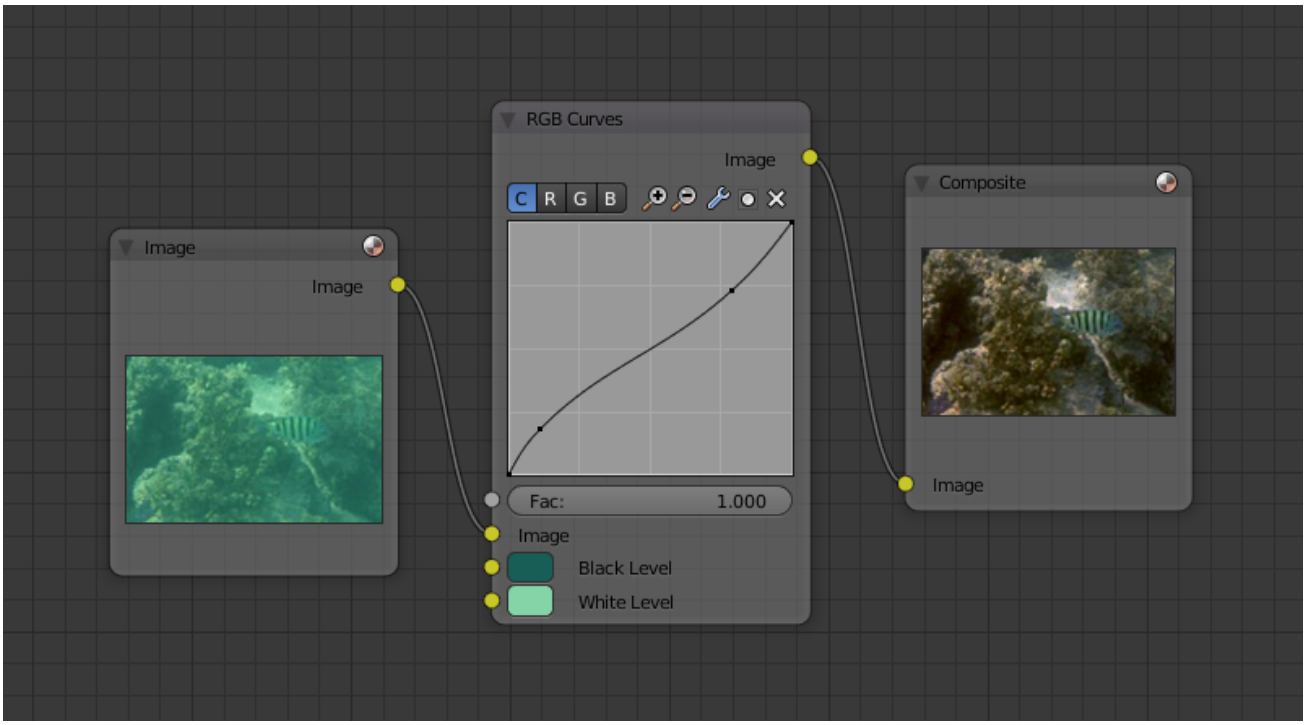


Fig. 2.1525: Color correction with Black/White Levels.

Effects

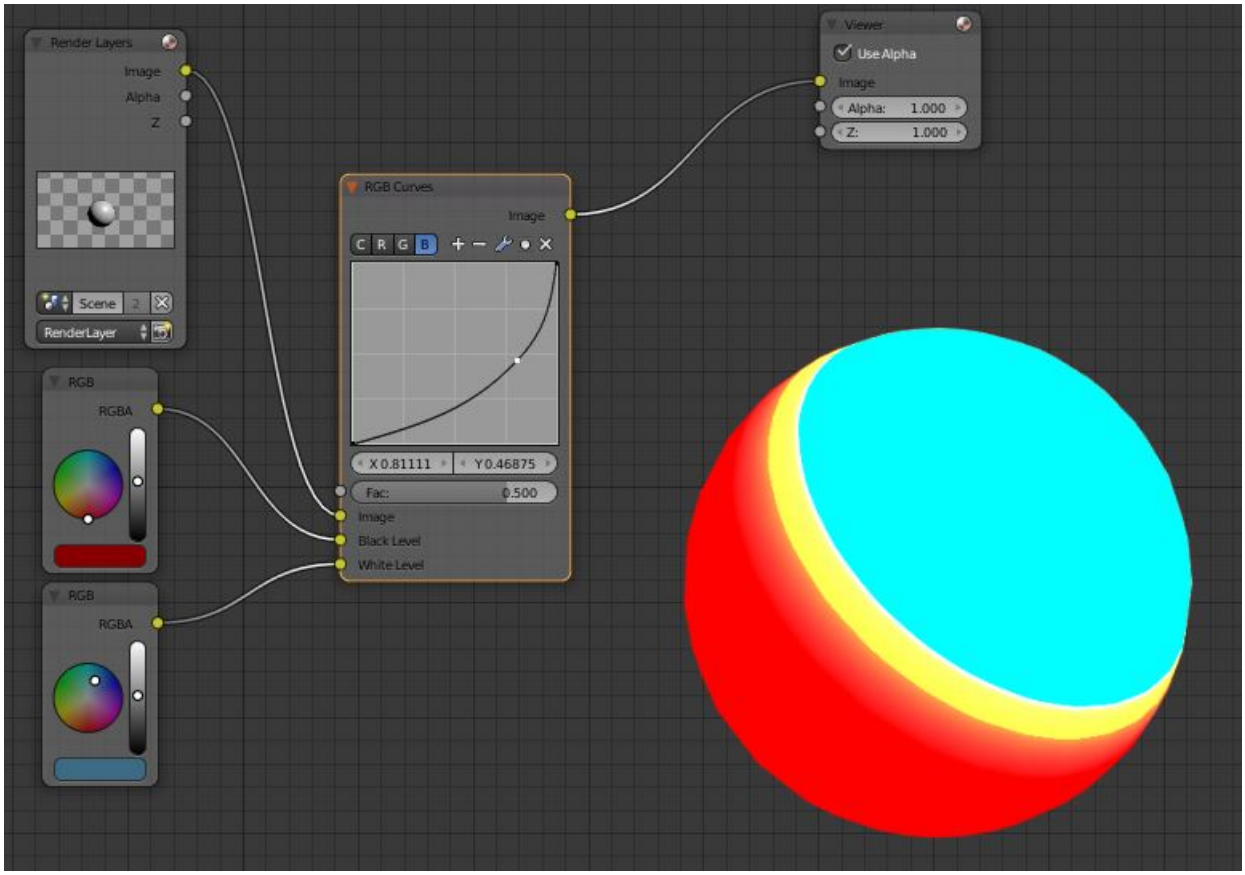


Fig. 2.1526: Changing colors.

Curves and Black/White Levels can also be used to completely change the colors of an image.

Note that e.g. setting Black Level to red and White Level to blue does not simply substitute black with red and white with blue as the example image might suggest. Levels do color scaling, not substitution, but depending on the settings they can result in the described color substitution.

(What really happens when setting Black Level to pure red and White Level to pure blue is that the red channel gets inverted, green gets reduced to zero and blue remains unchanged.)

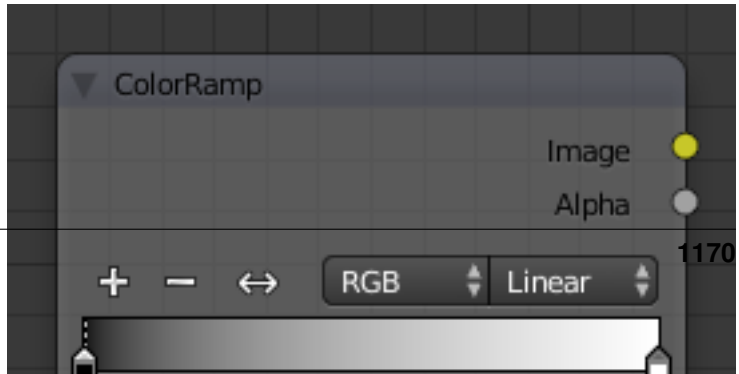
Because of this, the results of setting arbitrary Black/White Levels or RGB curves is hard to predict, but can be fun to play with.

Converter Nodes

As the name implies, these nodes convert the colors in the material in some way.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors with the use of a gradient.



Inputs

Factor The Factor input is used as an index for the color ramp.

Properties

Color Ramp For controls see *Color Ramp Widget*.

Outputs

Image Standard image output.

Alpha Standard alpha output.

Examples

Creating an Alpha Mask

A powerful but often overlooked feature of the Color Ramp is to create an Alpha Mask, or a mask that is overlaid on top of another image, and, like a mask, allows some of the background to show through. The example map below shows how to use the Color Ramp node to do this:

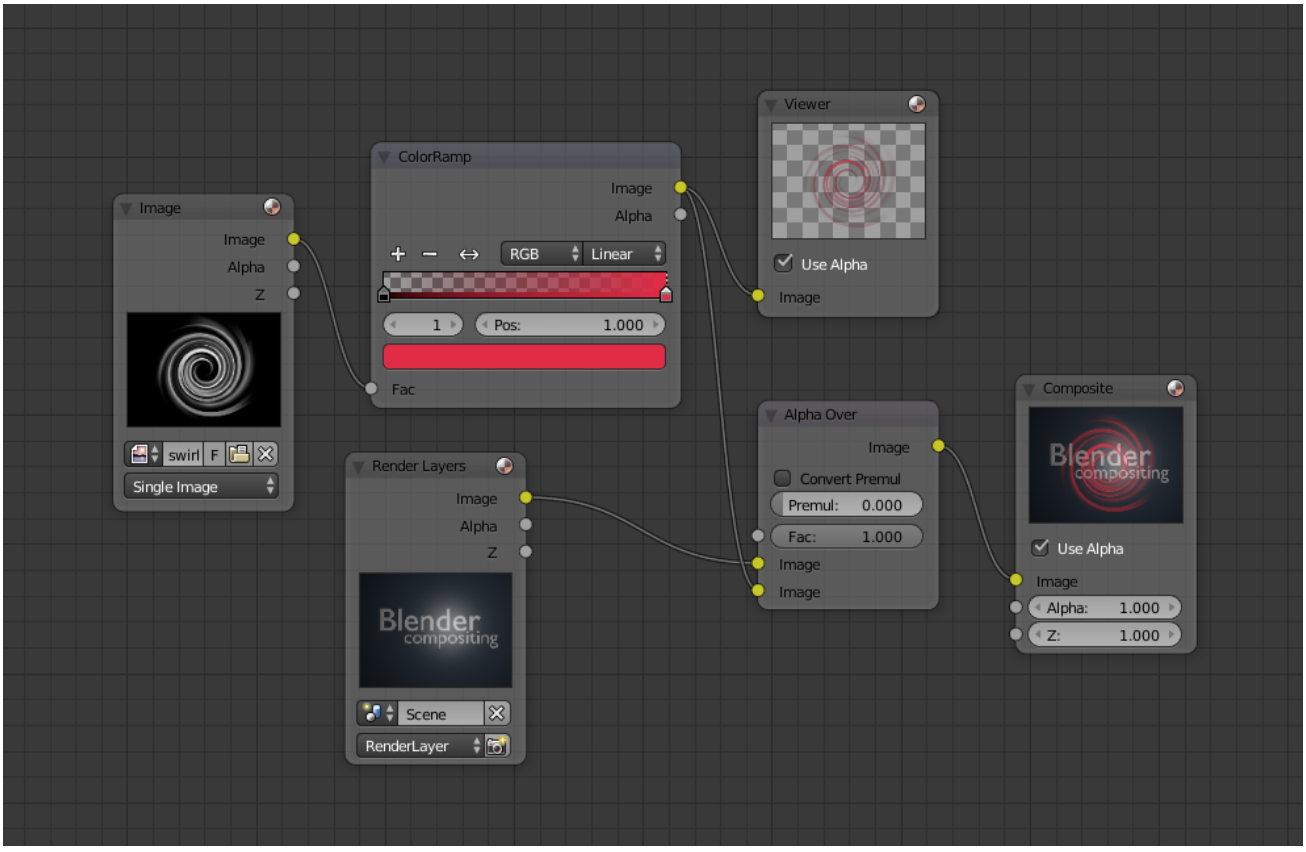


Fig. 2.1528: Using the Color Ramp node to create an alpha mask.

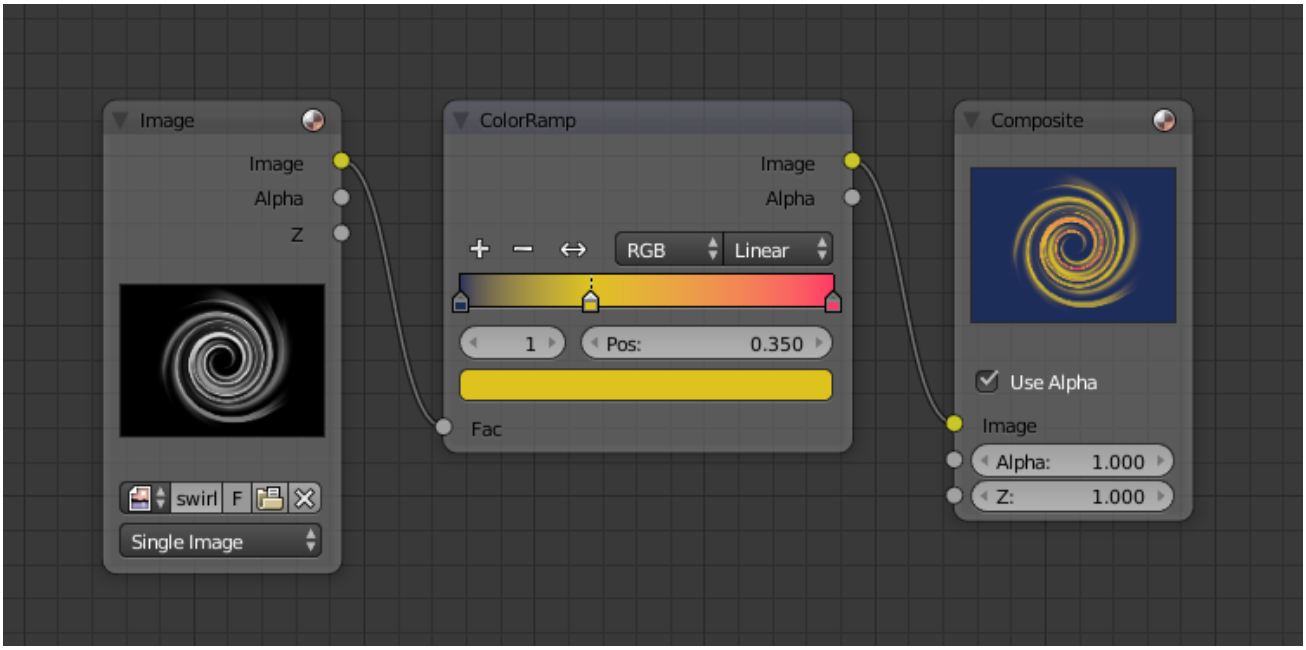
In the map above, a black and white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*. (Technically, we should have converted the image to a value using the RGB-to-BW node, buy hey, this works just as well since we are using a BW image as input.)

We have set the Color Ramp node to a purely transparent color on the left end of the spectrum, and a fully Red color on the right. As seen in the viewer, the Color Ramp node puts out a mask that is fully transparent where the image is black. Black is zero, so Color Ramp uses the color at the left end of the spectrum, which we have set to transparent. The Color Ramp image is fully red and opaque where the image is white (1.00).

We verify that the output image mask is indeed transparent by overlaying it on top of a other image.

Colorizing an Image

The real power of Color Ramp is that multiple colors can be added to the color spectrum. This example compositing map takes a boring BW image and makes it a flaming swirl!



In this example, we have mapped the shades of gray in the input image to three colors, blue, yellow, and red, all fully opaque (Alpha of 1.00). Where the image is black, Color Ramp substitutes blue, the currently selected color. Where it is some shade of gray, Color Ramp chooses a corresponding color from the spectrum (bluish, yellow, to reddish). Where the image is fully white, Color Ramp chooses red.

Combine/Separate Nodes

All of these nodes do essentially the same thing:

- Separate: Split out an image into its composite color channels.
- Combine: Re/combine an image from it is composite color channels.

This nodes could be use this to manipulate on each color channel independently. Each type is differentiate in the applied *color space*.

In compositing and texture context each node supports the Alpha channel. In the texture context only RGB color space is available. In the shading context of the Blender internal adds HSV and the Cycles shading context offers an additional pair of nodes to combine/separate a vector (XYZ).

The Combine nodes could also be used to input single color values. For RGBA and HSVA color spaces it is recommended to use the *RGB Node*. Some common operation could easier executed with the *Color Nodes*.

Separate/Combine RGBA Node

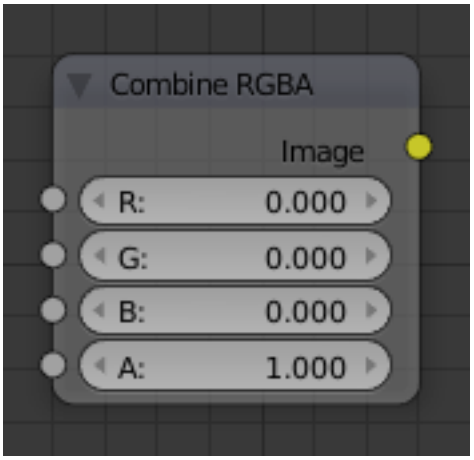


Fig. 2.1529: Combine RGBA Node.

Input/ Output

Image Standard image in/output.

- R (Red)
- G (Green)
- B (Blue)
- A (Alpha)

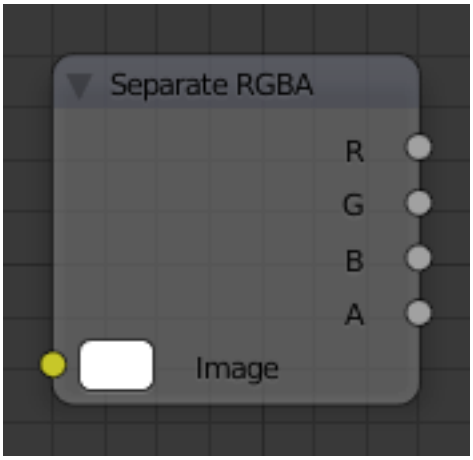


Fig. 2.1530: Separate RGBA Node.

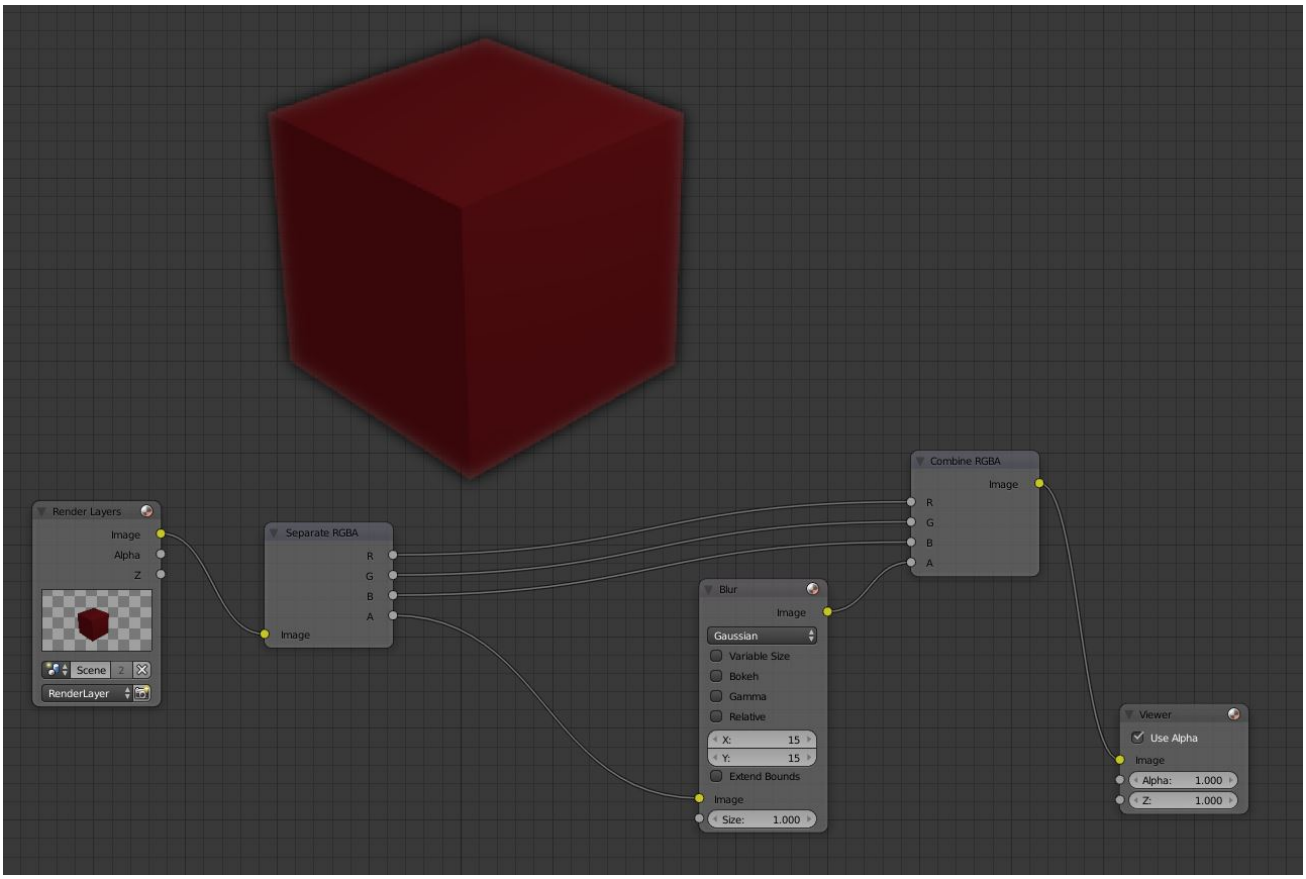
Properties

This node has no properties.

Examples

In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like anti-aliasing but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a broader scale will make the object appear to "phase" in and out, as an "out-of-phase" time-traveling sync effect.

In this node set up, we make all the reds become green, and all the green both Red and Blue, and remove Blue from the image completely.


Render Layers
Image
Alpha
Z
Scene
RenderLayer
Separate RGBA
R
G
B
A
Image
Combine RGBA
Image
R
G
B
A
Blur
Image
Gaussian
Variable Size
Bokeh
Gamma
Relative
X: 15
Y: 15
Extend Bounds
Image
Size: 1.000
Viewer
Use Alpha
Image
Alpha: 1.000
Z: 1.000

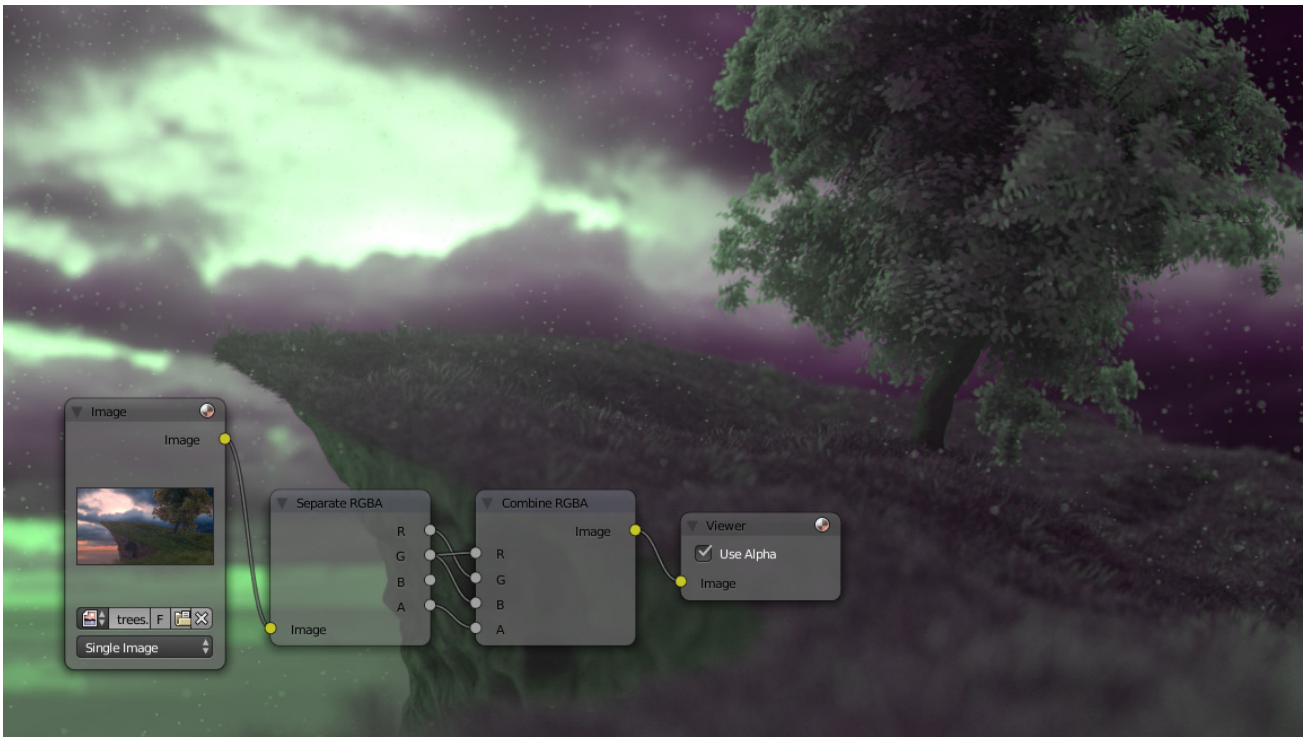


Image
Image
trees. F
Single Image
Separate RGBA
R
G
B
A
Image
Combine RGBA
Image
R
G
B
A
Viewer
Use Alpha
Image

Separate/Combine HSVA Nodes

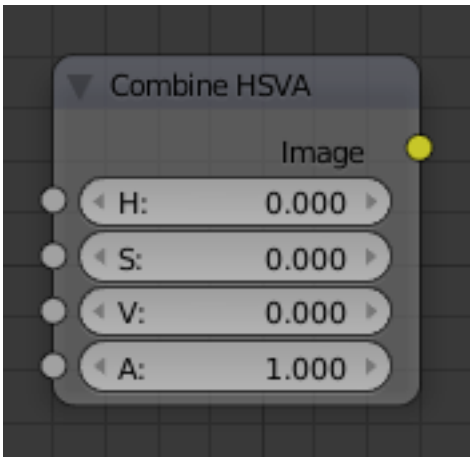


Fig. 2.1531: Combine HSVA Node.

Input/ Output

Image Standard image in/output.

- H (Hue)
- S (Saturation)
- V (Value)
- A (Alpha)

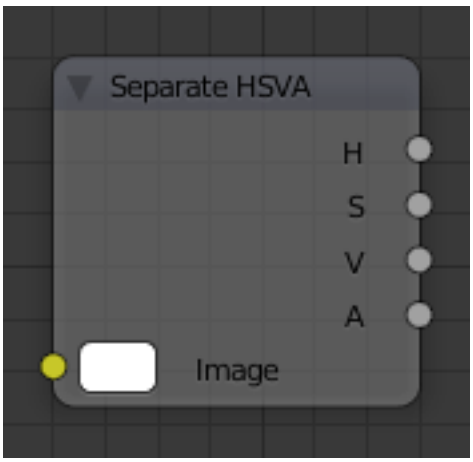


Fig. 2.1532: Separate HSVA Node.

Properties

This node has no properties.

Separate/Combine YUVA Node

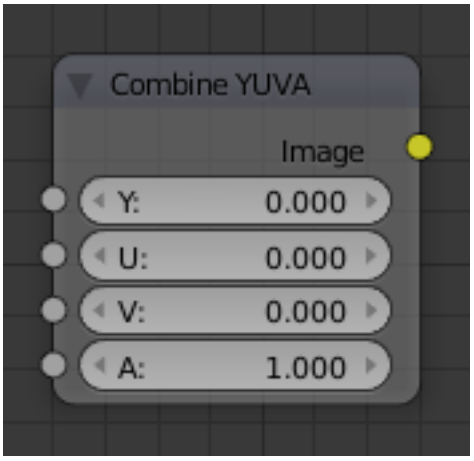


Fig. 2.1533: Combine YUVA Node.

Input/ Output

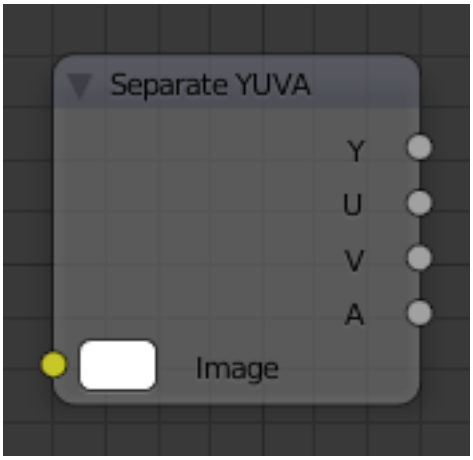


Fig. 2.1534: Separate YUVA Node.

Image Standard image in/output.

- Y (Luminance)
- U (U chrominance)
- V (V chrominance)
- A (Alpha)

Properties

This node has no properties.

Separate/Combine YCbCrA Node

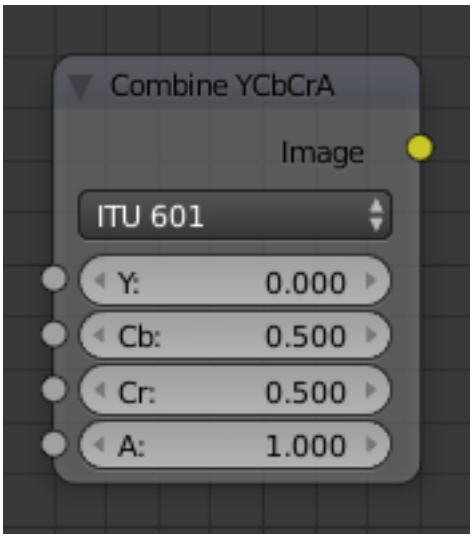


Fig. 2.1535: Combine YCbCrA Node.

Input/ Output

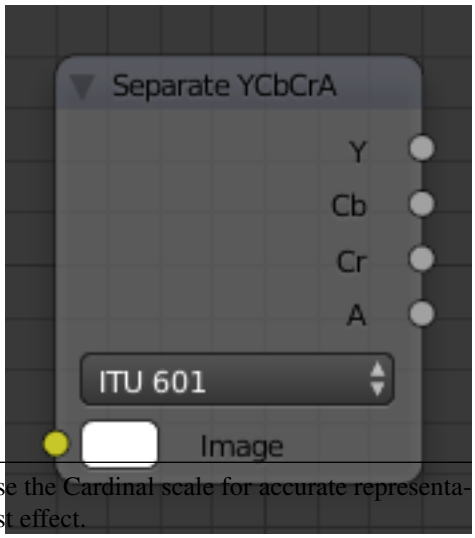


Fig. 2.1536: Separate YCbCrA Node.

Image Standard image in/output.

- Y (Luminance)
- Cb (Chrominance Blue)
- Cr (Chrominance Red)
- A (Alpha)

Properties

Mode ITU 601, ITU 709, Jpeg

Tip: If running these channels through a Color Ramp node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives high-contrast effect.

Math Node

This node performs math operations.

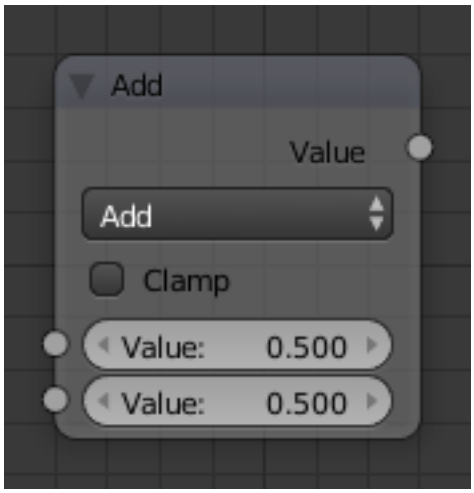


Fig. 2.1537: Math node.

Inputs

Value First numerical value. The trigonometric functions accept values in radians.

Value Second numerical value. This value is **not** used in functions that accept only one parameter like the trigonometric functions, Round and Absolute.

Properties

Operation Add, Subtract, Multiply, Divide, Sine, Cosine, Tangent, Arcsine, Arccosine, Arctangent, Power, Logarithm, Minimum, Maximum, Round, Less Than, Greater Than, Modulo, Absolute.

Clamp Limits the output to the range (0 to 1). See *clamp*.

Outputs

Value Numerical value output.

RGB to BW Node

This node maps a RGB color image to a grayscale by the luminance.

Inputs

Image Color image input.

Properties

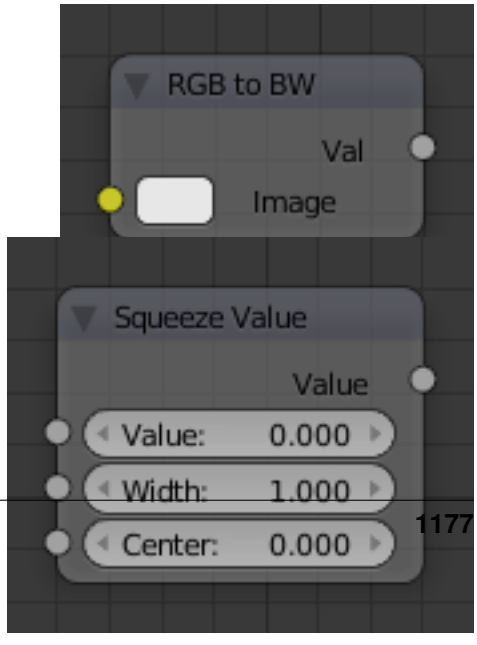
This node has no properties.

Outputs

Value Grayscale value output.

Squeeze Value Node

This node is used primarily in conjunction with the Camera Data node used. The camera data generate large output values, both in terms of the depth information as well as the extent in the width. With the squeeze Node high output values to an acceptable material for the node degree, i.e. to values between 0.0 - 1.0 scaled down.



Inputs

Value Any numeric value. The value can be provided by another node or set manually.

Width Determines the curve between sharp S-shaped at a width of 1 and stretched at a width of 0.1. Negative values reverse the course. The value can be provided by another node or set manually.

Center The center of the output value range. This input value is replaced by the output value of 0.5. The value can be provided by another node or set manually.

Properties

This node has no properties.

Outputs

Value A value in the range between 0 and 1.

Vector Math Node

This node performs vector math operation.

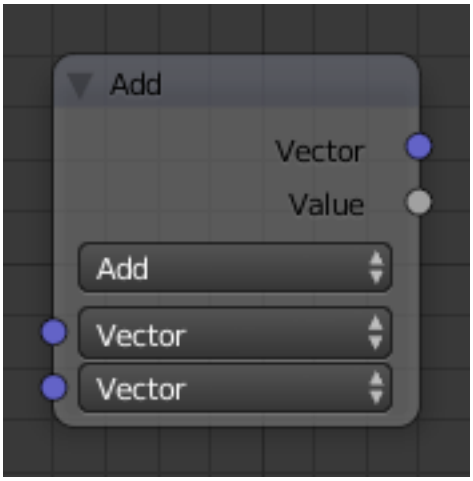


Fig. 2.1540: Vector Math node.

Inputs

Vector First vector input.

Vector Second vector input.

Properties

Operation Selector of the math function.

Add Adding input 1 and 2.

Subtract Subtracting input 1 and 2.

Average Averaging input 1 and 2.

Dot Product Algebraic operation that takes two equal-length sequences of input vectors. The result is a scalar value.

Cross Product Geometric binary operation on the two vectors inputs in three-dimensional space. It results in a vector which is perpendicular to both and therefore normal to the plane containing them.

Normalize Normalizing input 1 and 2.

Outputs

Vector Standard vector output.

Value Standard value output.

Input Nodes

Camera Data Node

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

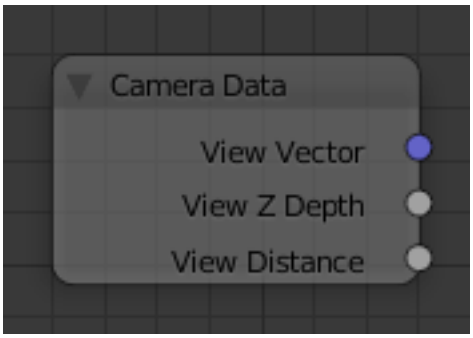


Fig. 2.1541: Camera Data node.

View Vector A Camera space vector from the camera to the shading point.

View Z Depth How far away each pixel is from the camera

View Distance Distance from the camera to the shading point.

Extended Material Node

Adds additional input and output sockets to the *Material Node*. Only the additional sockets are listed.

Inputs

Mirror Color of mirrored reflection.

Ambient Amount of global ambient color the material receives.

Emit The emissivity, which is the amount of light to emit.

Specular Transparency Is the alpha for the specular color.

Reflectivity The degree to which the material reflects light.

Alpha Transparency of the material by setting all pixels in the alpha channel to the given value.

Translucency Amount of diffuse shading on the back side.

Properties

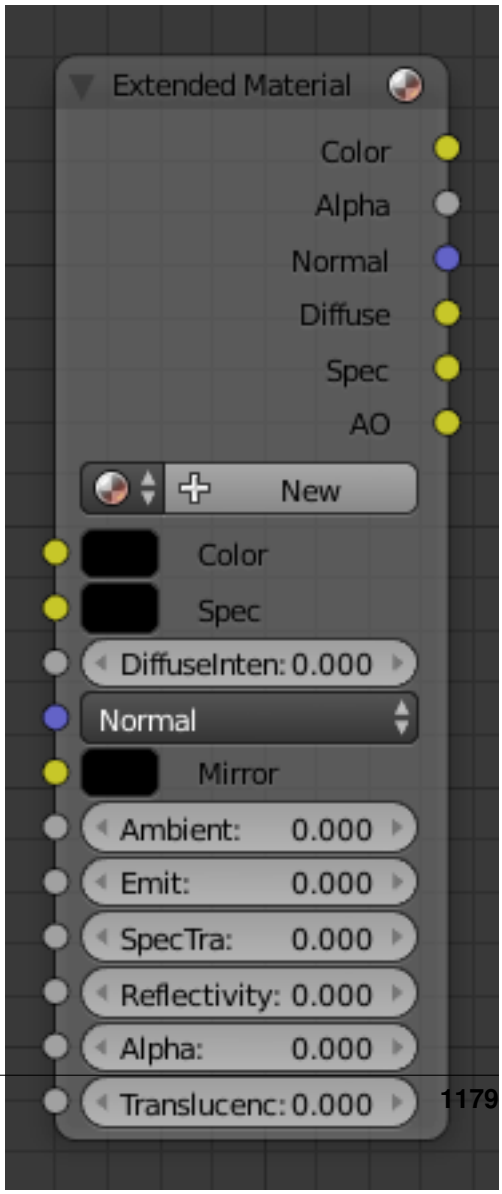
This node has no additional properties.

Outputs

Diffuse Value of the diffuse color.

Specular Value of the specular color.

AO Value of the Ambient Occlusion.



Geometry Node

The geometry node is used to specify how light reflects off the surface. This node is used to change a material's Normal response to lighting conditions.

Use this node to feed the Normal vector input on the Material node, to see how the material will look (i.e. shine, or reflect light) under different lighting conditions. Your choices are:

Inputs

This node has no inputs.

Properties

UV layer To select a listed UV map.

Color layer To select a listed vertex color data (Vertex Paint, Weight Paint).

Outputs

Global Global position of the surface.

Local Local position of the surface.

View Viewed position of the surface.

Orco Using the Original Coordinates of the mesh.

UV Using the UV coordinates of the mesh, selected in the field in bottom node.

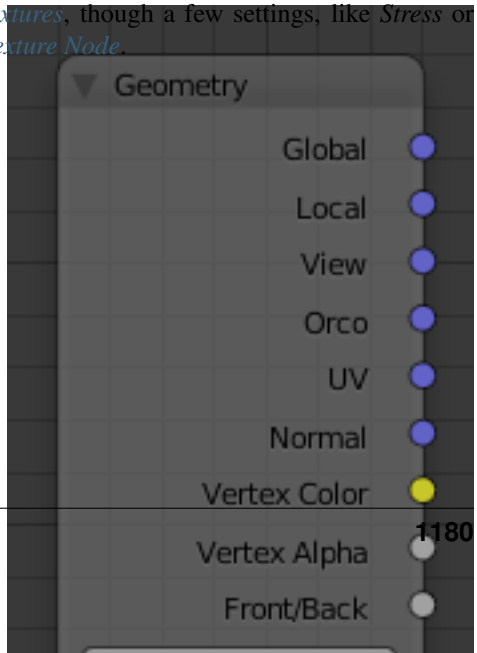
Normal Surface Normal; On a flat plane with one light above and to the right reflecting off the surface.

Vertex Color Allows for output value of group vertex colors, selected in the field in bottom node.

Vertex Alpha Allows for output alpha value of vertex.

Front/Back Allows for output to take into account front or back of surface is light relative the camera.

Note: These are exactly the same settings as in the *Mapping* panel for *Textures*, though a few settings, like *Stress* or *Tangent*, are missing here. Normally you would use this node as input for a *Texture Node*.



Geometry Node Example using a UV image

E.g.: To render a UV-mapped image, you would use the *UV* output and plug it into the *Vector* Input of a texture node. Then you plug the color output of the texture node into the color input of the material node, which corresponds to the setting on the *Map To* panel.

Lamp Data Node

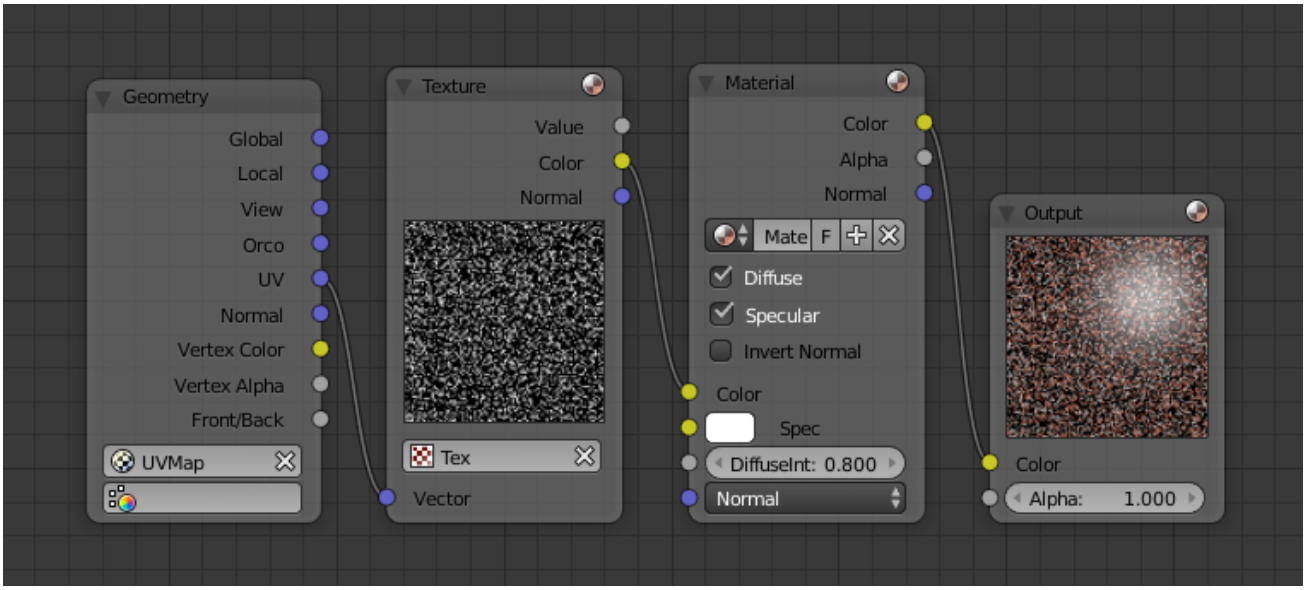


Fig. 2.1544: Setup to render a UV-Mapped Image Texture.

The Lamp Data node is used to obtain information related to a specified lamp object.

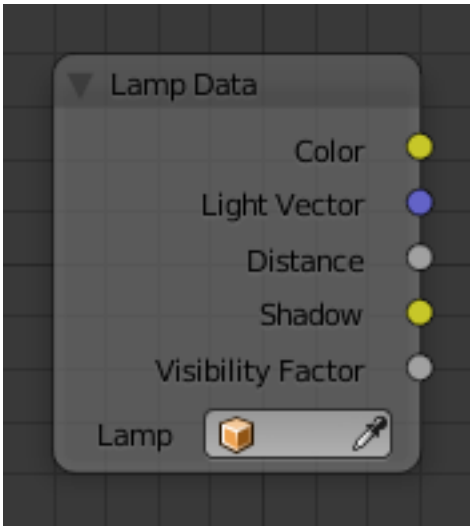


Fig. 2.1545: Lamp Data node.

Inputs

This node has no inputs.

Properties

Lamp field To select a listed lamp object.

Outputs

The light textures and the shadow textures affect the Color and Shadow outputs, respectively.

Color Lamp color multiplied by the lamp energy.

Light Vector A unit vector in the direction from the lamp to the shading point.

Distance Distance from the shading point to the lamp.

Shadow Shadow color that the other objects cast on the shading point.

Visibility Factor Light falloff ratio at the shading point.

Note: Portability to Various Scenes

If multiple materials use a Lamp Data node linking to the same lamp object, including the Lamp Data node into a node group is recommended. Otherwise, when the mesh objects are imported to the other scene, all the materials may need to be modified.

Material Node

The Material node is used for shading.

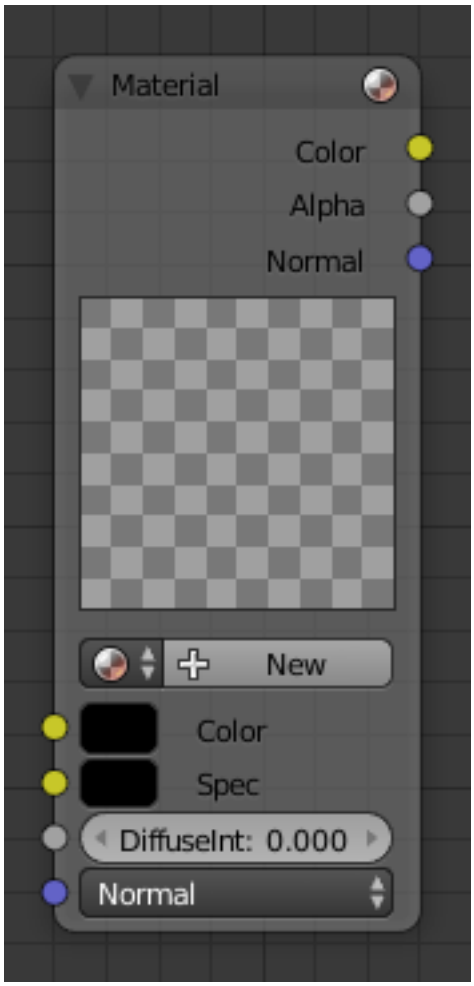


Fig. 2.1546: Material node.

Inputs

Color The diffuse color or texture.

Specular The specular color or texture that is reflected as the point of view get perpendicular to the light source reflecting off the surface.

Diffuse Intensity Amount of diffuse scattering. Can also be set by a texture.

Normal Standard normal input. Used for normal mapping.

Note: Normal Override

The normal input socket does not blend the source normal with the underlying geometry but completely overrides the normals.

Properties

Material field Can be used to browse and select materials.

Diffuse De/activate diffuse shading.

Specular De/activate specular shading.

Invert Normal Inverts the material input normal when activated. (which, is a combination of the 3D normal given to it by the 3D object plus the normal input point).

Outputs

Color Shaded color value.

Alpha Alpha transparency value.

Normal Direction of the normal.

Examples

Using the Material Node with Specularity

To make a material node actually generate a color, at least a basic input color have to be specified, and optionally a specularity color. The specularity color is the color that shines under intense light.

For example, consider the mini-map to the right. The base color, a dark blue, is connected from an RGB color generator node to the *Color* input socket. The specular color, yellow, is connected to the *Spec* input. Under *Normal* lighting conditions on a flat surface, this material will produce a deep blue color and, as the point of view approaches a spot perpendicular to the light, the yellow specular color mix in becomes visible.

Note: Enable Spec

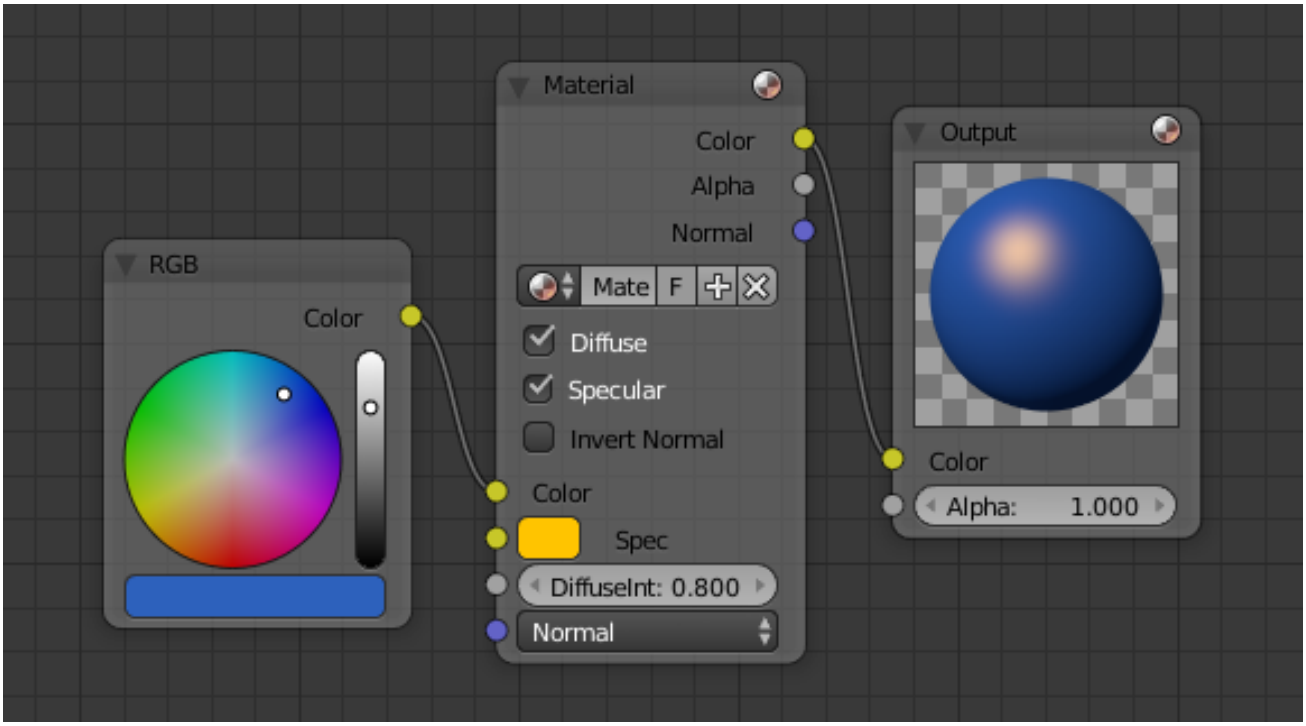


Fig. 2.1547: Material Node using Specularity.

To see specularity, the specular toggle has to be enabled, which is located just below the material color button in the node.

Particle Info Node

The *Particle Info* node is for objects instanced from a *Particle System*, this node give access to the data of the particle that spawned the instance.

Note: This node currently only supports parent particles, info from child particles is not available.

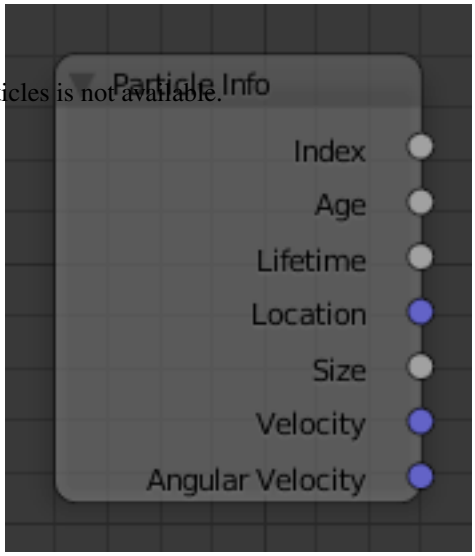


Fig. 2.1548: Particle Info Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Index Index number of the particle (from 0 to number of particles).

Age Age of the particle in frames.

Lifetime Total lifespan of the particle in frames.

Location Location of the particle.

Size Size of the particle.

Velocity Velocity of the particle.

Angular Velocity Angular velocity of the particle.

RGB Node

Inputs

This node has no input sockets.

Properties

The RGB node uses the *color picker widget*.

Outputs

Color / RGBA A single RGBA color value.

Texture Node

Can be used to input image or procedural textures.

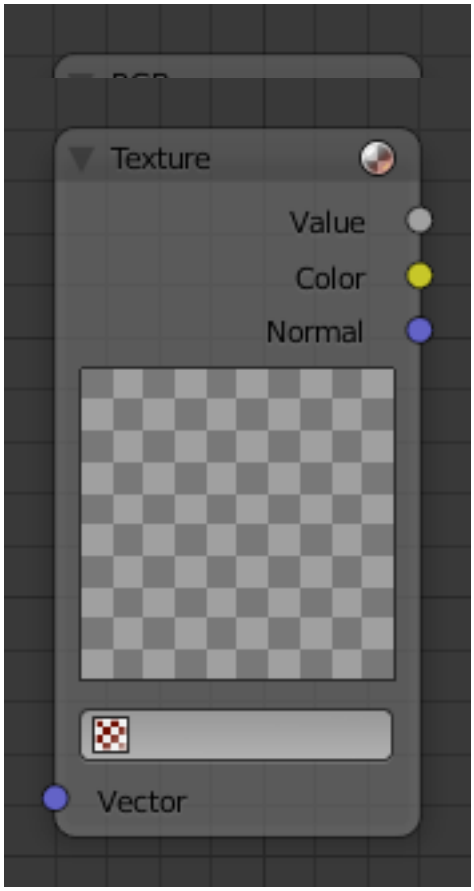


Fig. 2.1550: Texture node.

Inputs

Vector Uses for map the texture to a specific geometric space.

Properties

Texture The texture could be selected from a list of textures available in the current blend-file or link in textures. The textures themselves could not be edited in this note, but in the Texture panel.

Outputs

Value Straight black-and-white value of the texture.

Color Texture color output.

Normal The of normal map.

Example

In the example to the right, a cloud texture, as it would appear to a viewer, is added to a base purple material, giving a velvet effect.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.

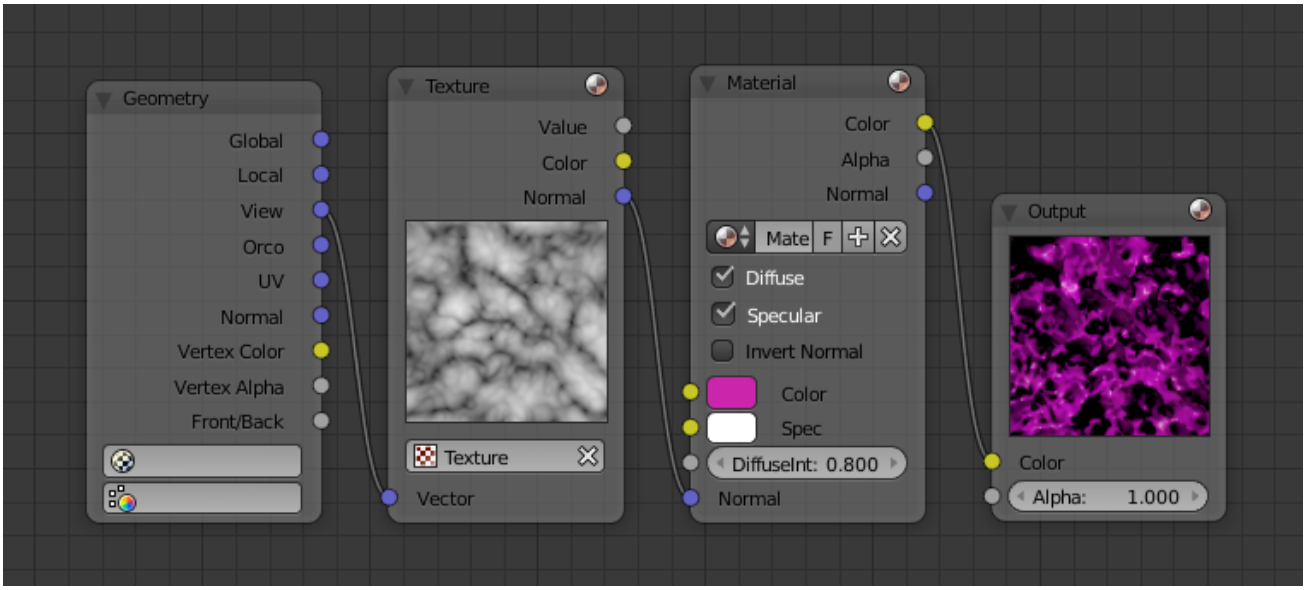


Fig. 2.1551: Example of the applying Texture node.

Inputs

This node has no input sockets.

Properties

Single numerical value (floating point).

Outputs

Value The value set in the options.

Example

In the example below the *Value Node* is used to control multiple values at once, this make the node a useful organizational tool.

Tip: From this you can also make different values proportional to each other by adding a *Math Node* in between the different links.

Fig. 2.1553: Example of the *Value Node*.

Output Node

Fig. 2.1552: Value Node.

The preview could also be used for a non shader color input.

Inputs

Color Color applied to the geometry.

Alpha Alpha transparency.

Properties

This node has no properties.

Outputs

This node has no outputs.

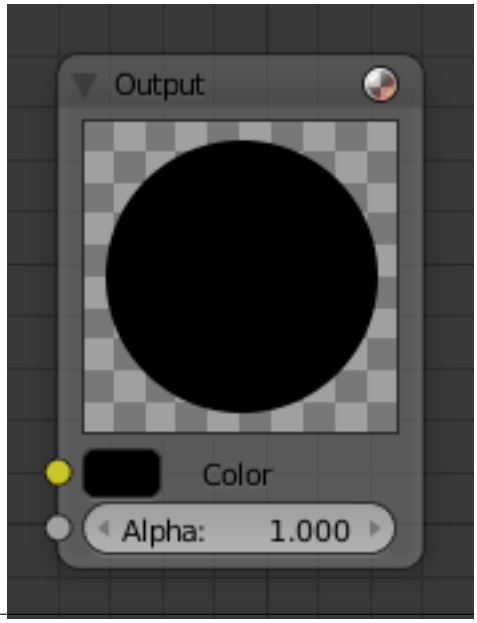


Fig. 2.1554: Output material node.

Note: Effective Output Node

The only Output node that is used for the Material in the end is the last selected, indicated by the darker background.

Vector Nodes

Vector nodes manipulate information about how light interacts with the material, multiplying vector sets, and other wonderful things which can become quite advanced. Even if you do not have experience with vector maths, you will find these nodes to be very useful.

Vectors, in general, are two or three element values, for example, surface normals are vectors. Vectors are also important for calculating shading.

Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.

Inputs

In the shader context the node also has an additional Factor property.

Factor Controls the amount of influence the node exerts on the output vector.

Vector Standard vector input.

Properties

Channel X, Y, Z

Curve For the curve controls see: *Curve widget*.

Outputs

Vector Standard vector output.

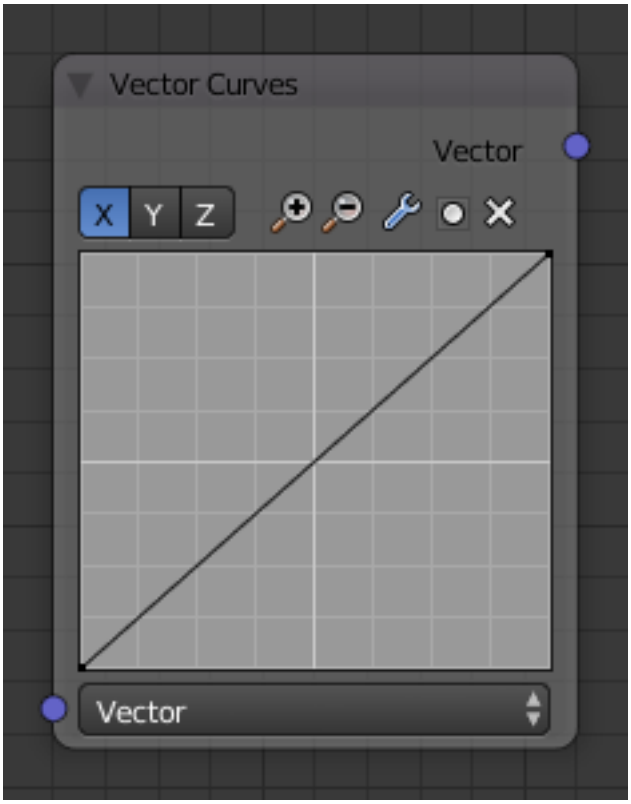


Fig. 2.1555: Vector Curves Node.

Mapping Node

Essentially mapping node allows the user to modify a mapping of system of 3D-coordinates. Mapping can be rotated and clamped if desired.

Typically used for modifying texture coordinates.

Inputs

Vector Standard vector input.

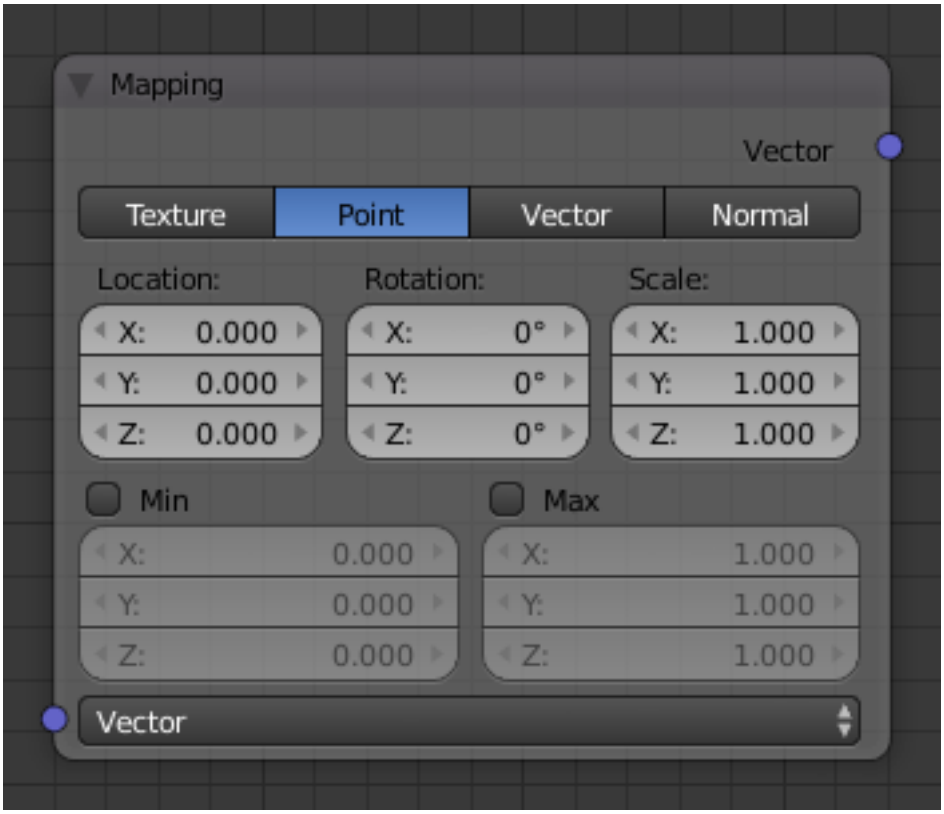


Fig. 2.1556: Mapping node.

Properties

The controls of the node have been ordered in X, Y, Z order.
Clamping can be enabled by Min and Max.

Vector type Type of vector that the mapping transforms.

Texture Transform a texture by inverse mapping the texture coordinates.

Point Transform a point.

Vector Transform a direction vector.

Normal Transform a normal vector with unit length.

Location Transform position vector.

Rotation Transform rotation vector.

Scale Transform scale vector.

Min Minimum clipping value.

Max Maximum clipping value.

Outputs

Vector Standard vector output.

Normal Node

The Normal node generates a normal vector and a dot product.

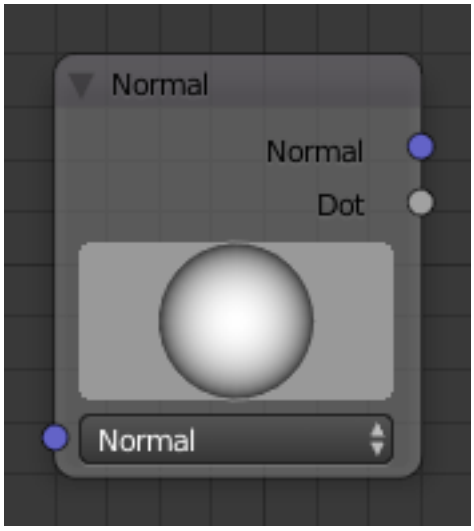


Fig. 2.1557: Normal Node.

Inputs

Normal Normal vector input.

Properties

Normal Direction To manually set a fixed normal direction vector. LMB click and drag on the sphere to set the direction of the normal.

Outputs

Normal Normal vector output.

Dot Dot product output. The dot product is a scalar value.

- If two normals are pointing in the same direction the dot product is 1.
- If they are perpendicular the dot product is zero (0).
- If they are antiparallel (facing directly away from each other) the dot product is -1.

Vector Transform Node

The *Vector Transform* node allows converting a Vector, Point or Normal between World <=> Camera <=> Object coordinate space.

Inputs

Vector Input Standard vector input.

Properties

Type Specifies the input/output type: Vector, Point or Normal.

Convert From Coordinate Space to convert from: World, Object or Camera.

Convert To Coordinate Space to convert to: World, Object or Camera.

Outputs

Vector Output The transformed output vector.

Examples

Todo.

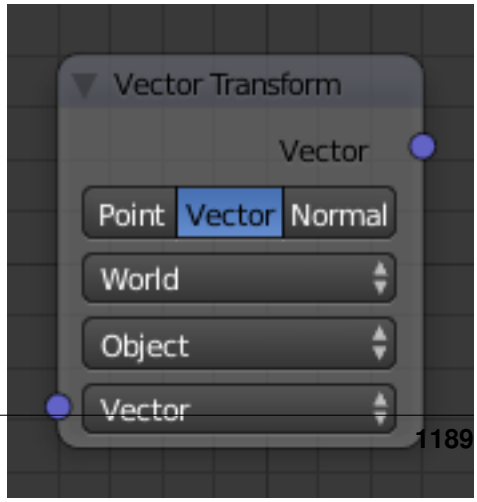


Fig. 2.1558: Vector Transform node.

Special Material Effects

Halo Rendering

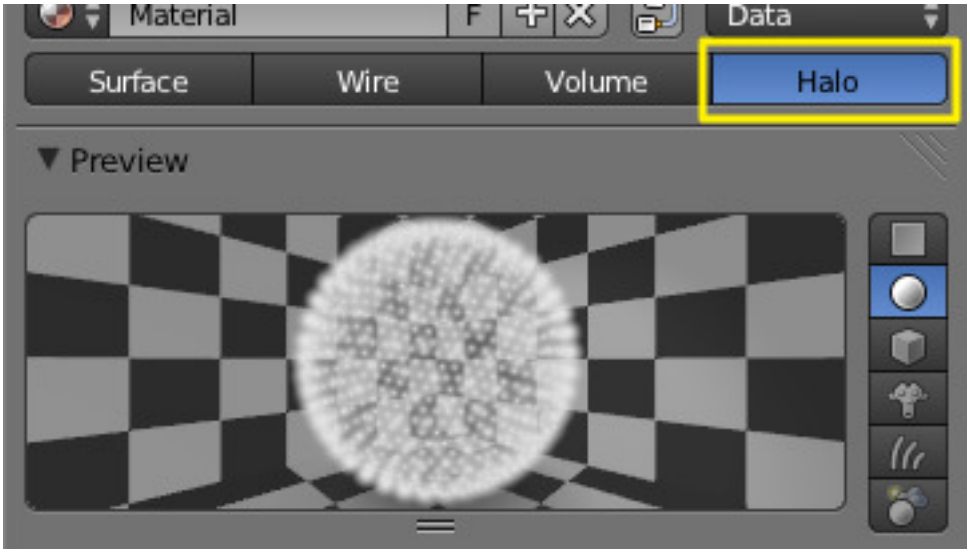


Fig. 2.1559: Activating helo rendering.

Halo materials renders each of the objects points as glowing dots or a little clouds of light. Although they are not really lights because they do not cast light into the scene like a lamp. These are called *Halos* because you can see them, but they do not have any substance.

Halos are rendered with vertex shaders and not with face shaders.

This material is useful for simulating special effects, like particle effects or lens flares.

Options

To enable *Halos*, press the *Halo* button in the *Material* menu's top panel.

As you will see in the 3D View, the mesh faces are no longer rendered. Instead just the vertex is rendered, since that is where each halo will originate. Halos can be hard to find in a crowded scene, so name it well for easy location in *the outliner*.

In the Properties editors, where we normally find the *Diffuse*, *Specular*, and *Shading* panels, we now see panels relative to the *Halo* characteristics:

Halo Panel

Alpha The transparency.

Diffuse Color The color of the halo itself.

Seed If non-zero, randomizes the ring dimension and line location. To use, give any (integer) number to start the random-number generator.

Size Sets the dimension of the halo

Hardness Sets the hardness of the halo. Similar to specular hardness

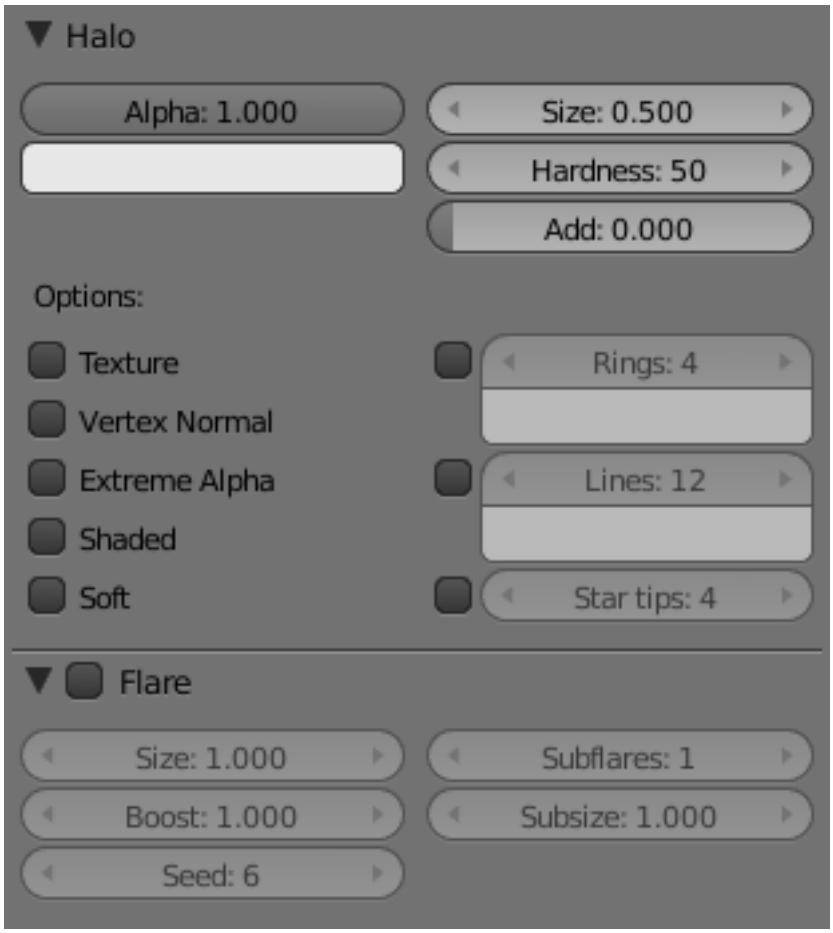


Fig. 2.1560: Halo panels.

Add Determine how much the halo colors are 'added to', rather than mixed with, the colors of the objects behind and together with other halos. By increasing Add, the Halo will appear to light up objects that move behind it or through the Halo field.

Fig. 2.1561: Effect of Add.

Texture Gives halo a texture. By default, textures are applied to objects with Object coordinates and reflects on the halos by affecting their color, as a whole, on the basis of the color of the vertex originating the halo. Enable this feature to have the texture take effect *within* the halo, and hence to have it with varying colors or transparencies; this will map the whole texture to *every* halo. This technique proves very useful when you want to create a realistic rain effect using particle systems, or similar.

Vertex Normal Use the vertex normal to specify the dimension of the halo.

Extreme Alpha Boosts alpha.

Shaded Lets halo receive light and shadows from external objects.

When shaded is enabled, the Halo will be affected by local light; a lamp will make it brighter and affect its diffuse color and intensity.

Soft Softens the edges of the halos at intersections with other geometry.

In addition, several other special effects are available. To enable some or all of these effects, set the number of points/rings, or set the color of each effect individually:

Rings Adds circular rings around to the halo.

Lines Adds lines from the center of the halo.

Star tips Gives the halo a star shape.

You cannot use color ramps. Lines, Rings and an assortment of special effects are available with the relevant toggle buttons, which include Flare, Rings, Lines, Star, Texture, Extreme Alpha, and Shaded. *Halo Variations* shows the result of applying a halo material to a single vertex mesh.

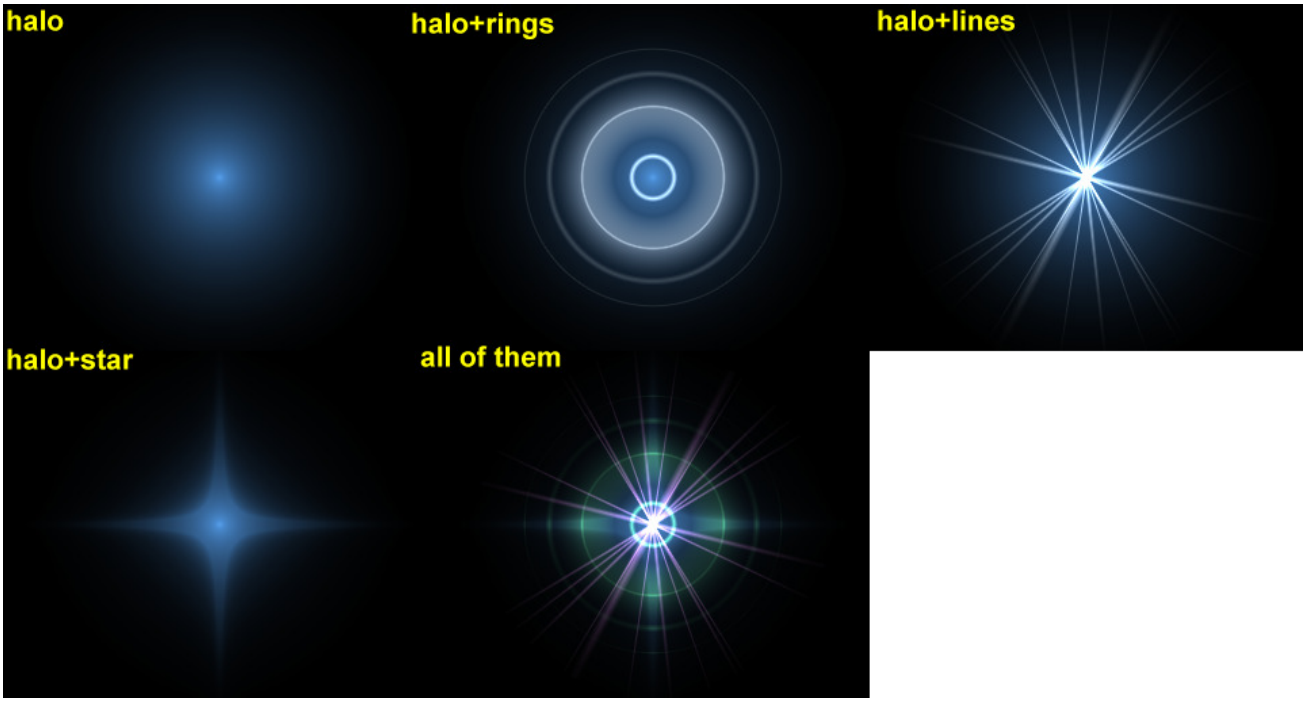


Fig. 2.1562: Halo Variations.

Flare Panel

Enabling Flare Renders the halo as a lens flare.

Size Sets the factor by which the flare is larger than the halo.

Boost Give the flare extra strength.

Seed Specifies an offset in the flare seed table.

Subflares Sets the number of subflares.

Subsize Sets the dimensions of the subflares, dots, and circles.

Lens Flares

Our eyes have been trained to believe that an image is real if it shows artifacts that result from the mechanical process of photography. *Motion blur*, *Depth of Field*, and *lens flares* are just three examples of these artifacts. The first two are discussed in the *chapter rendering*; the latter can be produced with special halos. A simulated lens flare tells the viewer that the image was created with a camera, which makes the viewer think that it is authentic.

We create lens flares in Blender from a mesh object using first the *Halo* button and then the *Flare* options in the *Shaders* Panel of the material settings. Try turning on *Rings* and *Lines*, but keep the colors for these settings fairly subtle. Play with the *Flares:* number and *Fl. seed:* settings until you arrive at something that is pleasing to the eye. You might need to play with *Boost:* for a stronger effect Fig. *Lens Flare.* settings.

Note that this tool does not simulate the physics of photons traveling through a glass lens; it's just an eye candy.

Blender's lens flare looks nice in motion, and disappears when another object occludes the flare mesh.

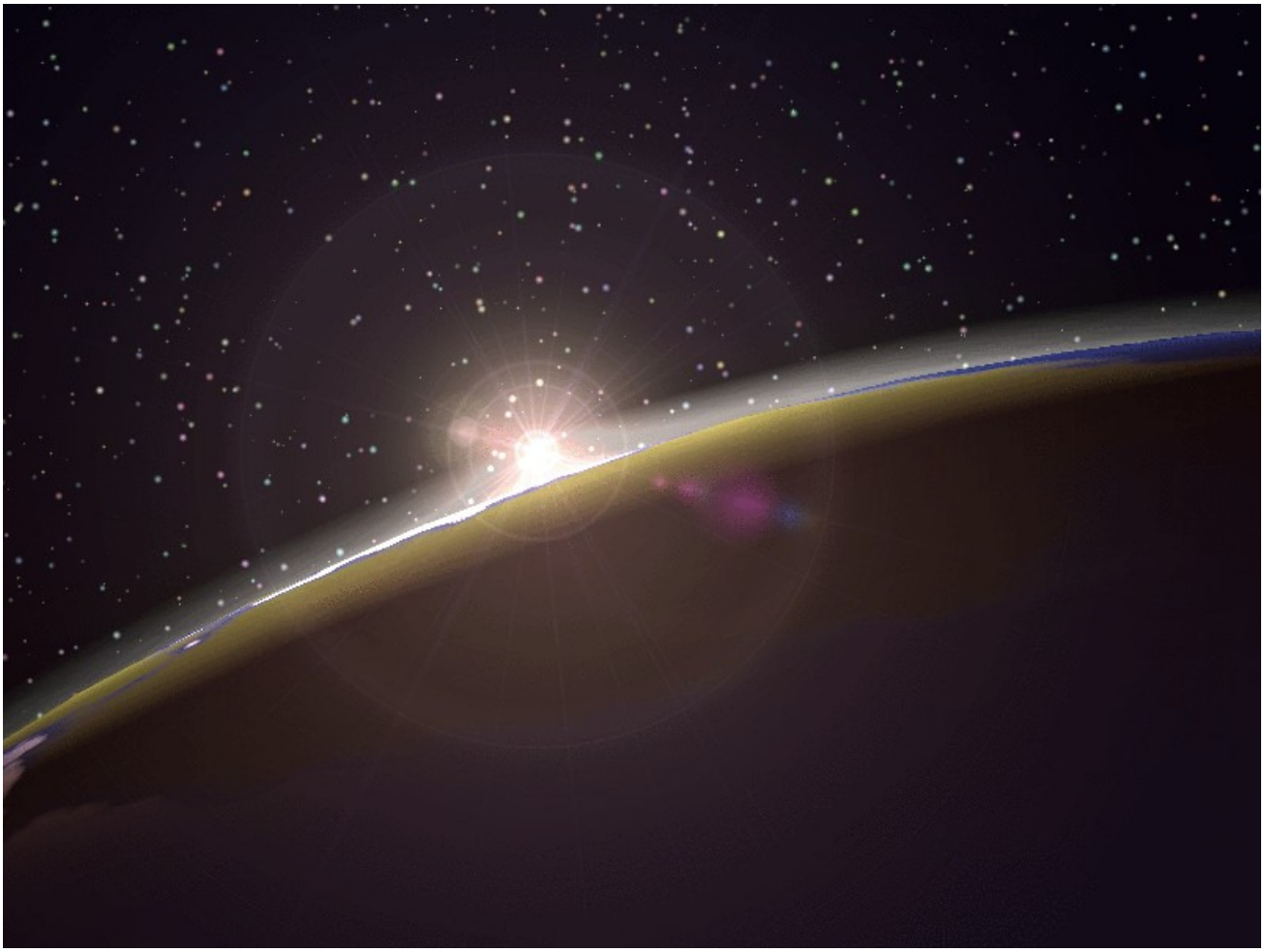


Fig. 2.1563: Lens Flare.

Halo Texturing

By default, textures are applied to objects with Object coordinates and reflects on the halos by affecting their color, as a whole, on the basis of the color of the vertex originating the halo. To have the texture take effect *within* the halo, and hence to have it with varying colors or transparencies press the *Texture* button; this will map the whole texture to *every* halo. This technique proves very useful when you want to create a realistic rain effect using particle systems, or similar.

Another Option is Shaded. When shaded is enabled, the Halo will be affect by local light; a lamp will make it brighter and affect its diffuse color and intensity.

Examples

Dotmatrix Display

Let us use a halo material to create a dotmatrix display:

1. To begin, add a grid with the dimensions 32×16. Then add a camera and adjust your scene so that you have a nice view of the billboard.
2. Use a 2D image program to create some red text on a black background, using a simple and bold font, you can just save the picture below on your hard drive...). *Dot matrix image texture.* shows an image 512 pixels wide by 64 pixels high, with some black space at both sides.

Fig. 2.1564: Dot matrix image texture.

- Add a material for the billboard, and set it to the type *Halo*. Set the *Halo Size* to 0.06 and when you render the scene you should see a grid of white spots.
- Add a Texture, then change to the Texture Buttons and make it an image texture. When you load your picture and render again you should see some red tinted dots in the grid.
- Return to the Material Buttons and adjust the *size X* parameter to about 0.5 then render again; the text should now be centered on the Billboard.
- To remove the white dots, adjust the material color to a dark red and render. You should now have only red dots, but the billboard is still too dark. To fix this enter *Edit Mode* for the board and copy all vertices using the `Shift-D` shortcut (take care not to move them!). Then adjust the brightness with the *Add* value in the Halo panel.

  You can now animate the texture to move over the billboard, using the *Offset X* value in the *Texture* tab of the Mapping panel. (You could use a higher resolution for the grid, but if you do you will have to adjust the size of the halos by shrinking them, or they will overlap. Fig. *Dot Matrix display.*.

Note: Note about material indices

Fig. 2.1565: Dot Matrix display.

Halo materials only work when applied using the first material index. Any material(s) in a subsequent material index will not be rendered.

Volume Rendering

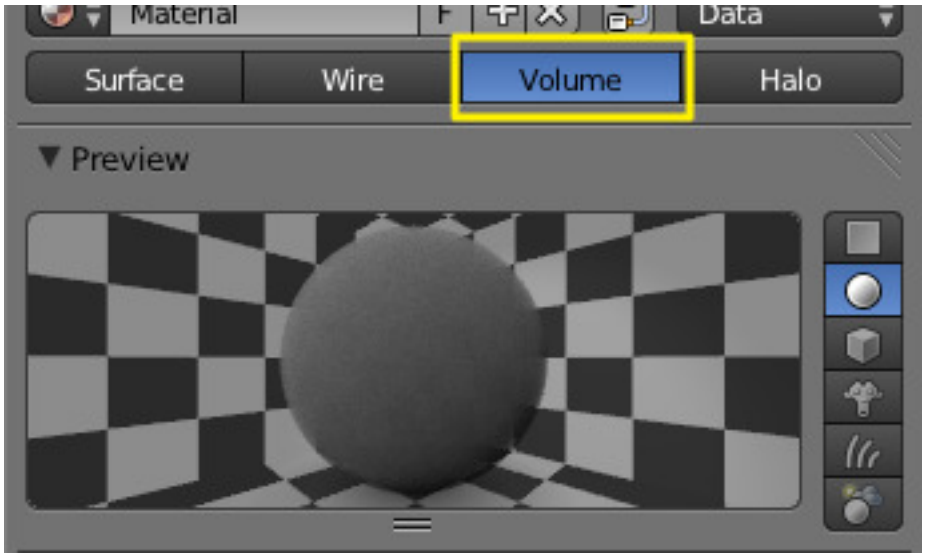


Fig. 2.1566: Activation volume rendering.

Volume rendering is a method for rendering light as it passes through participating media, within a 3D region. The implementation in Blender a physically based model, which represents the various interactions of light in a volume relatively realistically.

Rendering a volume is different then *Solid Render*. For volume light en-

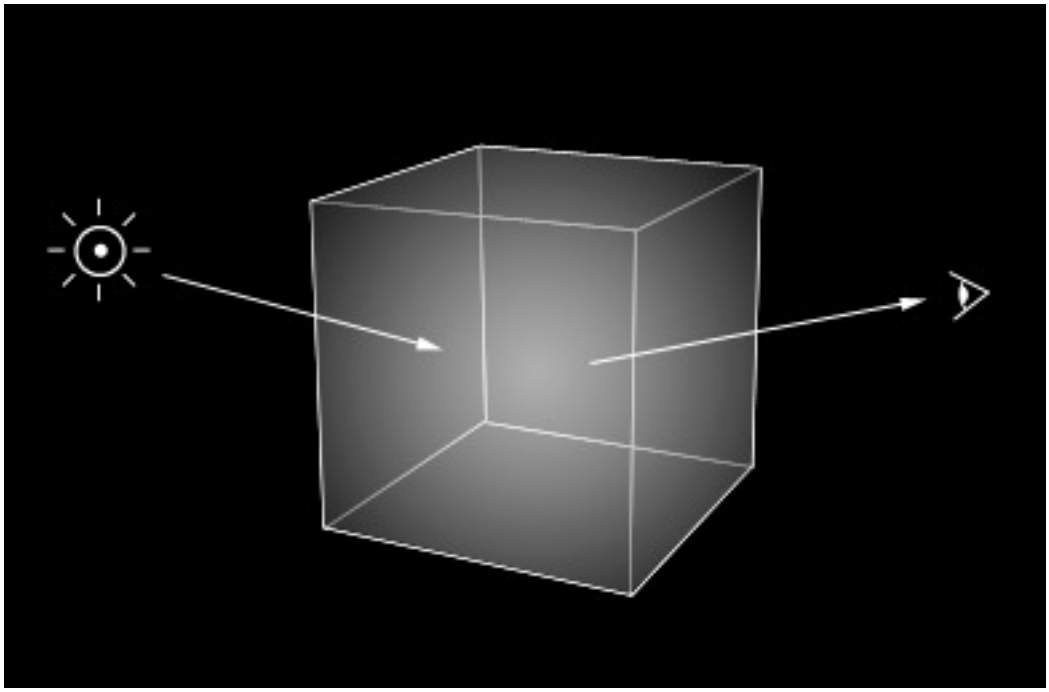


Fig. 2.1567: Volume rendering.

ters a 3D region of space (defined as the volume) that may be filled with small particles, such as smoke, mist or clouds. The light bounces around off the various molecules, being scattered or absorbed, until some light passes through the volume and reaches the camera. In order for that volume to be visible, the renderer must figure out how much material the light has passed through and how it has acted and reacted within that volume, the volume object needs to contain a 3D region of space, for example a *manifold* closed mesh, such as a cube, not just a flat surface like a plane. To get an image, the renderer has to step through that region, and see how much 'stuff' is there (density) in order to see how light is absorbed or scattered or whatever. This can be a time consuming process since it has to check a lot of points in space and evaluate the density at each.

Options

Density

Many things can happen to the light as it passes through the volume, which will influence the final color that arrives at the camera. These represent physical interactions that happen in the real world, and most of these are dependent on the density of the volume, which can either be a constant density throughout, or varied, controlled by a texture. It is by controlling the density that one can get the typical 'volumetric' effects such as clouds or thick smoke.

Density The base density of the material. Other densities from textures are added on top.

Density Scale A global multiplier to increase or decrease the apparent density. This can be useful for getting consistent results across different scene scales.

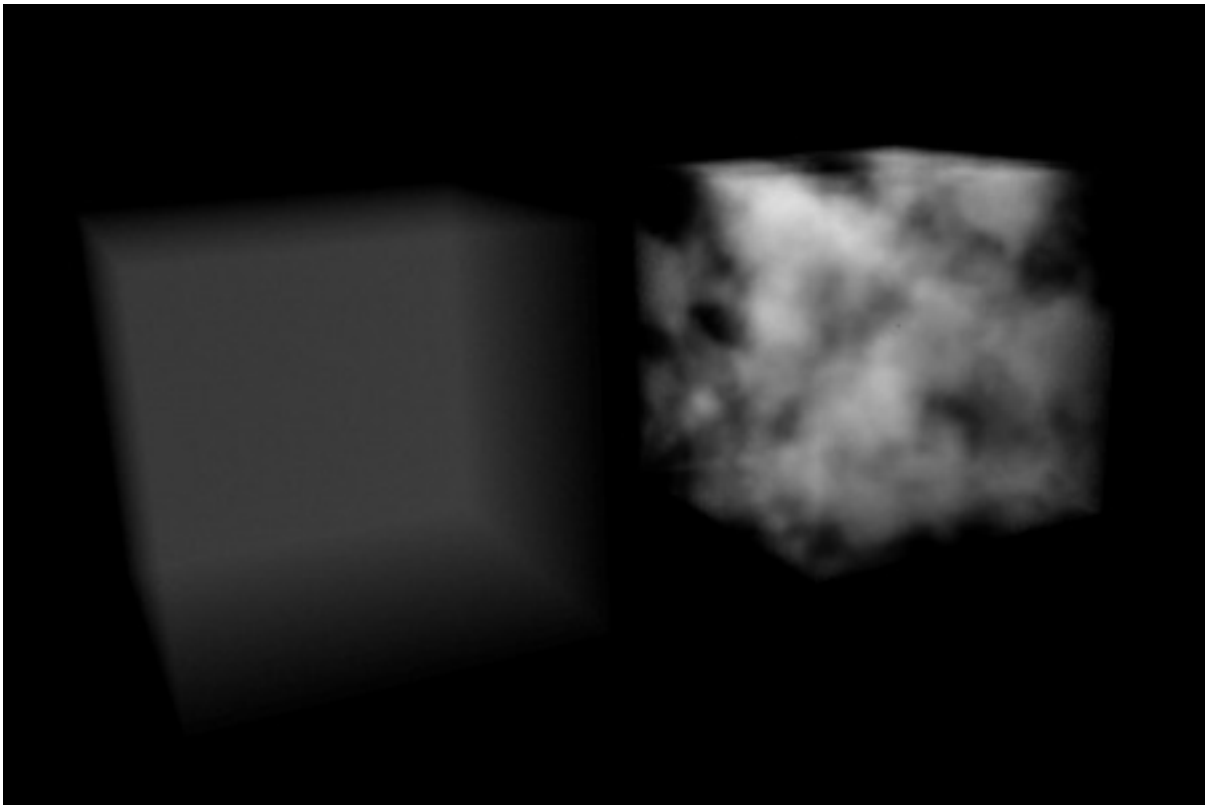


Fig. 2.1568: Constant density vs textured density.

Shading

When light enters a volume from an external source, it does not just pass straight through. Light gets scattered off tiny particles in the volume, and some proportion of that light reaches the camera. This property makes it possible to see light beams as they travel though a volume and are scattered towards the eye.

Scattering The amount of light that is scattered out of the volume. The more light that is scattered out of the volume, the less it will penetrate through the rest of the volume. Raising this parameter can have the effect of making the volume seem denser, as the light is scattered out quickly at the 'surface' of the volume, leaving the areas internal to the volume darker, as the light does not reach it.

Note in the examples below, the less light that is scattered out of the volume, the more easily it penetrates throughout the volume and to the shadow.

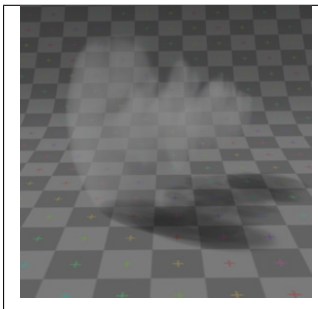


Fig. 2.1571: Scattering: 0.5.

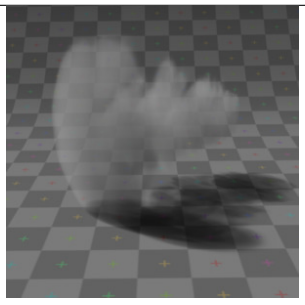


Fig. 2.1572: Scattering: 1.0.

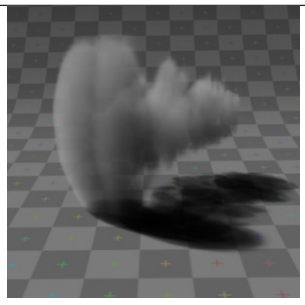


Fig. 2.1573: Scattering: 2.0.

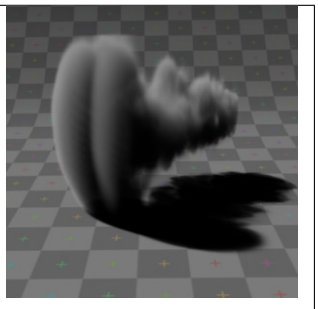


Fig. 2.1574: Scattering: 5.0.

Fig. 2.1569: Spot lamp scattering in a constant volume.

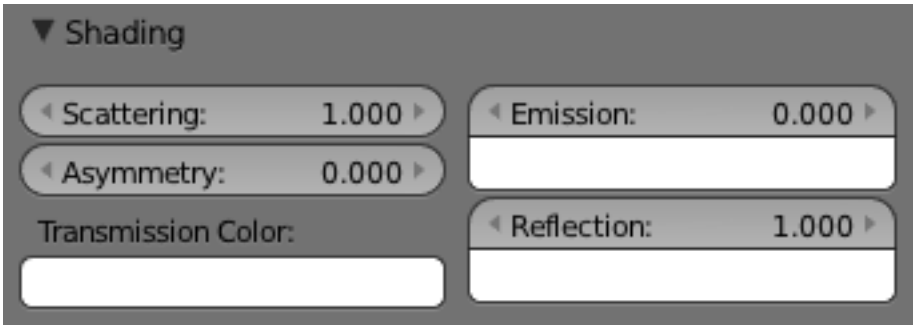


Fig. 2.1570: Shading options.

Asymmetry

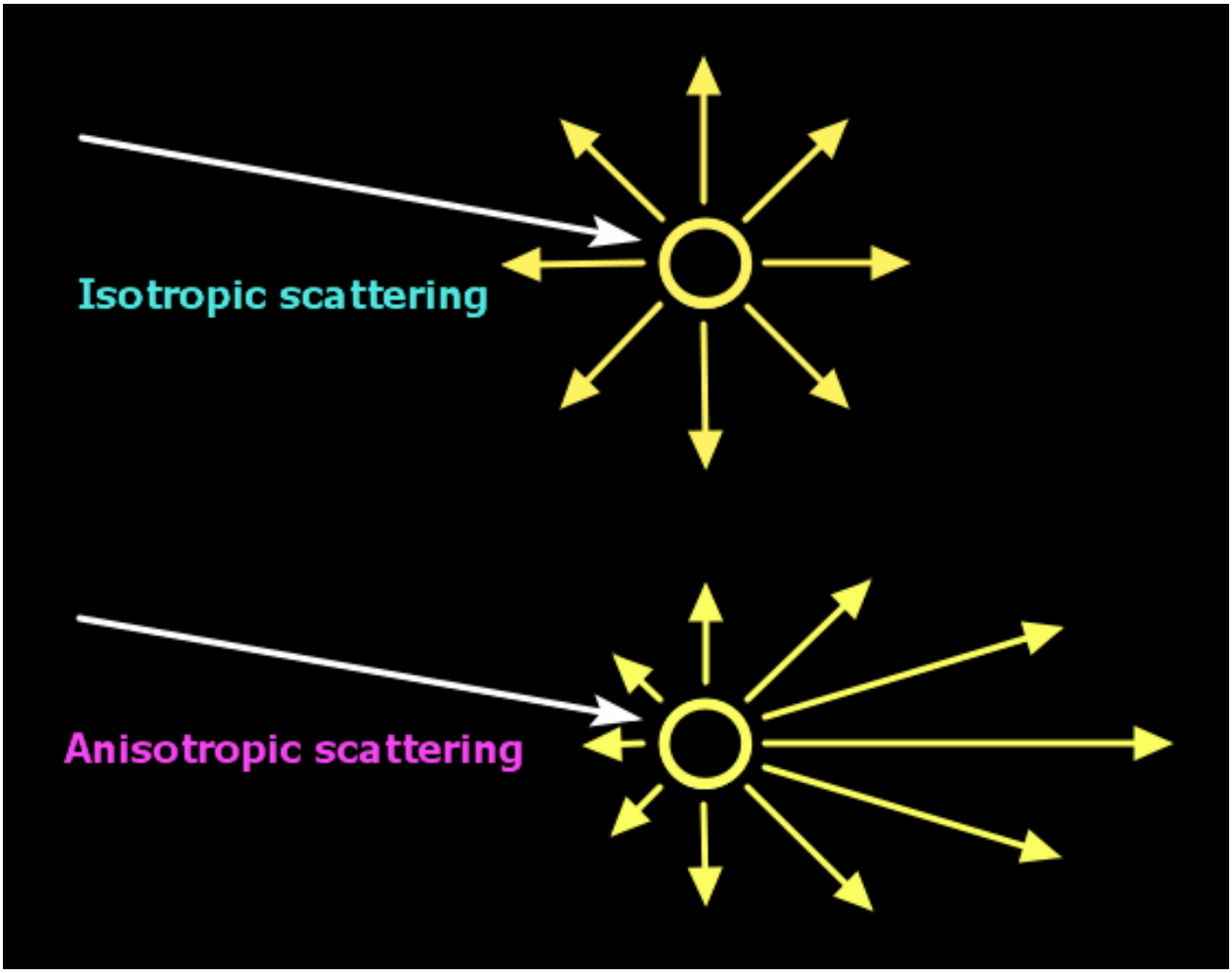


Fig. 2.1575: Isotropic and Anisotropic scattering.

The default method for scattering light in a volume is for the light to be deflected evenly in all directions, also known as Isotropic scattering. In real life different types of media can scatter light in different angular directions, known as Anisotropic scattering. Back-scattering means that light is scattered more towards the incoming light direction, and forward-scattering means it is scattered along the same direction as the light is traveling.

Asymmetry Asymmetry controls the range between back-scattering (-1.0) and forward-scattering (1.0). The default value of 0.0 gives Isotropic scattering (even in all directions).

Transmission

Transmission is a general term for light that is transmitted throughout a volume.

This transmitted light can be the result of various different interactions, for example:

- the left over result of incoming light after it has reflected/scattered out of the volume

- the left over result of light after being absorbed by the volume (and converted to heat)

  Here, the transmission color is used to set the end result color that light becomes after it is transmitted through the volume.

Transmission Color The resultant color of light that is transmitted through the volume.

Note in the examples below, as more light is scattered out of the volume, there is less available to be transmitted through.

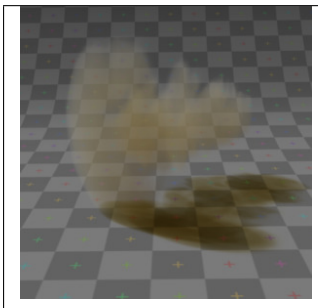


Fig. 2.1576: Transmission color: Yellow, Scattering: 0.5.

Fig. 2.1577: Transmission color: Yellow, Scattering: 1.0.

Fig. 2.1578: Transmission color: Yellow, Scattering: 2.0.

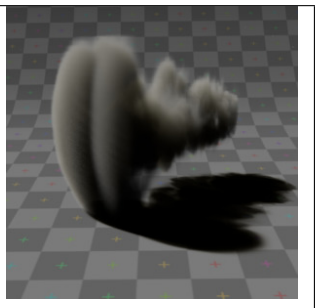


Fig. 2.1579: Transmission color: Yellow, Scattering: 5.0.

Emission

Some volumes can emit light where there was none before, via chemical or thermal processes, such as fire. This light is generated from the volume itself and is independent of light coming from external sources.

Currently, this emitted light does not affect other volumes or surfaces (similar to surface material type, 'Emit' option).

Emission Color The color of light that is emitted by the volume.

Emission An intensity multiplier for the emitted color, for scaling up and down.

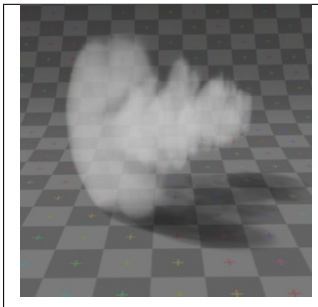


Fig. 2.1580: Emission 0.25, Scattering: 0.5.

Fig. 2.1581: Emission 0.25, Scattering: 1.0.

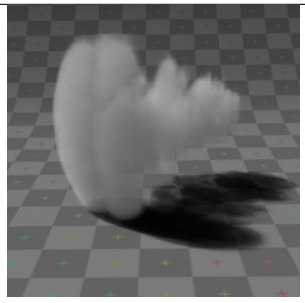


Fig. 2.1582: Emission 0.25, Scattering: 2.0.

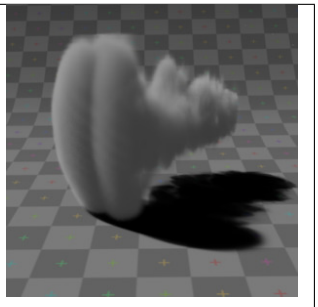


Fig. 2.1583: Emission 0.25, Scattering: 5.0.

Reflection

The *Reflection* parameters can be used to tint or scale the light that is scattered out of the volume. This only affects light that has come from lamps and been scattered out, it does not affect the color of transmitted or emitted light and is.

These settings are not physically correct, because they do not conserve energy. This means the light scattering out does not affect the remaining light, that is transmitted throughout the rest of the volume.

For example, physically speaking, if the orange components of the light are scattered out of the volume towards the camera, only the inverse of that (blue) will remain to continue penetrating through the volume, causing the volume to take on a multi-colored appearance, which can be difficult to use. To make it a bit easier to plainly set the color of the volume, you can use the reflection parameters to quickly set an overall tint.

Reflection Color The color of light that is scattered out of the volume.

Reflection An intensity multiplier for the reflection, for scaling up and down.

Hints

Ideally try to accomplish as much as you can with the other volume settings and lighting before using the reflection controls. If you stick to what is physically plausible, the material will act correctly, and be more predictable and usable in a wider range of lighting scenarios. Of course you can always break the rules too!

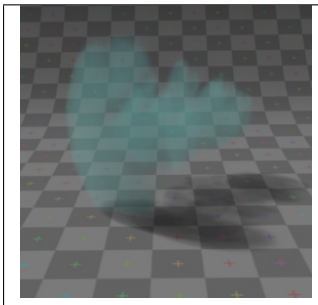


Fig. 2.1584: Reflection: Green, Scattering: 0.5.

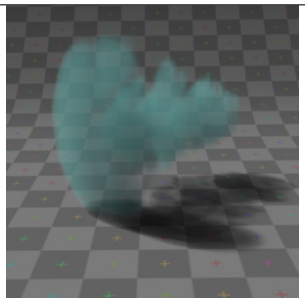


Fig. 2.1585: Reflection: Green, Scattering: 1.0.

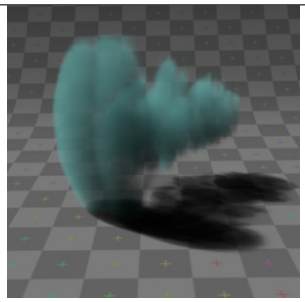


Fig. 2.1586: Reflection: Green, Scattering: 2.0.

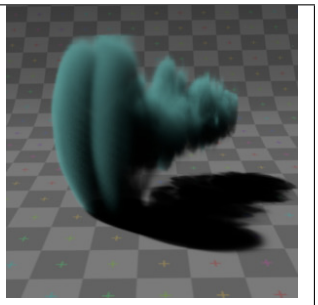


Fig. 2.1587: Reflection: Green, Scattering: 5.0.

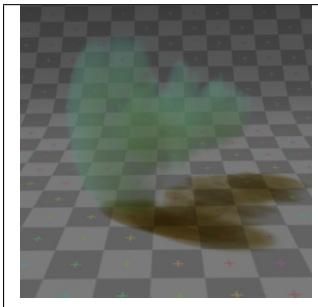


Fig. 2.1588: Reflection: Green, Transmission: Yellow, Scattering: 0.5.

Fig. 2.1589: Reflection: Green, Transmission: Yellow, Scattering: 1.0.

Fig. 2.1590: Reflection: Green, Transmission: Yellow, Scattering: 2.0.

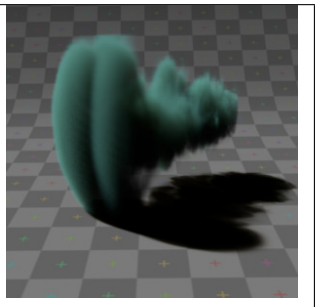


Fig. 2.1591: Reflection: Green, Transmission: Yellow, Scattering: 5.0.

Lighting

Several shading modes are available, providing a range of options between fast to render and physically accurate.

Lighting Mode

Shadeless Shadeless is the simplest, useful for thin, wispy mist or steam.

Shadowed Shadowed is similar, but with shadows of external objects.

Fig. 2.1592: Lighting options.

Shaded Shaded uses a volumetric single-scattering method, for self-shading the volume as light penetrates through.

Multiple Scattering Allows multiple scatter calculations.

Shaded + Multiple Scattering Combines Shaded and Multiple Scattering functionality.

Shaded Options

External Shadows Receive shadows from sources outside the volume (temporary).

Light Cache Pre-calculate the shading information into a voxel grid, speeds up shading at slightly less accuracy.

Resolution Resolution of the voxel grid, low resolutions are faster, high resolutions use more memory.

Multiple Scattering Options

Diffusion Diffusion factor, the strength of the blurring effect.

Spread Proportional distance over which the light is diffused.

Intensity Multiplier for multiple scattered light energy.

Transparency

The transparency settings are the same as *Solid Render* except you have less settings. For volume rendering you only have:

- Mask
- Z Transparency
- Raytrace

Integration

Step Calculation Method Method of calculating the step through the volume.

Randomized Randomized method of calculating the step.

Constant Constant method of calculating the step.

Step Size Distance between subsequent volume depth samples. Step Sizes determine how noisy the volume is. Higher values result in lower render times and higher noise.

Depth Cutoff Stop ray marching early if transmission drops below this luminance threshold. Higher values will give a speedups in dense volumes at the expense of accuracy.

Fig. 2.1593: Integration options.

Options

Fig. 2.1594: Material volume options.

Traceable Allow this material to calculate raytracing.

Full Oversample Force this material to render full shading/textures for all anti-aliasing samples.

Use Mist Use mist with this material (in world settings).

Light Group Limit lighting of this material to lamps in this group.

Exclusive Material uses this group exclusively. Lamps are excluded from other scene lighting.

Smoke and Fire

Create the Material

The material must be a volumetric material with a Density of 0, and a high Density Scale.

Smoke requires a complex material to render correctly. Select the big cube and go to the material tab. There change the material to 'Volume' and set the density to 0. If you set the density to values bigger than 0 the domain cube will be filled with the volume material. The other settings will affect the smoke, though. We'll cover those later.

Add the Texture

In addition, Smoke requires its own texture, you can use a volumetric texture known as *Voxel Data*. You must remember to set the domain object and change the influence.

Go to the texture tab and change the type to *Voxel Data*. Under the Voxel Data-Settings set the domain object to our domain cube (it should be listed

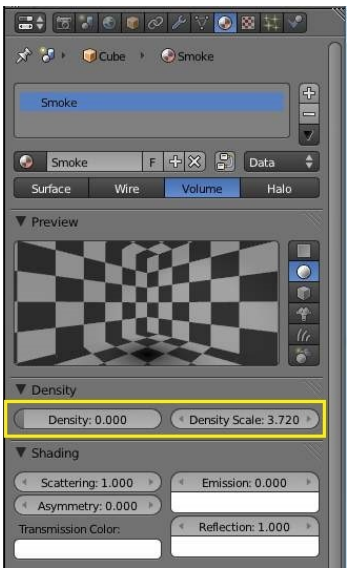


Fig. 2.1595: The Material Settings.

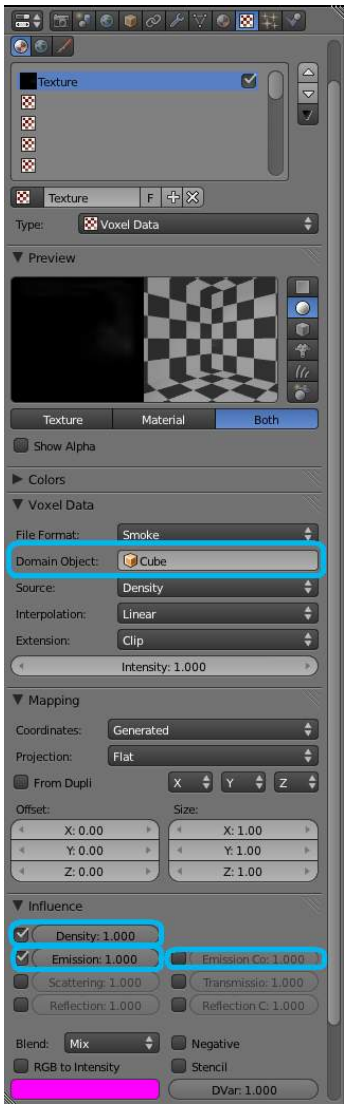


Fig. 2.1596: The texture settings.

just as 'Cube' since we are using Blender's default cube. Under Influence check 'Density' and leave it at 1.000 (Emission should be automatically checked, too). Now you should be able to render single frames. You can choose to color your smoke as well, by turning *Emission Color* back on.

Fig. 2.1597: Finished Result.

Tip: To see the smoke more clearly

Under the world tab, chose a very dark color for the horizon.

Smoke Simulator with fire texture

You can also turn your smoke into fire with another texture! To make fire, turn up the Emission Value in the Materials panel.

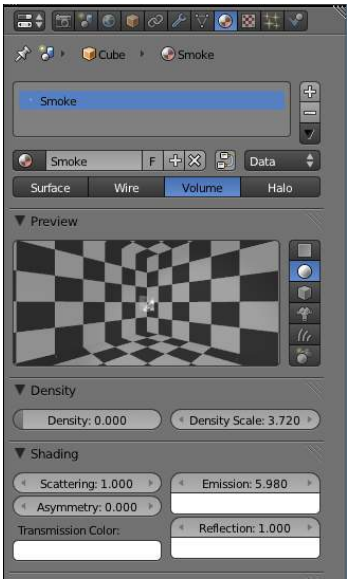


Fig. 2.1598: The Fire material.

Then, add another texture (Keep the old texture or the smoke will not show). Give it a fiery color ramp- which colors based on the alpha, and change the influence to emission and emission color. Change the blend to Multiply.

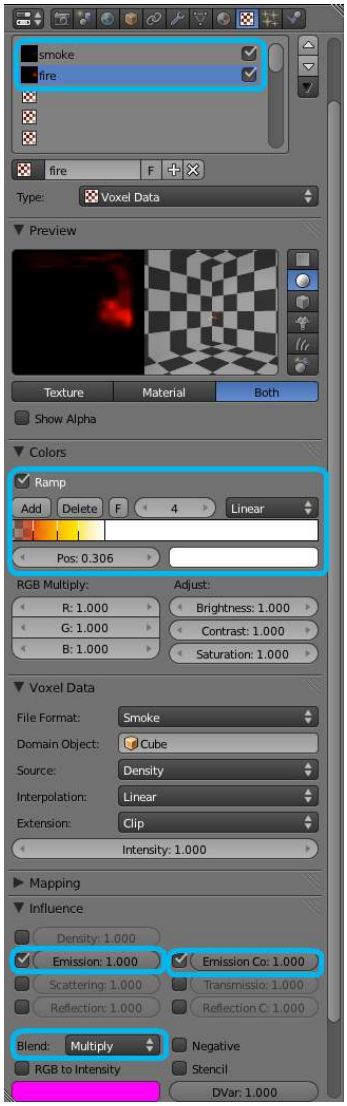


Fig. 2.1599: The fire texture settings.

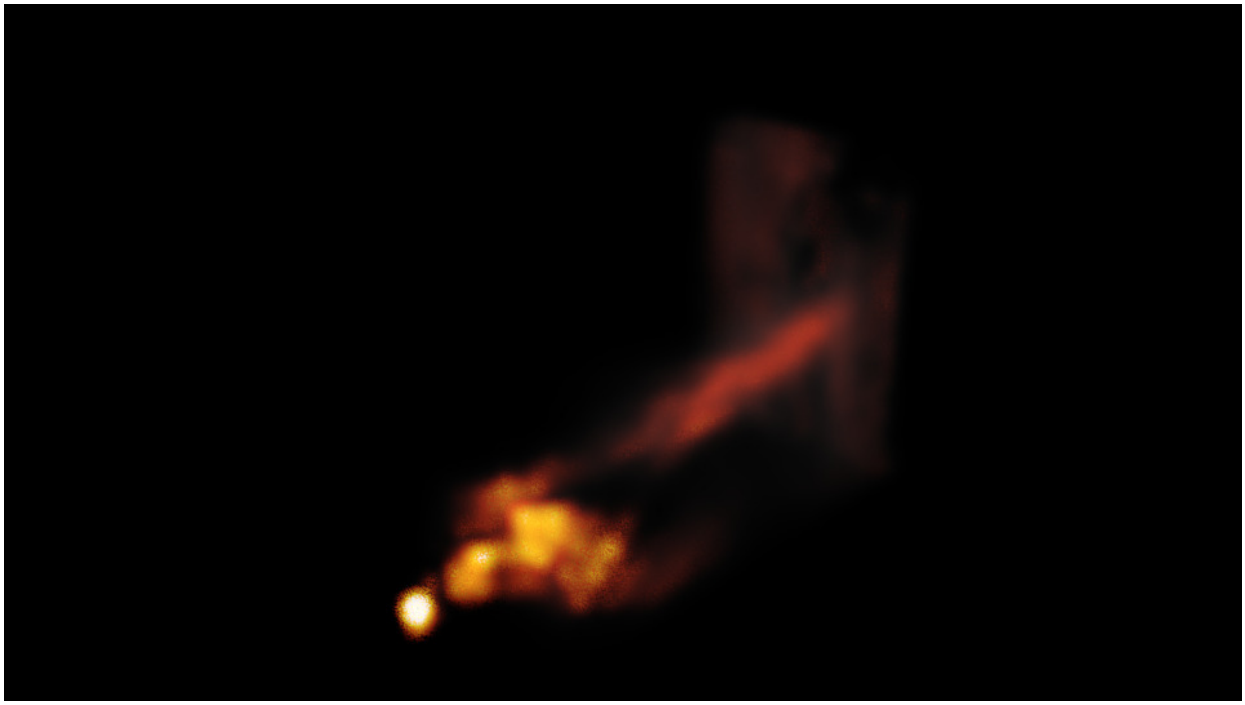


Fig. 2.1600: The fire render.

Wire Render

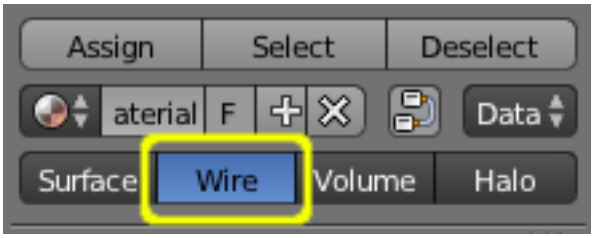


Fig. 2.1601: Wire Render.

The Wire Render option in the Materials section provides a way of showing a rendered image of the edges in an object. Each edge is rendered as a single-pixel image of the edges which make up the mesh. The colors, alpha and other relevant properties of the lines are selected with the same control panels as provided by the Surface rendered image.

Textures

Introduction

In CGI, texture mapping is a method to add detail to surfaces by projecting images and patterns onto those surfaces. The projected images and patterns can be set to affect not only color, but also specularity, reflection, transparency, and even fake 3-dimensional depth. Most often, the images and patterns are projected during render time, but texture mapping is also used to sculpt, paint and deform objects.

See also:

Texture processing for *Combined Textures* in the Compositor.

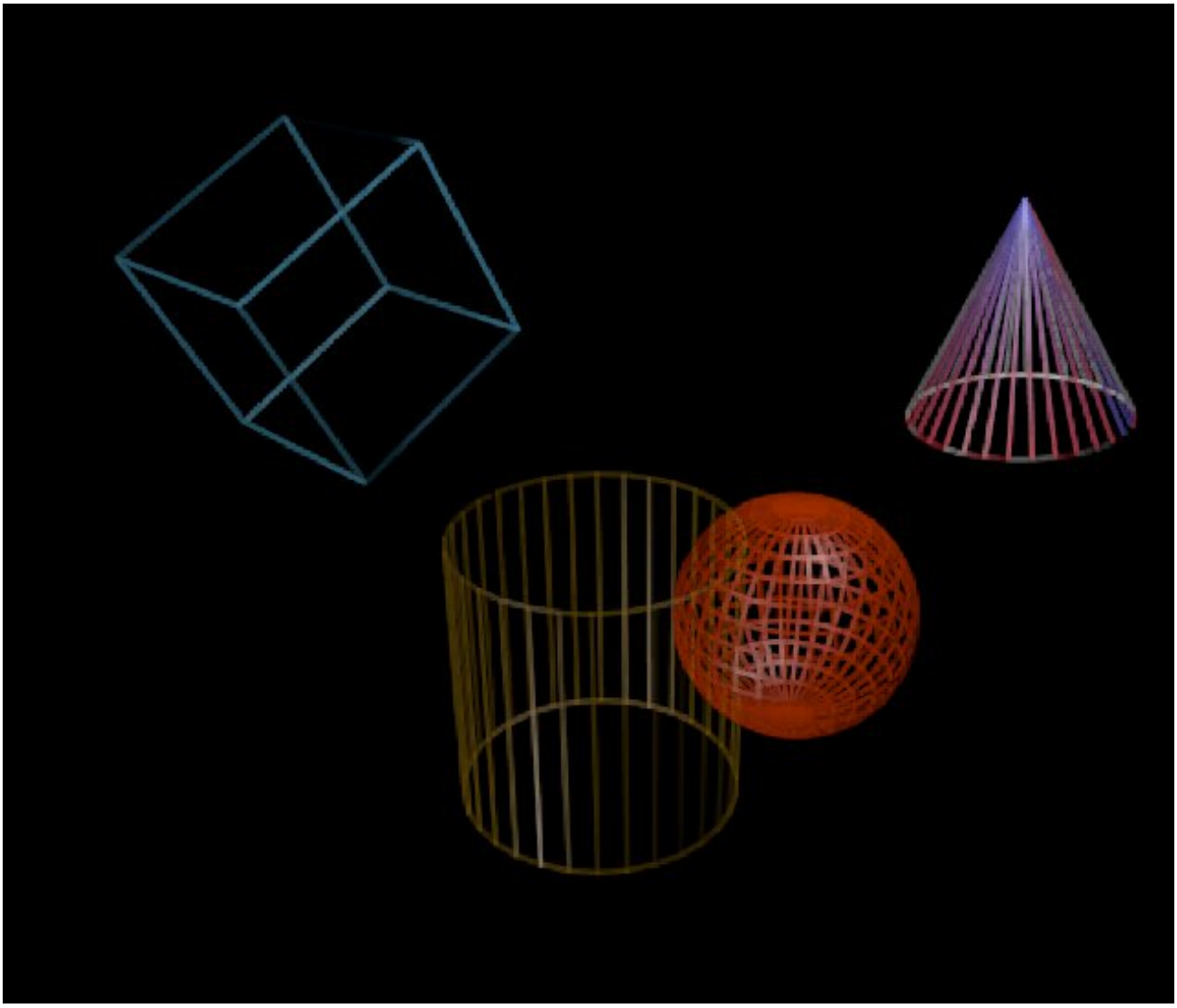


Fig. 2.1602: Wire Render.

Material Textures

The material settings that we've seen so far produce smooth, *uniform* objects, but such objects are not particularly true to reality, where uniformity tends to be uncommon and out of place. In order to deal with this unrealistic uniformity, Blender allows the user to apply *textures* which can modify the reflectivity, specularity, roughness and other surface qualities of a material.

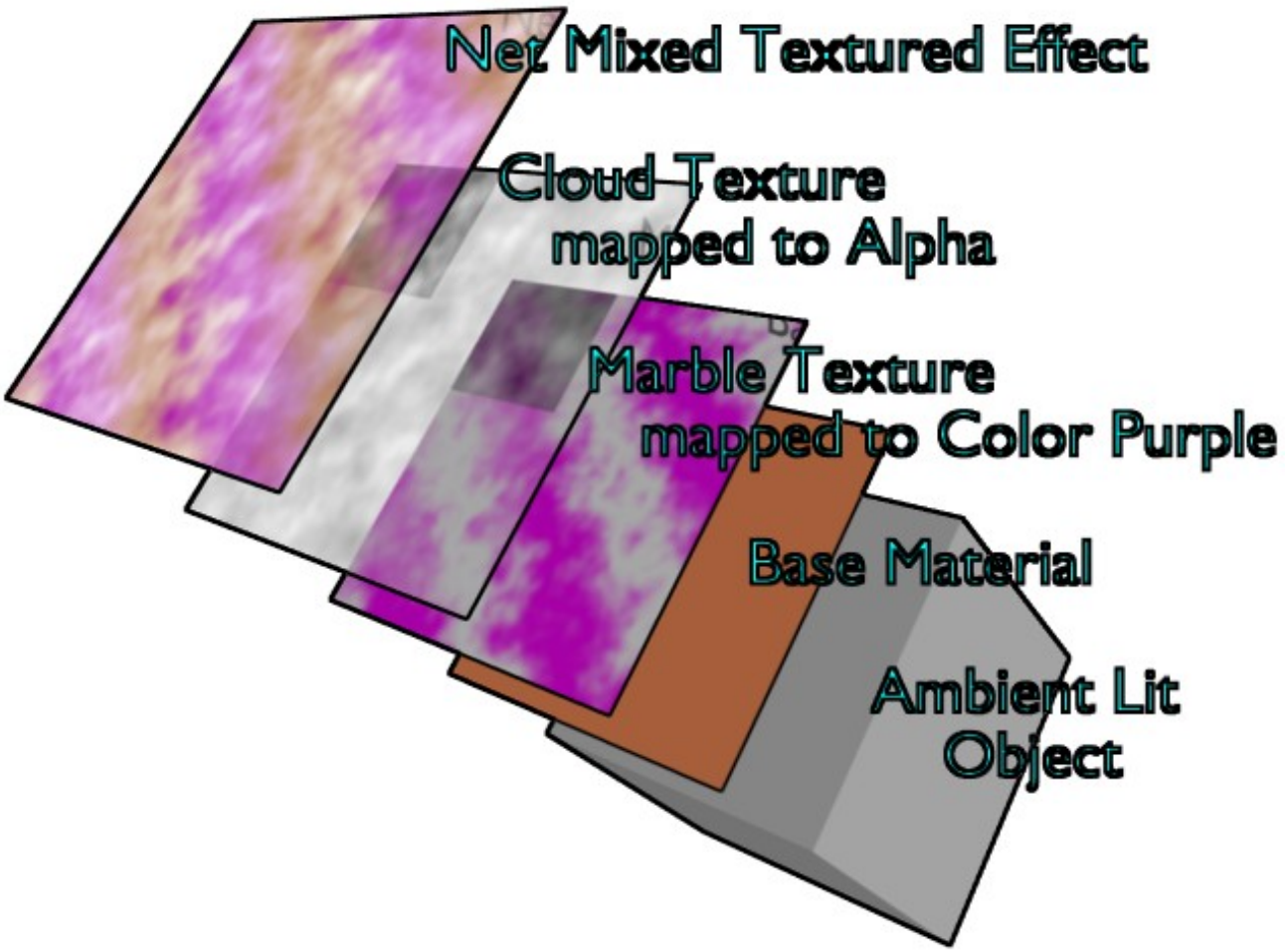


Fig. 2.1603: Textures Layer on base Material.

Textures are like additional layers on top of the base material. Textures affect one or more aspects of the object's net coloring. The net color you see is a sort of layering of effects, as shown in this example image. The layers, if you will, are:

- Your object, lit with *ambient* light based on your world settings.
- Your base *material*, which colors the whole surface in a uniform color that reacts to light, giving different shades of the diffuse, specular, and mirror colors based on the way light passes through and into the surface of the object.
- A *primary texture* layer that overlays a purple marble coloring.
- A *second cloud texture* that makes the surface transparent in a misty/foggy sort of way by affecting the Alpha value.
- These two textures are *mixed* with the base material to provide the net effect:

a cube of purplish-brown fog.

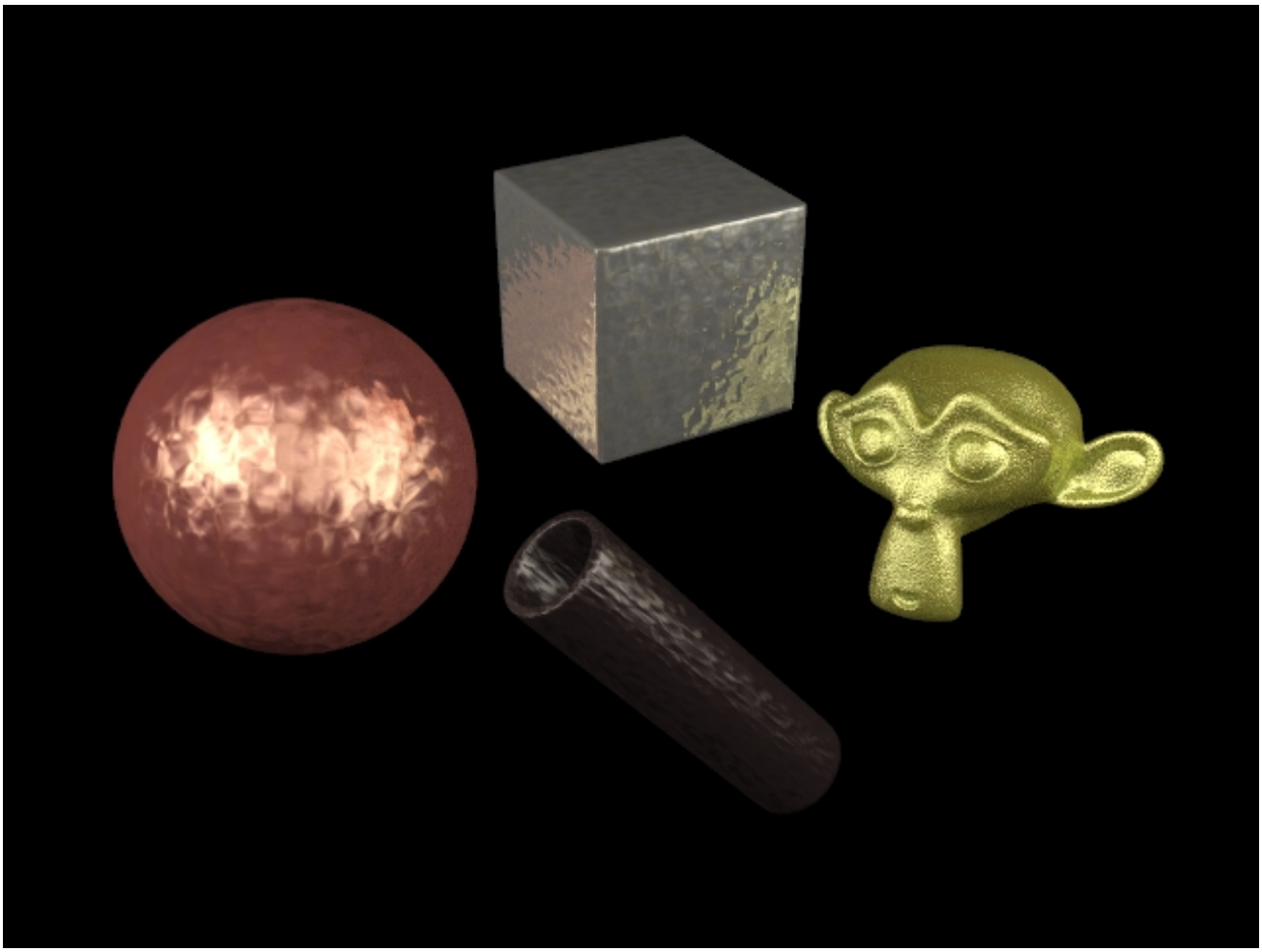


Fig. 2.1604: Some Metal Textures.

This notion of using *more than one* texture, to achieve a combined effect, is one of the "hidden secrets" of creating realistic-looking objects. If you carefully "look at the light" while examining any real-life object, you will observe that the final appearance of that object is best described as the combination, in different ways and in different amounts, of several distinct underlying visual characteristics. These characteristics might be more (or less) strongly apparent at different angles, under different lighting conditions, and so forth. Blender allows you to achieve this in many ways.

You can use "a stack of texture layers" as described in *this section*, or you can also use arbitrarily-complex networks of "texture nodes" as discussed *here*.

Texture Panel

In the Properties editor, choose the Texture tab: this will show the Texture panel.

Texture Context The radio button selects the texture data type, that is, the kind of texture that is being edited.

World *World Background*.

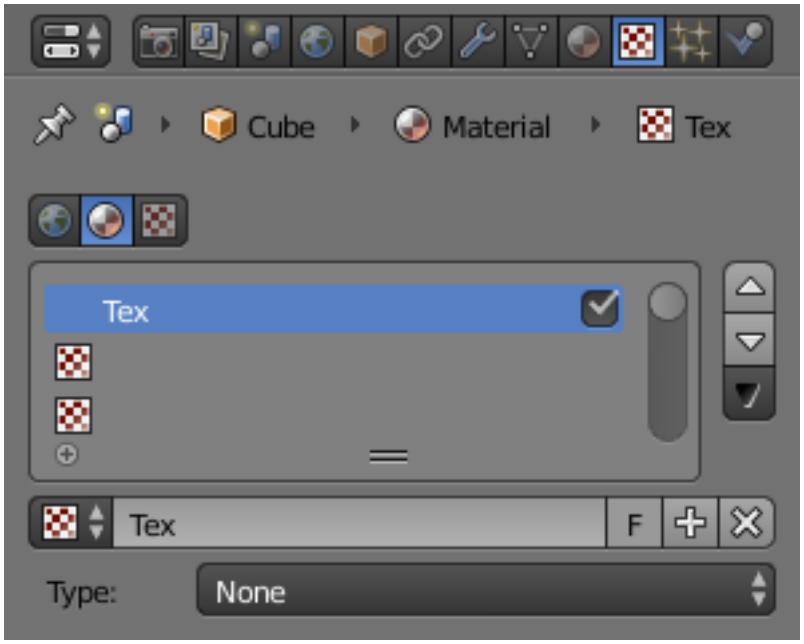


Fig. 2.1605: Texture panel.

Material/Lamp Material type is described in the following section. *Lamps Textures* in the lightning section.

Brush Brush textures are applied in *Painting & Sculpting*.

Textures Stack

Active Texture The Texture slots are displayed in a *List Views & Presets*. The order in the stack defines how textures are overlayed in the rendered image. The checkbox enables/disables the selected texture.

Texture Data-Block

Texture The Texture *Data-Block Menu* for the selected texture slot.

Texture Type

Texture Type Choose the type of texture that is used for the current texture data-block. These types are described in detail *in this section*.

Assigning a Texture

This page just shows how to add a texture to a slot. The *Texture Panel* is explained on the previous page.

Creating a new Texture Data-Block in a new Texture Slot

Select an empty slot, then click on the *New* button.

This will do two things:

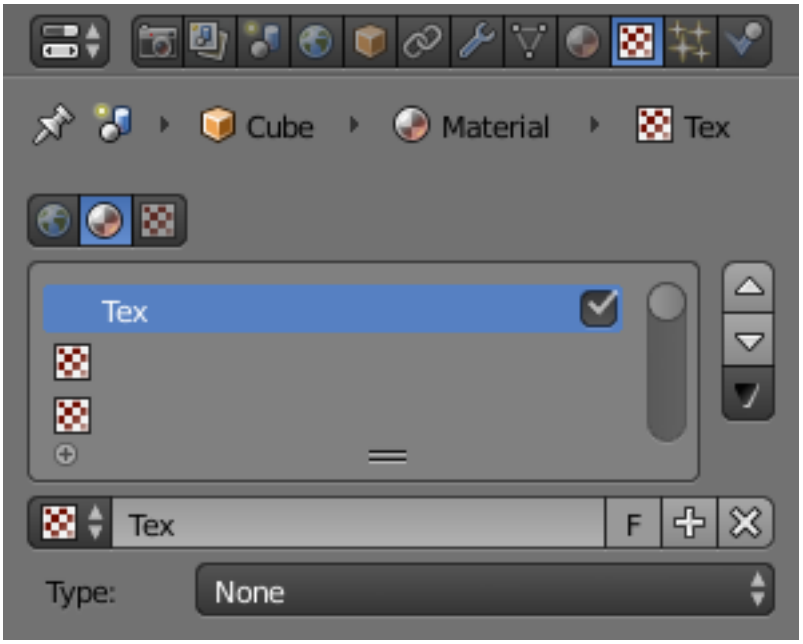


Fig. 2.1606: Texture panel.

- It will create a new texture data-block.
- Also, it will add a new slot in the textures stack.

Creating a new Texture Data-Block in a non-empty slot

Select a non-empty slot, then click on the *Plus* button.

This will do two things:

- It will create a new texture data-block, with a new name, by making a copy of the texture data-block assigned to the selected slot.
- It will assign this new data-block to the selected slot.

Sharing a Texture Data-Block in a non-empty slot

- Select a non-empty slot, then click on the *Browse* button. This will open a menu showing all the available Texture data-blocks in this file.
- Choose a texture data-block in the menu to assign it to the selected slot. This will share the chosen texture with more than one object, hence the *Number of users* shown in the texture data-block will increase by one.

Texture Properties

Preview

The texture preview panel provides a quick pre-visualization of how the texture looks on its own, without mapping.

Preview Choose to display only the flat texture, only the material *Preview*, or both side-by-side.

Texture, Material/World, Both

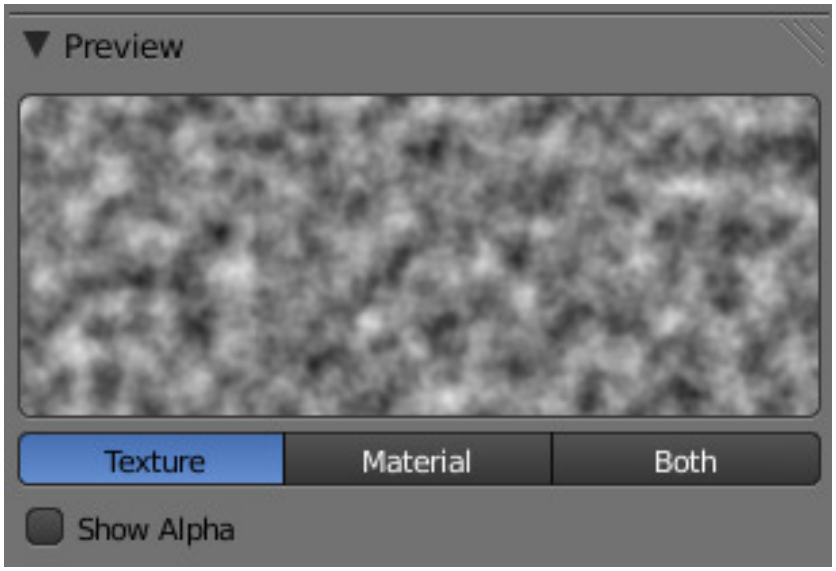


Fig. 2.1607: Preview panel.

Show Alpha Show alpha in preview:

- If Alpha: Use is checked in the *Image Sampling* panel, the image's alpha channel is displayed.
- If Alpha: Use is unchecked, an alpha channel based on averaged rgb values is displayed like it would be used by the Alpha slider in the *Influence* panel.

Colors

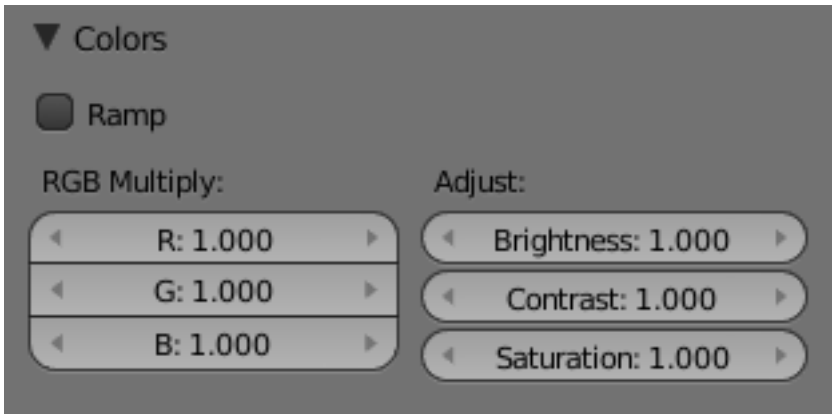


Fig. 2.1608: Colors panel.

The *Ramp* button activates a color ramp which allows you to remap the colors of a texture to new ones. See *Ramps* for information on using ramps.

The color of a texture can be modified with the *Brightness*, *Contrast*, and *Saturation* buttons. All textures with RGB-Values, including *Images* and *Environment Maps*, may be modified with the RGB sliders.

R, G, B Tint the color of a texture by brightening each red, green and blue channel.

Brightness Change the overall brightness/intensity of the texture.

Contrast Change the contrast of the texture.

Saturation Change the saturation of the texture.

Mapping

Textures need mapping coordinates, to determine how they are applied to the object. The mapping specifies how the texture will ultimately wrap itself to the object.

For example, a 2D image texture could be configured to wrap itself around a cylindrical shaped object.

Fig. 2.1609: Mapping panel.

Coordinates

Mapping works by using a set of coordinates to guide the mapping process. These coordinates can come from anywhere, usually the object to which the texture is being applied to.

Global The scene's global 3D coordinates. This is also useful for animations; if you move the object, the texture moves across it. It can be useful for letting objects appear or disappear at a certain position in space.

Object Uses an object as source for coordinates. Often used with an *Empty*, this is an easy way to place a small image at a given point on the object. This object can also be animated, to move a texture around or through a surface.

Object Select the name of an object.

Generated The original undeformed coordinates of the object. This is the default option for mapping textures.

UV UV mapping is a very precise way of mapping a 2D texture to a 3D surface. Each vertex of a mesh has its own UV co-ordinates which can be unwrapped and laid flat like a skin. You can almost think of UV coordinates as a mapping that works on a 2D plane with its own local coordinate system to the plane on which it is operating on. This mapping is especially useful when using 2D images as textures, as seen in *UV Mapping*. You can use multiple textures with one set of UV coordinates.

Layer UV layer to use for mapping.

Strand/Particle Uses normalized 1D strand texture coordinate or particle age (X) and trail position (Y). Use when texture is applied to hair strands or particles.

Window The rendered image window coordinates. This is well suited to blending two objects.

Normal Uses the direction of the surface's normal vector as coordinates. This is very useful when creating certain special effects that depend on viewing angle.

Reflection Uses the direction of the reflection vector as coordinates. This is useful for adding reflection maps. You will need this input when Environment Mapping.

Stress Uses the difference of edge length compared to original coordinates of the mesh. This is useful, for example, when a mesh is deformed by modifiers.

Tangent Uses the optional tangent vector as texture coordinates.

Projection

Flat Flat mapping gives the best results on single planar faces. It does produce interesting effects on the sphere, but compared to a sphere-mapped sphere the result looks flat. On faces that are not in the mapping plane the last pixel of the texture is extended, which produces stripes on the cube and cylinder.

Cube Cube mapping often gives the most useful results when the objects are not too curvy and organic (notice the seams on the sphere).

Tube Tube mapping maps the texture around an object like a label on a bottle. The texture is therefore more stretched on the cylinder. This mapping is of course very good for making the label on a bottle or assigning stickers to rounded objects. However, this is not a cylindrical mapping so the ends of the cylinder are undefined.

Sphere Sphere mapping is the best type for mapping a sphere, and it is perfect for making planets and similar objects. It is often very useful for creating organic objects. It also produces interesting effects on a cylinder.

Inheriting coordinates from the parent object

From Dupli Duplis instanced from vertices, faces, or particles, inherit texture coordinates from their parent.

Coordinate Offset, Scaling and Transformation

Offset The texture co-ordinates can be translated by an offset. Enlarging of the Offset moves the texture towards the top left.

Size These buttons allow you to change the mapping of axes between the texture's own coordinate system, and the mapping system you choose (Generated, UV, etcetera.) More precisely, to each axis of the texture corresponds one of four choices, that allow you to select to which axis in the mapping system it maps! This implies several points:

- For 2D textures (such as images), only the first two rows are relevant, as they have no Z data.
- You can rotate a 2D picture a quarter turn by setting the first row (i.e. X texture axis) to Y, and the second row (Y texture axis) to X.

- When you map no texture axis (i.e. the three "void" buttons are set), you will get a solid uniform texture, as you use zero dimension (i.e. a dot, or pixel) of it (and then Blender extends or repeats this point's color along all axes.)
- When you only map one texture axis (i.e. two "void" buttons are enabled) you will get a "striped" texture, as you only use one dimension (i.e. a line of pixel) of it, (and then Blender stretches this line along the two other axes).
- The same goes, for 3D textures (i.e. procedural ones), when one axis is mapped to nothing, Blender extends the plan ("slice") along the relevant third axis.

Influence

Introduction

Not only can textures affect the color of a material, they can also affect many of the other properties of a material. The different aspects of a material that a texture influences are controlled in the *Influence* panel.

Note: Texture options for *Surface* and *Wire* materials and in some cases also for *Volume* and *Halo* materials.

Surface and Wire materials

Diffuse

Intensity Amount texture affects affects diffuse reflectivity

Color Amount texture affect the basic color or RGB value of the material

Alpha Influences the opacity of the material. Also use *Z Transparency* for light and if combining multiple channels.

Translucency Influences the Translucency amount.

Specular

Intensity Amount texture affect specular reflectivity

Color Influences the *Specular* color, the color of the reflections created by the lamps on a glossy material.

Hardness Influences the specular hardness amount. A DVar of 1 is equivalent to a Hardness of 130, a DVar of 0.5 is equivalent to a Hardness of 65.

Shading

Ambient Influences the amount of Ambient light the material receives.

Emit Influences the amount of light Emitted by the material.

Mirror Influences the mirror color. This works with environment maps and ray-traced reflection.

Ray Mirror Influences the strength of raytraced mirror reflection.

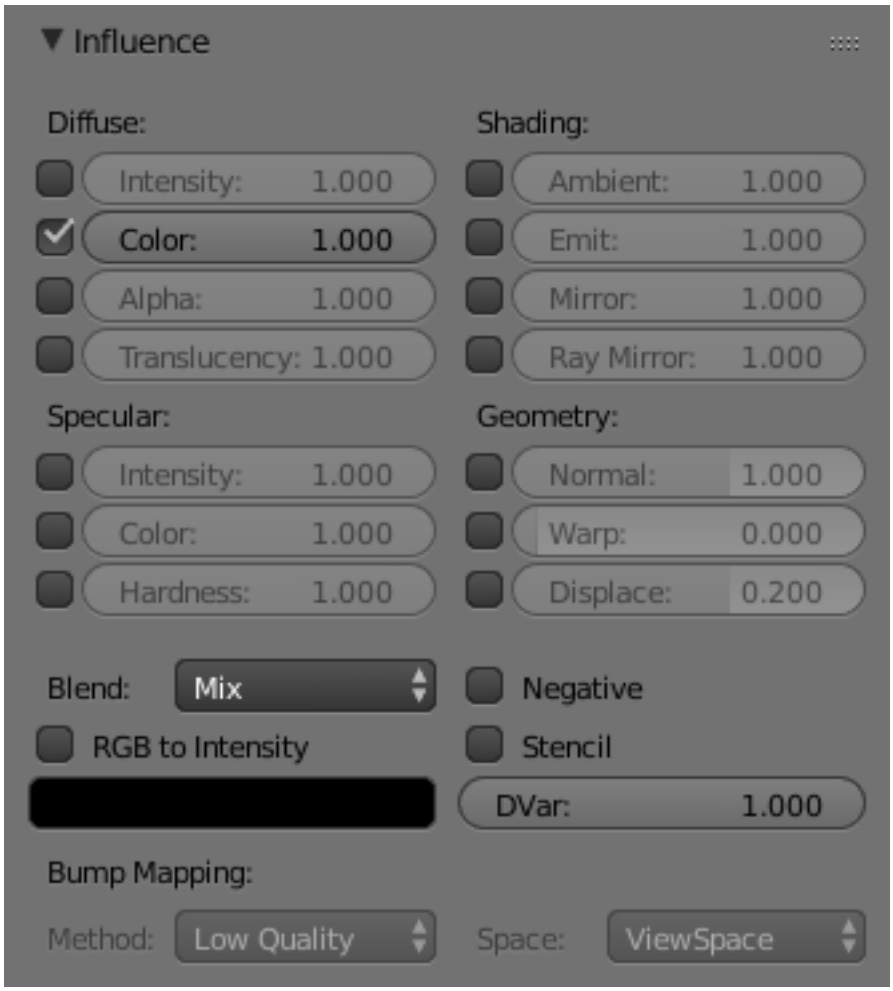


Fig. 2.1610: Texture Influence panel for a Surface material.

Geometry

Normal Commonly called bump mapping, this alters the direction of the surface normal. This is used to fake surface imperfections or unevenness via bump mapping, or to create reliefs.

Warp *Warp* allows textures to influence/distort the texture coordinates of a next texture channel. The distortion remains active over all subsequent channels, until a new Warp has been set. Setting the factor at zero cancels out the effect.

Displace Influences the Displacement of vertices, for using *Displacement Maps*.

Other Controls

Bump Mapping Settings for bump mapping.

Method Best Quality, Default, Compatible, Original

Space Texture Space, Object Space, View Space

Volume materials

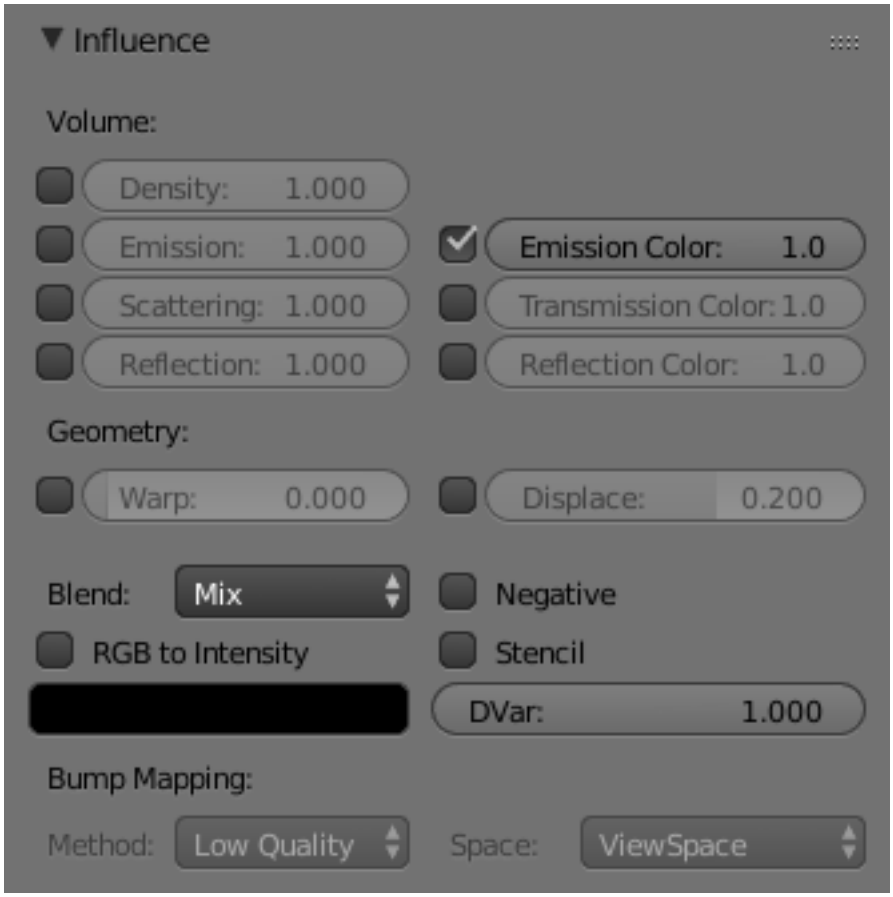


Fig. 2.1611: Texture Influence panel for Volume material.

Special texture options for *Volume* materials.

Density Causes the texture to affect the volume's density.

Emission Causes the texture to affect the volume's emission.

Scattering Amount the texture affects scattering.

Reflection Amount the texture affects brightness of out-scattered light

Emission Color Amount the texture affects emission color.

Transmission Amount the texture affects result color after light has been scattered/absorbed.

Reflection Color Amount the texture affects color of out-scattered light.

Halo materials

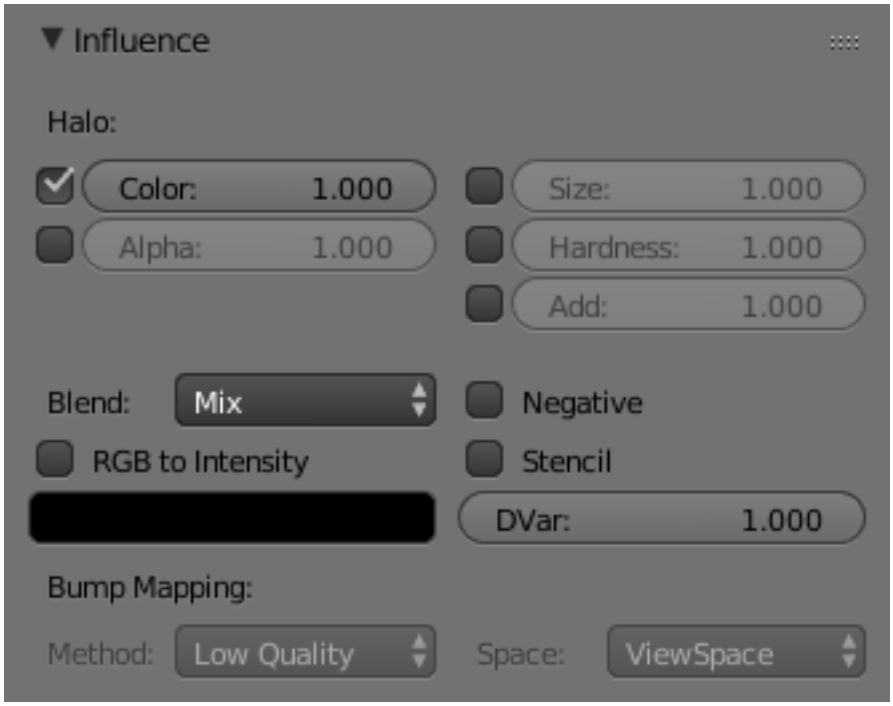


Fig. 2.1612: Texture Influence panel for a Halo material.

Special texture options for *Halo* materials.

Size Amount the texture affects ray mirror.

Hardness Amount the texture affects hardness.

Add Amount the texture affects translucency.

Texture Blending Modes

Blending Modes are different methods of controlling how the texture influences material properties. While a blending mode defines the specific operation performed, blending factor controls the amount, the overall "strength" of this operation. For textures such blending factor is set via sliders in the Influence panel.

Blend Blending operation to perform. See *Color Blend Modes* for details on each blending mode.

RGB to Intensity With this option enabled, an RGB texture (affects color) is used as an intensity texture (affects a value).

Blend Color If the texture is mapped to Color, what color is blended in according to the intensity of the texture?

Negative The effect of the Texture is negated. Normally white means on, black means off, *Negative* reverses that.

Stencil The active texture is used as a mask for all following textures. This is useful for semitransparent textures and "Dirt Maps". Black sets the pixel to "untexturable". The *Stencil* mode works similar to a layer mask in a 2D program. The effect of a stencil texture cannot be overridden, only extended. You need an intensity map as input.

Destination Value The value (not for RGB) with which the Intensity texture blends with the current value. Two examples:

- The *Emit* value is normally 0. With a texture mapped to *Emit* you will get maximal effect, because *DVar* is 1 by default. If you set *DVar* to 0 no texture will have any effect.
- If you want transparent material, and use a texture mapped to *Alpha*, nothing happens with the default settings, because the *Alpha* value in the *Material* panel is 1. So you have to set *DVar* to 0 to get transparent material (and of course *Z Transparency* also). This is a common problem for beginners. Or do it the other way round: set *Alpha* to 0 and leave *Dvar* on 1. Of course the texture is used inverted then.

Bump and Normal Maps

Normal Maps and *Bump Maps* both serve the same purpose: they simulate the impression of a detailed 3D surface, by modifying the shading as if the surface had lots of small angles, rather than being completely flat. Because it is just modifying the shading of each pixel, this will not cast any shadows and will not obstruct other objects. If the camera angle is too flat to the surface, you will notice that the surface is not really shaped.

Both *Bump Maps* and *Normal Maps* work by modifying the normal angle (the direction pointing perpendicular from a face), which influences how a pixel is shaded. Although the terms *Normal Map* and *Bump Map* are often used synonymously, there are certain differences.

Bump maps These are textures that store an *intensity*, the relative height of pixels from the viewpoint of the camera. The pixels seem to be moved by the required distance in the direction of the face normals. (The "bump" consists only of a displacement, which takes place along the existing, and unchanged, normal-vector of the face.) You may either use grayscale pictures or the intensity values of a RGB-Texture (including images).

Normal maps These are images that store a *direction*, the direction of normals directly in the RGB values of an image. They are much more accurate, as rather than only simulating the pixel being away from the face along a line, they can simulate that pixel being moved at any direction, in an arbitrary way. The drawbacks to normal maps are that unlike bump maps, which can easily be painted by hand, normal maps usually have to be generated in some way, often from higher resolution geometry than the geometry you are applying the map to.

Normal maps in Blender store a normal as follows:

- Red maps from (0 - 255) to X (-1.0 - 1.0)
- Green maps from (0 - 255) to Y (-1.0 - 1.0)
- Blue maps from (0 - 255) to Z (0.0 - 1.0)

Since normals all point towards a viewer, negative Z-values are not stored (they would be invisible anyway). In Blender we store a full blue range, although some other implementations also map blue colors (128 - 255) to (0.0 - 1.0). The latter convention is used in "Doom 3" for example.

Workflow

The steps involved in making and using Bump and Normal Maps is:

- Model a highly detailed ("hi-poly") model.
- Bake the Bump and/or Normal maps.
- Make a low-poly, less detailed model.
- Map the map to the low-poly model using a common coordinate system.

Consult the Modeling section for how to model a highly detailed model using the Mesh tools. How much detail you put in is totally up to you. The more ridges and details (knobs, creases, protrusions) you put in, the more detailed your map will be.

Baking a map, simply put, is to take the detail of a high polygon mesh, and apply it to a similar object. The similar object is identical to the high-poly mesh except with less vertices. Use the *Render Bake* feature in Blender to accomplish this.

Modeling a low-poly using Blender's Mesh editing tools. In general, the same or similar faces should exist that reflect the model. For example, a highly detailed ear may have 1000 faces in the high-poly model. In the low-poly model, this may be replaced with a single plane, oriented in the same direction as the detailed ear mesh. *(Tip:* Blender's *multi-resolution mesh* modeling feature can be used to good effect here.)

Mapping is the process of applying a texture to the low-poly mesh. Consult the *Textures Mapping section* for more information on applying a texture to a mesh's material.Special considerations for Bump and Normal Maps is:

- When using a Bump map, map the texture to *Normal* and enable *No RGB*.
- When using a Normal map, map the texture to *Normal*.

The coordinate systems of the two objects must match. For example, if you bake using a UV map of the high-poly model, you must UV map the low poly model and line up its UV coordinates to match the outline of the high-poly image (see *UV unwrapping* to line up with the high-poly map edges.

Displacement Maps

Displacement mapping allows a texture input to manipulate the position of vertices on rendered geometry. Unlike *Normal or Bump mapping*, where the shading is distorted to give an illusion of a bump (discussed on the previous page), Displacement Maps create real bumps, creases, ridges, etc in the actual mesh. Thus, the mesh deformations can cast shadows, occlude other objects, and do everything that changes in real geometry can do, but, on the other hand, requires a lot more vertices to work.

Options

In the *Influence panel*, the strength of the displacement is controlled by the *Displace* and *Normal* sliders:

- If a texture provides only normal information (e.g. *Stucci*), vertices move according to the texture's normal data. The normal displacement is controlled by the *Normal* slider.
- If a texture provides only intensity information (e.g. *Magic*, derived from color), vertices move along the directions of their normals (a vertex has no normal itself, it is the resulting vector of the adjacent faces). White pixels move outward in the direction of the normal, black pixels move in the opposite direction. The amount of displacement is controlled with the *Displace* slider.

  The two modes are not exclusive. Many texture types provide both information (*Clouds*, *Wood*, *Marble*, *Image*). The amount of each type can be mixed using the respective sliders. Intensity displacement gives a smoother, more continuous surface, since the vertices are displaced only outward. Normal displacement gives a more aggregated surface, since the vertices are displaced in multiple directions.

  The depth of the displacement is scaled with an object's scale, but not with the relative size of the data. This means if you double the size of an object in object mode, the depth of the displacement is also doubled, so the relative displacement appears the same. If you scale inside *Edit Mode*, the displacement depth is not changed, and thus the relative depth appears smaller.

Hints

Displacement maps move the rendered faces, not the physical mesh faces. So, in 3D View the surface may appear smooth, but render bumpy. To give a detailed surface, there has to be faces to displace and have to be very small. This creates the trade-off between using memory and CPU time versus render quality.

From best to worst, displacement works with these object types using the methods listed to control the render face size:

***Subdivision Surface* Meshes** Rendered face size is controlled with render subdivision level. Displacement really likes smooth normals.

Manually (*Edit Mode*) *subdivided* meshes Control render faces with number of subdivides. (This can be combined with the above methods). Displaces exactly the same Simple Subdivision Surface, however, the overhead of drawing extra faces can slow down editing.

Meta Objects Control render faces with render wiresize. Small wire == more faces.

The following are available, but currently do not work well. It is recommended that you convert these to meshes before rendering.

Open *NURBS Surfaces* Control render faces with U/V *Surface Resolution*. Higher numbers give more faces. (Note normal errors).

Closed NURBS Surfaces Control with *Surface Resolution* controls. (Note the normal errors, and how implicit seam shows).

Curves* and *Text Control with *Surface Resolution* controls. Higher gives more render faces. (Note that the large flat surfaces have few render faces to displace).

Note: Displace Modifier

If you want more control over your displacement, you will probably want to use the *Displace Modifier*. This feature has lots of different options so that you can customize the displacement exactly to your liking.

Texture Types

Image or Movie

Introduction

The term *Image Texture* simply means that a graphic image, which is a pixel grid composed of R, G, B, and sometimes Alpha values. It is used as the input source to the texture. As with other types of textures, this information can be used in a number of ways, not only as a simple "decal".

Video textures are a some kind of Image textures and based on movie file or sequence of successive numbered separate images. They are added in the same way that image textures are.

When the Texture Type *Image or Movie* is selected, three new panels present themselves allowing to control most aspects of how image textures are applied: *Image*, *Image Sampling*, and *Image Mapping*.

About Image Based Texturing

Texture images take up precious memory space, often being loaded into a special video memory bank that is very fast and very expensive, so it is often very small. So, keep the images as small as possible. A 64×64 image takes up only one fourth the memory of a 128×128 image.

For photo-realistic rendering of objects in animations, often larger image textures are used, because the object might be zoomed in on in camera moves. In general, you want to use a texture sized proportionally to the number of pixels that it will occupy in the final render. Ultimately, you only have a certain amount of physical RAM to hold an image texture and the model and to provide work space when rendering your image.

For the most efficient memory usage, image textures should be square, with dimensions as powers of 2, such as 32×32, 64×64, 128×128, 256×256, 1024×1024, 2048×2048, and 4096×4096.

If you can re-use images across different meshes, this greatly reduces memory requirements. You can re-use images if you map those areas of the meshes that "look alike" to a layout that uses the common image. In the overview below, the left image is re-used for both the sphere and a portion of the monkey. The monkey uses two layouts, one which has one UV map of a few faces, and another that has three maps.

When using file textures, it is very important that you have *Mapped the UVs* of the mesh, and they are laid out appropriately.

You do not have to UV map the *entire* mesh. The sphere above on the left has some faces mapped, but other faces use procedural materials and textures. Only use UV Textures for those portions of your mesh where you want very graphic, precise detail. For example, a model of a vase only needs UV Texture for the rim where decorative artwork is incorporated. A throw pillow does not need a different image for the back as the front; in fact many throw pillows have a fabric (procedural material) back.

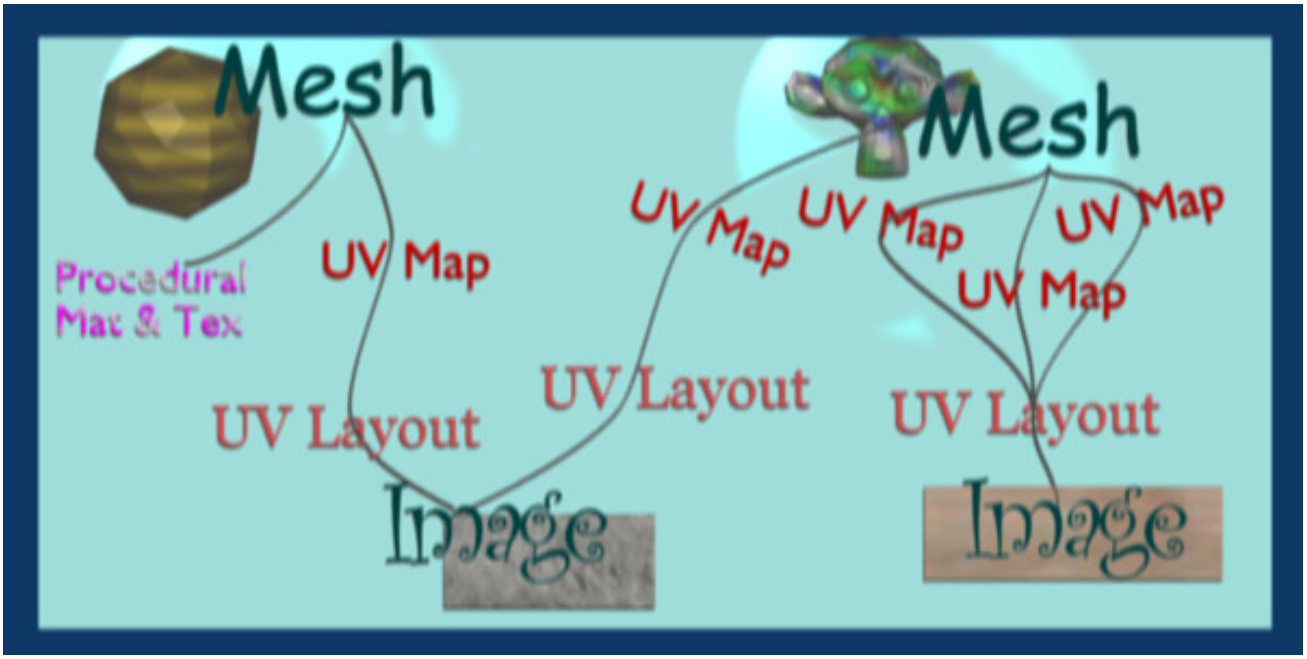


Fig. 2.1613: How all the parts of UV Texturing work together.

As another example, you should UV map both eyes of a head to the same image (unless you want one bloodshot and the other clear). Mapping both sides of a face to the same image might not be advisable, because the location of freckles and skin defects are not symmetrical. You could of course change the UV map for one side of the face to slightly offset, but it might be noticeable. Ears are another example where images or section of an images can be mapped to similar faces.

UV Textures vs. Procedural Textures

A Material Texture, that has a Map Input of UV, and is an image texture that is mapped to Color, is equivalent to a UV Texture. It provides much more flexibility, because it can be sized and offset, and the degree to which it affects the color of your object can be controlled in the Map To panel. In addition, you can have different images for each texture channel; one for color, one for alpha, one for normals, one for specularity, one for reflectivity, *etc*. Procedural textures, like Clouds, are incredibly simple and useful for adding realism and details to an image.

UV Texture	Procedural Texture
Image maps to precise coordinates on the selected faces of the mesh.	Pattern is generated dynamically, and is mapped to the entire mesh (or portion covered by that material).
The Image maps once to a range of mesh faces specifically selected.	Maps once to all the faces to which that material is assigned; either the whole mesh or a portion.
Image is mapped once to faces.	Size XYZ in the Map Input allows tiling the texture many times across faces. Number of times depends on size of mesh.
Affect the color and the alpha of the object.	Can also affect normals (bumpiness), reflectivity, emit, displacement, and a dozen other aspects of the mesh's appearance; can even warp or stencil subsequent textures.
Can have many for a mesh.	Can be layered, up to 10 textures can be applied, layering on one another. Many mix methods for mixing multiple channels together.
Any Image type (still, video, rendered). Generated test grid available.	Many different types: clouds, wood grain, marble, noise, and even magic.
Provides the UV layout for animated textures.	Noise is the only animated procedural texture.
Takes very limited graphics memory	Uses no or little memory; instead uses CPU compute power.

So, in a sense, a single UV texture for a mesh is simpler but more limited than using multiple textures (mapped to UV coordinates), because they do one specific thing very well: adding image details to a range of faces of a mesh. They work together if the procedural texture maps to the UV coordinates specified in your layout. As discussed earlier, you can map multiple UV textures to different images using the UV Coordinate mapping system in the Map Input panel.

Workflow

The process consists of the following steps:

- Create the Mesh. *Unwrap* it into one or more *UV Layouts*.
- Create one or more Materials for the Mesh.
- Create one or more images for each UV Layout and aspect of the texture. Either - Paint directly on the mesh using Texture Paint in the 3D View, - Load and/or edit an image in the UV/Image Editor, or - Bake the existing materials into an image for the UV/Image Editor.
- Apply those images as UV Textures to the mesh to affect one or more aspects of the mesh. This is done by using one or more of the numerous Map To options. For example, - Map to Color to affect the diffuse coloring of the mesh, - Map to Nor to affect the normal direction to give the surface a bumpy or creased look, or - Map to Spec (specularity) to make certain areas look shiny and oily.
- Layer the Textures to create a convincing result.

Using Images and Materials

To use an image as the color and alpha (transparency) of the texture, you can create an image in an external paint program and tell the UV/Image Editor to Open that file as the texture, or you can create a New image and save it as the texture.

If you want to start off by creating an image using an external paint program, you will want to save an outline of your UV faces by using the *Save UV Face Layout* tool located in the UVs menu. This is discussed *here*.

Creating an Image Texture

To create an image within Blender, you have to first create a *New Blank* Image with a uniform color or test grid. After that, you can color the image using the:

- Vertex colors as the basis for an image.
- Render Bake image based on how the mesh looks in the scene.

After you have created your image, you can modify it using Blender's built-in *Texture Paint* or any external image painting program.

Note: See Texture in 3D View but does not Render

You may be able to see the texture in Textured display mode in the 3D View; this is all that is required to have textures show up in Blender's Game Engine. Rendering, however, requires a material. You must have a *Face Textures* material assigned to the mesh for it to render using the UV Texture. In the Material settings, Add New material to a selected object and enable *Face Textures*.

Examples

There may be one UV Layout for the face of a character, and another for their clothes. Now, to texture the clothes, you need to create an image at least for the Color of the clothes, and possible a "bump" texture to give the fabric the appearance of some weave by creating a different image for the Normal of the clothes. Where the fabric is worn, for example at the elbows and knees, the sheen, or Specularity, of the fabric will vary and you will want a different image that tells Blender how to vary the Specularity. Where the fabric is folded over or creased, you want another image that maps Displacement to the mesh to physically deform the mesh. Each of these are examples of applying an image as a texture to the mesh.

As another example, the face is the subject of many questions and tutorials. In general, you will want to create a Material that has the basic skin color, appropriate shaders, and sub-surface scattering. Then you will want to layer on additional UV Textures for:

- Freckle map for Color and Normal aspects.
- Subdermal veins and tendons for Displacement.
- Creases and Wrinkles and skin cell stratification for Normal.
- Makeup images for Color.
- Oily maps for Specularity.
- For a zombie, Alpha transparency where the flesh has rotted away.
- Under chin and inside nostrils that receive less Ambient light.
- Thin skin is more translucent, so a map is needed for that.

Each image is mapped by using another Texture Channel. Each of these maps are images which are applied to the different aspects (Color, Normal, Specularity) of the image. Tileable images can be repeated to give a smaller, denser pattern by using the Texture controls for repeat or size.

Layering UV Textures

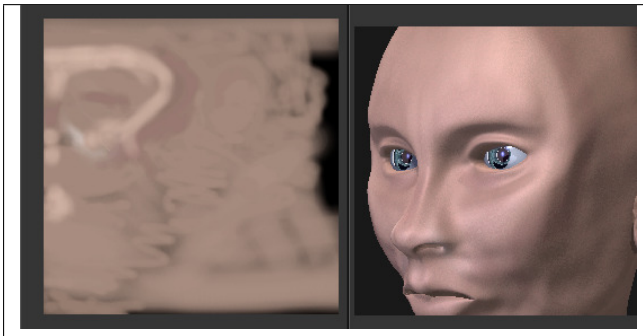


Fig. 2.1614: Base UV Texture.

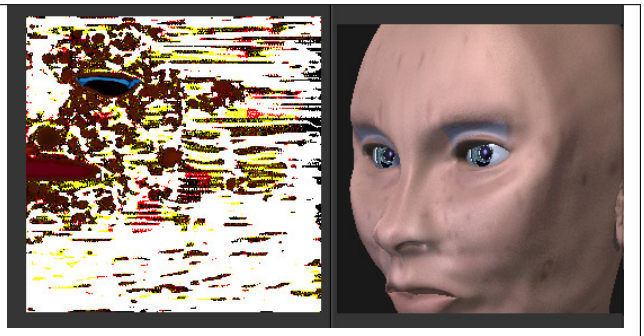


Fig. 2.1615: Layered UV Texture.

Great textures are formed by layering images on top of one another. You start with a base layer, which is the base paint. Each successive layer on top of that is somewhat transparent to let the bottom layers show through, but opaque where you want to add on to details.

To avoid massive confusion, all image textures for a mesh usually use the same UV map. If you do, each image will line up with the one below it, and they will layer on top of one another like the examples shown to the right. To do this, just create one UV Texture (map) as described in this section. Then, create material image textures as described in the procedural materials section. Instead of mapping to Original Coordinates (OrCo), map to UV.

Use that map name repeatedly in the *Material* → *Textures* → *Map Input* panel by selecting *UV* and typing the name in the text field. In the example to the right, our UV Texture is called "Head" (you may have to expand the image to see the panel settings). Then, the image texture shown will be mapped using the UV coordinates. In the "Base UV Texture" example to the right, the face has two textures UV mapped; one for a base color, and another for spots, blemishes and makeup.

Both textures use the same UV Texture map as their Map Input, and both affect Color. The Makeup texture is transparent except where there is color, so that the base color texture shows through. Note that the colors were too strong on the image, so they amount of the diffuse color affects is turned down to 60% in the second layer (the blemish layer).

Normally, we think of image textures affecting the color of a mesh. Realism and photo-realistic rendering is a combination of many different ways that light interacts with the surface of the mesh. The image texture can be Mapped To not only color, but also *Normal* (bumpiness) or *Reflection* or any of the other attributes specified in the Map To panel.

If you paint a gray-scale image (laid out according to the UV Layout) with white where the skin is oily and shiny, and dark where it is not, you would map that input image according to the UV Layout, but have it affect Specularity (not color).

To make portions of a mesh transparent and thus reveal another mesh surface underneath, you would paint a gray-scale image with black where you want the texture transparent, map input to UV, and map it to Alpha (not color).

To make portions of a mesh, like a piece of hot metal, appear to glow, you would use a gray-scale image mapped to Emit.

Believe it or not, this is only "the tip of the iceberg!" If everything that is been described here just is not enough for you, the *texture nodes* feature, introduced in recent versions of Blender, enables you to layer and combine textures in almost any way you can imagine.

Mix and Match Materials

You can mix and match procedural materials and textures, vertex paint, and UV textures onto the same mesh.

The image to the right has a world with a red ambient light. The material has both Vertex Color Paint and Face Textures enabled, and receives half of ambient light. A weak cloud texture affects color, mixing in a tan color. The right vertices are vertex painted yellow and the left is unpainted procedural gray. The UV Texture is a stock arrow image from the public domain texture CD. Scene lighting is a white light off to the right. From this information and the User Manual thus far, you should now be able to recreate this image.

You can also assign *multiple materials* to the mesh based on which faces you want to be procedural and which you want to be texture-mapped. Just do not UV map the faces you want to be procedural.

You can use UV Textures and Vertex Paint (V in the 3D View) simultaneously, if both are enabled in the Material settings. The vertex colors are used to modulate the brightness or color of the UV image texture:

- UV Texture is at the base *(Face Textures)*
- Vertex paint affects its colors, then
- Procedural textures are laid on top of that,
- Area lights shine on the surface, casting shadows and what not, and finally
- Ambient light lights it up.

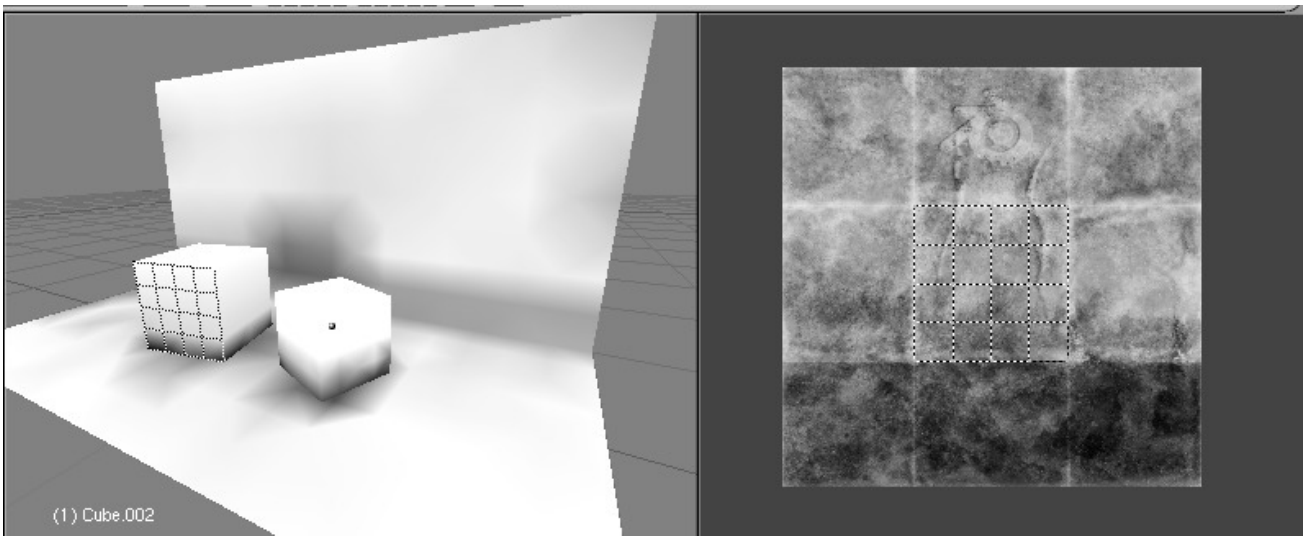


Fig. 2.1616: Vertex colors modulate texture.

A UV Layout can only have one image, although you can tile and animate the image. Since a layout is a bunch of arranged UV Maps, and a UV Map maps many mesh faces, a face can therefore only have one UV Texture image, and the UV coordinates for that face must fit entirely on the image. If you want a face to have multiple images, split the face into parts, and assign each part its own image. *(Or* you can get fancy with Nodes, but that is another story ...)

Using Alpha Transparency

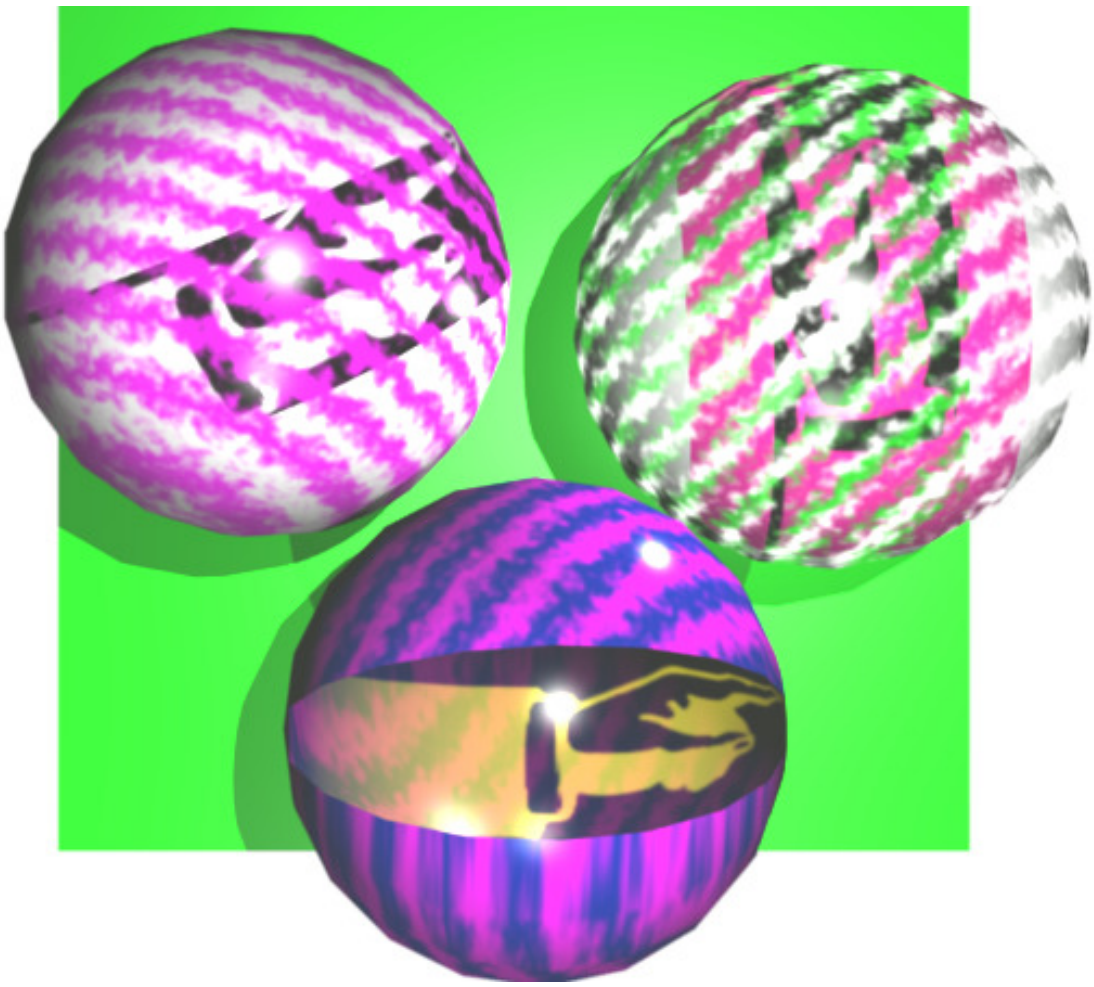


Fig. 2.1617: Alpha UV Textures.

Alpha 0.0 (transparent) areas of a UV Image render as black. Unlike a procedural texture, they do not make the base None material transparent, since UV Textures do not operate on the base procedural material. The UV texture overrides any procedural color underneath. Procedural Textures are applied on top of UV Textures, so a procedural image texture would override any UV Texture. Transparent (black) areas of a procedural texture mapped to alpha operate on top of anything else, making the object transparent in those places. The only thing that modulates visible parts of a UV Texture are the Vertex Colors. In the example to the right, the finger image is transparent at the cuff and top of the finger and is used as a UV Texture. All three balls have a base material of blue and a marbling texture. The base material color is not used whenever Face Textures is enabled.

The top left ball has not had any vertex painting, and the finger is mapped to the middle band, and the texture is mapped to a pink color. As you can see, the base material has Vertex Color Paint and Face Textures enabled; the base color blue is not used, but the texture is. With no vertex painting, there is nothing to modulate the UV Texture colors, so the finger shows as white. Transparent areas of the UV Image show as black.

The top right ball has had a pink vertex color applied to the vertical band of faces (in the 3D View editor, select the faces in UV Paint Mode, switch to Vertex Paint Mode, pick a pink color, and *Paint* → *Set Vertex Colors*). The finger is mapped to the middle vertical band of faces, and Vertex Color and Face Textures are enabled. The texture is mapped to Alpha black

and multiplies the base material alpha value which is 1.0. Thus, white areas of the texture are 1.0, and 1.0 times 1.0 is 1.0 so that area is opaque and shows. Black areas of the procedural texture, 0.0, multiply the base material to be transparent. As you can see, the unmapped faces (left and right sides of the ball) show the vertex paint (none, which is gray) and the painted ones show pink, and the middle stripe that is both painted and mapped change the white UV Texture areas to pink. Where the procedural texture says to make the object transparent, the green background shows through. Transparent areas of the UV Texture insist on rendering black.

The bottom ball uses multiple materials. Most of the ball (all faces except the middle band) is a base material that does not have Face Textures (nor Vertex Color Paint) enabled. Without it enabled, the base blue material color shows and the pink color texture is mixed on top. The middle band is assigned a new material (2 Mat 2) that *does* have vertex paint and Face Textures enabled. The middle band of faces were vertex painted yellow, so the white parts of the finger are yellow. Where the pink texture runs over the UV texture, the mixed color changes to green, since pink and yellow make a green.

If you want the two images to show through one another, and mix together, you need to use Alpha. The base material can have an image texture with an Alpha setting, allowing the underlying UV Texture to show through.

To overlay multiple UV images, you have several options:

- Create multiple UV Textures which map the same, and then use different images (with Alpha) and Blender will overlay them automatically.
- Use the *Composite Nodes* to combine the two images via the Alpha Over node, creating and saving the composite image. Open that composited image as the UV Texture.
- Use an external paint program to alpha overlay the images and save the file, and load it as the face's UV Texture
- Define two objects, one just inside the other. The inner object would have the base image, and the outer image the overlaid image with a material alpha less than one (1.0).
- Use the *Material nodes* to combine the two images via the Alpha Over or Mix node, thus creating a third noded material that you use as the material for the face. Using this approach, you will not have to UV map; simply assign the material to the face using the Multiple Materials.

Options

Image

In the *Image* panel we tell Blender which source file to use.

Image The Image *Data-Block Menu*. For the options see *Image Settings*.

Image Sampling

In the *Image Sampling* panel we can control how the information is retrieved from the image.

Fig. 2.1618: Background image.

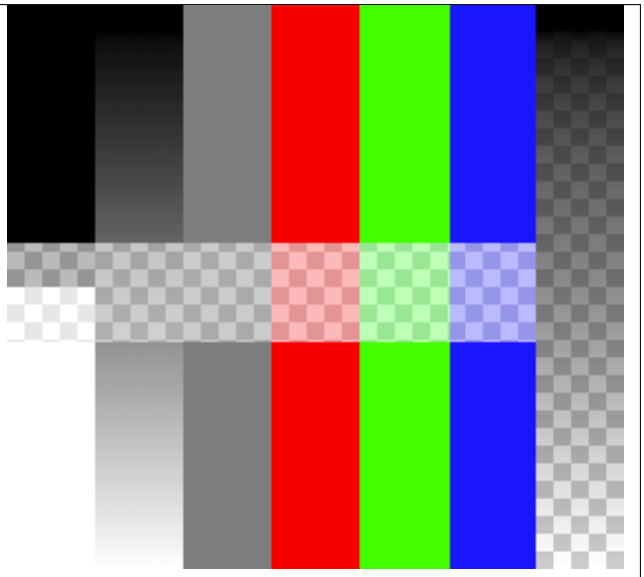


Fig. 2.1619: Foreground image.

The two images presented here are used to demonstrate the different image options. The *background image* is an ordinary JPG-file, the *foreground image* is a PNG-file with various alpha and grayscale values. The vertical bar on the right side of the foreground image is an Alpha blend, the horizontal bar has 50% alpha.

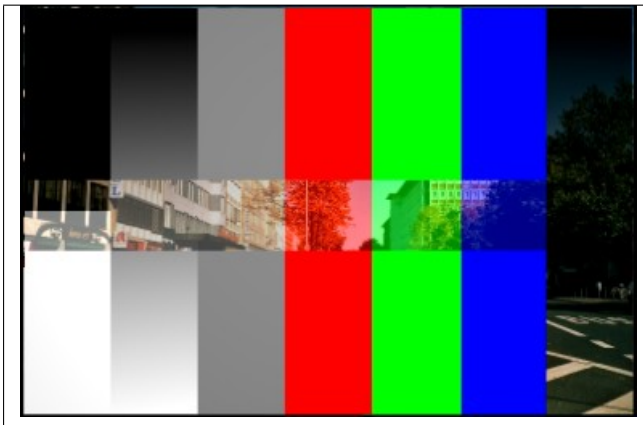


Fig. 2.1620: Foreground image with *Use* alpha. The alpha values of the pixels are evaluated.

Fig. 2.1621: Foreground image with *Calculate* alpha.

Alpha Options related to transparency.

Use Works with PNG and TGA files since they can save transparency information (Foreground Image with Use Alpha). Where the alpha value in the image is less than 1.0, the object will be partially transparent and stuff behind it will show.

Calculate Calculate an alpha based on the RGB values of the Image. Black (0, 0, 0) is transparent, white (1, 1, 1) opaque. Enable this option if the image texture is a mask. Note that mask images can use shades of gray that translate to semi-transparency, like ghosts, flames, and smoke/fog.

Invert Reverses the alpha value. Use this option if the mask image has white where you want it transparent and vice-versa.

Flip X/Y Axis Rotates the image 90 degrees counterclockwise when rendered.

Normal Map This tells Blender that the image is to be used to create the illusion of a bumpy surface, with each of the three RGB channels controlling how to fake a shadow from a surface irregularity. Needs specially prepared input pictures. See *Bump and Normal Maps*.

Normal Map Space:

- *Tangent*

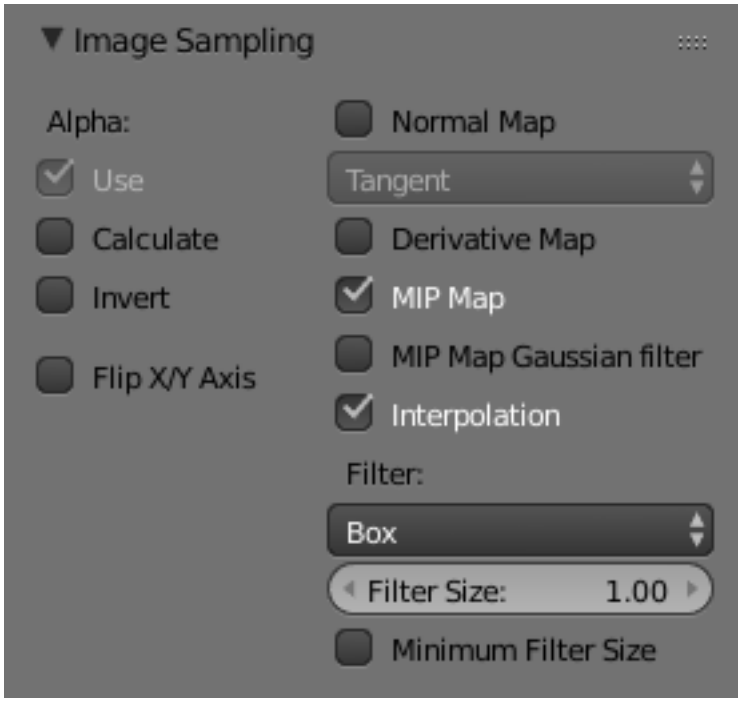


Fig. 2.1622: Image Sampling panel.

- *Object*
- *World*
- *Camera*

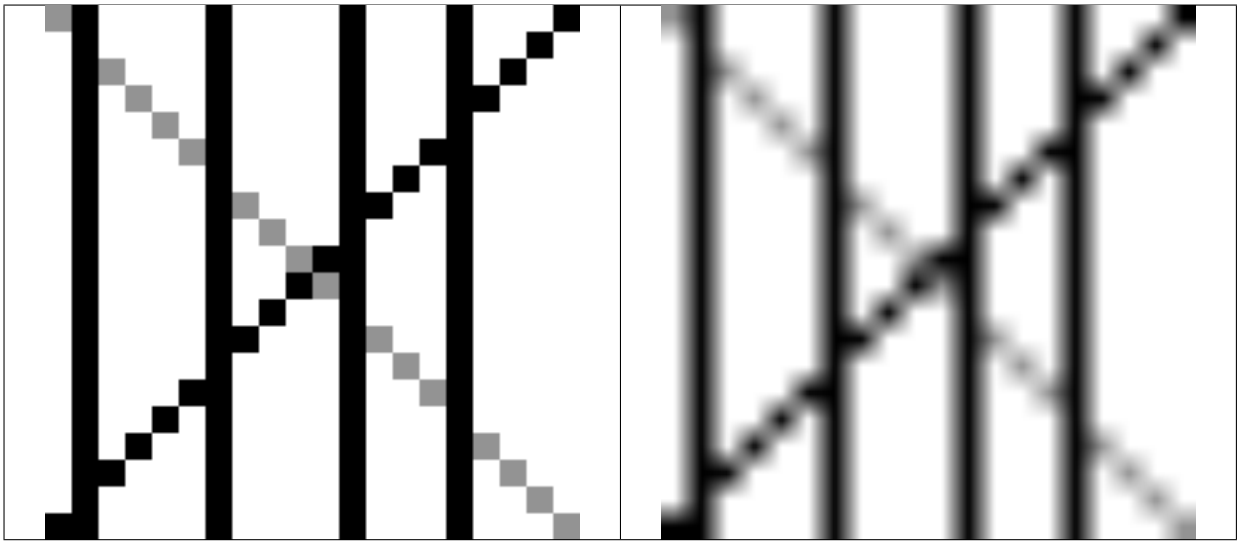
Derivative Map Use red and green as derivative values.

MIP Map MIP Maps are pre-calculated, smaller, filtered Textures for a certain size. A series of pictures is generated, each half the size of the former one. This optimizes the filtering process. By default, this option is enabled and speeds up rendering (especially useful in the *Game Engine*). When this option is OFF, you generally get a sharper image, but this can significantly increase calculation time if the filter dimension (see below) becomes large. Without MIP Maps you may get varying pictures from slightly different camera angles, when the Textures become very small. This would be noticeable in an animation.

MIP Map Gaussian filter Used in conjunction with MIP Map, it enables the MIP Map to be made smaller based on color similarities. In the *Game Engine*, you want your textures, especially your MIP Map textures, to be as small as possible to increase rendering speed and frame rate.

Interpolation This option interpolates the pixels of an image. This becomes visible when you enlarge the picture. By default, this option is on. Turn this option off to keep the individual pixels visible and if they are correctly anti-aliased. This last feature is useful for regular patterns, such as lines and tiles; they remain 'sharp' even when enlarged considerably. When you enlarge this 10×10 pixel Image  the difference with and without *Interpolation* is clearly visible. Turn this image off if you are using digital photos to preserve crispness.

Table 2.81: Enlarged Image texture without and with *Interpolation*



Filter The filter size used in rendering, and also by the options *MipMap* and *Interpolation*. If you notice gray lines or outlines around the textured object, particularly where the image is transparent, turn this value down from 1.0 to 0.1 or so.

Texture Filter Type Texture filter to use for image sampling. Just like a *pixel* represents a *pic* ture *el* ement, a *texel* represents a *tex* ture *el* ement. When a texture (2D texture space) is mapped onto a 3D model (3D model space), different algorithms can be used to compute a value for each pixel based on samplings from several texels.

Box A fast and simple nearest-neighbor interpolation known as Monte Carlo integration

EWA (Elliptical Weighted Average) One of the most efficient direct convolution algorithms developed by Paul Heckbert and Ned Greene in the 1980s. For each texel, EWA samples, weights, and accumulates texels within an elliptical footprint and then divides the result by the sum of the weights.

Eccentricity Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower

FELINE (Fast Elliptical Lines) Uses several isotropic probes at several points along a line in texture space to produce an anisotropic filter to reduce aliasing artifacts without considerably increasing rendering time.

Probes Number of probes to use. An integer between 1 and 256. Further reading: McCormack, J; Farkas, KI; Perry, R; Jouppi, NP (1999) Simple and Table Feline: Fast Elliptical Lines for Anisotropic Texture Mapping, WRL

Area Area filter to use for image sampling.

Eccentricity Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower.

Filter Size The filter size used by MIP Map and Interpolation.

Minimum Filter Size Use Filter Size as a minimal filter value in pixels.

Image Mapping

In the *Image Mapping* panel, we can control how the image is mapped or projected onto the 3D model.

Extension

Extend Outside the image the colors of the edges are extended.

Clip Clip to image size and set exterior pixels as transparent. Outside the image, an alpha value of 0.0 is returned. This allows you to 'paste' a small logo on a large object.

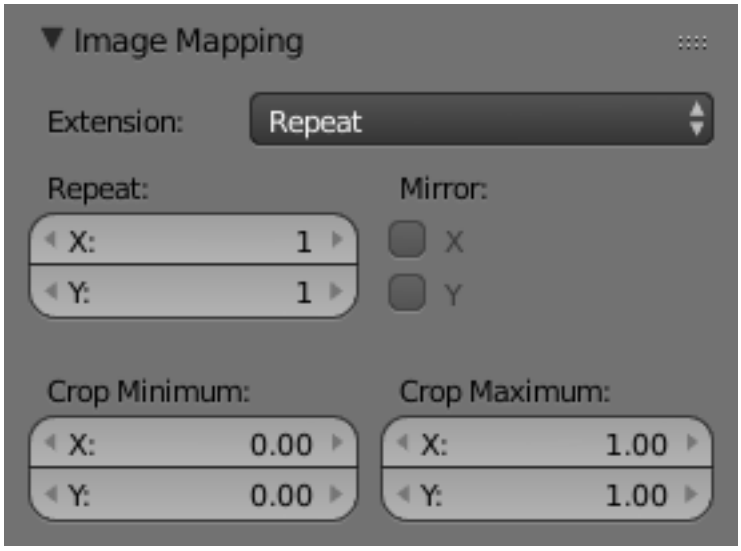


Fig. 2.1623: Image Mapping panel.

Clip Cube Clips to cubic-shaped area around the images and sets exterior pixels as transparent. The same as Clip, but now the 'Z' coordinate is calculated as well. An alpha value of 0.0 is returned outside a cube-shaped area around the image.

Repeat The image is repeated horizontally and vertically.

Repeat X/Y repetition multiplier.

Mirror Mirror on X/Y axes. This buttons allow you to map the texture as a mirror, or automatic flip of the image, in the corresponding X and/or Y direction.

Checker Checkerboards quickly made. You can use the option *size* on the *Mapping* panel as well to create the desired number of checkers.

Even / Odd Set even/odd tiles

Distance Governs the distance between the checkers in parts of the texture size.

Crop Minimum / Crop Maximum The offset and the size of the texture in relation to the texture space. Pixels outside this space are ignored. Use these to crop, or choose a portion of a larger image to use as the texture.

Procedural Textures

Introduction

Procedural textures are textures that are defined mathematically. They are generally relatively simple to use, because they do not need to be mapped in a special way. This does not mean that procedural textures cannot become very complex.

These types of textures are 'real' 3D. By that we mean that they fit together perfectly at the edges and continue to look like what they are meant to look like even when they are cut; as if a block of wood had really been cut in two. Procedural textures are not filtered or anti-aliased. This is hardly ever a problem: the user can easily keep the specified frequencies within acceptable limits.

These are the available types:

- *Blend*
- *Clouds*
- *Distorted Noise*
- *Magic*

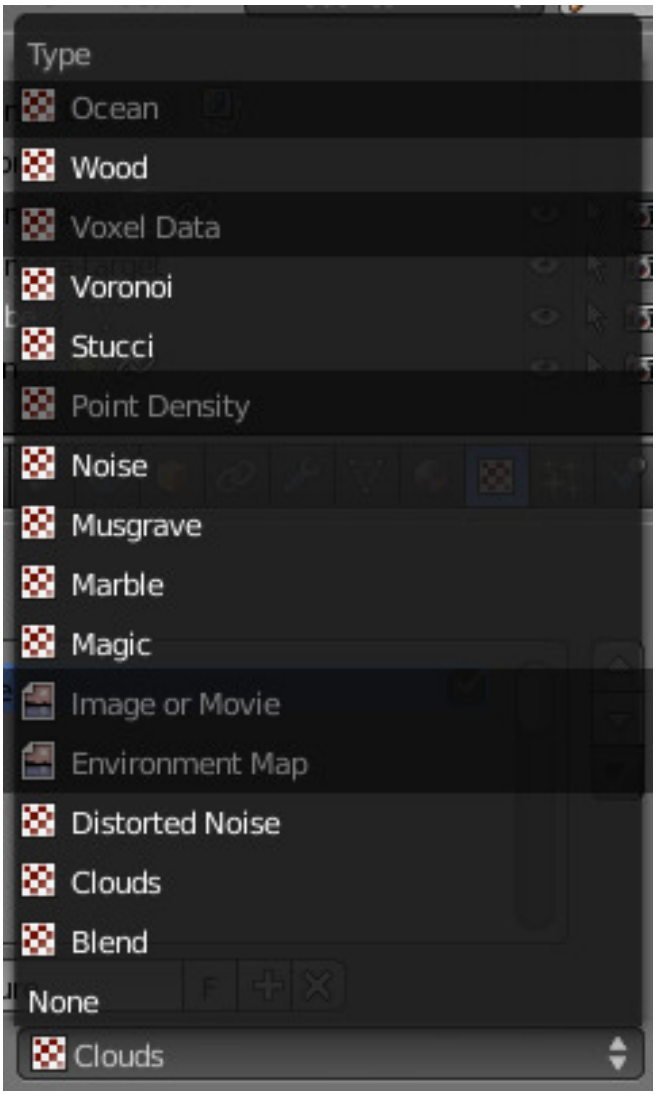


Fig. 2.1624: The Texture Type list in the Texture panel of the Texture Buttons. (Non procedural textures darkened out.).

- *Marble*
- *Musgrave*
- *Noise*
- *Stucci*
- *Voronoi*
- *Wood*

Common options

Noise Basis

Each noise-based Blender texture (with the exception of Voronoi and simple noise) has a *Noise Basis* setting that allows the user to select which algorithm is used to generate the texture. This list includes the original Blender noise algorithm. The *Noise Basis* settings makes the procedural textures extremely flexible (especially *Musgrave*).

The *Noise Basis* governs the structural appearance of the texture:

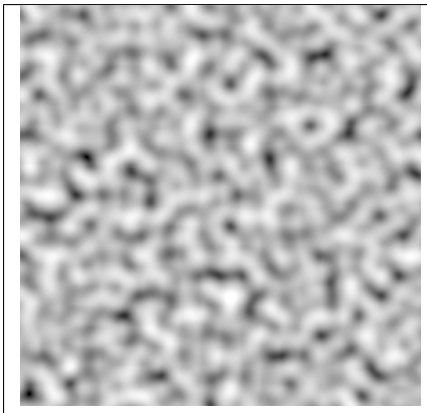


Fig. 2.1625: Blender Original.

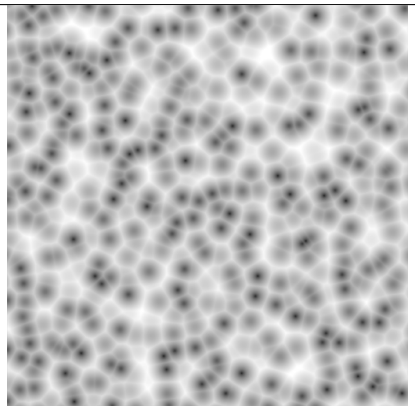


Fig. 2.1626: Voronoi F1.

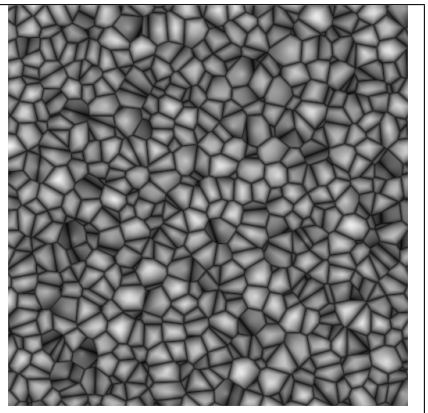


Fig. 2.1627: Voronoi F2-F1.

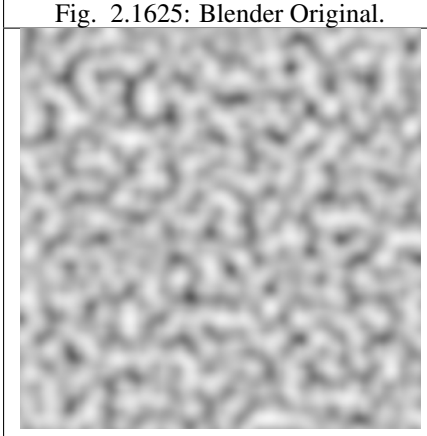


Fig. 2.1628: Original Perlin.

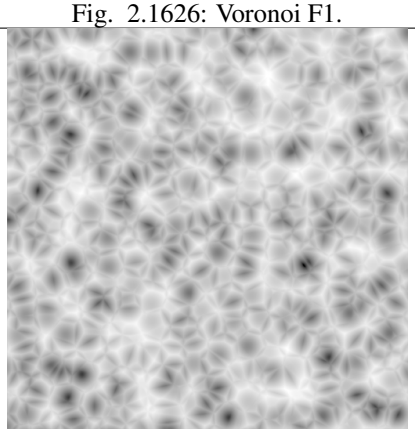


Fig. 2.1629: Voronoi F2.

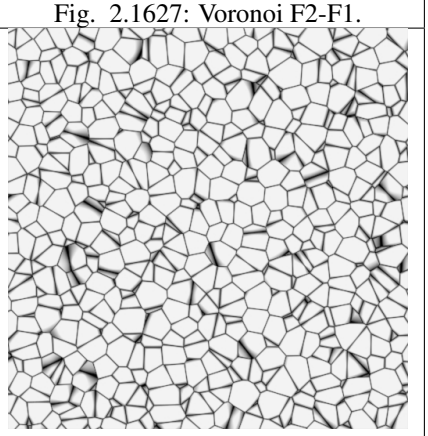


Fig. 2.1630: Voronoi Crackle.

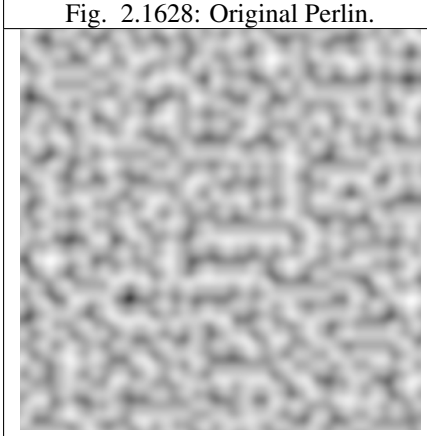


Fig. 2.1631: Improved Perlin.

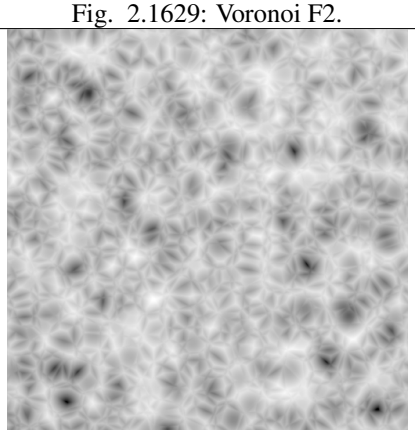


Fig. 2.1632: Voronoi F3.

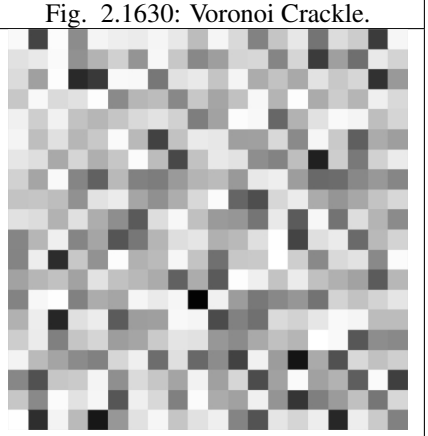


Fig. 2.1633: Cell Noise.

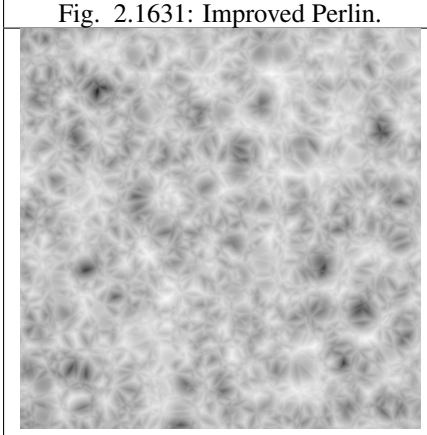


Fig. 2.1634: Voronoi F4.

There are two more possible settings for *Noise Basis*, which are relatively similar to *Blender Original*: Improved Perlin and Original Perlin.

Nabla

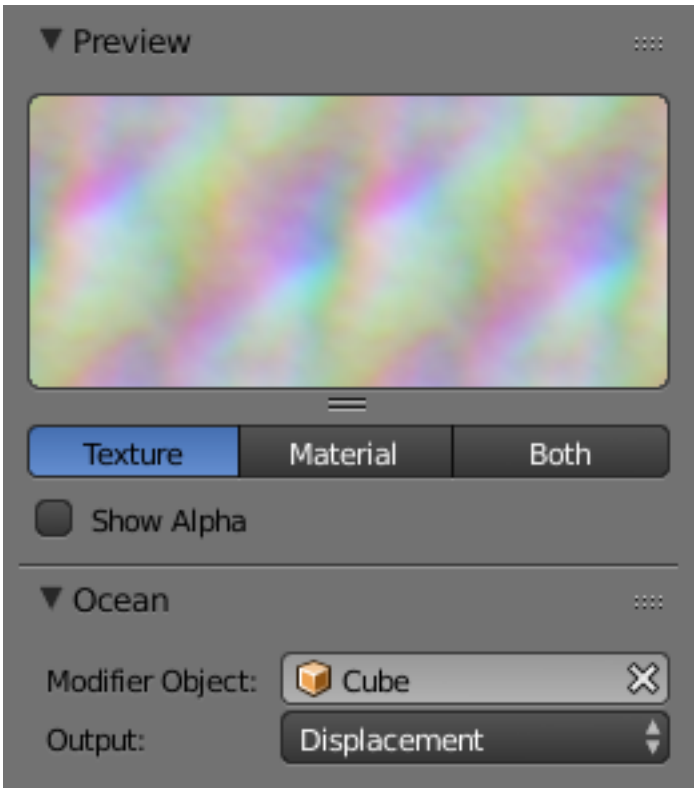
Almost all procedural textures in Blender use derivatives for calculating normals for texture mapping (with as exception *Blend* and *Magic*). This is important for Normal and Displacment Maps. The strength of the effect is controlled with the *Nabla* Number Button.

Hints

Use the size buttons in the *Mapping* panel to set the size that the procedural textures are mapped to.

Procedural textures can either produce colored textures, intensity only textures, textures with alpha values and normal textures. If intensity only ones are used the result is a black and white texture, which can be greatly enhanced by the use of ramps. If on the other hand you use ramps and need an intensity value, you have to switch on *No RGB* in the *Mapping* panel.

Ocean Texture



Texture generated by an *Ocean Modifier*.

Options

Modifier Object Object containing the ocean modifier.

Output The data that is output by the texture.

Displacement Output XYZ displacement in RGB channels.

Foam Output foam (wave overlap) amount in single channel.

Eigenvalues Positive Eigenvalues.

Eigenvector (-) Negative Eigenvectors.

Eigenvector (+) Positive Eigenvectors.

Blend

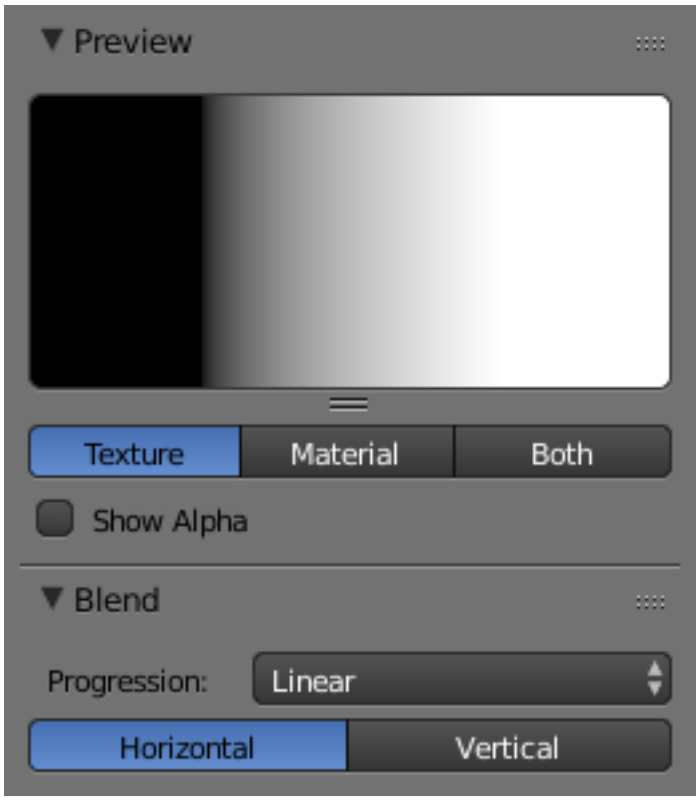


Fig. 2.1635: Blend Texture Panels.

Often used for This is one of the most frequently used procedural textures. You can use blend textures to blend other textures together (with *Stencil*), or to create nice effects (especially with the *Mapping: Normal* trick).

Note: Remember that if you use a ramp to create a custom blending, you may have to use *No RGB*, if the *Mapping* value needs an intensity input.

Result(s) Intensity. The Blend texture generates a smoothly interpolated progression.

Options

Progression Profile of blend.

Linear A linear progression.

Quadratic A quadratic progression.

Easing A flowing, non-linear progression.

Diagonal A diagonal progression.

Spherical A progression with the shape of a three-dimensional ball.

Quadratic Sphere A quadratic progression with the shape of a three-dimensional ball.

Radial A radial progression: *Horizontal* / *Vertical*. The direction of the progression is flipped a quarter turn.

Clouds

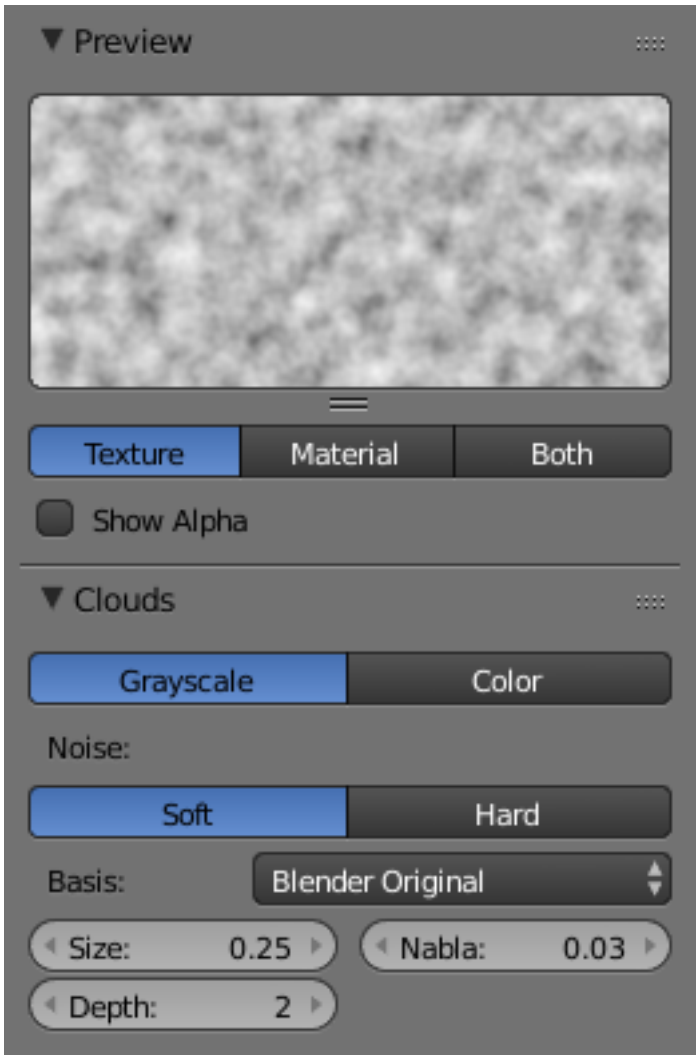


Fig. 2.1636: Clouds Texture Panels.

Clouds represent Perlin noise. In addition, each noise-based Blender texture (with the exception of Voronoi and simple noise) has a "Noise Basis" setting that allows the user to select which algorithm is used to generate the texture.

Often used for Clouds, Fire, Smoke. Well-suited to be used as a Bump map, giving an overall irregularity to the material.

Result(s) *Greyscale* (default) or RGB *Color*

Options

Greyscale The standard noise, gives an intensity

Color The noise gives an RGB value

Noise *Soft* or *Hard*, changes contrast and sharpness

Size The dimension of the Noise table

Depth The depth of the *Clouds* calculation. A higher number results in a long calculation time, but also in finer details.

Distorted Noise

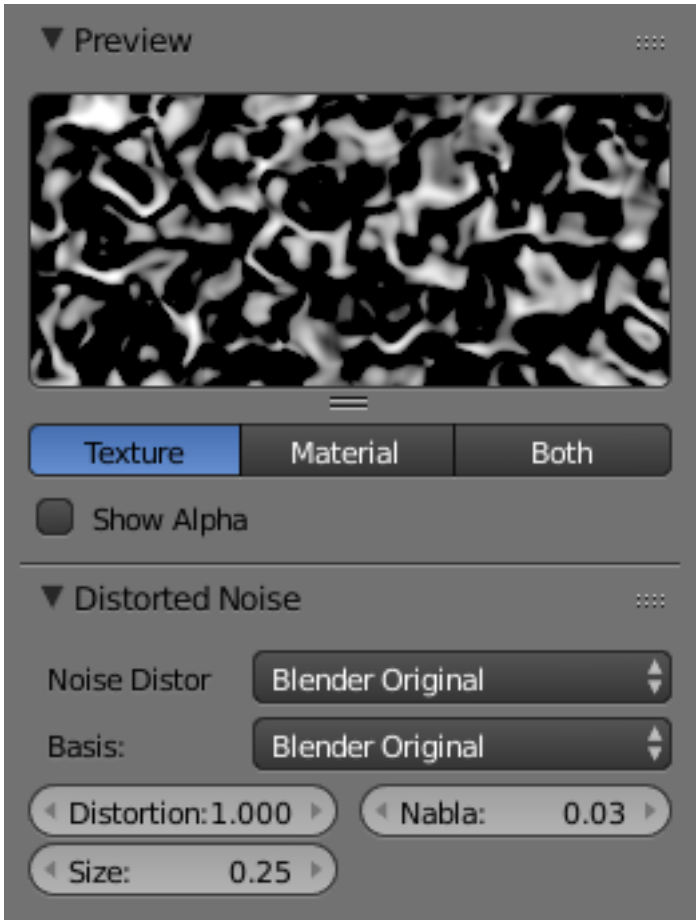


Fig. 2.1637: Distorted Noise Texture Panels.

Distortion Noise takes the option that you pick from *Noise Basis* and filters it, to create hybrid pattern.

Often used for Grunge, very complex and versatile.

Result(s) Intensity.

Options

Noise Distortion The texture to use to distort another.

Basis The texture to be distorted.

Noise The size of the noise generated.

Distortion The amount that *Distortion Noise* affects *Basis*.

Magic

Often used for Not frequently used. It can be used for "Thin Film Interference", if you set *Mapping* to *Reflection* and use a relatively high *Turbulence*.

Result(s) RGB color. The RGB components are generated independently with a sine formula.

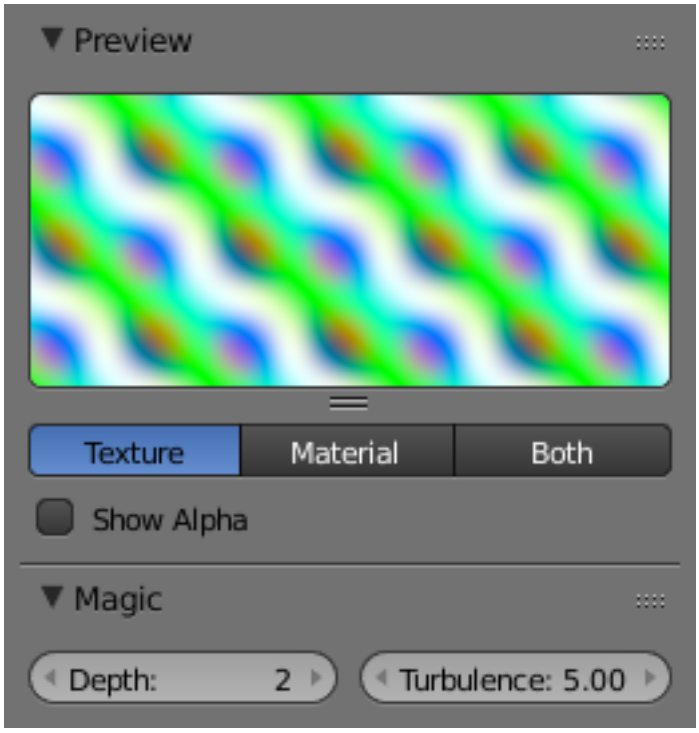


Fig. 2.1638: Magic Texture Panels.

Options

Depth The depth of the calculation. A higher number results in a long calculation time, but also in finer details.

Turbulence The strength of the pattern.

Marble

Often used for Marble, Fire, Noise with a structure.

Result(s) Intensity value only.

Bands are generated based on the sine, saw, or triangular formula and noise turbulence.

Options

Marble Type Three settings for soft to more clearly defined *Marble*.

Soft, Sharp, Sharper

Noise basis Shape of wave to produce bands.

Sine, Saw, Triangle

Noise Type The noise function works with two methods.

Soft, Hard

Size The dimensions of the noise table

Depth The depth of the *Marble* calculation. A higher value results in greater calculation time, but also in finer details.

Turbulence The turbulence of the sine bands.

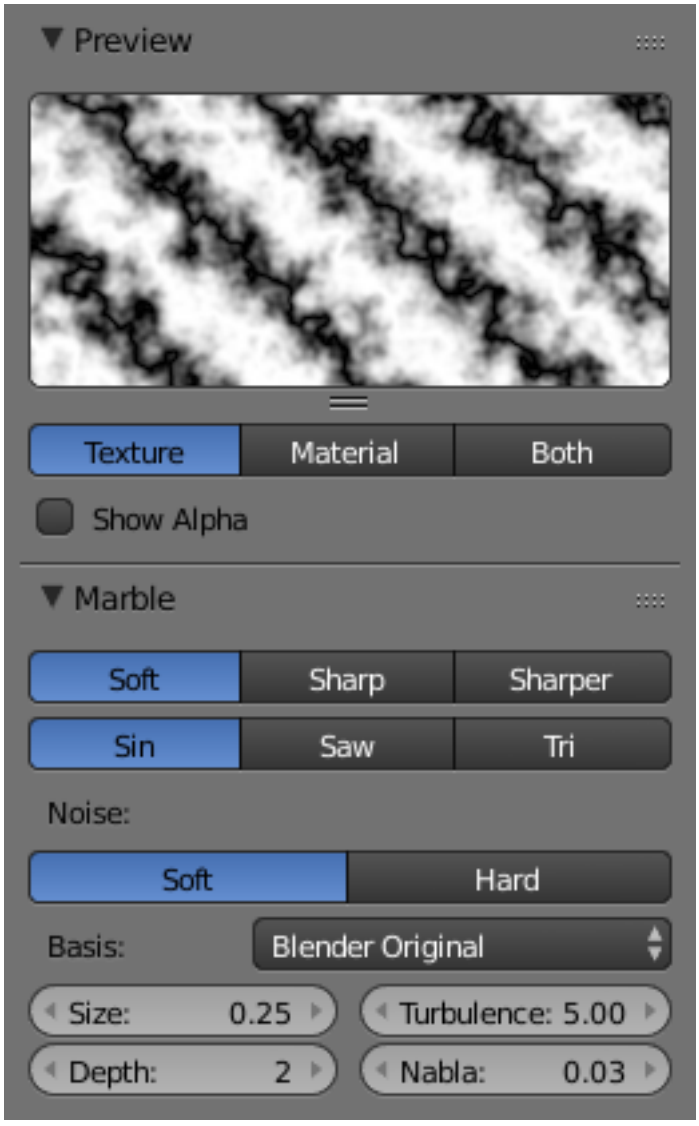


Fig. 2.1639: Marble Texture Panels.

Musgrave

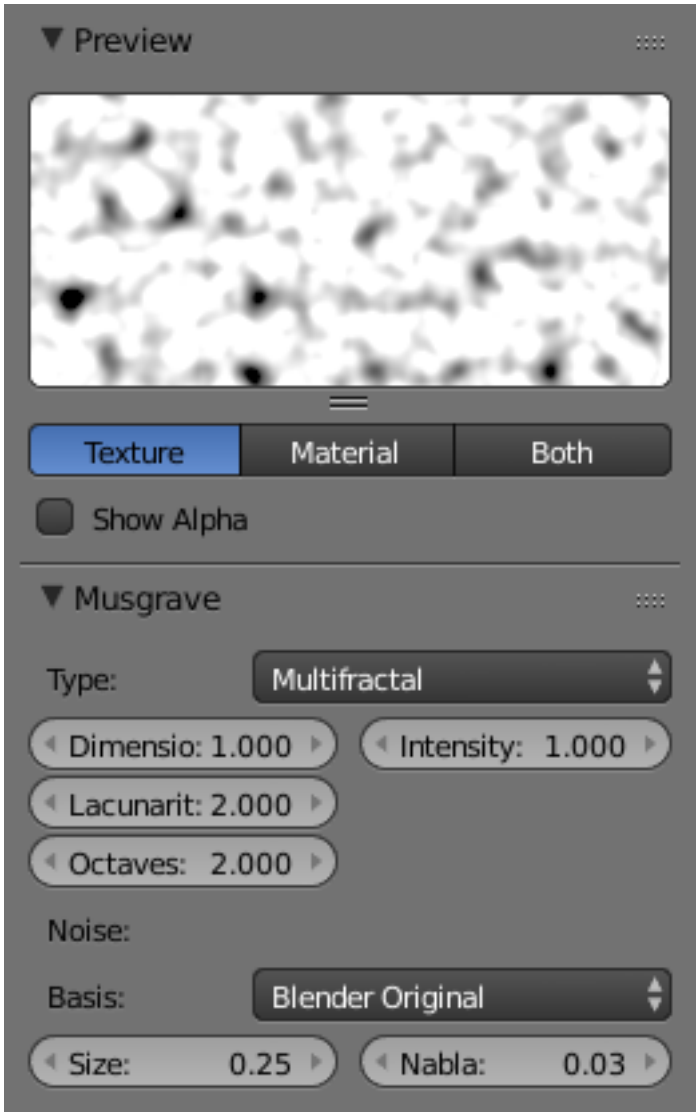


Fig. 2.1640: Musgrave Texture Panels.

Often used for Organic materials, but it is very flexible. You can do nearly everything with it.

Result(s) Intensity.

Options

Type This procedural texture has five noise types on which the resulting pattern can be based and they are selectable from a select menu at the top of the tab. The five types are:

- Hetero Terrain
- Fractal Brownian Motion (fBm)
- Hybrid Multifractal
- Ridged Multifractal
- Multifractal

These noise types determine the manner in which Blender layers successive copies of the same pattern on top of each other at varying contrasts and scales.

Examples with Basis: Voronoi: F1, Dimension: 0.5, Lacunarity: 0.15, Octave: 2.0 .

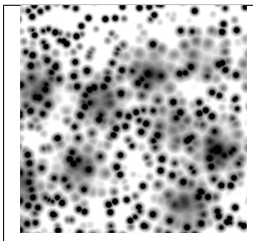


Fig. 2.1641: Hetero Terrain.

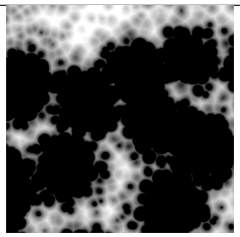


Fig. 2.1642: Fractal Brownian Motion.

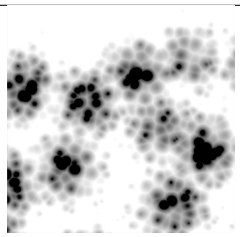


Fig. 2.1643: Hybrid Multifractal.

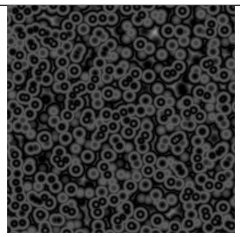


Fig. 2.1644: Ridged Multifractal.

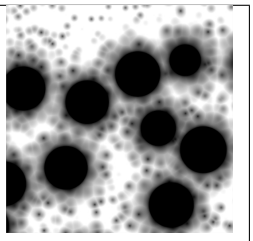


Fig. 2.1645: Multi-fractal.

The main noise types have four characteristics:

Dimension Fractal dimension controls the contrast of a layer relative to the previous layer in the texture. The higher the fractal dimension, the higher the contrast between each layer, and thus the more detail shows in the texture.

Lacunarity Lacunarity controls the scaling of each layer of the Musgrave texture, meaning that each additional layer will have a scale that is the inverse of the value which shows on the button. i.e. Lacunarity = 2 –> Scale = 1/2 original.

Octaves Octave controls the number of times the original noise pattern is overlayed on itself and scaled/contrasted with the fractal dimension and lacunarity settings.

Intensity Light intensity. Called *Offset* for *Hetero Terrain*.

The *Hybrid Multifractal* and *Ridged Multifractal* types have these additional settings:

Offset Both have a "Fractal Offset" button that serves as a "sea level" adjustment and indicates the base height of the resulting bump map. Bump values below this threshold will be returned as zero.

Gain Setting which determines the range of values created by the function. The higher the number, the greater the range. This is a fast way to bring out additional details in a texture where extremes are normally clipped off.

Noise

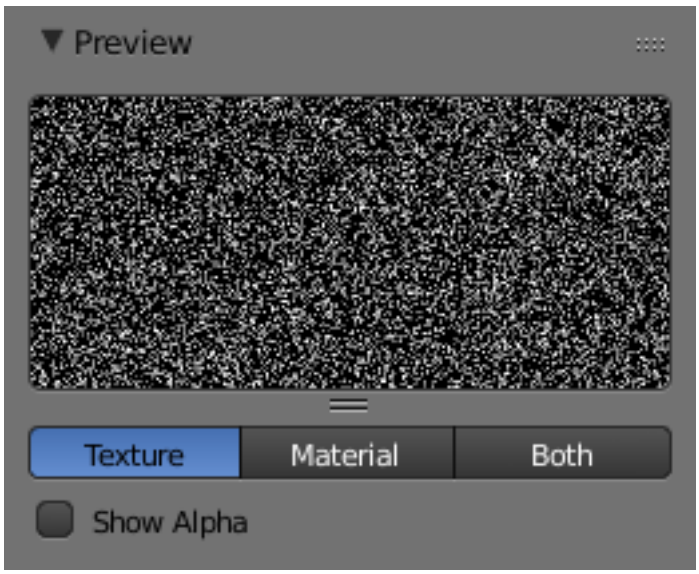


Fig. 2.1646: Noise Texture Panel.

Although this looks great, it is not Perlin Noise! This is a true, randomly generated Noise. This gives a different result every time, for every frame, for every pixel.

Options

There are no options for this noise.

Often used for White noise in an animation. This is not well suited if you do not want an animation. For material displacement or bump, use clouds instead.

Result(s) Intensity.

Stucci

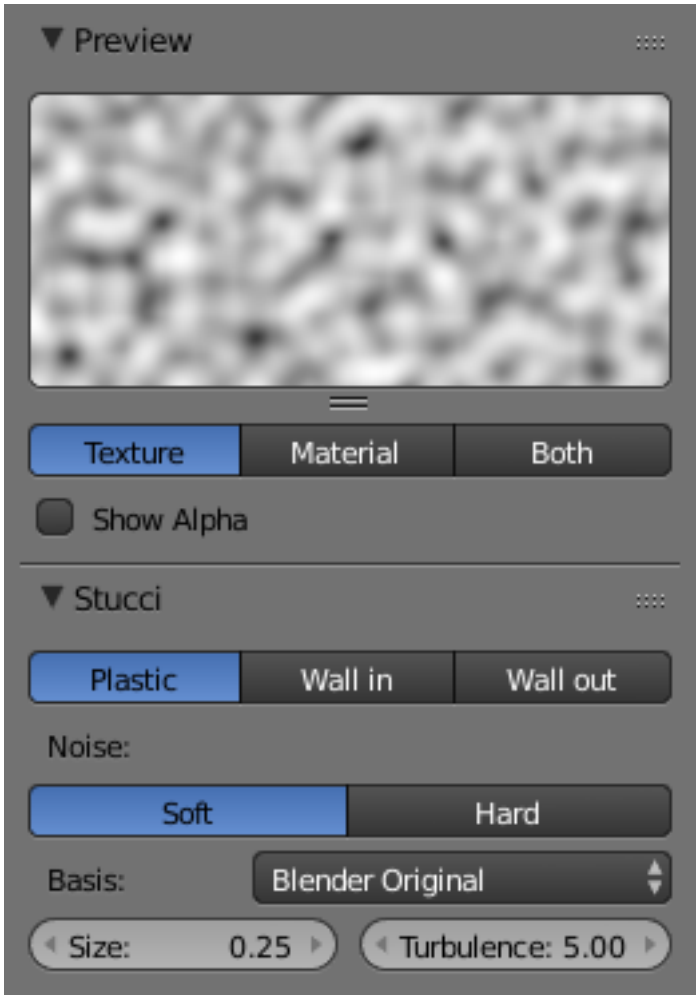


Fig. 2.1647: Stucci Texture Panels.

The *Stucci* texture is based on noise functions.

Often used for Stone, Asphalt, Oranges. Normally for Bump-Mapping to create grainy surfaces.

Result(s) Normals and Intensity.

Options

Plastic / Wall In / Wall out Plastic is the standard Stucci, while the "walls" is where Stucci gets it name. This is a typical wall structure with holes or bumps.

Soft / Hard There are two methods available for working with Noise.

Size Dimension of the Noise table.

Turbulence Depth of the *Stucci* calculations.

Voronoi

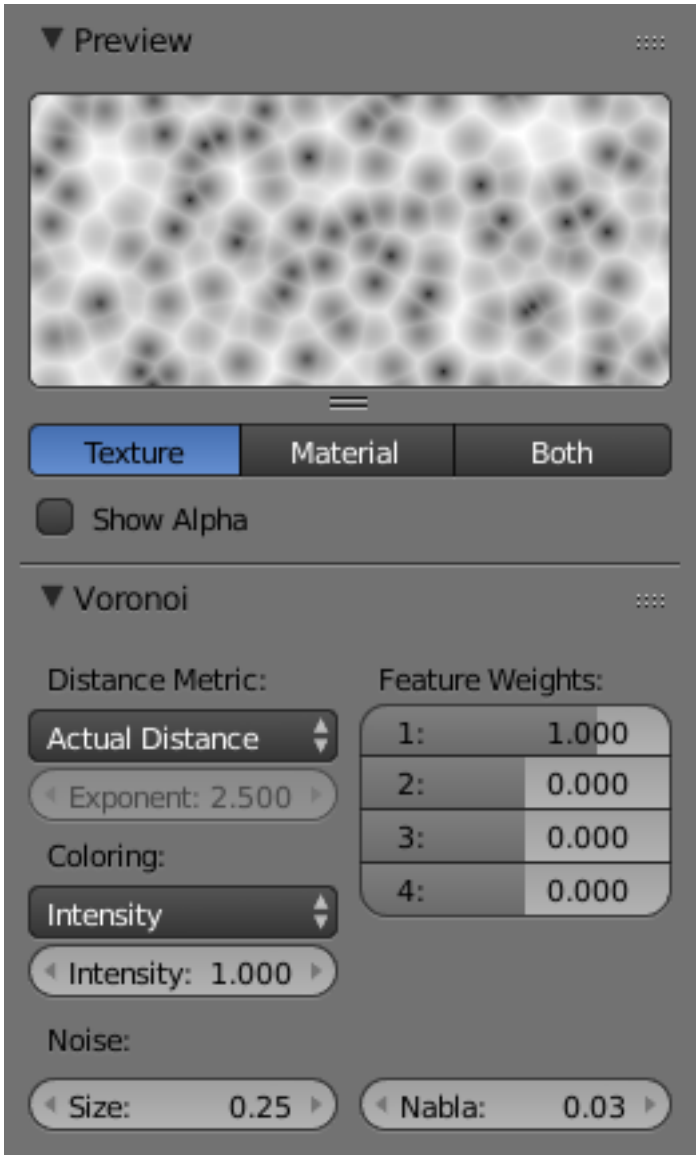


Fig. 2.1648: Voronoi Texture Panels.

Often used for Very convincing Metal, especially the "Hammered" effect. Organic shaders (e.g. scales, veins in skin).

Result(s) Intensity (default) and Color.

Options

Distance Metric This procedural texture has seven Distance Metric options. These determine the algorithm to find the distance between cells of the texture. These options are:

- Minkovsky
- Minkovsky 4

- Minkovsky 1/2
- Chebychev
- Manhattan
- Distance Squared
- Actual Distance

The *Minkovsky* setting has a user definable value (the *Exponent* button) which determines the Minkovsky exponent *e* of the distance function:

$(x^e + y^e + z^e)^{1/e}$

A value of one produces the *Manhattan* distance metric, a value less than one produces stars (at 0.5, it gives a *Minkovsky 1/2*), and higher values produce square cells (at 4.0, it gives a *Minkovsky 4*, at 10.0, a *Chebychev*). So nearly all Distance Settings are basically the same – a variation of *Minkowsky*.

You can get irregularly-shaped rounded cells with the *Actual Distance* / *Distance Squared* options.

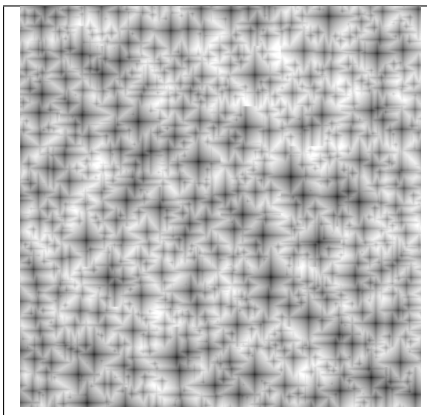


Fig. 2.1649: Minkovsky Exponent: 0.5 (Minkovsky 1/2).

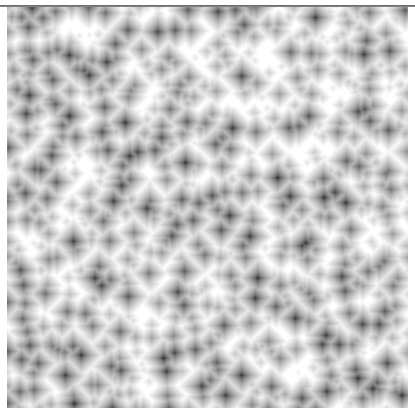


Fig. 2.1650: Minkovsky Exponent: 1 (Manhattan).

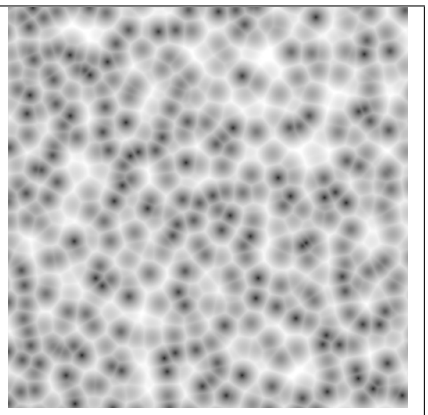


Fig. 2.1651: Minkovsky Exponent: 2 (Actual Distance).

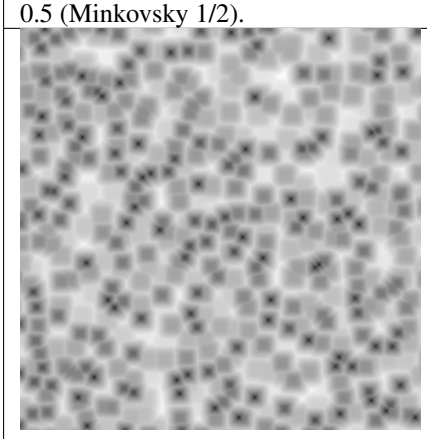


Fig. 2.1652: Minkovsky Exponent: 4 (Minkovsky 4).

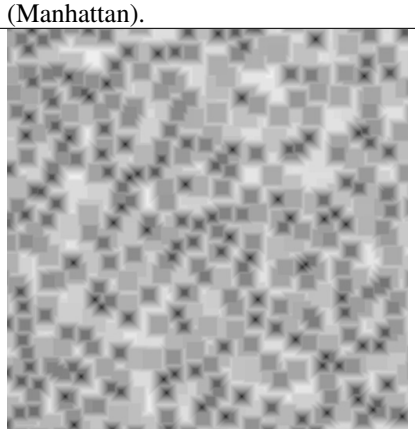


Fig. 2.1653: Minkovsky Exponent: 10 (Chebychev).

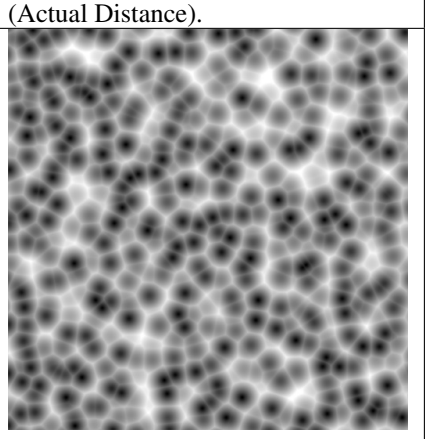


Fig. 2.1654: Distance Squared (More contrast than Actual Distance).

Feature Weights These four sliders at the bottom of the Voronoi panel represent the values of the four Worley constants, which are used to calculate the distances between each cell in the texture based on the distance metric. Adjusting these values can have some interesting effects on the end result...

Coloring Four settings (*Intensity*, *Position*, *Position and Outline*, and *Position, Outline, and Intensity*) that can use four different noise basis as methods to calculate color and intensity of the texture output. This gives the Voronoi texture you create with the "Worley Sliders" a completely different appearance and is the equivalent of the noise basis setting found on the other textures.

Wood

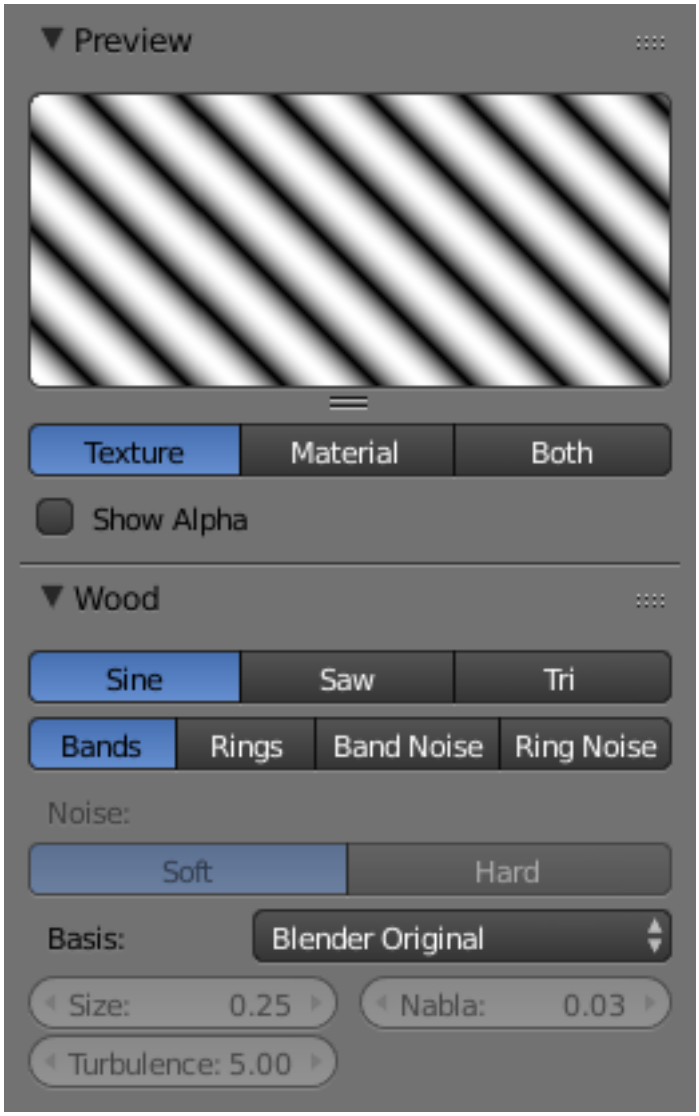


Fig. 2.1655: Wood Texture Panels.

Often used for Woods and ring-shaped patterns.

Result(s) Intensity only.

Options

Noise Basis Shape of wave to produce bands

Sine, Saw, Triangle

Wood Type Set the bands to either straight or ring-shaped, with or without turbulence.

Bands, Rings, Band Noise, Ring Noise

Noise Type There are two methods available for the Noise function

Soft, Hard

Size Dimension of the Noise table

Turbulence Turbulence of the Band Noise and Ring Noise types

Environment Maps

Environment maps take a render of the 3D scene and apply it to a texture, to use for faking reflections. If you want to achieve a very realistic result, raytraced reflections are a good solution. Environment Maps are another way to create reflective surfaces, but they are not so simple to set up.

So why should one use Environment Maps?

- The main reason is probably that they can be much faster than raytracing reflections. In certain situations they need to be calculated only once, and may be reused like any ordinary texture. You may even modify the precalculated Environment Map in an image editor.
- Environment maps can also be blurred and render even faster because the resolution can then be lowered. Blurring a reflection with the raytracer always adds to the render time, sometimes quite a lot.
- *Halos* (a visualization type for particles) are not visible to raytraced reflections, so you need to setup environment maps to reflect them.
- *Keypoint strands* (another visualization type for particles) are also not visible to raytraced reflections, so you need to setup environment maps to reflect them.

Just as we render the light that reaches the viewing plane using the camera to define a viewpoint, we can render the light that reaches the surface of an object (and hence, the light that might ultimately be reflected to the camera). Blender's environment mapping renders a cubic image map of the scene in the six cardinal directions from any point. When the six tiles of the image are mapped onto an object using the *Reflection* input coordinates, they create the visual complexity that the eye expects to see from shiny reflections.

Note: It is useful to remember here that the true goal of this technique is *believability*, not *accuracy*. The eye does not need a physically accurate simulation of the light's travel; it just needs to be lulled into believing that the scene is real by seeing the complexity it expects. The most unbelievable thing about most rendered images is the sterility, not the inaccuracy.

Options

Important: For correct results, the mapping of an environment map texture must be set to *Reflection* (reflection coordinates) in the Map Input panel of the Material tab.

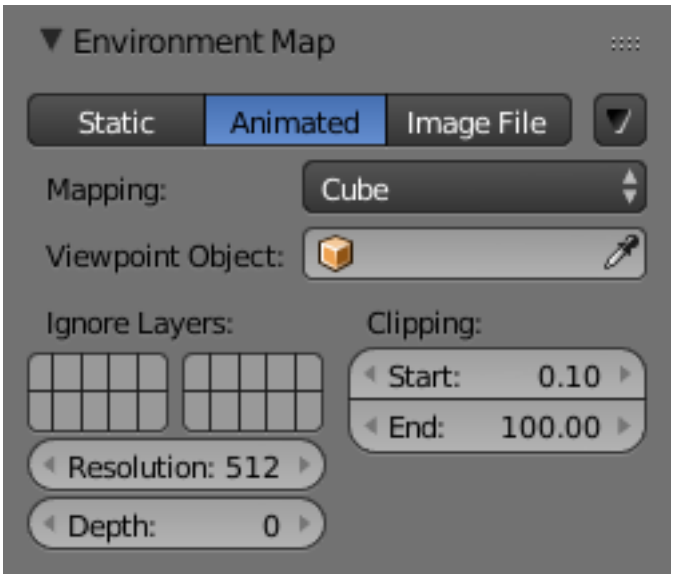


Fig. 2.1656: Reflecting plane Environment Map settings.

Blender allows three types of environment maps, as you can see in Fig. *Reflecting plane Environment Map settings.*:

Static The map is only calculated once during an animation or after loading a file.

Animated The map is calculated each time a rendering takes place. This means moving Objects are displayed correctly in mirroring surfaces.

Image File When saved as an image file, environment maps can be loaded from disk. This option allows the fastest rendering with environment maps, and also gives the ability to modify or use the environment map in an external application.

When using planar reflections, if the camera is the only moving object and you have a reflecting plane, the Empty must move too and you must use *Animated* environment map. If the reflecting object is small and the Empty is in its center, the environment map can be *Static*, even if the object itself rotates since the Empty does not move. If, on the other hand, the Object translates the Empty should follow it and the environment map be of *Animated* type.

Specials

Clear Environment Map Clears the currently rendered environment map from memory. This is useful to refresh a *Static* environment maps and you have changed things in your scene since the last time the environment map was rendered. *Animated* environment maps do this automatically on every render.

Save Environment Map Saves the currently stored static environment map to disk as an image file. This can be loaded again with *Load*.

Clear All Environment Maps Does the same as *Free Data*, but with all environment maps in the scene. This is a useful shortcut when using recursive environment maps (when the *Depth* is greater than 0).

Note: Environment Map calculation can be disabled at a global level by the *Environment Map* Tog Button in the Render Panel of the Rendering Buttons.

Viewpoint Object Environment maps are created from the perspective of a specified object. The location of this object will determine how 'correct' the reflection looks, though different locations are needed for different reflecting surfaces. Usually, an Empty is used as this object:

- For planar reflections, the object should be in a location mirrored from the camera, on the other side of the plane of reflection (see Examples). This is the most accurate usage of Environment maps.
- For spherical reflections, the object should be in the center of the sphere. Generally, if the reflecting sphere's object center point is in the center of its vertices, you can just use the name of the actual sphere object as the *Viewpoint Object*
- For irregular reflections, there is no hard and fast rule, you will probably need to experiment and hope that the inaccuracy does not matter.

Ignore Layers The layers to exclude from the environment map creation. Since environment maps work by rendering the scene from the location of the *Viewpoint Object*, you will need to exclude the actual reflecting surface from the environment map, otherwise it will occlude other objects that should be reflected on the surface itself.

Eg. If you are rendering an environment map from the center of a sphere, all the environment map will show by default is the inside of the sphere. You will need to move the sphere to a separate layer, then exclude that layer from the environment map render, so that the environment map will show (and hence reflect) all the objects outside the sphere.

Resolution The resolution of the cubic environment map render. Higher resolutions will give a sharper texture (reflection), but will be slower to render.

Depth The number of recursive environment map renders. If there are multiple reflecting objects using environment maps in the scene, some may appear solid, as they will not render each other's reflections. In order to show reflections within reflections, the environment maps need to be made multiple times, recursively, so that the effects of one environment map can be seen in another environment map. See Examples.

Clipping Start/End The clipping boundaries of the virtual camera when rendering the environment map. Sets the minimum and maximum distance from the camera that will be visible in the map.

Environment Map Sampling

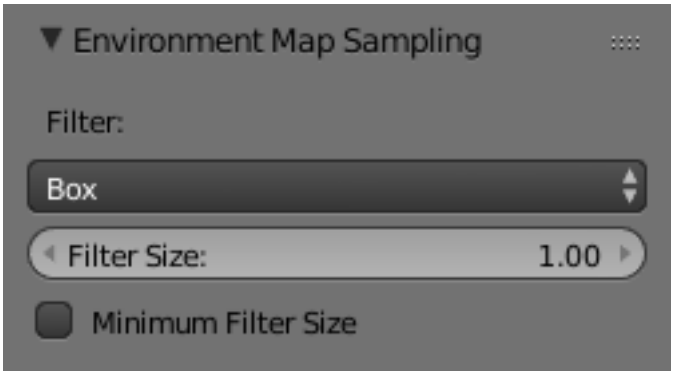


Fig. 2.1657: Environment Map Sampling.

Filter

Box Box Filter

EWA Elliptical Weighted Average. One of the most efficient direct convolution algorithms developed by Paul Heckbert and Ned Greene in the 1980s. For each texel, EWA samples, weights, and accumulates texels within an elliptical footprint and then divides the result by the sum of the weights.

Eccentricity Maximum eccentricity (higher gives less blur at distant/oblique angles, but is also slower)

FELINE FELINE (Fast Elliptical Lines), uses several isotropic probes at several points along a line in texture space to produce an anisotropic filter to reduce aliasing artifacts without considerably increasing rendering time.

Probes Maximum number of samples (higher gives less blur at distant/oblique angles, but is also slower)

Area

Eccentricity Maximum eccentricity (higher gives less blur at distant/oblique angles, but is also slower)

Filter Size The amount of blurring applied to the texture. Higher values will blur the environment map to fake blurry reflections.

Minimum Filter Size Use Filter Size as a minimal filter value in pixels.

Examples

In this example, an empty is used as the *Viewpoint Object* of the reflecting plane's environment map. It is located in the specular position of the camera with respect to the reflecting surface. (This is possible, strictly speaking, only for planar reflecting surfaces.) Ideally, the location of the empty would mirror the location of the camera across the plane of the polygon onto which it is being mapped.

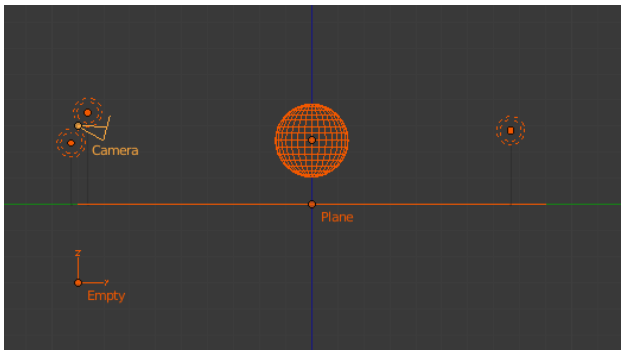


Fig. 2.1658: Planar reflection example.

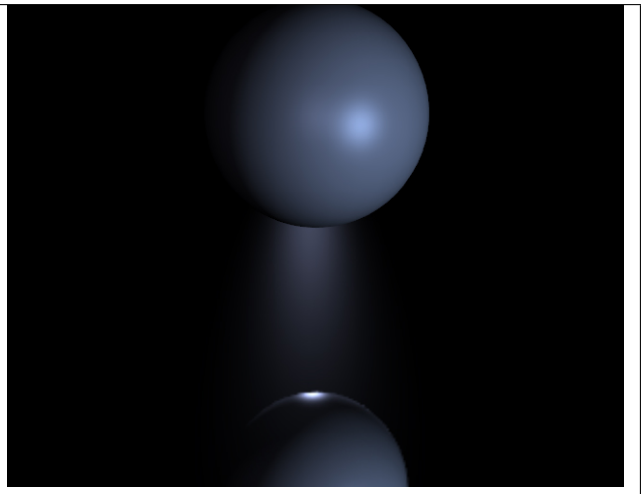


Fig. 2.1659: Sphere on a reflecting surface.

The following images show the effect of the *Depth*. The first render has depth set to 0. This means the environment map on the plane has rendered before the environment map of the sphere, so the sphere's reflection is not shown. By raising the *Depth*, the environment map is rendered recursively, in order to get reflections of reflections.

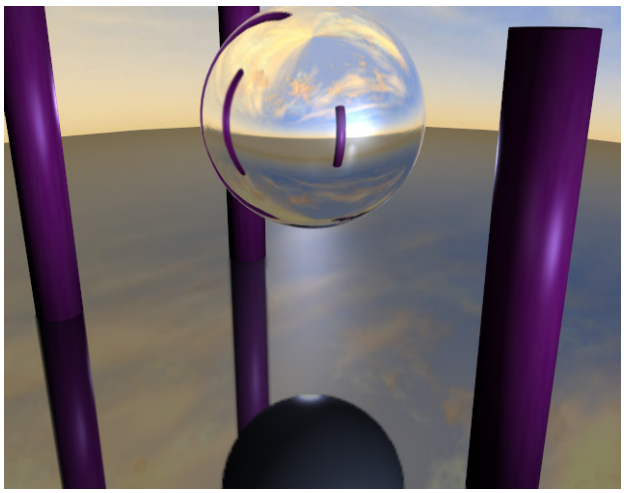


Fig. 2.1660: Reflecting sphere on a reflecting surface.

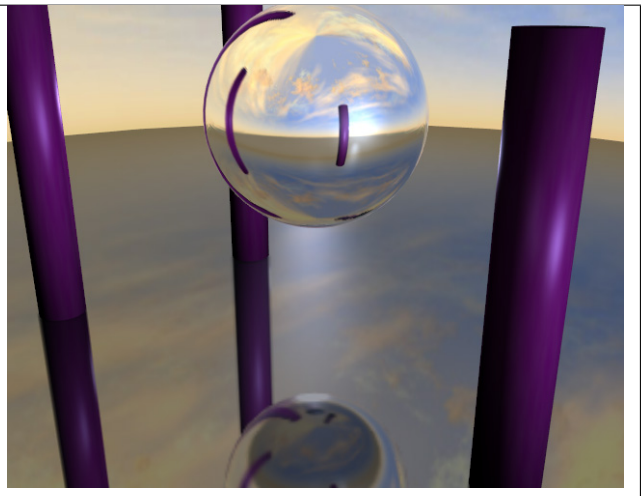


Fig. 2.1661: Reflecting sphere on a reflecting surface with multiple reflections.

Limitations

Because environment maps are calculated from the exact location of the *Viewpoint Object's* object center, and not from actual reflecting surface, they can often be inaccurate, especially with spheres. In the following image, the rectangular prism and the smaller spheres are touching the sides of the large reflecting sphere, but because the environment map is calculated from the center of the sphere, the surrounding objects look artificially far away.

Volumetric Textures

Voxel Data

Voxel data renders a voxel source, working very similarly to an image texture, but in 3D. Various input data source types are available (such as smoke voxel data, or external files), as well as various interpolation methods.

The default voxel data source, Smoke, is used for rendering smoke simulations. Other sources include binary raw formats, and Image Sequence, which can be used to stack a sequence of images into a 3D representation, which is a common format

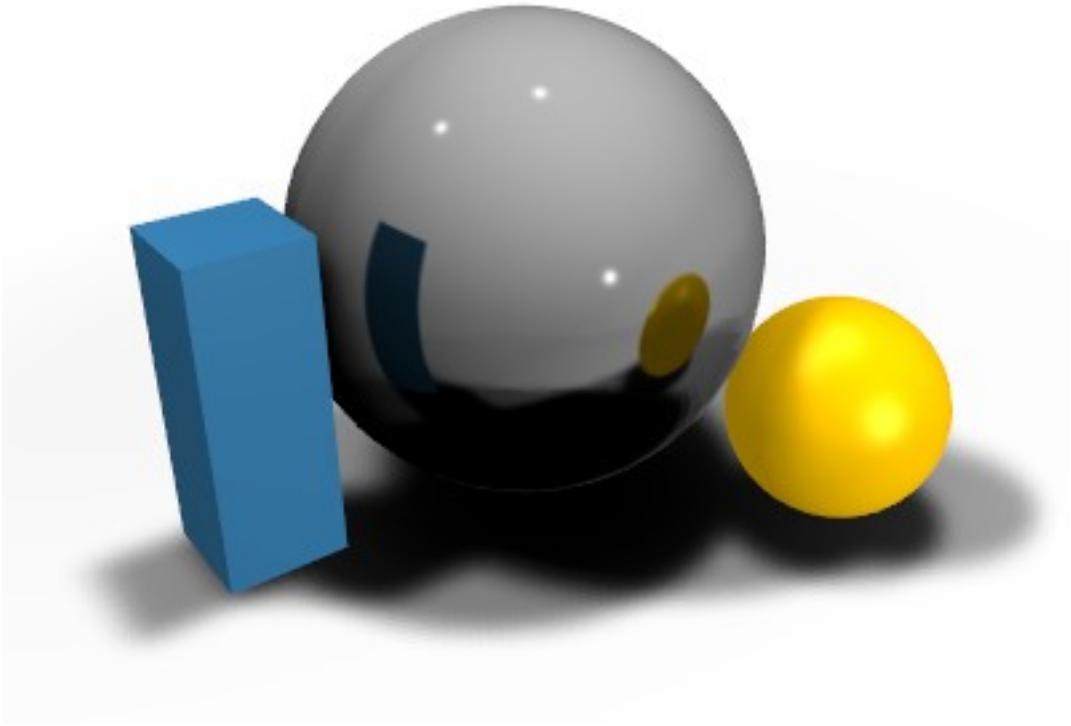


Fig. 2.1662: Inaccurate spherical reflection, the colored objects are artificially offset.

for medical volume data such as CT scans.

Settings

File Format

Blender Voxel Default binary voxel file format.

8 bit RAW 8 bit grayscale binary data.

Image Sequence Generate voxels from a sequence of image slices.

Smoke Render voxels from a Blender smoke simulation.

Source Path The external source data file to use for 8 bit Raw data and Blender Voxel formats.

Domain Object (Smoke) Object used as the smoke simulation domain.

Source

Smoke Use smoke density and color as texture data.

Flame Use flame temperature as texture data.

Heat Use smoke heat as texture data. Values from -2.0 to 2.0 are used.

Velocity Use smoke velocity as texture data.

Resolution Resolution of the voxel grid when using 8 bit Raw data.

Interpolation

Nearest Neighbor No interpolation, fast but blocky and low quality.

Linear Good smoothness and speed.

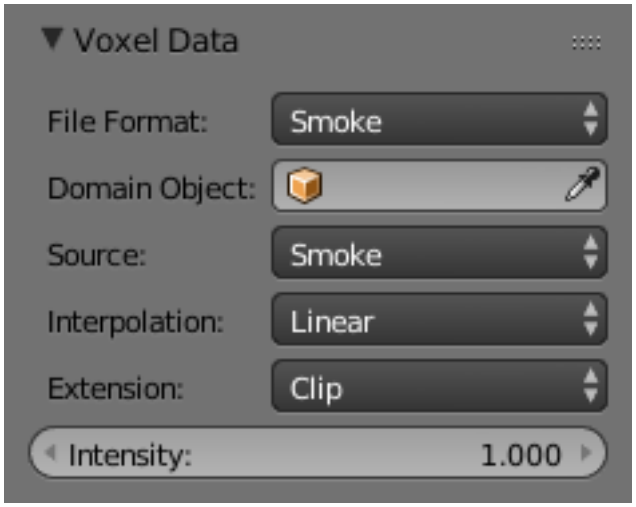


Fig. 2.1663: Voxel Data panel.

Quadratic Mid-range quality and speed.

Cubic Catmull-Rom Smoothed high quality interpolation, but slower.

Extension

Extend Extend by repeating edge pixels of the image.

Clip Clip to image size and set exterior pixels as transparent.

Repeat Cause the image to repeat horizontally and vertically.

Intensity Multiplier for intensity values.

Point Density Texture

Point density renders a given point cloud (object vertices or particle system) as a 3D volume, using a user-defined radius for the points. Internally, the system uses a BVH data structure for fast range lookups.

The rendered points are spherical by default, with various smooth falloff options, as well as simple Turbulence options for displacing the result with noise, adding fine detail. When using Point Density with a particle system, additional particle info such as particle velocity, age, and speed, can be visualized using a color/alpha ramp gradient.

Options

Point Data

Particle System Particle System, Generate point density from a particle system.

Object Vertices Object Vertices, Generate point density from an object's vertices.

Object Object to tak epoint data from.

Radius Radius of the points.

System Particle system to use.

Falloff

Standard Todo.

Smooth Todo.

Soft Todo.

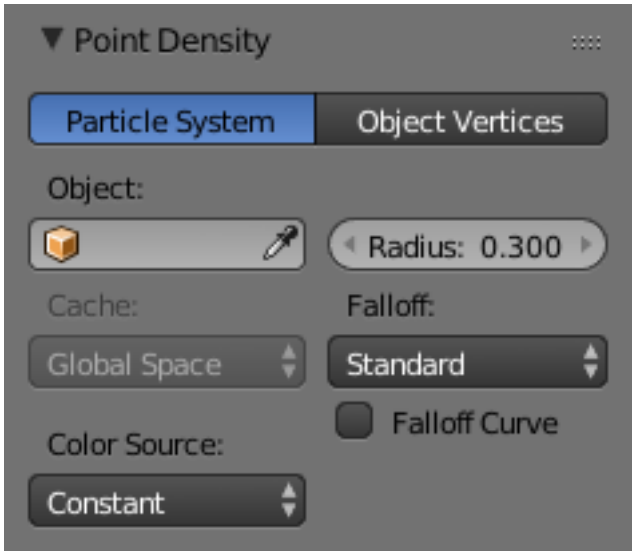


Fig. 2.1664: Point Density panel.

Softness Todo.

Constant Todo. Density is constant within lookup radius.

Root Todo.

Particle Age Todo.

Particle Velocity Todo.

Velocity Scale Todo.

Falloff Curve Use a custom falloff.

Cache Coordinate system to cache particles in.

Global Space Todo.

Emit Object Space Todo.

Emit Object Location Todo.

Color Source Data to derive the color results from.

Constant Constant color

Particle Color Sources

Particle Age Lifetime mapped as 0.0 - 1.0 intensity.

Particle Speed Particle speed (absolute magnitude of velocity) mapped as 0.0 - 1.0 intensity. An additional color ramp can be used to convert intensity to RGB colors.

Scale Multiplier to bring particle speed within an acceptable range.

Particle Velocity XYZ velocity mapped to RGB colors.

Scale Multiplier to bring particle speed within an acceptable range.

Vertex Color Sources

Vertex Color Use a vertex color layer for coloring the point density texture.

> **Note:** Vertex colors are defined per face corner. A single vertex can have as many different colors as faces it is part of. The actual color of the point density texture is averaged from all vertex corners.

Vertex Weight Use a weights from a vertex group as intensity values. An additional color ramp can be used to convert intensity to RGB colors.

Vertex Normals Use object-space vertex normals as RGB values.

Turbulence

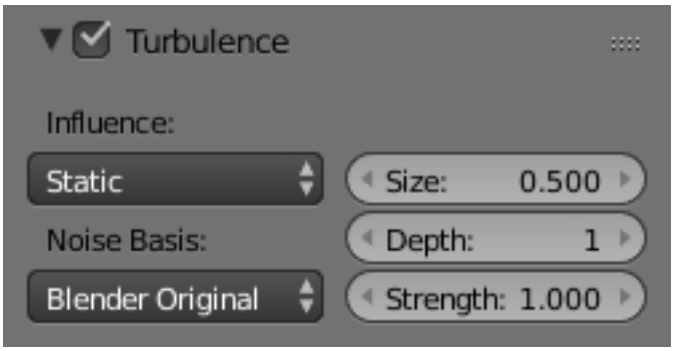


Fig. 2.1665: Turbulence panel.

Adds directed noise to the density at render time.

Influence Method for driving added turbulent noise.

Static Noise patterns will remain unchanged, faster and suitable for stills.

Particle Velocity Turbulent noise driven by particle velocity.

Particle Age Turbulent noise driven by the particle's age between birth and death.

Global Time Turbulent noise driven by the global current frame.

Noise Basis See *Here*.

Size Scale of the turbulent noise.

Depth Level of detail in the added turbulent noise.

Turbulence Strength Strength of the added turbulent noise.

Texture Nodes

Introduction

As an alternative to using the *Texture Stack*, Blender includes a node-based texture generation system, which enables textures creation by combining colors, patterns and other textures in the same way as shader writing with *Material Nodes*.

These textures can be used in the same places as regular textures: They can be placed in texture channels, in material nodes, in particle systems, and even inside other textures.

Note: Node-based textures do **not** work for realtime display, they will only be visible in rendered images.

Using Texture Nodes

To use texture nodes with the current texture, open the *Node Editor* and set it to *Texture* mode by clicking the "Texture" icon ( ) in its header.

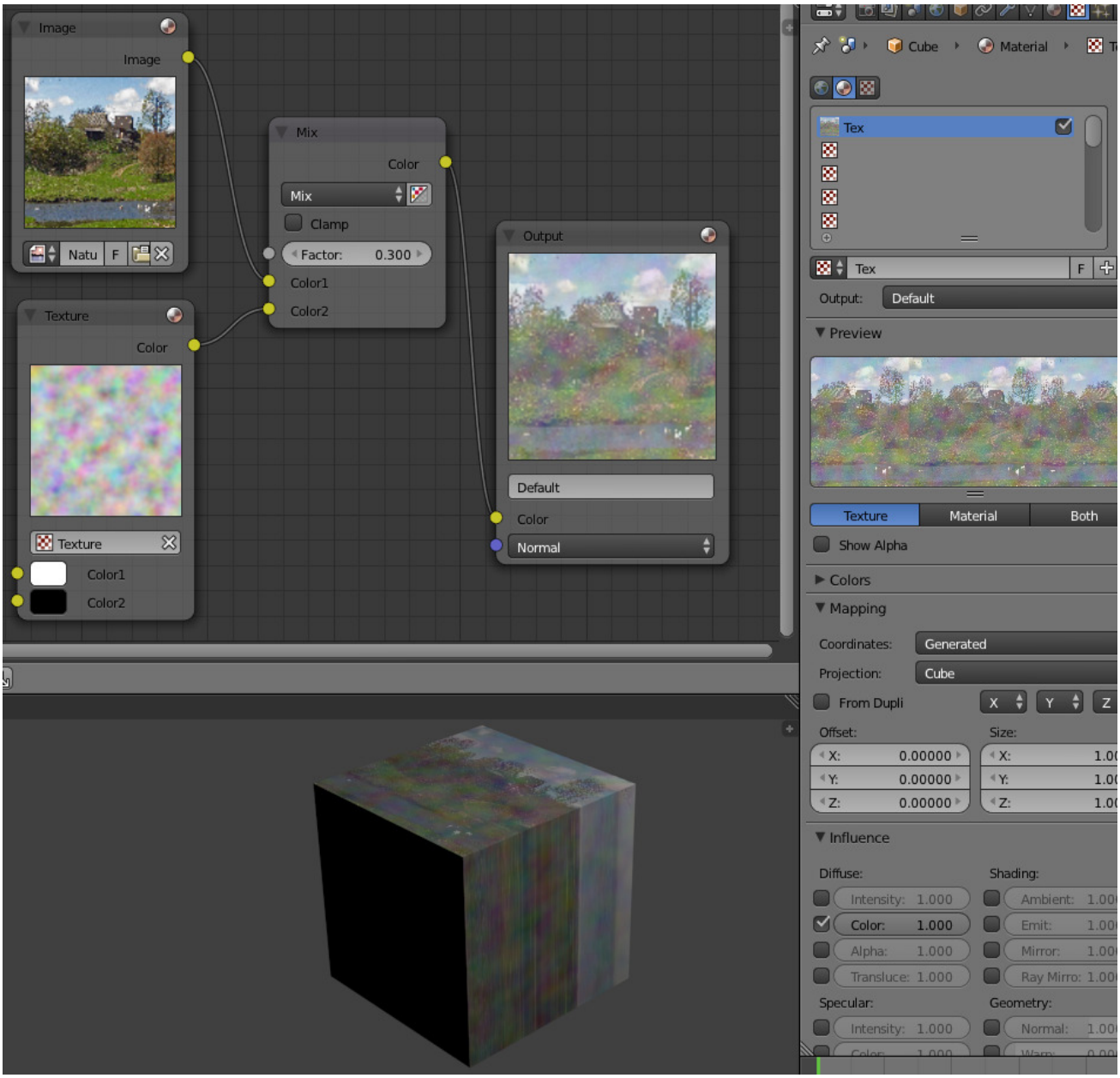


Fig. 2.1666: Combined textures based on nodes.

To start adding nodes, a material has to be to selected. A new texture can be created by either clicking the *New* button in the Node editor, or the *New* button in the texture panel. Once a texture is selected, it can be toggled to a function as a regular texture or a node texture by clicking the *Use Nodes* option in the Node Editor.

The default node setup will appear: a red-and-white checkerboard node connected to an *Output* named "Default". For *texture* nodes, multiple Outputs can exist in the node setup. Compare to other types of node contexts, which are limited to one active Output node. See the next section for details.

For instructions on how to add, remove and manipulate the nodes in the tree, see the *Node Editor manual*.

Using Multiple Outputs

Each texture defined with Texture Nodes can have several outputs, which can then be used for different things. For example, a texture that defines both a diffuse (color) map and a normal map. This can be done by:

- Create two texture slots in the texture list, and set them to the same texture data-block.
- Add two *Output* nodes to the node tree, and type new names into their *Name* text-boxes: e.g. "Diffuse" for one and "Normal" for the other.
- Underneath the texture list view in the texture panel, a selector with the names of the outputs are shown. For each entry in the texture list, select the desired output by changing the menu entry e.g. set on to *Diffuse* and the other to *Normal*).

These named outputs could be used, when the material is defined with Material Nodes. In this case, Texture Channels are probably not used. Instead, insert the *Texture* nodes into the Material Node tree by using *Add* → *Input* → *Texture*. Inside the just added texture node the output to use can then be selected (e.g. *Diffuse* or *Normal*).

Node Types

Color Nodes

Invert Node

This node inverts the colors in the input image, producing a negative.

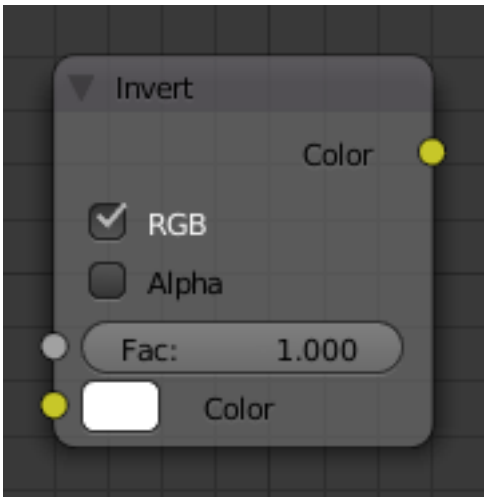


Fig. 2.1667: Invert Node.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Color Standard image input.

Properties

In the compositing context this node has the following properties.

RGB De/activation of the color channel inversion.

Alpha De/activation of the alpha channel inversion.

Outputs

Color Standard image output.

Mix Node

This node mixes images by working on the individual and corresponding pixels of the two input images. Called "MixRGB" in the shader and texture context.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image The background image. The image size and resolution sets the dimensions of the output image.

Image The foreground image.

Properties

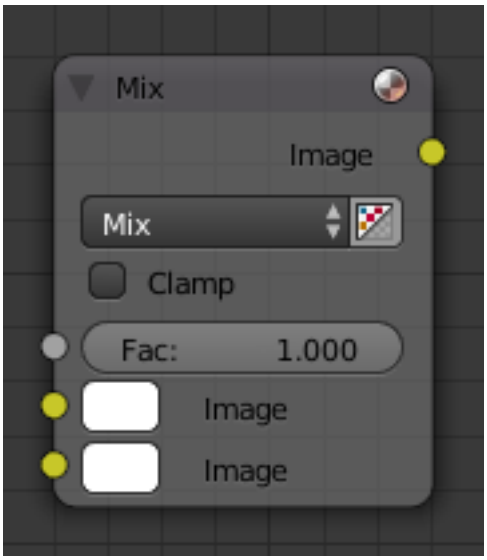


Fig. 2.1668: Mix Node.

Mix The Blend types could be selected in the select menu. See *Color Blend Modes* for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Dodge, Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Use Alpha If activated, by clicking on the *Color and Alpha* icon, the Alpha channel of the second image is used for mixing. When deactivated, the default, the icon background is a light gray. The alpha channel of the base image is always used.

Clamp Limit the highest color value to not exceed 1.

Outputs

Image Standard image output.

Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.

Some explanation of the mixing methods above might help you use the Mix node effectively:

Add adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.

Subtract Taking Blue away from white leaves Red and Green, which combined make Yellow. Taking Blue away from Purple leaves Red. Use this to desaturate an image. Taking away yellow makes an image bluer and more depressing.

Multiply Black (0.00) times anything leaves black. Anything times White (1.00) is itself. Use this to mask out garbage, or to colorize a black-and-white image.

Hue Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').

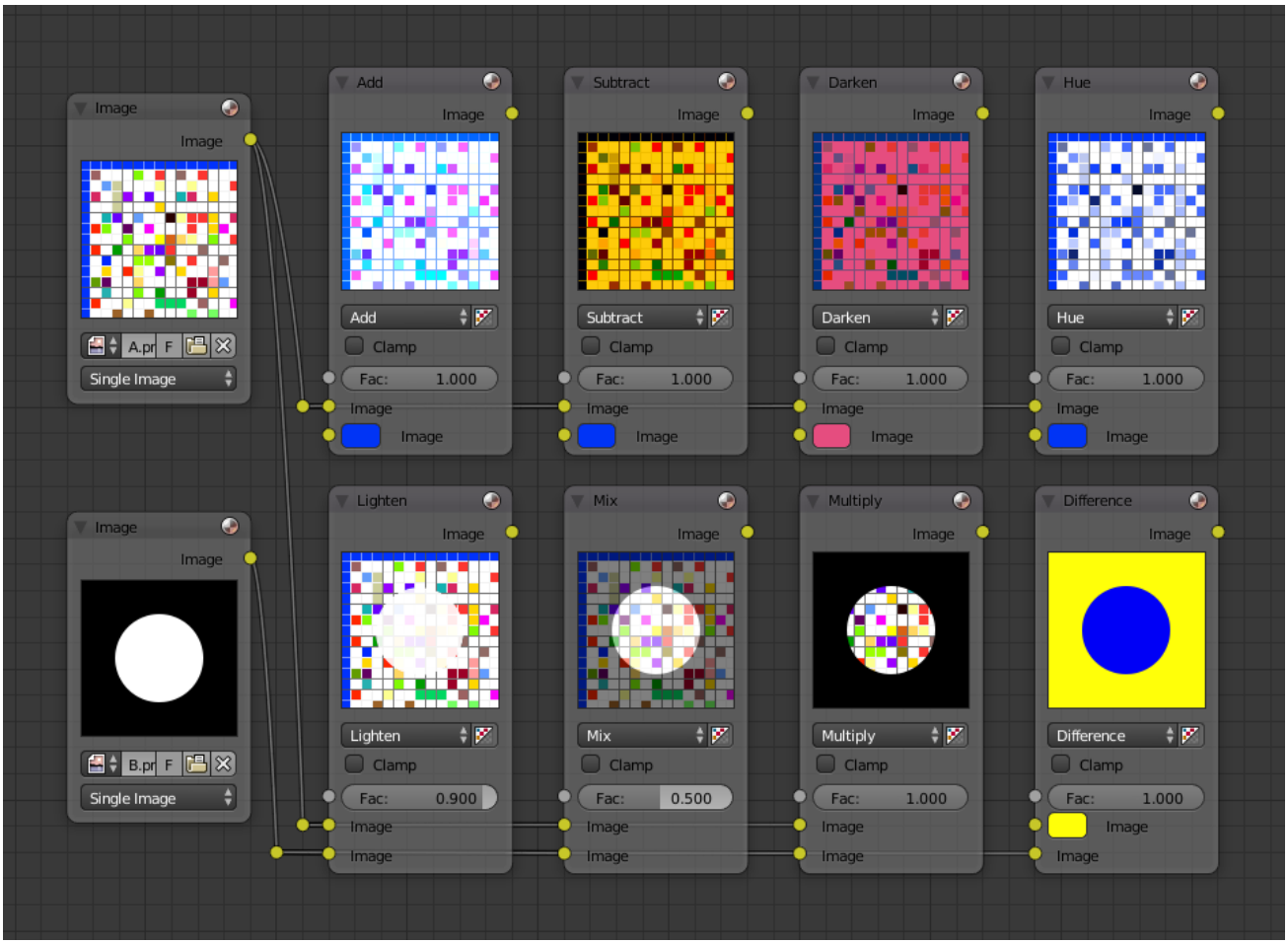


Image
Image
A.pr
F
Single Image
Add
Image
Add
Clamp
Fac: 1.000
Image
Image
Subtract
Image
Subtract
Clamp
Fac: 1.000
Image
Image
Darken
Image
Darken
Clamp
Fac: 1.000
Image
Image
Hue
Image
Hue
Clamp
Fac: 1.000
Image
Image
Image
Image
B.pr
F
Single Image
Lighten
Image
Lighten
Clamp
Fac: 0.900
Image
Image
Mix
Image
Mix
Clamp
Fac: 0.500
Image
Image
Multiply
Image
Multiply
Clamp
Fac: 1.000
Image
Image
Difference
Image
Difference
Clamp
Fac: 1.000
Image
Image

Mix Combines the two images, averaging the two.

Lighten Like bleach makes your whites whiter. Use with a mask to lighten up a little.

Difference Kinda cute in that it takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two verrry similar images to see what had been done to one to make it the other; sorta like a change log for images. You can use this to see a watermark (see *Watermark images*) you have placed in an image for theft detection.

Darken with the colors set here, is like looking at the world through rose-colored glasses.

Contrast Enhancement

Here is a small map showing the effects of two other common uses for the RGB Curve: *Darken* and *Contrast Enhancement*. You can see the effect each curve has independently, and the combined effect when they are *mixed* equally.

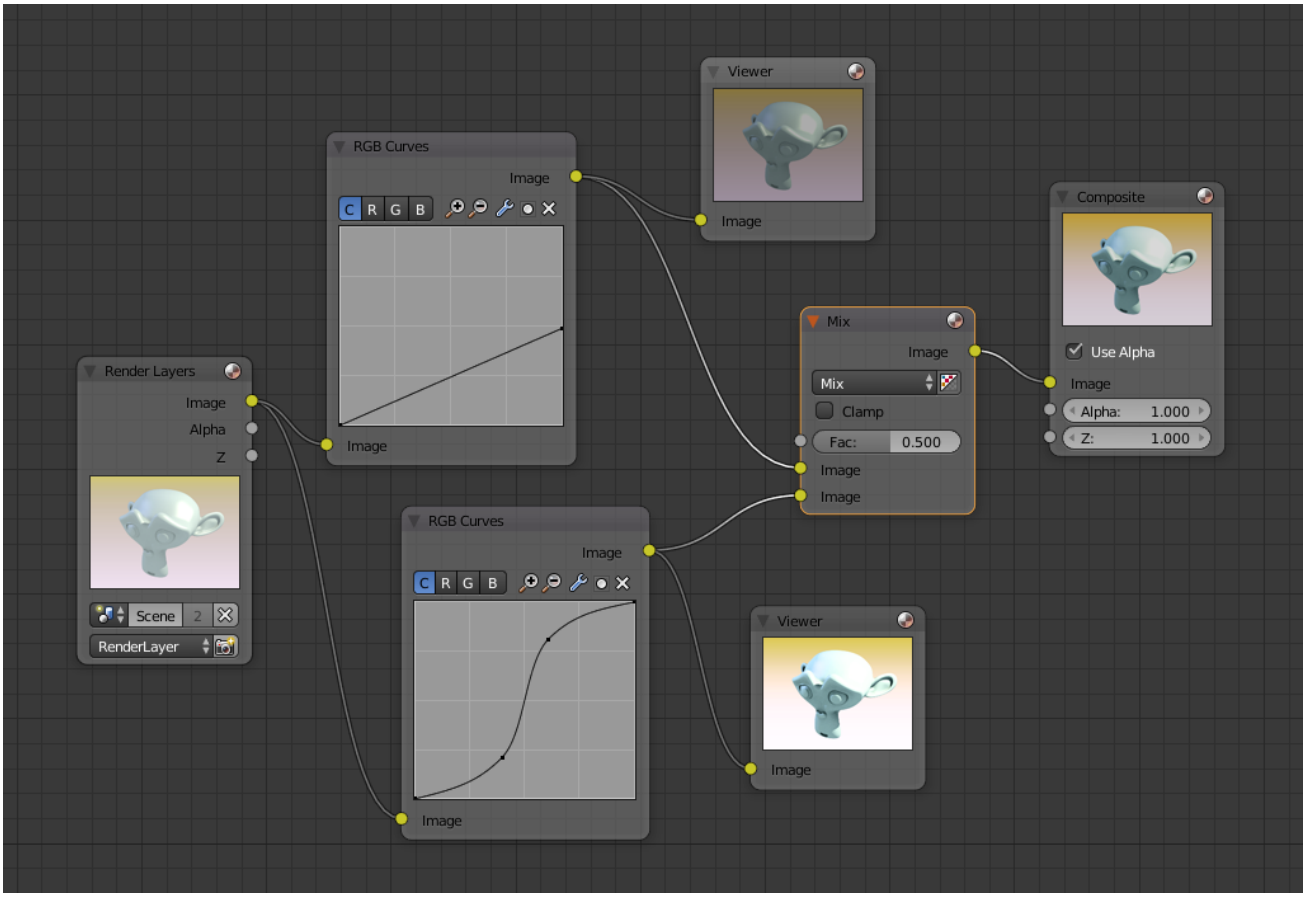


Fig. 2.1669: Example node setup showing “Darken”, “Enhance Contrast” and “Mix” nodes for composition.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast.

In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB (S shaped) curve scales the output such that

middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus, whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better.

Watermark images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising. Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting service, or simply to track their image's proliferation throughout the web. Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding Your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it is up to you. In the example below, we are encoding the watermark in a specific location in the image using the *Translate* node; this helps later because we only have to look at a specific location for the mark. We then use the RGB to BW node to convert the image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one-tenth of its original intensity. The Add node adds the corresponding pixels, make the ones containing the mark ever-so-slightly brighter.

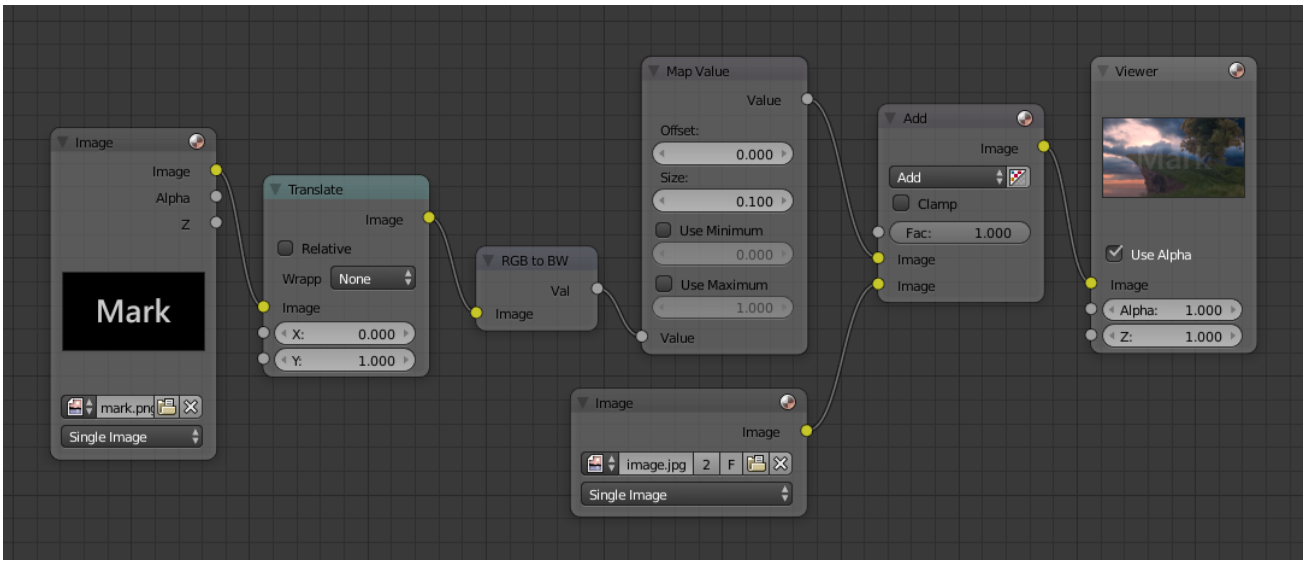


Fig. 2.1670: Embedding your mark in an Image using a Mark and Specific Position.

Of course, if you *want* people to notice your mark, do not scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Note: Additional uses

You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface, or as a special effect to make words appear on the possessed girl's forearm. yuk.

Decoding an Image for your Watermark

When you see an image that you think might be yours, use the node map below to compare it to your stock image (pre-watermarked original). In this map, the Mix node is set to Difference, and the Map Value node amplifies any difference. The result is routed to a viewer, and you can see how the original mark stands out, clear as a bell:

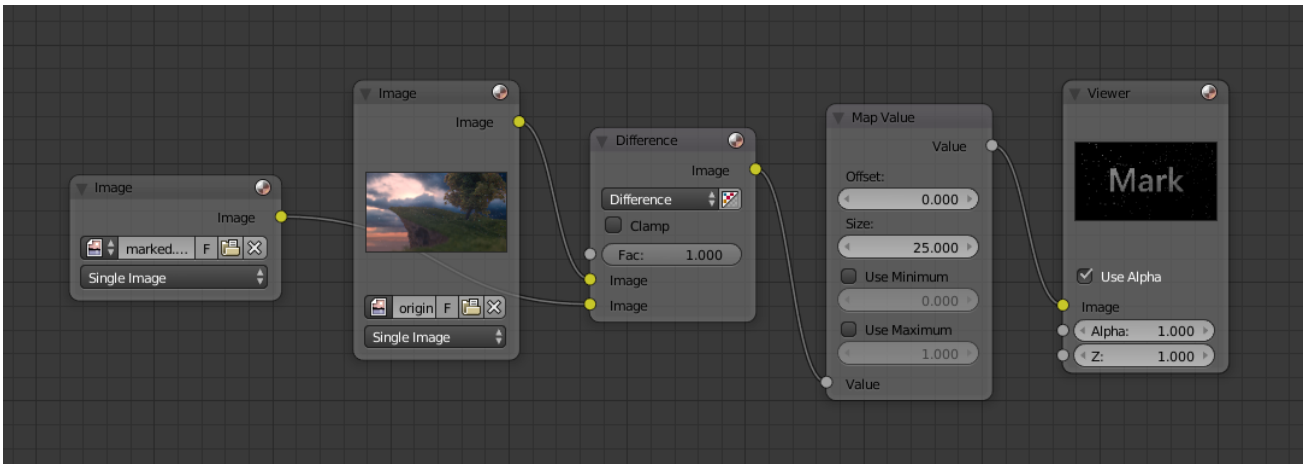
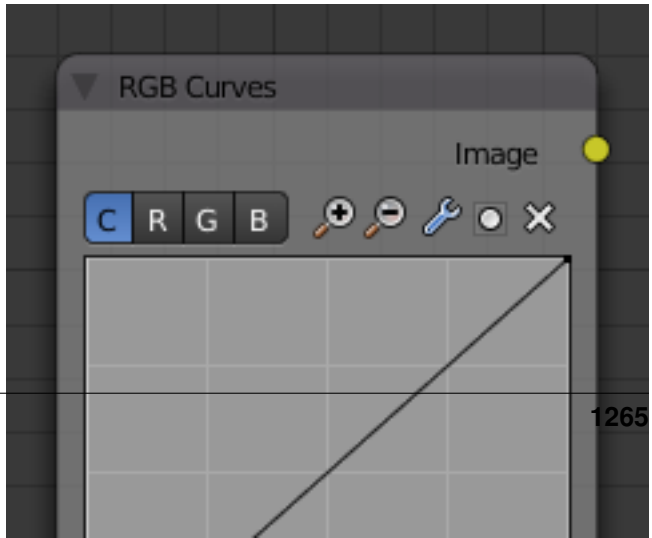


Fig. 2.1671: Checking an image for your watermark.

Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding map in one scene, and the decoding map in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to `JPEG` compression of 50%.

RGB Curves Node

This node allows color corrections for each color channel and levels adjustments in the compositing context.



Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Black Level Defines the input color that is (linear) mapped to black.

White Level Defines the input color that is (linear) mapped to white.

Tip: To define the levels, use the *eye dropper* to select a color sample of a displayed image.

Properties

Channel Clicking on one of the channels displays the curve for each.

C (Combined RGB), R (Red), G (Green), B (Blue), L (Luminance)

Curve A Bézier curve that varies the input levels (x-axis) to produce an output level (y-axis). For the curve controls see: *Curve widget*.

Outputs

Image Standard image output.

Examples

Here are some common curves you can use to achieve desired effects:

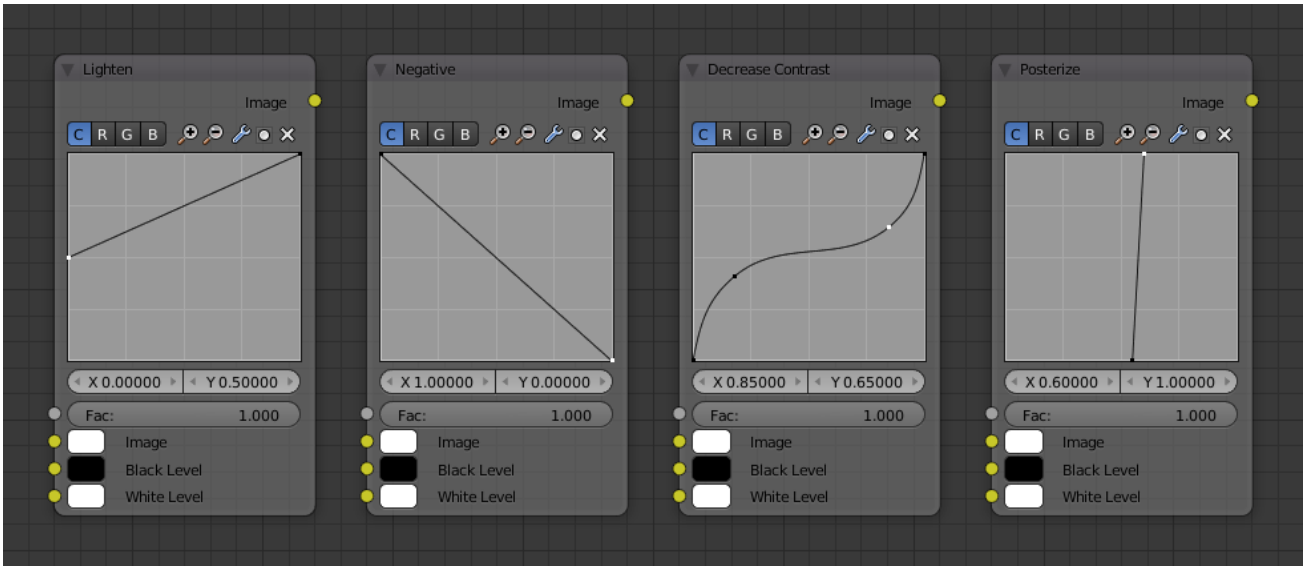


Fig. 2.1673: From left to right: 1. Lighten 2. Negative 3. Decrease Contrast 4. Posterize.

Color correction using Curves

In this example, the image has way too much red in it, so we run it through an RGB node and reduce the Red channel by about half.

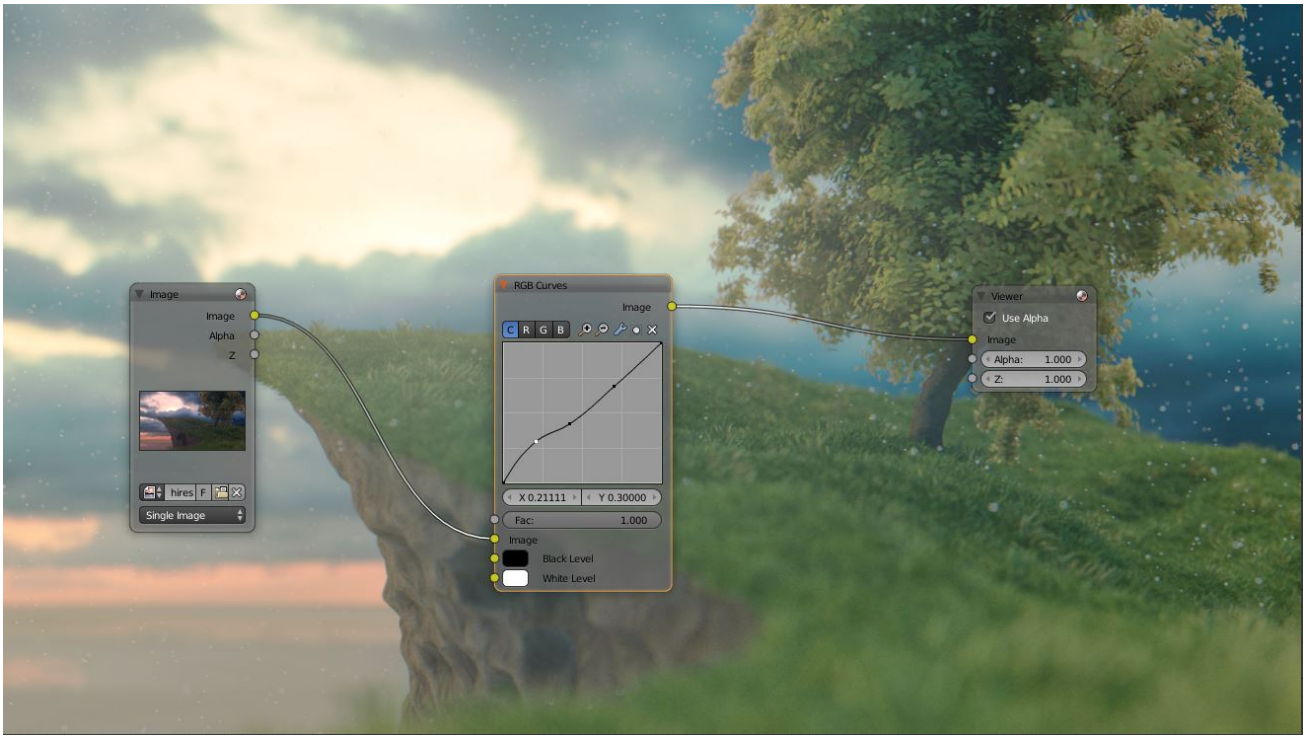


Fig. 2.1674: Color correction with curves.

We added a middle dot so we could make the line into a sideways exponential curve. This kind of curve evens out the amount of a color in an image as it reaches saturation. Also, read on for examples of the Darken and Contrast Enhancement curves.

Color correction using Black/White Levels

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the UV/Image editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side-effect of setting Black and White Levels.

Effects

Curves and Black/White Levels can also be used to completely change the colors of an image.

Note that e.g. setting Black Level to red and White Level to blue does not simply substitute black with red and white with blue as the example image might suggest. Levels do color scaling, not substitution, but depending on the settings they can result in the described color substitution.

(What really happens when setting Black Level to pure red and White Level to pure blue is that the red channel gets inverted, green gets reduced to zero and blue remains unchanged.)

Because of this, the results of setting arbitrary Black/White Levels or RGB curves is hard to predict, but can be fun to play with.

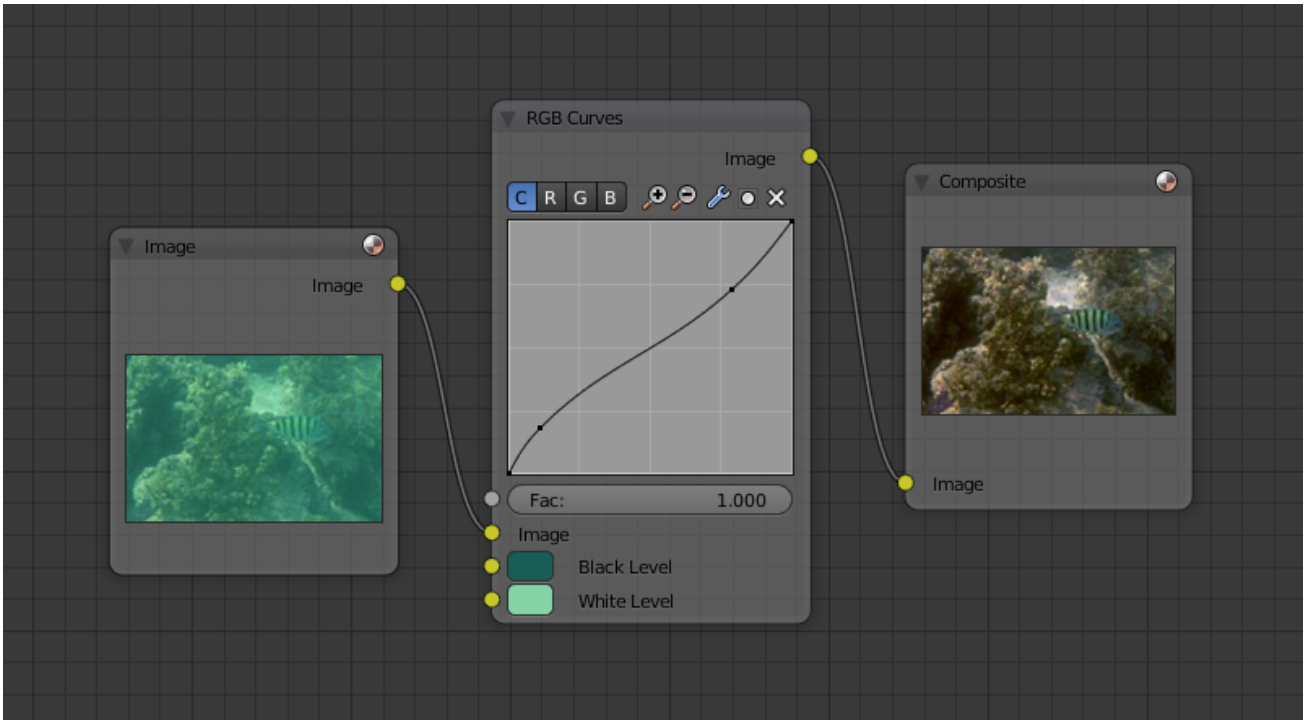


Fig. 2.1675: Color correction with Black/White Levels.

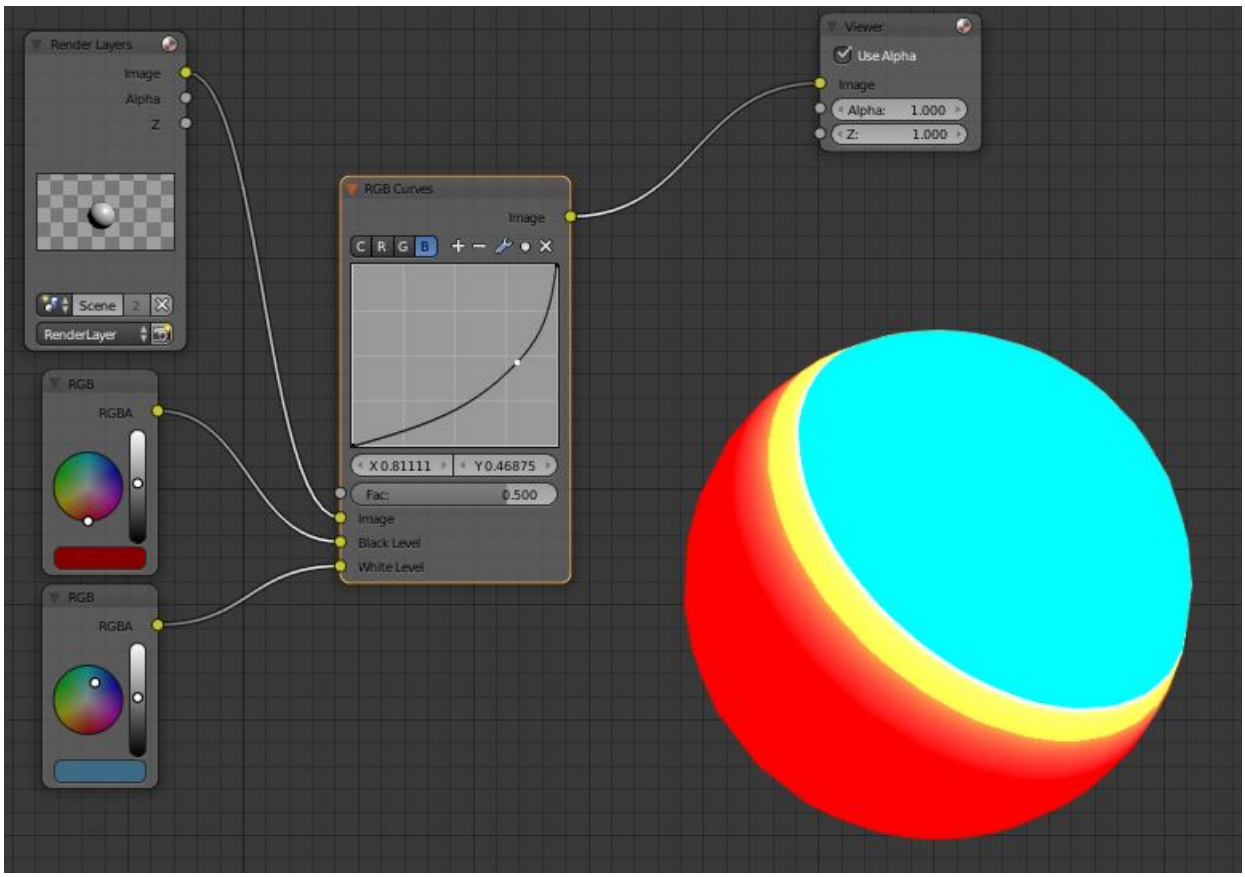


Fig. 2.1676: Changing colors.

Hue Saturation Value Node

This node applies a color transformation in the HSV color space. Called "Hue Saturation Value" in shader and texture context.

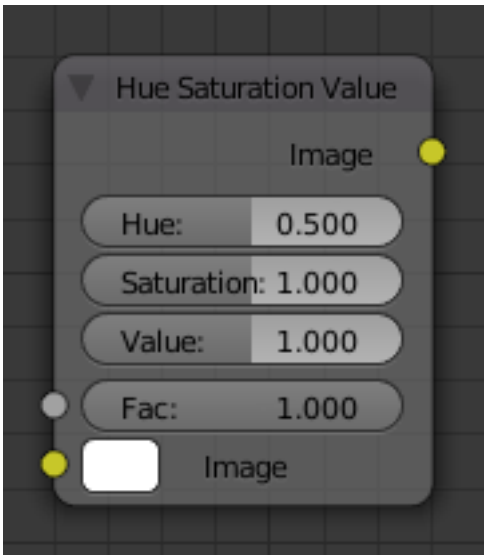


Fig. 2.1677: Hue Saturation Node.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Properties

The transformations are relative shifts. In the shader and texture context the following properties are available as input sockets.

Hue Specifies how the hue rotation of the image. 360° are mapped to (0 to 1). The hue shift of 0 (-180°) and 1 (+180°) have the same result.

Saturation A saturation of 0 removes hues from the image, resulting in a grayscale image. A shift greater 1.0 increases saturation.

Value Value is the overall brightness of the image. De/Increasing values shift an image darker/lighter.

Outputs

Image Standard image output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are vice versa A blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (recall that white, minus blue, equals yellow). A yellow image, with a Hue setting at 0 or 1, is blue.

Hue and Saturation work together. So, a Hue of 0.5 keeps the blues the same shade of blue, but *Saturation* can deepen or lighten the intensity of that color.

Gray & White are neutral hues A gray image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with *Value*. This applies to all shades of gray, from black to white; wherever the values are equal.

Changing the effect over time The Hue and Saturation values can be animated with a *Time Node* or by animating the property.

Note: Tinge

This HSV node simply shifts hues that are already there. To colorize a gray image, or to add a tint to an image, use a mix node to add in a static color from an RGB input node with your image.

HSV Example

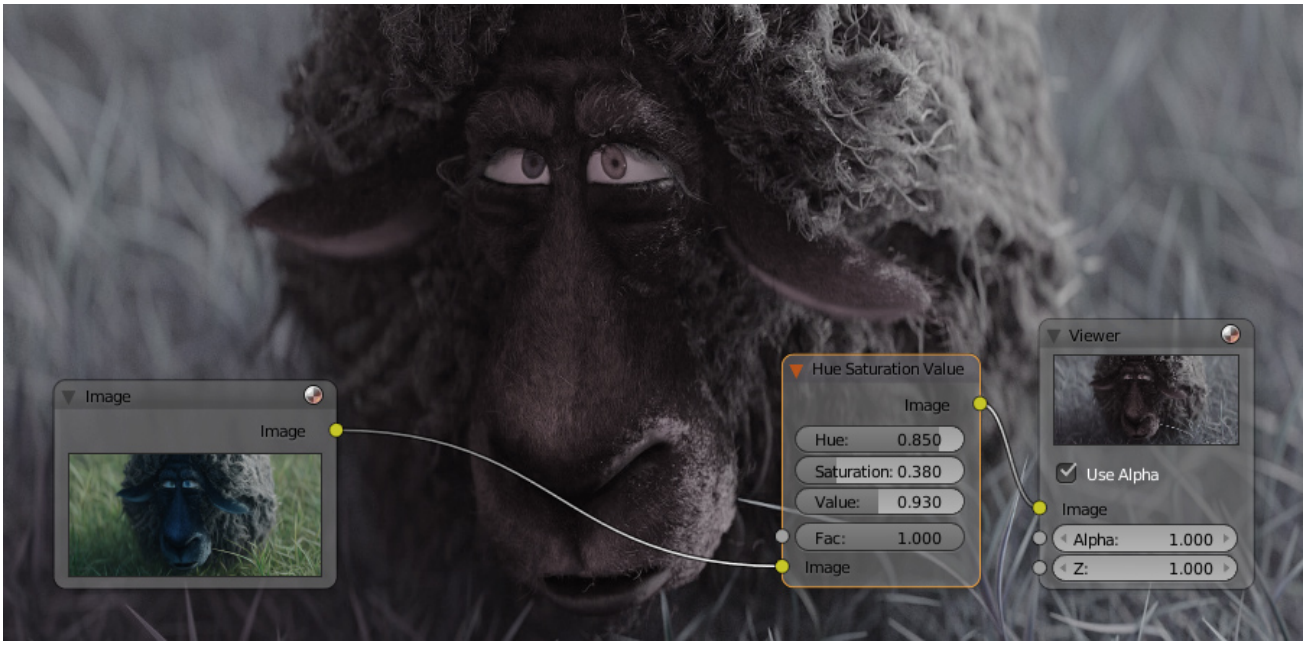


Fig. 2.1678: A basic example.

Combine/Separate Nodes

All of these nodes do essentially the same thing:

- Separate: Split out an image into its composite color channels.
- Combine: Re/combine an image from it is composite color channels.

This nodes could be use this to manipulate on each color channel independently. Each type is differentiate in the applied *color space*.

In compositing and texture context each node supports the Alpha channel. In the texture context only RGB color space is available. In the shading context of the Blender internal adds HSV and the Cycles shading context offers an additional pair of nodes to combine/separate a vector (XYZ).

The Combine nodes could also be used to input single color values. For RGBA and HSVA color spaces it is recommended to use the *RGB Node*. Some common operation could easier executed with the *Color Nodes*.

Separate/Combine RGBA Node

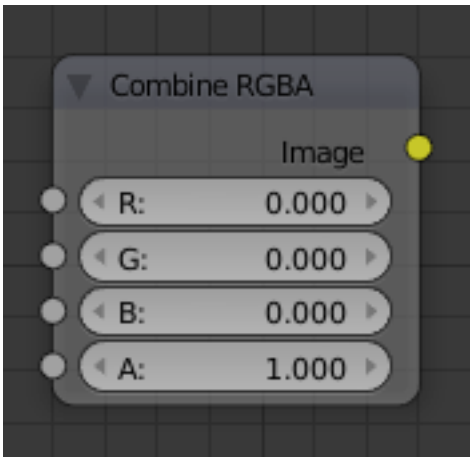


Fig. 2.1679: Combine RGBA Node.

Input/ Output

Image Standard image in/output.

- R (Red)
- G (Green)
- B (Blue)
- A (Alpha)

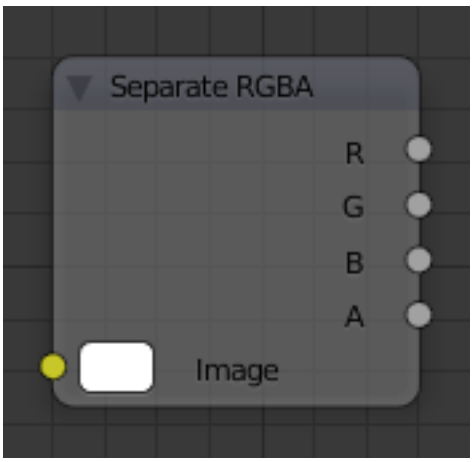


Fig. 2.1680: Separate RGBA Node.

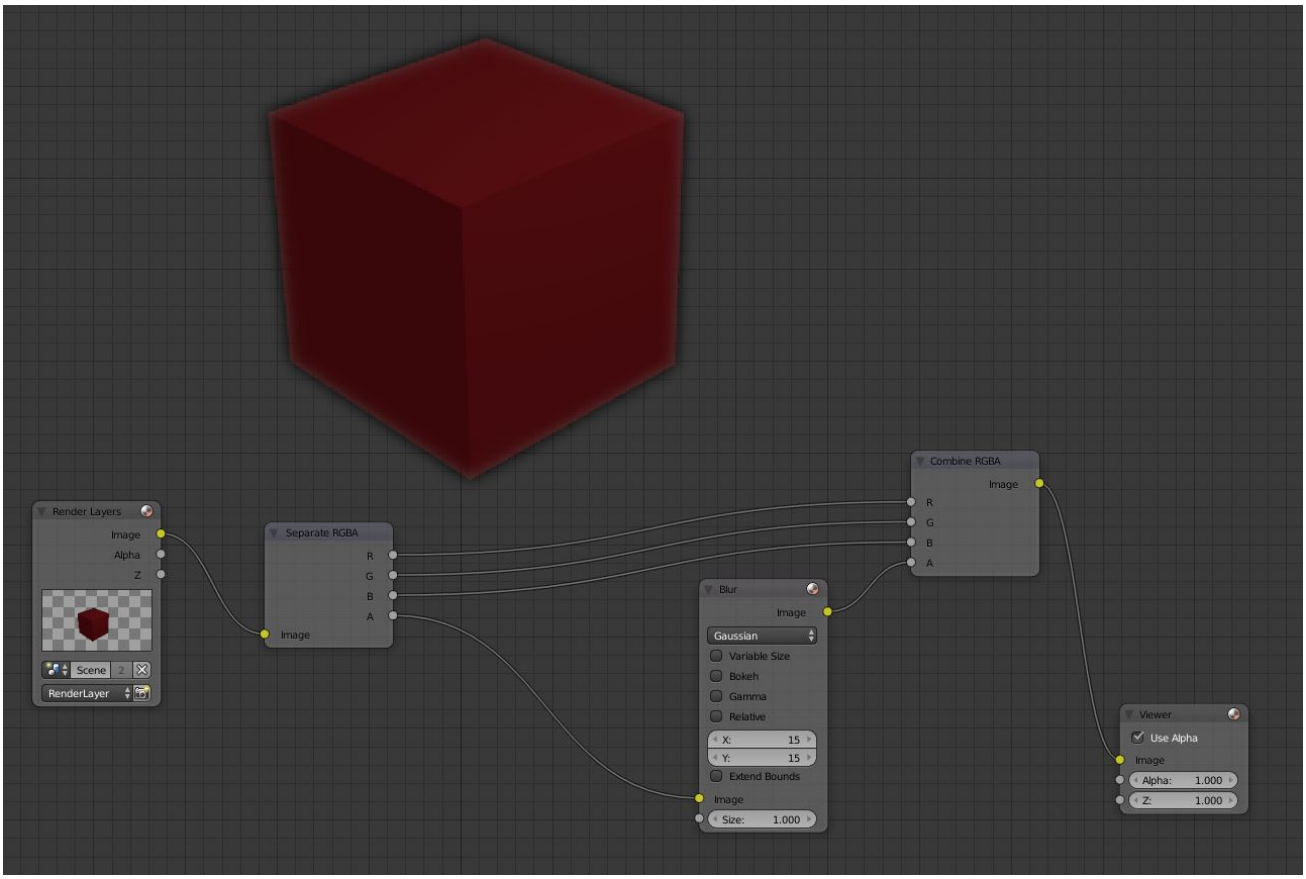
Properties

This node has no properties.

Examples

In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like anti-aliasing but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a broader scale will make the object appear to "phase" in and out, as an "out-of-phase" time-traveling sync effect.

In this node set up, we make all the reds become green, and all the green both Red and Blue, and remove Blue from the image completely.


Render Layers
Image
Alpha
Z
Scene
RenderLayer
Separate RGBA
R
G
B
A
Image
Combine RGBA
Image
R
G
B
A
Blur
Image
Gaussian
Variable Size
Bokeh
Gamma
Relative
X: 15
Y: 15
Extend Bounds
Image
Size: 1.000
Viewer
Use Alpha
Image
Alpha: 1.000
Z: 1.000

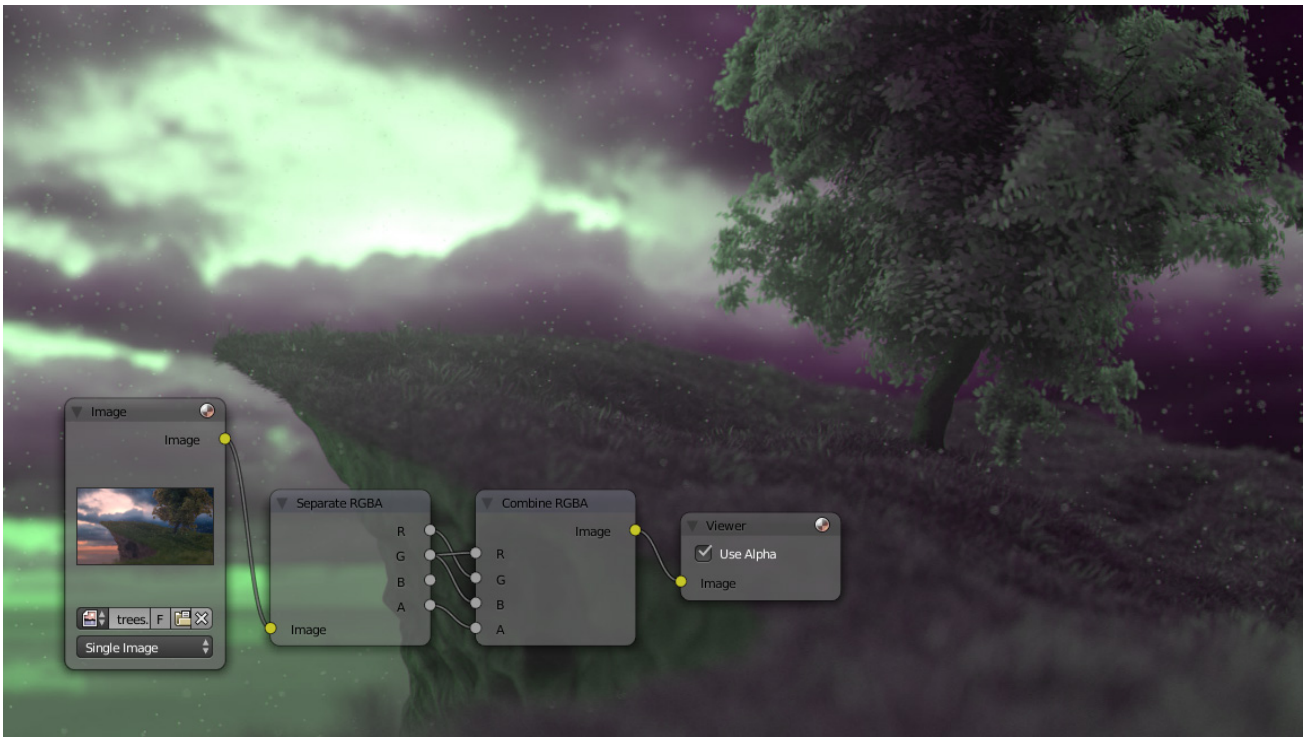


Image
Image
trees. F
Single Image
Separate RGBA
R
G
B
A
Image
Combine RGBA
Image
R
G
B
A
Viewer
Use Alpha
Image

Separate/Combine HSVA Nodes

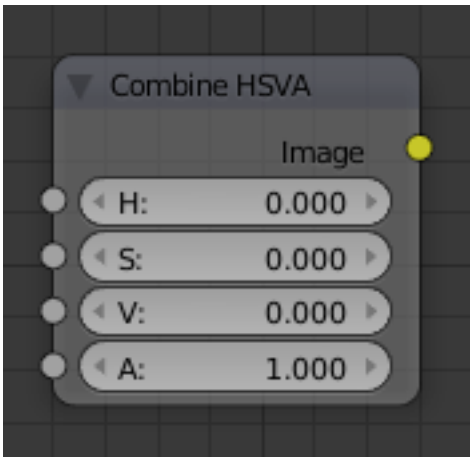


Fig. 2.1681: Combine HSVA Node.

Input/ Output

Image Standard image in/output.

- H (Hue)
- S (Saturation)
- V (Value)
- A (Alpha)

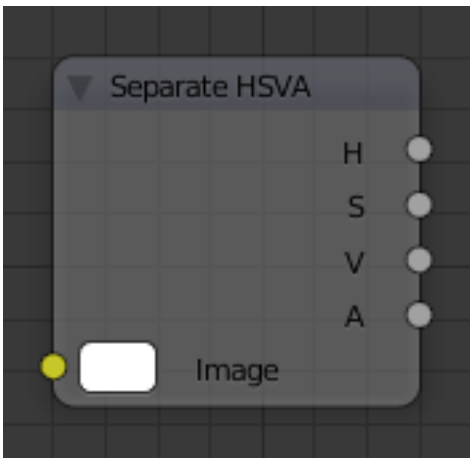


Fig. 2.1682: Separate HSVA Node.

Properties

This node has no properties.

Separate/Combine YUVA Node

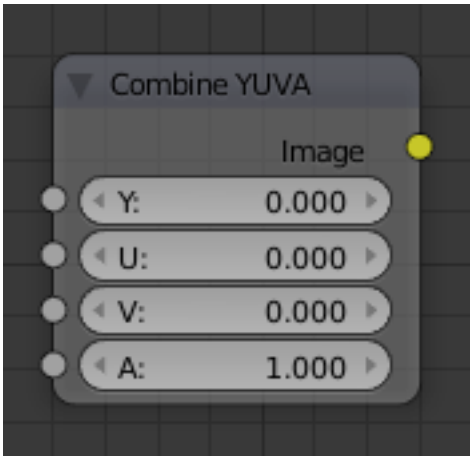


Fig. 2.1683: Combine YUVA Node.

Input/ Output

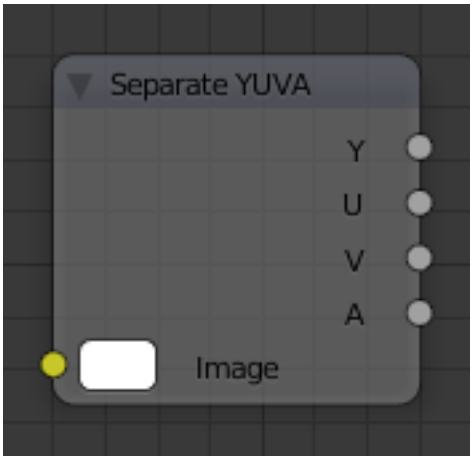


Fig. 2.1684: Separate YUVA Node.

Image Standard image in/output.

- Y (Luminance)
- U (U chrominance)
- V (V chrominance)
- A (Alpha)

Properties

This node has no properties.

Separate/Combine YCbCrA Node

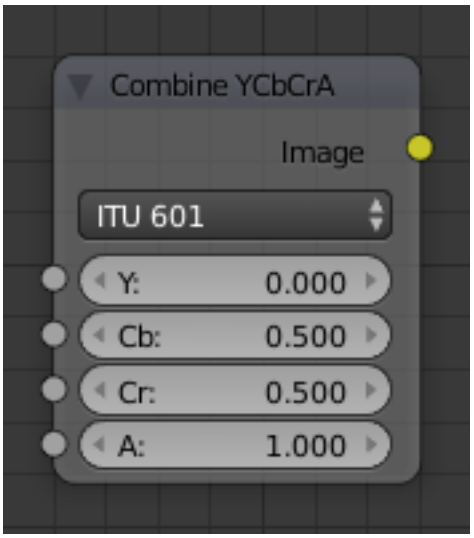


Fig. 2.1685: Combine YCbCrA Node.

Input/ Output

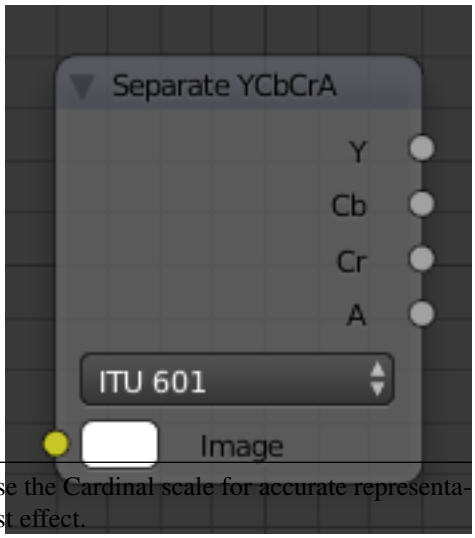


Fig. 2.1686: Separate YCbCrA Node.

Image Standard image in/output.

- Y (Luminance)
- Cb (Chrominance Blue)
- Cr (Chrominance Red)
- A (Alpha)

Properties

Mode ITU 601, ITU 709, Jpeg

Tip: If running these channels through a Color Ramp node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives high-contrast effect.

Converter Nodes

Color Ramp Node

The Color Ramp Node is used for mapping values to colors with the use of a gradient.

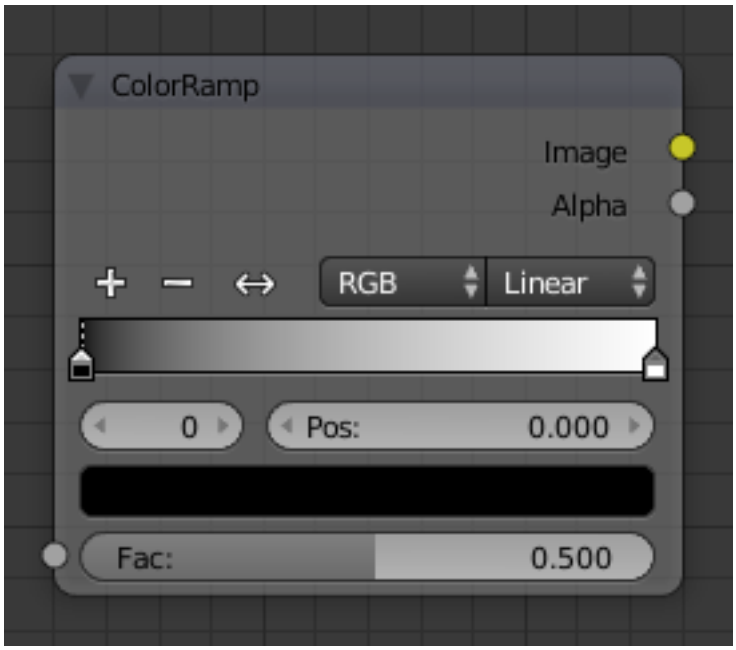


Fig. 2.1687: Color Ramp Node.

Inputs

Factor The Factor input is used as an index for the color ramp.

Properties

Color Ramp For controls see *Color Ramp Widget*.

Outputs

Image Standard image output.

Alpha Standard alpha output.

Examples

Creating an Alpha Mask

A powerful but often overlooked feature of the Color Ramp is to create an Alpha Mask, or a mask that is overlaid on top of another image, and, like a mask, allows some of the background to show through. The example map below shows how to use the Color Ramp node to do this:

In the map above, a black and white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*. (Technically, we should have converted the image to a value using the RGB-to-BW node, buy hey, this works just as well since we are using a BW image as input.)

We have set the Color Ramp node to a purely transparent color on the left end of the spectrum, and a fully Red color on the right. As seen in the viewer, the Color Ramp node puts out a mask that is fully transparent where the image is black. Black is zero, so Color Ramp uses the color at the left end of the spectrum, which we have set to transparent. The Color Ramp image is fully red and opaque where the image is white (1.00).

We verify that the output image mask is indeed transparent by overlaying it on top of a other image.

Colorizing an Image

The real power of Color Ramp is that multiple colors can be added to the color spectrum. This example compositing map takes a boring BW image and makes it a flaming swirl!

In this example, we have mapped the shades of gray in the input image to three colors, blue, yellow, and red, all fully opaque (Alpha of 1.00). Where the image is black, Color Ramp substitutes blue, the currently selected color. Where it is

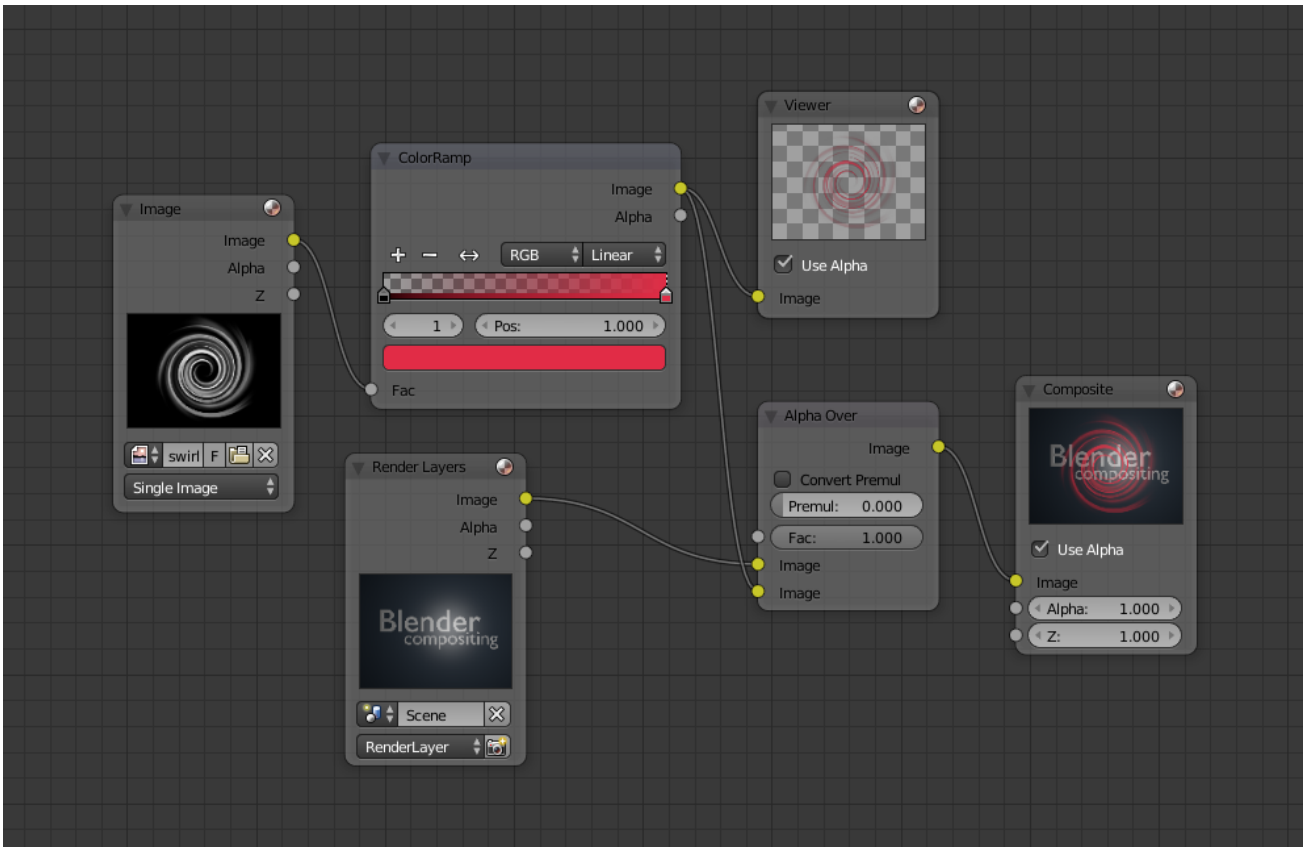
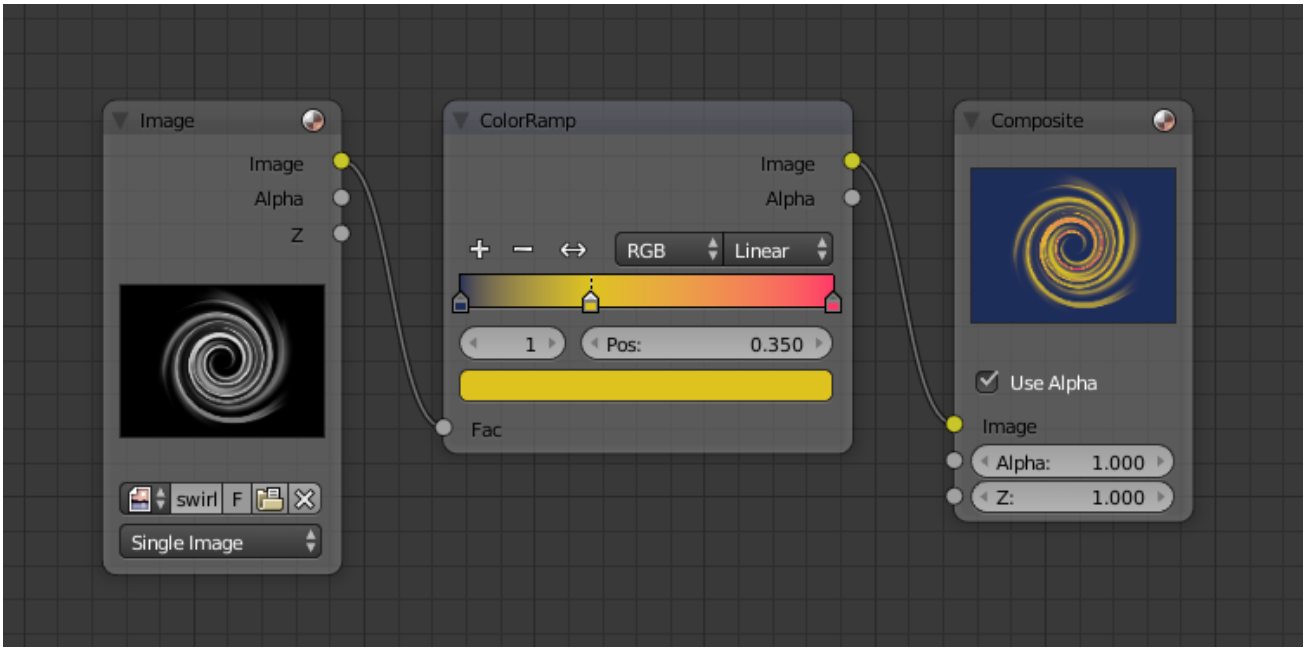


Fig. 2.1688: Using the Color Ramp node to create an alpha mask.



some shade of gray, Color Ramp chooses a corresponding color from the spectrum (bluish, yellow, to reddish). Where the image is fully white, Color Ramp chooses red.

Distance Node

Computes the distance between two 3D coordinates.

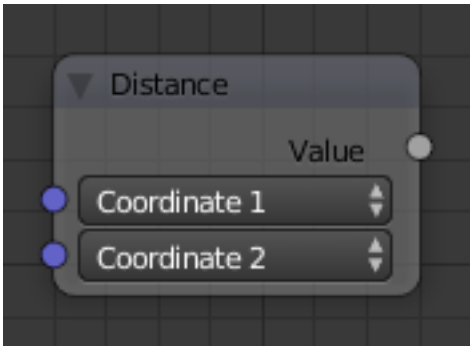


Fig. 2.1689: Distance node.

Inputs

Coordinate 1 First coordinate point.

Coordinate 2 Second coordinate point.

Properties

This node has no properties.

Outputs

Value Calculated distance.

Math Node

This node performs math operations.

Inputs

Value First numerical value. The trigonometric functions accept values in radians.

Value Second numerical value. This value is **not** used in functions that accept only one parameter like the trigonometric functions, Round and Absolute.

Properties

Operation Add, Subtract, Multiply, Divide, Sine, Cosine, Tangent, Arcsine, Arccosine, Arctangent, Power, Logarithm, Minimum, Maximum, Round, Less Than, Greater Than, Modulo, Absolute.

Clamp Limits the output to the range (0 to 1). See *clamp*.

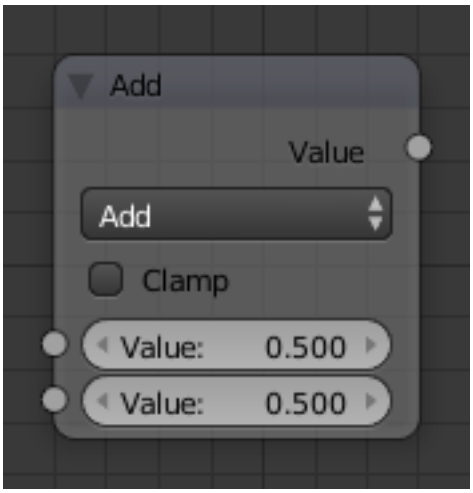


Fig. 2.1690: Math node.

Outputs

Value Numerical value output.

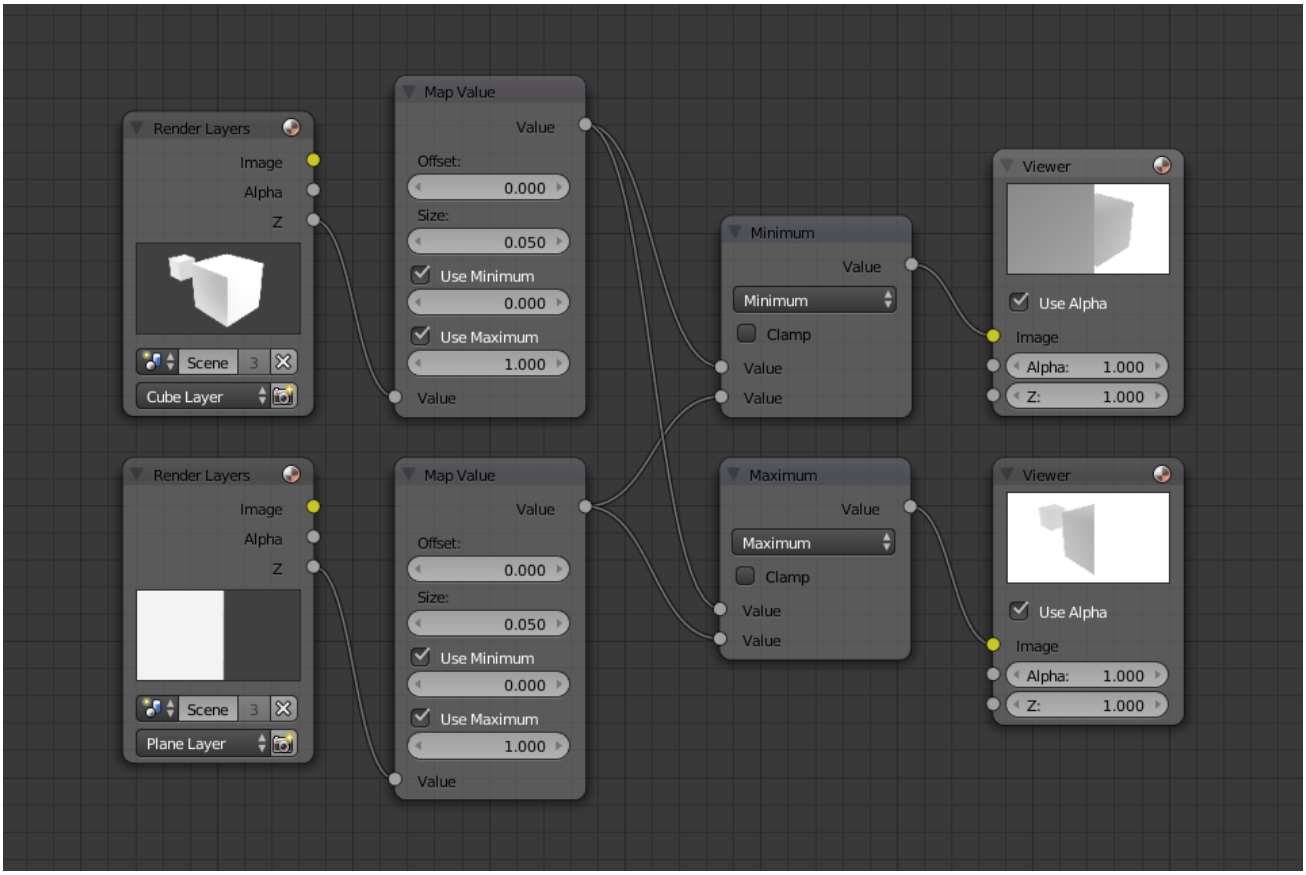


Fig. 2.1691: Example.

Examples

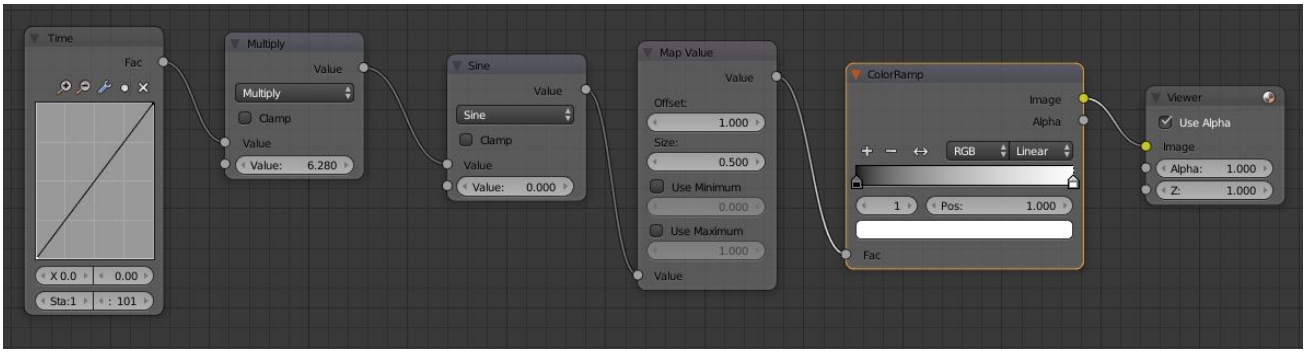
Manual Z-Mask

This example has one scene input by the top *Render Layer* node, which has a cube that is about 10 BU from the camera. The bottom Render Layer node inputs a scene (FlyCam) with a plane that covers the left half of the view and is 7 BU from the camera. Both are fed through their respective Map Value nodes to divide the Z buffer by 20 (multiply by 0.05, as shown in the Size field) and clamped to be a min/ max of 0.0/ 1.0 respectively.

For the minimum function, the node selects those Z values where the corresponding pixel is closer to the camera; so it chooses the Z values for the plane and part of the cube. The background has an infinite Z value, so it is clamped to 1.0 (shown as white). In the maximum example, the Z values of the cube are greater than the plane, so they are chosen for the left side, but the plane (FlyCam) Render layers Z are infinite (mapped to 1.0) for the right side, so they are chosen.

Using Sine Function to Pulsate

This example has a *Time* node putting out a linear sequence from 0 to 1 over the course of 101 frames. The green vertical line in the curve widget shows that frame 25 is being put out, or a value of 0.25. That value is multiplied by 2 × pi and converted to 1.0 by the Sine function, since we all know that $sin(2pi/4) = sin(pi/2) = +1.0$ Since the sine function can put out val-



ues between (-1.0 to 1.0), the *Map Value* node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by one-half (thus scaling the output between 0 to 1). The default *Color Ramp* converts those values to a grayscale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, $sin(pi/2) = 1.0$. Like having your own visual color calculator! Animating this node setup provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move a scene in/out of focus. Alter a color channel value to make a color "pulse".

Brightening/Scaling a Channel

This example has a *Math: Multiply* node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a *Map Value node* with min() and max() enabled to clamp the output to valid values. With this approach, you could use a logarithmic function to make a high-dynamic range image. For this particular example, there is also a *Brighten/Contrast node* that might give simpler control over brightness.

Quantize/Restrict Color Selection

In this example, we want to restrict the color output to only 256 possible values. Possible use of this is to see what the image will look like on an 8-bit cell phone display. To do this, we want to restrict the R, G and B values of any pixel to be one of a certain value, such that when they are combined, will not result in more than 256 possible values. The number of possible values of an output is the number of channel values multiplied by each other, or Q = R × G × B.

Since there are three channels and 256 values, we have some flexibility how to quantize each channel, since there are a lot of combinations of R × G × B that would equal 256. For example, if {R, G, B} = {4, 4, 16}, then $4416 = 256$. Also, {6, 6, 7} would give 252 possible values. The difference in appearance between {4, 4, 16} and {6, 6, 7} is that the first set (4, 4, 16} would have fewer shades of red and green, but lots of shades of blue. The set {6, 6, 7} would have a more even distribution of colors. To get better image quality with fewer color values, give possible values to the predominant colors in the image.

Theory

Two Approaches to Quantizing to six values.

To accomplish this quantization of an image to 256 possible values, let us use the set {6, 6, 7}. To split up a continuous range of values between 0 and 1 (the full Red spectrum) into six values, we need to construct an algorithm or function that takes any input value but only puts out six possible values, as illustrated by the image to the right. We want to include zero as true black, with five other colors in between. The approach shown produces {0, 0.2, 0.4, 0.6, 0.8, 1}. Dividing 1.0 by 5 equals 0.2, which tells how far apart each quantified value is from the other.

So, to get good even shading, we want to take values that are 0.16 or less and map them to 0.0; values between 0.16 and 0.33 get fixed to 0.2; color band values between 0.33 and 0.5 get quantized to 0.4, and so on up to values between 0.83 and 1.0 get mapped to 1.0.

Note: Function f(x)

An algebraic function is made up of primitive mathematical operations (add, subtract, multiply, sine, cosine, etc) that operate on an input value to provide the desired output value.

Spreadsheet showing a function.

The theory behind this function is scaled truncation. Suppose we want a math function that takes in a range of values between 0 and 1, such as 0.552, but only outputs a value of 0.0, 0.2, 0.4, etc. We can imagine then that we need to get that range 0 to 1 powered up to something 0 to 6 so that we can chop off and make it a whole number. So, with six divisions, how can we do that? The answer is we multiply the range by 6. The output of that first math Multiply Node is a range of values between 0 and 6. To get even divisions, because we are using the rounding function (see documentation above), we want any number plus or minus around a whole number will get rounded to that number. So, we subtract a half, which shifts everything over. The round() function then makes that range 0 to 5. We then divide by 5 to get back a range of numbers between 0 and 1 which can then be

combined back with the other color channels. Thus, you get the function $f(x,n) = round(xn - 0.5)/(n-1)$ where "n" is the number of possible output values, and "x" is the input pixel color and $f(x,n)$ is the output value. There is only one slight problem, and that is for the value exactly equal to 1, the formula result is 1.2, which is an invalid value. This is because the round function is actually a roundup function, and exactly 5.5 is rounded up to 6. So, by subtracting 0.501, we compensate and thus 5. 499 is rounded to 5. At the other end of the spectrum, pure black, or 0, when 0.501 subtracted, rounds up to 0 since the Round() function does not return a negative number.

Sometimes using a spreadsheet can help you figure out how to put these nodes together to get the result that you want. Stepping you through the formula for $n = 6$ and $x = 0.70$, locate the line on the spreadsheet that has the 8-bit value 179 and R value 0.7. Multiplying by 6 gives 4.2 . Subtracting 1/2 gives 3.7, which rounds up to 4.4 divided by 5 = 0.8. Thus, f(0.7, 6) = 0.8 or an 8-bit value of 204. You can see that this same 8-bit value is output for a range of input values.

Reality

To implement this function in Blender, consider the node setup above. First, feed the image to the Separate RGB node. For the Red channel, we string the math nodes into a function that takes each red color, multiplies (scales) it up by the desired number of divisions (6), offsets it by 0.5, rounds the value to the nearest whole number, and then divides the image pixel color by 5. So, the transformation is {0 to 1} becomes {0 to 6}, subtracting centers the medians to {-0.5 to 5.5} and the rounding to the nearest whole number produces {0, 1, 2, 3, 4, 5} since the function rounds down, and then dividing by five results in six values {0.0, 0.2, 0.4, 0.6, 0.8, 1.0}.

The result is that the output value can only be one of a certain set of values, stair-stepped, because of the rounding function of the math node node setup. Copying this one channel to operate on Green and Blue gives the node setup below. To get the 6:6:7, we set the three Multiply Nodes to {6, 6, 7} and the divide nodes to {5, 5, 6}.

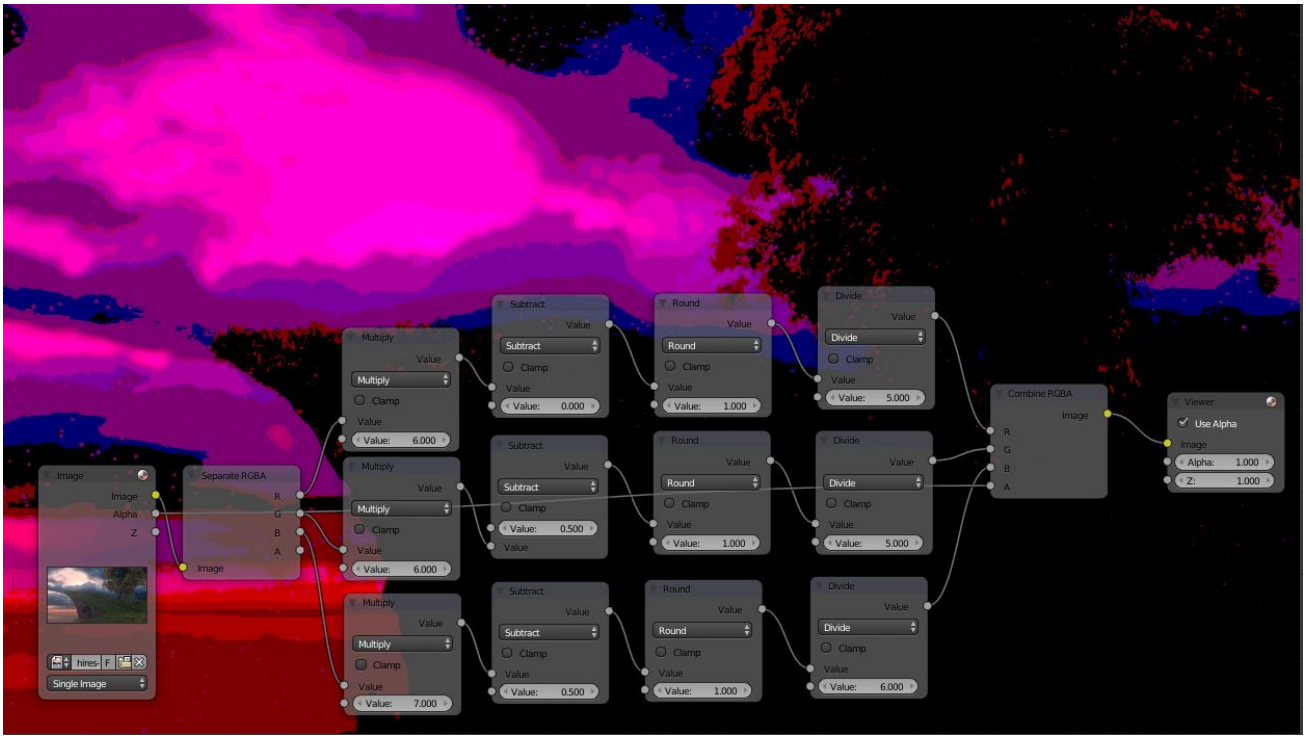
If you make this into a node group, you can easily re-use this setup from project to project. When you do, consider using a math node to drive the different values that you would have to otherwise set manually, just to error-proof your work.

Summary

Normally, an output render consists of 32- or 24-bit color depth, and each pixel can be one of the millions of possible colors. This node setup example takes each of the Red, Green and Blue channels and normalizes them to one of a few values. When all three channels are combined back together, each color can only be one of 256 possible values.

While this example uses the Separate/Combine RGB to create distinct colors, other Separate/Combine nodes can be used as well. If using the YUV values, remember that U and V vary between (-0.5 to +0.5), so you will have to first add on a half to bring the range between 0 and 1, and then after dividing, subtract a half to bring in back into standard range.

The `JPG` or `PNG` image format will store each of the colors according to their image standard for color depth (e.g. `JPG` is 24-bit), but the image will be very very small since reducing color depth and quantizing colors are essentially what the `JPEG` compression algorithm accomplishes.



You do not have to reduce the color depth of each channel evenly. For example, if blue was the dominant color in an image, to preserve image quality, you could reduce Red to 2 values, Green to 4, and let the blue take on $256/(24)$ or 32 values. If using the HSV, you could reduce the Saturation and Value to 2 values (0 or 1.0) by Multiply by 2 and Divide by 2, and restrict the Hue to 64 possible values.

You can use this node setup to quantize any channel; alpha, speed (vector), z-values, and so forth.

RGB to BW Node

This node maps a RGB color image to a grayscale by the luminance.

Fig. 2.1692: RGB to BW Node.

Inputs

Image Color image input.

Properties

This node has no properties.

Outputs

Value Grayscale value output.

Value to Normal Node

Computes a normal map.

Inputs

Val The height map to compute the normal map from.

Nabla Size of derivative offset used for calculating normals.

Properties

This node has no properties.

Outputs

Normal Standard normal output.

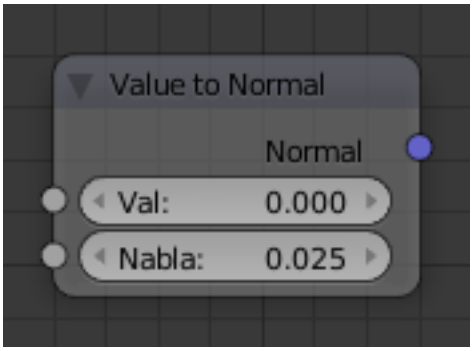


Fig. 2.1693: Value to Normal node.

Distort Nodes

These nodes allow you to change the mapping of a texture.

At Node

Returns the color of a texture at the specified coordinates.

Fig. 2.1694: At node.

Inputs

Texture Standard image input.

Coordinates The point at which to sample the color. For images, the space is between -1 and 1 for X and Y. If the coordinates are not spatially varying, the node will return a single color.

Properties

This node has no properties.

Outputs

Texture Standard image output.

Rotate Node

Rotate the texture coordinates of an image or texture.

Inputs

Color Standard image input.

Turns The number of times to rotate the coordinates 360 degrees about the specified axis.

Axis The axis to rotate the mapping about.

Properties

This node has no properties.

Outputs

Color Standard image output.

Scale Node

Scale the texture coordinates of an image or texture.

Inputs

Color Standard image input.

Scale The amount to scale the coordinates in each of the three axes.

Properties

This node has no properties.

Outputs

Color Standard image output.

Translate Node

Translate the texture coordinates of an image or texture.

Inputs

Color Standard image input.

Offset The amount to offset the coordinates in each of the three axes.

Properties

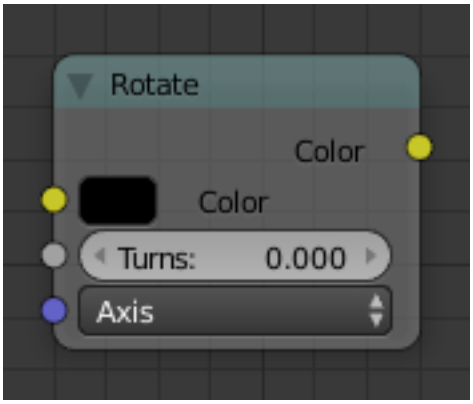
This node has no properties.

Outputs

Color Standard image output.

Input Nodes

Input nodes provide input data for other nodes.



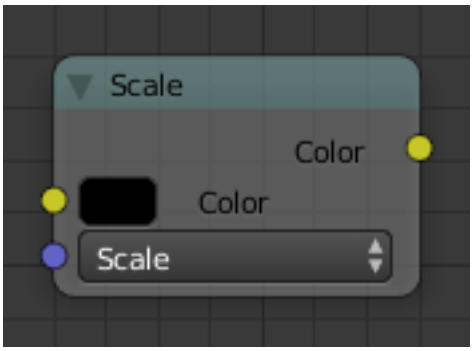


Fig. 2.1696: Scale node.

Fig. 2.1697: Translate node.

Coordinates Node

Inputs

This node has no inputs.

Properties

This node has no properties.

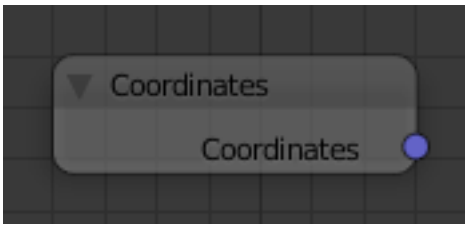


Fig. 2.1698: Coordinates node.

Outputs

Coordinates The Coordinates node outputs the geometry local coordinates, relative to its bounding box as RGB colors:

- Red channel corresponds to X value.
- Green channel corresponds to Y value.
- Green channel corresponds to Z value.

Image Node

The image node can be used to load an external image.

Inputs

This node has no inputs.

Properties

Image See *Data-Block Menu*.

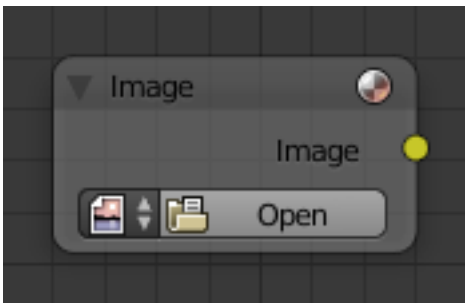


Fig. 2.1699: Image node.

Outputs

Color Standard color output.

Texture Node

The texture node can be used to load another node based or non-node based texture.

Inputs

These two colors can be used to remap a grayscale texture.

Color 1 White Level.

Color 2 Black Level.

Properties

Texture The texture could be selected from a list of textures available in the current blend-file or link in textures. The textures themselves could not be edited in this note, but in the Texture panel.

Outputs

Color Standard color output.

Time Node

The *Time node* generates a factor value (from 0.00 to 1.00) that changes according to the curve was drawn as time progresses through the *Timeline*.

Inputs

This node has no input sockets.

Properties

Curve The Y-value defined by the curve is the factor output. For the curve controls see: *Curve widget*.

Tip: Flipping the curve around reverses the time input, but doing so is easily overlooked in the node setup.

Start, End Start frame and End frame of the range of time specifying the values the output should last. This range becomes the X-axis of the graph. The time input could be reversed by specifying a start frame greater than the end frame.

Outputs

Factor A speed of time factor (from 0.00 to 1.00) relative to the frame rate defined in the *Render Dimensions Panel*. The factor changes according to the defined curve.

Hint: Output values

The *Map Value* node can be used to map the output to a more appropriate value. With sometimes curves, it is possible that the Time node may output a number larger than one or less than zero. To be safe, use the Min/Max clamping function of the Map Value node to limit output.

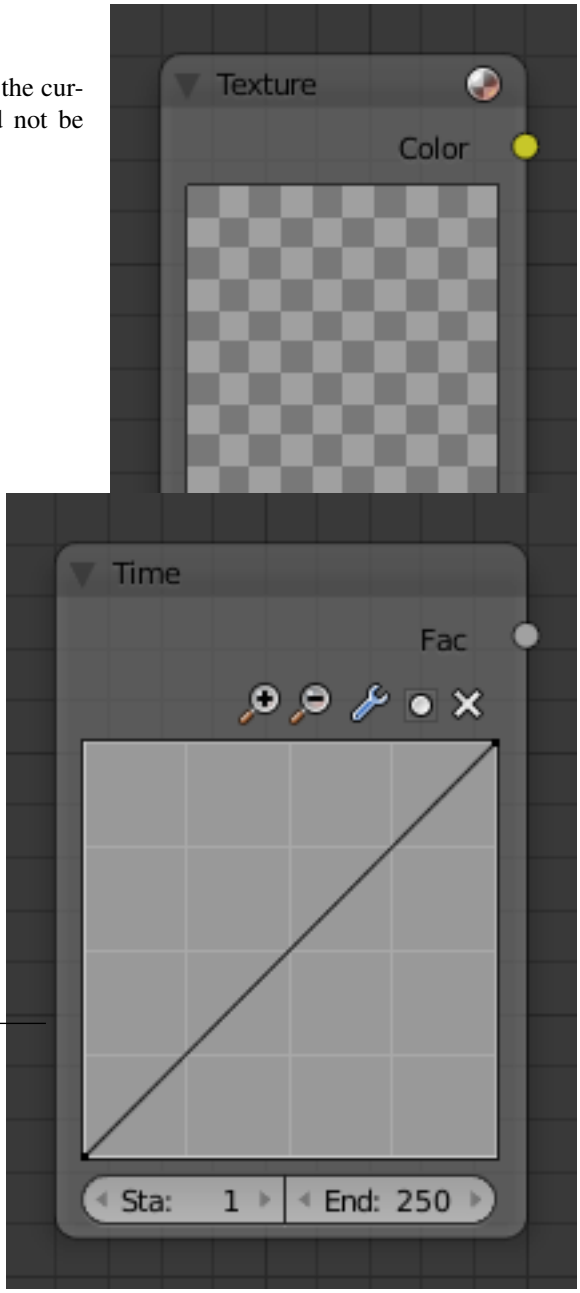


Fig. 2.1701: Time Node.

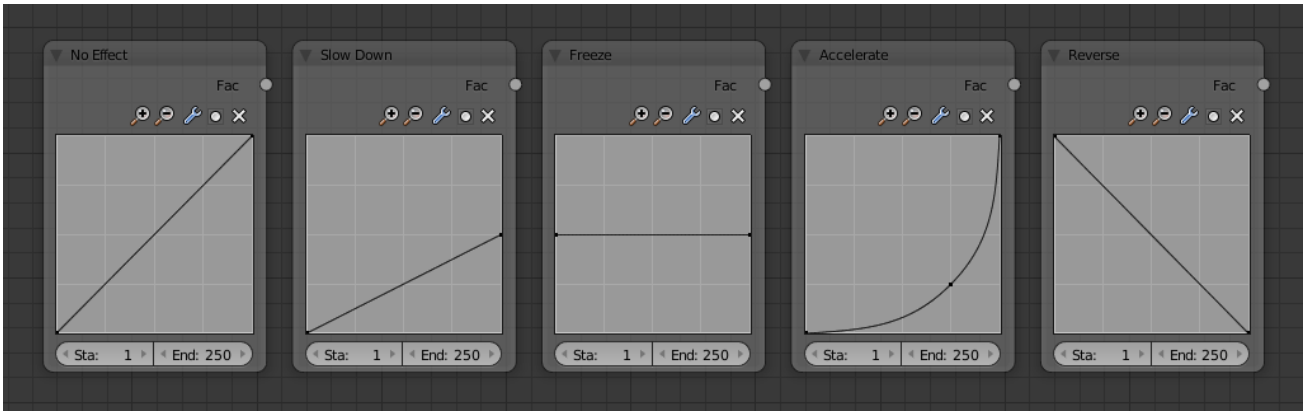


Fig. 2.1702: Time controls from left to right: no effect, slow down, freeze, accelerate, reverse

Example

Output Nodes

These nodes serve as an outputs for node textures.

Output Node

This node contains the result of the node texture.

Multiple output nodes can exist in a node texture, however, only one of them is active. The active one is set in the Texture Panel in the *Output* selector.

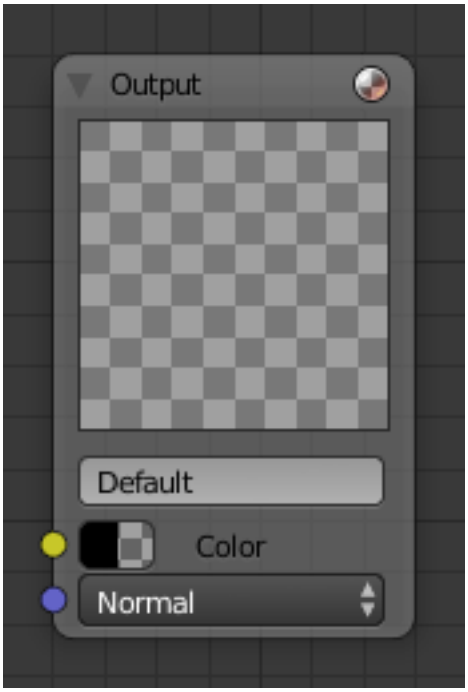


Fig. 2.1703: Output node.

Inputs

Color The color data that the texture renders.

Normal The normal map that the texture will output.

Properties

File path Output ID.

Outputs

This node has no outputs.

Viewer Node

The viewer node can be used to preview the results of a node.

Inputs

Color Standard image input.

Properties

This node has no inputs.

Outputs

This node has no outputs.

Pattern Nodes

Checker Node

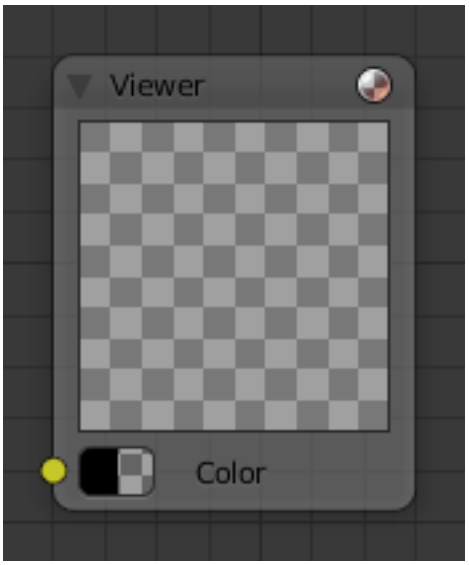


Fig. 2.1704: Viewer node.

The checker node creates a checkerboard pattern.

Inputs

color 1, color 2 Image inputs setting the color of the squares.

Size The scale of the checker pattern.

Properties

This node has no properties.

Outputs

Color Standard image output.

Bricks Node

The Bricks node creates a brick like pattern.

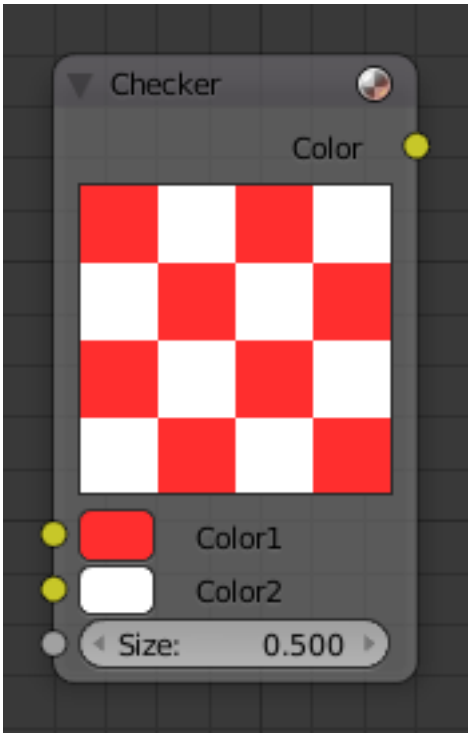


Fig. 2.1705: Checker node.

Inputs

Bricks 1, Bricks 2 Sets the color range of the bricks. Brick colors are chosen randomly between these two colors.

Mortar Sets the mortar color, in between the bricks.

Thickness Sets the thickness of the mortar.

Bias The bias of randomly chosen colors, between (-1 to 1). -1 Makes all bricks Color 1, and a value of 1 makes them all Color 2.

Brick Width Sets the horizontal size of all the bricks.

Row Height Sets the vertical size of all the bricks.

Properties

Offset The relative offset of the next row of bricks.

Frequency Offset every N rows. The brick pattern offset repeats every N rows.

Squash Scales the bricks in every N rows by this amount.

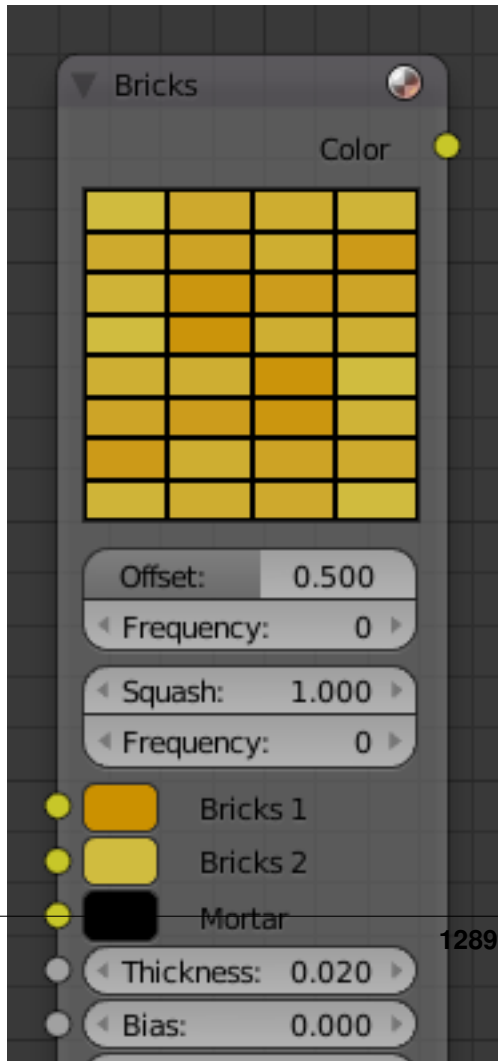
Frequency Squash every N rows.

Outputs

Color Standard color output.

Texture Nodes

These nodes generat procedural textures, and function just like their non node based counterparts.



Common Options

Color 1/Color 2 Remaps the procedural texture with these colors. These do not function in the Magic node.

Blend Node

See *Here*

Clouds Node

See *Here*

Distorted Noise Node

See *Here*

Fig. 2.1707: Blend node.

Magic Node

See *Here*

Marble Node

See *Here*

Musgrave Node

See *Here*

Noise Node

Noise

See *Here*

Stucci Node

See *Here*

Fig. 2.1708: Clouds node.

Fig. 2.1709: Distorted Noise node.

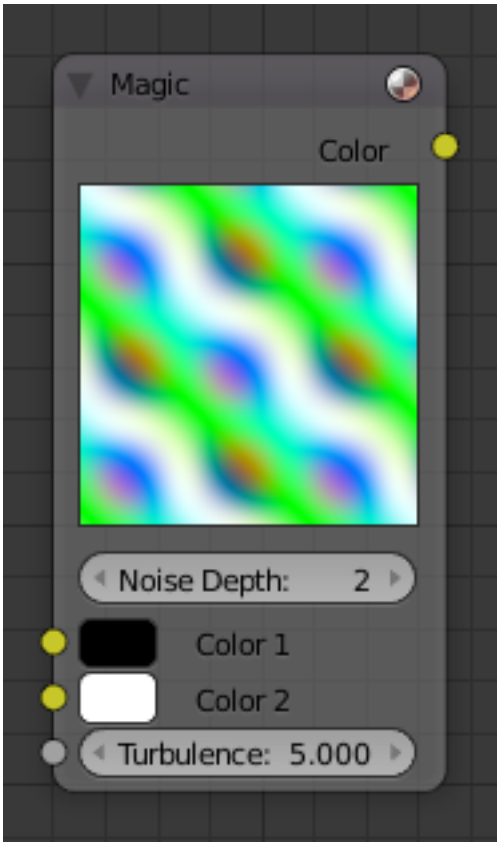


Fig. 2.1710: Magic node.

Voronoi Node

See *Here*

Wood Node

See *Here*

Lighting

Introduction

Lighting is a very important topic in rendering, standing equal to modeling, materials and textures. The most accurately modeled and textured scene will yield poor results without a proper lighting scheme, while a simple model can become very realistic if skillfully lit.

Viewing Restrictions

The color of an object and the lighting of your scene is affected by:

- Your ability to see different colors (partial color blindness is common).
- The medium in which you are viewing the image (e.g. an LCD panel versus printed glossy paper).

Fig. 2.1711: Marble node.

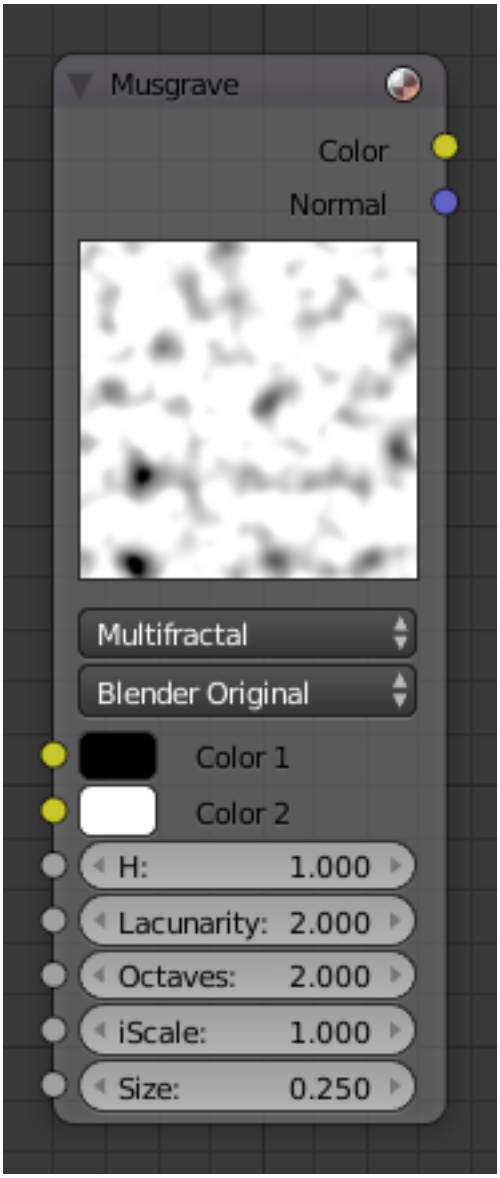


Fig. 2.1712: Musgrave.

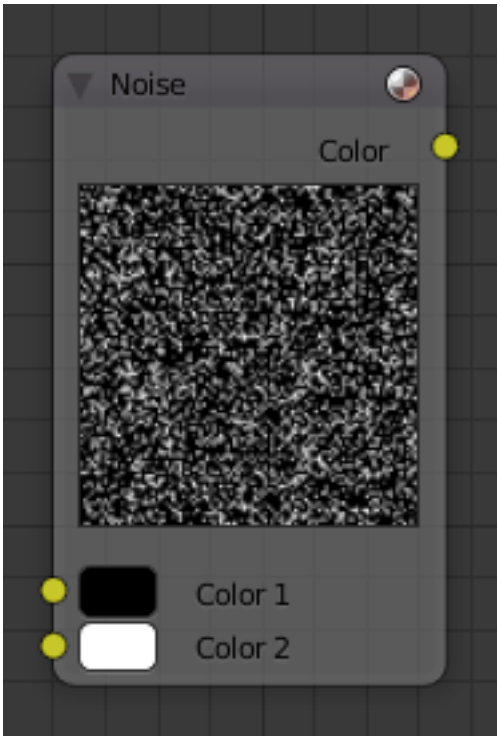


Fig. 2.1713: Noise.

- The quality of the image (e.g. a `jpeg` at 0.4 compression versus 1.0).
- The environment in which you are viewing the image (e.g. a CRT monitor with glare versus in a dark room, or in a sunshiny blue room).
- Your brain's perception of the color and intensity relative to those objects around it and the world background color, which can be changed using color manipulation techniques using Blender *Composite Nodes*.

Global Influences

In Blender, the elements under your control which affect lighting are:

- The color of the world *ambient light*.
- The use of *Ambient Occlusion* as a way to cast that ambient light onto the object.
- The degree to which the ambient light colors the *material* of the object.
- The use of *Indirect lighting*, where the color of one object radiates onto another.
- The *lamps* in your scene.

The physics of light bouncing around in the real world is simulated by Ambient Occlusion (a world setting), buffer shadows (which approximate shadows being cast by objects), ray tracing (which traces the path of photons from a light source). Also, within Blender you can use *Indirect lighting*. Ray tracing, ambient occlusion, and indirect lighting are computer-intensive processes. Blender can perform much faster rendering with its internal scan line renderer, which is a very good scan line renderer indeed. This kind of rendering engine is much faster since it does not try to simulate the real behavior of light, assuming many simplifying hypotheses.

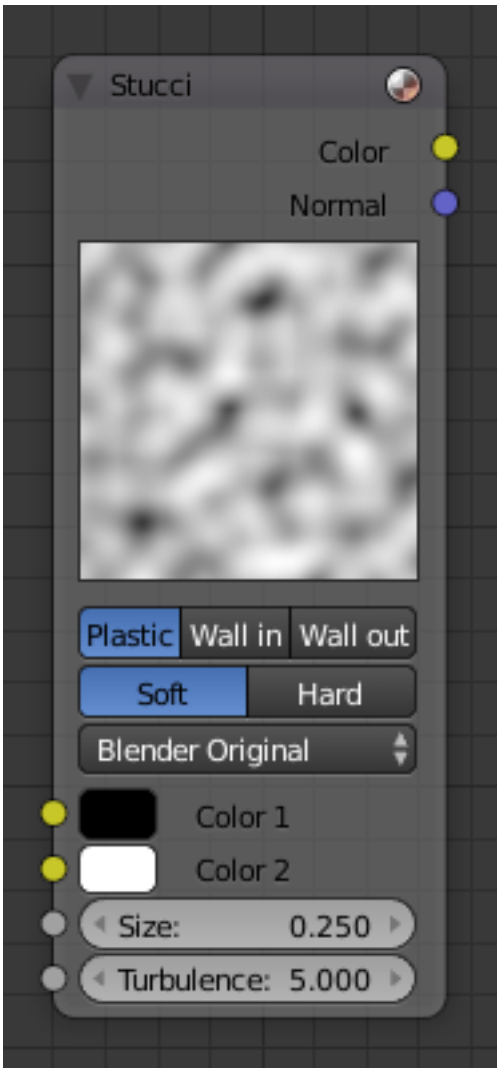


Fig. 2.1714: Stucci.

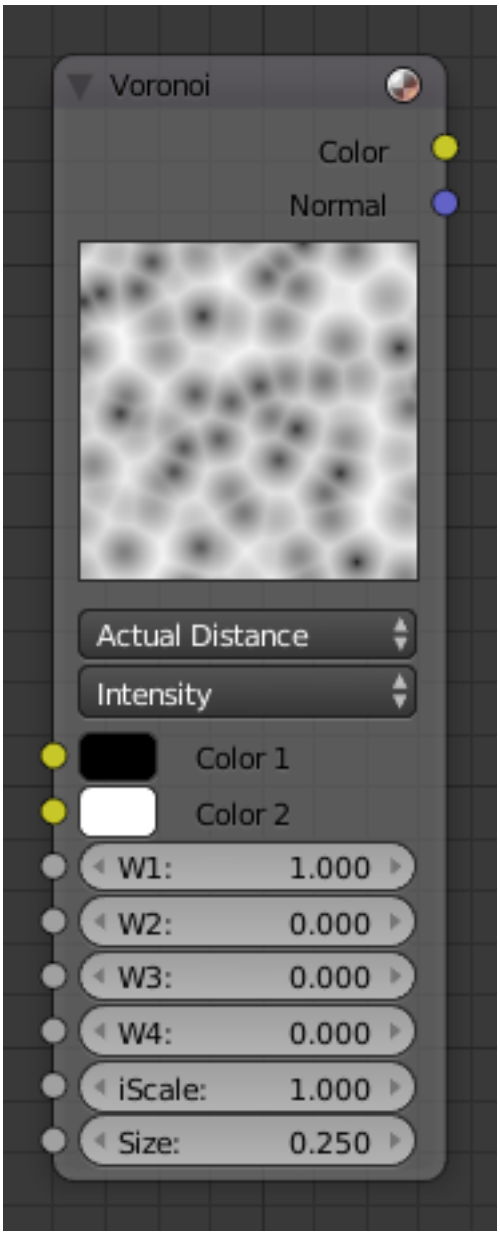


Fig. 2.1715: Voronoi node.

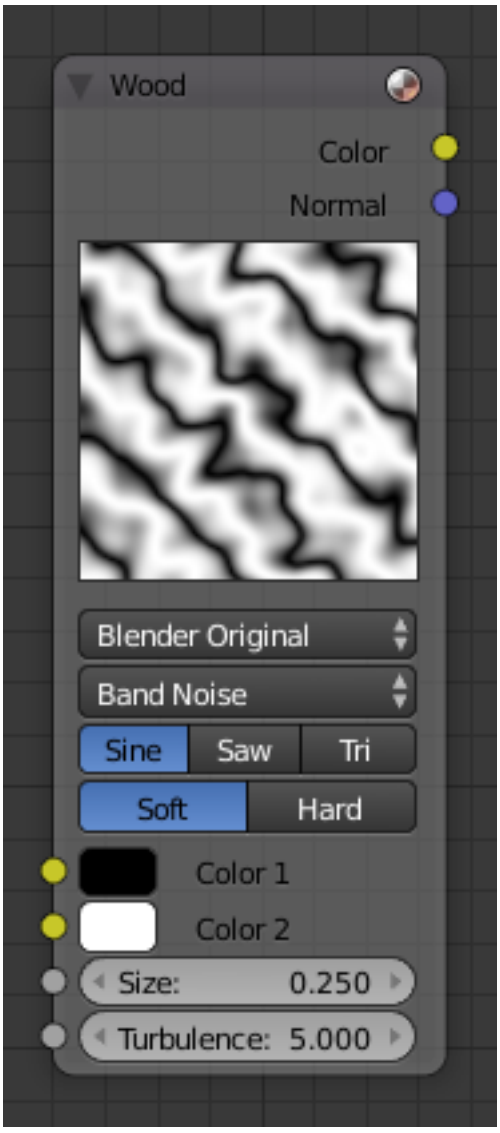


Fig. 2.1716: Wood node.

Lighting Settings

Only after the above global influences have been considered, do you start adding light from lamps in your scene. The main things under your control are the:

- Type of light used (*Sun*, *Spot*, *Lamp*, *Hemi*, etc.).
- Color of the light.
- Position of the light and its direction.
- Settings for the light, including energy and falloff.

Then you are back to how that material's *shader* reacts to the light.

This chapter attempts to address the above, including how lights can work together in rigs to light your scene. In this chapter we will analyze the different type of lights in Blender and their behavior; we will discuss their strong and weak points. We will also describe many lighting rigs, including the ever-popular three-point light method.

Lighting in the Workflow

In this user manual we have placed Lighting before Materials; you should set up your lighting before assigning materials to your meshes. Since the material shaders react to light, without proper lighting, the material shaders will not look right, and you will end up fighting the shaders, when it is really the bad lighting that is causing you grief. All of the example images in this section do not use any material setting at all on the ball, cube or background.

Overriding Materials to Reset Lighting

If you have started down the road of assigning materials, and are now fiddling with the lighting, we suggest that you create a default, generic gray material – no *Vertex Color*, no *Face Texture*, no *Shadeless*, just plain old middle gray with RGB(0.8, 0.8, 0.8). Name this "Gray".

Next go to the *Render Layer* tab. In the *Layer* panel, select your new "Gray" material in the *Material* field. This will override any materials you may have set, and render everything with this color. Using this material, you can now go about adjusting the lighting. Just empty this field to get back to your original materials.

Lights

Lamp Panel

Lamp A *Data-Block Menu*. Its list shows all light settings used in the current scene.

Texture Count Shows the count of textures in the lamp texture stack.

Preview

A quick preview of the light settings.

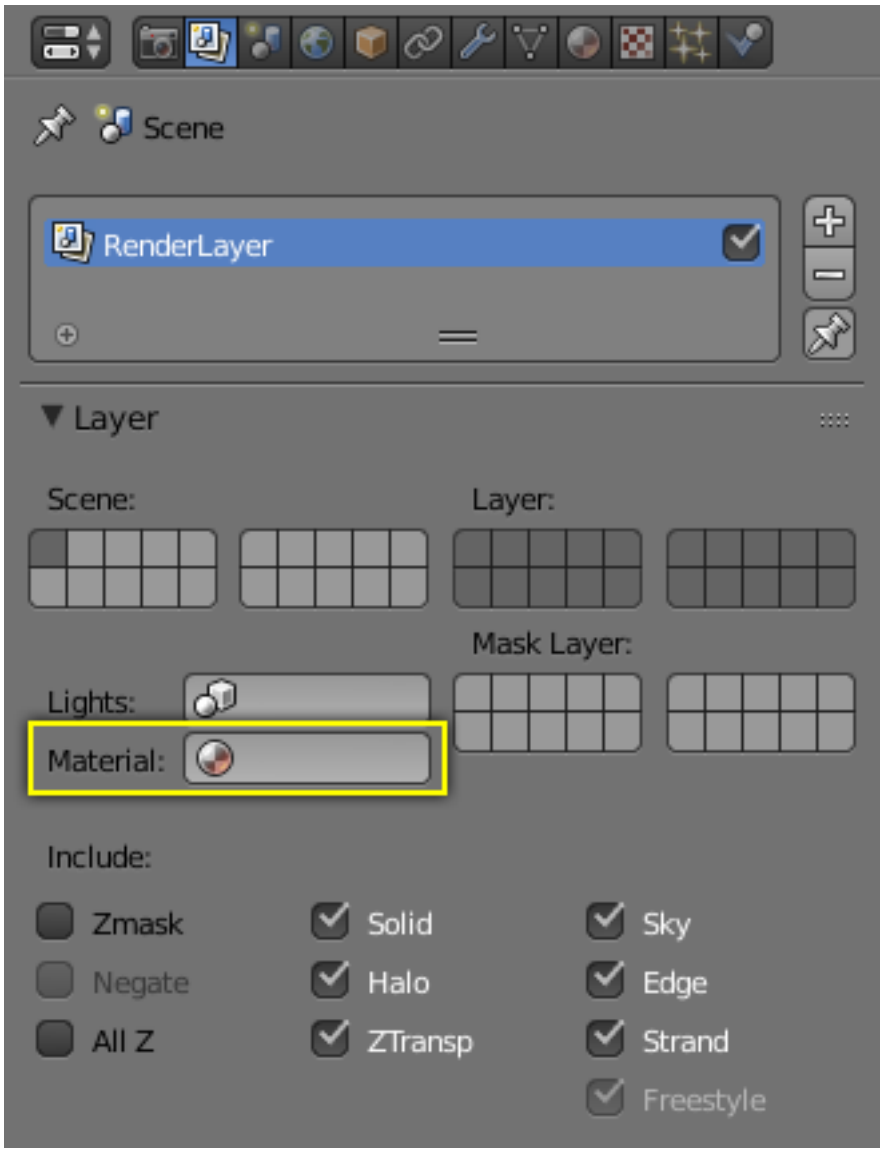


Fig. 2.1717: Material field in the Render Layers panel.

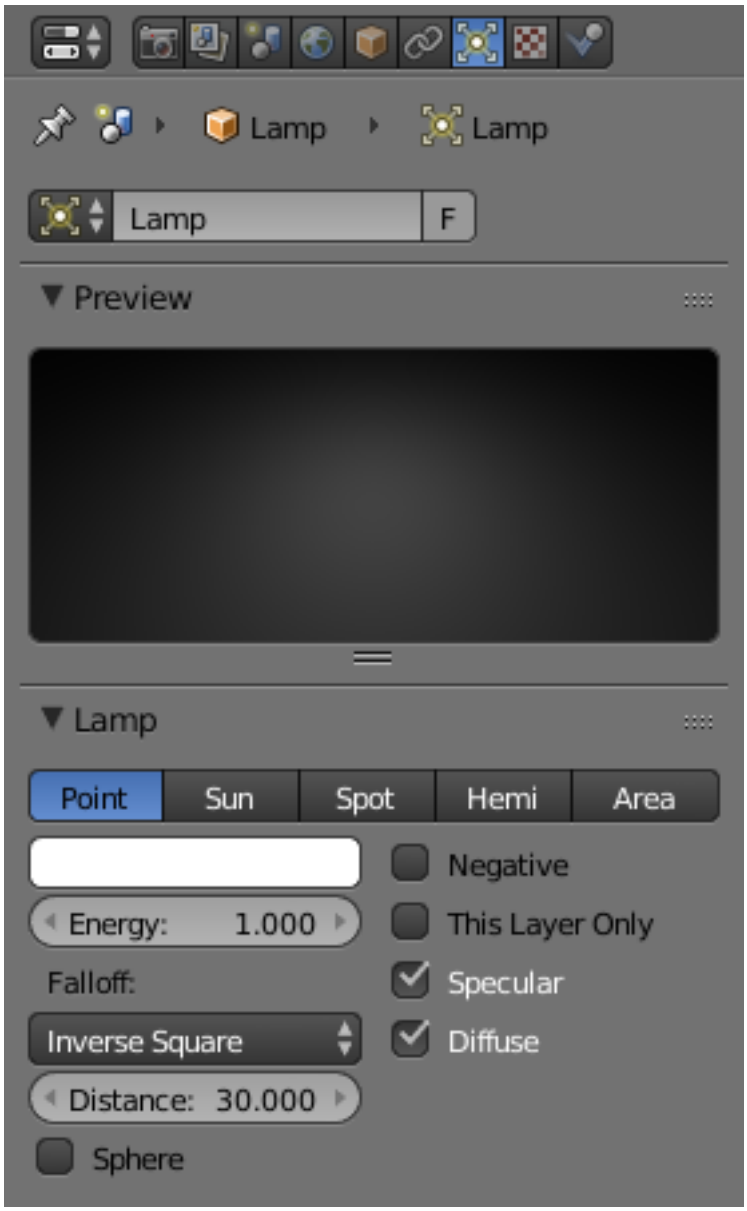


Fig. 2.1718: Lamp tab.

Lamp

Type *Types of lamps* available in Blender Internal. They share all or some of the options listed here:

Color The color of the light source's illumination.

Energy The intensity of the light source's illumination from (0.0 to 10.0).

Falloff See *Light Attenuation*.

Distance The *Distance* number button indicates the number of Blender Units (BU) at which the intensity of the current light source will be half of its intensity. Objects less than the number of BU away from the lamp will get more light, while objects further away will receive less light. Certain settings and lamp falloff types affect how the *Distance* is interpreted, meaning that it will not always react the same; see the page about *light falloff*.

The *Sun* and *Hemi* Lamps are another class of Lamps which uses a constant falloff. Those lamps do not have a *Distance* parameter, and are often called "Base Lighting Lamps".

Influence

Every lamp has a set of switches that control which objects receive its light, and how it interacts with materials.

Negative Let the lamp cast negative light. The light produced by the lamp is *subtracted* from the irradiance on the surfaces it hits, which darkens these surfaces instead of brightening them.

This Layer Only The Lamp only illuminates objects on the same layer the lamp is on. Causes the lamp to only light objects on the same layer.

Specular The Lamp creates specular highlights.

Diffuse The Lamp affects diffuse shading.

Light Attenuation

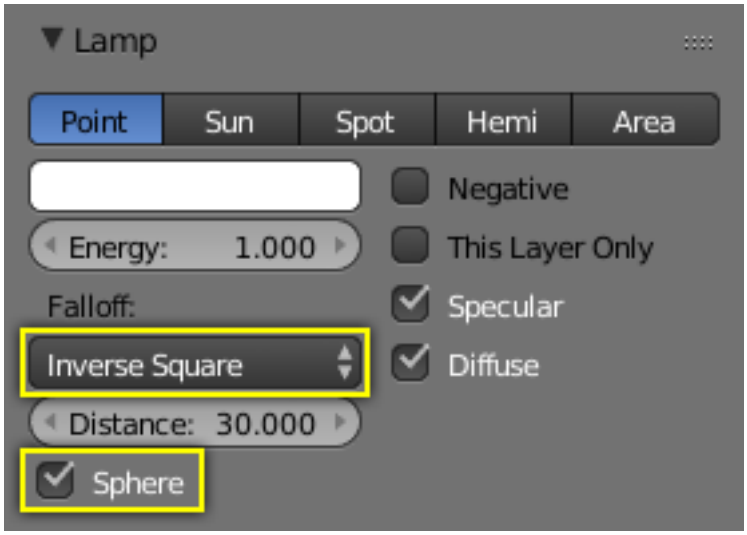


Fig. 2.1719: Lamp panel, falloff options highlighted.

There are two main controls for light falloff for *Point* and *Spot* lamps:

- The lamp *Falloff* type selector.
- And the *Sphere* checkbox.

Falloff types

Lin/Quad Weighted

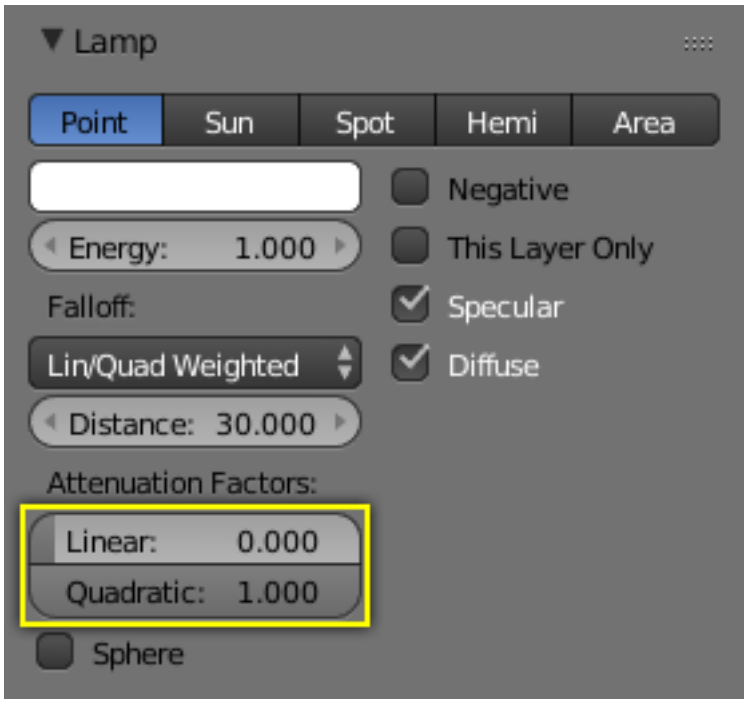


Fig. 2.1720: Lamp panel with Lin/Quad Weighted Falloff options highlighted.

When this setting is chosen, two sliders are shown, *Linear* and *Quadratic*, which control respectively the "linearness" and "quadraticness" of the falloff curve.

This lamp falloff type is in effect allowing the mixing of the two light attenuation profiles (linear and quadratic attenuation types).

Linear

This slider input field can have a value between (0.0 to 1.0). A value of 1.0 in the *Linear* field and 0.0 in the *Quadratic* field in effect means that the light from this source is completely linear. This means that at the number of Blender Units distance specified in the *Distance* field, this light source's intensity will be half the value it was originally.

When the *Quadratic* slider is set to 0.0, the formula for working out the attenuation at a particular range for full linear attenuation is:

$I = E(D/(D + Lr))$

Where:

- I is the calculated Intensity of light.
- E is the current *Energy* slider setting.

- *D* is the current setting of the *Distance* field.
- *L* is the current setting of the *Linear* slider.
- *r* is the distance from the lamp where the light intensity gets measured.

Quadratic

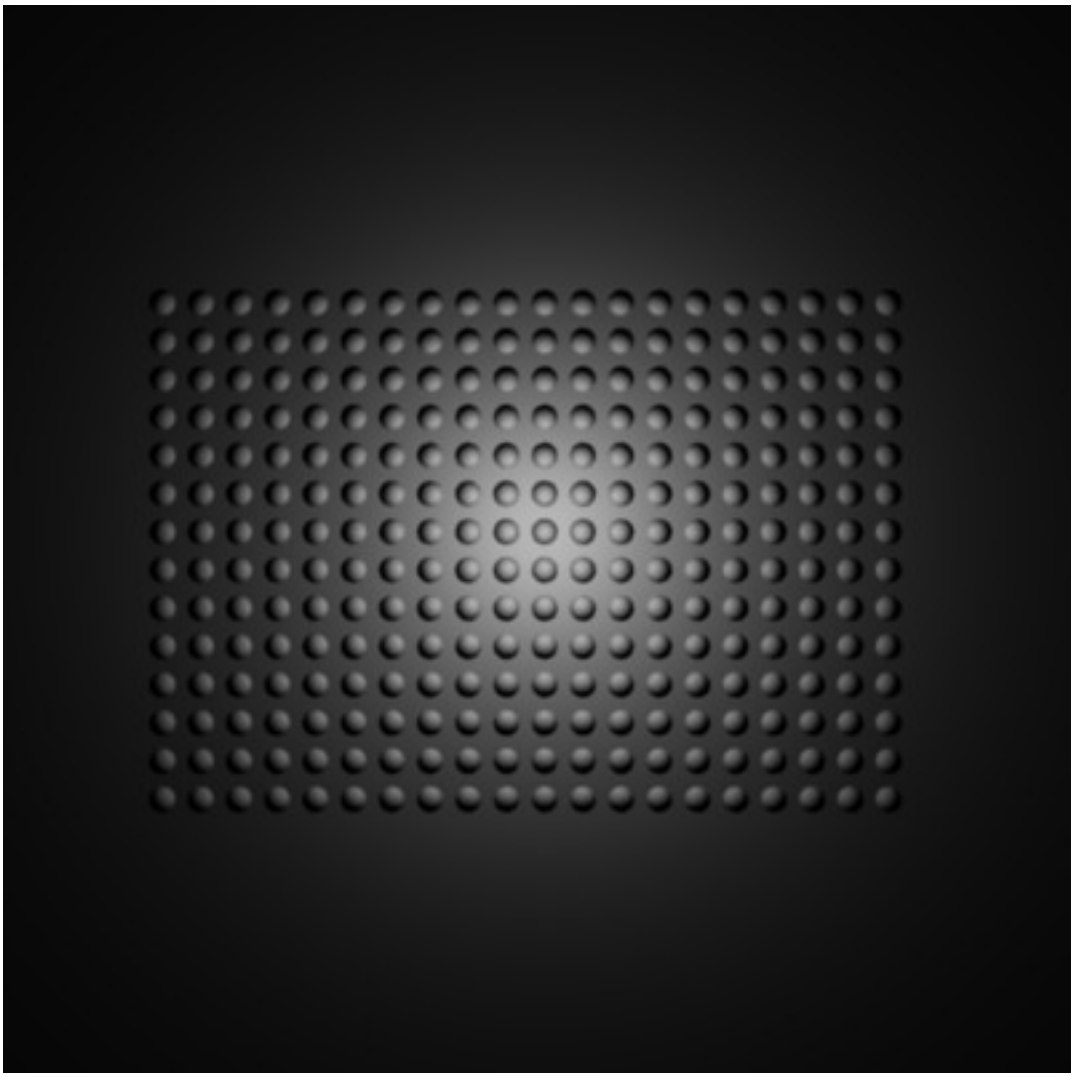


Fig. 2.1721: Lamp with Lin/Quad Weighted falloff default settings.

This slider input field can have a value between (0.0 to 1.0). A value of 1.0 in the *Quadratic* field and 0.0 in the *Linear* field means that the light from this source is completely quadratic.

Quadratic attenuation type lighting is considered a more accurate representation of how light attenuates (in the real world). In fact, fully quadratic attenuation is selected by default. For *Lin/Quad Weighted* lamp fallout see Fig. *Lamp with Lin/Quad Weighted falloff default settings.*.

Here again, the light intensity is half when it reaches the *Distance* value from the lamp. Comparing the quadratic falloff to the linear falloff, the intensity decays much slower at distances lower than the set *Distance*, but it attenuates much quicker after *Distance* is reached.

When the *Linear* slider is set to 0.0, the formula for working out the attenuation at a particular range for full quadratic attenuation is:

$I = E(D^2/(D^2 + Qr^2))$

Where:

- *I* is the calculated Intensity of light.
- *E* is the current *Energy* slider setting.
- *D* is the current setting of the *Distance* field.
- *Q* is the current setting of the *Quad* slider.
- *r* is the distance from the lamp where the light intensity gets measured.

Mixing "Linear" and "Quad"

If both the *Linear* and *Quad* slider fields have values greater than 0.0, then the formula used to calculate the light attenuation profile changes to this:

$I = E(D/(D + Lr))(D^2/(D^2 + Qr^2))$

Where:

- *I* is the calculated Intensity of light.
- *E* is the current *Energy* slider setting.
- *D* is the current setting of the *Distance* field.
- *L* is the current setting of the *Linear* slider.
- *Q* is the current setting of the *Quad* slider.
- *r* is the distance from the lamp where the light intensity gets measured.

Zeroing both "Linear" and "Quad"

If both the *Linear* and *Quadratic* sliders have 0.0 as their values, the light intensity will not attenuate with distance. This does not mean that the light will not get darker, rather it will, but only because the energy the light has is spread out over a wider and wider distance. The total amount of energy in the spread-out light will remain the same, though. The light angle also affects the amount of light you see. It is in fact the behavior of light in the deep space vacuum.

If what you want is a light source that does not attenuate and gives the same amount of light intensity to each area it hits, you need a light with properties like the *Constant* lamp *Falloff* type.

Also, when the *Linear* and *Quad* sliders are both 0.0 values the *Distance* field ceases to have any influence on the light attenuation, as shown by the equation above.

Graphical Summary

Below is a graph summarizing the lin/quad attenuation type, showing attenuation with or without the *Sphere* option (described later).

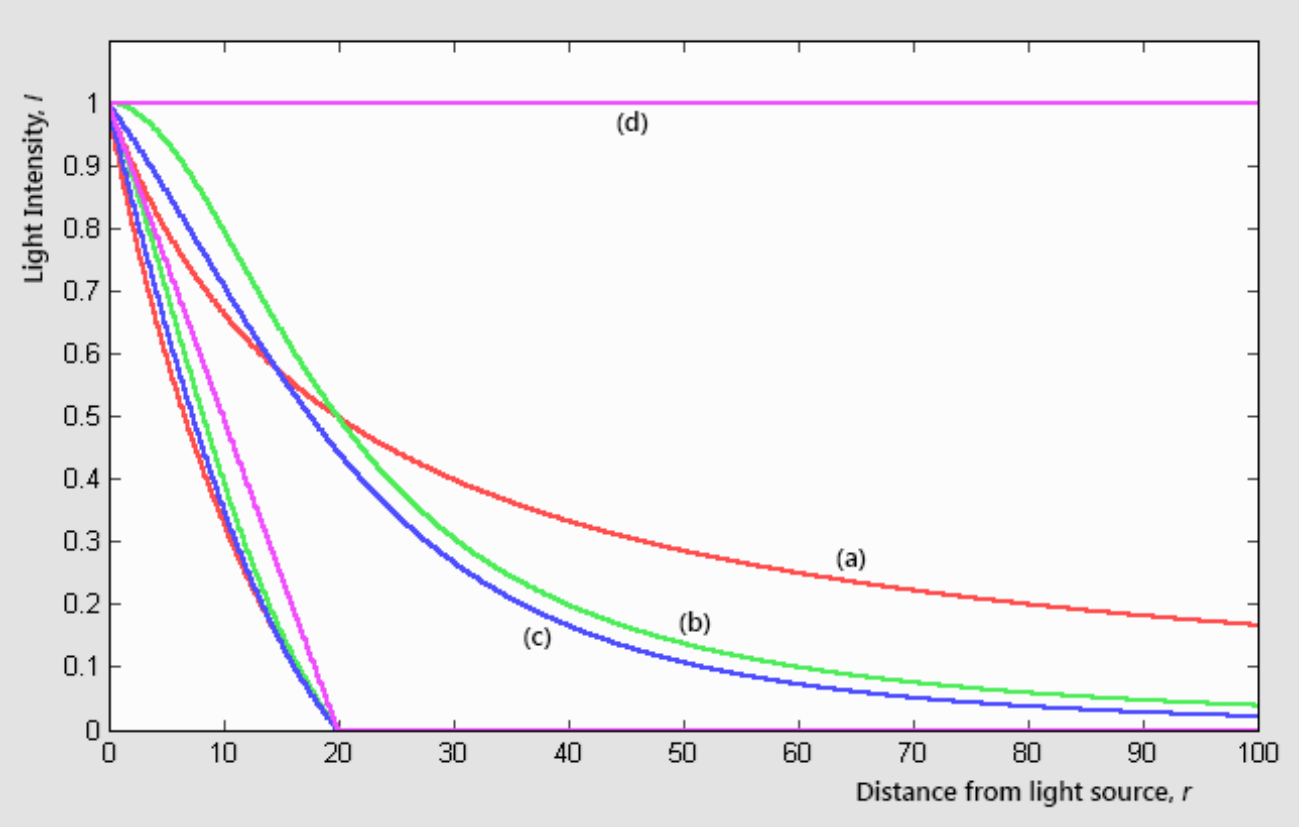


Fig. 2.1722: Light Attenuation:

1. Linear (Linear=1.0, Quad=0.0);
2. Quadratic (Linear=0.0, Quad=1.0);
3. Linear and quadratic (Linear=Quad=0.5);
4. Null (Linear=Quad=0.0);

Also shown in the graph the “same” curves, in the same colors, but with the Sphere button turned on.

Custom Curve

The *Custom Curve* lamp *Falloff* type is very flexible.

Most other lamp falloff types work by having their light intensity start at its maximum (when nearest to the light source) and then with some predetermined pattern decrease their light intensity when the distance from the light source increases.

When using the *Custom Curve* Lamp Falloff type, a new panel is created called *Falloff Curve*. This *Falloff Curve* profile graph allows the user to alter how intense light is at a particular point along a light's attenuation profile (i.e. at a specific distance from the light source).

The *Falloff Curve* profile graph has two axes, the Distance-axis and the Intensity-axis.

Distance axis It represents the position at a particular point along a light source's attenuation path. The far left is at the position of the light source and the far right is the place where the light source's influence would normally be completely attenuated.

Intensity axis It represents the intensity at a particular point along a light source's attenuation path. Higher intensity is represented by being higher up the intensity axis, while lower intensity light is represented by being lower down on the intensity axis.

Altering the *Falloff Curve* profile graph is easy. Just `LMB` click on a part of the graph you want to alter and drag it where you want it to be. If when you click you are over or near one of the tiny black square handles, it will turn white, indicating that this handle is now selected, and you will be able to drag it to a new position. If when you click on the graph you are not near a handle, one will be created at the point that you clicked, which you can then drag where you wish. You can also create handles at specific parts of the graph, clicking with `LMB` while holding `Ctrl`; it will create a new handle at the point you have clicked.

In the example below (the default for the *Falloff Curve* Profile Graph), the graph shows that the intensity of the light starts off at its maximum (when near the light), and linearly attenuates as it moves to the right (further away from the light source).

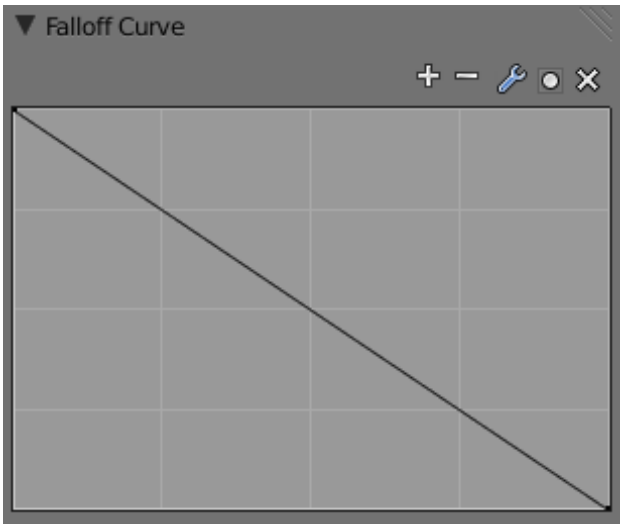


Fig. 2.1723: Default Falloff Curve panel graph.

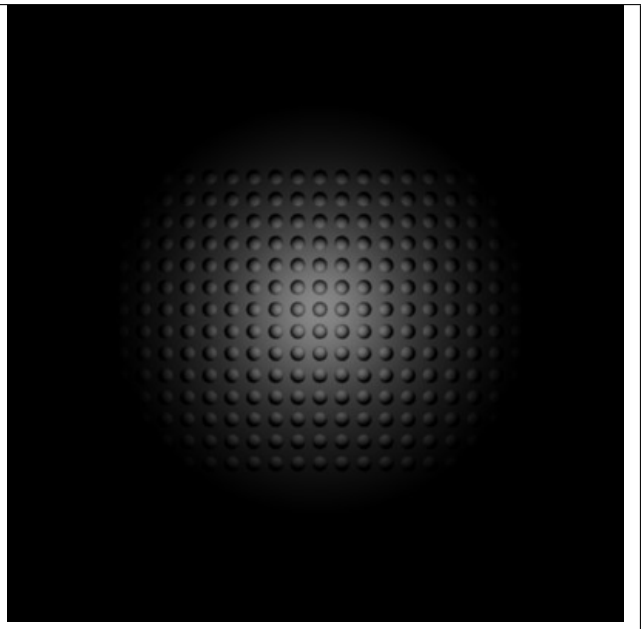


Fig. 2.1724: Render showing the Custom Curve lamp falloff type effect with default settings.

If you want to have a light attenuation profile that gets more intense as it moves away from the light source, you could alter the graph as below:

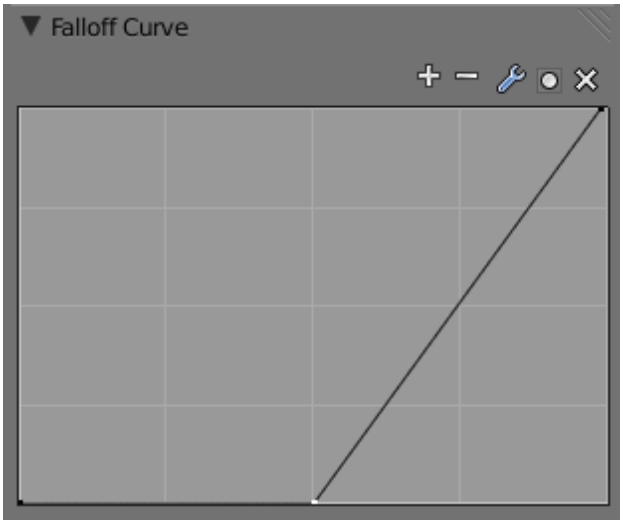


Fig. 2.1725: Falloff Curve for reversed attenuation.

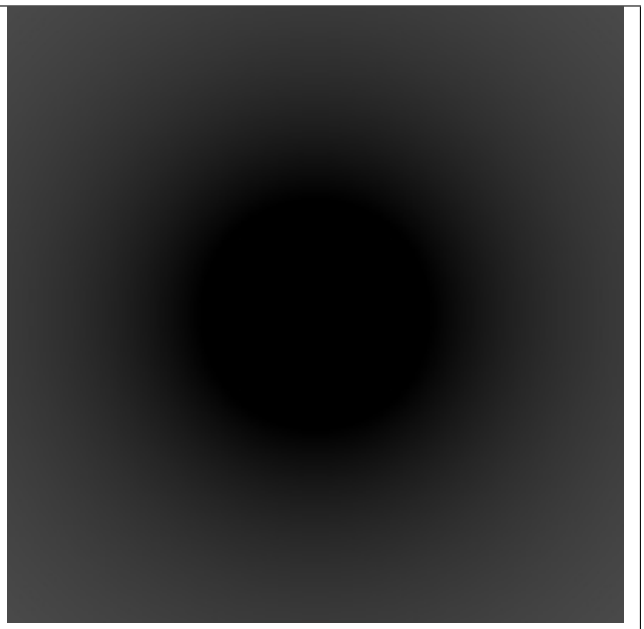


Fig. 2.1726: Falloff Curve for reversed attenuation rendered.

You are obviously not just limited to simple changes such as reversing the attenuation profile, you can have almost any profile you desire.

Here is another example of a different *Falloff Curve* profile graph, along with its resultant render output:

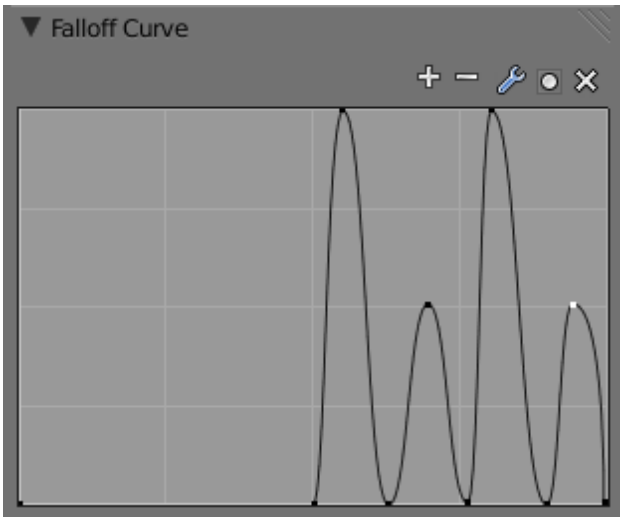


Fig. 2.1727: Oscillating attenuation profile.

Fig. 2.1728: Render showing the effects of a "wavelet" profile graph on the light attenuation.

Inverse Square

This lamp falloff type attenuates its intensity according to inverse square law, scaled by the *Distance* value. Inverse square is a sharper, realistic decay, useful for lighting such as desk lamps and street lights. This is similar to the old *Quad* option (and consequently, to the new *Lin/Quad Weighted* option with *Linear* to 0.0 and *Quad* to 1.0), with slight changes.

Inverse Linear

This lamp falloff type attenuates its intensity linearly, scaled by the *Distance* value. This is the default setting, behaving the same as the default in previous Blender versions without *Quad* switched on, and consequently, like the new *Lin/Quad Weighted* option with *Linear* to 1.0 and *Quad* to 0.0. This is not physically accurate, but can be easier to light with.

Constant

This lamp falloff type does not attenuate its intensity with distance. This is useful for distant light sources like the sun or sky, which are so far away that their falloff is not noticeable. *Sun* and *Hemi* lamps always have constant falloff.

Inverse Coefficients

This lamp falloff type combines the *Inverse Square*, *Inverse Linear* and *Constant* modes into a single inverse-quadratic formula:

$I = E(1.0/(Qr^2 + Lr + C))$

Where:

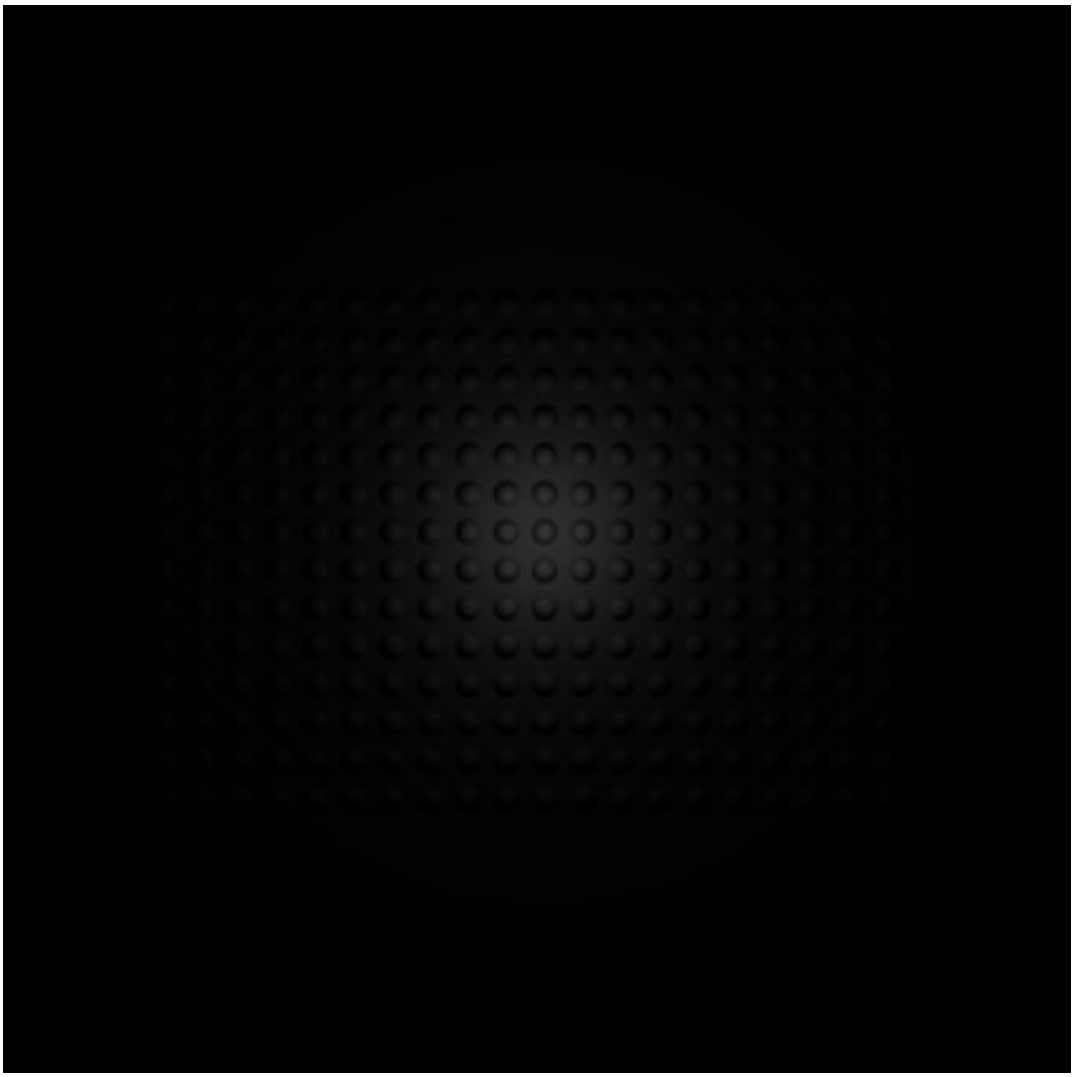


Fig. 2.1729: Render showing the Inverse Square lamp falloff type effect with default settings.

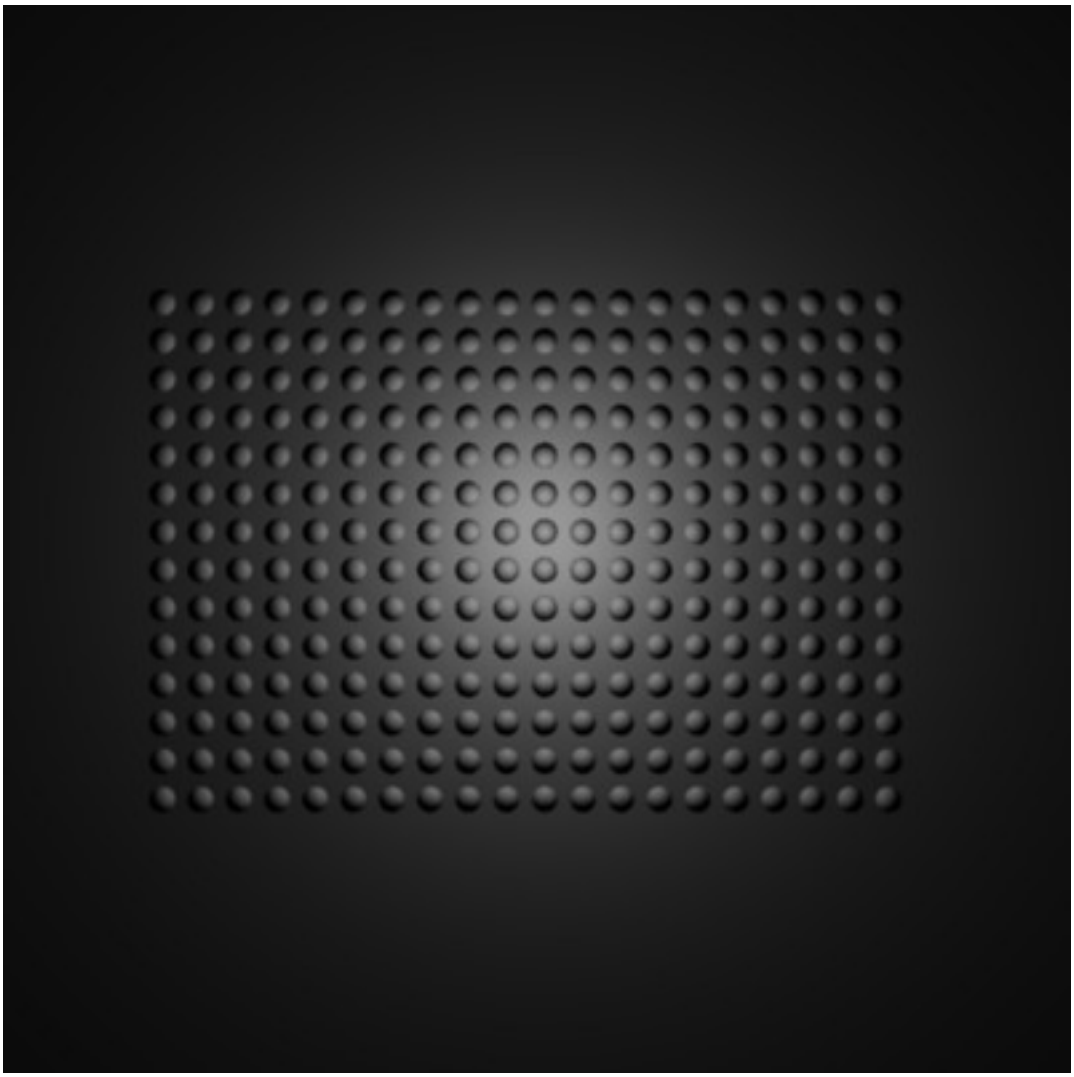


Fig. 2.1730: Render showing the Inverse Linear lamp falloff type effect with default settings.

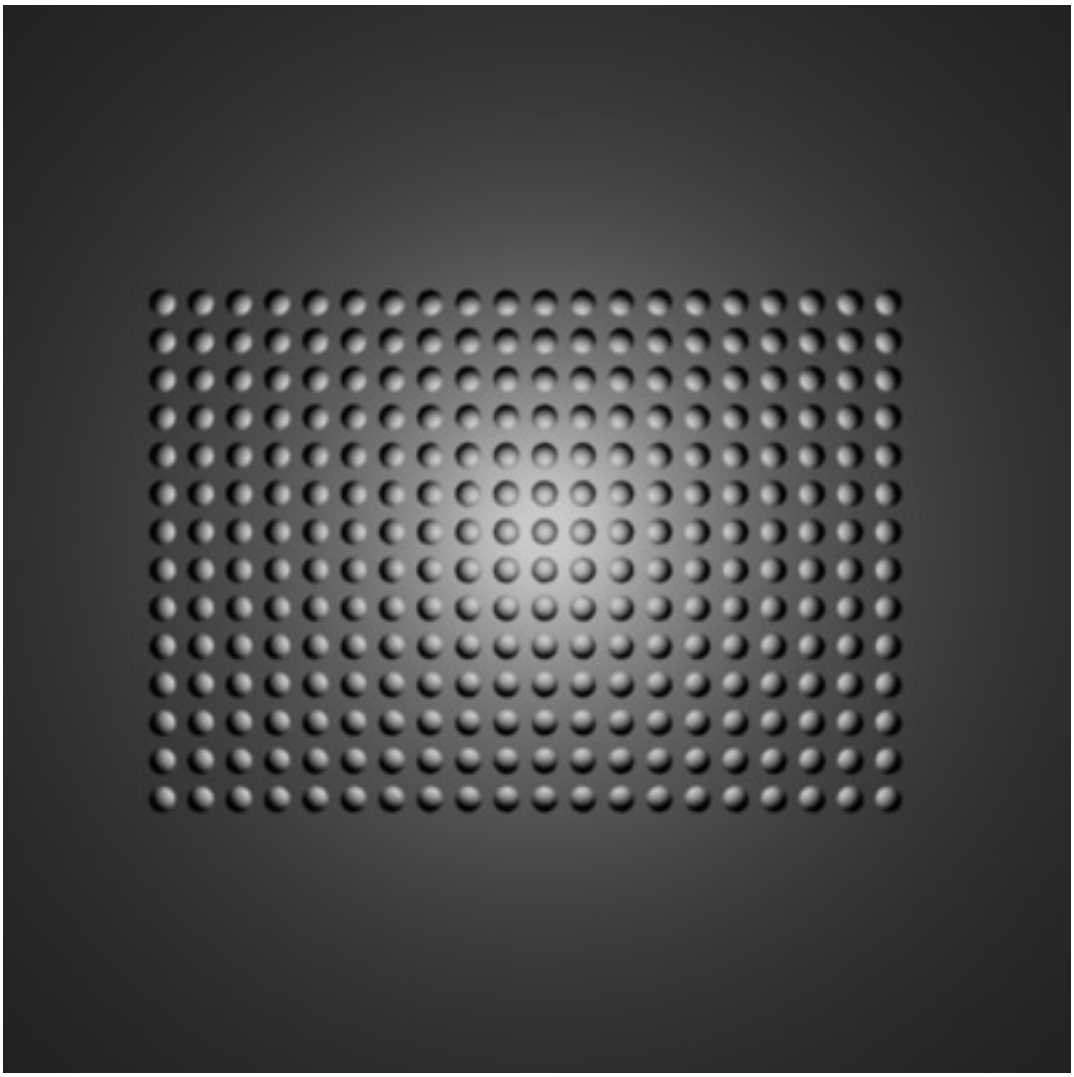


Fig. 2.1731: Render showing the Constant lamp falloff type effect with default settings.

- *I* is the calculated Intensity of light.
- *E* is the current *Energy* slider setting.
- *C* is the current setting of the *Constant* slider setting.
- *L* is the current setting of the *Linear* slider setting.
- *Q* is the current setting of the *Quadratic* slider setting.
- *r* is the distance from the lamp where the light intensity gets measured.

Such a falloff model is commonly used in real-time rendering applications via a shading language like GLSL.

Sphere

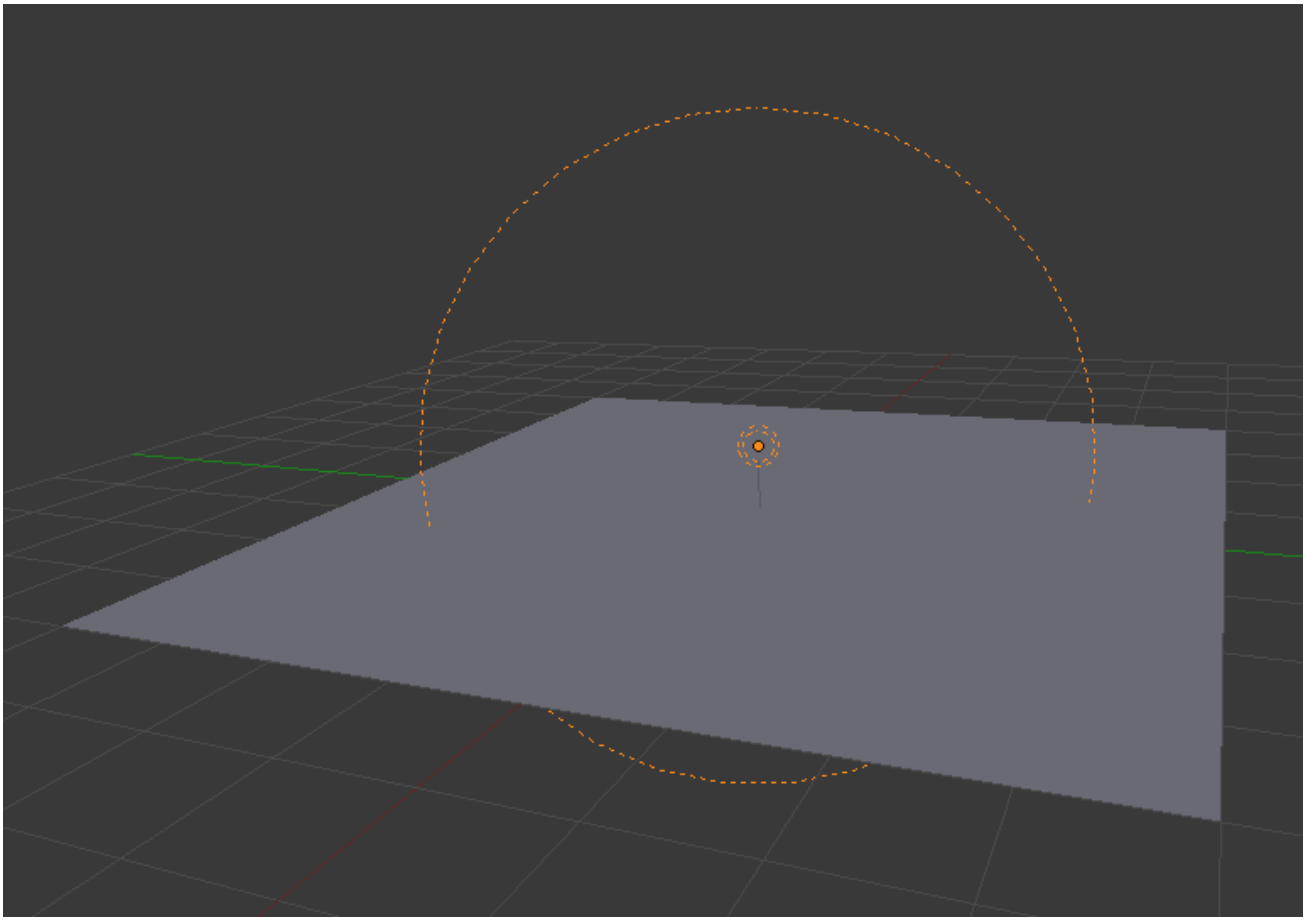


Fig. 2.1732: Screenshot of the 3D View editor, showing the Sphere light clipping circle.

The *Sphere* option restricts the light illumination range of a *Lamp* or *Spot* lamp, so that it will completely stop illuminating an area once it reaches the number of Blender Units away from the Lamp, as specified in the *Distance* field.

When the *Sphere* option is active, a dotted sphere will appear around the light source, indicating the demarcation point at which this light intensity will be null.

The *Sphere* option adds a term to the chosen attenuation law, whatever it is:

$I' = I(D - r)/D$ if $r < D$;

$I' = 0$ otherwise;

Where:

- *I'* is the required Intensity of light (with the *Sphere* option activated).
- *I* is the intensity of light calculated by the chosen attenuation law (without the *Sphere* option).
- *D* is the current setting of the *Distance* field.
- *r* is the distance from the lamp where the light intensity gets measured.

See the graphic at the end of the description of the *Lin/Quad Weighted* attenuation option.

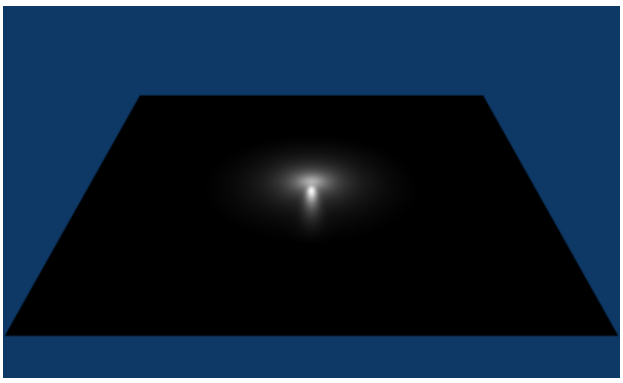


Fig. 2.1733: Render showing the light attenuation of a Constant falloff light type with the Sphere option active.

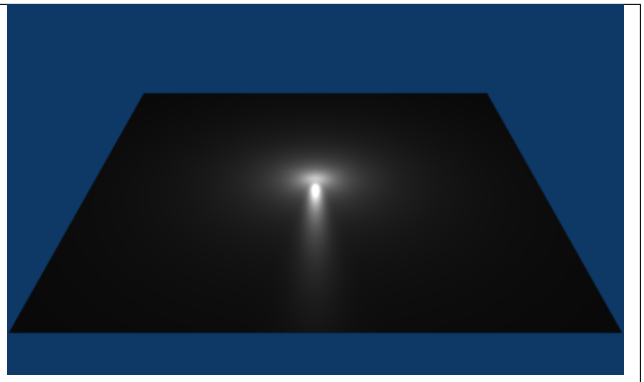


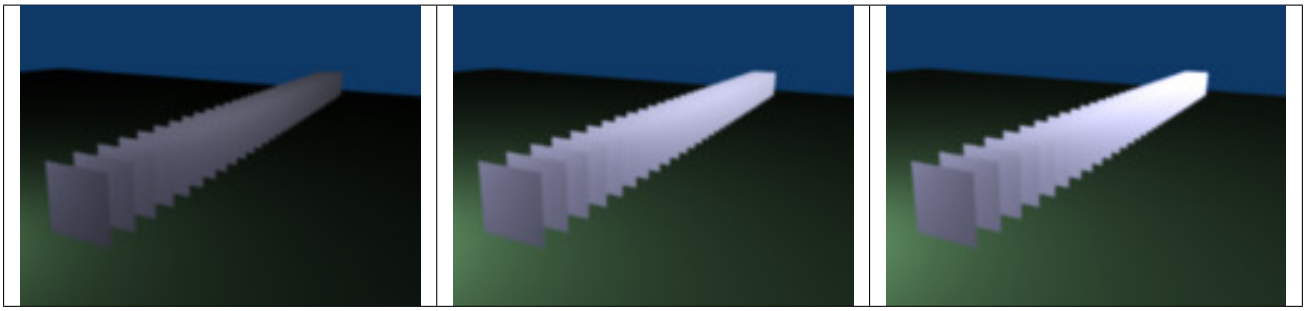
Fig. 2.1734: Render showing the light attenuation of a Constant falloff light type with the Sphere option deactivated.

Examples

Distance Example

In this example, the *Lamp* has been set pretty close to the group of planes. This causes the light to affect the front, middle and rear planes more dramatically. Looking at the figure below, you can see that as the Distance is increased, more and more objects become progressively brighter.

Table 2.82: Distance: 1000.



The *Distance* parameter is controlling where the light is falling – at a linear rate by default – to half its original value from the light's origin. As you increase or decrease this value, you are changing where this half falloff occurs. You could think of *Distance* as the surface of a sphere and the surface is where the light's intensity has fallen to half its strength in all

directions. Note that the light's intensity continues to fall even after *Distance*. *Distance* just specifies the distance where half of the light's energy has weakened.

Notice in Fig. *Distance: 1000.*, that the farthest objects are very bright. This is because the falloff has been extended far into the distance, which means the light is very strong when it hits the last few objects. It is not until 1000 units that the light's intensity has fallen to half of its original intensity.

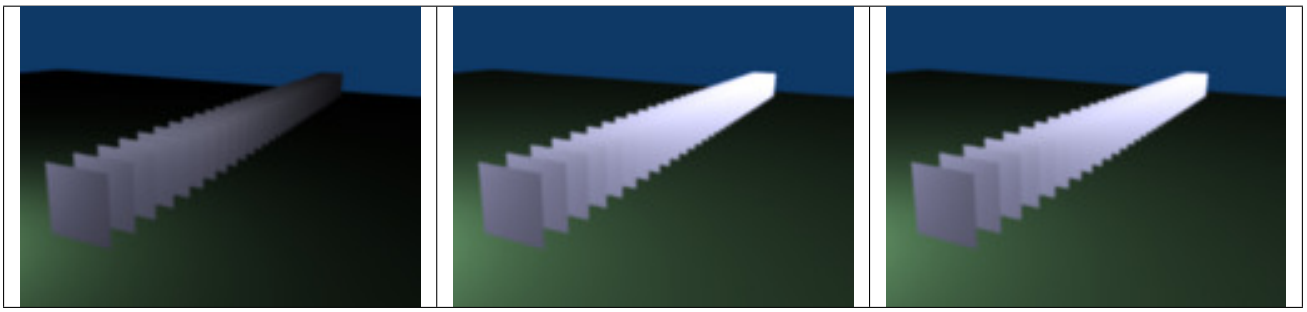
Contrast this with Fig. *Distance: 100.*, where the falloff occurs so soon that the farther objects are barely lit. The light's intensity has fallen by a half by time it even reaches the tenth object.

You may be wondering why the first few planes appear to be dimmer? This is because the surface angle between the light and the object's surface normal is getting close to oblique. That is the nature of a *Lamp* light object. By moving the light infinitely far away you would begin to approach the characteristics of the *Sun* lamp type.

Inverse Square Example

Inverse Square makes the light's intensity falloff with a non-linear rate, or specifically, a quadratic rate. The characteristic feature of using *Inverse Square* is that the light's intensity begins to fall off very slowly but then starts falling off very rapidly. We can see this in the Fig. *Inverse Square selected. (with the specified distances).* images.

Table 2.83: Inverse Square with 1000.



With *Inverse Square* selected, the *Distance* field specifies where the light begins to fall off faster, roughly speaking; see the light attenuation description in *Falloff types* for more info.

In Fig. *Inverse Square with 10.*, the light's intensity has fallen so quickly that the last few objects are not even lit.

Both Fig. *Inverse Square with 100.* and Fig. *Inverse Square with 1000.* appear to be almost identical and that is because the *Distance* is set beyond the farthest object's distance which is at about 40 BU out. Hence, all the objects get almost the full intensity of the light.

As above, the first few objects are dimmer than farther objects because they are very close to the light. Remember, the brightness of an object's surface is also based on the angle between the surface normal of an object and the ray of light coming from the lamp.

This means there are at least two things that are controlling the surface's brightness: intensity and the angle between the light source and the surface's normal.

Sphere Example

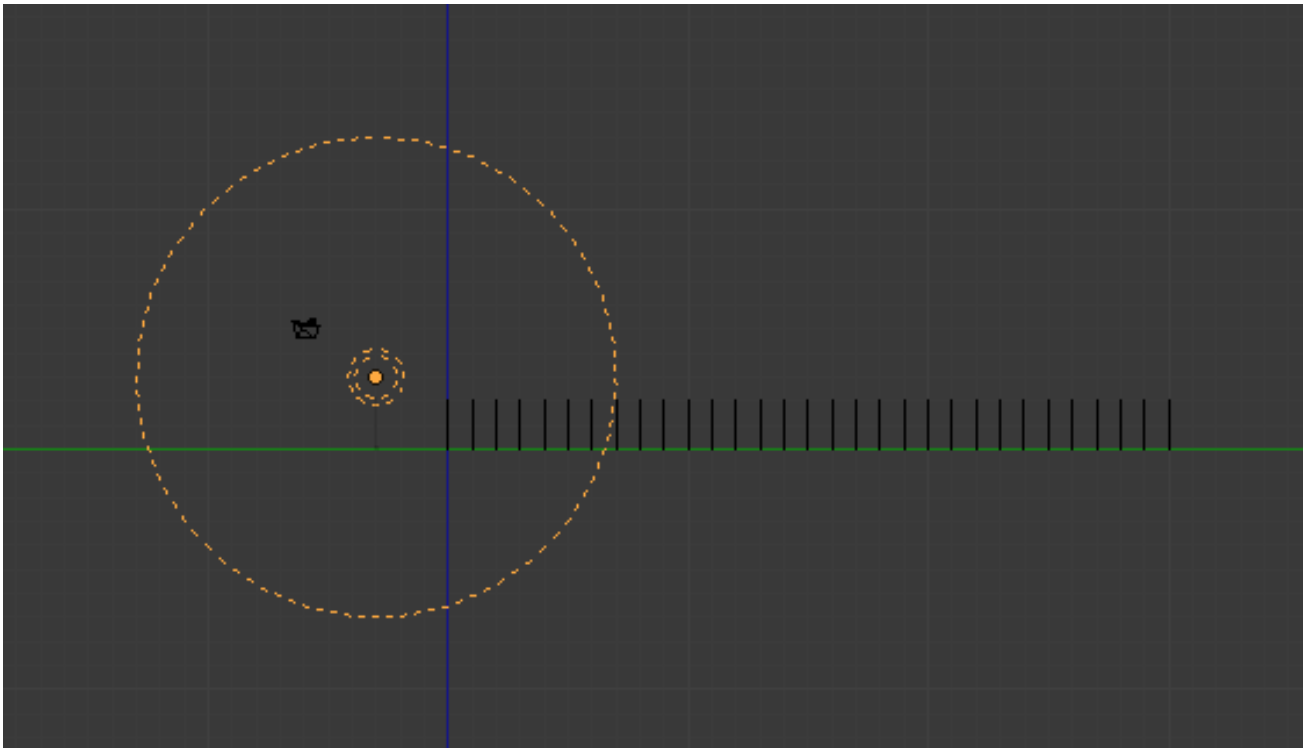


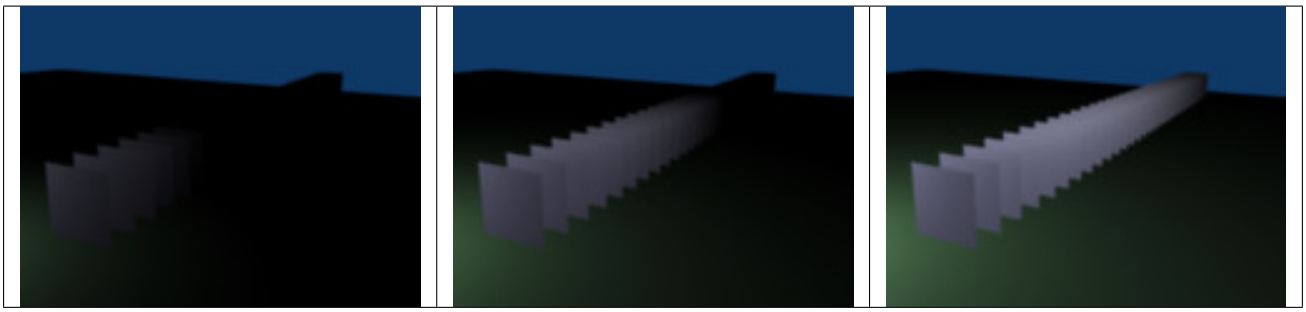
Fig. 2.1741: Clipping Sphere.

Sphere indicates that the light's intensity is null at the *Distance* distance and beyond, regardless of the chosen light's falloff. In Fig. *Clipping Sphere.* you can see a side view example of the setup with *Sphere* enabled and a distance of 10.

Any objects beyond the sphere receive no light from the lamp.

The *Distance* field is now specifying both where the light's rays become null, and the intensity's ratio falloff setting. Note that there is no abrupt transition at the sphere: the light attenuation is progressive (for more details, see the descriptions of the *Sphere* and *Falloff types* above).

Table 2.84: Sphere with 40.



In Fig. *Sphere with 10.*, the clipping sphere's radius is 10 units, which means the light's intensity is also being controlled by 10 units of distance. With a linear attenuation, the light's intensity has fallen very low even before it gets to the first object.

In Fig. *Sphere with 20.*, the clipping sphere's radius is now 20 BU and some light is reaching the middle objects.

In Fig. *Sphere with 40.*, the clipping sphere's radius is now 40 units, which is beyond the last object. However, the light does not make it to the last few objects because the intensity has fallen to nearly 0.

Hint: If a *Lamp* light is set to not cast shadows, it illuminates through walls and the like. If you want to achieve some nice effects like a fire, or a candle-lit room interior seen from outside a window, the *Sphere* option is a must. By carefully working on the *Distance* value you can make your warm firelight shed only within the room, while illuminating outside with a cool moonlight, the latter achieved with a *Sun* or *Hemi* light or both.

Lamps Textures

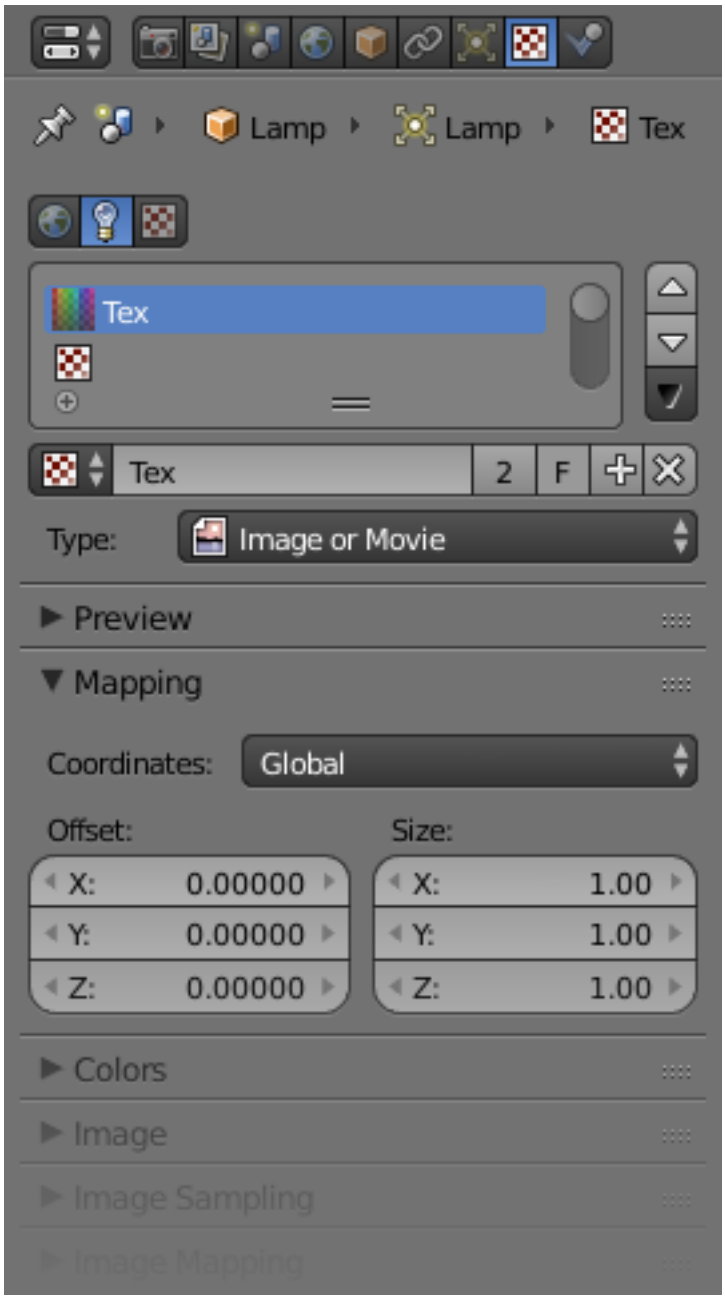


Fig. 2.1745: Lamp Texture panels.

When a new lamp is added, it produces light in a uniform, flat color.

While this might be sufficient in simple renderings, more sophisticated effects can be accomplished through the use of *textures*. Subtle textures can add visual nuance to a lamp, while hard textures can be used to simulate more pronounced effects, such as a disco ball, dappled sunlight breaking through treetops, or even a projector. These textures are assigned to one of ten channels, and behave exactly like material textures, except that they affect a lamp's color and intensity, rather than a material's surface characteristics.

Options

The lamp textures settings are grouped into two panels. Here we will only talk about the few things that differ from object material textures; see the *Materials* and *Textures* chapters for details about the standard options.

The texture-specific and the *Mapping* panels remain the same. However, you will note there are much fewer *Mapping* options. You can only choose between *Global*, *View* or another *Object* 's texture *coordinates* (since a lamp has no texture coordinates by itself), and you can scale or offset the texture.

The *Mapping* panel is also a subset of its regular material's counterpart. You can only map a lamp texture to its regular, basic *Color* and/or to its *Shadow* color. As you can only affect colors, and a lamp has no texture coordinates on its own, the *Diffuse*, *Specular*, *Shading*, and *Geometry* options have disappeared.

Lamps Related Settings

Here are some options closely related to light sources, without being lamps settings.

Lighting Groups

Materials

By default, materials are lit by all lamps in all visible layers, but a material (and thus all objects using that material) can be limited to a single group of lamps. This sort of control can be incredibly useful, especially in scenes with complex lighting setups. To enable this, navigate to the *Material* menu's *Options* panel and select a group of lamps in the *Light Group* field. Note that a *light group* must be created first.

If the *Exclusive* button is enabled, lights in the specified group will *only* affect objects with this material.

Render Layers

There is a similar control located in the *Layer panel* of the *Render Layers* tab. If a light group name is selected in this *Light* field, the scene will be lit exclusively by lamps in the specified group.

See also:

- *Lamps Introduction*
- *Shadows*

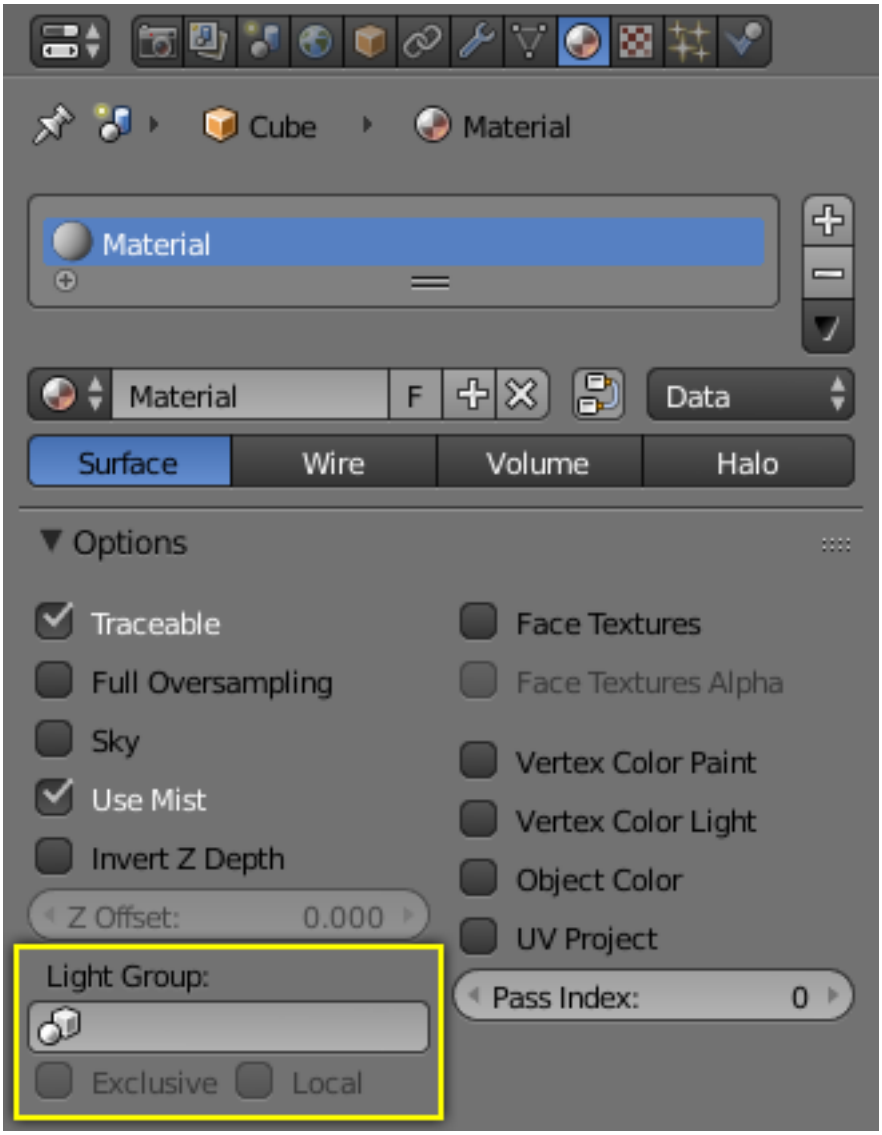


Fig. 2.1746: Light Group options for Materials.

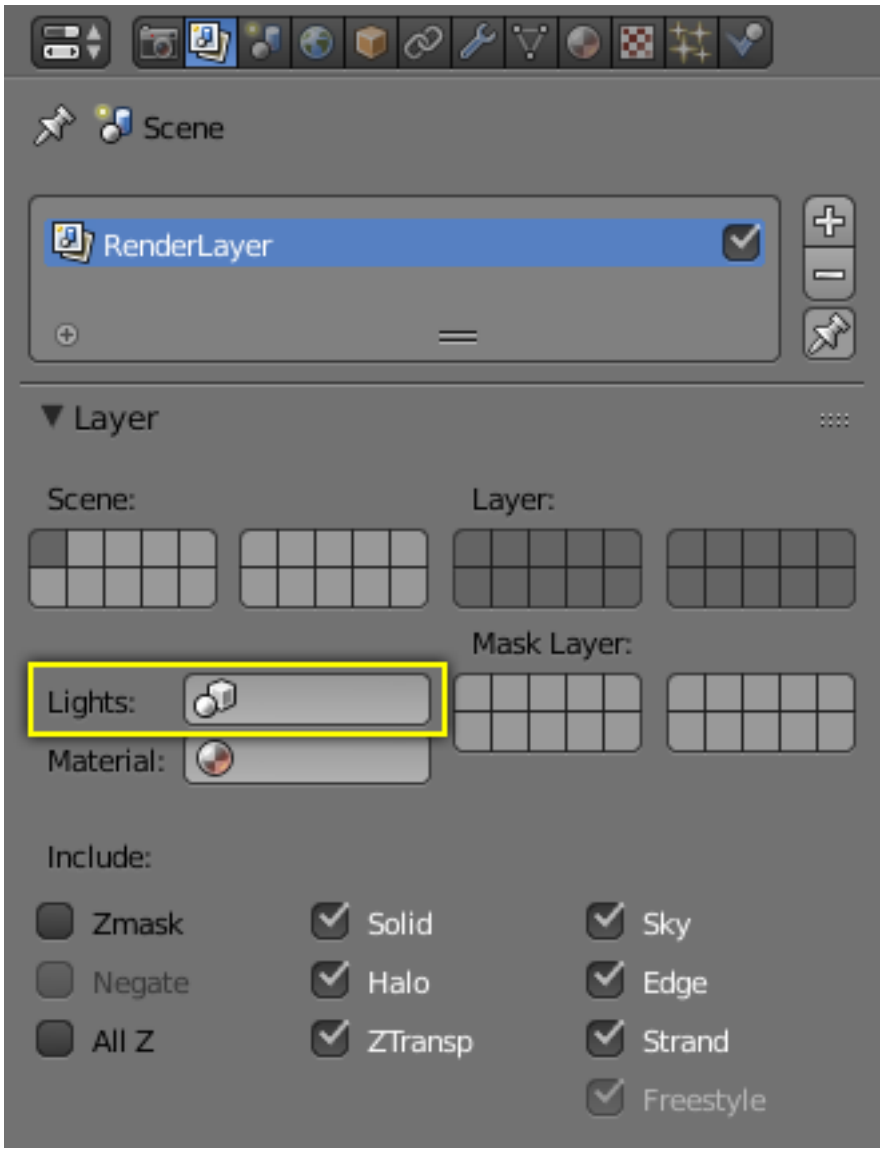


Fig. 2.1747: Light Group options for Render Layers.

- *Materials Introduction*

Shadows

Introduction

Light would not even exist without its counterpart: shadows. Shadows are a darkening of a portion of an object because light is being partially or totally blocked from illuminating the object. They add contrast and volume to a scene; there is nearly no place in the real world without shadows, so to get realistic renders, you will need them. Blender supports the following kinds of shadows:

- *Lamps: Ray-traced Shadows*
- *Lamps: Buffered Shadows*
- *Ambient occlusion*
- *Indirect lighting*

Ambient occlusion really is not a shadow based on light *per se*, but based on geometry. However, it does mimic an effect where light is prevented from fully and uniformly illuminating an object, so it is mentioned here. Also, it is important to mention ambient lighting, since increasing *Ambient* decreases the effect of a shadow.

You can use a combination of ray-traced and buffer shadows to achieve different results. Even within ray-traced shadows, different lamps cast different patterns and intensities of shadow. Depending on how you arrange your lamps, one lamp may wipe out or override the shadow cast by another lamp.

Shadows is one of those trifectas in Blender, where multiple things have to be set up in different areas to get results:

- The lamp has to cast shadows (ability and direction).
- An opaque object has to block light on its way (position and layer).
- Another object's material has to receive shadows (*Shadow* and *Receive Transparent* enabled).
- The render engine has to calculate shadows (*Shadow* for buffered shadows, *Shadow* and *Ray* for ray-traced shadows).

  For example, the simple *Lamp*, *Area*, and *Sun* light has the ability to cast ray shadows, but not buffer shadows. The *Spot* light can cast both, whereas the *Hemi* light does not cast any. If a *Sun* lamp is pointing sideways, it will not cast a shadow from a sphere above a plane onto the plane, since the light is not traveling that way. All lamps able to cast shadows share some common options, described in the *Shadow Panel*.

  Just to give you more shadow options (and further confuse the issue), lamps and materials can be set to respectively **only** cast and receive shadows, and not light the diffuse/specular aspects of the object. Also, render layers can turn on/off the shadow pass, and their output may or may not contain shadow information...

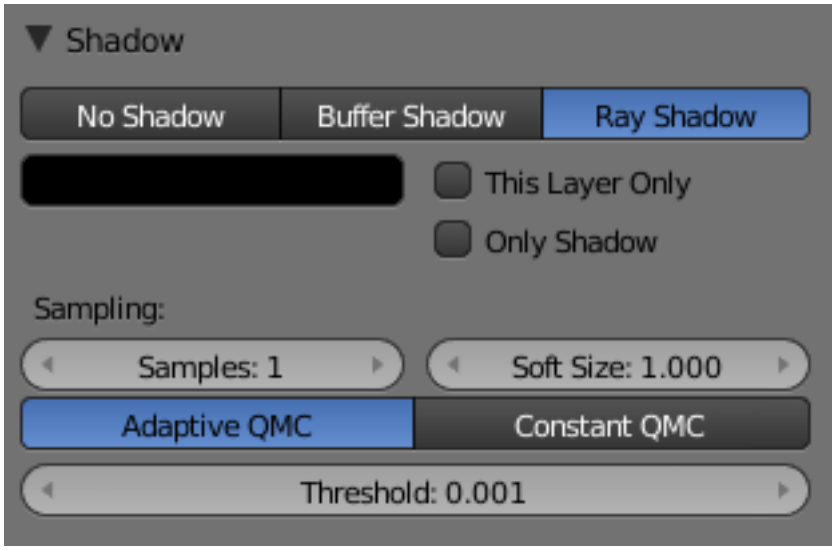


Fig. 2.1748: Ray Shadow enabled for a lamp.

Lamps: Ray-traced Shadows

Ray-traced shadows produce very precise shadows with very low memory use, but at the cost of processing time. This type of shadowing is available to all lamp types except *Hemi*.

As opposed to buffered shadows (*Lamps: Buffered Shadows*), ray-traced shadows are obtained by casting rays from a regular light source, uniformly and in all directions. The ray-tracer then records which pixel of the final image is hit by a ray light, and which is not. Those that are not are obviously obscured by a shadow.

Each light casts rays in a different way. For example, a *Spot* light casts rays uniformly in all directions within a cone. The *Sun* light casts rays from an infinitely distant point, with all rays parallel to the direction of the *Sun* light.

For each additional light added to the scene, with ray-tracing enabled, the rendering time increases. Ray-traced shadows require more computation than buffered shadows but produce sharp shadow borders with very little memory resource usage.

To enable ray-traced shadows, three actions are required:

- Enable *Shadows* globally in the *Render* menu's *Shading* panel.
- Enable *Ray tracing* globally from the same panel.
- Enable ray-traced shadows for the light using the *Ray Shadow* button in the *Light* menu's *Shadow* panel. This panel varies depending on the type of light.

All lamps able to cast ray-traced shadows share some common options, described in *Ray-traced Properties*.

Ray-traced shadows can be cast by the following types of lamp:

- *Point lamp*
- *Spot lamp*
- *Area lamp*
- *Sun lamp*

Lamps: Buffered Shadows

Fig. 2.1749: Buffer Shadow enabled for a Spot lamp.

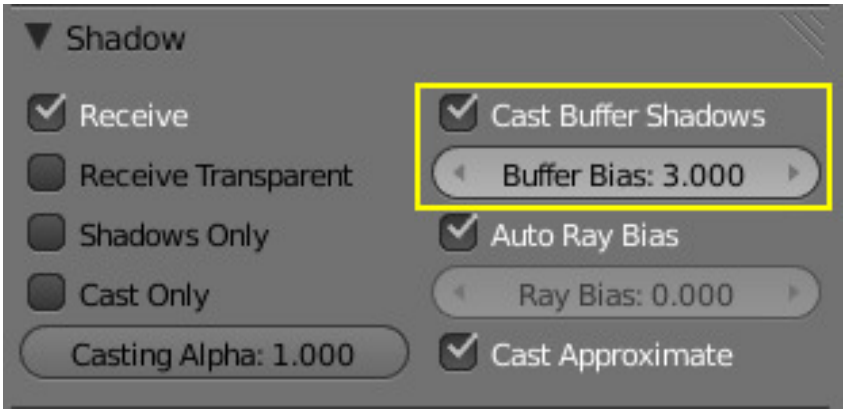


Fig. 2.1750: Cast Buffer Shadows enabled for a material.

Buffered shadows provide fast-rendered shadows at the expense of precision and/or quality. Buffered shadows also require more memory resources as compared to ray tracing. Using buffered shadows depends on your requirements. If you are rendering animations or cannot wait hours to render a complex scene with soft shadows, buffer shadows are a good choice.

For a scanline renderer – and Blender's built-in engine *is*, among other things, a scanline renderer – shadows can be computed using a *shadow buffer*. This implies that an "image", as seen from the spot lamp's point of view, is "rendered" and that the distance – in the image – for each point from the spot light is saved. Any point in the "rendered" image that is farther away than any of those points in the spot light's image is then

considered to be in shadow. The shadow buffer stores this image data.

To enable buffered shadows these actions are required:

- Enable shadows globally from the *Scene* menu's *Gather* panel by selecting *Approximate*.
- Enable shadows for the light using the *Buffer Shadow* button in the *Lamp* menu's *Shadow* panel.
- Make sure the *Cast Buffer Shadows* options is enabled in each *Material* 's *Shadow* panel.
- The *Spot lamp* is the only lamp able to cast buffered shadows.

Shadow Panel

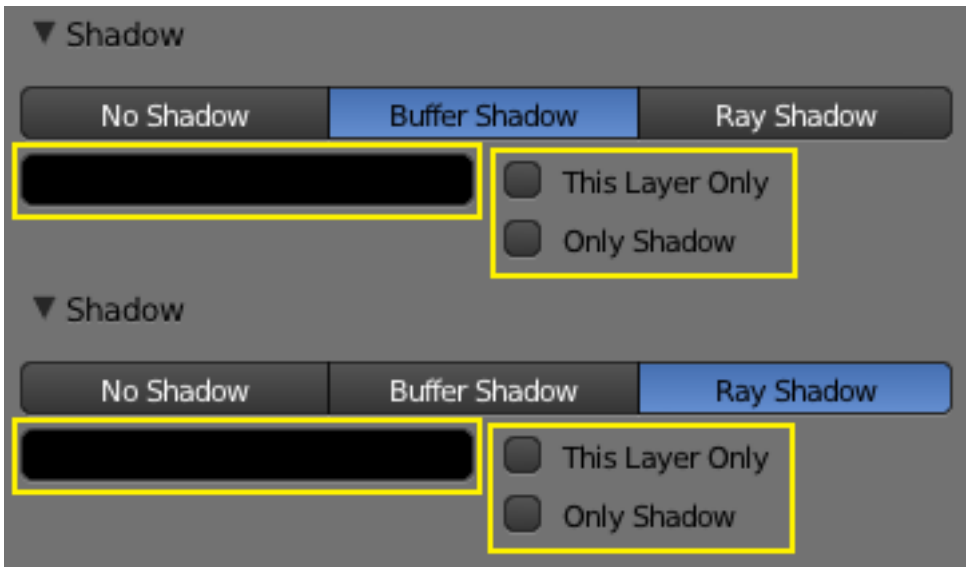


Fig. 2.1751: Common shadowing options for lamps.

All lamps able to cast shadows. Share some options, described below:

Shadow Method

No Shadow The lamp casts no shadow.

Buffered Shadow The *Spot lamp* is the only lamp able to cast buffered shadows.

Raytraced Shadows *Ray-traced Properties*.

This Layer Only When this option is enabled, only the objects on the same layer as the light source will cast shadows.

Only Shadow The light source will not illuminate an object but will generate the shadows that would normally appear. This feature is often used to control how and where shadows fall by having a light which illuminates but has no shadow, combined with a second light which does not illuminate but has *Only Shadow* enabled, allowing the user to control shadow placement by moving the "Shadow Only" light around.

Shadow color This color picker control allows you to choose the color of your cast shadows (black by default). The images below were all rendered with a white light and the shadow color was selected independently.

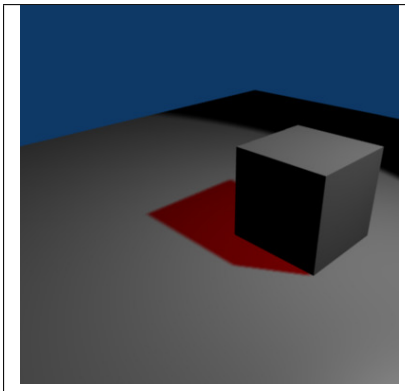


Fig. 2.1752: Red colored shadow example.

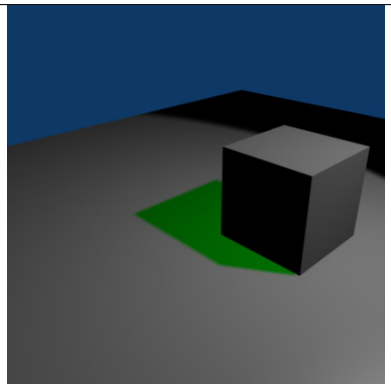


Fig. 2.1753: Green colored shadow example.

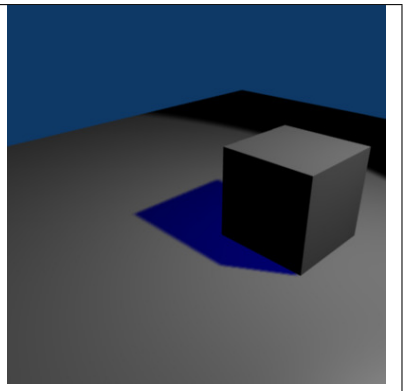


Fig. 2.1754: Blue colored shadow example.

Although you can select a pure white color for a shadow color, it appears to make a shadow disappear.

Raytraced Shadows

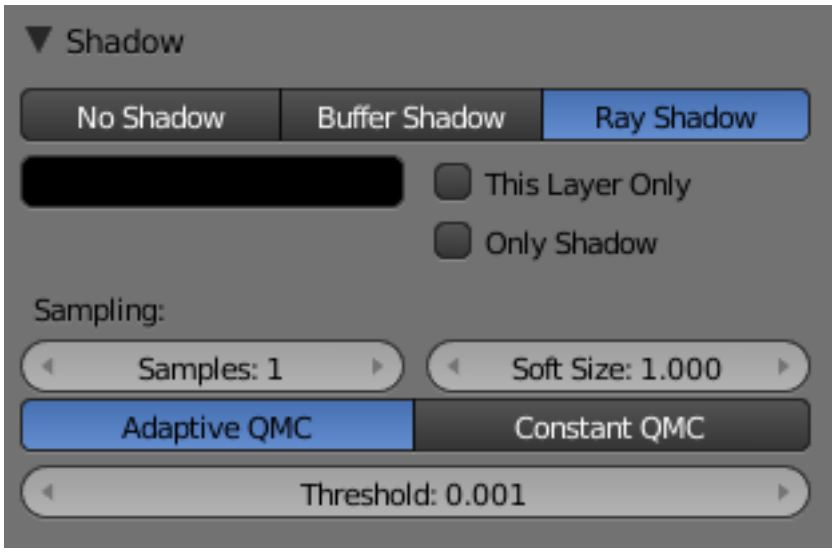


Fig. 2.1755: Ray shadowing options for lamps.

Most lamp types (*Lamp*, *Spot* and *Sun*) share the same options for the ray-traced shadows generation, which are described below. Note that the *Area* lamp, even though using most of these options, have some specifics described in its *own ray-traced shadows page*.

Ray Shadow The *Ray Shadow* button enables the light source to generate ray-traced shadows. When the *Ray Shadow* button is selected, another set of options is made available, those options being:

Shadow sample generator type Method for generating shadow samples: Adaptive QMC is fastest, Constant QMC is less noisy but slower. This allows you to choose which algorithm is to be used to generate the samples that will serve to compute the ray-traced shadows (for now, mainly two variants of Quasi-Monte Carlo, see *Quasi-Monte Carlo method*):

Constant QMC The *Constant QMC* method is used to calculate shadow values in a very uniform, evenly distributed way. This method results in very

good calculation of shadow value but it is not as fast as using the *Adaptive QMC* method; however, *Constant QMC* is more accurate.

Adaptive QMC The *Adaptive QMC* method is used to calculate shadow values in a slightly less uniform and distributed way. This method results in good calculation of shadow value but not as good as *Constant QMC*. The advantage of using *Adaptive QMC* is that it is in general much quicker while being not much worse than *Constant QMC* in terms of overall results.

Samples Number of extra samples taken (samples x samples). This slider sets the maximum number of samples that both *Constant QMC* and *Adaptive QMC* will use to do their shadow calculations. The maximum value is 16: the real number of samples is actually the square of it, so setting a sample value of 3 really means $3^2 = 9$ samples will be taken.

Soft Size Light size for ray shadow sampling. This slider determines the size of the fuzzy/diffuse/penumbra area around the edge of a shadow. *Soft Size* only determines the width of the soft shadow size, not how graded and smooth the shadow is. If you want a wide shadow which is also soft and finely graded, you must also set the number of samples in the *Samples* field higher than 1; otherwise this field has no visible effect and the shadows generated will not have a soft edge. The maximum value for *Soft Size* is 100.0.

Below is a table of renders with different *Soft Size* and *Samples* settings showing the effect of various values on the softness of shadow edges:

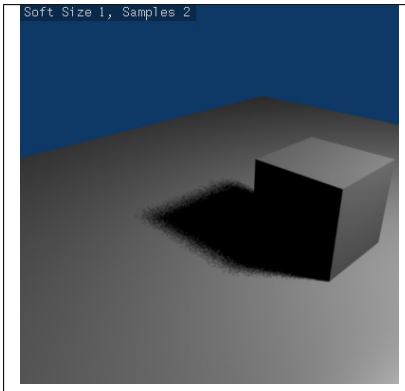


Fig. 2.1756: Soft Size: 1.0, Samples: 2.

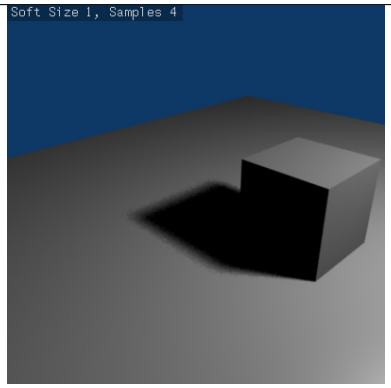


Fig. 2.1757: Soft Size: 1.0, Samples: 4.

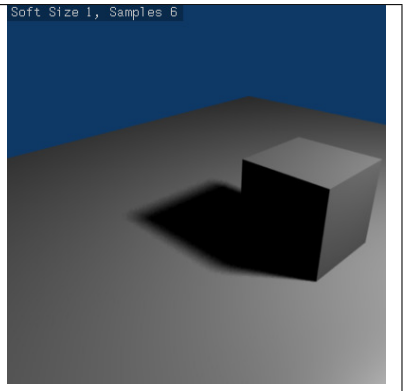


Fig. 2.1758: Soft Size: 1.0, Samples: 6.

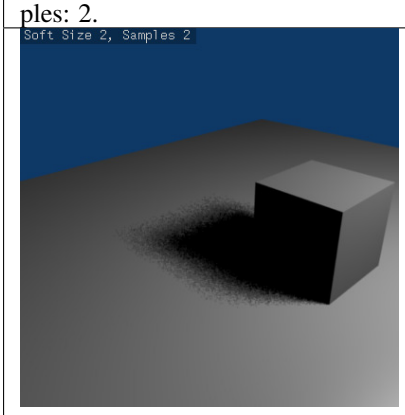


Fig. 2.1759: Soft Size: 2.0, Samples: 2.

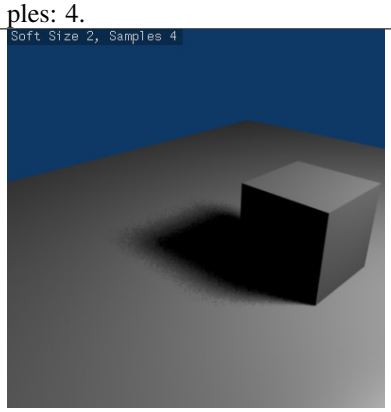


Fig. 2.1760: Soft Size: 2.0, Samples: 4.

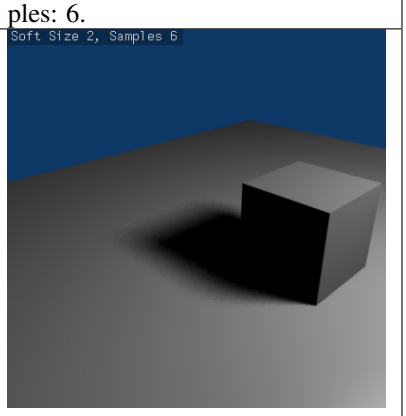


Fig. 2.1761: Soft Size: 2.0, Samples: 6.

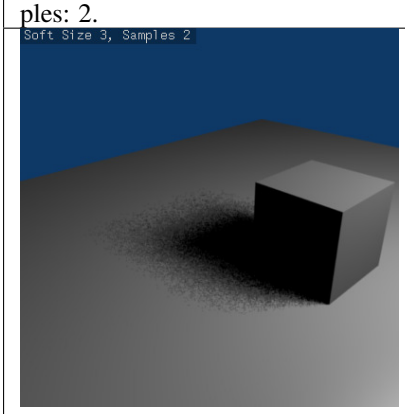


Fig. 2.1762: Soft Size: 3.0, Samples: 2.

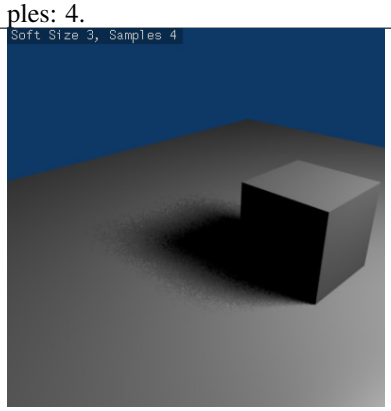


Fig. 2.1763: Soft Size: 3.0, Samples: 4.

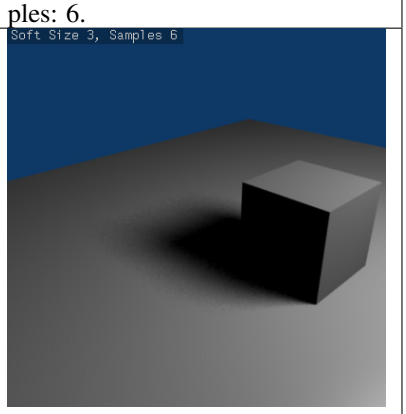


Fig. 2.1764: Soft Size: 3.0, Samples: 6.

Below is an animated version of the above table of images showing the effects:

Fig. 2.1765: Animated version renders with different Soft Size and Samples settings showing the effect of various values on the softness of shadow edges.

Threshold Threshold for Adaptive Sampling. This field is used with the *Adaptive QMC* shadow calculation method. The value is used to determine if the *Adaptive QMC* shadow sample calculation can be skipped based on a threshold of how shadowed an area is already. The maximum *Threshold* value is 1.0.

Quasi-Monte Carlo method

The Monte Carlo method is a method of taking a series of samples/readings of values (any kind of values, such as light values, color values, reflective states) in or around an area at random, so as to determine the correct actions to take in certain calculations which usually require multiple sample values to determine overall accuracy of those calculations. The Monte Carlo method tries to be as random as possible; this can often cause areas that are being sampled to have large irregular gaps in them (places that are not sampled/read). This in turn can cause problems for certain calculations (such as shadow calculation).

The solution to this was the Quasi-Monte Carlo method.

The Quasi-Monte Carlo method is also random, but tries to make sure that the samples/readings it takes are also better distributed (leaving less irregular gaps in its sample areas) and more evenly spread across an area. This has the advantage of sometimes leading to more accurate calculations based on samples/reading.

See also:

- *Lamp Light Raytraced Shadows*
- *Spot Light Raytraced Shadows*
- *Area Light Raytraced Shadows*
- *Sun Light Raytraced Shadows*

Volumetric Lighting

"Volumetric lighting is a technique used in 3D computer graphics to add lighting effects to a rendered scene. It allows the viewer to see beams of light shining through the environment; seeing sunbeams streaming through an open window is an example of volumetric lighting, also known as God rays. The term seems to have been introduced from cinematography and is now widely applied to 3D modeling and rendering especially in the field of 3D gaming. In volumetric lighting, the light cone emitted by a light source is modeled as a transparent object and considered as a container of a "volume": as a result, light has the capability to give the effect of passing through an actual three dimensional medium (such as fog, dust, smoke, or steam) that is inside its volume, just like in the real world."

—According to Wikipedia, volumetric lighting.

A classic example is the search light with a visible halo/shaft of light being emitted from it as the search light sweeps around.

By default Blender does not model this aspect of light. For example when Blender lights something with a *Spot* light, you see the objects and area on the floor lit but not the shaft/halo of light coming from the spotlight as it progresses to its target and would get scattered on the way.

The halo/shaft of light is caused in the real world by light being scattered by particles in the air, some of which get diverted into your eye and that you perceive as a halo/shaft of light. The scattering of light from a source can be simulated in Blender using various options, but by default is not activated.

The only lamp able to create volumetric effects is the *Spot lamp* (even thought you might consider some of the *"Sky & Atmosphere" effects* of the *Sun* lamp as volumetric as well).

See also:

- *Mist*
- *Smoke*
- *Volumetric Materials*

Lamp Types

Introduction

Blender comes equipped with five different lamp types, each with its own unique strengths and limitations. Here are the available lamps:

- *Point* is an omni-directional point light source, similar to a light bulb.
- *Spot* is a directional point light source, similar to ... a spot.
- *Area* is a source simulating an area which is producing light, as windows, neons, TV screens.
- *Hemi* simulates a very wide and far away light source, like the sky.
- *Sun* simulates a very far away and punctual light source, like the sun.

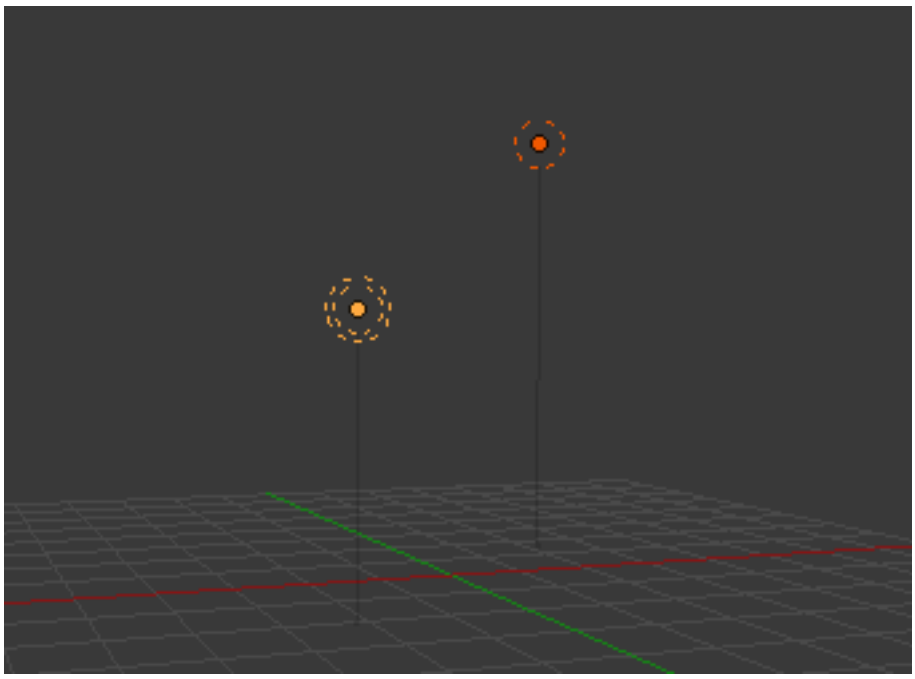


Fig. 2.1766: Visual height and shadow markers of two points lamps. Ray Shadow is enabled on the left lamp.

You can add new lamps to a scene using the *Add* menu in the top header, or with → *Add* → *Lamp*, `Shift-A`.

Once added, a lamp's position is indicated in the 3D View by a solid dot in a circle, but most types also feature dashed wire-frames that help describe their orientation and properties. While each type is represented differently, there are some visual indicators common to all of them:

Shadows If shadows are enabled, an additional dashed circle is drawn around the solid circle. This makes it easier to quickly determine if a lamp has shadows enabled.

Vertical Height Marker This is a dim gray line, which helps locate the lamp's position relative to the global XY plane.

Point

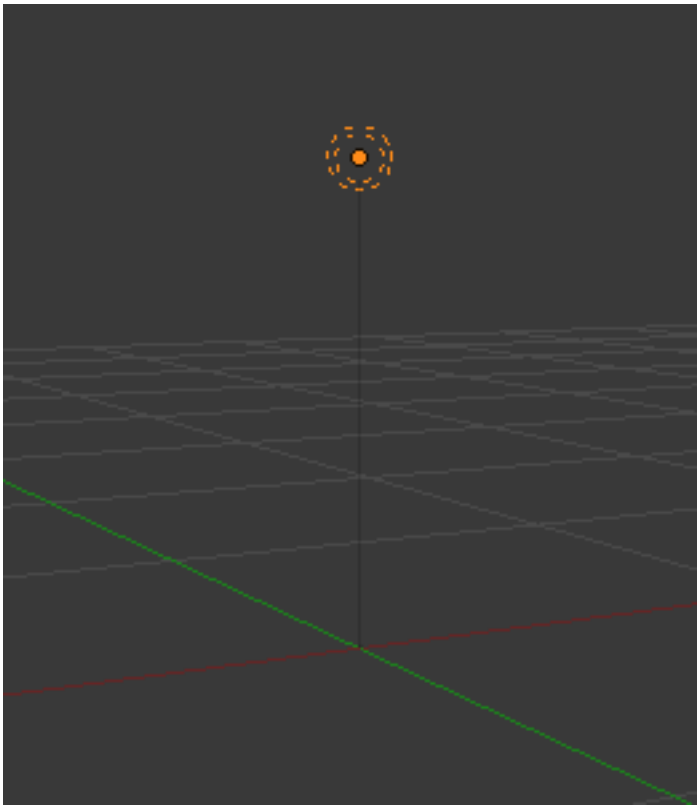


Fig. 2.1767: Point lamp.

The *Point* lamp is an omni-directional point of light, that is, a point radiating the same amount of light in all directions. It's visualized by a plain, circled dot. Being a point light source, the direction of the light hitting an object's surface is determined by the line joining the lamp and the point on the surface of the object itself.

Light intensity/energy decays based on (among other variables) distance from the *Point* lamp to the object. In other words, surfaces that are further away are rendered darker.

Lamp Options

Distance, Energy and Color These settings are common to most types of lamps, and are described in *Light Properties*.

Negative, This Layer Only, Specular, and Diffuse These settings control what the lamp affects, as described in *What Light Affects*.

Falloff and Sphere These settings control how the light of the *Lamp* decays with distance. See *Light Attenuation* for details.

Shadows

The *Point* light source can only cast ray-traced shadows. It shares with other lamp types the common shadow options described in *Shadow Panel*.

The ray-traced shadows settings of this lamp are shared with other lamps, and are described *Raytraced Properties*.

Fig. 2.1768: Without ray shadows.

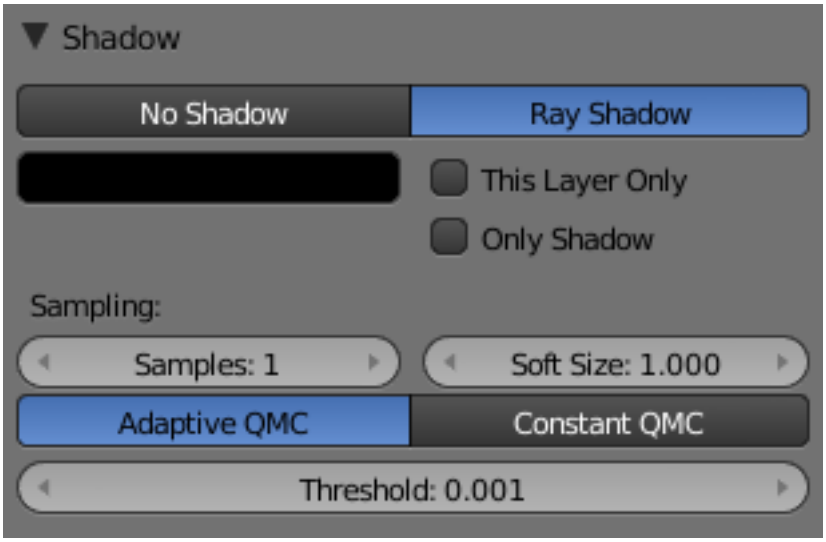


Fig. 2.1769: Point lamp with ray shadows and Adaptive QMC sample generator enabled.

Sun

Introduction

A *Sun* lamp provides light of constant intensity emitted in a single direction. A *Sun* lamp can be very handy for a uniform clear daylight open-space illumination. In the 3D View, the *Sun* light is represented by an encircled black dot with rays emitting from it, plus a dashed line indicating the direction of the light.

This direction can be changed by rotating the *Sun* lamp, like any other object, but because the light is emitted in a constant direction, the location of a *Sun* lamp does not affect the rendered result (unless you use the *"sky & atmosphere" option*).

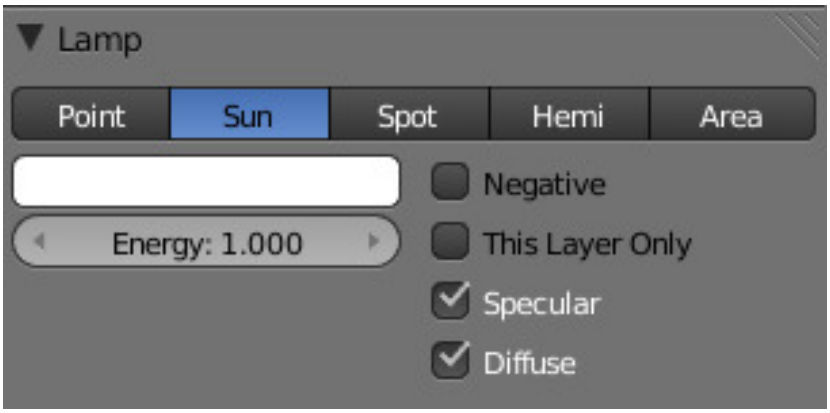


Fig. 2.1770: Sun lamp panel.

Lamp options

Energy and Color These settings are common to most types of lamps, and are described in *Light Properties*.

Negative, This Layer Only, Specular, and Diffuse These settings control what the lamp affects, as described in *What Light Affects*.

The *Sun* lamp has no light falloff settings: it always uses a constant attenuation (i.e. no attenuation!).

Sky & Atmosphere

Fig. 2.1771: Sky & Atmosphere panel.

Various settings for the appearance of the sun in the sky, and the atmosphere through which it shines, are available. For details, see *Sky and Atmosphere*.

Shadow

The *Sun* light source can only cast ray-traced shadows. It shares with other lamp types the same common shadowing options, described in *Shadow Panel*.

The ray-traced shadows settings of this lamp are shared with other lamps, and are described in *Raytraced Properties*.

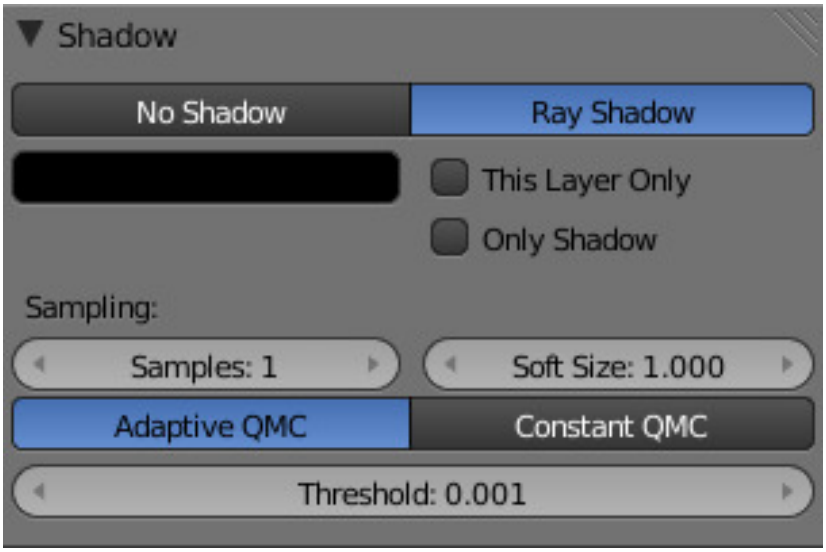


Fig. 2.1772: Shadow panel.

Sky & Atmosphere

This panel allows you to enable an effect that simulates various properties of real sky and atmosphere: the scattering of sunlight as it crosses the kilometers of air overhead. For example, when the Sun is high, the sky is blue (and the horizon, somewhat whitish). When the Sun is near the horizon, the sky is dark blue/purple, and the horizon turns orange. The dispersion of the atmosphere is also more visible when it is a bit foggy: the farther away an object is, the more "faded" in light gray it is... Go out into the countryside on a nice hot day, and you will see.

To enable this effect, you have to use a *Sun* light source. If, as usual, the *position* of the lamp has no importance, its *rotation* is crucial: it determines which hour it is. As a starting point, you should reset rotation of your *Sun* (with `Alt-R`, or typing `0` in each of the three *Rotation* fields X, Y, Z in the *Transform* panel). This way, you will have a nice mid-day sun (in the tropics).

Now, there are two important angles for the *Sky/Atmosphere* effect: the "incidence" angle (between the light direction and the X-Y plane), which determines the "hour" of the day (as you might expect, the default rotation – straight down – is "mid-day", a light pointing straight up is "midnight", and so on...). And the rotation around the Z axis determines the position of the sun around the camera.

In fact, to have a good idea of where the sun is in your world, relative to the camera in your 3D View, you should always try to have the dashed "light line" of the lamp crossing the center of the camera (its "focal" point), as shown in (The dashed "light line" of the *Sun lamp* crossing the camera focal point). This way, in camera view (`Numpad0`, center area in the example picture), you will see where the "virtual" sun created by this effect will be.

It is important to understand that the *position* of the sun has no importance for the effect: only its *orientation* is relevant. The position just might help you in your scene design.

Fig. 2.1773: Sky & Atmosphere panel.

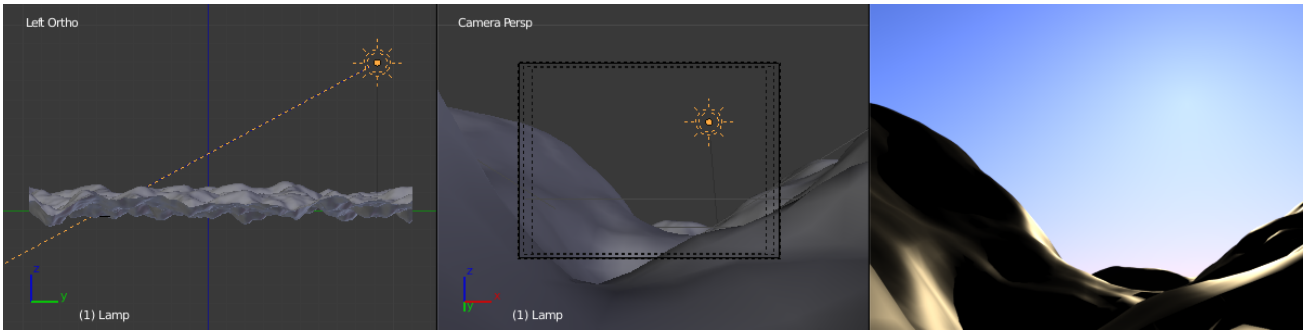


Fig. 2.1774: The dashed "light line" of the Sun lamp crossing the camera focal point.

Options

Sun & Sky Presets

- Classic
- Desert
- Mountain

Sky

Sky This button enables the sky settings: it will create a "sky", with a "sun" if visible, and mix it with the background as defined in *World* settings.

Turbidity This is a general parameter that affects sun view, sky and atmosphere; it is an atmosphere parameter where low values describe clear sky, and high values shows more foggy sky. In general, low values give a clear, deep blue sky, with "little" sun; high values give a more reddish sky, with a big halo around the sun. Note that this parameter is one which can really modify the "intensity" of the sun lighting. See examples below.

Here are its specific controls:

Blending The select menu shows various mix methods. The one selected will be used to blend the sky and sun with the background defined in the *World* settings. The mixing methods are the same as described e.g. in the *Mix Compositing Node* page.

Factor Controls how much the sky and sun effect is applied to the World background.

Color space These buttons allows you to select which color space the effect uses, with the following choices:

- CIE
- REC709
- SMPTE
- Exposure

This number button allows you to modify the exposure of the rendered Sky and Sun (0.0 for no correction).

Horizon

Brightness Controls brightness of colors at the horizon. Its value should be in the range (0.0 to 10.0); values near zero means no horizontal brightness, and large values for this parameter increase horizon brightness. See examples below.

Spread Controls spread of light at the horizon. Its value should be in the range (0.0 to 10.0); values low in the range result in less spread of light at horizon, and values high in the range result in horizon light spread in through all the sky.

Sun

Brightness Controls the sun brightness. Its value should be in the range (0.0 to 10.0); with low values the sky has no sun and with high values the sky only has sun.

Size Controls the size of sun. Its values should be in the range (0.0 to 10.0), but note that low values result in large sun size, and high values result in

small sun size. Note that the overall brightness of the sun remains constant (set by *Brightness*), so the larger the sun (the smaller *Size*), the more it "vanishes" in the sky, and *vice versa*.

Back Light For "Back Scatter Light", result on sun's color, high values result in more light around the sun. Its values range is (-1.0 to 1.0). Negative values result in less light around sun.

Atmosphere

Atmosphere This button enables the atmosphere settings. It will not modify the background, but it tries to simulate the effects of an atmosphere: scattering of the sunlight in the atmosphere, its attenuation, ...

Intensity

Sun Sets sun intensity. Its values are in range (0.0 to 10.0). High values result in bluer light on far objects.

Distance This factor is used to convert Blender units into an understandable unit for atmosphere effect, it starts from 0 and high values result in more yellow light in the scene.

Scattering

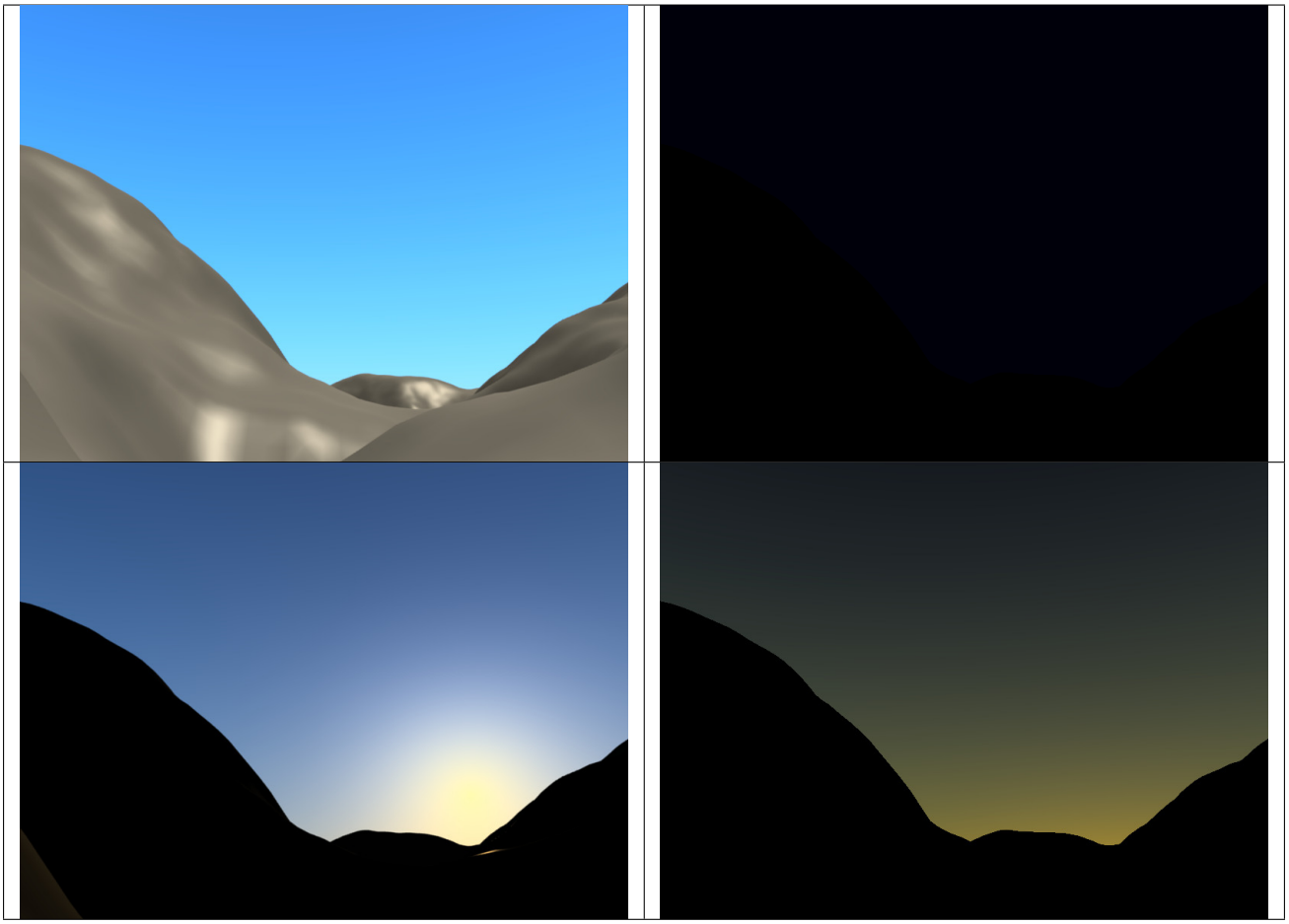
Inscattering This factor can be used to decrease the effect of light inscattered into atmosphere between the camera and objects in the scene. This value should be 1.0 but can be changed to create some nice, but not realistic, images.

Extinction This factor can be use to decrease the effect of extinction light from objects in the scene. Like *Inscattering* factor, this parameter should be 1.0 but you can change it; low values result in less light extinction. Its value is in the range (0.0 to 1.0).

Examples

First, let us see what happens when we modify the orientation of the sun:

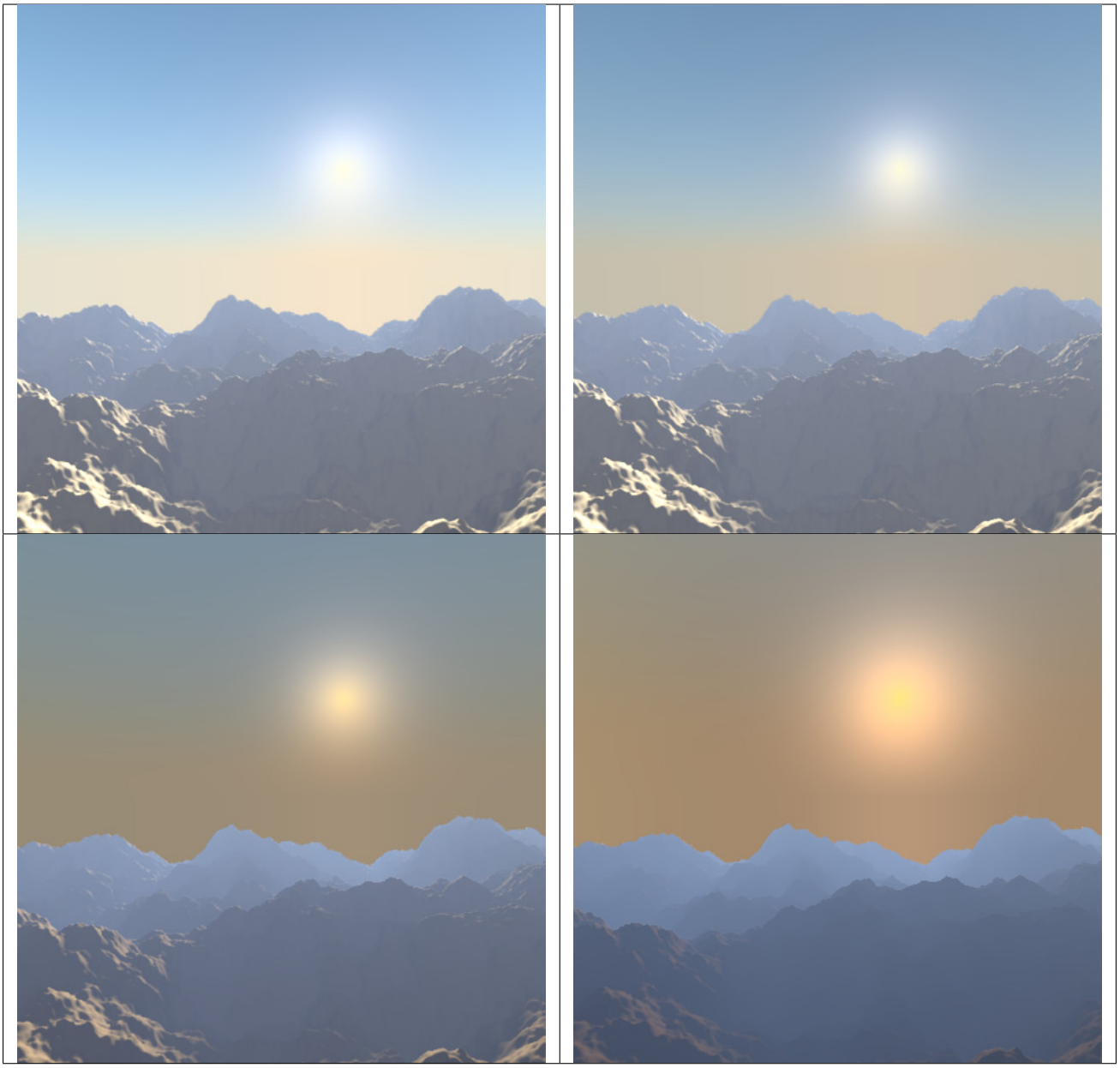
Table 2.85: Sun slightly below the horizon (end of twilight).



The 2.4 blend-file of these examples.

And now, the effects of various settings (examples created with this 2.4 blend-file):

Table 2.86: Turbidity: 10.0.



Sky

Table 2.87: Horizon Brightness: 1.13.

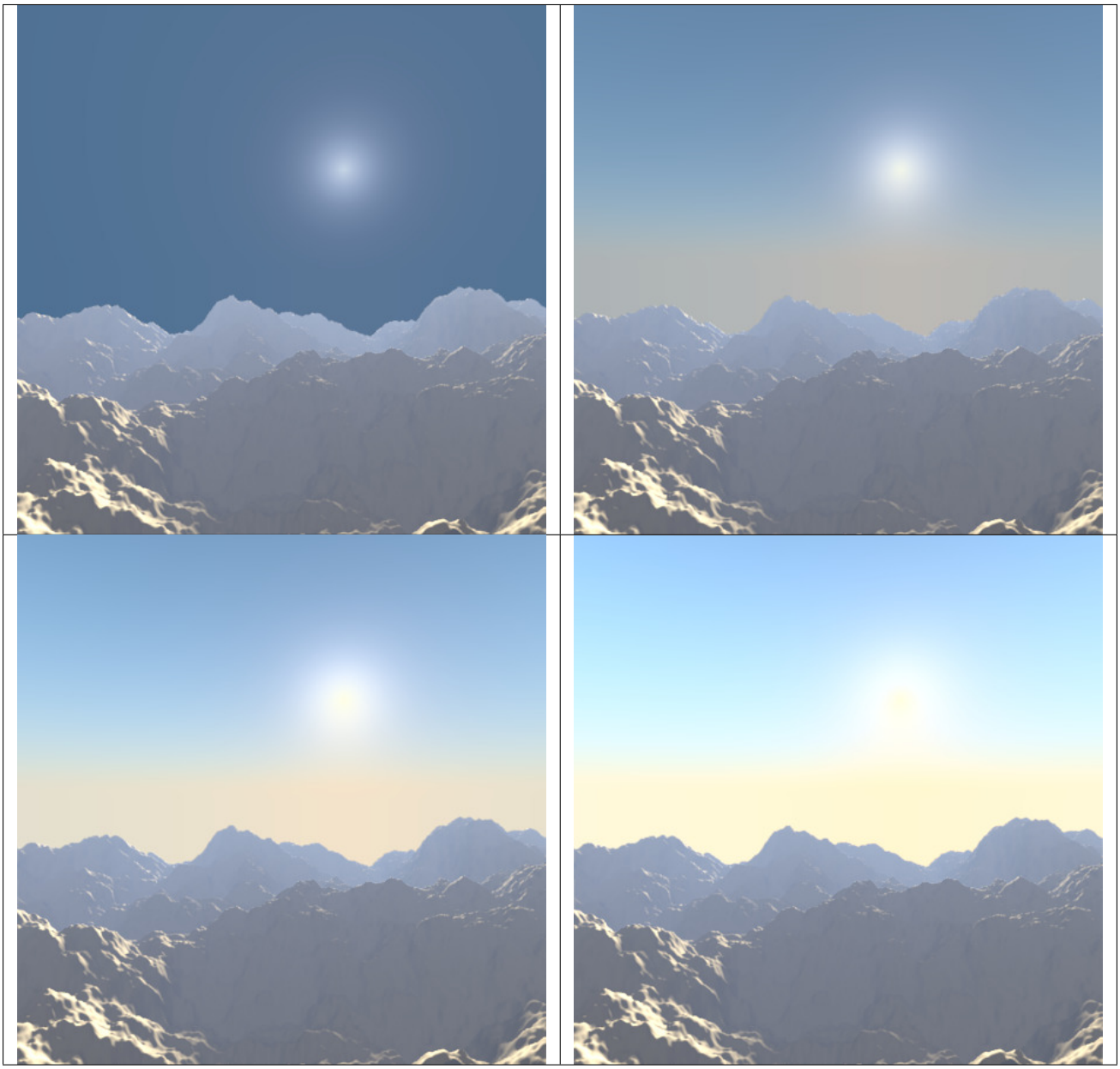


Table 2.88: Horizon Spread: 5.0.

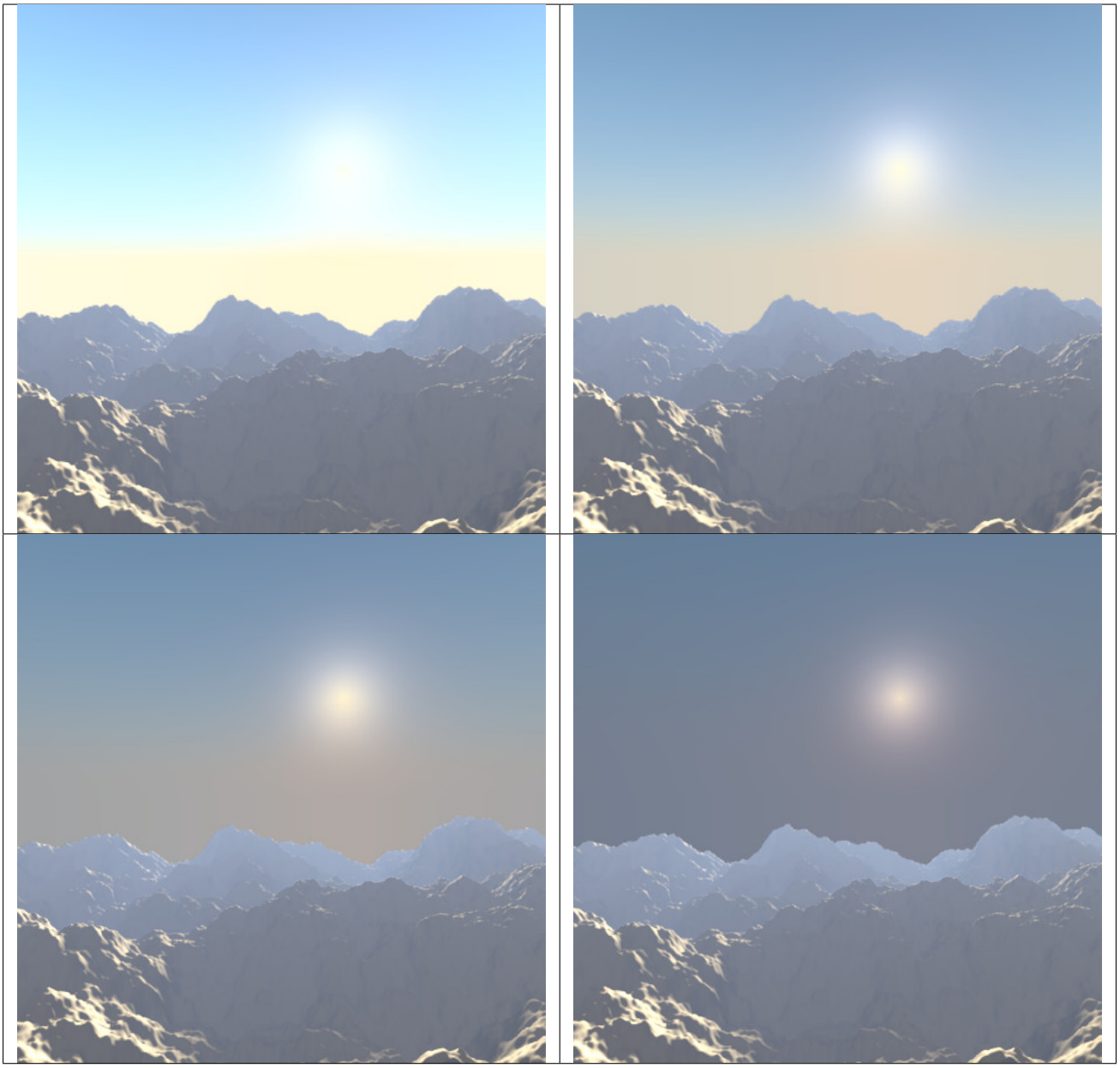


Table 2.89: Sun Brightness: 1.0.

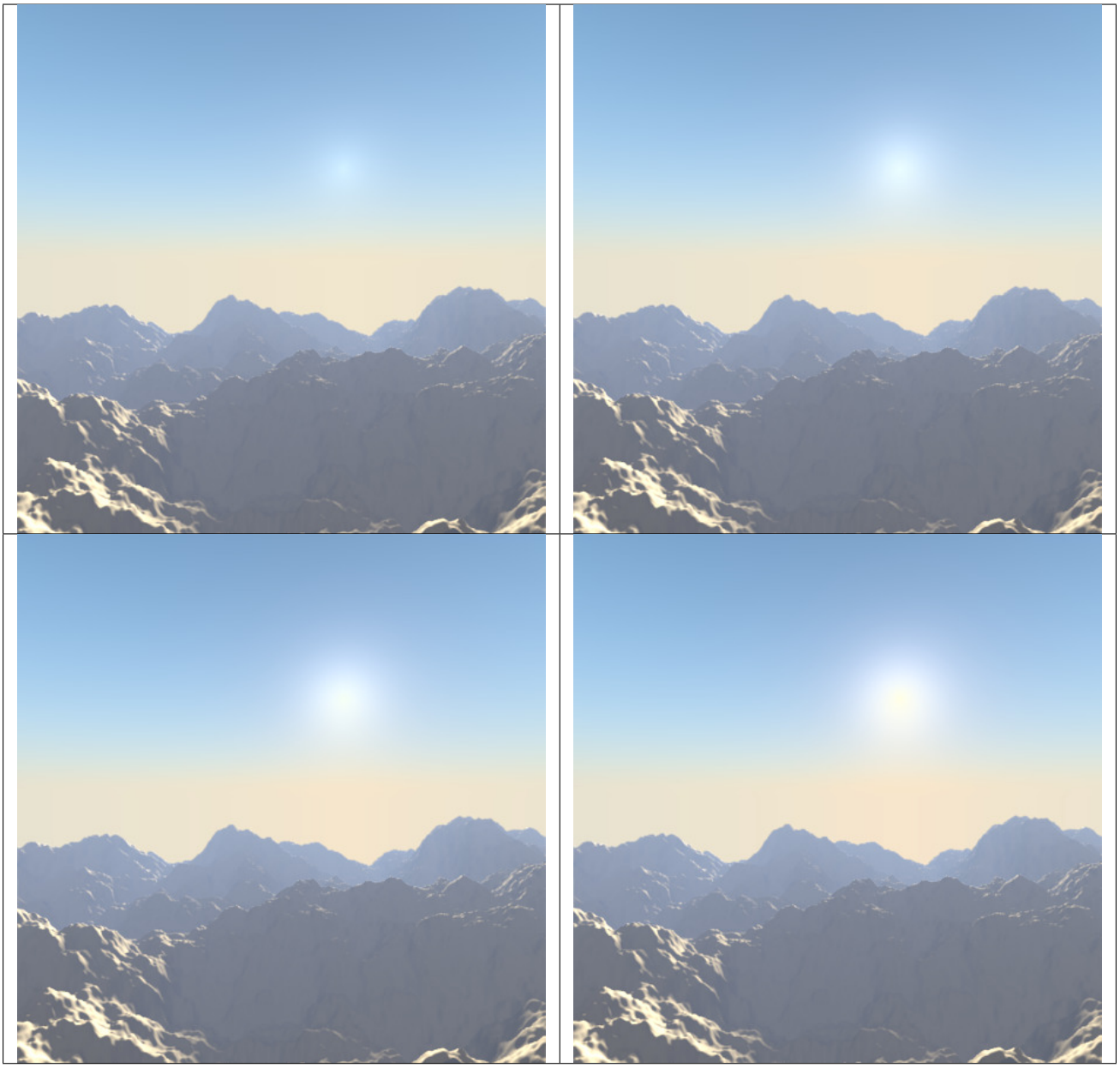


Table 2.90: Sun Size: 10.0.

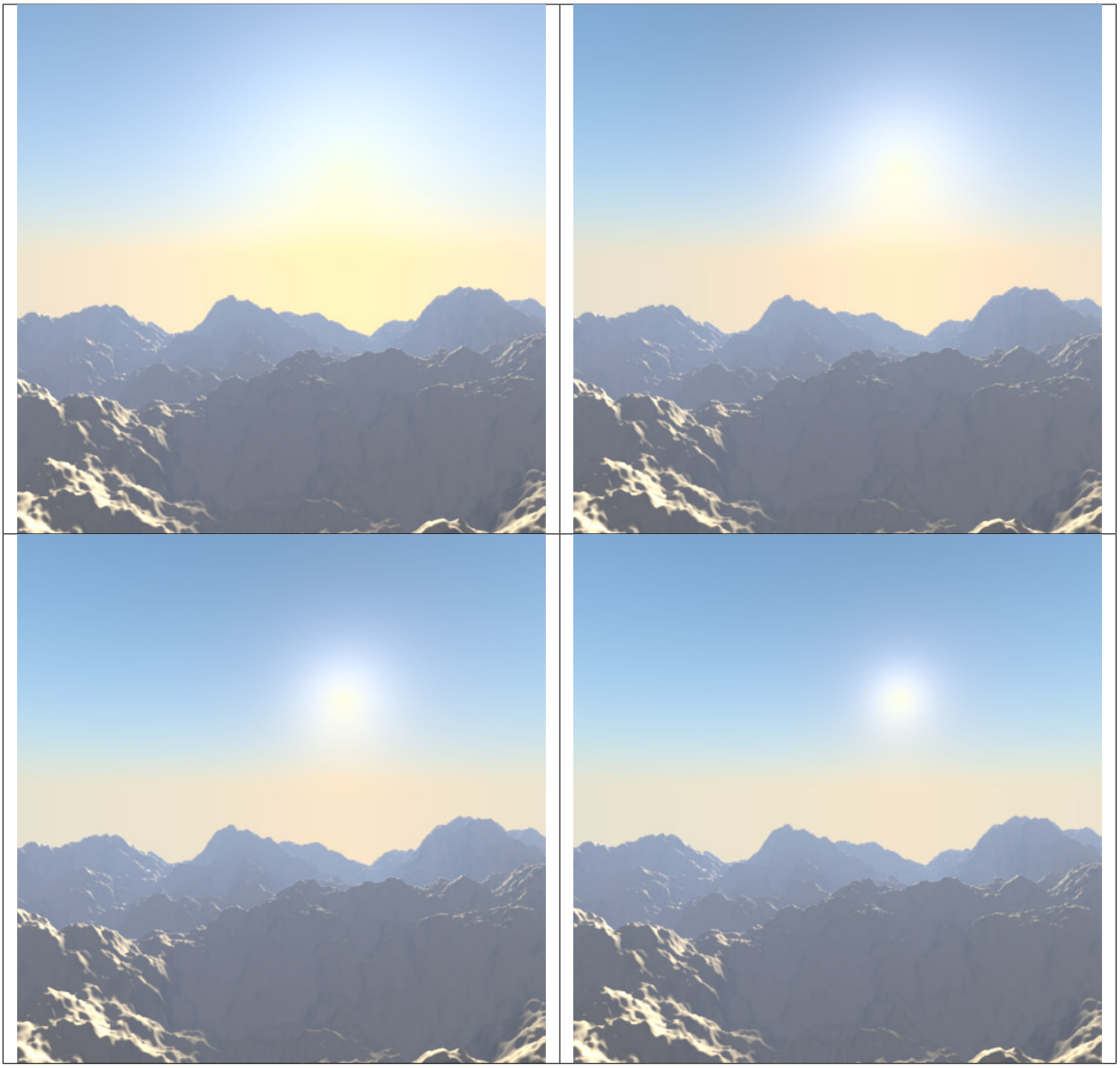
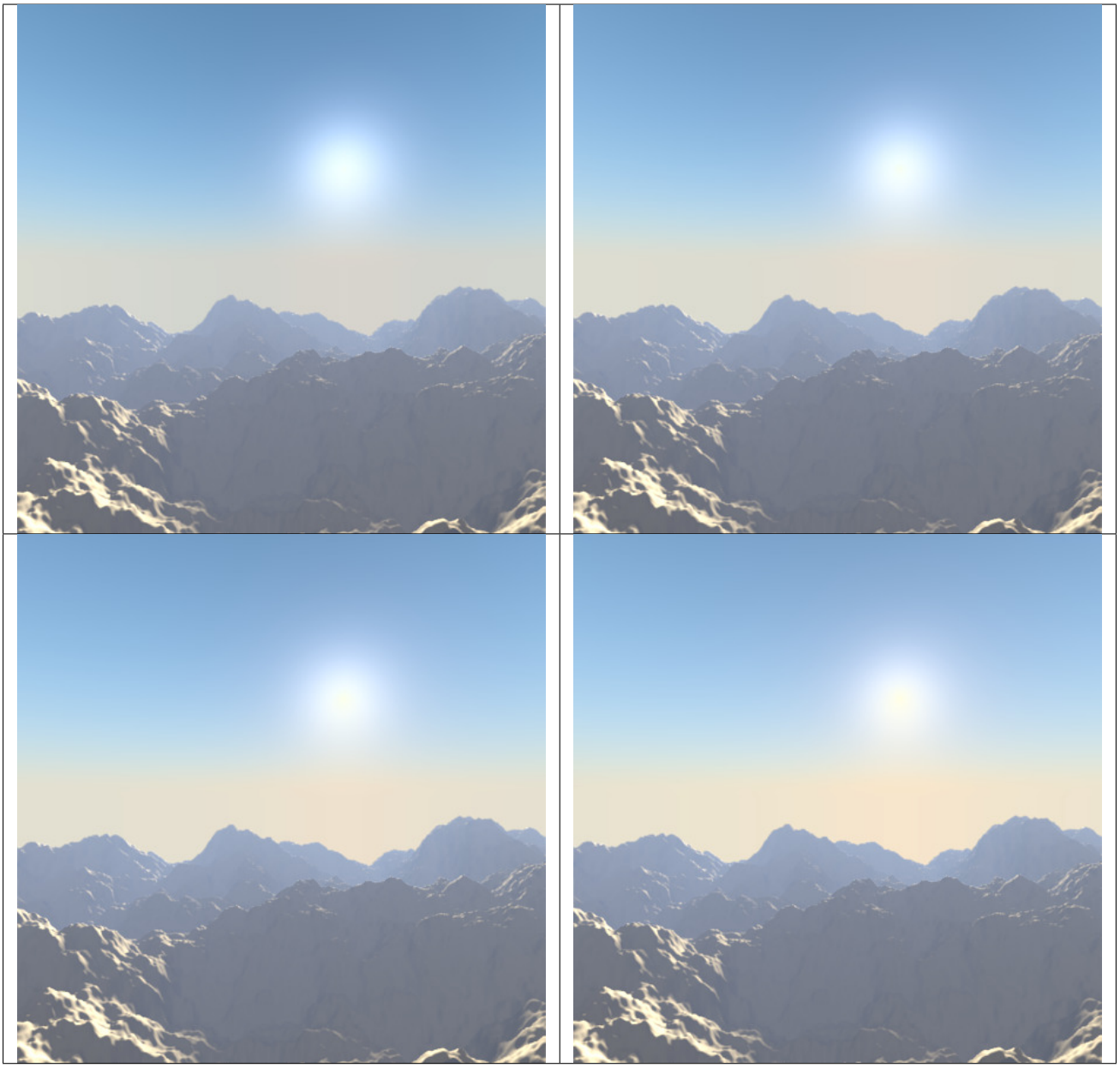


Table 2.91: Back Light: 1.0.



Atmosphere

For all renders below, *Hor.Bright* is set to 0.2, and *Sun Bright* to 2.0.

Table 2.92: Sun Intensity: 10.0.

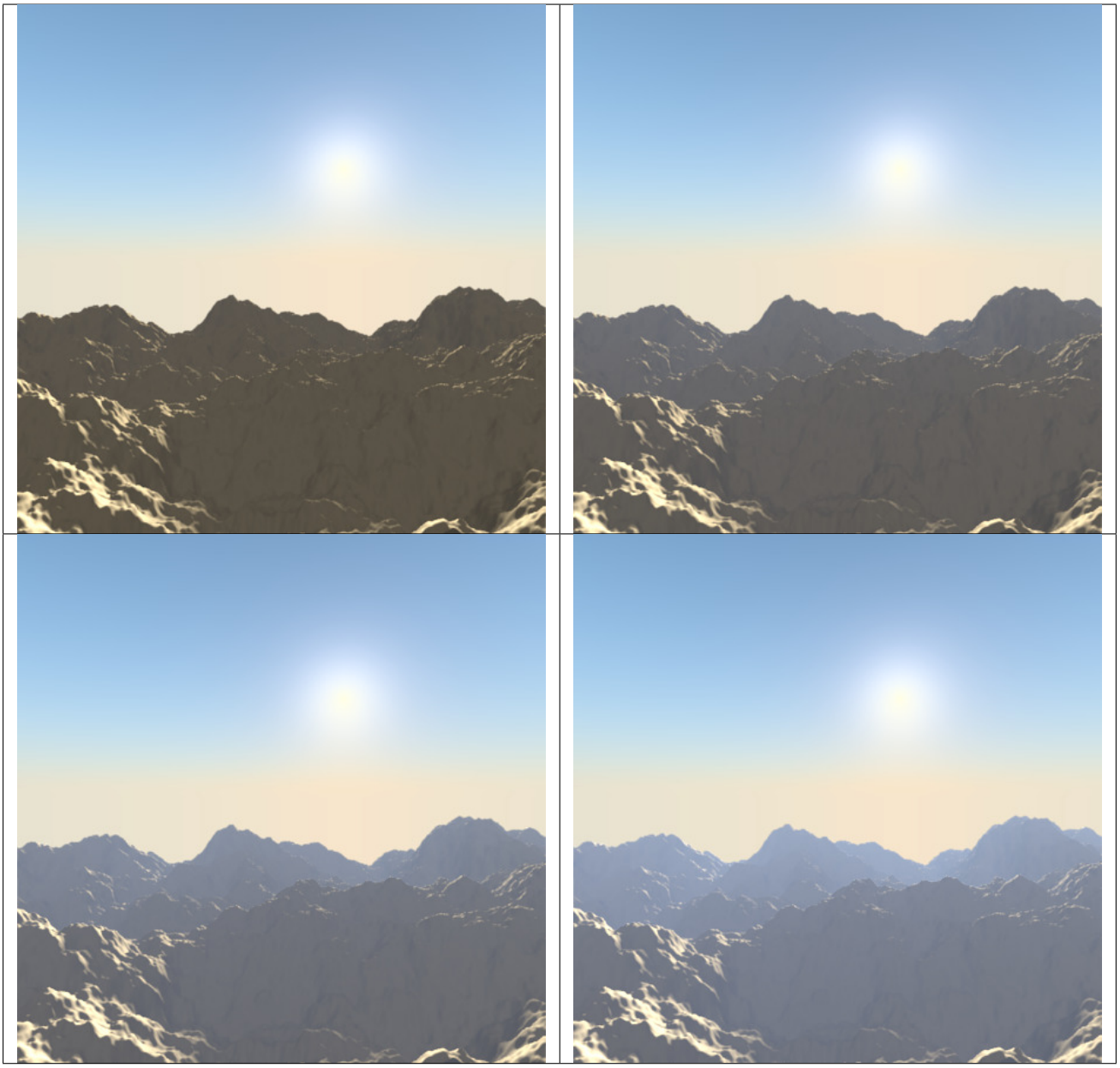


Table 2.93: Inscattering: 1.0.

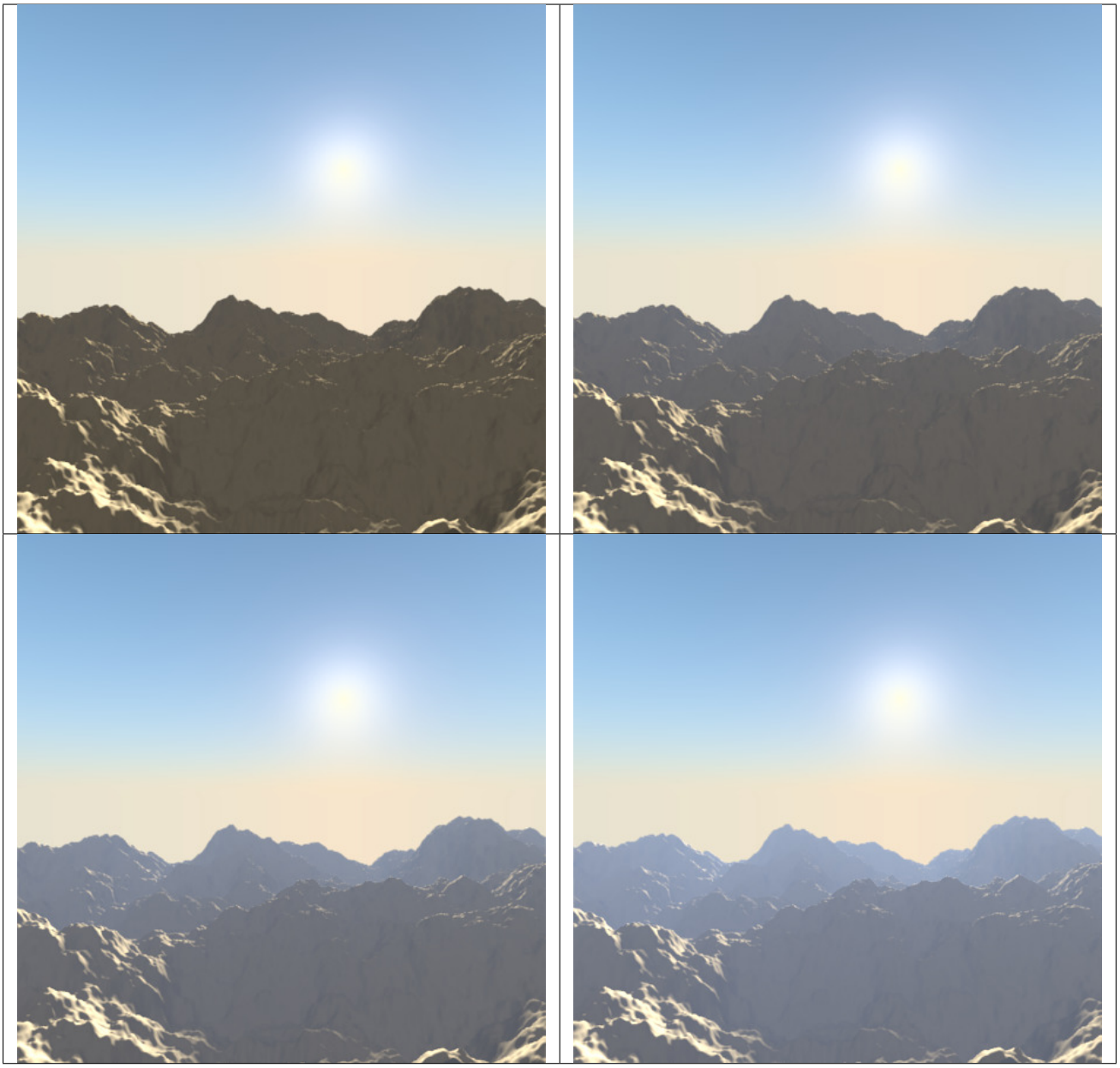


Table 2.94: Extinction: 1.0.

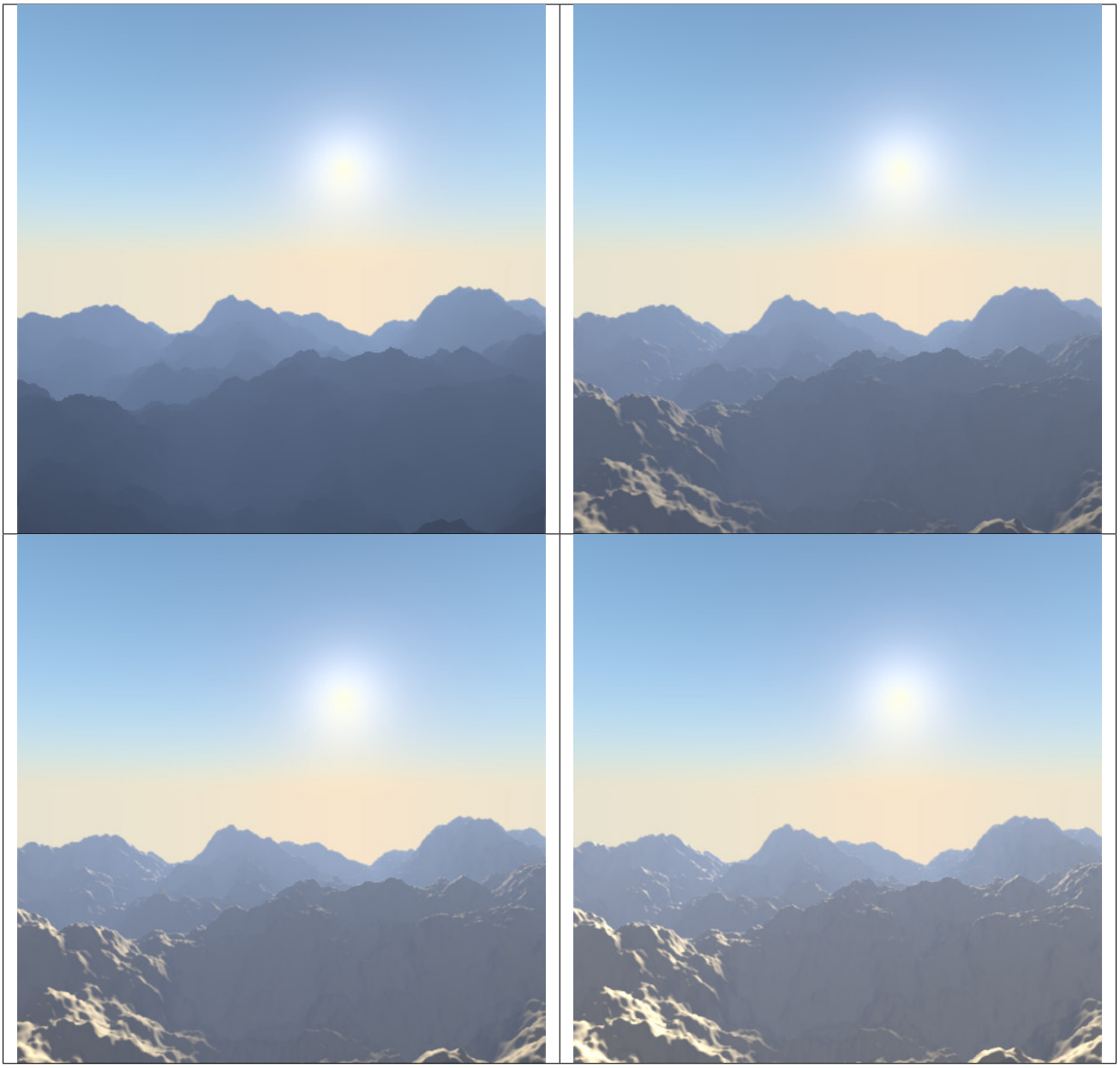


Table 2.95: Distance: 4.0.

Hints and limitations

To always have the *Sun* pointing at the camera center, you can use a *Track To constraint* on the sun object, with the camera as target, and -Z as the "To" axis (use either X or Y as "Up" axis). This way, to modify height/position of the sun in the rendered picture, you just have to move it; orientation is automatically handled by the constraint. Of course, if your camera itself is moving, you should also add e.g. a *Copy Location constraint* to your *Sun* lamp, with the camera as target and the *Offset* option activated... This way, the sun light will not change as the camera moves around.

If you use the default *Add* mixing type, you should use a very dark-blue world color, to get correct "nights"...

This effect works quite well with a *Hemi* lamp, or some ambient occlusion,

to fill in the *Sun* shadows.

Atmosphere shading currently works incorrectly in reflections and refractions and is only supported for solid shaded surfaces. This will be addressed in a later release.

Spot

Introduction

A *Spot* lamp emits a cone-shaped beam of light from the tip of the cone, in a given direction.

The *Spot* light is the most complex of the light objects and indeed, for a long time, among the most used thanks to the fact that it was the only one able to cast shadows. Nowadays, with a ray tracer integrated into Blender's internal render engine, all lamps can cast shadows (except *Hemi*). Even so, *Spot* lamps' shadow buffers are much faster to render than ray-traced shadows, especially when blurred/softened, and spot lamps also provide other functionality such as "volumetric" halos.

Lamp options

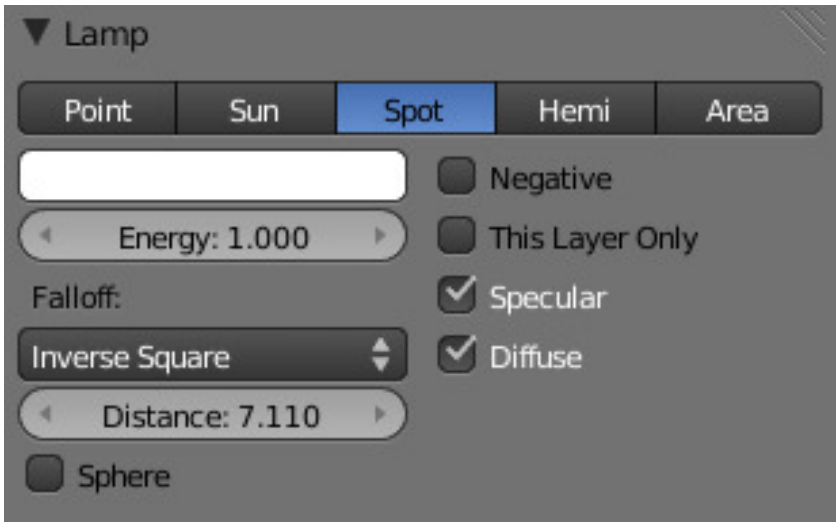


Fig. 2.1819: Common Lamp options of a Spot.

Distance, Energy and Color These settings are common to most types of lamps, and are described in *Light Properties*.

This Layer Only, Negative, Diffuse and Specular These settings control what the lamp affects, as described in *What Light Affects*.

Light Falloff and Sphere These settings control how the light of the *Spot* decays with distance. See *Light Attenuation* for details.

Shadows

Spotlights can use either ray-traced shadows or buffered shadows. Either of the two can provide various extra options. Ray-traced shadows are generally more accurate, with extra capabilities such as transparent shadows, although they are quite slower to render.

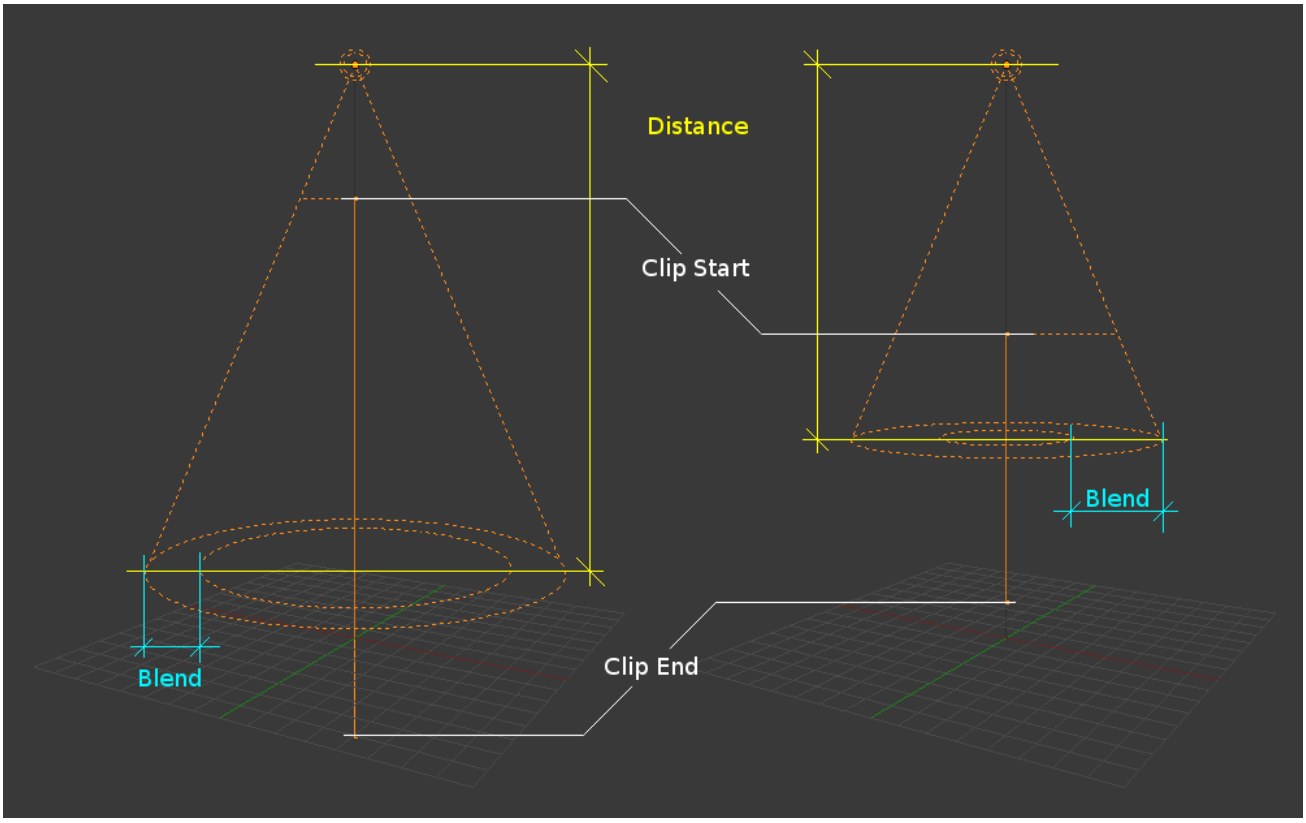


Fig. 2.1820: Changing the Spot options also changes the appearance of the spotlight as displayed in the 3D View.

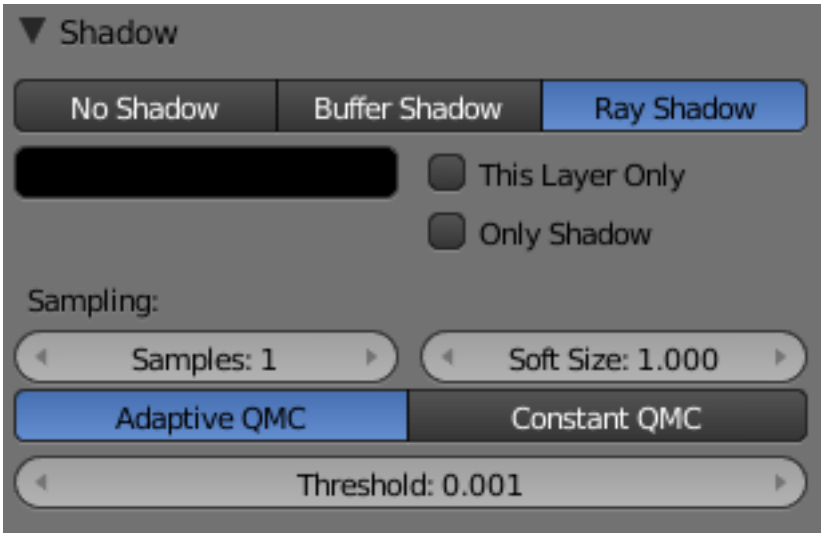


Fig. 2.1821: Shadow panel set to Ray Shadow.

No Shadow Choose this to turn shadows off for this spot lamp. This can be useful to add some discreet directed light to a scene.

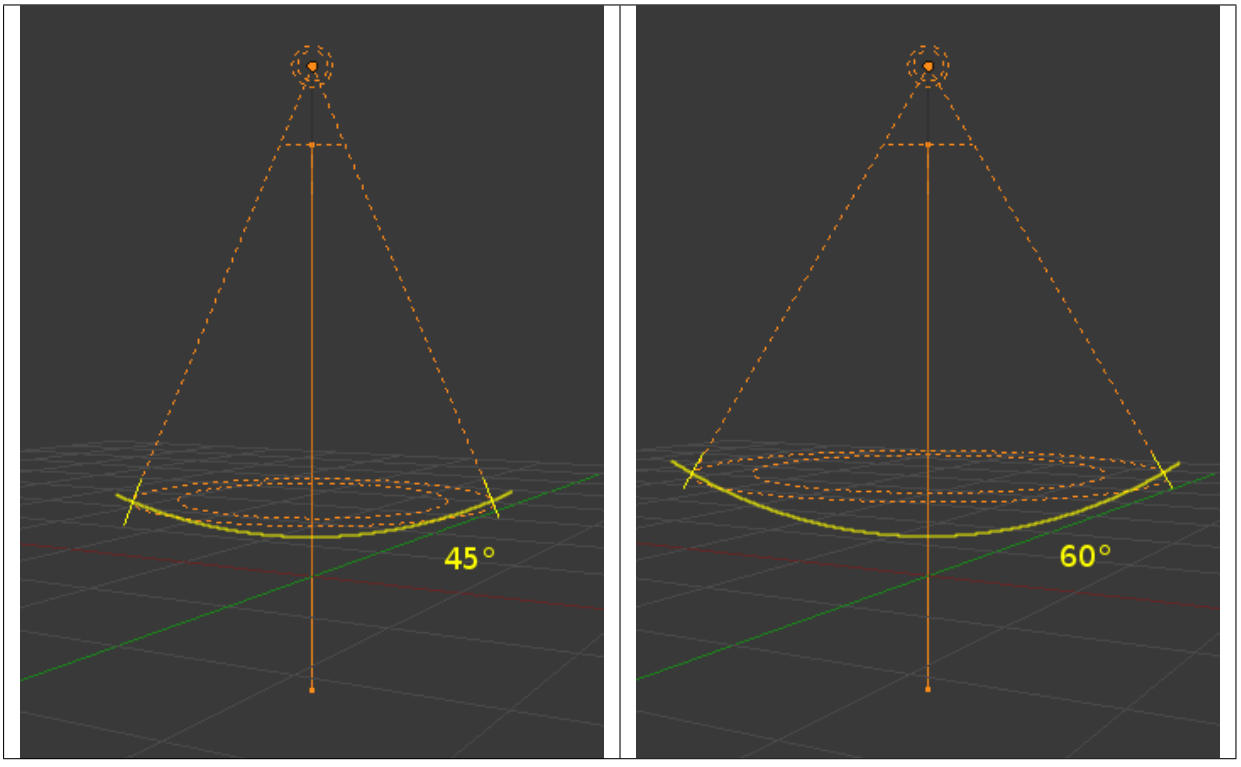
Buffer Shadow *Buffered Shadows* are also known as depth map shadows. Shadows are created by calculating differences in the distance from the light to scene objects. See *Buffered Shadows* for full details on using this feature. Buffered shadows are more complex to set up and involve more faking, but the speed of rendering is a definite advantage. Nevertheless, it shares with other lamp types common shadow options described in *Shadow Panel*.

Ray Shadow The ray-traced shadows settings of this lamp are shared with other lamps, and are described in *Raytraced Properties*.

Spot Shape

Size The size of the outer cone of a *Spot*, which largely controls the circular area a *Spot* light covers. This slider in fact controls the angle at the top of the lighting cone, and can be between (1.0 to 180.0).

Table 2.96: Changing the spot *Size* option.



Blend The *Blend* slider controls the inner cone of the *Spot*. The *Blend* value can be between (0.0 to 1.0). The value is proportional and represents that amount of space that the inner cone should occupy inside the outer cone *Size*.

The inner cone boundary line indicates the point at which light from the *Spot* will start to blur/soften; before this point its light will mostly be full strength. The larger the value of *Blend* the more blurred/soft the edges of the spotlight will be, and the smaller the inner cone's circular area will be (as it starts to blur/soften earlier).

To make the *Spot* have a sharper falloff rate and therefore less blurred/soft edges, decrease the value of *Blend*. Setting *Blend* to 0.0 results in very

sharp spotlight edges, without any transition between light and shadow.

The falloff rate of the *Spot* lamp light is a ratio between the *Blend* and *Size* values; the larger the circular gap between the two, the more gradual the light fades between *Blend* and *Size*.

Blend and *Size* only control the *Spot* light cone's aperture and softness ("radial" falloff); they do not control the shadow's softness as shown below.

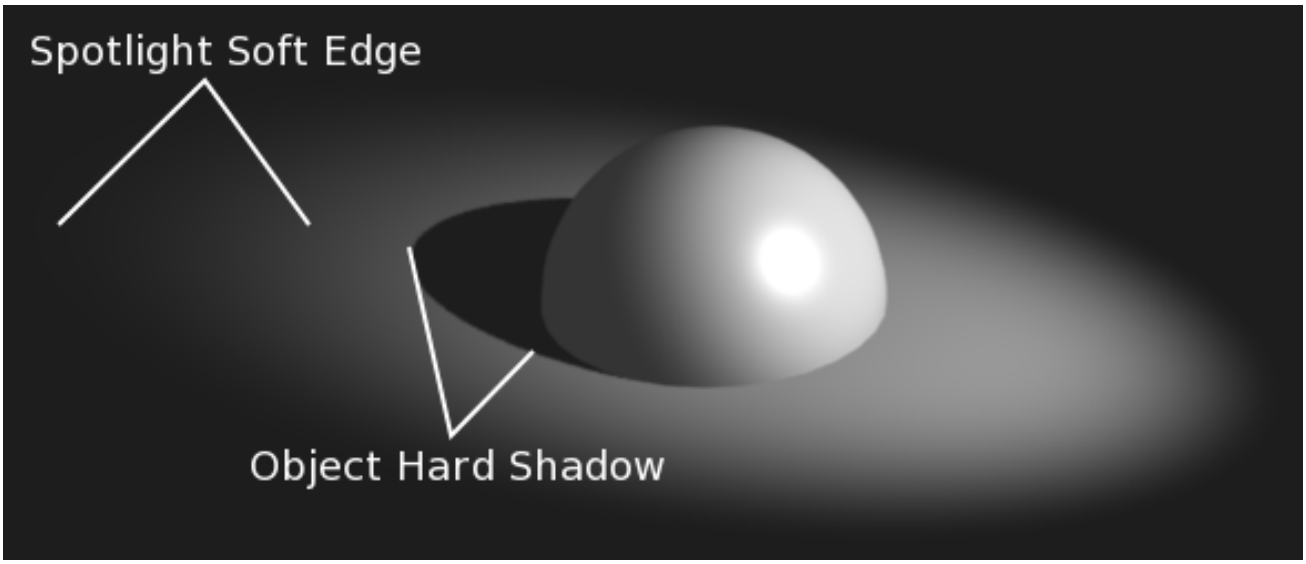


Fig. 2.1822: Render showing the soft edge spotlighted area and the sharp/hard object shadow.

Notice in the picture above that the object's shadow is sharp as a result of the ray tracing, whereas the spotlight edges are soft. If you want other items to cast soft shadows within the *Spot* area, you will need to alter other shadow settings.

Square The *Square* button makes a *Spot* light cast a square light area, rather than the default circular one.

Show Cone Draw a transparent cone in 3D View to visualize which objects are contained in it.

Halo Adds a volumetric effects to the spot lamp. See *Spot Halos*.

Spot Buffered Shadows

Spotlights can use either *Raytraced Shadows* or buffered shadows. Either of the two can provide various extra options.

Raytraced shadows are generally more accurate, with extra capabilities such as transparent shadows, although they are quite slower to render.

Buffered shadows are more complex to set up and involve more faking, but the speed of rendering is a definite advantage. Nevertheless, it shares with other lamp types common shadows options described in *Shadow Panel*.

Shadow Buffer Types

When the *Buffer Shadow* button is activated, the currently selected *Spot* light generates shadows, using a "shadow buffer" rather than using ray-

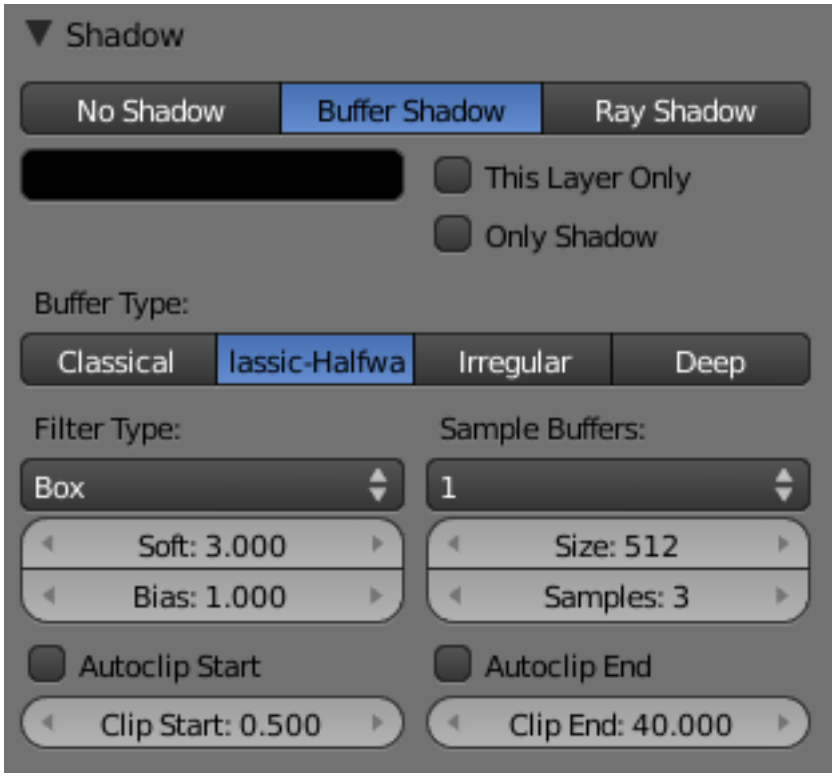


Fig. 2.1823: Buffer Shadow enabled for a Spot lamp.

tracing, and various extra options and buttons appear in the *Shadow* panel.

Buffer Type There more than one way to generate buffered shadows. The shadow buffer generation type controls which generator to use.

There are four shadow generation types, those being:

- Classical
- Classic-Halfway
- Irregular
- Deep

For more information on the different shadow generation methods see these links:

- Development Release Logs 2.43: Irregular Shadow Buffer.
- Blender Nation: Blender Gets Irregular Shadow Buffers.
- Development Release Logs 2.43: Shadow Buffer Halfway Average.

"Classical" and "Classic-Halfway"

Classical A shadow generation which used to be the Blender default and unique method for generation of buffered shadows. It used an older way of generating buffered shadows, but it could have some problems with accuracy of the generated shadows and can be very sensitive to the resolution of the shadow buffer *Shadow Buffer* → *Size*, different *Bias* values, and all the self-shadowing issues that brings up.

The *Classical* method of generating shadows is obsolete and is really only still present to allow for backward compatibility with older versions of

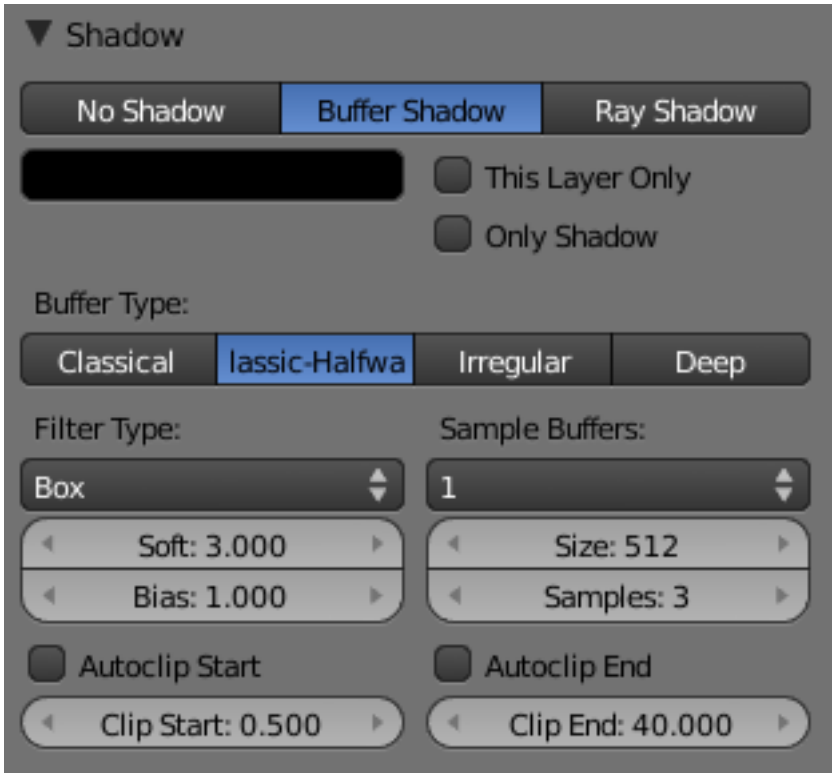


Fig. 2.1824: Buffer Shadowset to Classic-Halfway.

Blender. In most other cases you will want to use *Classic-Halfway* instead.

Classic-Halfway This shadow buffer type is an improved shadow buffering method and is the default option selected in Blender. It works by taking an averaged reading of the first and second nearest Z depth values allowing the *Bias* value to be lowered and yet not suffer as much from self-shadowing issues.

Not having to increase *Bias* values helps with shadow accuracy, because large *Bias* values can mean small faces can lose their shadows, as well as preventing shadows being overly offset from the larger *Bias* value.

Classic-Halfway does not work very well when faces overlap, and biasing problems can happen.

Here are now the options specific to these generation methods:

Size The *Size* number button can have a value from (512 to 10240). *Size* represents the resolution used to create a shadow map. This shadow map is then used to determine where shadows lay within a scene.

As an example, if you have a *Size* with a value of 1024, you are indicating that the shadow data will be written to a buffer which will have a *square* resolution of 1024×1024 pixels/samples from the selected spotlight.

The higher the value of *Size*, the higher resolution and accuracy of the resultant shadows, assuming all other properties of the light and scene are the same, although more memory and processing time would be used. The reverse is also true – if the *Size* value is lowered, the resultant shadows can be of lower quality, but would use less memory and take less processing time to calculate.

As well as the *Size* value affecting the quality of generated shadows, another property of *Spot* lamps that affects the quality of their buffered shad-

ows is the angle of the spotlights lighted area (given in the *Spot Shape* panel's *Size* field).

As the spot shape *Size* value is increased, the quality of the cast shadows degrades. This happens because when the *Spot* lighted area is made larger (by increasing spot shape *Size*), the shadow buffer area has to be stretched and scaled to fit the size of the new lighted area.

The *Size* resolution is not altered to compensate for the change in size of the spotlight, so the quality of the shadows degrades. If you want to keep the generated shadows the same quality, as you increase the spot shape *Size* value, you also need to increase the buffer *Size* value.

Note: The above basically boils down to

If you have a spotlight that is large you will need to have a larger buffer *Size* to keep the shadows good quality. The reverse is true also – the quality of the generated shadows will usually improve (up to a point) as the *Spot* lamp covers a smaller area.

Filter Type The *Box*, *Tent*, and *Gauss* filter types control what filtering algorithm to use to anti-alias the buffered shadows.

They are closely related to the *Samples* number button, as when this setting is set to 1, shadow filtering is disabled, so none of these buttons will have any effect what soever.

Box The buffered shadows will be anti-aliased using the "box" filtering method. This is the original filter used in Blender. It is relatively low quality and is used for low resolution renders, as it produces very sharp anti-aliasing. When this filter is used, it only takes into account oversampling data which falls within a single pixel, and does not take into account surrounding pixel samples. It is often useful for images which have sharply angled elements and horizontal/vertical lines.

Tent The buffered shadows will be anti-aliased using the "tent" filtering method. It is a simple filter that gives sharp results, an excellent general purpose filtering method. This filter also takes into account the sample values of neighboring pixels when calculating its final filtering value.

Gauss The buffered shadows will be anti-aliased using the "Gaussian" filtering method. It produces a very soft/blurry anti-aliasing. As result, this filter is excellent with high resolution renders.

The *Anti-Aliasing page* in the Render chapter will give more information on the various filtering/distribution methods and their uses.

Samples The *Samples* number button can have a value between (1 and 16). It controls the number of samples taken per pixel when calculating shadow maps.

The higher this value, the more filtered, smoothed and anti-aliased the shadows cast by the current lamp will be, but the longer they will take to calculate and the more memory they will use. The anti-aliasing method used is determined by having one of the *Box*, *Tent* or *Gauss* buttons activated (see above).

Having a *Samples* value of 1 is similar to turning off anti-aliasing for buffered shadows.

Soft The *Soft* number button can have a value between (1.0 to 100.0). It indicates how wide an area is sampled when doing anti-aliasing on buffered

shadows. The larger the *Soft* value, the more graduated/soft the area that is anti-aliased/softened on the edge of generated shadows.

Sample Buffers The *Sample Buffers* setting can be set to values (1, 4 or 9), and represents the number of shadow buffers that will be used when doing anti-aliasing on buffered shadows.

This option is used in special cases, like very small objects which move and need to generate really small shadows (such as strands). It appears that normally, pixel width shadows do not anti-alias properly, and that increasing *Buffer Size* does not help much.

So this option allows you to have a sort of extra sample pass, done above the regular one (the one controlled by the *Box* / *Tent* / *Gauss*, *Samples* and *Soft* settings).

The default 1 value will disable this option.

Higher values will produce a smoother anti-aliasing – but be careful: using a *Sample Buffers* of 4 will require four times as much memory and process time, and so on, as Blender will have to compute that number of sample buffers.

"Irregular"

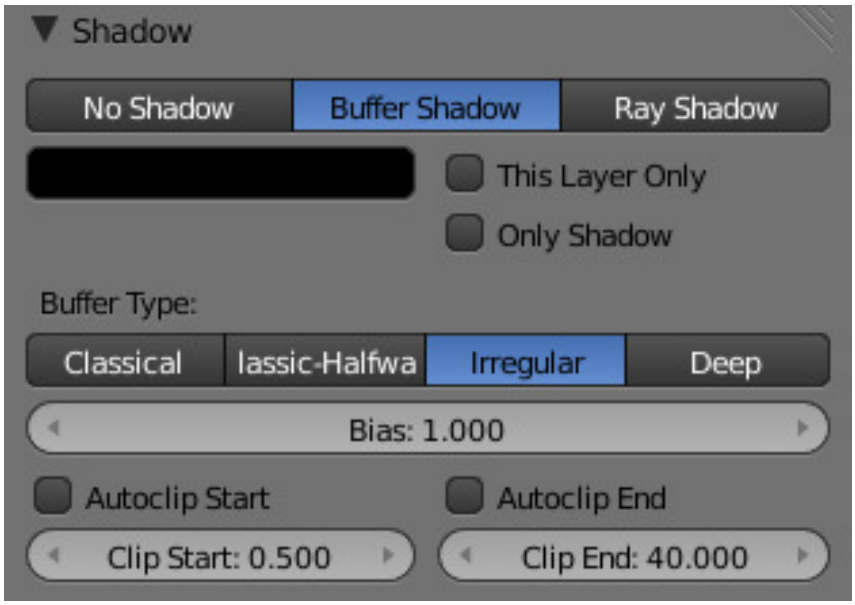


Fig. 2.1825: Buffer Shadow set to Irregular.

Irregular shadow method is used to generate sharp/hard shadows that are placed as accurately as raytraced shadows. This method offers very good performance because it can be done as a multi-threaded process.

This method supports transparent shadows. To do so, you will first need to setup the shadow setting for the object which will receive the transparent shadow *Material* → *Shadow* → *Cat Buffer Shadows and Buffer Bias*.

Deep generation method

Compress Deep shadow map compression threshold.

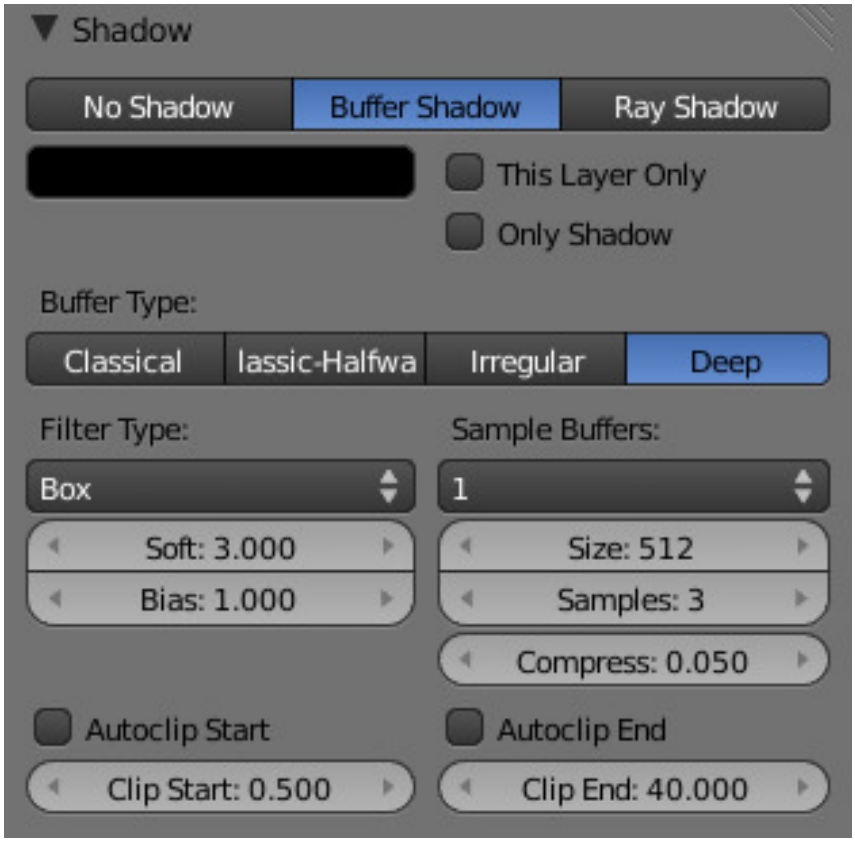


Fig. 2.1826: Buffer Shadow set to Deep.
Deep Shadow buffer supports transparency and better filtering, at the cost of more memory usage and processing time.

Common options

The following settings are common to all buffered shadow generation method.

Bias The *Bias* number button can have a value between (0.001 to 5.0). *Bias* is used to add a slight offset distance between an object and the shadows cast by it. This is sometimes required because of inaccuracies in the calculation which determines weather an area of an object is in shadow or not.

Making the *Bias* value smaller results in the distance between the object and its shadow being smaller. If the *Bias* value is too small, an object can get artifacts, which can appear as lines and interference patterns on objects. This problem is usually called "self shadowing", and can usually be fixed by increasing the *Bias* value, which exists for that purpose!

Other methods for correcting self shadowing include increasing the size of the *Shadow Buffer Size* or using a different buffer shadow calculation method such as *Classic-Halfway* or *Irregular*.

Self shadowing interference tends to affect curved surfaces more than flat ones, meaning that if your scene has a lot of curved surfaces it may be necessary to increase the *Bias* value or *Shadow Buffer Size* value.

Having overly large *Bias* values not only places shadows further away from their casting objects, but can also cause objects that are very small to not cast any shadow at all. At that point altering *Bias*, *Shadow Buffer Size* or *Spot Size* values, among other things, may be required to fix the problem.

Note: Finer Bias tuning

You can now refine the *Bias* value independently for each *Material*, using the *Bias* slider (*Material* menu, *Shadow* panel). This value is a factor by which the *Bias* value of each *Spot* buffered shadows lamp is multiplied, each time its light hits an object using this material. The (0.0 and 1.0) values are equivalent. They do not alter the lamp's *Bias* original value.

Clip Start & Clip End When a *Spot* light with buffered shadows is added to a scene, an extra line appears on the *Spot* 3D View representation.

The start point of the line represents *Clip Start* 's value and the end of the line represents *Clip End* 's value. *Clip Start* can have a value between (0.1 to 1000.0), and *Clip End*, between (1.0 to 5000.0). Both values are represented in Blender Units.

Clip Start indicates the point after which buffered shadows can be present within the *Spot* light area. Any shadow which could be present before this point is ignored and no shadow will be generated.

Clip End indicates the point after which buffered shadows will not be generated within the *Spot* light area. Any shadow which could be present after this point is ignored and no shadow will be generated.

The area between *Clip Start* and *Clip End* will be capable of having buffered shadows generated.

Altering the *Clip Start* and *Clip End* values helps in controlling where shadows can be generated. Altering the range between *Clip Start* and *Clip End* can help speed up rendering, save memory and make the resultant shadows more accurate.

When using a *Spot* lamp with buffered shadows, to maintain or increase quality of generated shadows, it is helpful to adjust the *Clip Start* and *Clip End* such that their values closely bound around the areas which they want to have shadows generated at. Minimizing the range between *Clip Start* and *Clip End*, minimizes the area shadows are computed in and therefore helps increase shadow quality in the more restricted area.

Autoclip Start & Autoclip End As well as manually setting *Clip Start* and *Clip End* fields to control when buffered shadows start and end, it is also possible to have Blender pick the best value independently for each *Clip Start* and *Clip End* field.

Blender does this by looking at where the visible vertices are when viewed from the *Spot* lamp position.

Hints

Any object in Blender can act as a camera in the 3D View. Hence you can select the *Spot* light and switch to a view from its perspective by pressing `Ctrl-Numpad0`.

Spot Volumetric Effects

Spot lights also can produce "volumetric" effects. See *Volumetric Light* for more information about what it means.

Halo The *Halo* button allows a *Spot* lamp to have a volumetric effect applied to it. This button must be active if the volumetric effect is to be visible. Note

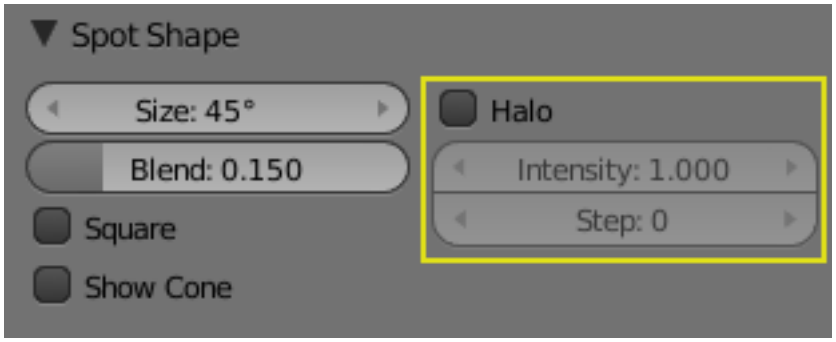


Fig. 2.1827: Spot lamps's Halo options.

that if you are using buffered shadows, you have extra options described in the *Spot Buffered Shadows* page.

Intensity The *Intensity* slider controls how intense/dense the volumetric effect is that is generated from the light source. The lower the value of the *Intensity* slider, the less visible the volumetric effect is, while higher *Intensity* values give a much more noticeable and dense volumetric effect.

Step This field can have a value between (0 to 12). It is used to determine whether this *Spot* will cast volumetric shadows, and what quality those volumetric shadows will have. If *Step* is set to a value of 0, then no volumetric shadow will be generated. Unlike most other controls, as the *Step* value increases, the quality of volumetric shadows decreases (but take less time to render), and *vice versa*.

Tip: Step values

A value of 8 for *Halo Step* is usually a good compromise between speed and accuracy.

Blender only simulates volumetric lighting in *Spot* lamps when using its internal renderer. This can lead to some strange results for certain combinations of settings for the light's *Energy* and the halo's *Intensity*. For example, having a *Spot* light with null or very low light *Energy* settings but a very high halo *Intensity* setting can result in a dark/black halo, which would not happen in the real world. Just be aware of this possibility when using halos with the internal renderer.

Note: The halo effect can be greatly enhanced when using buffered shadows: when the halo's *Step* is not null, they can create "volumetric shadows". See the page about *Spot Buffered Shadows* for more information.

See also:

- *Shadows*
- *Spot Lamp*
- *Spot Buffered Shadows*

Hemi Lamp

Fig. 2.1828: Hemi light conceptual scheme.

The *Hemi* lamp provides light from the direction of a 180- hemisphere,

designed to simulate the light coming from a heavily clouded or otherwise uniform sky. In other words, it is a light which is shed, uniformly, by a glowing dome surrounding the scene.

Similar to the *Sun* lamp, the *Hemi* 's location is unimportant, while its orientation is key.

The *Hemi* lamp is represented with four arcs, visualizing the orientation of the hemispherical dome, and a dashed line representing the direction in which the maximum energy is radiated, the inside of the hemisphere.

Options

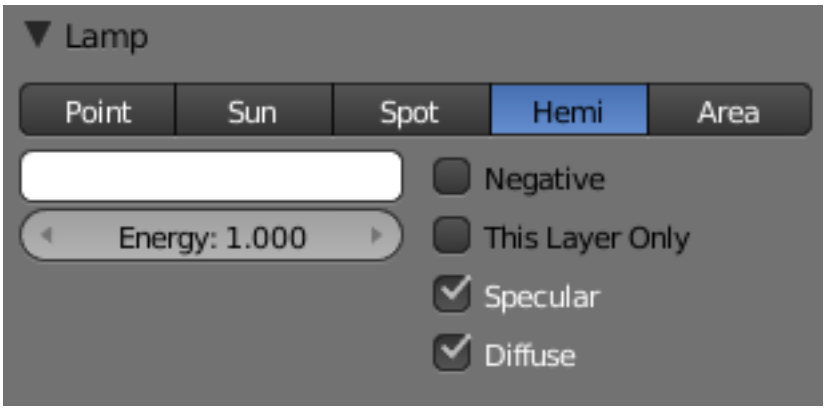


Fig. 2.1829: Hemi lamp's panel.

Energy and Color These settings are common to most types of lamps, and are described in *Light Properties*.

Layer, Negative, Specular, and Diffuse These settings control what the lamp affects, as described in *What Light Affects*.

The *Hemi* lamp has no light falloff settings: it always uses a constant attenuation (i.e. no attenuation).

Since this lamp is the only lamp which cannot cast any shadow, the *Shadow* panel is absent.

Area

Introduction

The *Area* lamp simulates light originating from a surface (or surface-like) emitter. For example, a TV screen, your supermarket's neon lamps, a window, or a cloudy sky are just a few types of area lamp. The area lamp produces shadows with soft borders by sampling a lamp along a grid the size of which is defined by the user. This is in direct contrast to point-like artificial lights which produce sharp borders.

Lamp options

Distance, Energy and Color These settings are common to most types of lamps, and are described in *Light Properties*.

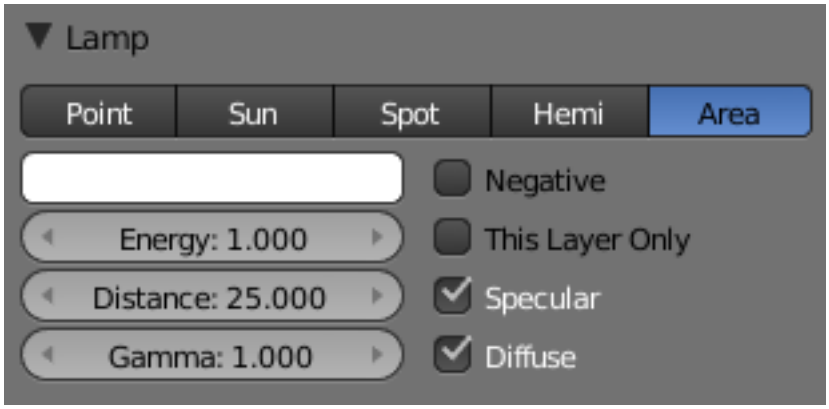


Fig. 2.1830: Commons Options.

Note that the *Distance* setting is much more sensitive and important for *Area* lamps than for others; usually any objects within the range of *Distance* will be blown out and overexposed. For best results, set the *Distance* to just below the distance to the object that you want to illuminate.

Gamma Amount to gamma correct the brightness of illumination. Higher values give more contrast and shorter falloff.

The *Area* lamp does not have light falloff settings. It uses an "inverse quadratic" attenuation law. The only way to control its falloff is to use the *Distance* and/or *Gamma* settings.

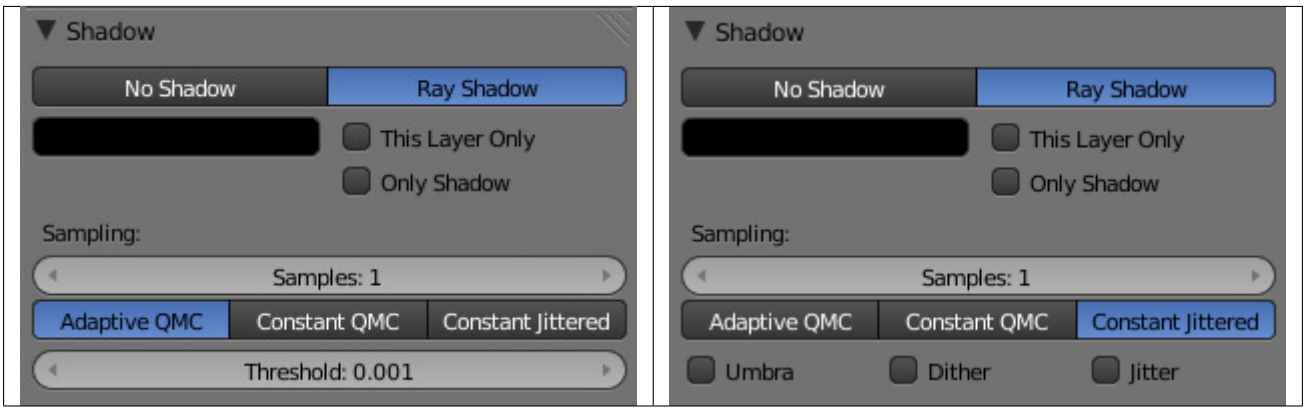
This Layer Only, Negative, Specular and Diffuse These settings control what the lamp affects, as described in *What Light Affects*.

Shadows

Area light ray-traced shadows are described here: *Raytraced Shadows*.

When an *Area* light source is selected, the *Shadow* panel has the following default layout:

Table 2.97: Constant Jittered settings.



Area Shape

The shape of the area light can be set to *Square* or *Rectangle*.

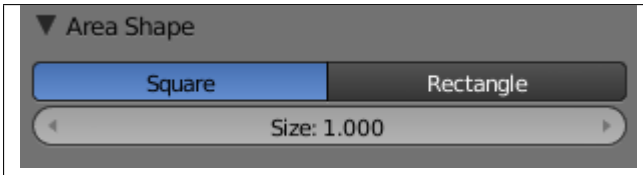


Fig. 2.1833: Square options.

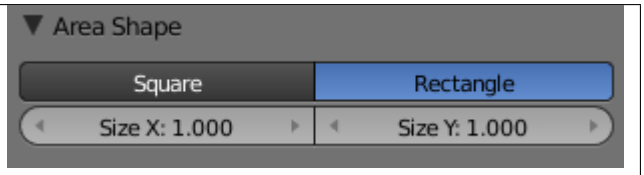


Fig. 2.1834: Rectangle options.

Square / Rectangular Emit light from either a square or a rectangular area

Size / Size X / Size Y Dimensions for the *Square* or *Rectangle*

Note: Shape Tips

Choosing the appropriate shape for your *Area* light will enhance the believability of your scene. For example, you may have an indoor scene and would like to simulate light entering through a window. You could place a *Rectangular* area lamp in a window (vertical) or from neons (horizontal) with proper ratios for *Size X* and *Size Y*. For the simulation of the light emitted by a TV screen a vertical *Square* area lamp would be better in most cases.

Area Raytraced Shadows

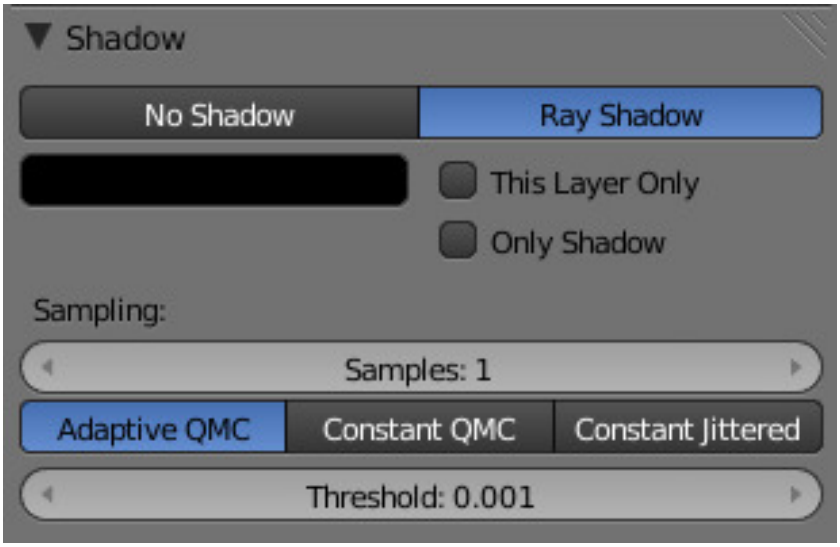


Fig. 2.1835: Adaptive QMC settings.

The *Area* light source can only cast ray-traced shadows. The ray-traced shadows settings of this lamp are mostly shared with other lamps, as described in *Raytraced Properties*. However, there are some specifics with this lamp, which are detailed below:

Shadow Samples

Samples This have the same role as with other lamps, but when using a *Rectangle* Area lamp, you have two samples settings: *Samples X* and *Samples Y*, for the two axes of the area plane. Note also that when using the *Constant Jittered* sample generator method, this is more or less equivalent to the

number of virtual lamps in the area. With QMC sample generator methods, it behaves similarly to with *Lamp* or *Spot* lamps.

Sample Generator Types

Adaptive QMC / Constant QMC These common setting are described in *Shadow Panel*.

Constant Jittered The *Area* lamp has a third sample generator method, *Constant Jittered*, which is more like simulating an array of lights. It has the same options as the old one: *Umbra*, *Dither* and *Jitter*.

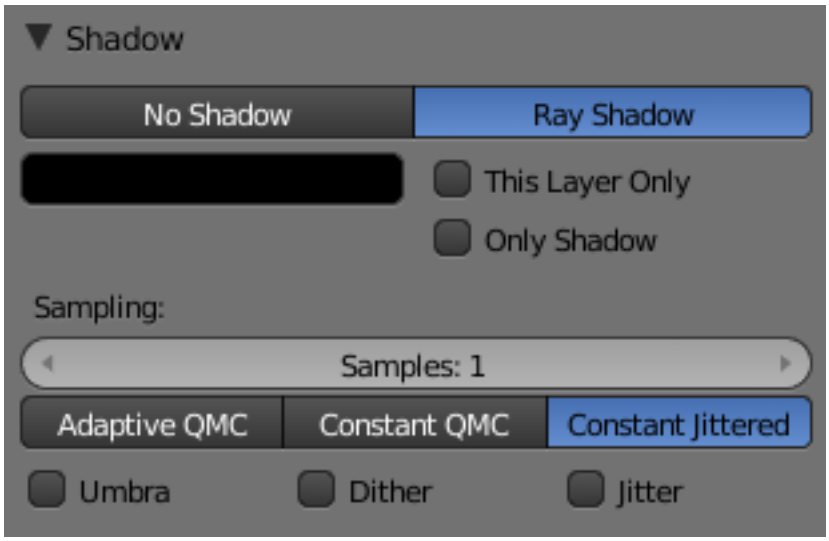


Fig. 2.1836: Constant Jittered settings.

The following three parameters are only available when using the *Constant Jittered* sample generator method, and are intended to artificially boost the "soft" shadow effect, with possible loss in quality:

Umbra Emphasizes the intensity of shadows in the area fully within the shadow rays. The light transition between fully shadowed areas and fully lit areas changes more quickly (i.e. a sharp shadow gradient). You need *Samples* values equal to or greater than 2 to see any influence of this button.

Dither Applies a sampling over the borders of the shadows, similar to the way anti-aliasing is applied by the *OSA* button on the borders of an object. It artificially softens the borders of shadows; when *Samples* is set very low, you can expect poor results, so *Dither* is better used with medium *Samples* values. It is not useful at all with high *Samples* values, as the borders will already appear soft.

Jitter Adds noise to break up the edges of solid shadow samples, offsetting them from each other in a pseudo-random way. Once again, this option is not very useful when you use high *Samples* values where the drawback is that noise generates quite visible graininess.

Technical Details

The Fig. *Principles behind the Area light.* picture helps to understand how the soft shadows are simulated.

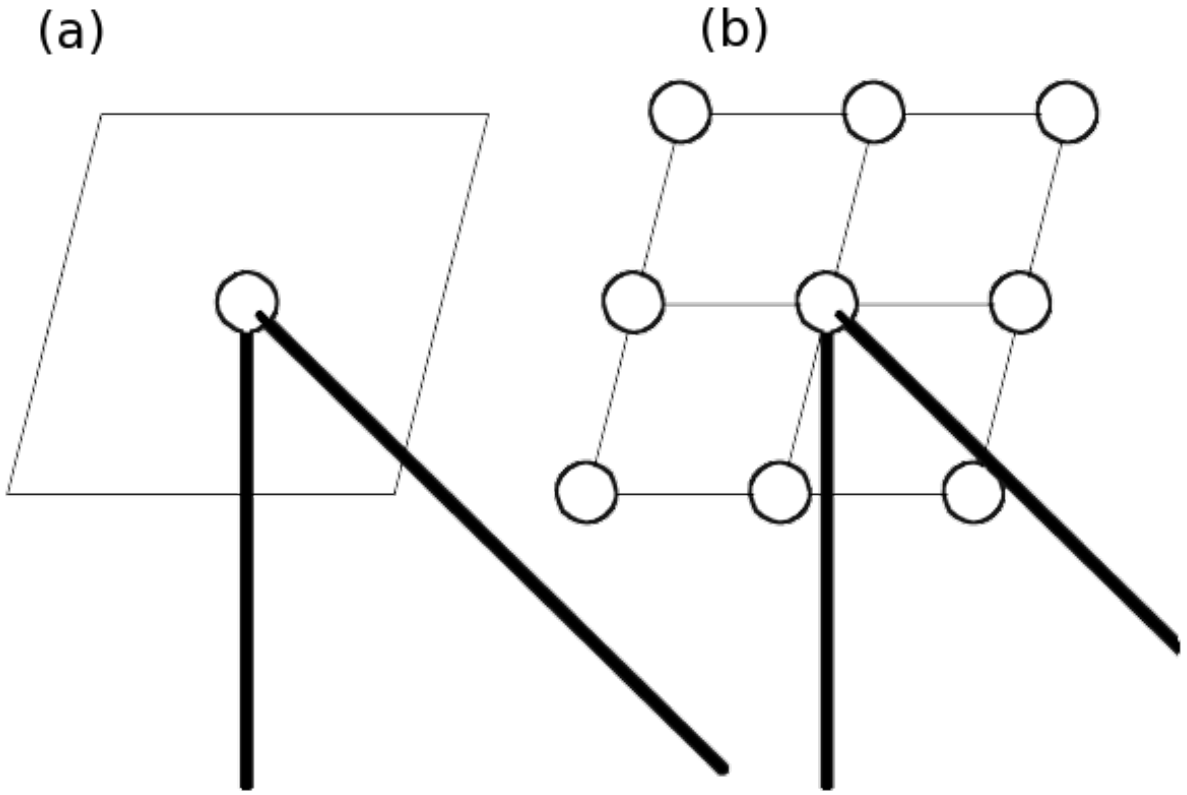


Fig. 2.1837: Principles behind the Area light.

"(a)" is the *Area* light as defined in Blender. If its shape is *Square*, then the softness of the shadow is defined by the number of light *Samples* in each direction of the shape. For example, "(b)" illustrates the equivalent case of an *Area* light (*Square* shape), with *Samples* set at 3 on the *Shadow and Spot* panel.

The *Area* lamp is then considered as a grid with a resolution of three in each direction, and with a light "dupliverted" at each node for a total of nine lights.

In case "(a)", the energy *E* is *E*/1, and in case "(b)", the energy of each individual pseudo-light is equal to *E*/ (nbr. of lights). Each pseudo-light produces a faint shadow (proportional to its energy), and the overlay of the shadows produces the soft shadow (it is darker where the individual shadows overlap, and lighter everywhere else).

Hints

You will note that changing the *Size* parameter of your area lamp does not affect the lighting intensity of your scene. On the other hand, rescaling the lamp using the `S` in the 3D View could dramatically increase or decrease the lighting intensity of the scene. This behavior has been coded this way so that you can fine tune all your light settings and then decide to scale up (or down) the whole scene without suffering from a drastic change in the lighting intensity. If you only want to change the dimensions of your *Area* lamp, without messing with its lighting intensity, you are strongly encouraged to use the *Size* button(s) instead.

If your computer is not very fast, when using the *Constant Jittered* sample generator method, you could find it useful to set a low *Samples* value (like 2) and activate *Umbra*, *Dither*, and/or *Jitter* in order to simulate slightly softer shadows. However, these results will never be better than the same lighting with high *Samples* values.

Lighting Rigs

A rig is a standard setup and combination of objects; there can be lighting rigs, or armature rigs, etc. A rig provides a basic setup and allows you to start from a known point and go from there. Different rigs are used for different purposes and emulate different conditions; the rig you start with depends on what you want to convey in your scene. Lighting can be very confusing, and the defaults do not give good results. Further, very small changes can have a dramatic effect on the mood and colors.

In all the lighting rigs, the default camera is always positioned nearly 15 degrees off dead-on, about 25 BU (Blender Units) back and 9 BU to the side of the subject, at eye level, and uses a long lens of 80 mm. Up close, a 35 mm lens will distort the image. A long lens takes in more of the scene. A dead-on camera angle is too dramatic and frames too wide a scene to take in. So now you know; next time you go to a play, sit off-center and you will not miss the action happening on the sidelines and will have a greater appreciation for the depth of the set. Anyway, enough about camera angles; this is about lighting.

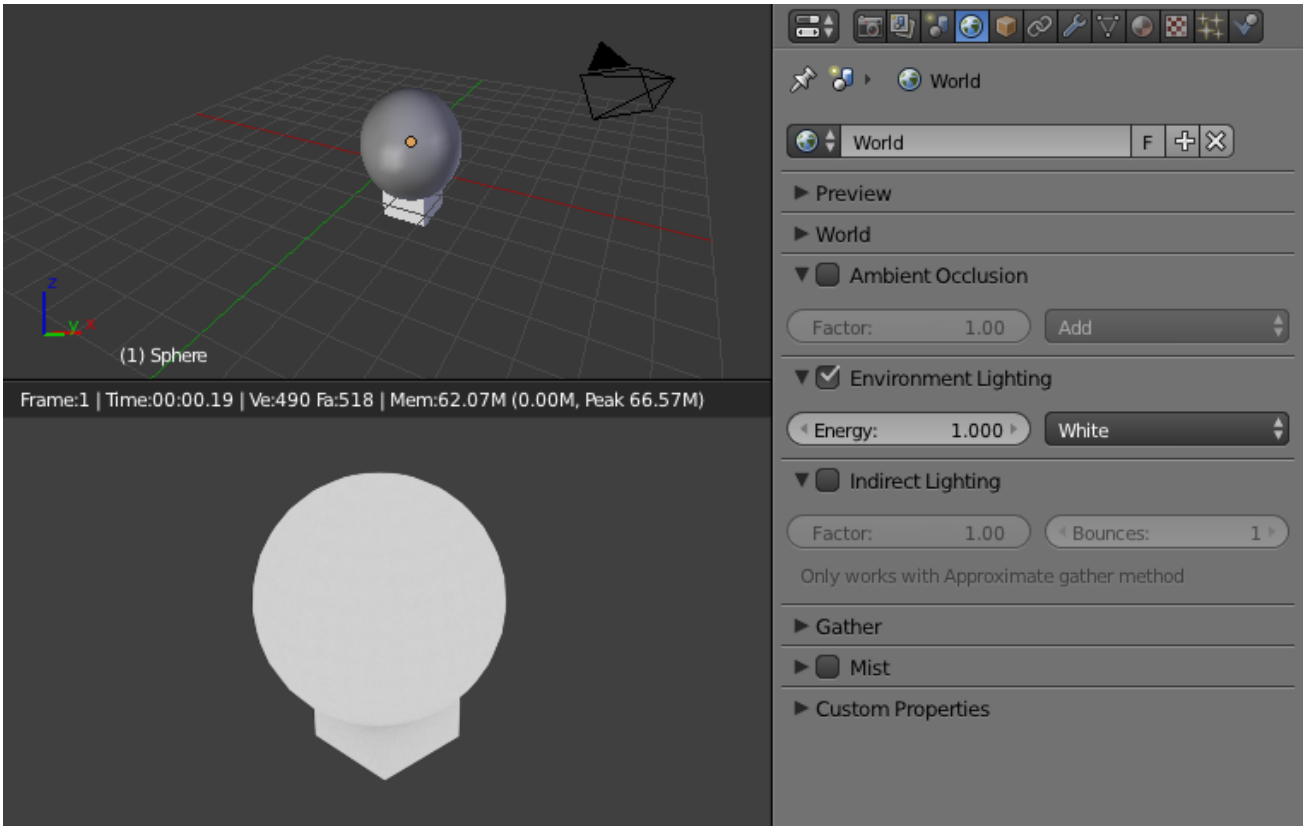


Fig. 2.1838: Environment (Ambient) lighting only.

Environment or Ambient Only

In the *World* tab, there is a panel *Environment Lighting*, where you enable environment or ambient lighting of your scene. Ambient light is the scattered light that comes from sunlight being reflected off every surface it hits, hitting your object, and traveling to camera.

Ambient light illuminates, in a perfectly balanced, shadeless way, without casting shadows. You can vary the intensity of the ambient light across your scene via *ambient occlusion*. The ambient color is a sunny white.

Single Rig

The sole, or key, spot light rig provides a dramatic, showy, yet effective illumination of one object or a few objects close together. It is a single *Spot* light, usually with a hard edge. Halos are enabled in this render to remind you of a smoky nightclub scene. It is placed above and directly in front of the subject; in this case 10 BU in front and 10 BU high, just like a stage, it shines down at about a 40 degrees angle. We use quadratic attenuation.

You can make the spot wider by increasing *Size Spot Shape* and softening the edge by increasing *Blend Spot Shape*, and parent it to the main actor, so that the spot follows him as he moves around. Objects close to the main actor will naturally be more lit and your viewer will pay attention to them.

Moving this spot directly overhead and pointing down gives the interrogation effect. At the opposite end of the show-off emotional spectrum is one soft candlelight (*Point* lamp, short falloff *Distance*, yellow light)

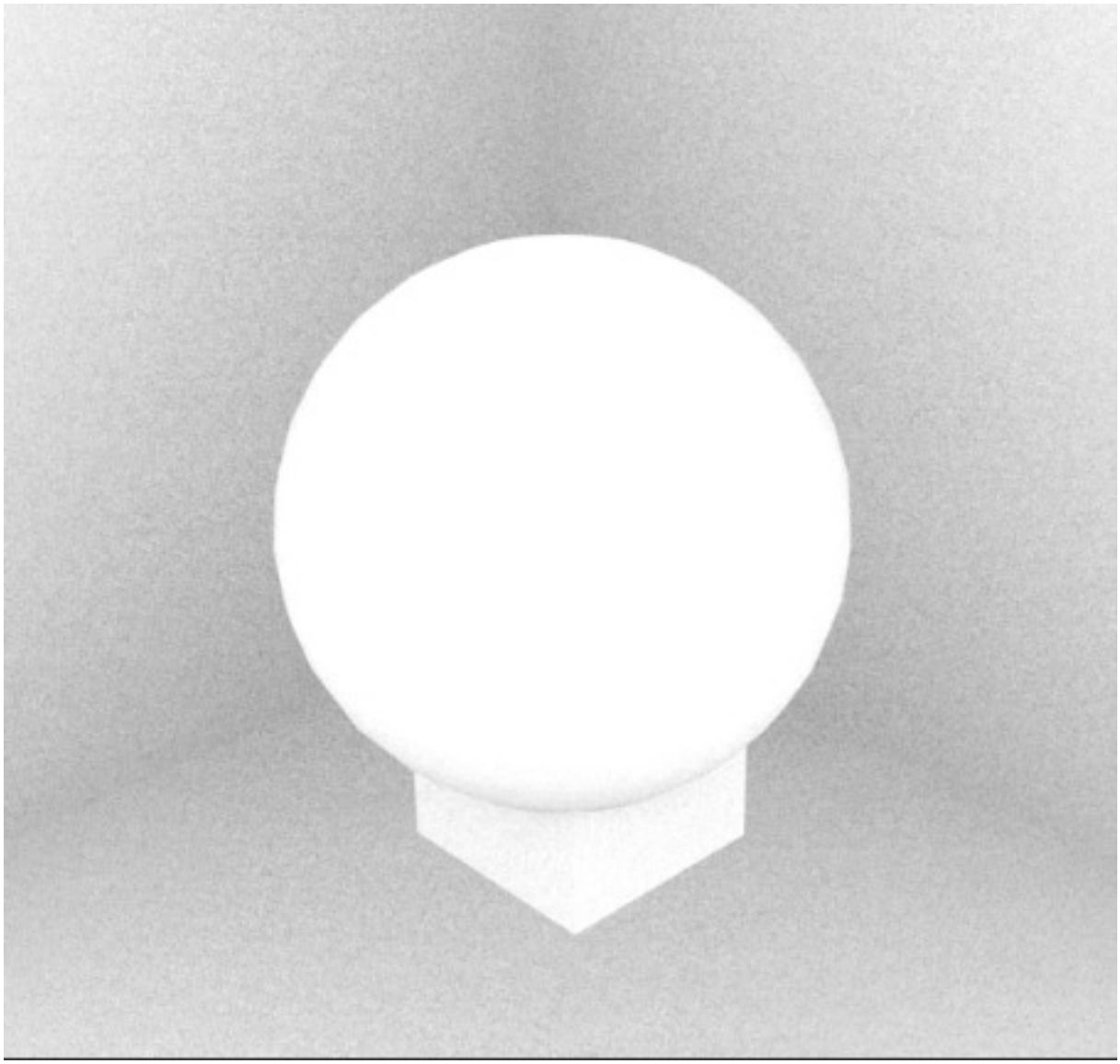


Fig. 2.1839: Ambient occlusion.

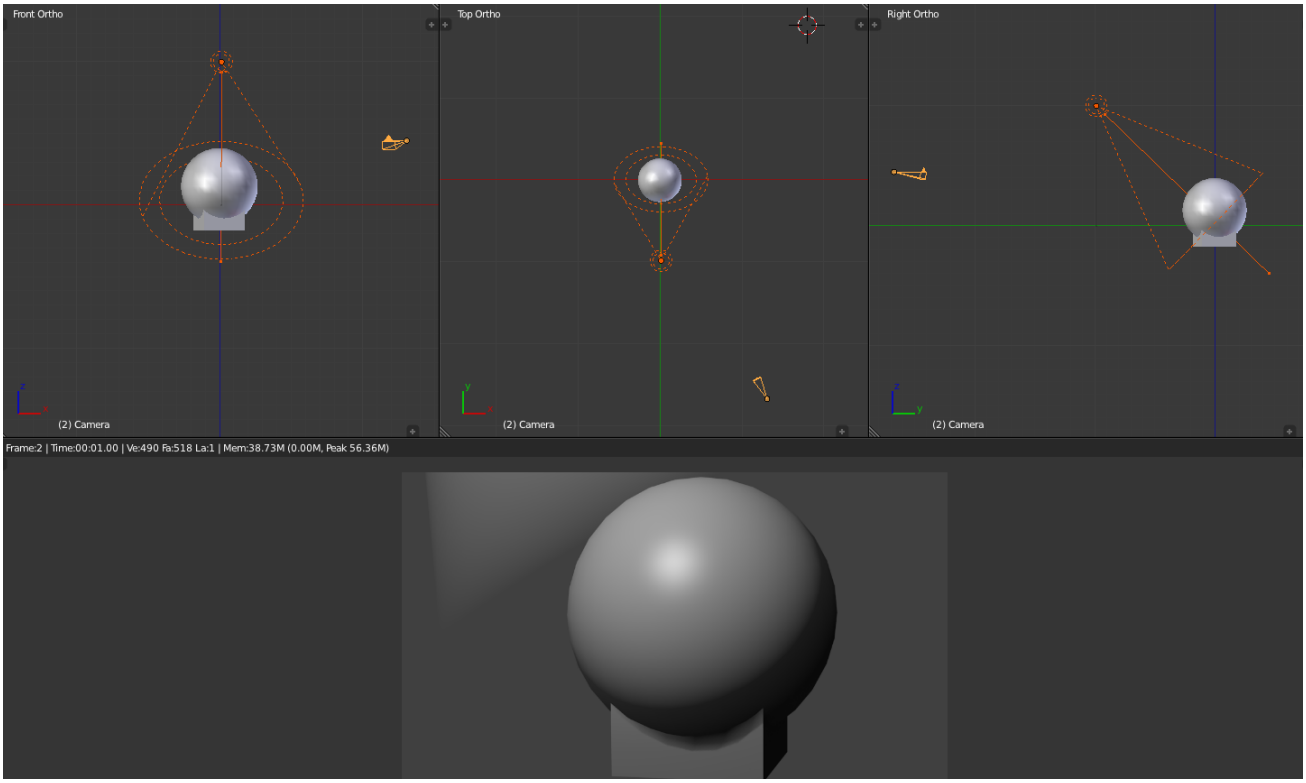


Fig. 2.1840: Standard Spot light rig.

placed really up close to the subject, dramatizing the fearful "lost in the darkness" effect.

Somewhere in the macabre spectrum is a hard spot on the floor shining upward. For fun, grab a flashlight, head into the bathroom and close the door. Turn out the light and hold the flashlight under your chin, pointing up. Look in the mirror and turn it on. From this you can see that lighting, *even* with a single light, varying the intensity, location and direction, changes *everything* in a scene.

Use this rig, with *Environment Lighting* light (and props receiving and being lit by ambient light in their material settings) for scenes that feature one main actor or a product being spotlighted. Do not use this rig for big open spaces or to show all aspects of a model.

Two-Point Rig

The two-point lighting rig provides a balanced illumination of an object. Shown to the right are the views of the standard two-point lighting rig. It is called the two-point because there are two points of light. The standard two-point lighting rig provides a balanced illumination of untextured objects hanging out there in 3D space. This rig is used in real studios for lighting a product, especially a glossy one.

Both lights are almost the same but do different things. Both emulate very wide, soft light by being *Hemi*. In real life, these lights bounce light off the inside of a silver umbrella.

Notice how we use low *Energy* to bring out the dimensionality of the sphere; I cannot stress that enough. Hard, bright lights actually flatten

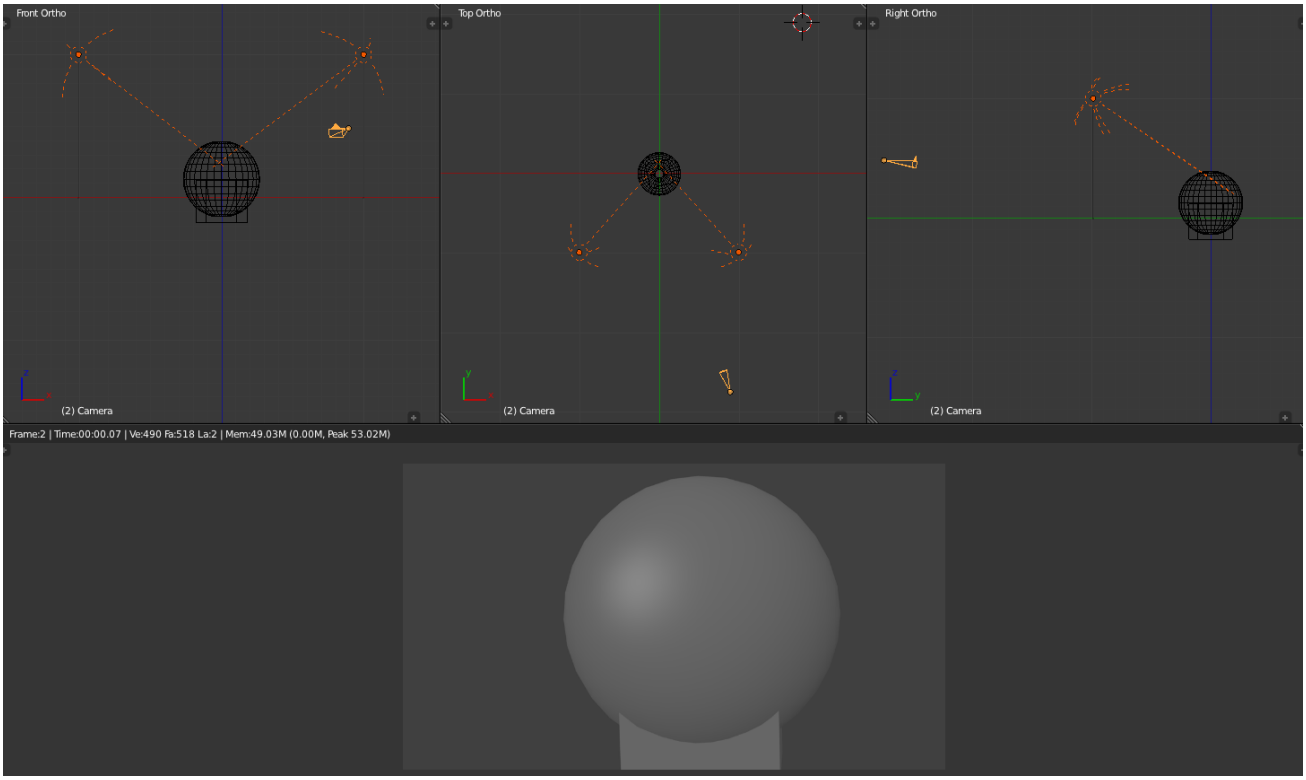


Fig. 2.1841: Standard two-point light rig.

it and make you squint. Soft lights allow your eye to focus. We disable specular for right *Hemi*, so we do not get that shiny forehead or nose.

The lamp on the left however, lets it be known that it is there by enabling specular; specular flare is that bright spot that is off center above midline on the sphere.

Use this rig to give even illumination of a scene, where there is no main focus. The *Hemi* 's will light up background objects and props, so *Environment Lighting* is not that important. At the opposite end of the lighting spectrum, two narrow *Spot* lights at higher power with a hard edge gives a "This is the Police, come out with your hands up" kind of look, as if the subject is caught in the crossfire.

Three-Point Rigs

The standard three-point lighting rig is the most common illumination of objects and scenes bar none. If you want to show off your model, use this rig. As you can see, the untextured unmaterialized sphere seems to come out at you. There are multiple thesis on this rig, and you will use one of two:

- Studio: Used in a real studio to film in front of a green screen or backdrop. Use this rig when you are rendering your CG objects to alpha into the scene so that the lighting on the actors *and* your CG objects is the same.
- Standard: Used in real life to light actors on a set, and gives some backlighting to highlight the sides of actors, making them stand out more and giving them depth.

Studio rig

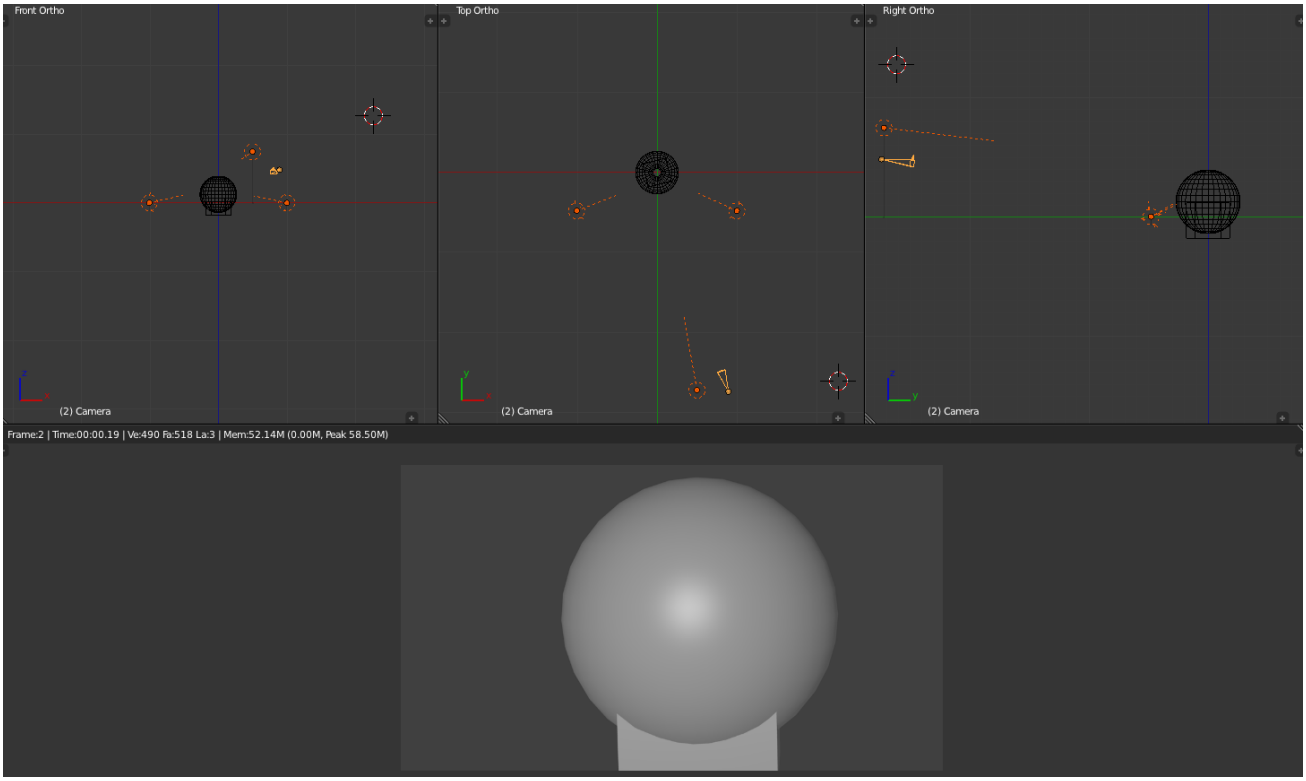


Fig. 2.1842: Studio three-point light rig.

Shown to the right are the "Studio" top, front, and side views of the standard three-point lighting rig. It changes the dynamics of the scene, by making a brighter "key" light give some highlights to the object, while two side "fill" lights soften the shadows created by the key light.

In the studio, use this rig to film a talking head (actor) in front of a green screen, or with multiple people, keeping the key light on the main actor. This rig is also used to light products from all angles, and the side fill lights light up the props.

The key light is the *Area* light placed slightly above and to the left of the camera. It allows the specular to come out. It is about 30 BU back from the subject, and travels with the camera. A little specular shine lets you know there is a light there, and that you are not looking at a ghost. In real life, it is a spot with baffles, or blinders, that limit the area of the light.

The two sidelights are reduced to only fill; each of them are *Hemi* lights placed 20 BU to the side and 5 BU in front of the subject, at ground level. They do not cause a spotshine on the surface by disabling specular, and at ground level, light under the chin or any horizontal surfaces, countering the shadows caused by the key light.

Use this rig to give balanced soft lighting that also highlights your main actor or object. It combines the best of both the single rig and the two-point rig, providing balanced illumination and frontal highlights. For a wide scene, you may have to pull the sidelights back to be more positioned like the two-point rig.

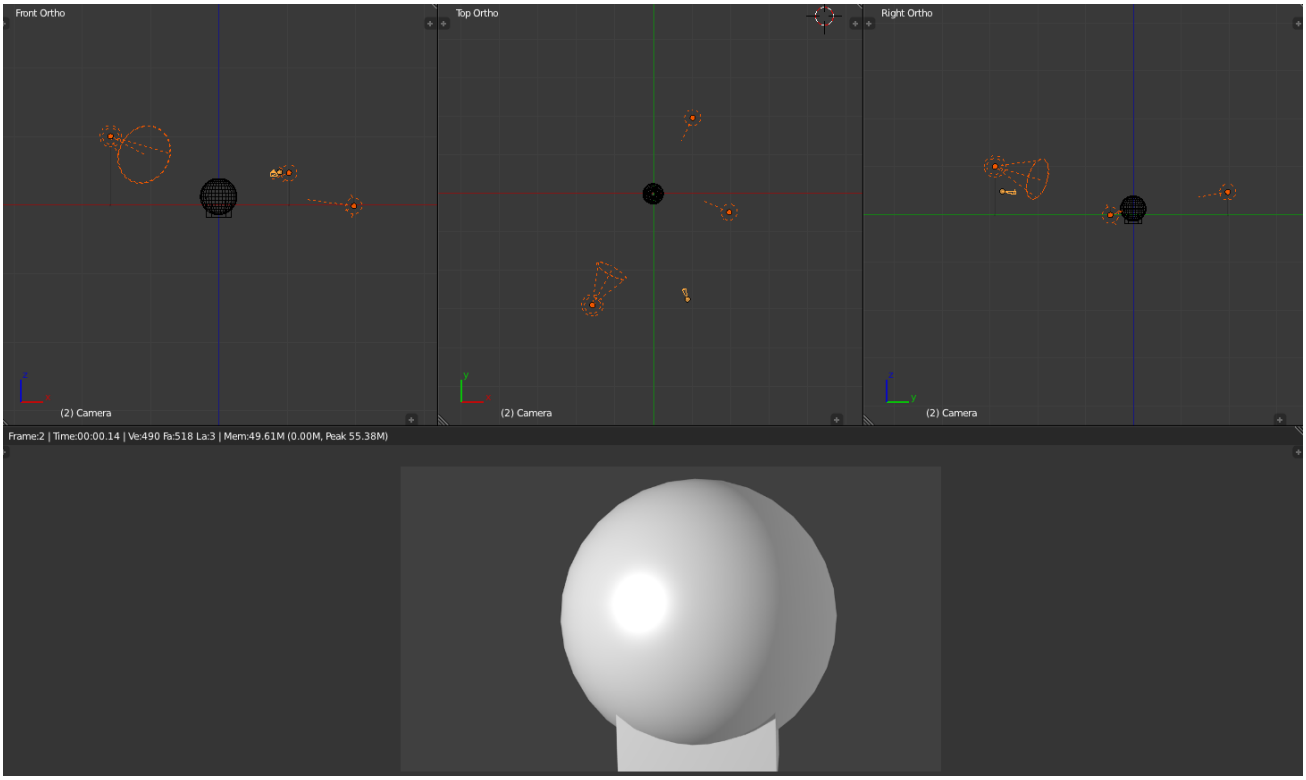


Fig. 2.1843: Standard three-point light rig.

Standard Rig

Without a curtain in back of your main subject, you have depth to work with. The left fill light has been moved behind the subject (so it is now called a backlight) and is just off-camera, while the right side fill light remains the same. The keylight gives you specular reflection so you can play with specularity and hardness in your object's material settings. The key light gives that "in-the-spotlight" feel, highlighting the subject, while the backlight gives a crisp edge to the subject against the background. This helps them stand out.

In this rig, the key light is a fairly bright spot light. Use a slighter tinge of yellow because the light is so bright; it is the only light for that side. The other sidelight has been moved in back and raised to eye (camera) level. You need to cut the energy of the backlight in half, or when it is added to the remaining sidelight, it will light up the side too much and call too much attention to itself. You can vary the angle and height of the backlight to mimic a sun lighting up the objects.

Use this rig in normal 3D animations to light the main actor. Use this rig especially if you have transparent objects (like glass) so that there is plenty of light to shine through them to the camera. The tricky part here is balancing the intensities of the lights so that no one light competes with or overpowers the others, while making sure all three work together as a team.

Four-point Rig

The four-point lighting rig provides a better simulation of outside lighting, by adding a *Sun* lamp 30 Blender Units above, 10 to the side, and 15 BU

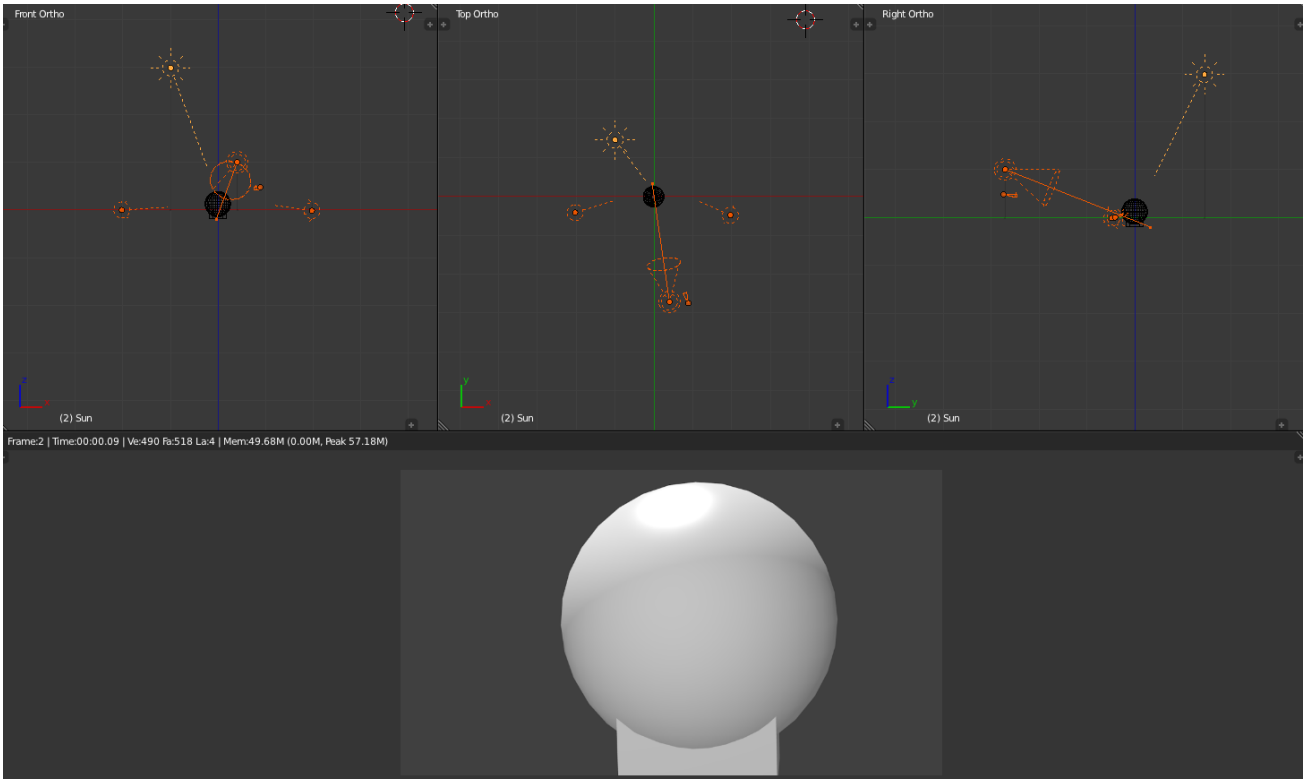


Fig. 2.1844: Four-point light rig.

behind the subject. This sunlight provides backlighting and fills the top of the subject; even producing an intentional glare on the top of their head, telling you there is a sun up there. Notice it is colored yellow, which balances out the blue sidelights.

Changing the key light to a *Spot*, select *Inverse Square*, disable *Specular* and pure white light combines with and softens the top sun flare while illuminating the face, resulting in a bright sunshine effect. Two lights above means sharper shadows as well, so you might want to adjust the side fill lights. In this picture, they are still *Hemi*, disable *Specular*.

Use this rig when the camera will be filming from behind the characters, looking over their shoulder or whatnot, because the sun provides the backlight there. Also use this rig when you have transparent objects, so there is light to come through the objects to the camera.

Another spot for the fill light is shining up onto the main actor's face, illuminating the underside of his chin and neck. This gets rid of a sometimes ugly shadow under the chin, which if not corrected, can make the actor look fat or like they have a double chin; otherwise distracting. It evens out the lighting of the face.

Troubleshooting

If you run into a problem with your render, where there are really bright areas, or really dark ones, or strange shadows, or lines on your objects, here are some good steps to debugging what is wring:

1. First, try deactivating all materials (create a default, gray one, and enter its name in the *Mat* field, *Layer* panel, the *Render Layer* tab to get back all your normal materials, just erase this text field!). See if you get those prob-

lems with just grayness objects. If you do not have the problem anymore, that should tell you that you have got a materials-interacting-with-light problem. Check the material settings, especially ambient, reflection and all those little buttons and sliders in the *Material* tab. You can set some lights to affect only certain materials, so if there is an issue with only a few objects being really bright, start with those.

2. Then start "killing" lights (e.g. moving them to an unused layer); regress all the way back to one light, make sure it is smooth, then add them in one by one. As they add together, reduce power in the tested ones so they merge cleanly, or consider not adding it at all, or, especially, reduce the energy of the lamp you just introduced.
3. You can also set lights to only light objects on a layer, so again, if some of the gray spheres have weirdness, check for that as well. Again, you may have done some of this accidentally, so sometimes deleting the light and re-adding it with defaults helps you reset to a known-good situation.
4. Negative lights can be very tricky, and make your model blotchy, so pay special attention to your use of those special lights. Shadow-only lights can throw off the look of the scene as well. Overly textured lights can make your scene have random weird colors. Do not go too far off a slight tinge of blue or yellow or shades of white, or your material may show blue in the *Material* tab but render green, and you will be very confused.
5. Look at your environment settings *World* tab: *Horizon*, *Zenith*, and *Environment Lighting*.

Camera

Introduction

A *Camera* is an object that provides a means of rendering images from Blender. It defines which portion of a scene is visible in the rendered image. By default a scene contains one camera. However, a scene can contain more than one camera, but only one of them will be used at a time. So you will only need to add a new camera if you are making cuts between them. See *Animating Cameras*.

Changing the Active Camera

Reference

Mode: Object Mode
Hotkey: `Ctrl-Numpad0`

The *active* camera is the camera that is currently being used for rendering and camera view `Numpad0`.

Select the camera you would like to make active and press `Ctrl-Numpad0` (by doing so, you also switch the view to camera view). In order to render, each scene **must** have an active camera.

The active camera can also be set in the *Scene* tab of the *Properties Editor*.

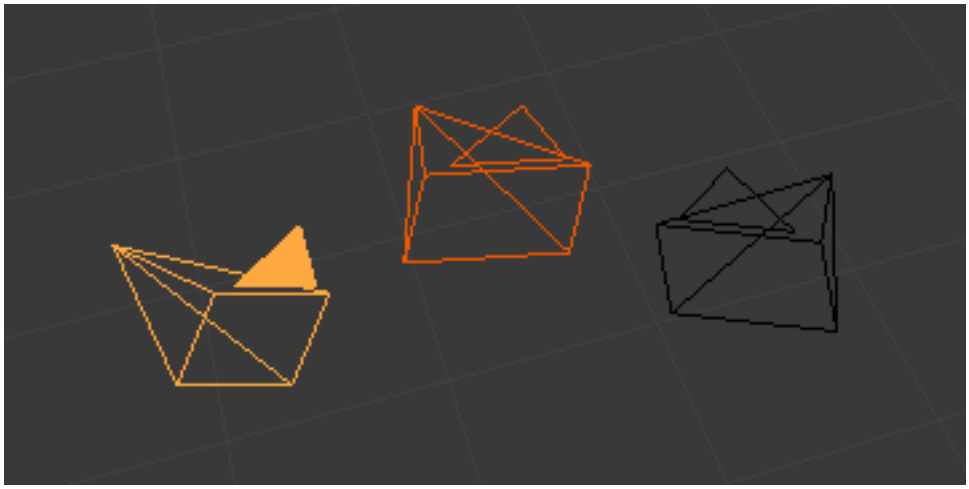


Fig. 2.1845: Active camera (left one).

The camera with the solid triangle on top is the active camera.

Warning: The active camera, as well as the layers, can be specific to a given view, or global (locked) to the whole scene. See *Local Camera*.

Render Border

Reference

Mode: All modes
Menu: *View* → *Render Border*
Hotkey: `Ctrl-B`

While in camera view, you can define a subregion to render by drawing out a rectangle within the camera's frame. Your renders will now be limited to the part of scene visible within the render border. This can be very useful for reducing render times for quick previews on an area of interest.

The border can be disabled by disabling the *Border* option in the *Dimensions* panel in the *Render* tab or by activating the option again.

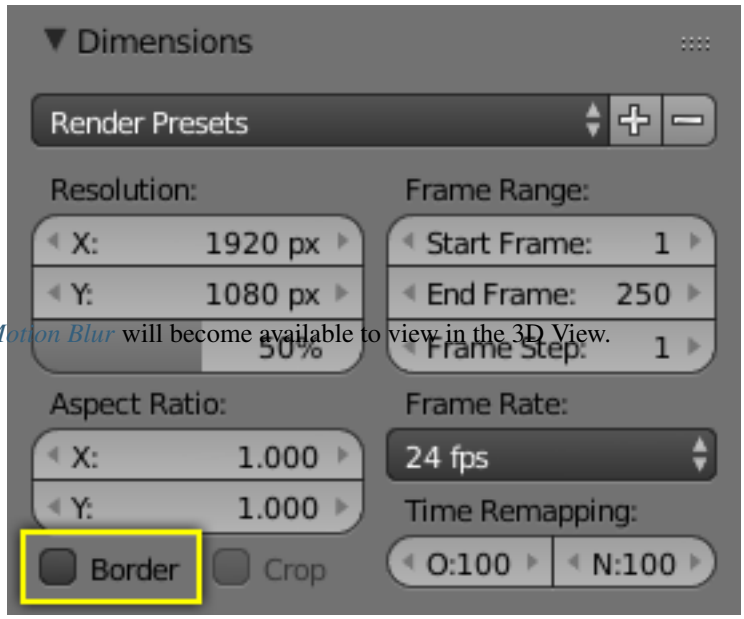
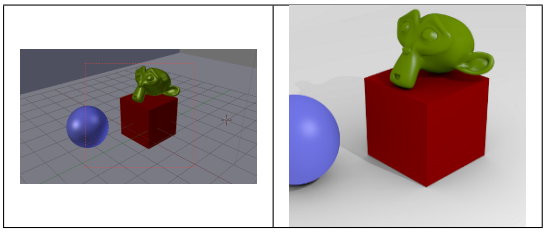


Fig. 2.1846: Render Border toggle.

Note: When Render Border is activated, *Sampled Motion Blur* will become available to view in the 3D View.

Table 2.98: Render border and associated render.



Object Data

Reference

Mode: Object Mode
Editor: *Properties* → *Camera*

Cameras are invisible in renders, so they do not have any material or texture settings. However, they do have *Object* and *Editing* setting panels available which are displayed when a camera is the selected (active!) object.

Lens

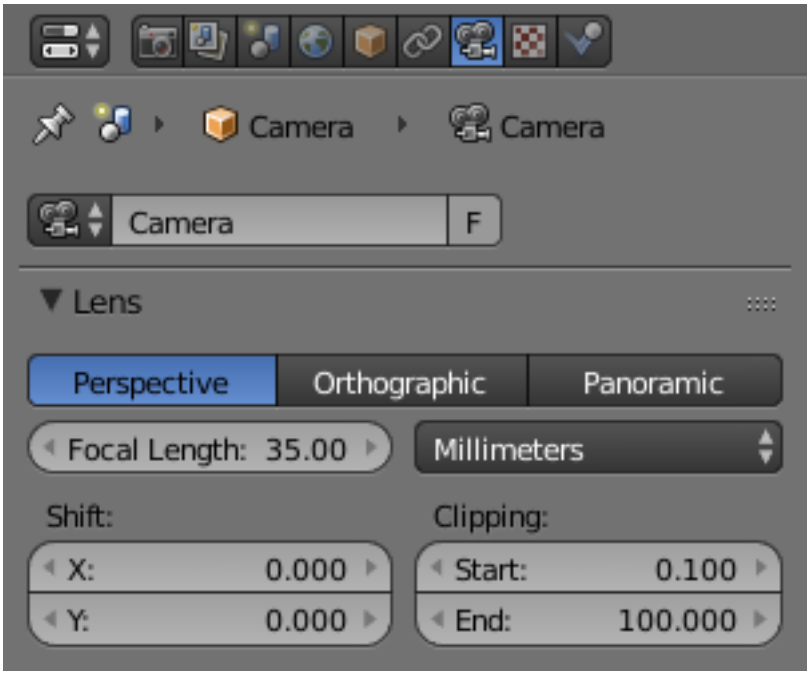


Fig. 2.1847: Camera Lens panel.

The camera lens options control the way 3D objects are represented in a 2D image.

Lens Type

There are three different lens types:

- *Perspective*
- *Orthographic*
- *Panoramic*

Perspective

This matches how you view things in the real-world. Objects in the distance will appear smaller than objects in the foreground, and parallel lines (such as the rails on a railroad) will appear to converge as they get farther away.

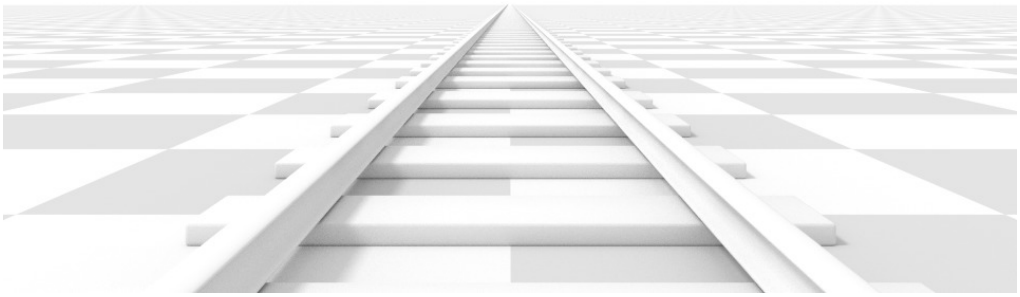


Fig. 2.1848: Render of a train track scene with a *Perspective* camera.

Settings which adjust this projection include:

- Focal length
- *Shift*
- *Sensor size*

Focal length The *focal length* setting controls the amount of zoom, i.e. the amount of the scene which is visible all at once. Longer focal lengths result in a smaller FOV (Field of View) (more zoom), while short focal lengths allow you to see more of the scene at once (larger FOV, less zoom).

Lens Unit The focal length can be set either in terms of millimeters or the actual *Field of View* as an angle.

Orthographic

With *Orthographic* perspective objects always appear at their actual size, regardless of distance. This means that parallel lines appear parallel, and do not converge like they do with *Perspective*.

Orthographic Scale This controls the apparent size of objects in the camera.

Note that this is effectively the only setting which applies to orthographic perspective. Since parallel lines do not converge in orthographic mode (no vanishing points), the lens shift settings are equivalent to translating the camera in the 3D View.

Fig. 2.1849: Render of the same scene as above, but with a focal length of 210mm instead of 35mm.

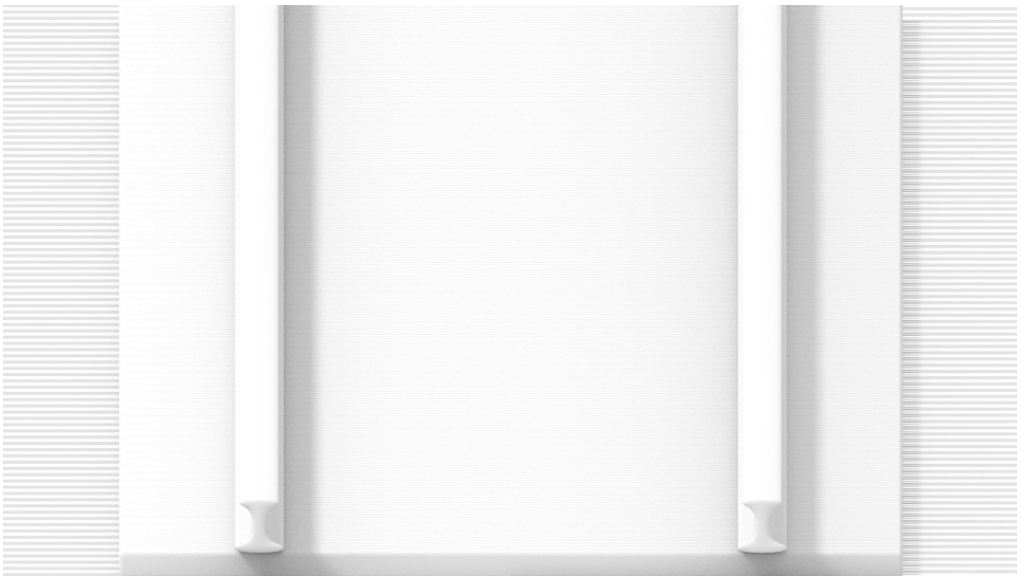


Fig. 2.1850: Render from the same camera angle as the previous examples, but with orthographic perspective.

Panoramic

Panoramic cameras are only supported in the Cycles render engine. See *the Cycles documentation*.

Shift

The *Shift* setting allows for the adjustment of *vanishing points*. *Vanishing points* refer to the positions to which parallel lines converge. In this example, the most obvious vanishing point is at the end of the railroad.

To see how this works, take the following examples:

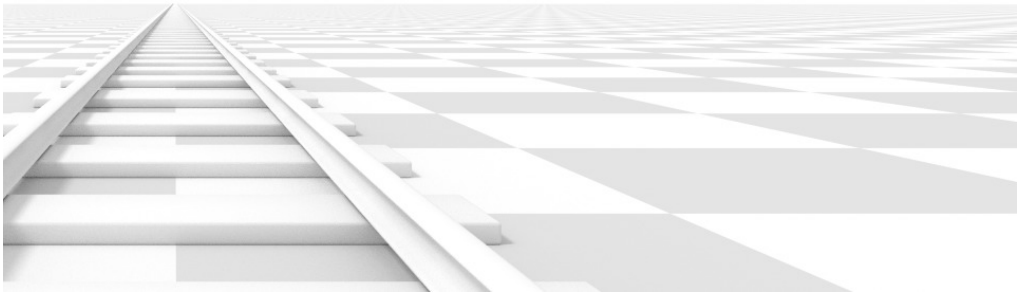


Fig. 2.1851: Render of a train track scene with a horizontal lens shift of 0.330.

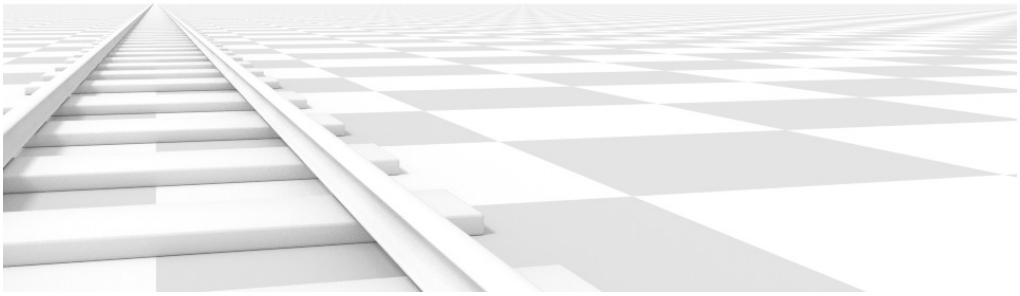


Fig. 2.1852: Render of a train track scene with a rotation of the camera object instead of a lens shift.

Notice how the horizontal lines remain perfectly horizontal when using the lens shift, but do get skewed when rotating the camera object.

Using lens shift is equivalent to rendering an image with a larger FOV and cropping it off-center.

Clipping

Set the clipping limits with the *Start* and *End* values.

Only objects within the limits are rendered.

For OpenGL display, setting clipping distances to limited values is important to ensure sufficient rasterization precision. Ray tracing renders do not suffer from this issue so much, and as such more extreme values can safely be set.

When *Limits* in the *Display* panel is enabled, the clip bounds will be visible as two yellow connected dots on the camera line of sight.

Note: The *3D View* editor contains settings similar to the camera, see the *3D View options page* for more details.

Camera

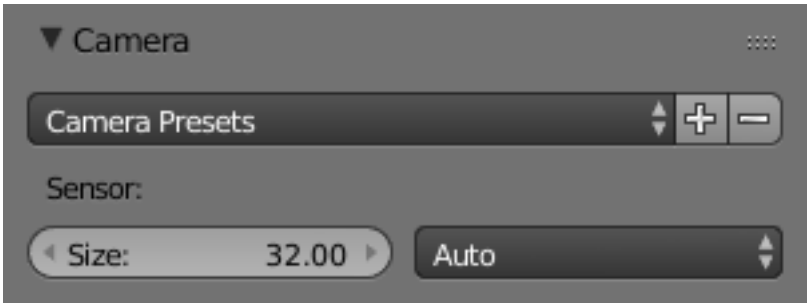


Fig. 2.1853: Camera Presets panel.

Camera Presets TODO.

Sensor size This setting is an alternative way to control the focal-length, it is useful to match the camera in Blender to a physical camera & lens combination, e.g. for *motion tracking*.

Depth of Field

Fig. 2.1854: Camera Depth of Field Panel.

Real world cameras transmit light through a lens that bends and focuses it onto the sensor. Because of this, objects that are a certain distance away are in focus, but objects in front and behind that are blurred.

The area in focus is called the *focal point* and can be set using either an exact value, or by using the distance between the camera and a chosen object:

Focus Object Choose an object which will determine the focal point. Linking an object will deactivate the distance parameter. Typically this is used to give precise control over the position of the focal point, and also allows it to be animated or constrained to another object.

Distance Sets the distance to the focal point when no *Focus Object* is specified. If *Limits* are enabled, a yellow cross is shown on the camera line of sight at this distance.

> **Hint:** Hover the mouse over the *Distance* property and press `E` to use a special *Depth Picker*. Then click on a point in the 3D View to sample the distance from that point to the camera.

High Quality In order for the viewport to offer an accurate representation of depth of field, like a render, you must enable High Quality. Without it, you may notice a difference in shading.

Viewport F-stop Controls the real-time focal blur effect used during sequencer or OpenGL rendering and, when enabled, camera views in the 3D View. The amount of blur depends on this setting, along with Focal Length and Sensor Size. Smaller Viewport F-stop values result in more blur.

Blades Add a number of polygonal *blades* to the blur effect, in order to achieve a a *bokeh effect* in the viewport. To enable this feature, the blades must be set to at least 3 (3 sides, triangle)

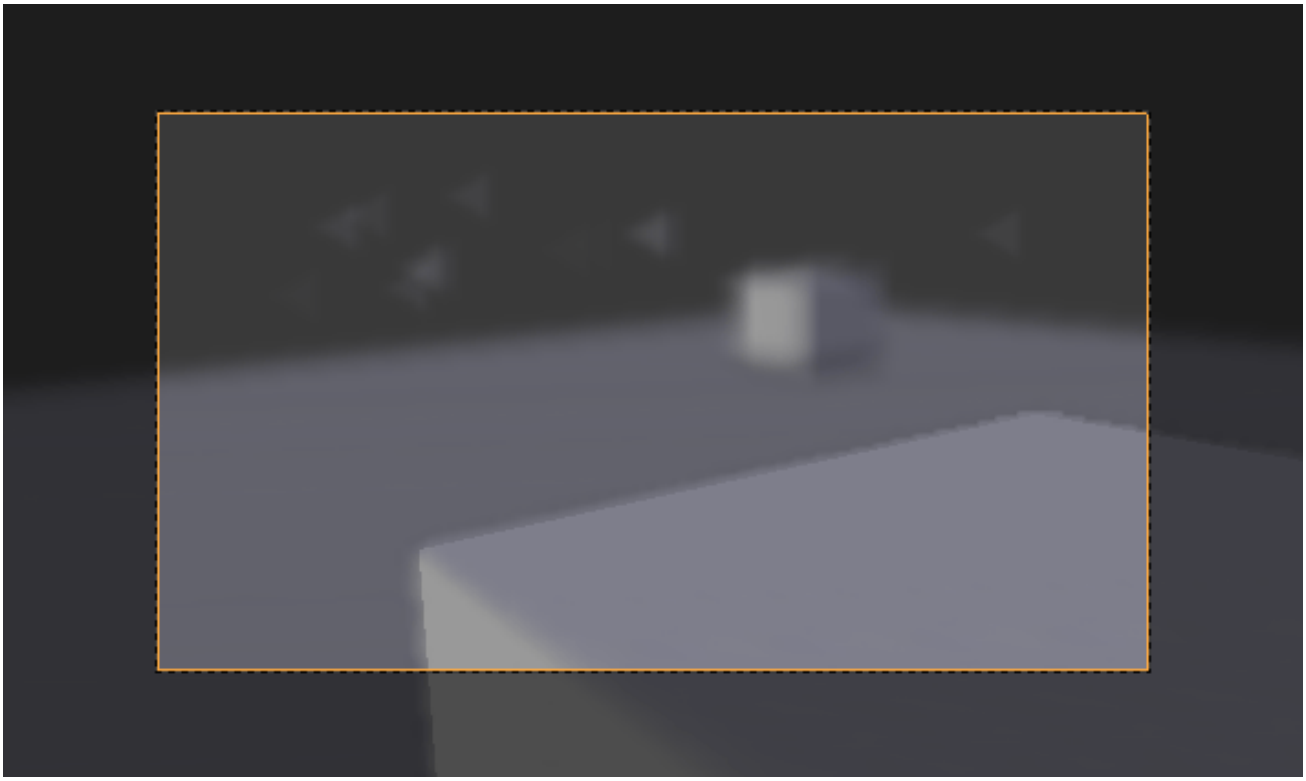


Fig. 2.1855: The viewport bokeh effect with the blades set to 3.

Display

Limits Shows a line which indicates *Start* and *End Clipping* values.

Mist Toggles viewing of the mist limits on and off. The limits are shown as two connected white dots on the camera line of sight. The mist limits and other options are set in the *World* panel, in the *Mist section*.

Sensor Displays a dotted frame in camera view.

Name Toggle name display on and off in camera view.

Size Size of the camera icon in the 3D View. This setting has no effect on the render output of a camera, and is only a cosmetic setting. The camera icon can also be scaled using the standard Scale `S` transform key.

Passepartout, Alpha This mode darkens the area outside of the camera's field of view, based on the *Alpha* setting.

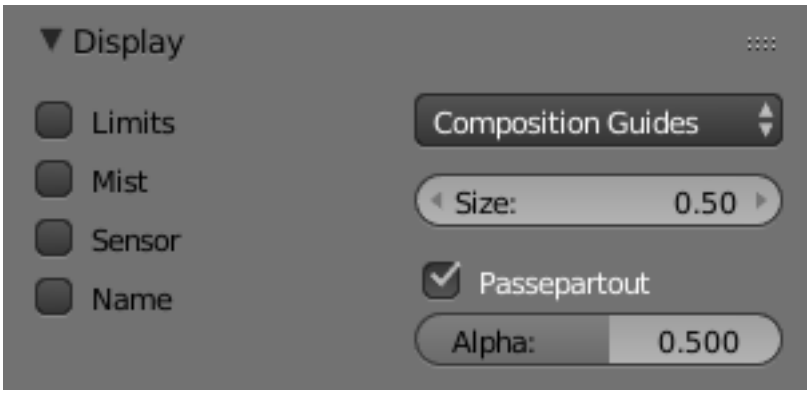


Fig. 2.1856: Camera Display Panel.

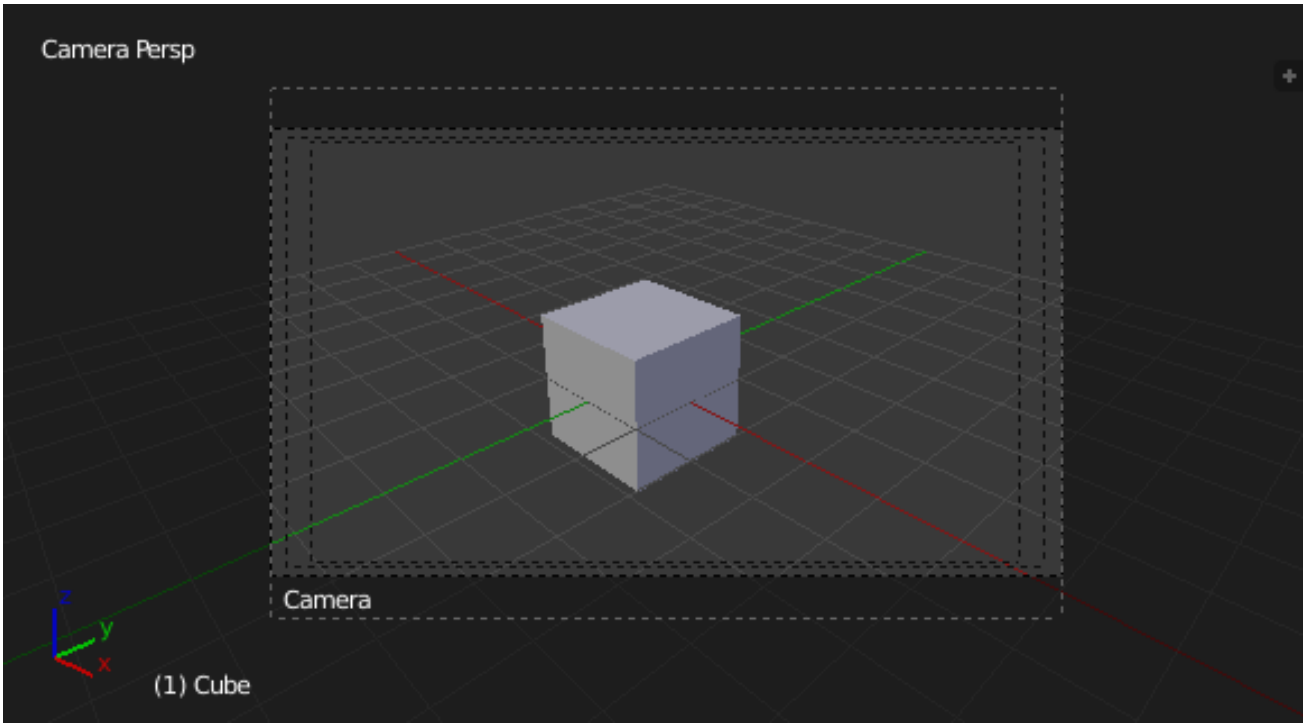


Fig. 2.1857: Camera view displaying safe areas, sensor and name.

Composition Guides

Composition Guides are available from the menu, which can help when framing a shot. There are eight types of guides available:

Center Adds lines dividing the frame in half vertically and horizontally.

Center Diagonal Adds lines connecting opposite corners.

Thirds Adds lines dividing the frame in thirds vertically and horizontally.

Golden Divides the width and height into Golden proportions (About 0.618 of the size from all sides of the frame).

Golden Triangle A Draws a diagonal line from the lower-left to upper-right corners, then adds perpendicular lines that pass through the top left and bottom right corners.

Golden Triangle B Same as A, but with the opposite corners.

Harmonious Triangle A Draws a diagonal line from the lower-left to upper-right corners, then lines from the top left and bottom right corners to 0.618 the lengths of the opposite side.

Harmonious Triangle B Same as A, but with the opposite corners.

Safe Areas

Safe areas are guides used to position elements to ensure that the most important parts of the content can be seen across all screens.

Different screens have varying amounts of *overscan*. (specially older TV sets). That means that not all content will be visible to all viewers, since parts of the image surrounding the edges are not shown. To work around this problem TV producers defined two areas where content is guaranteed to be shown: action safe and title safe.

Modern LCD/plasma screens with purely digital signals have no *overscan*, yet safe areas are still considered best practice and may be legally required for broadcast.

In Blender, safe areas can be set from the Camera and Sequencer views.

Fig. 2.1858: The Safe areas panel found in the camera properties, and the view mode of the sequencer.

Main Safe Areas

Title Safe Also known as *Graphics Safe*. Place all important information (graphics or text) inside this area to ensure it can be seen by the majority of viewers.

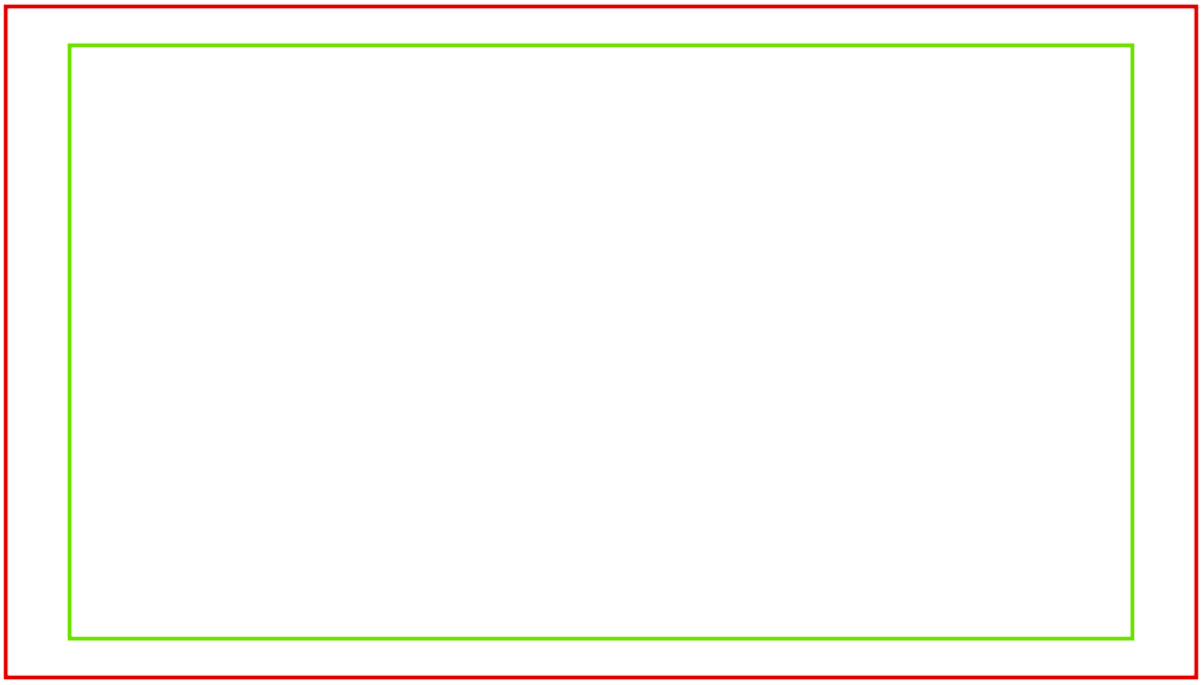


Fig. 2.1859: Red line: Action safe. Green line: Title safe.

Action Safe Make sure any significant action or characters in the shot are inside this area. This zone also doubles as a sort of "margin" for the screen which can be used to keep elements from piling up against the edges.

Tip: Legal Standards

Each country sets a legal standard for broadcasting. These include, among other things, specific values for safe areas. Blender defaults for safe areas follow the EBU (European Union) standard. Make sure you are using the correct values when working for broadcast to avoid any trouble.

Center-Cuts

Center-cuts are a second set of safe areas to ensure content is seen correctly on screens with a different aspect ratio. Old TV sets receiving `16:9` or `21:9` video will cut off the sides. Position content inside the center-cut areas to make sure the most important elements of your composition can still be visible in these screens.

Blender defaults show a `4:3` (square) ratio inside `16:9` (wide-screen).

World

Introduction

The world buttons let you set up the shading of your scene in general. It can provide ambient color, and special effects such as mist, but a very common use of a *World* is to shade a background color.

These are accessible via the *World* tab.

You have:

Background The color and texture of the world background, with special settings for mapping coordinates.

Fig. 2.1860: Cyan line: action center safe. Blue line: title center safe.

Mist Add a mist to your scene to enhance the feeling of depth.

While these world settings offers a simple way of adding effects to a scene, *compositing nodes* are often preferred, though more complex to master, for the additional control and options they offer. For example, filtering the Z value (distance from camera) or normals (direction of surfaces) through compositing nodes can further increase the depth and spacial clarity of a scene.

World panel

World The World *Data-Block Menu*.

Texture Count Shows the count of textures in the world texture stack.

Note: Background Image in 3D

To use an image as a background image in your 3D View, for example as a reference when doing a model, see *using a Background Image*

Preview

Shows a view inside a sphere, on which the background textures are mapped.

World (Background)

Sky

How colors below are interpreted depends on which kind of *Sky* is chosen.

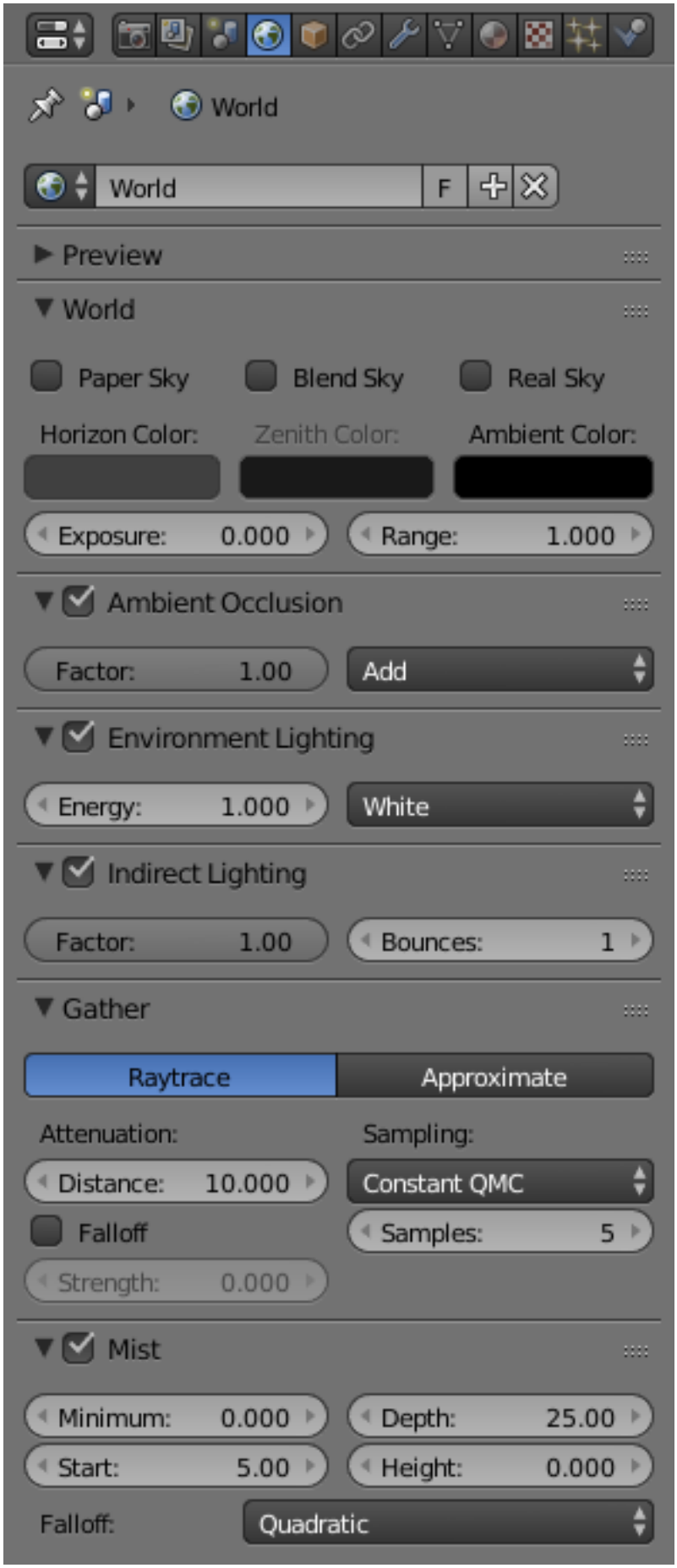


Fig. 2.1861: World tab.

Fig. 2.1862: World panel.

None Enabled If none of these three buttons is checked, your background will just be plain flat color (using the horizon one).

Paper Sky If this option is added, the gradient keeps its characteristics, but it is clipped in the image (it stays on a horizontal plane (parallel to XY plane): what ever the angle of the camera may be, the horizon is always at the middle of the image).

Blend Sky The background color is blended from horizon to zenith. If only this button is pressed, the gradient runs from the bottom to the top of the rendered image regardless of the camera orientation.

Real Sky If this option is added, the gradient produced has two transitions, from nadir (same color as zenith) to horizon to zenith; the blending is also dependent on the camera orientation, which makes it more realistic. The horizon color is exactly at the horizon (on the XY plane), and the zenith color is used for points vertically above and below the camera.

See also:

When using a *Sun Lamp* options for *Sky & Atmosphere* are available in the *Lamp* tab.

Colors

Horizon Color The RGB color at the horizon.

Zenith Color The RGB color at the zenith (overhead).

Ambient Color *Ambient Light*. See also *Indirect Lighting*.

Exposure and Range

See *Exposure and Range*.

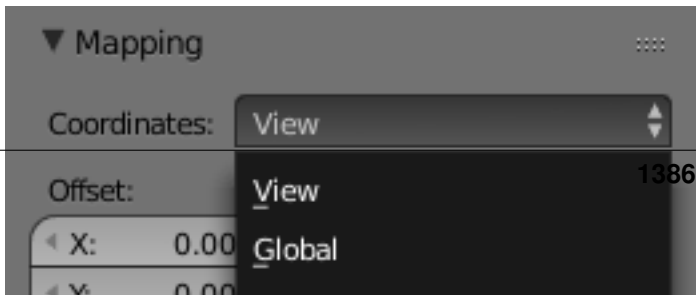
Textures

Mapping

Instead of a color, or blend of two colors, Blender can use an 2D image which it maps to a very large Box or sphere which encompasses the entire scene, or which it maps to a virtual space around the scene.

The World textures are accessible in the texture menu (just select *World* first, then *Texture*. They are used much like the Materials textures, except for a couple of differences. The textures can be mapped according to:

Texture Coordinates



View The default orientation, aligned with the co-ordinates of the final render.

Global Uses global coordinates.

Angular Map Used to wrap a standard hemisphere angular map around the scene in a dome. This can be used for image based lighting with *Ambient Occlusion* set to sky color. You will generally need a high dynamic range image (HDRI) angular map. (It will look like a weird spherical image).

Sphere Sphere mapping, similar to that of materials.

Tube Wrap the rectangular texture around in a cylinder, similar to that of materials.

Object Position the texture relative to a specified object's local texture space.

Influence

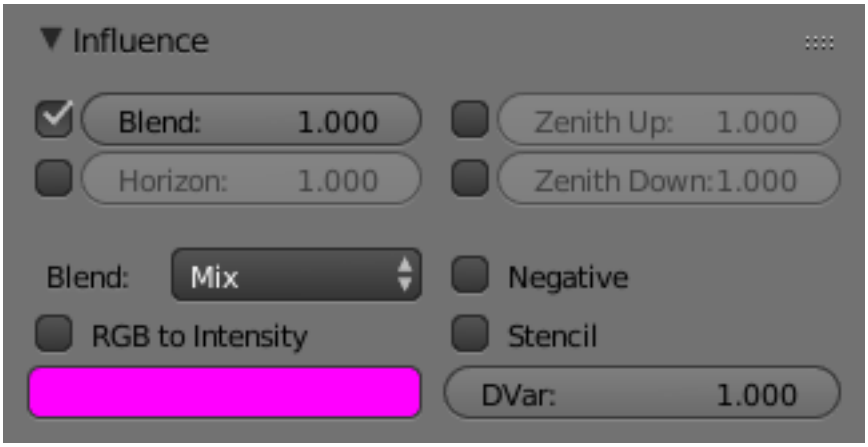


Fig. 2.1864: Texture Influence panel.

The texture affects color only, but in four different ways:

Blend Makes the Horizon color appear where the texture is non-zero.

Horizon Affect the color of the horizon.

Zenith Up Affect the zenith color overhead.

Zenith Down Affect the zenith color underneath.

If you are disappointed that your camera appears to carry the texture with it rather than rotate through the texture, you should check the Real Sky checkbox in the World panel.

Exposure and Range

Reference

Mode: All modes
Panel: *Render Layer* → *Color management*

Exposure and *Range* are similar to the "Color Curves" tool in Gimp or Photoshop.

These controls affect the rendered image, and the results are baked into the render. For information on achieving similar affects with render controls, see *Color Management and Exposure*.

Previously Blender clipped color directly with 1.0 (or 255) when it exceeded the possible RGB space. This caused ugly banding and overblown highlights when light overflowed Fig. *Utah Teapot.*.

Using an exponential correction formula, this now can be nicely corrected.

Options

Fig. 2.1865: Exposure and Range sliders.

Exposure The exponential curvature, with (0.0 to 1.0) (linear to curved).

Range The range of input colors that are mapped to visible colors (0.0 to 1.0).

So without *Exposure* we will get a linear correction of all color values:

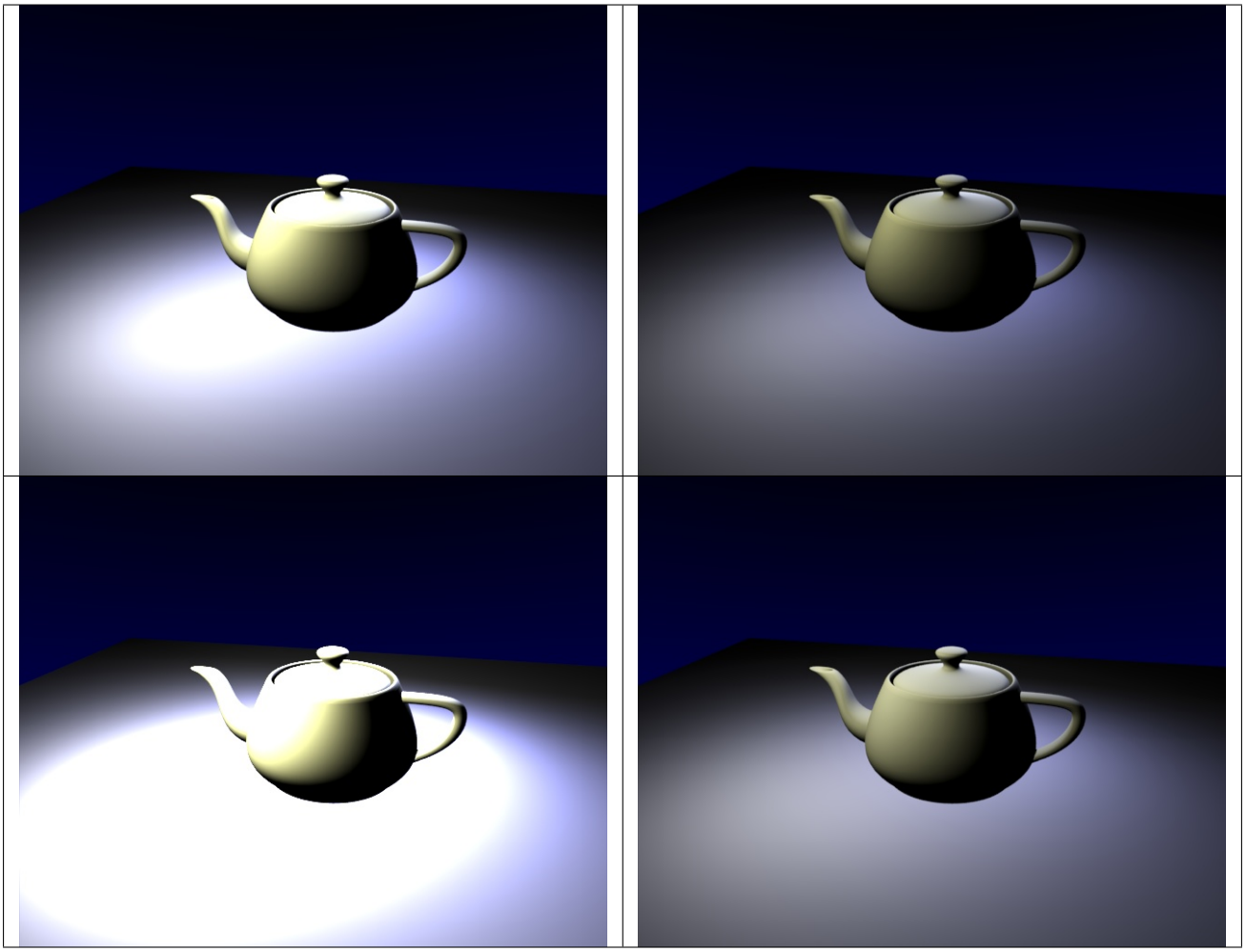
Range > 1.0 The picture will become darker; with *Range* = 2.0, a color value of 1.0 (the brightest by default) will be clipped to 0.5 (half bright) (*Range*: 2.0).

Range < 1.0 The picture will become brighter; with *Range* = 0.5, a color value of 0.5 (half bright by default) will be clipped to 1.0 (the brightest) (*Range*: 0.5).

Examples

With a linear correction every color value will get changed, which is probably not what we want. *Exposure* brightens the darker pixels, so that the darker parts of the image will not be changed at all (*Range* : 2.0, *Exposure* : 0.3).

Table 2.99: Range: 2.0, Exposure: 0.3.



Hint: Try to find the best *Range* value, so that overexposed parts are barely not too bright. Now turn up the *Exposure* value until the overall brightness of the image is satisfying. This is especially useful with area lamps.

Environment Lighting

Environment light provides light coming from all directions.

Light is calculated with a ray-traced method which is the same as that used by Ambient Occlusion. The difference is that Environment lighting takes into account the "ambient" parameter of the material shading settings, which indicates the amount of ambient light/color that that material receives.

Also, you can choose the environment color source (white, sky color, sky texture) and the light energy.

Energy Defines the strength of environment light.

Environment Color Defines where the color of the environment light comes from.

Using both settings simultaneously produces better global lighting.

It is good for mimicking the sky in outdoor lighting. Environment lighting can be fairly noisy at times.

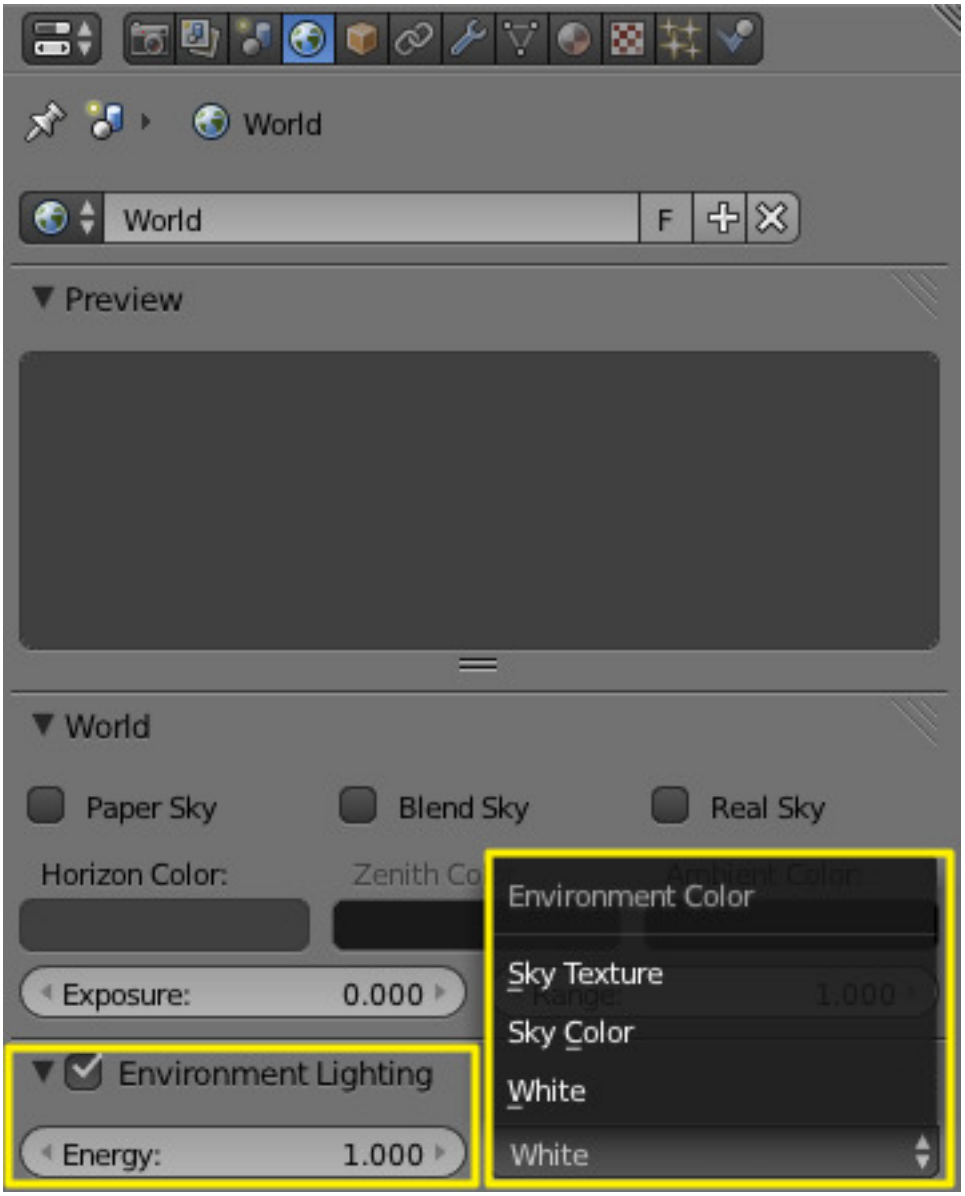
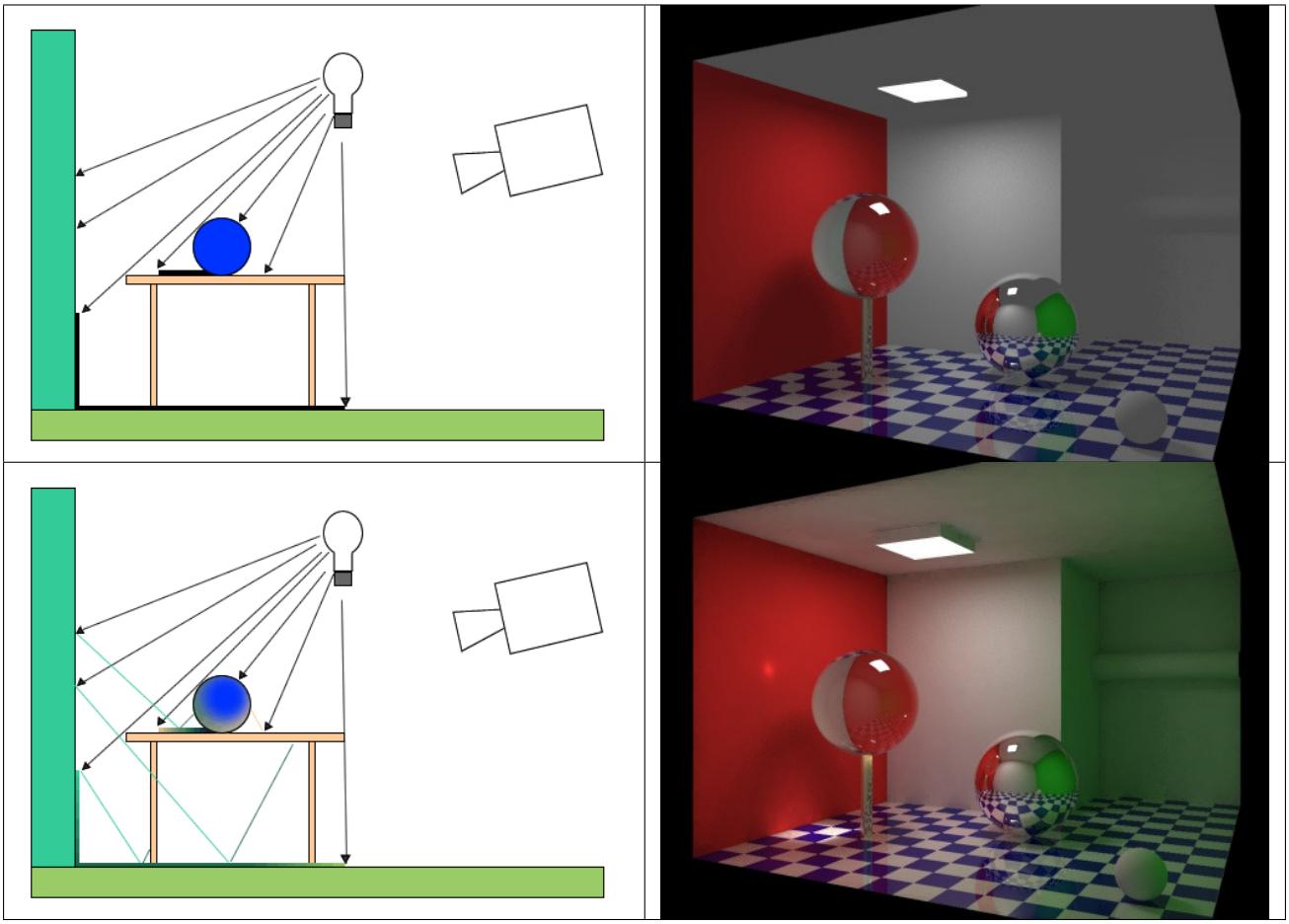


Fig. 2.1870: Environment Lighting panel.

Indirect Lighting

Indirect Lighting adds indirect light bouncing of surrounding objects. It models the light that is reflected from other surfaces to the current surface. Is more comprehensive, more physically correct, and produces more realistic images. It is also more computationally expensive. Take a look at the following examples of a scene lit with Direct Lighting and both Direct and Indirect Lighting:

Table 2.100: Direct and Indirect Lighting Render.



Indirect Lighting only works with Approximate gather method.

Options

The *Indirect Lighting* panel contains two options:

Factor Defines how much surrounding objects contribute to light.

Bounces Number of indirect diffuse light bounces.

The *Gather* panel contains settings for the indirect lighting quality. Note that these settings also apply to Environment Lighting and Ambient Occlusion.

Approximate

The *Approximate* method gives a much smoother result for the same amount of render time, but as its name states, it is only an approximation of the *Raytrace* method, which implies it might produce some artifacts and it cannot use the sky's texture as the base color.

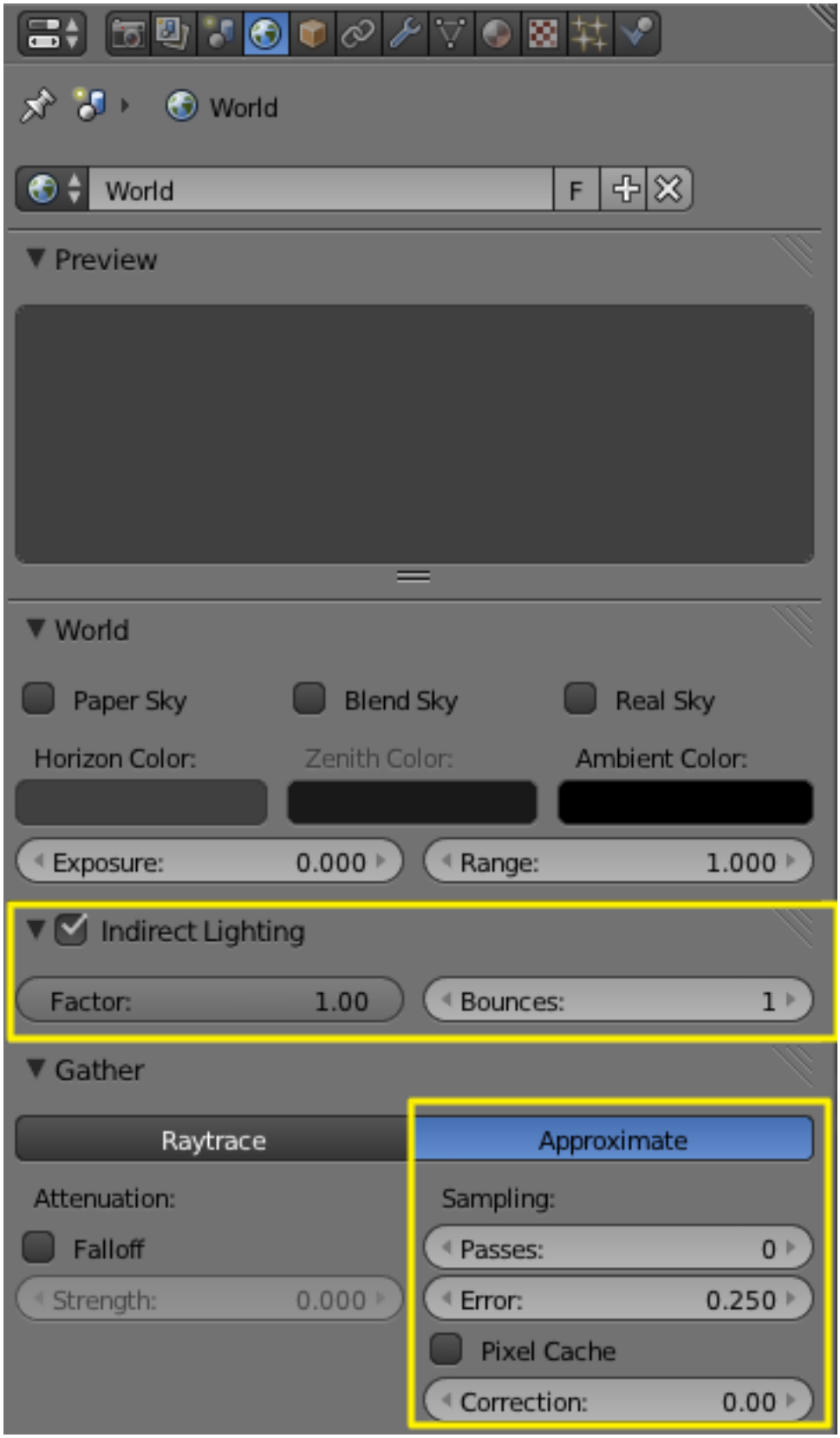


Fig. 2.1875: Indirect Lighting parameters.

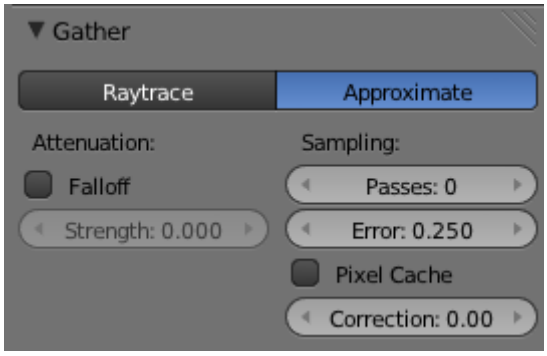


Fig. 2.1876: The Indirect Lighting panel, Approximate method.

This method seems to tend to "over-occlude" the results. You have two complementary options to reduce this problem:

Passes Set the number of pre-processing passes, between (0 to 10) passes. Keeping the pre-processing passes high will increase render time, but will also clear some artifacts and over-occlusions.

Error This is the tolerance factor for approximation error (i.e. the max allowed difference between approximated result and fully computed result). The lower, the slower the render, but the more accurate the results... Ranges between (0.0 to 10.0), defaults to 0.250.

Pixel Cache When enabled, it will keep values of computed pixels to interpolate it with its neighbors. This further speeds up the render, generally without visible loss in quality...

Correction A correction factor to reduce over-occlusion. Ranges between (0.0 to 1.0) correction.

Ambient Occlusion

Ambient Occlusion is a sophisticated ray-tracing calculation which simulates soft global illumination shadows by faking darkness perceived in corners and at mesh intersections, creases, and cracks, where ambient light is occluded, or blocked.

There is no such thing as AO in real life; AO is a specific not-physically-accurate (but generally nice-looking) rendering trick. It basically samples a hemisphere around each point on the face, sees what proportion of that hemisphere is occluded by other geometry, and shades the pixel accordingly.

It is got nothing to do with light at all; it is purely a rendering trick that tends to look nice because generally in real life surfaces that are close together (like small cracks) will be darker than surfaces that do not have anything in front of them, because of shadows, dirt, etc.

The AO process, though, approximates this result; it is not simulating light bouncing around or going through things. That is why AO still works when you do not have any lights in the scene, and it is why just switching on AO alone is a very bad way of "lighting" a scene.

You must have ray tracing enabled as a *Render* panel option in the *Shading* section for this to work.

You must have an ambient light color set as you desire. By default, the ambient light color (world) is black, simulating midnight in the basement during a power outage. Applying that color as ambient will actually darken all colors. A good outdoor mid-day color is RGB(0.9, 0.9, 0.8) which is a whitish yellow sunny kind of color on a bright-but-not-harshly-bright day.

Options

Factor The strength of the AO effect, a multiplier for addition.

Ambient Occlusion is composited during the render. Two blending modes are available:

Add The pixel receives light according to the number of non-obstructed rays. The scene is lighter. This simulates global illumination.

Multiply Ambient occlusion is multiplied over the shading, making things darker.

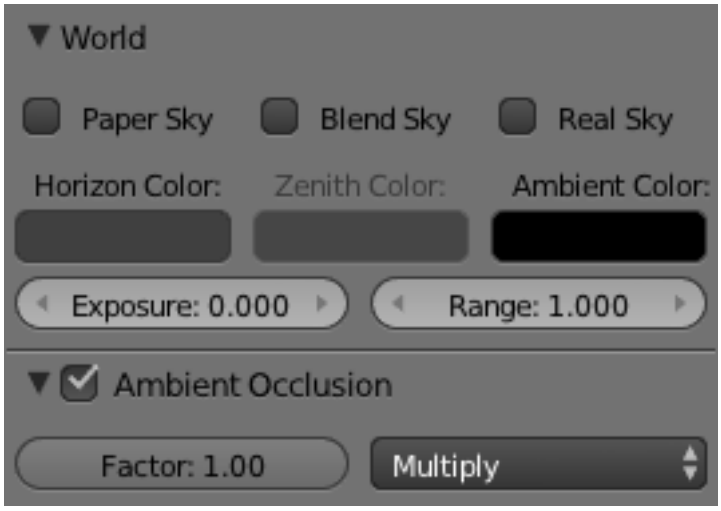


Fig. 2.1877: The World panel with ambient color sliders highlighted.

Note: If *Multiply* is chosen, there must be other light sources; otherwise the scene will be pitch black. In the other two cases the scene is lit even if no explicit light is present, just from the AO effect. Although many people like to use AO alone as a quick shortcut to light a scene, the results it gives will be muted and flat, like an overcast day. In most cases, it is best to light a scene properly with Blender's standard lamps, then use AO on top of that, set to *Multiply*, for the additional details and contact shadows.

The *Gather* panel contains settings for the ambient occlusion quality. Note that these settings also apply to Environment Lighting and Indirect Lighting.

Ambient occlusion has two main methods of calculation: *Raytrace* and *Approximate*.

Gather

Raytrace

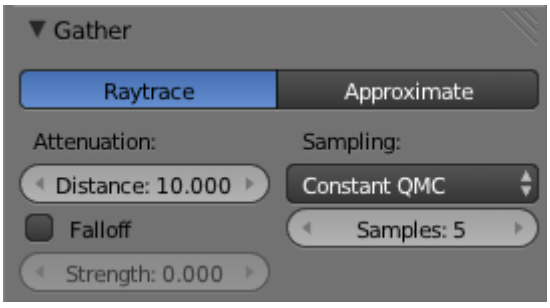


Fig. 2.1878: The Amb Occ panel, Raytrace method.

The *Raytrace* method gives the more accurate, but also the more noisy results. You can get a nearly noiseless image, but at the cost of render time... It is the only option if you want to use the colors of your sky's texture.

Attenuation Length of rays defines how far away other faces may be and still have an occlusion effect. The longer this distance, the greater impact that far-away geometry will have on the occlusion effect. A high *Distance* value also means that the renderer has to search a greater area for geometry that occludes, so render time can be optimized by making this distance as short as possible for the visual effect that you want.

Sampling

Sampling Meathod

Constant QMC The base Quasi-Monte Carlo, gives evenly and randomly distributed rays.

Adaptive QMC An improved method of QMC, that tries to determine when the sample rate can be lowered or the sample skipped, based on its two settings:

Threshold The limit below which the sample is considered fully occluded ("black") or un-occluded ("white"), and skipped.

Adapt to Speed A factor to reduce AO sampling on fast-moving pixels. As it uses the *Vector* render pass, that must also be enabled (see *render passes page*).

Note: About QMC

See also the *raytraced shadows page* for more info about the Quasi-Monte Carlo sampling method.

Constant Jittered The historical sample method, more prone to "bias" artifacts...

Bias The angle (in radians) the hemisphere will be made narrower (i.e. the hemisphere will no longer be a real hemisphere: its section will no longer be a semicircle, but an arc of a circle of: pi - *bias* radians).

The bias setting allows you to control how smooth "smooth" faces will appear in AO rendering. Since AO occurs on the original faceted mesh, it is possible that the AO light makes faces visible even on objects with "smooth" on. This is due to the way AO rays are shot, and can be controlled with the *Bias* slider. Note that while it might even happen with QMC sampling methods, it is much more visible with the *Constant Jittered* one and anyway, you have no *Bias* option for QMC.

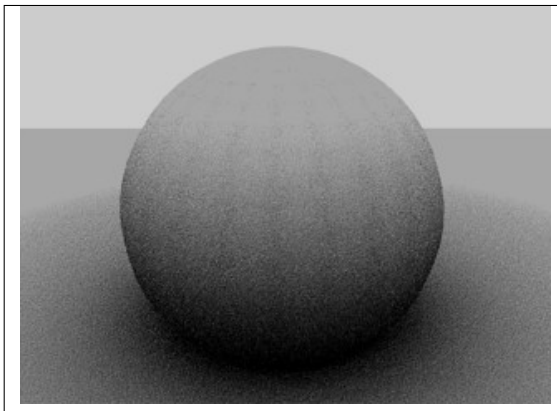


Fig. 2.1879: 24×24 UV Sphere with Bias: 0.05 (default). Note the facets on the sphere's surface even though it is set to smooth.

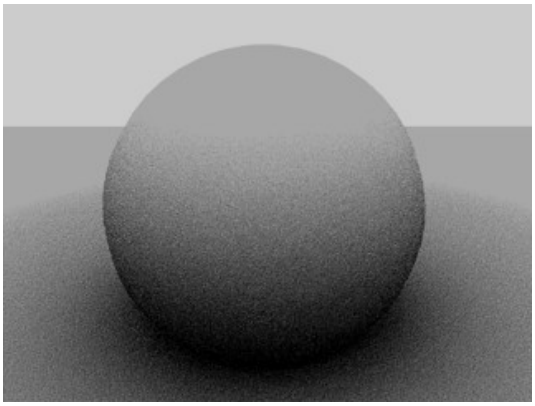


Fig. 2.1880: Raising the Bias to 0.15 removes the faceted artifacts.

Samples The number of rays used to detect if an object is occluded. Higher numbers of samples give smoother and more accurate results, at the expense of slower render times. The default value of 5 is usually good for previews. The actual number of rays shot out is the square of this number (i.e. *Samples* at 5 means 25 rays). Rays are shot at the hemisphere according to a random pattern (determined by the sample methods described above); this causes differences in the occlusion pattern of neighboring pixels unless the number of shot rays is big enough to produce good statistical data.

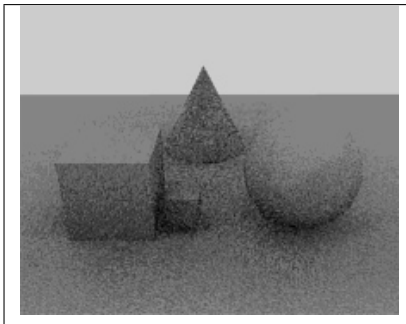


Fig. 2.1881: Ambient Occlusion with 3 Samples.

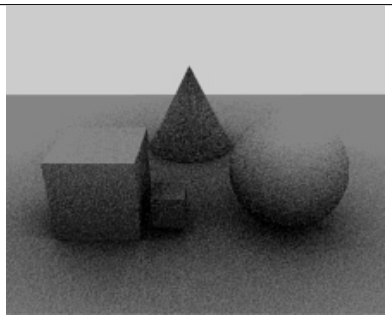


Fig. 2.1882: Ambient Occlusion with 6 Samples.

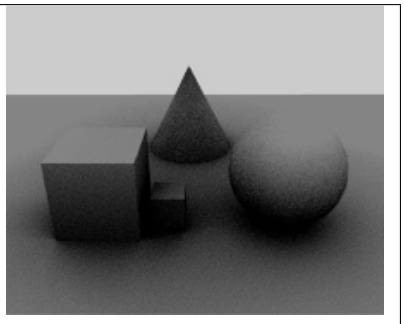


Fig. 2.1883: Ambient Occlusion with 12 Samples.

Approximate

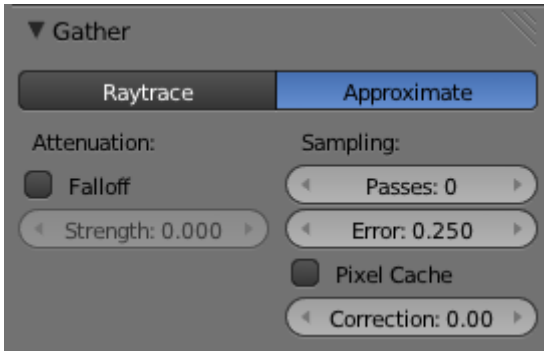


Fig. 2.1884: The Amb Occ panel, Approximate method.

The *Approximate* method gives a much smoother result for the same amount of render time, but as its name states, it is only an approximation of the *Raytrace* method, which implies it might produce some artifacts and it cannot use the sky's texture as the base color.

This method seems to tend to "over-occlude" the results. You have two complementary options to reduce this problem:

Passes Set the number of pre-processing passes, between (0 to 10) passes. Keeping the pre-processing passes high will increase render time but will also clear some artifacts and over-occlusions.

Error This is the tolerance factor for approximation error (i.e. the max allowed difference between approximated result and fully computed result). The lower, the slower the render, but the more accurate the results... Ranges between (0.0 to 10.0), defaults to 0.250.

Pixel Cache When enabled, it will keep values of computed pixels to interpolate it with its neighbors. This further speeds up the render, generally without visible loss in quality...

Correction A correction factor to reduce over-occlusion. Ranges between (0.0 to 1.0) correction.

Common Settings

Falloff When activated, the distance to the occluding objects will influence the "depth" of the shadow. This means that the further away the occluding geometry is, the lighter its "shadow" will be. This effect only occurs when the *Strength* factor is higher than 0.0. It mimics light dispersion in the atmosphere...

Strength Controls the attenuation of the shadows enabled with *Use Falloff*. Higher values give a shorter shadow, as it falls off more quickly (corresponding to a more foggy/dusty atmosphere). Ranges from (0.0 to 10.0), default is 0.0, which means no falloff.

Technical Details

Ambient occlusion is calculated by casting rays from each visible point, and by counting how many of them actually reach the sky, and how many, on the other hand, are obstructed by objects.

The amount of light on the point is then proportional to the number of rays which have "escaped" and have reached the sky. This is done by firing a hemisphere of shadow rays around. If a ray hits another face (it is occluded) then that ray is considered "shadow", otherwise it is considered "light". The ratio between "shadow" and "light" rays defines how bright a given pixel is.

Hints

Ambient occlusion is a ray-tracing technique (at least with the *Raytrace* method), so it tends to be slow. Furthermore, performance severely depends on octree size, see the *rendering chapter* for more information.

Mist

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender makes objects farther away more transparent (decreasing their Alpha value) so that they mix more of the background color with the object color. With Mist enabled, the further the object is away from the camera the less its alpha value will be.

Option

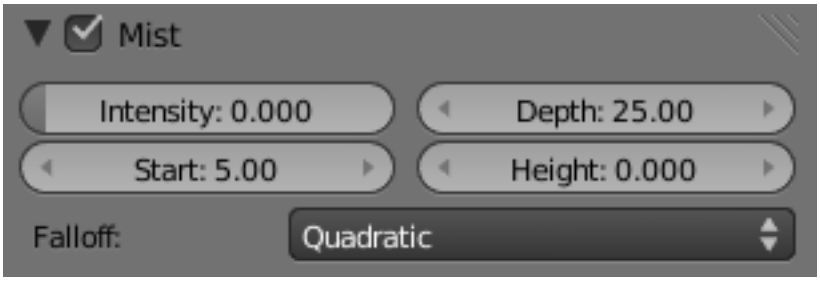


Fig. 2.1885: Mist panel.

Mist check box Toggles mist on and off.

Minimum An overall minimum intensity, or strength, of the mist.

Start The distance from the camera at which the mist starts to fade in

Depth The distance from *Start* of the mist, that it fades in over. Objects further from the camera than *Start* + *Depth* are completely hidden by the mist.

Height Makes the mist intensity decrease with height, for a more realistic effect. If greater than 0, it sets, in Blender units, an interval around z=0 in which the mist goes from maximum intensity (below) to zero (above).

Falloff The decay rate of the mist (*Quadratic*, *Linear*, *Inverse Quadratic*). These settings control the rate of change of the mist's strength further and further into the distance.

Note: Mist distances

To visualize the mist distances in the 3D View, select your camera, go to the camera menu, and enable *Show Mist*.

The camera will show mist limits as a line projecting from the camera starting from *Start* and of distance *Depth*.

To get a better view to evaluate the *Mist* visualization, `Shift-Numpad1` with the camera selected and `Numpad5` to toggle perspective view on and off. This will place the 3D View right over the camera looking down.

Transparency

Because *Mist* works by adjusting transparency, this can sometimes cause objects to be partially transparent when they should not be. One workaround is to set the Mist settings as desired, but turn Mist off. The Mist data is still available for compositing even though it is off. Use *Do Composite* and the *Node Editor* to feed the Mist pass to an *Alpha Over* to blend the background color (or a render layer with just the sky) with the rendered image. This produces the mist effect but since Mist is off the object transparency (or lack of) is preserved.

Examples

Fig. 2.1886: Mist example.

In this example (blend-file) the *Mist* → *Height* options has been limited to create smoke covering the floor.

Settings

Render Layers

This section covers only the Render Layer settings appropriate for the Blender Render engine. For the engine-independent settings, see *this section*.

Light Override Enter the name of a light group, and the scene will be lit with only those lights.

Examples of where this might be used:

- Using multiple Render Layers with different light group overrides adds the possibility to tweak light intensity and color in the compositor, i.e. to avoid re-renders.
- Speed up a draft render by using only a few lights instead of all of them.

Include Options

Each render layer has its own set of features which can be enabled/disabled for the whole layer. This could be used to save render time and gives control over the passes:

Zmask Actives masking with the selected Mask Layers. Only render what is in *front* of the solid Z values.

Negate Only render what is *behind* the solid Z values.

All Z Z-values are computed for everything in view, not just those things that are rendered. When disabled, objects not included in the render have no ("infinite") z value.

Solid Solid faces are rendered. All normal meshes are solid faced.

Halo Halo materials are rendered.

ZTransp Transparency may be Z-based or Ray-traced. If Z-based, enabling *Ztransp* renders transparent areas with the z-value of what is behind the transparent area.

Sky Turning on Sky renders the sky, as defined in your material world settings. Otherwise, a black alpha transparent background is rendered.

Edge If Edge is enable in the Post Processing panel, objects in this Render Layer are given an outline edge. Turning on Edge pulls in the Edge settings from the Render tab, and adds an outline to the objects. Edges also have to be enabled on the Render tab.

Strand Strands are strings of static particles that are colored as part of the material settings; they look like strands of hair or grass.

Freestyle Render the Freestyle strokes on this layer.

Render Passes

Render Passes are necessary because of the different things the Blender Render Engine must calculate to give you the final image. In each 'pass' the engine calculates different interactions between objects.

Render Passes In Detail

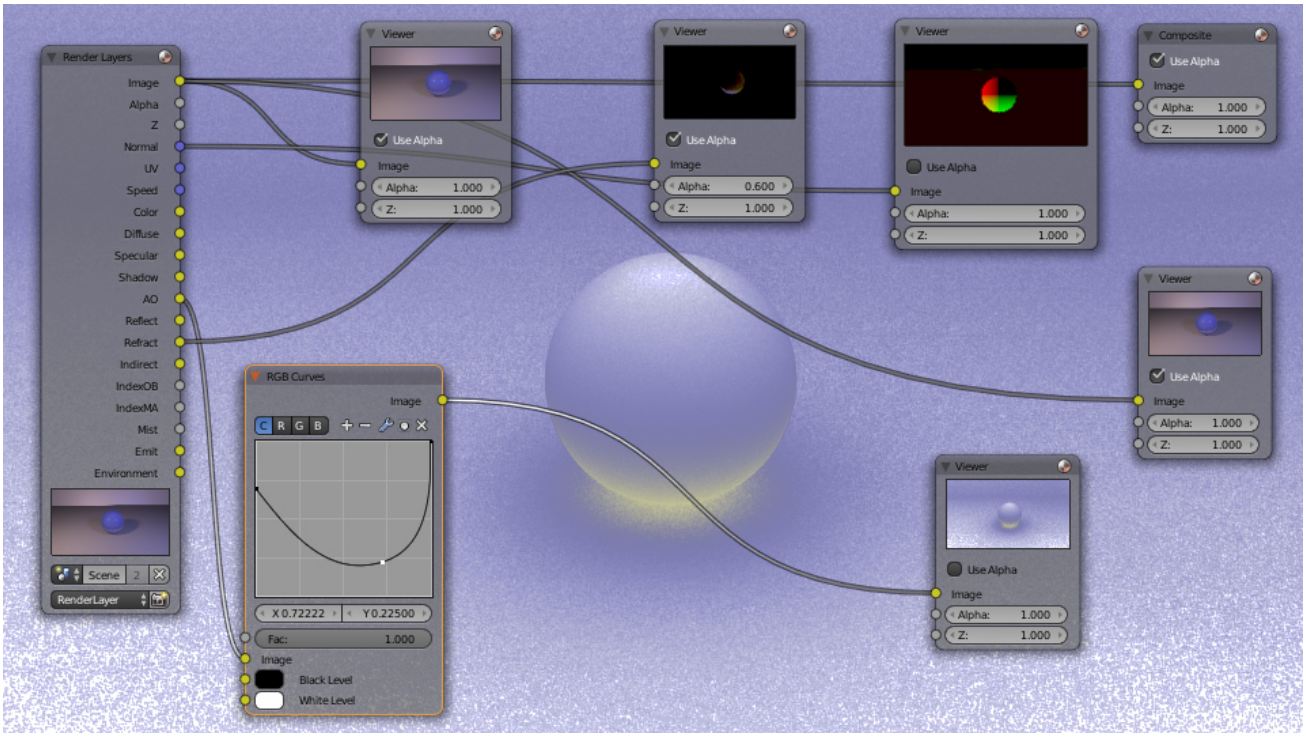
Everything you see in a render must be calculated for the final image. All interactions between objects in your scene, lighting, cameras, background images, world settings, etc., must all be separately calculated in different passes for different reasons, such as calculating shadows.

In a render, every pixel has been calculated several times to make sure it will show the right color for the right part of the image. Various things that are calculated in a standard render include:

- Where are shadows cast?
- How is ambient light in the environment blocked (occluded) by objects in the scene?
- How is light reflected off mirrored surfaces? Like shadows, lines are calculated, except this time they come from the camera and bounce off mirrored surfaces, so that when these lines hit an object, the engine calculates that this is what the camera should see.
- How is light bent (refracted) as is passes through transparent objects? Does it go straight through? Does it bend? If so, at what depth in the object?
- What designated objects are in the scene, and what is their outline? Should the object appear blurred, or should it appear in sharp focus?
- How fast is something moving (velocity)? Should it appear blurred given our frame rate or is it slow enough to still be focused on properly?
- How far away from the camera are objects' surfaces (Z-depth)? Can the object's surfaces be seen at all, or are they being blocked by another object's geometry?

- Does an object have a normal vector (bumpmap)? Do shadows and apparent geometry need to be calculated for any objects?
- Is there any specularity ? Are objects with textures such as metal shiny at all?

The answer to each of the above questions is an image or map, as shown below:



Each Render Pass puts out an image or a map. For the purposes of this example, a Render Layer was defined to produce all possible outputs. When a Render Layer input-node was added to the node diagram and the Render Layer input-node was subsequently associated with the Render Layer, all of the layer's outputs appeared as connection points on the right-hand (output) side of the node.

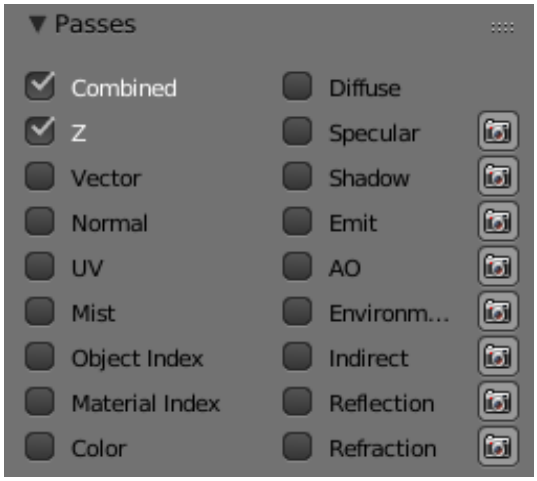
Render Passes that produce Images can be directly viewed in a viewer, or, if they are the only pass that is rendered, saved as the render image. If the pass is enabled, it can be saved in a multilayer OpenEXR format.

If the Render Pass output is not an image but is a map, it needs to be translated into something that we can see. For example, the Z-depth map is an array of values that specifies how far away from the camera each pixel is; values range between +/-3,000,000 Blender Units or so. The intermediate node you see above, between the Render Layer output socket and the Viewer node input socket (such as Map Value) does this translation or scaling. You must use that specific kind of translation node to get good results if you intend on operating on that map as an image. You must then, after making any adjustments, run the map back through that node to re-scale it back to the original before saving.

Selecting Render Passes

Render Passes are the various distinct outputs that the renderer is able to generate. All of the following render outputs are normally combined into a single output known, appropriately enough, as the *Combined* output. But you can also select any of them to be output as a separate pass. (If you do so, in most cases you can choose whether to *also* continue to include it in the Combined output.)

Some of these outputs must be enabled and used within your scene (and not just selected in the Render Layer panel) in order to show anything. For example, if you do not have any lights in your scene, or those lights have been set to not cast shadows, or objects in the limelight do not have materials which have been set to receive shadows, the *Shadow* pass will be blank; there is simply nothing to show you. If you have not enabled *Ambient Occlusion* in your World environment settings, the *AO* pass will be blank, even if you select it here.



To save time and disk space, you have to tell Blender each of the passes to render in the Render Layers panel (which we first introduced on *the previous page*):

Combined This renders everything in the image, even if it is not necessary. ("The whole enchilada," so to speak.) This is all the options below, blended into a single output, *except* those options which you have indicated should be omitted from this pass, as indicated with the camera button.

Z The Z-depth map; how far away each pixel is from the camera. Used for Depth-Of-Field (DOF). The depth map is inverse linear *(1/distance)* from the camera clip start.

Vector The direction and speed things are moving. Used with Vector Blur.

Normal Calculates lighting and apparent geometry for a bumpmap (an image which is used to fake detail on an object) or for changing the apparent direction of light falling on an object.

UV Allows texturing after rendering. See UV node.

Mist Deliver Mist factor pass.

Object Index Masks selected objects. See *ID Mask Node*.

Color The color of materials without shading.

Diffuse The diffuse shading of materials.

Specular Specular highlights.

Shadow Shadows cast. Make sure shadows are cast by your lights (positive or negative), and received by materials. To use this pass, mix multiply it with the Diffuse pass.

Emit Emission pass.

AO Ambient Occlusion. Make sure it is turned on in your environment and that Ray Tracing is enabled.

Environment Environment lighting.

Indirect Indirect lighting pass.

Reflection Reflection off mirrors and other reflective surfaces (highly waxed white floors, for example). Mix Add this pass to Diffuse to use it.

Refraction Refraction of colors through transparent meshes. Mix Add this pass to the Diffuse pass to use it.

When you enable a pass, the appropriate socket on the Render Layers node shows up like magic, and can be used as shown in the example above.

Excluding Render Passes

As we said, the *Combined* output is an amalgam of several outputs which are *also* available separately. When you select one of these outputs, they will be provided separately *and also* included in the Combined pass.

When you enable the Camera icon that is beside several of the pass options, the particular pass will be excluded from the combined pass. They will be made available separately *but not* included in the combined pass.

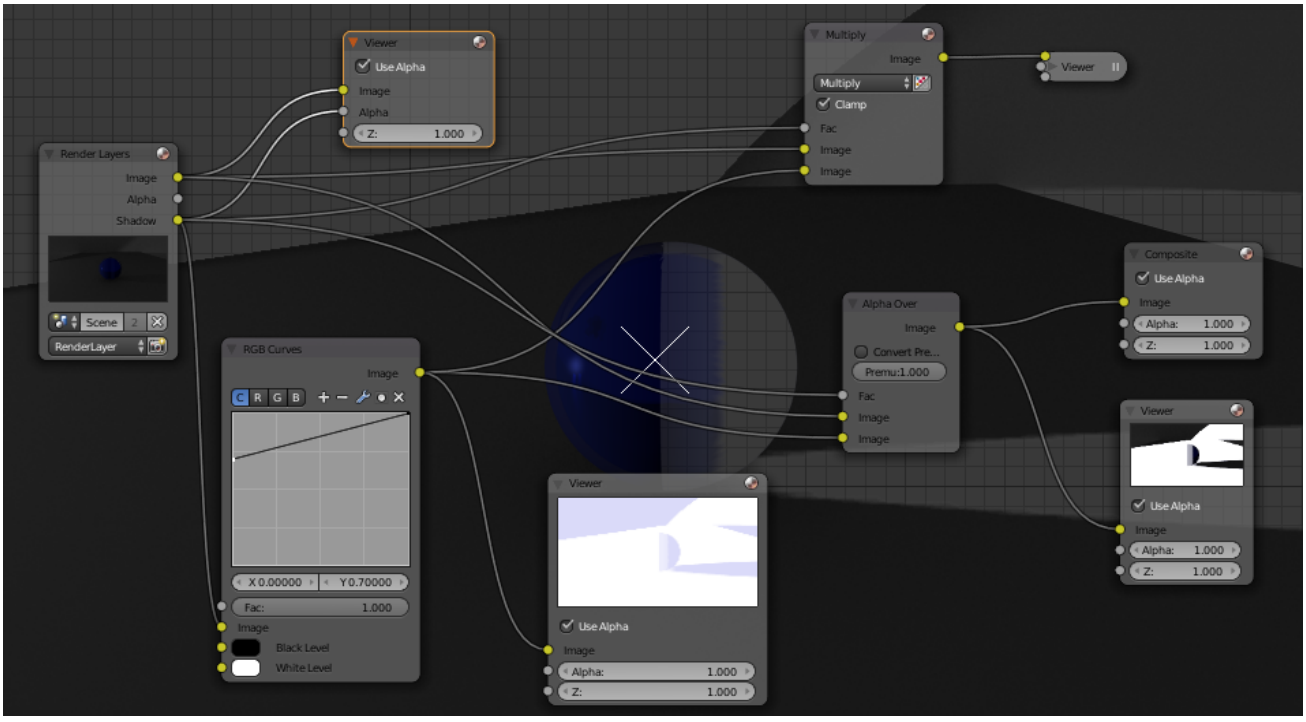
Using Render Passes

The primary purpose of Render Passes is to enable you to process the various outputs in different ways, by constructing networks of render nodes. You can achieve many special effects, and economize considerably on the render times of complicated scenes, by creative and effective use of this facility. We'll show you a few examples of this in just a moment.

Quite a bit of information about the typical uses for some of the passes is discussed elsewhere:

- Image: Since this is the main product, all of Blender uses it.
- Alpha: See the *Alpha Over* node and all of the *Matte* nodes.
- Z: See the *Defocus* node.
- Vector: See the *Vector Blur* node.
- Normal: See the *Normal* node.

Recoloring Shadows



Let us run the Shadow buffer through a colorization node setup, then recombine it; all your shadows will be artificially colored. Lots of threads in this node setup are shown to the right, so let us walk through it. On the left is the Render Layer input node: it refers to one of the Render Layers that we have defined for our scene. In the scene, we have a reflective ball on a pedestal standing in front of a backdrop. Everything (except the ball) is gray. We use a standard four-light rig: backfill placed high, two sidefills at ground level, and a key light above and to the left of camera. Suzanne, a monkey-shaped geometry, is standing in front of the key light, so her shadow is cast into the scene on the floor. The ball casts shadows onto the backdrop and floor.

The output channels of the Render Layer node are determined by which buttons we selected when defining our Render Layer. The top two viewers show you the image output using the Shadow as the Alpha channel, and the node next to it just the Shadow channel. Where the Shadow is dark, the image in the left viewer is transparent. We have used the Shadow to cut out parts of the image.

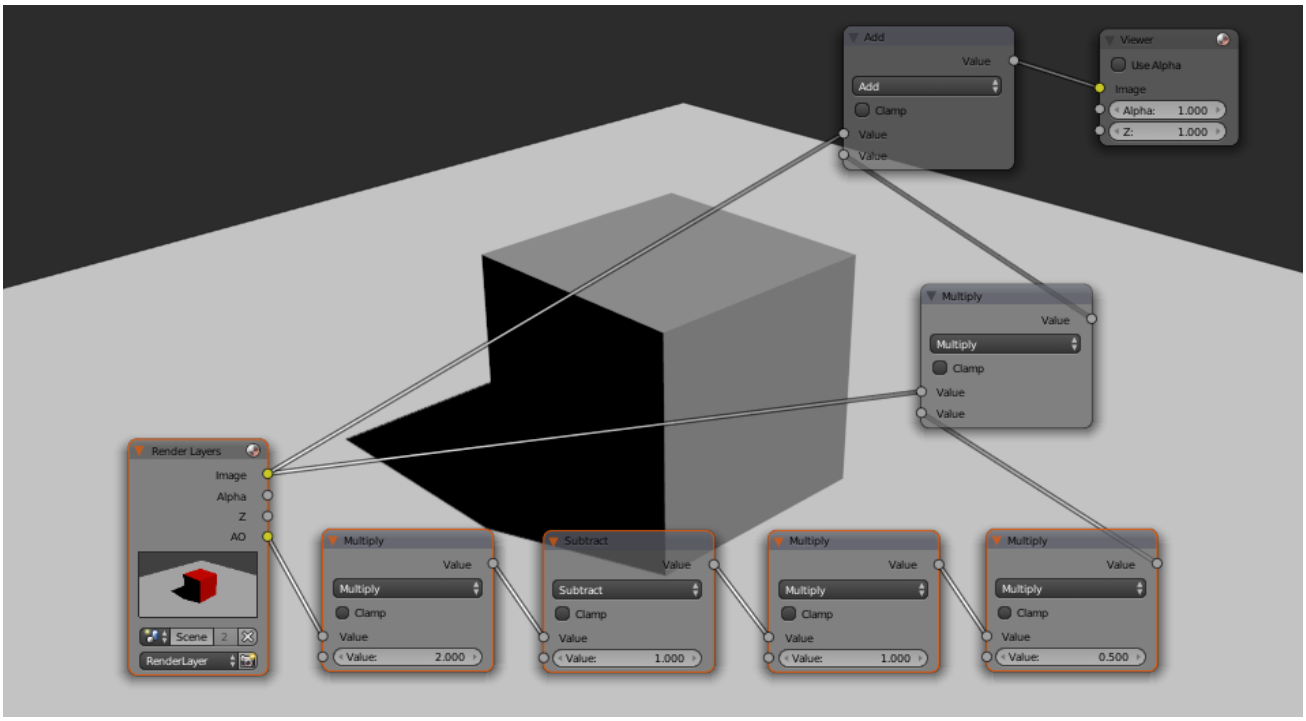
We then take the shadow through an RGB Curve, which is set to magnify just the Blue by 75%; so a gray shadow of RGB(40, 40, 40) becomes RGB(40, 40, 40×1.75=70). That blue-tinged shadow is shown in the bottom viewer. Now we have two options: Alpha Over and Mix. For either option:

- Use the Shadow map as a Factor.
- Feed the Blue Shadow to the Top Socket.
- Feed the core or base image to the Bottom Socket.

The resulting image is the same in either case; a blue shadow. Note that Suzanne's reflection is not blue; there is a different Render Pass for that.

You could just as easily swap in another image entirely; for example, the shadow map from another render layer. You can even take an image from another project entirely and use that instead (using the Image Input node), to get a different effect. (For example, an effect similar to a *Star Wars Episode One* movie poster, where Anakin Skywalker already casts the shadow of Darth Vader.)

Compositing Ambient Occlusion



AO is a geometry-based dirt shader, making corners darker. It is separately enabled in the World settings and computed as a separate pass. When enabled, it has one of three Modes: *Add*, *Subtract* , *Both* and a variable *Energy* level (which changes the intensity of the shading). The third variable is the amount of Ambient light that the material receives. If it does not receive any, then ambient occlusion does not affect it. Based on these variables, Blender computes an AO pass. If you call it out as a separate pass and wish to composite it back into your image, you will need to enable the Color and Diffuse pass as well.

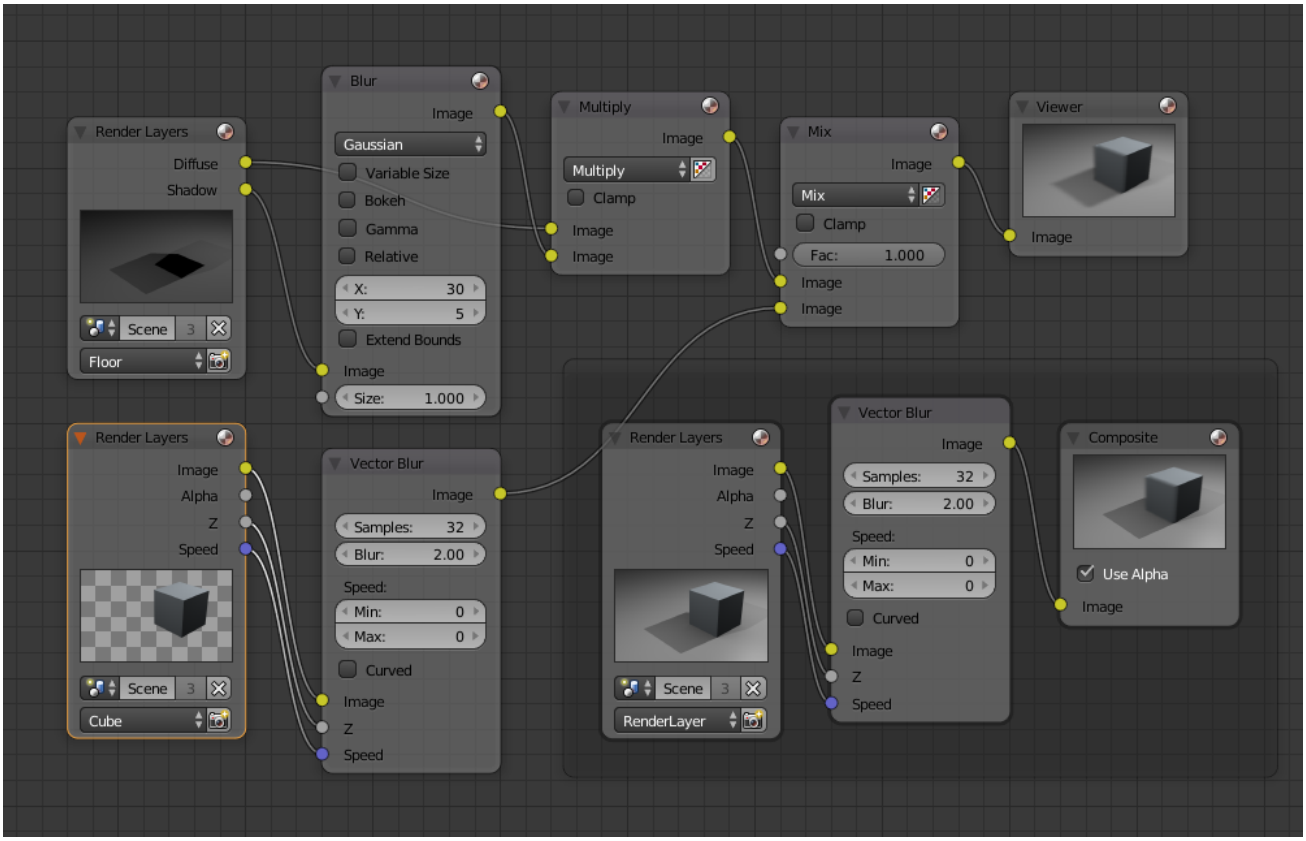
To configure your setup, consider the example image above.

1. First, depending on the AO mode do one of the following: If AO mode is Add: directly use the AO pass. If AO mode is Sub: Calculate AO - 1, or if AO mode is Both: Calculate 2 × AO - 1.
2. Multiply the output of Step 1 with the AO energy level.
3. Multiply the output of Step 2 with the material's ambience value. If you have materials which receive different ambience light levels (0.5 is the default), one would have to create an ambience map based on Object ID.
4. Multiply the output of Step 3 with the color pass.

5. Add the output of Step 4 to the diffuse pass.

If shadows, colored ambient light, specularity, reflections, and/or refractions are involved they have to be added to the diffuse pass before adding the converted AO pass.

Vector Blurring Shadows



When using Vector Blur instead of Motion Blur, objects in motion are blurred, but objects at rest (with respect to the camera) are not blurred. The crossover is the shadow of the object in motion. Above, we have a cube in motion across a ground plane. If we just ran the combined pass through Vector Blur, you can see the result in the lower right-hand corner; the box is blurred, but its shadow is sharply in focus, and thus the image does not look realistic.

Therefore, we need to separate out the diffuse and shadow passes from the floor by creating a "Floor" render layer. That render layer has Diffuse and Shadow passes enabled, and only renders the floor object (layer 2). Another render layer ("Cube") renders the Z and Vector passes, and only renders the cube (on layer 1). Using the Blur node, we blur the shadow pass, and then combine the diffuse and blurred shadow by multiplying them together in a Mix Multiply node; we then have a blurred shadow on a crisp ground plane. We can then mix the vector-blurred object to provide a realistic-looking image.

Conclusion

Render Passes can be manipulated to give you almost complete control over your final image. Causing objects to cast shadows that are not really their shadows, making objects appear out of focus or sharply in focus like a real camera, manipulating colors just for final post-processing or just reconfiguring your render passes to save render time, are all things which you might wish to manipulate the render engine for.

Motion Blur

Blender's animations are by default rendered as a sequence of *perfectly still* images. While great for stop-motion and time-lapses, this is unrealistic, since fast-moving objects do appear to be blurred in the direction of motion, both in a movie

frame and in a photograph from a real-world camera.

Blender has two ways to achieve motion blur:

Sampled Motion Blur

Blender can be made to render the current frame and some more 'virtual' frames in between it and the next frame, then merge them all together to obtain an image where moving objects are 'blurred'.

This method is slow, but produces good results. It can be activated in the *Sampled Motion Blur* panel of the render settings. This kind of motion blur is done during the render.

Motion Samples Set the number of samples to take for each frame. The higher the samples, the smoother the blur effect, but the longer the render, as each virtual intermediate frame has to be rendered.

Shutter Time (in frames) the shutter is open. If you are rendering at 24 fps, and the Shutter is set to 0.5, the time in between frames is 41.67 ms, so the shutter is open for half that, 20.83 ms.

Note: Samples are taken only from the *next* frame, not the previous one. Therefore the blurred object will appear to be slightly ahead of how it would look without motion blur.

Vector Blur

Vector Blur is faster but sometimes has unwanted side-effects which are sometimes difficult to avoid.

Vector blur is a process done in compositing (post-render time), by rendering the scene without any blur, plus a pass that has movement information for each pixel. This information is a vector map which describes a 2D or 3D direction and magnitude. The compositor uses that data to blur each pixel in the given direction.

Examples

To better grasp the concept, let us assume that we have a cube 2 units wide, uniformly moving 1 unit to the right at each frame. The scale beneath the cube helps in appreciating the movement of 1 Blender unit.

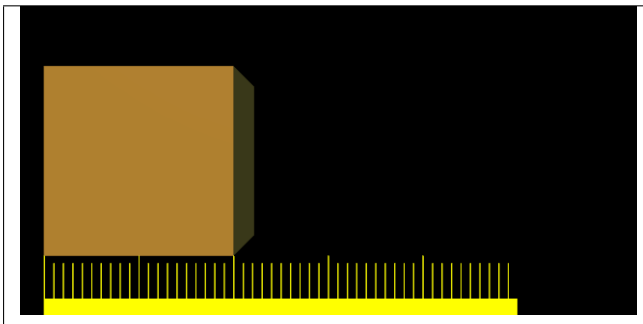


Fig. 2.1887: Frame 1 of the moving cube without Motion Blur.

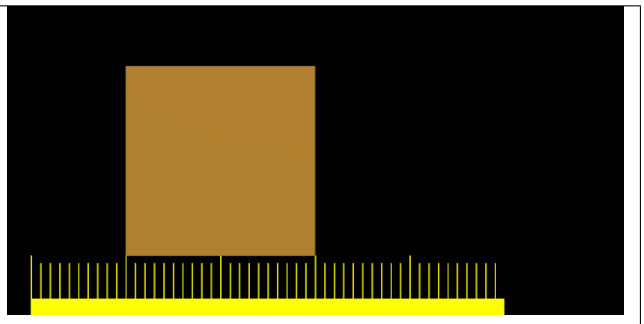


Fig. 2.1888: Frame 2 of the moving cube without Motion Blur.

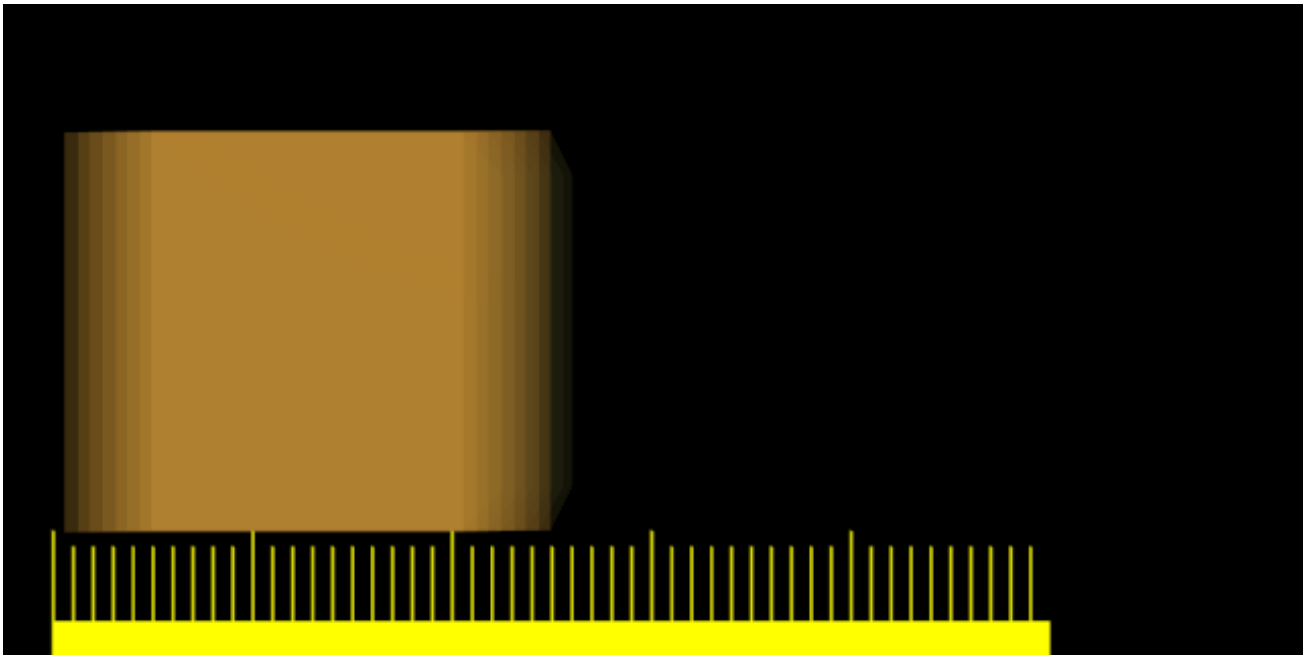


Fig. 2.1889: Frame 1 when Sampled Motion Blur is enabled and eight 'intermediate' frames are computed. Shutter is set to 0.5 , thus the image eight samples are rendered between frame 1 and halfway to frame 2.

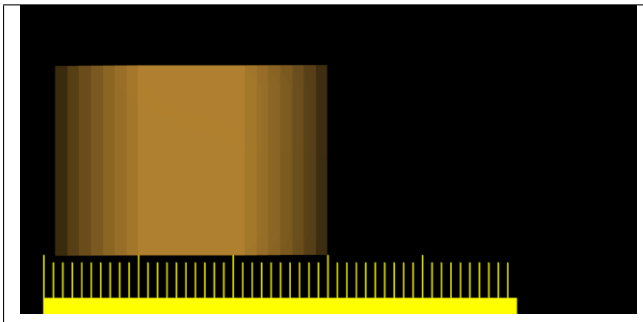


Fig. 2.1890: The effect of an increased shutter value.

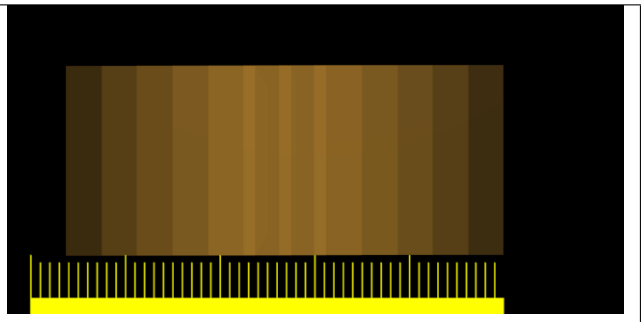


Fig. 2.1891: Even further increased shutter value.

Values greater than 1 are physically impossible in a real-world camera, but can be used to exaggerate the effect. Better results than those shown can be obtained by using higher samples than 8, but, of course, since as many *separate* renders as samples are needed, a Motion Blur render takes that many times more time than a non-Motion Blur one.

Hints

Sampled Motion Blur can be used as an additional form of *Anti-Aliasing*, since aliasing artifacts are computed differently for each sample and averaged together at the end.

Anti-Aliasing

A computer generated image is made up of pixels; each pixel can of course only be a single color. In the rendering process the rendering engine must therefore assign a single color to each pixel on the basis of what object is shown in that pixel. This often leads to poor results, especially at sharp boundaries, or where thin lines are present, and it is particularly evident for oblique lines.

To overcome this problem, which is known as *Aliasing*, it is possible to resort to an Anti-Aliasing technique. Basically, each pixel is 'oversampled', by rendering it as if it were five pixels or more, and assigning an 'average' color to the rendered pixel.

The buttons to control Anti-Aliasing, or Over Sampling (OSA), are below the rendering button in the *Render* Panel (*Render Panel.*).

Options

Anti-Aliasing (check box) Enables oversampling.

Samples The number of samples per pixel. Higher value produces better edges, but slows down the rendering.

5, 8, 11, 16

By default, we use in Blender a fixed "Distributed Jitter" table. The samples within a pixel are distributed and jittered in a way that guarantees two characteristics:

- Each sample has equal distances to its neighbor samples.
- The samples cover all sub-pixel positions equally, both horizontally and vertically.

The images below show Blender sample patterns for 5, 8, 11 and 16 samples. To show that the distribution is equalized over multiple pixels, the neighbor pixel patterns were drawn as well. Note that each pixel has an identical pattern.

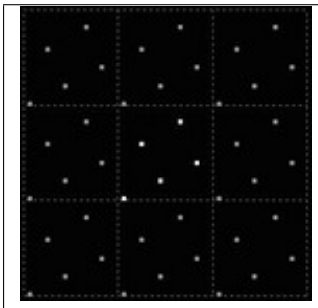


Fig. 2.1892: 5 samples.

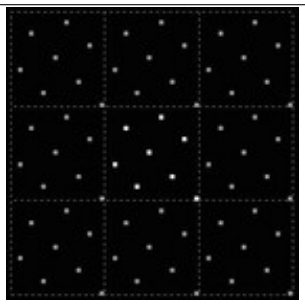


Fig. 2.1893: 8 samples.

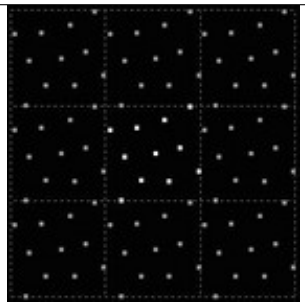


Fig. 2.1894: 11 samples.

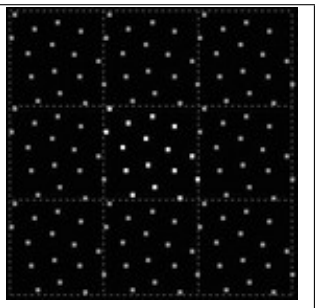


Fig. 2.1895: 16 samples.

Full Sample For every anti-aliasing sample, save the entire Render Layer results. This solves anti-aliasing issues with compositing.

Filtering

When the samples have been rendered, we've got color and alpha information available per sample. It then is important to define how much each sample contributes to a pixel.

The simplest method is to average all samples and make that the pixel color. This is called using a "Box Filter". The disadvantage of this method is that it does not take into account that some samples are very close to the edge of a pixel, and therefore could influence the color of the neighbor pixel(s) as well.

Filter menu: Set The filter type to use to 'average' the samples:

Box A low-quality box-shaped curve.

> **Note:** This filter is relatively low quality. You can see that only the samples within the pixel itself are added to the pixel's color. For the other filters, the formula ensures that a certain amount of the sample color gets distributed over the other pixels as well.

Tent A simplistic filter that gives sharp results.

Quadratic A Quadratic curve.

Cubic A Cubic curve.

Gauss Gaussian distribution, the most blurry.

Catmull-Rom Catmull-Rom filter, gives the most sharpening.

Mitchell-Netravali A good all-around filter that gives reasonable sharpness.

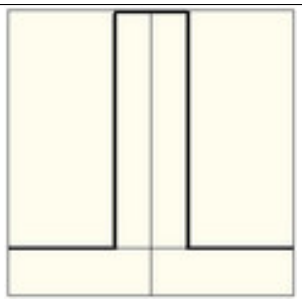


Fig. 2.1896: Box.

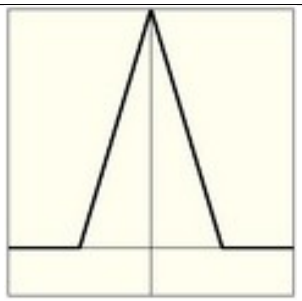


Fig. 2.1897: Tent.

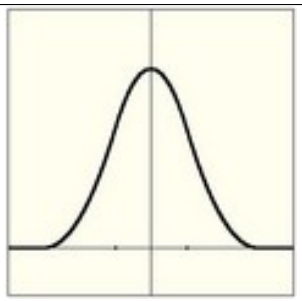


Fig. 2.1898: Quadratic.

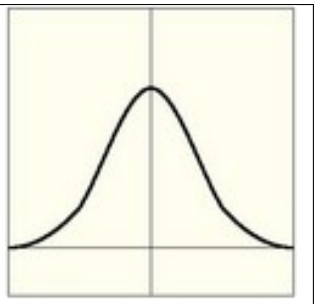


Fig. 2.1899: Cubic.

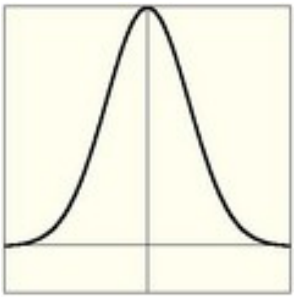


Fig. 2.1900: Gaussian.

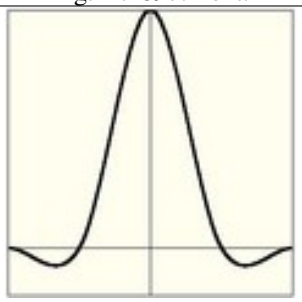


Fig. 2.1901: Catmull-Rom.

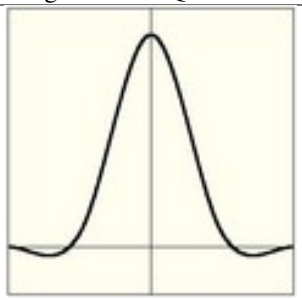


Fig. 2.1902: Mitchell-Netravali.

Filter Size

Making the filter size value smaller will squeeze the samples more into the center, and blur the image more. A larger filter size makes the result sharper. Notice that the last two filters also have a negative part; this will give an extra sharpening result.

Examples

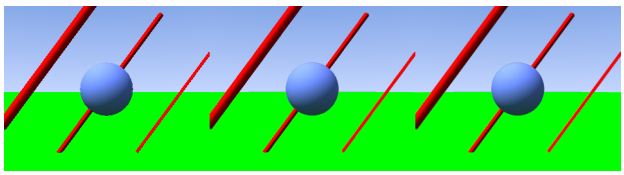


Fig. 2.1903: Todo.

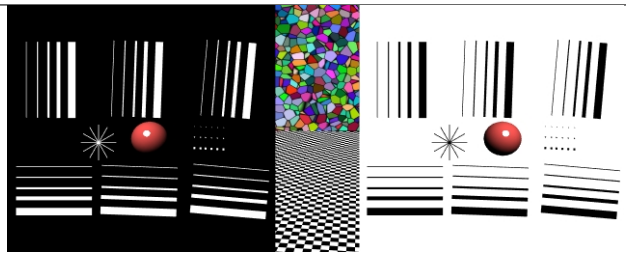


Fig. 2.1904: AA 8, Box filter.

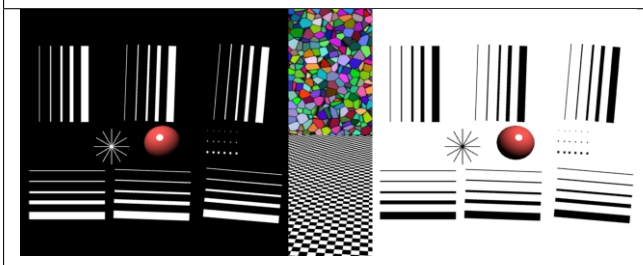


Fig. 2.1905: AA 8, Tent filter.

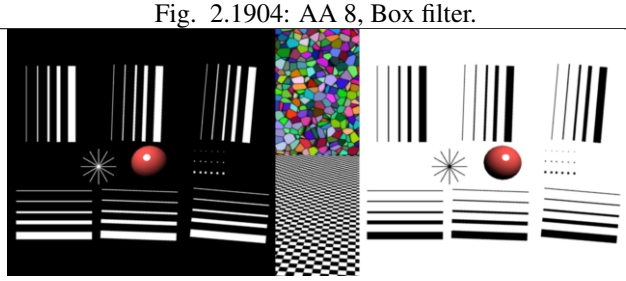


Fig. 2.1906: AA 8, Quadratic filter.

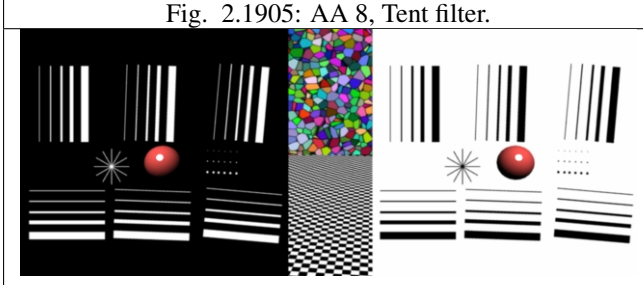


Fig. 2.1907: AA 8, Cubic filter.

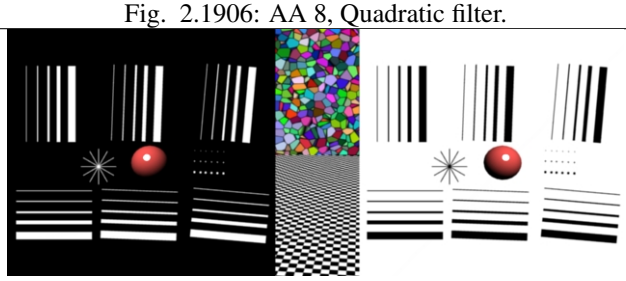


Fig. 2.1908: AA 8, Gaussian filter.

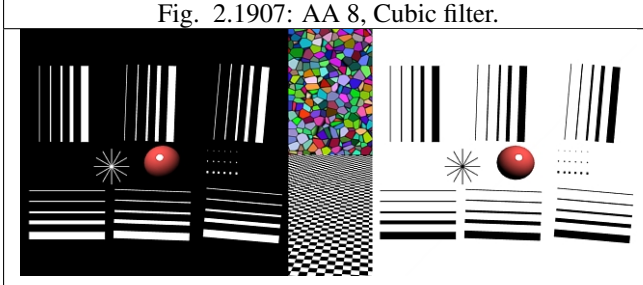


Fig. 2.1909: AA 8, Catmull-Rom filter.

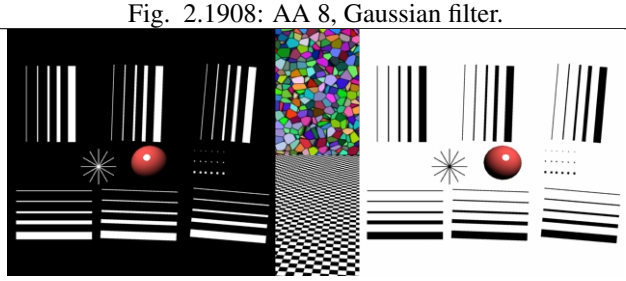


Fig. 2.1910: AA 8, Mitchell-Netravali filter.

Post Processing

Edge Rendering

Blender's toon shaders can give your rendering a comic-book-like or manga-like appearance, affecting the shades of colors. The effect is not perfect since real comics and manga also usually have china ink outlines. Blender can add this feature as a post-processing operation.

Options

Edge This makes Blender search for edges in your rendering and add an 'outline' to them.

Before repeating the rendering it is necessary to set some parameters:

Fig. 2.1911: A scene with Toon materials.

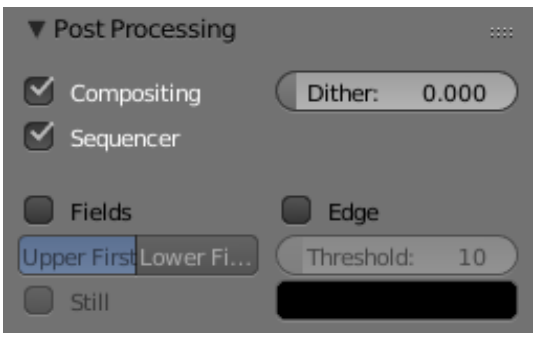


Fig. 2.1912: Toon edge buttons.

Threshold The threshold of the angle between faces for drawing edges, from 0 to 255. A value of 10 would just give outline of object against the background, whereas higher settings start to add outlines on surface edges, starting with sharper angles. At maximum intensity, edge will even faintly display geometry subdivided edge lines in areas of imperfect smoothing.

Color RGB The color of the rendered edges.

Examples

Fig. 2.1913: Scene re-rendered with toon edge set.

It is possible to separate out the edge layer using a render layer dedicated to that purpose. The alpha channel is 0 where there is no edge, and 1 where the edge is. By separating out the edge layer, you can blur it, change its color, mask it, etc. The image above shows how to do this. In this example, an Edge render layer is created that only has the Sky and Edge layers The other render layer omits the Edge layer, so it returns just the normal image. On the output panel *Edge* is enabled with a width of 10 in black. Then that layer goes through a blur node. Using the Alpha Over node, then the cube is composited on top of the blurred edge. The result gives a soft-shadow kind of effect. Note that Premultiply is set, because the Edge image already has an alpha of 1.0 set.

Fields

The TV standards prescribe that there should be 25 frames per second (PAL) or 30 frames per second (NTSC). Since the phosphors of the screen do not maintain luminosity for very long, this could produce a noticeable flickering.

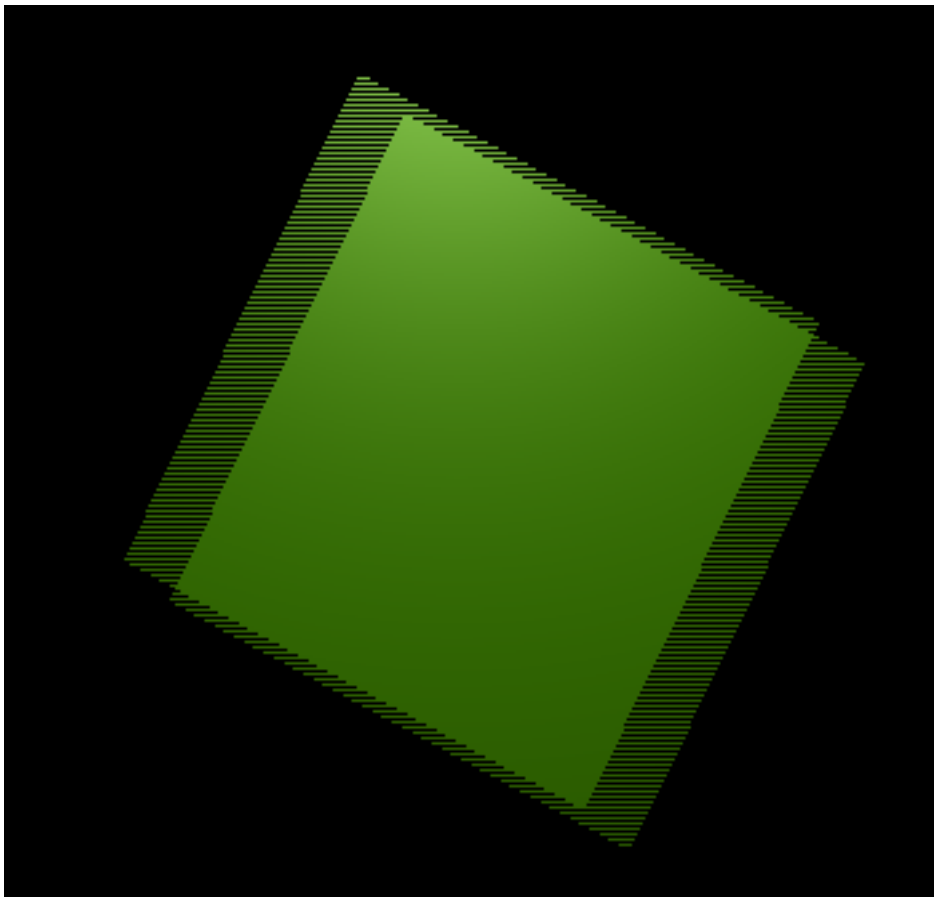


Fig. 2.1914: Field Rendering result.

To minimize this, a TV does not represent frames as a computer does ('progressive' mode), but rather represents half-frames, or *fields* at a double refresh rate, hence 50 half frames per second on PAL and 60 half frames per second on NTSC. This was originally bound to the frequency of power lines in Europe (50Hz) and the US (60Hz).

In particular, fields are "interlaced" in the sense that one field presents all the even lines of the complete frame and the subsequent field the odd ones.

Since there is a non-negligible time difference between each field (1/50 or 1/60 of a second) merely rendering a frame the usual way and splitting it into two half frames does not work. A noticeable jitter of the edges of moving objects would be present.

Options

Fields

Enable field rendering. When the *Fields* button in the *Render* Panel is pressed (*Post Processing* section), Blender prepares each frame in two passes. On the first it renders only the even lines, then it advances in time by half a time step and renders all the odd lines. This produces odd results on a PC screen (Field Rendering result). but will show correctly on a TV set.

Upper First / Lower First Toggles between rendering the even and odd frames first.

Still Disables the half-frame time step between fields (*x*).

Note: Setting up the correct field order

Blender's default setting is to produce Even fields *before* Odd fields; this complies with European PAL standards. Odd fields are scanned first on NTSC.

Of course, if you make the wrong selection things are even worse than if no Field rendering at all was used!

If you are really confused, a simple trick to determine the correct field order is to render a short test animation of a white square moving from left to right on a black background. Prepare one version with odd field order and another with even field order, and look at them on a television screen. The one with the right field order will look smooth and the other one horrible. Doing this simple test will save you *hours* of wasted rendering time...

Note: Fields and Composite Nodes

Nodes are currently not field-aware. This is partly due to the fact that in fields, too much information is missing to do good neighborhood operations (blur, vector blur etc.). The solution is to render your animation at double the frame rate without fields and do the interlacing of the footage afterwards.

Render Baking

Baking, in general, is the act of pre-computing something in order to speed up some other process later down the line. Rendering from scratch takes a lot of time depending on the options you choose. Therefore, Blender allows you to "bake" some parts of the render ahead of time, for select objects. Then, when you press Render, the entire scene is rendered much faster, since the colors of those objects do not have to be recomputed.

Render baking creates 2D bitmap images of a mesh object's rendered surface. These images can be re-mapped onto the object using the object's UV coordinates. Baking is done for each individual mesh, and can only be done if that mesh has been UV-unwrapped. While it takes time to set up and perform, it saves render time. If you are rendering a long animation, the time spent baking can be much less than time spent rendering out each frame of a long animation.

Use Render Bake in intensive light/shadow solutions, such as AO or soft shadows from area lights. If you bake AO for the main objects, you will not have to enable it for the full render, saving render time.

Use *Full Render* or *Textures* to create an image texture; baked procedural textures can be used as a starting point for further texture painting. Use *Normals* to make a low-resolution mesh look like a high-resolution mesh. To do that, UV-unwrap a high-resolution, finely sculpted mesh and bake its normals. Save that normal map, and *Mapping* (texture settings) the UV

of a similarly unwrapped low-resolution mesh. The low-resolution mesh will look just like the high-resolution, but will have much fewer faces/polygons.

Advantages

- Can significantly reduce render times.
- Texture painting made easier.
- Reduced polygon count.
- Repeated renders are made faster, multiplying the time savings.

Disadvantages

- Object must be UV-unwrapped.
- If shadows are baked, lights and object cannot move with respect to each other.
- Large textures (e.g. 4096×4096) can be memory intensive, and be just as slow as the rendered solution.
- Human (labor) time must be spent unwrapping and baking and saving files and applying the textures to a channel.

Options

Bake Mode

Full Render

Bakes all materials, textures, and lighting except specularity and SSS.

Ambient Occlusion

Bakes ambient occlusion as specified in the World panels. Ignores all lights in the scene.

Fig. 2.1915: Ambient Occlusion.

Normalized Normalize without using material's settings.

Shadow

Bakes shadows and lighting.

Normals

Bakes tangent and camera-space normals (amongst many others) to an RGB image.

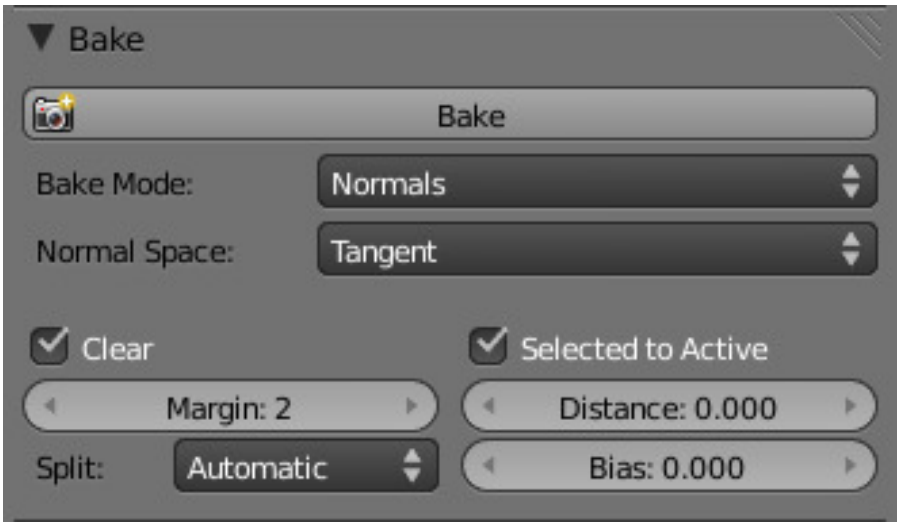


Fig. 2.1916: Normals.

Normal Space Normals can be baked in different spaces:

Camera space Default method.

World space Normals in world coordinates, dependent on object transformation and deformation.

Object space Normals in object coordinates, independent of object transformation, but dependent on deformation.

Tangent space Normals in tangent space coordinates, independent of object transformation and deformation. This is the new default, and the right choice in most cases, since then the normal map can be used for animated objects too.

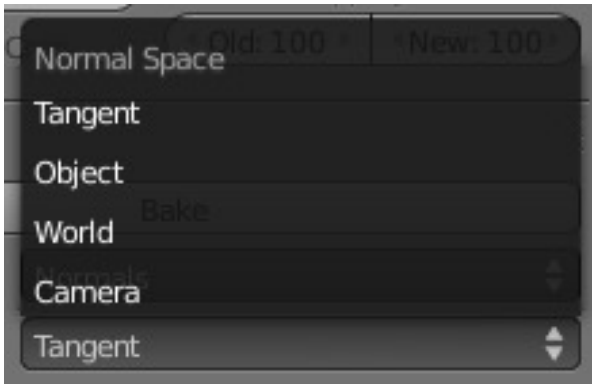


Fig. 2.1917: Normal Space.

For materials the same spaces can be chosen as well, in the image texture options, next to the existing *Normal Map* setting. For correct results, the setting here should match the setting used for baking.

Textures

Bakes colors of materials and textures only, without shading.

Displacement

Similar to baking normal maps, displacement maps can also be baked from a high-res object to an unwrapped low-res object, using the *Selected to Active* option.

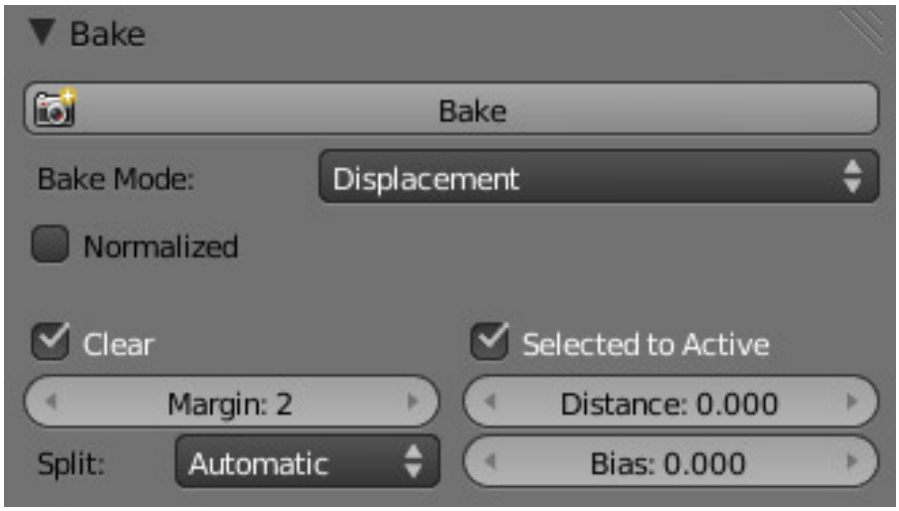


Fig. 2.1918: Displacement.

Normalized Normalize to the distance.

When using this in conjunction with a Subdivision Surface and Displacement modifier within Blender, it is necessary to temporarily add a heavy Subdivision Surface modifier to the 'low res' model before baking. This means that if you then use a displacement modifier on top of the Subdivision Surface, the displacement will be correct, since it is stored as a relative difference to the subdivided geometry, rather than the original base mesh (which can get distorted significantly by a Subdivision Surface). The higher the render subdivision level while baking, the more accurate the displacements will be. This technique may also be useful when saving the displacement map out for use in external renderers.

Emission

Bakes Emit, or the Glow color of a material.

Alpha

Bakes Alpha values, or transparency of a material.

Mirror Color and Intensity

Bakes Mirror color or intensity values.

Specular Color and Intensity

Bakes specular color or specular intensity values.

Additional Options

Clear If selected, clears the image to selected background color (default is black) before baking render.

Margin Baked result is extended this many pixels beyond the border of each UV "island," to soften seams in the texture.

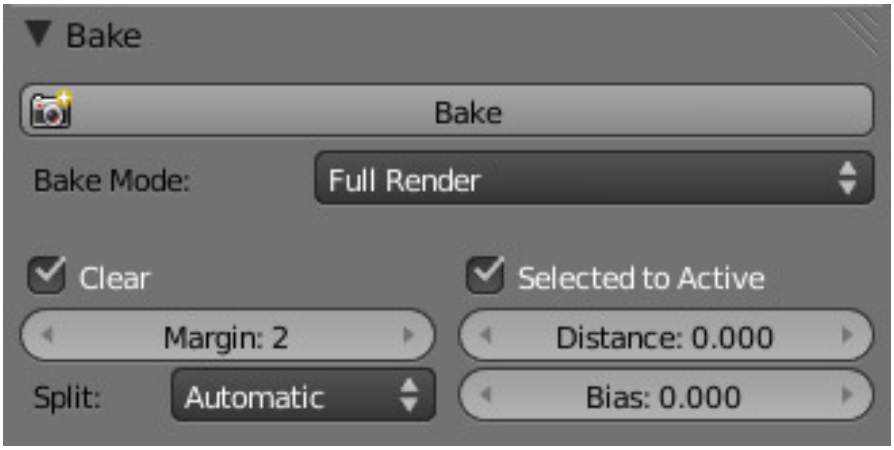


Fig. 2.1919: Full Render.

Split

Fixed Slit quads predictably (0, 1, 2) (0, 2, 3).

Fixed alternate Slit quads predictably (1, 2, 3) (1, 3, 0).

Automatic Split quads to give the least distortion while baking.

Select to Active Enable information from other objects to be baked onto the active object.

Distance Controls how far a point on another object can be away from the point on the active object. Only needed for *Selected to Active*. A typical use case is to make a detailed, high poly object, and then bake its normals onto an object with a low polygon count. The resulting normal map can then be applied to make the low poly object look more detailed.

Bias Bias towards further away from the object (in Blender units)

Note: Mesh Must be Visible in Render

If a mesh is not visible in regular render, for example because it is disabled for rendering in the Outliner or has the DupliVerts setting enabled, it cannot be baked to.

Workflow

1. In a 3D View editor, select a mesh and enter UV/Face Select mode
2. *Unwrap the mesh object*
3. In a UV/Image Editor, either create a new image or open an existing one. If your 3D View is in textured display mode, you should now see the image mapped to your mesh. Ensure that all faces are selected.
4. In the Bake panel at the bottom of the *Render menu*, bake your desired type of image (*Full Render* etcetera.)
5. When rendering is complete, Blender replaces the image with the Baked image.
6. Save the image.
7. Apply the image to the mesh as a UV texture. For displacement and normal maps, refer to *Bump and Normal Maps*. For full and texture bakes, refer to *Textures*.
8. Refine the image using the process described below, or embellish with *Texture Paint* or an external image editor.

Optimization

Render Quality

Many factors go into the quality of the rendered image. Rendering a scene without changing any of the render settings is probably going to produce a rather unpleasant image. In previous chapters, you have learned how to model, shade, texture, and light scenes. Optimizing settings with respect to those areas will help to produce quality images, but there are some important settings that come into play before pressing the render button. These can directly affect the look of the rendered image.

The next section covers render layers and render passes, both of which allow you to compose an image from several elements of a scene. In some cases it is necessary to render effects straight out of the renderer, rather than creating them in "post."

Color Management and Exposure

One important aspect of 3D rendering that is often overlooked is color management. Without color management, or more commonly, linear rendering, render engines interpret scene lighting correctly, but display them incorrectly on your monitor. Blender simplifies this workflow, but it is important to know how the color space of a rendered image factors into your pipeline.

See also:

Color Management and Exposure.

Anti-Aliasing

Anti-Aliasing removes jagged edges that appear in contrasting areas of color. This is a very important aspect of render quality. Without this render setting, images usually appear particularly CG and amateur.

See also:

Anti-Aliasing.

Exposure (Lighting)

Exposure is, in physical terms, the length of time a camera's film or sensor is exposed to light. Longer exposure times create a brighter image. In CG, the recorded light values are offset to simulate longer or shorter exposures. This can be achieved through lighting settings, or better, through *Color Management settings*

See also:

Exposure (Lighting).

Motion Blur

Cameras have a certain shutter speed, or the length of time the film is exposed to the image. Things that are in motion while the picture is taken will have some degree of blurring. Faster-moving objects will appear more blurred than slower objects. This is an important effect in CG because it is an artifact that we expect to see, and when it is missing, an image may not be believable.

See also:

Motion Blur.

Performance Considerations

Optimizing Render Performance

"A watched pot never boils" is the old saying, but you may wonder why your render takes so long to create, or worse, crashes mid-way through! Well, there is lots going on and lots you can do to speed up rendering or enable a complicated render to complete. Also, it is possible to render a very complicated scene on a mediocre PC by being "render-smart". Here is a "top ten" list of things to do or not do in order to speed up rendering or even avoid crashes during scene render. Some options may decrease the quality of your render, but for draft renders you may not care.

If you get the message "Malloc returns nil", in plain English that means the memory allocator tried to get more physical memory for Blender but came back empty-handed. This means that you do not have enough memory available to render the scene, and Blender cannot continue. You will need to do one or more of the following tasks on this page in order to render.

Hardware Improvements

- Install more system memory.
- Upgrade your CPU to a multi-core/multiprocessor
- Upgrade your OpenGL video drivers
- Get faster memory, up to your PC's motherboard limit.
- Use or set up a render farm using all available PCs in your house, or use a render farm.

Operating System Configuration

- Increase Blender's processing priority through your OS.
- Increase your swap file space used by the OS for memory swapping. Also called virtual memory pagefile size, up to the size of your physical memory.
- Use a system-monitor to check if any other processes are using significant CPU or RAM, which can be closed.
- Render in *background mode* (from the command line), saves extra memory.

Blender Settings

- Increase the MEM Cache Limit in the User Preferences System & OpenGL tab.
- Switch to an Orthographic camera, and render your own "parts" of the scene as separate images, and then paste those parts together in GIMP. An old trick in making your own panorama with a real camera is to take three or so pictures of a very wide (beach sunset) scene, where you take one picture, rotate to the right, snap another, then another, and when you get the pictures developed, you overlap them to make a very wide landscape image. Do the same in Blender: render out one shot to a file, then move the camera to look at a different area of the scene, and render that shot. Each shot will be of a smaller area and thus take in fewer polygons/faces. Be sure that when you position your camera that you snap overlapping shots, so that you can then match them up. If you do not want to use GIMP, you can use compositing nodes and the Translate node to match them up in Blender.
- Minimize the render window (and the Blender window, if the UV/image editor is used). ATI users report dramatic speedup on a per frame basis, which adds up over the frame range.
- Use the Big Render script to render sub-sections of the overall image, and then paste them together.

Scene and Specific Objects

1. Remove lamps, or move them to unrendered layers, or tie them to layers.
2. Turn off some lamp's shadows, using only one or two main sun lamps to cast shadows. A few "shadows only" lights will render faster than every light having shadows on.
3. Use Buffer Shadows rather than ray-traced Shadows
4. Bake your shadows using Render Baking Full Render bake on surfaces that do not move. Use that texture for that mesh, then disable shadows for that material.
5. Simplify meshes (remove polygons). The more vertices you have in camera, the more time it takes to render.
6. Remove Doubles, or use the Decimator mesh edit feature.
7. Remove Subdivision Surface and Multires modifiers.
8. Delete backsides of meshes (removing unseen geometry).
9. Render just a few objects at a time; in the beginning of your project, render the background objects and sets that will not change and will always be in the background.
10. Put the buildings on another layer, and through render layers, do not render them. Then composite them back in later.
11. Make the camera static so that you can better accomplish the above two ideas.
12. Avoid use of Area lights.
13. Make materials Shadeless.
14. Render Bake AO and textures, and then make those materials Shadeless.
15. Decrease the Clip distance for spot lights.
16. Decrease the Clip distance for the camera.
17. Turn off world AO.
18. Turn off Material SSS.
19. Use smaller image textures. A 256×256 image takes only 1% of the memory that a 2k image does, often with no loss of quality in the ultimate render.
20. Reduce Subdivision Surfaces. Each level quadruples (4x) the number of faces from the previous level.
21. Reduce Multires.
22. Make a matte render of background objects, like buildings, and put the image of them on a billboard in the scene instead of the object themselves. This will reduce vertex/face count.
23. If you have lots of linked instances of an object, use DupliFaces, as these are instanced. If you have 100 of them, Blender will only store the geometry for 1 (Instances themselves take a small amount of memory).

Render Settings

Output Panel

- Disable *Edge* rendering.
- *Save Buffers*.
- Render to an *UV/Image Editor*, not a pop-up. *Render Window*.
- Use multiple *Threads* on a multi-core CPU (with multiple *Parts*).
- Decrease the frame count of the animation (and use a lower framerate for the same duration of animation). For example, render 30 frames at 10 frames per second for a 3-second animation, instead of 75 frames at 25 frames per second.

Render Layers Panel

- Render only the Layers of interest.
- Render with all lights set to one simple spot (enter its name in the *Light:* field).
- Render with one material override (enter its name in the Mat: field).
- Disable unnecessary Render Passes, such as Z, or only render the pass of interest, such as Diffuse.

Shading Panel

- Turn off Shadows.
- Turn off Environment Mapping.
- Turn off Panoramic Rendering.
- Turn off Raytracing.
- Turn off SSS Subsurface Scattering.
- Turn off or lower oversampling/aliasing OSA.
- Turn off or lower Motion Blur.
- Render in Parts. This will also allow you to render **huge** images on a weak PC. On a multi-core PC, it will assign a thread to each part as well.
- Increase the octree resolution.
- Render at a percentage size of your final resolution (like 25%).
- Turn off *Fields* rendering.
- Use *Border* rendering to render a subset of the full image.

Bake Panel

- Bake Full Render creates a UV Texture that colors the objects based on materials, and then uses that UV Texture shadeless instead of the material.
- Bake Ambient Occlusion only.
- Bake textures for objects.
- Baking Normals or Displacement does not speed up render time, and are used for other things.

Format Panel

- Render at a lower resolution. Smaller pictures take less time to render.
- Choose a faster CODEC or CODEC settings.
- Render in black and white (*BW* button).
- If using `FFMPEG`, do not activate *Multiplex audio*.
- If using `FFMPEG`, *Autosplit Output* (*Video* panel button).
- Render only RGB if you just need color; the A channel (*RGBA* button) takes more memory and is unused when saving a movie file.

Multi-Pass Compositing

Another strategy that can be used to address the problem of long (re-)render times is to structure your workflow from the ground up so that you make aggressive use of *compositing*, as described in the "Post-Production" section. In this approach, you break down each shot into components that can be rendered separately, then you combine those separately-rendered elements to achieve the finished clip. For instance:

- If the camera is not moving, then neither is the background: only a single frame is needed. (The same is true of any non-moving object within the frame.) These individual elements, having been generated *once,* can be re-used as many times as necessary over as many frames as necessary.

- Both shadows and highlights can be captured separately from the objects that are being illuminated or shadowed, such that the intensity, color, and depth of the effect can be adjusted later without re-rendering.
- Start by using lights that do not cast shadows. (Shadow calculations are big time-killers.) Then, use "shadow-only" lights (which cast shadows, but do not cast light) to create shadows *only* where you judge that they are actually necessary. (It is very often the case that only a few of the shadows which could exist in the scene actually matter, and that the rest of them simply will not be noticed.)
- Tricky lighting situations can be avoided by handling the objects separately, then combining the individually-rendered clips and "tweaking" the result.

This is a very familiar idea. Modern sound recordings, for example, always use a "multi-track" approach. Individual components of the song are captured separately and in isolation, then the components are "mixed" together. The "final mix" then goes through additional processing stages, called *mastering*, to produce the finished product(s). (In fact, the features and design of modern sound-processing software are directly comparable to that of Blender's node-based compositor.)

There are compelling advantages to this approach:

- If something is "not quite right," you do not necessarily have to start over from scratch.
- In practice, the deadline-killer is *re-* rendering, which ordinarily must be done (in its entirety) just because "'one little thing' about the shot is wrong." Compositing helps to avoid this, because (ideally...) only the specific parts that are found to be in error must be repeated. (Or, maybe, the error can be blocked out with a "garbage matte" and a corrected version can be inserted in its place.
- Sometimes you might find that it is *almost* what you wanted, but now you would like to *add* this and maybe *take away* that." A compositing-based approach enables you to do just that, and furthermore, to do so *non-destructively.* In other words, having generated the "addition" (or the "mask") as a separate channel of information, you can now fine-tune its influence in the overall "mix", or even change your mind and remove it altogether, all without permanently altering anything.
- **By and large, these stages work *two-* dimensionally, manipulating what is by that time** "a raster bitmap with R, G, B, Alpha and Z-Depth information," so they are consistently fast.
- Since each discrete rendering task has been simplified, the computer can carry them out using much fewer resources.
- The tasks can be distributed among several different computers.
- "After all, the scene does not actually have to be *physically perfect,* to be *convincing*". A compositing-based approach lets you take full advantage of this. You can focus your attention (and Blender's) upon those specific aspects of the scene which will actually make a noticeable difference. It is possible to save a considerable amount of time by consciously choosing to exclude less-important aspects which (although "technically correct") probably will not be noticed.

Of course, this approach is not without its own set of trade-offs. You must devise a workable asset-management system for keeping track of exactly what material you have, where it is, whether it is up-to-date, and exactly how to re-create it. You must understand and use the "library linking" features of Blender to allow you to refer to objects, nodes, materials, textures and scenes in a carefully-organized collection of other files. You need to have a very clear notion, *in advance*, of exactly what the finished shot must consist of and what the task breakdown must be. You must be a scrupulous note-taker and record-keeper. But sometimes this is the best way, if not the *only* way, to accomplish a substantial production.

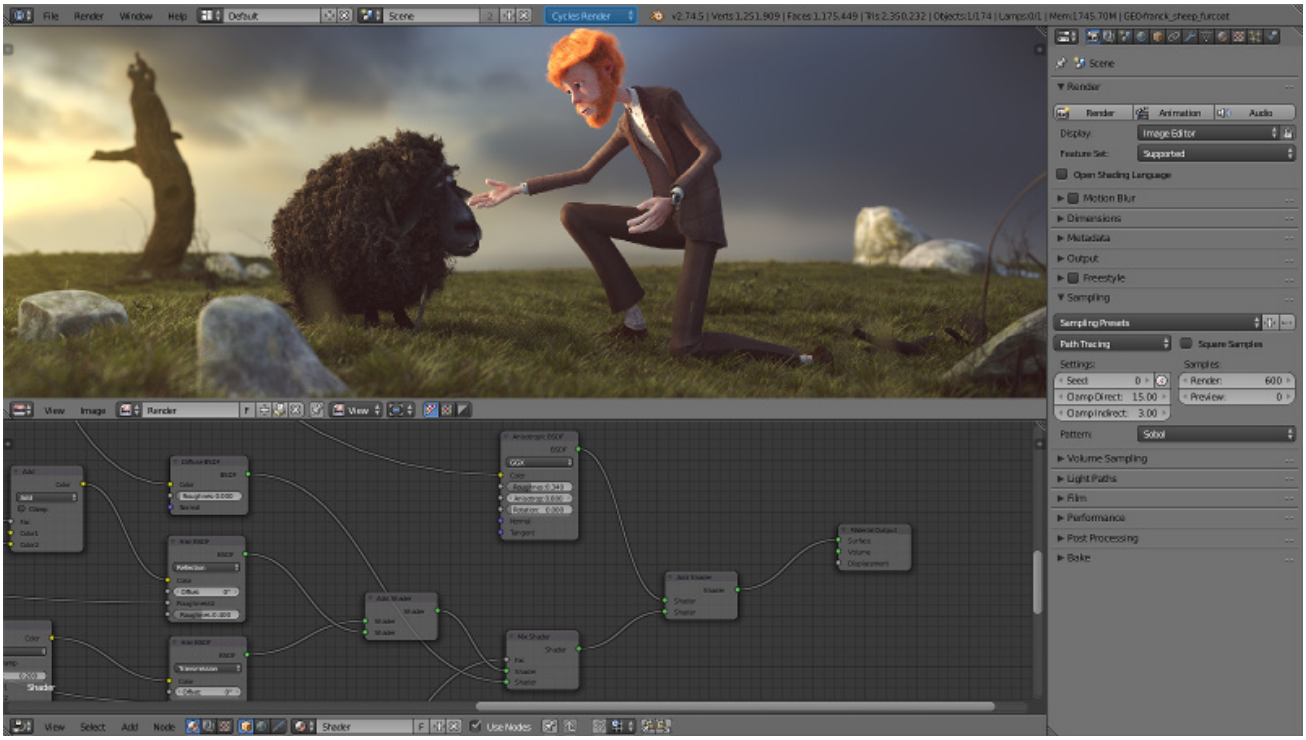
2.9.3 Cycles Render Engine

Introduction

Cycles is Blender's ray-tracing production render engine.

To use Cycles, it must be set as the active render engine in the top header. Once that is done, interactive rendering can be started by setting a 3D View editor to draw mode Rendered using `Shift-Z`. The render will keep updating as modifications are done, such as changing a material color, changing a lamp's intensity or moving objects around. To perform a full render go to *Properties* → *Render* here you can either choose to render a still image or an *Animation*.

Cycles may be able to use your GPU (Graphics Processing Unit, or Graphics Card) to render. To see if and how you can use your GPU for rendering, see the documentation on *GPU Rendering*.



Note: Cycles Outside of Blender

Since its release under a permissive open-source (Apache 2.0) license, it's also in use by other 3D tools, such as Poser and Rhino. Cycles can be used as part of Blender and as stand-alone, making it a flexible solution for ray-traced rendering.

See also:

- Blender.org's Cycles Gallery showing examples of what Cycles can render.
- Developer documentation is available as well.

Render Settings

Integrator

The integrator is the rendering algorithm used to compute the lighting. Cycles currently supports a path tracing integrator with direct light sampling. It works well for various lighting setups, but is not as suitable for caustics and some other complex lighting situations.

Rays are traced from the camera into the scene, bouncing around until they find a light source such as a lamp, an object emitting light, or the world background. To find lamps and surfaces emitting light, both indirect light sampling (letting the ray follow the surface BSDF) and direct light sampling (picking a light source and tracing a ray towards it) are used.

Sampling

Sample Method There are two integrator modes that can be used: *Path Tracing* and *Branched Path Tracing*.

Square Samples Square the amount samples.

Seed Seed value for integrator to get different noise patterns.

Animate Seed This button which can be found on the right side of the *Seed* value can be used to give different seed values. It is a good idea to enable this when making animation because in the real world each frame has a different noise pattern.

Clamp Direct This option limits the maximum intensity a sample from rays which have not yet bounced can contribute to a pixel. Setting this option to 0.0 disables clamping altogether. Lower have a greater affect (dimmer samples) on the resulting image than higher values.

Note: A common issue encountered with *Path Tracing* is the occurrence of "fireflies": improbable samples that contribute very high values to pixels. This option provides a way to limit that. However, note that as you clamp out such values, other bright lights/reflections will be dimmed as well.

Care must be taken when using this setting to find a balance between mitigating fireflies and losing intentionally bright parts. It is often useful to clamp indirect bounces separately, as they tend to cause more fireflies than direct bounces. See the *Clamp Indirect* setting.

Clamp Indirect The same as *Clamp Direct*, but for rays which have bounced multiple times.

Pattern Random sampling pattern used by the integrator.

Sobol Uses a Sobol pattern to decide the random sapling pattern used by the integrator. See Sobol sequence on Wikipedia for more information.

Correlated Multi-Jitter Uses a Correlated Multi-Jitter pattern to decide the random sapling pattern used by the integrator. See this Pixar paper for more information.

Path Tracing

The *Path Tracing* integrator is a pure path tracer; at each hit it will bounce light in one direction and pick one light to receive lighting from. This makes each individual sample faster to compute, but will typically require more samples to clean up the noise.

Render Samples Number of paths to trace for each pixel in the final render. As more samples are taken, the solution becomes less noisy and more accurate.

Preview Samples Number of samples for viewport rendering.

Branched Path Tracing

The *Branched Path Tracing* integrator is similar, but at the first hit it will split the path for different surface components and will take all lights into account for shading instead of just one. This makes each sample slower, but will reduce noise, especially in scenes dominated by direct or one-bounce lighting. To get the same number of diffuse samples as in the path tracing integrator, note that e.g. 250 path tracing samples = 10 AA samples x 25 diffuse samples. The Sampling panel shows this total number of samples.

AA Render Samples Number of samples to take for each pixel in the final render. More samples will improve antialiasing.

AA Preview Samples Number of samples for viewport rendering.

Diffuse Samples Number of diffuse bounce samples to take for each AA sample.

Glossy Samples Number of glossy bounce samples to take for each AA sample.

Transmission Samples Number of transmission bounce samples to take for each AA sample.

AO Samples Number of ambient occlusion samples to take for each AA sample.

Mesh Light Samples Number of mesh light samples to take for each AA sample.

Subsurface Samples Number of subsurface scattering samples to take for each AA sample.

Volume Samples Number of volume scattering samples to take for each AA sample.

Bounces

Max Bounces Maximum number of light bounces. For best quality, this should be set to the maximum. However, in practice, it may be good to set it to lower values for faster rendering. Setting it to maximum 0 bounces results in direct lighting only.

Min Bounces Minimum number of light bounces for each path, after which the integrator uses Russian Roulette to terminate paths that contribute less to the image. Setting this higher gives less noise, but may also increase render time considerably. For a low number of bounces, it is strongly recommended to set this equal to the maximum number of bounces.

Diffuse Bounces Maximum number of diffuse bounces.

Glossy Bounces Maximum number of glossy bounces.

Transmission Bounces Maximum number of transmission bounces.

Volume Bounces Maximum number of volume scattering bounces.

Transparency

Transparency Max Maximum number of transparency bounces.

Transparency Min Minimum number of transparency bounces, after which Russian Roulette termination is used.

Transparent Shadows For direct light sampling, use transparency of surfaces in between to produce shadows affected by transparency of those surfaces.

Tricks

Reflective Caustics While in principle path tracing supports rendering of caustics with a sufficient number of samples, in practice it may be inefficient to the point that there is just too much noise. This option can be unchecked, to disable reflective caustics.

Refractive Caustics The same as above, but for refractive caustics.

Filter Glossy When using a value higher than 0.0, this will blur glossy reflections after blurry bounces, to reduce noise at the cost of accuracy. 1.0 is a good starting value to tweak.

Some light paths have a low probability of being found while contributing much light to the pixel. As a result these light paths will be found in some pixels and not in others, causing fireflies. An example of such a difficult path might be a small light that is causing a small specular highlight on a sharp glossy material, which we are seeing through a rough glossy material. In fact in such a case we practically have a caustic.

With path tracing it is difficult to find the specular highlight, but if we increase the roughness on the material, the highlight gets bigger and softer, and so easier to find. Often this blurring will hardly be noticeable, because we are seeing it through a blurry material anyway, but there are also cases where this will lead to a loss of detail in lighting.

See also:

See *Reducing Noise* for examples of the clamp settings in use.

Geometry

Volume Sampling

Step Size Distance between volume shader samples when rendering the volume. Lower values give more accurate and detailed results but also increased render time.

Max Steps Maximum number of steps through the volume before giving up, to protect from extremely long render times with big objects or small step sizes.

Subdivision Rate

These settings are used to control the *True Displacement*.

Note: These Options are only available if *Experimental Feature Set* is turned on.

Render Size of *micropolygons* in pixels.

Preview Size of *micropolygons* in pixels while preview rendering.

Max Subdivisions Stop subdividing when this level is reached even if the dice rate would produce finer *tessellation*.

Light Paths

Ray Types

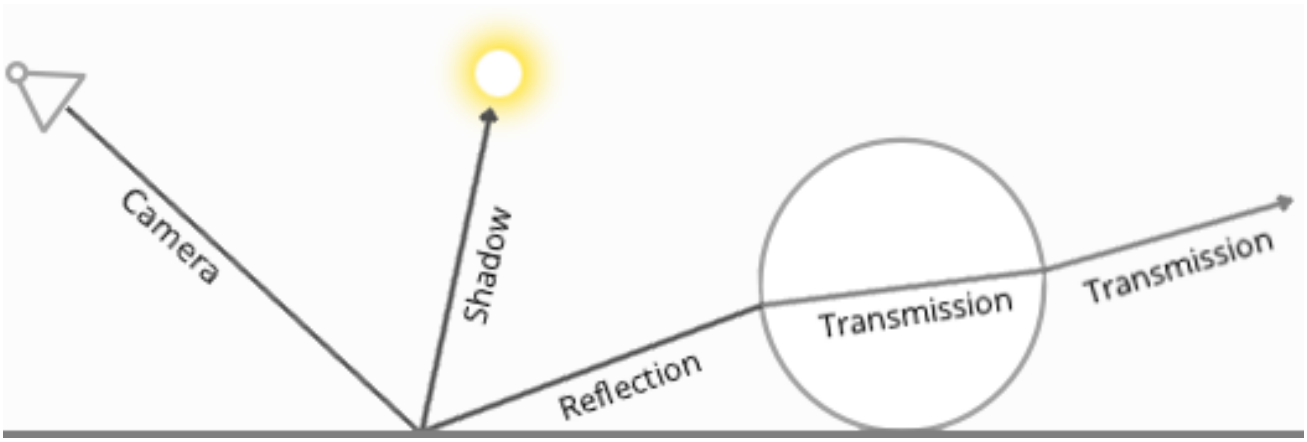
Ray types can be divided into four categories:

- Camera: the ray comes straight from the camera.
- Reflection: the ray is generated by a reflection off a surface.
- Transmission: the ray is generated by a transmission through a surface.
- Shadow: the ray is used for (transparent) shadows.

Reflection and transmission rays can further have these properties:

- Diffuse: the ray is generated by a diffuse reflection or transmission (translucency).
- Glossy: the ray is generated by a glossy specular reflection or transmission.
- Singular: the ray is generated by a perfectly sharp reflection or transmission.

The Light Path node can be used to find out the type of ray the shading is being computed for.



Bounce Control

The maximum number of light bounces can be controlled manually. While ideally this should be infinite, in practice a smaller number of bounces may be sufficient, or some light interactions may be intentionally left out for faster convergence. The number of diffuse reflection, glossy reflection and transmission bounces can also be controlled individually.

Light paths are terminated probabilistically when specifying a minimum number of light bounces lower than the maximum. In that case paths longer than minimum will be randomly stopped when they are expected to contribute less light to the image. This will still converge to the same image, but renders faster while possibly being noisier.

A common source of noise is caustics, which are diffuse bounces followed by a glossy bounce (assuming we start from the camera). An option is available to disable these entirely.

Transparency

The transparent BSDF (Bidirectional scattering distribution function) shader is given special treatment. When a ray passes through it, light passes straight on, as if there was no geometry there. The ray type does not change when passing through a transparent BSDF.

Alpha pass output is also different for the transparent BSDF. Other transmission BSDF s are considered opaque, because they change the light direction. As such they cannot be used for alpha-over compositing, while this is possible with the transparent BSDF.

The maximum number of transparent bounces is controlled separately from other bounces. It is also possible to use probabilistic termination of transparent bounces, which might help rendering many layers of transparency.

Note that while semantically the ray passes through as if no geometry was hit, rendering performance is affected as each transparency step requires executing the shader and tracing a ray.

Ray Visibility

Objects can be set to be invisible to particular ray types:

- Camera
- Diffuse reflection
- Glossy reflection
- Transmission
- Shadow

Properties Editor *Object* → *Cycles Settings* → *Ray visibility*.

This can be used, for example, to make an emitting mesh invisible to camera rays. For duplicators, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

In terms of performance, using these options is more efficient that using a shader node setup that achieves the same effect. Objects invisible to a certain ray will be skipped in ray traversal already, leading to fewer rays cast and shaders executed.

Render Layers and Passes

Layers

This section covers only the Render Layer settings appropriate for the Blender Render engine. For the engine-independent settings, see *this section*.

Exclude Scene layers are shared between all render layers; however, sometimes it is useful to leave out some object influence for a particular render layer. That is what this option allows you to do.

Passes

Lighting Passes

Diffuse Direct Direct lighting from diffuse BSDFs. We define direct lighting as coming from lamps, emitting surfaces, the background, or ambient occlusion after a single reflection or transmission off a surface. BSDF color is not included in this pass.

Diffuse Indirect Indirect lighting from diffuse BSDFs. We define indirect lighting as coming from lamps, emitting surfaces or the background after more than one reflection or transmission off a surface. BSDF color is not included in this pass.

Diffuse Color Color weights of diffuse BSDFs. These weights are the color input socket for BSDF nodes, modified by any Mix and Add Shader nodes.

Glossy Direct, Indirect, Color Same as above, but for glossy BSDFs.

Transmission Direct, Indirect, Color Same as above, but for transmission BSDFs.

Subsurface Direct, Indirect, Color Same as above, but for subsurface BSDFs.

Emission Emission from directly visible surfaces.

Environment Emission from the directly visible background. When the film is set to transparent, this can be used to get the environment color and composite it back in.

Shadow Shadows from lamp objects.

Ambient Occlusion Ambient occlusion from directly visible surfaces. BSDF color or AO factor is not included; i.e. it gives a 'normalized' value between 0 and 1.

Note: *Transparent BSDFs are given special treatment*. A fully transparent surface is treated as if there is no surface there at all; a partially transparent surface is treated as if only part of the light rays can pass through. This means it is not included in the Transmission passes; for that a glass BSDF with index of refraction 1.0 can be used.

Combining

All these lighting passes can be combined to produce the final image as follows:

Data Passes

Combined The final combination of render passes with everything included.

Z Distance in *BU* to any visible surfaces.

> **Note:** The Z pass only uses one sample. When depth values need to be blended in case of motion blur or *DOF*, use the mist pass.

Mist Distance to visible surfaces, mapped to the 0.0-1.0 range. When enabled, settings are in *Properties → World → Mist Pass*.

Normal Surface normal used for shading.

Vector Motion vectors for the vector blur node. The four components consist of 2D vectors giving the motion towards the next and previous frame position in pixel space.

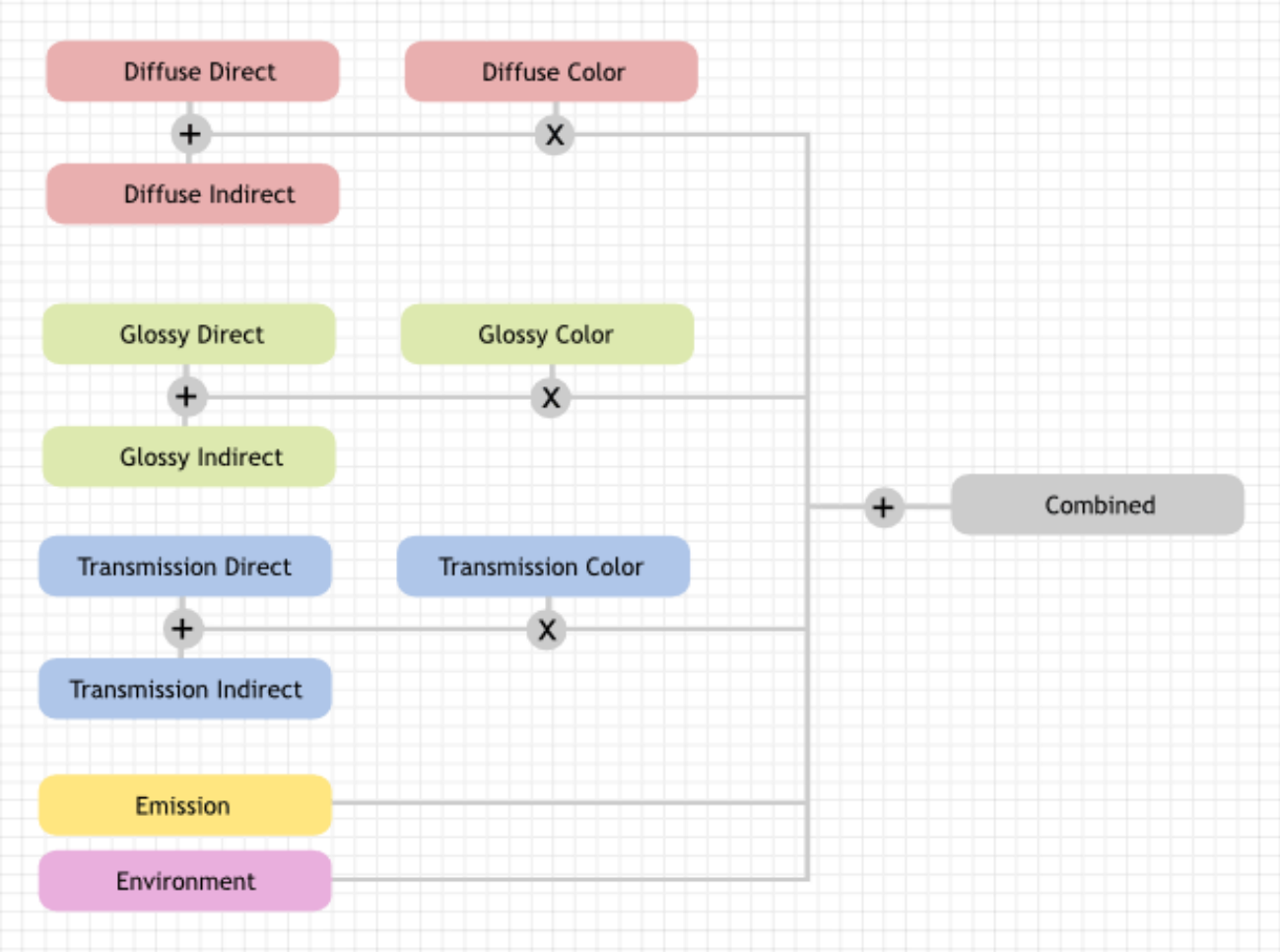
UV Default render UV coordinates.

Object Index Creates a mask of the object that can be later read by the *ID Mask Node* in the compositor.

Material Index Creates a mask of the material that can be later read by the *ID Mask Node* in the compositor.

Note: The Z, Object Index and Material Index passes are not anti-aliased.

Alpha Threshold Z, Index, normal, UV and vector passes are only affected by surfaces with alpha transparency equal to or higher than this threshold. With value 0.0 the first surface hit will always write to these passes, regardless of transparency. With higher values surfaces that are mostly transparent can be skipped until an opaque surface is encountered.


Diffuse Direct
Diffuse Color
+
X
Diffuse Indirect
Glossy Direct
Glossy Color
+
X
Glossy Indirect
+
Combined
Transmission Direct
Transmission Color
+
X
Transmission Indirect
Emission
Environment

Motion Blur

Blender's animations are by default rendered as a sequence of *perfectly still* images. While great for stop-motion and time-lapses, this is unrealistic, since fast-moving objects do appear to be blurred in the direction of motion, both in a movie frame and in a photograph from a real-world camera.

Fig. 2.1920: Cycles Motion Blur Example.

Note: If there are particles or other physics system in a scene, be sure to bake them before rendering, otherwise you might not get correct or consistent motion.

Options

Position Controls at what point the shutter opens in relation to the frame.

- End on frame
- Center on frame
- Start on frame

Shutter Speed Time between frames over which motion blur is computed. Shutter time 1.0 blurs over the length of 1 frame, 2.0 over the length of two frames, from the previous to the next.

Shutter Curve Curve defining how the shutter opens and closes.

Shutter Type Replicates CMOS cameras by rendering a rolling shutter effect using scanlines.

- Top Bottom: Renders rolling shutter from the top of the image from the bottom.

Rolling Shutter Duration Controls balance between pure rolling shutter effect and pure motion blur effect. With zero being no rolling shutter and one being all rolling shutter.

Warning: An object modifier setup that changes mesh topology over time will cause severe problems.

Common examples of this are animated booleans, deformation before edge-split, remesh, skin or decimate modifiers.

Fig. 2.1921: Cycles Motion Blur Settings.

Object Properties

Motion blur settings can also be applied per object in the *Object Properties* tab of the Properties editor.

Deformation Use deformation motion blur for the object.

Steps Controls accuracy of deformation motion blur, more steps gives more memory usage. The actual number of steps is $2^{steps-1}$.

Object Settings

These are options that are scattered though out Blender, and are often only available in certain contexts.

Adaptive Subdivision

Note: Implementation not finished yet, marked as an *Experimental Feature Set*.

When using the Experimental Feature Set the *Subdivision Surface Modifier* gets changed to control the subdivision of a mesh at the time of rendering. For this, all the other settings are the same except the *View* and *Render* settings. These previously mentioned settings get removed/renamed and the following settings are added:

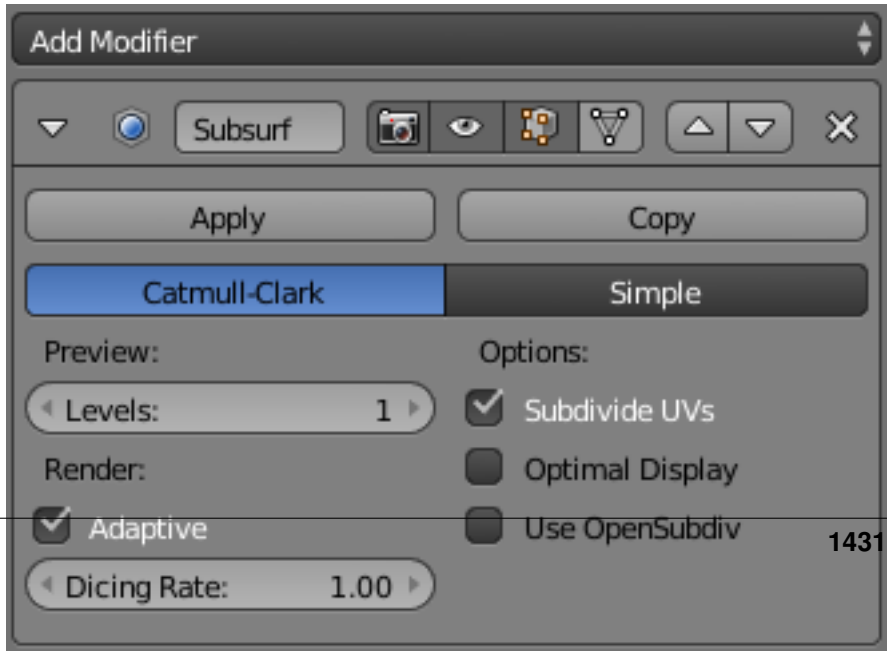
None

Preview

Levels The levels of subdivision to see in the 3D View, this works the same as the *View* setting on the original *Subdivision Modifier*.

Render

Adaptive Use OpenSubdiv to give different subdivision levels to near and far objects automatically. This allows nearer object to get more subdivisions and far objects to get less.



Dicing Rate When using *Adaptive* the *Render Levels* property gets changed to *Dicing Rate*, this property is used to multiply the *scene dicing rate*.

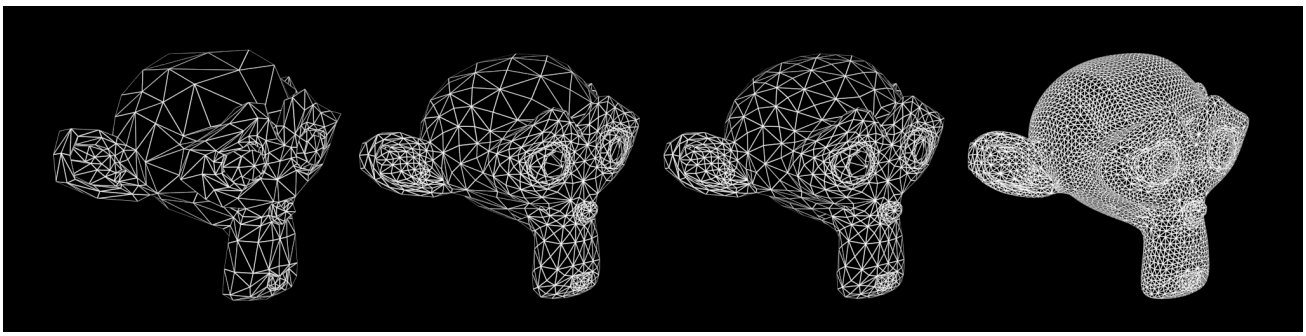


Fig. 2.1923: Subdivision Off/On, Dicing Rate: 1.0 - 0.3 - 0.05 (Monkeys look identical in viewport, no modifiers).

Levels The levels of subdivision to see in the final render, this works the same as the *Render* setting on the original *Subdivision Modifier*.

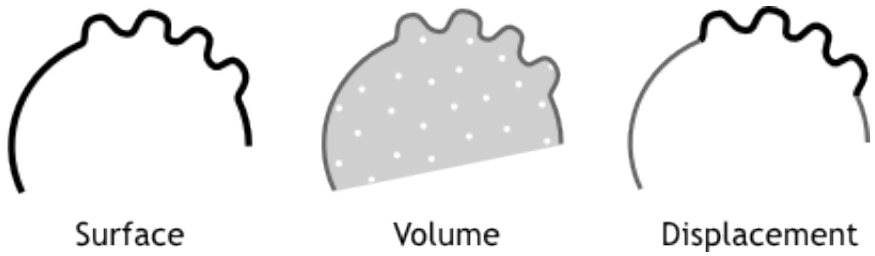
Known limitations

- Missing support for UV subdivision.
- Creases do not match Blender creases currently.
- Instanced are currently uninstanced, leading to increased memory usage. For those it is better to use non-adaptive subdivision still.
- Multi-view renders can have some inconsistencies between views.
- Editing displacement shaders while using *True Displacement* does not update the viewport.

Materials

Introduction

Materials define the appearance of meshes, curves and other objects. They consist of three shaders, defining the appearance of the surface of the mesh, the volume inside the mesh, and displacement of the surface of the mesh.



Surface Shader

The surface shader defines the light interaction at the surface of the mesh.

See also:

Surface Shader.

Volume Shader

When the surface shader does not reflect or absorb light, it enters into the volume. If no volume shader is specified, it will pass straight through to the other side of the mesh.

If it is defined, a volume shader describes the light interaction as it passes through the volume of the mesh. Light may be scattered, absorbed, or emitted at any point in the volume.

A material may have both a surface and a volume shader, or only one of either. Using both may be useful for materials such as glass, water or ice, where you want some of the light to be absorbed as it passes through the surface, combined with e.g. a glass or glossy shader at the surface.

See also:

Volume Shader.

Displacement

The shape of the surface and the volume inside it may be altered by displacement shaders. This way, textures can then be used to make the mesh surface more detailed.

Depending on the settings, the displacement may be virtual, only modifying the surface normals to give the impression of displacement, which is

known as bump mapping, or a combination of real and virtual displacement.

See also:

Displacement.

Energy Conservation

The material system is built with physics-based rendering in mind, cleanly separating how a material looks and which rendering algorithm is used to render it. This makes it easier to achieve realistic results and balanced lighting, though there are a few things to keep in mind.

In order for materials to work well with global illumination, they should be, speaking in terms of physics, energy conserving. That means they cannot reflect more light than comes in. This property is not strictly enforced, but if colors are in the range 0.0 to 1.0, and BSDF s are only mixed together with the Mix Shader node, this will automatically be true.

It is however, possible to break this, with color values higher than 1.0 or using the Add Shader node, but one must be careful when doing this to keep materials behaving predictably under various lighting conditions. It can result in a reflection adding light into the system at each bounce, turning a BSDF into a kind of emitter.

Surface

The surface shader defines the light interaction at the surface of the mesh. One or more BSDF's specify if incoming light is reflected back, refracted into the mesh, or absorbed.

Emission defines how light is emitted from the surface, allowing any surface to become a light source.

Terminology

BSDF stands for bidirectional scattering distribution function. It defines how light is reflected and refracted at a surface.

Reflection BSDF s Reflect an incoming ray on the same side of the surface.

Transmission BSDF s Transmit an incoming ray through the surface, leaving on the other side.

Refraction BSDF s are a type of *Transmission*, Transmitting an incoming ray and changing its direction as it exits on the other side of the surface.

BSDF Parameters

A major difference from non-physically based renderers is that direct light reflection from lamps and indirect light reflection of other surfaces are not decoupled, but rather handled using a single BSDF. This limits the possibilities a bit, but we believe overall it is helpful in creating consistent-looking renders with fewer parameters to tune.

For glossy BSDF s, the *roughness* parameter controls the sharpness of the reflection, from 0.0 (perfectly sharp) to 1.0 (very soft). Compared to *hardness* or *exponent* parameters, it has the advantage of being in the range 0.0..1.0, and as a result gives more linear control and is more easily textureable. The relation is roughly: *roughness* = 1 - 1/*hardness*

Volume

Volume rendering can be used to render effects like fire, smoke, mist, absorption in glass, and many other effects that cannot be represented by surface meshes alone.

To set up a volume, you create a mesh that defines the bounds within which the volume exists. In the material you typically remove the surface nodes and instead connect volume nodes to define the shading inside the volume. For effects such as absorption in glass you can use both a surface and volume shader. The world can also use a volume shader to create effects such as mist.

Volume Shaders

Cycles supports three volume shader nodes, that model particular effects as light passes through the volume and interacts with it:

- Volume Absorption will absorb part of the light as it passes through the volume. This can be used to shade for example black smoke or colored glass objects, or mixed with the volume scatter node. This node is somewhat similar to the transparent BSDF node, it blocks part of the light and lets other light pass straight through.
- Volume Scatter lets light scatter in other directions as it hits particles in the volume. The anisotropy defines in which direction the light is more likely to scatter. A value of 0 will let light scatter evenly in all directions (somewhat similar to the diffuse BSDF node), negative values let light scatter mostly backwards, and positive values let light scatter mostly forward. This can be used to shade white smoke or clouds for example.
- Emission will emit light from the volume. This can be used to shade fire for example.

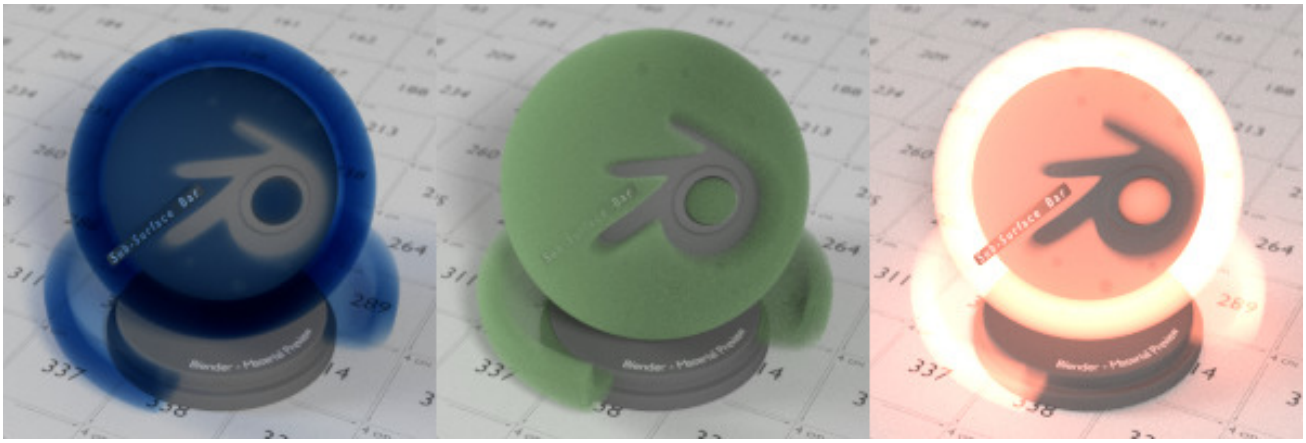


Fig. 2.1924: Volume Shader: Absorption/Absorption + Scatter/Emission.

Density

All volume shaders have a density input. The density defines how much of the light will interact with the volume, getting absorbed or scattered, and how much will pass straight

through. For effects such as smoke you would specify a density field to indicate where in the volume there is smoke and how much (density bigger than 0), and where there is no smoke (density equals 0).

Volumes in real life consist of particles, a higher density means there are more particles per unit volume. More particles means there is a higher chance for light to collide with a particle and get absorbed or scattered, rather than passing straight through.

Volume Material

Interaction with the Surface Shader

A material may have both a surface and a volume shader, or only one of either. Using both may be useful for materials such as glass, water or ice, where you want some of the light to be absorbed as it passes through the surface, combined with e.g. a glass or glossy shader at the surface.

When the surface shader does not reflect or absorb light, it enters into the volume. If no volume shader is specified, it will pass straight through to the other side of the mesh. If it is defined, a volume shader describes the light interaction as it passes through the volume of the mesh. Light may be scattered, absorbed, or emitted at any point in the volume.

Mesh Topology

Meshes used for volume render should be closed and *manifold*. That means that there should be no holes in the mesh. Each edge must be connected to exactly two faces such that there are no holes or T-shaped faces where three or more faces are connected to an edge.

Normals must point outside for correct results. The normals are used to determine if a ray enters or exits a volume, and if they point in a wrong direction, or there is a hole in the mesh, then the renderer is unable

to decide what is the inside or outside of the volume.

These rules are the same as for rendering glass refraction correctly.

Volume World

A volume shader can also be applied to the entirely world, filling the entire space.

Currently, this is most useful for night time or other dark scenes, as the world surface shader or sun lamps will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However, for modeling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Smoke

Creating a smoke material for cycles can be difficult however the image below shows a good setup on how to do this.

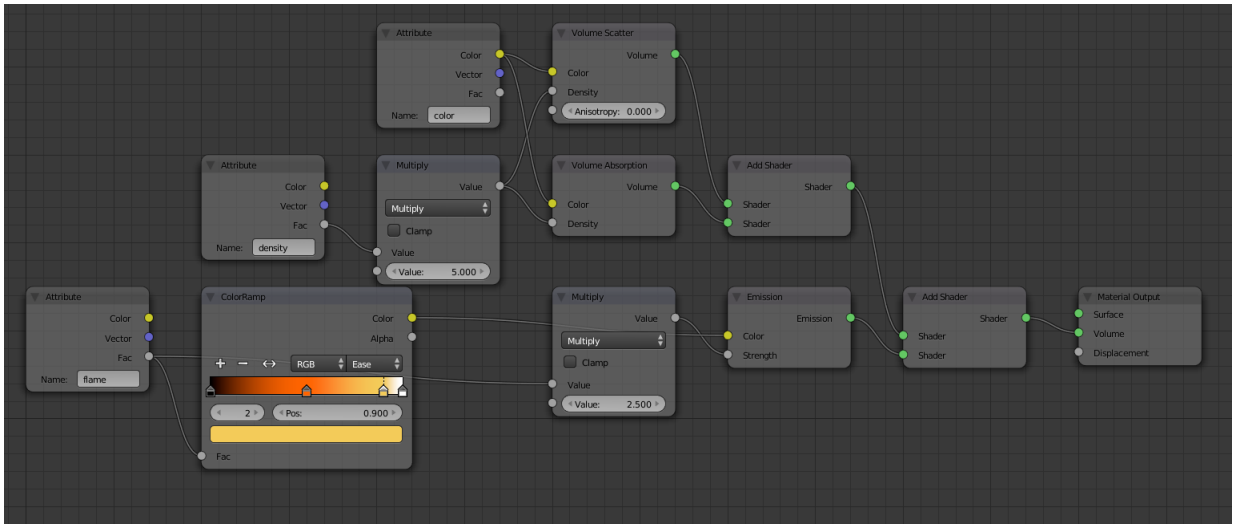


Fig. 2.1925: Smoke and Fire Material.

Scattering Bounces

Real world effects such as scattering in clouds or subsurface scattering require many scattering bounces. However, unbiased rendering of such effects is slow and noisy. In typical movie production scenes only 0 or 1 bounces might be used to keep render times under control. The effect you get when rendering with zero volume bounces is what is known as "single scattering", the effect from more bounces is "multiple scattering".

For rendering materials like skin or milk, the subsurface scattering shader is an approximation of such multiple scattering effects that is significantly more efficient but not as accurate.

For materials such as clouds or smoke that do not have a well defined surface, volume rendering is required. These look best with many scattering bounces, but in practice one might have to limit the number of bounces to keep render times acceptable.

Limitations

Currently, the following are not not support:

- Correct ray visibility for volume meshes

Not available on GPU:

- Equi Angular/MIS Volume Sampling
- Volume Multi Light sampling

Displacement

The shape of the surface and the volume inside its mesh may be altered by the displacement shaders. This way, textures can then be used to make the mesh surface more detailed.

There are two types of displacement methods that can be used: *True Displacement* and *Bump Mapping*. Depending on the settings, the displacement may be virtual, only modifying

the surface normals to give the impression of displacement, known as bump mapping, or a combination of real and virtual displacement.

Tip: It is also possible to use the both method by choosing *Displacement + Bump* in the *Material Setttings*.

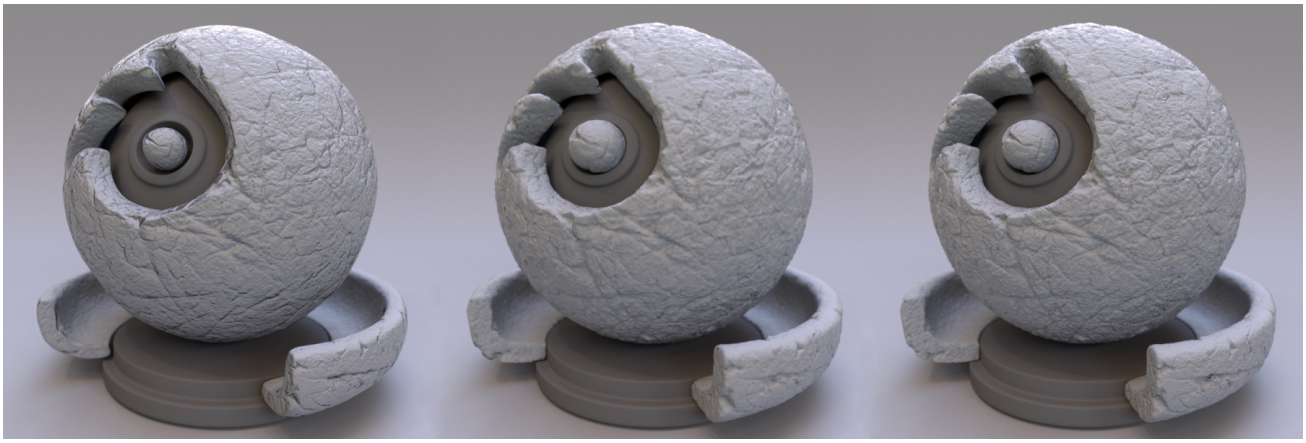


Fig. 2.1926: Subdivision Rate 2, Bump, True, Both

Bump Mapping

When using the *Bump* method for displacement a "bump map" is used to create fake displacement by using light and shadow effects. A bump map is actually one of the older types displacement methods (see *True Displacement* for a newer method).

Typically, bump maps are grayscale images with 8-bits of color information. This means that they only have 256 different shades of black, gray, or white. These grayscale values are used to tell Blender two thing: up or down.

When values in a bump map are close to 50% gray, there is little to no detail that comes through on the surface. When values get closer to white, the effect start to appear as if they are pulling out the surface. To contrast that, when values closer to black, they appear to be pushing into the surface.

Bump maps are really great for creating tiny details on a model, for example, pores or wrinkles on skin. Bump maps can be created in a 2D drawing, or photo editing application

just remember to save the image as a greyscale to save memory while rendering.

Warning: Because bump mapping is a fake effect, it is easily broken when viewing a model at the wrong angle. This means that it is not recommended for animations.

True Displacement

Note: Implementation not finished yet, marked as an *Experimental Feature Set*

Different from bump mapping, *True Displacement* is not a fake effect. When using *True Displacement* the actual mesh geometry will be displaced before render. This gives the best quality results, if the mesh is finely subdivided. As a result this method is also the most memory intensive.

When using true displacement you should not just use a bump map as the displacement texture. Different from bump maps displacement maps should not use 8-bits when saved. While you can use 8-bit textures, they do not translate into 3D space well. Instead, you should save the images with either 16 or 32-bits.

Tip: In order to get the appropriate amount of subdivision it is recommended to use *Adaptive Subdivision*

See also:

The *Displace Modifier* can also be used to displace a mesh.

Controls

You may find that there is a limit to using *True Displacement* compared to using the *Displace Modifier*. However, These can be easy fixed with using a *Math Node*. In the example below is a node setup to give the same settings as the *Displace Modifier*.

In the example above a math node is used twice, the first math node uses the add operator. This operation can be used to control the mid-level of the displacement. The second math node uses the multiply operation to

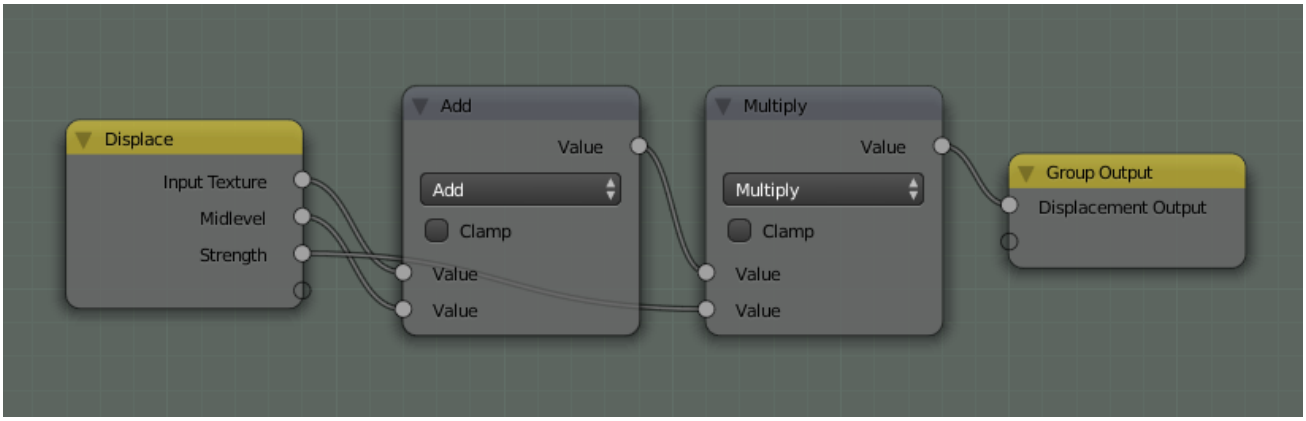


Fig. 2.1927: Math nodes used to add Mid-level and Strength.

control how strong the displacement effect is. Higher values would give you larger displacement and lower values give smaller displacement.

Material Settings

Surface

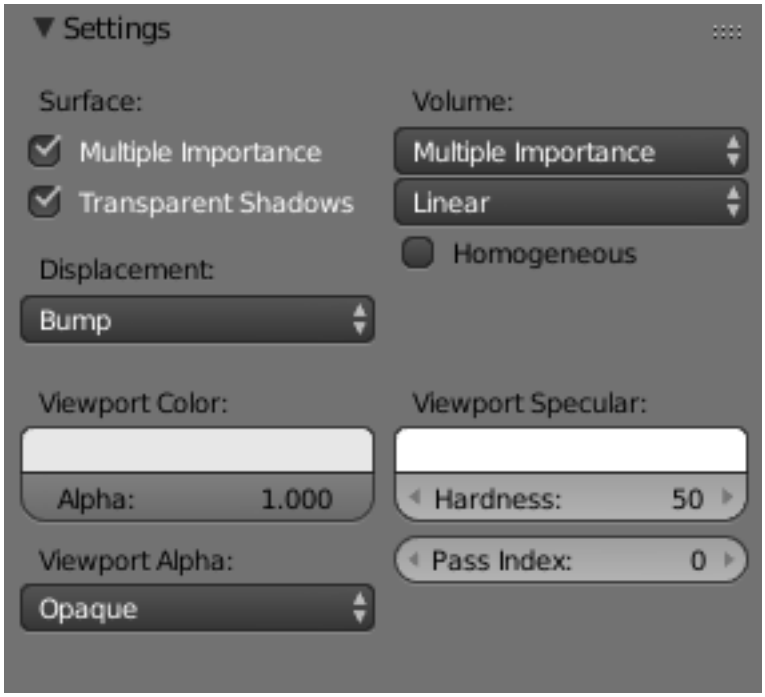


Fig. 2.1928: Material Settings.

Multiple Importance Sample By default objects with emitting materials use both direct and indirect light sampling methods, but in some cases it may lead to less noise overall to disable direct light sampling for some materials. This can be done by disabling the *Multiple Importance Sample* option. This is especially useful on large objects that emit little light compared to other light sources.

This option will only have an influence if the material contains an emission node; it will be automatically disabled otherwise.

Transparent Shadows Use transparent shadows if it contains a *Transparent BSDF*, disabling will render faster but will not give accurate shadows.

Volume

Sampling Method Options are *Multiple Importance*, *Distance*, or *Equiangular*. If you have got a pretty dense volume that is lit from far away then distance sampling is usually more efficient. If you have got a light inside or near the volume then equiangular sampling is better. If you have a combination of both, then the multiple importance sampling will be better.

Interpolation Controls the type of interpolation to use for smoke simulations.

Linear Good smoothness and speed.

Cubic Smoothed high quality interpolation, but slower.

Homogeneous Volume Assume volume has the same density everywhere (not using any textures), for faster rendering. For example absorption in a glass object would typically not have any textures, and by knowing this we can avoid taking small steps to sample the volume shader.

Displacement

Note: These Options are only available if *Experimental Feature Set* is turned on.

Displacement Method Method used preform *Displacement* on materials.

True Displacement Mesh vertices will be displaced before rendering, modifying the actual mesh. This gives the best quality results, if the mesh is finely subdivided. As a result, this method is also the most memory intensive.

Bump Mapping When executing the surface shader, a modified surface normal is used instead of the true normal. This is a quick alternative to true displacement, but only an approximation. Surface silhouettes will not be accurate and there will be no self-shadowing of the displacement.

Displacement + Bump Both methods can be combined, to do displacement on a coarser mesh, and use bump mapping for the final detail.

Viewport Settings

Viewport Color

Color TODO.

Alpha TODO.

Viewport Specular

Color TODO.

Hardness TODO.

Viewport Alpha

Blend Mode *Blend modes* for transparent faces.

Opaque Render color of textured face as color.

Add Render transparent and add color of face.

Alpha Clip Use the image alpha values clipped with no blending (binary alpha).

Alpha Blend Render polygon transparent, depending on alpha channel of the texture.

Alpha Sort Sort faces for correct alpha drawing (slow, use *Alpha Clip* instead when possible).

Alpha Anti-Aliasing Use texture alpha ad an anti-aliasing mask, requires multi-sample OpenGL display.

Pass Index

Pass Index Index number for the *Material Index render pass*. This can be used to give a mask to a material and then be read with the *ID Mask Node* in the compositor.

Texture Editing

3D View draw types, UV mapping, and texture painting work somewhat differently when Cycles is enabled. UV Maps no longer get image textures assigned themselves; rather they must always be assigned by adding an image texture node to a material.

3D View Draw Types

The Texture draw types used for Blender Internal have been replaced by three others in Cycles:

Texture This draw mode is used for editing, painting and mapping individual textures. Lighting is the same as in solid mode, so this is similar to the existing textured solid for Blender Internal. The texture drawn is the active image texture node for the material.

Material A simplified version of the entire material is drawn using GLSL shaders. This uses solid lighting, and also is mostly useful for editing, painting and mapping textures, but while seeing how they integrate with the material.

Rendered In this draw mode the render engine does the drawing, interactively refining the full rendered image by taking more samples. Unlike offline rendering, objects still use the viewport rather than render resolution and visibility.

Texture Properties

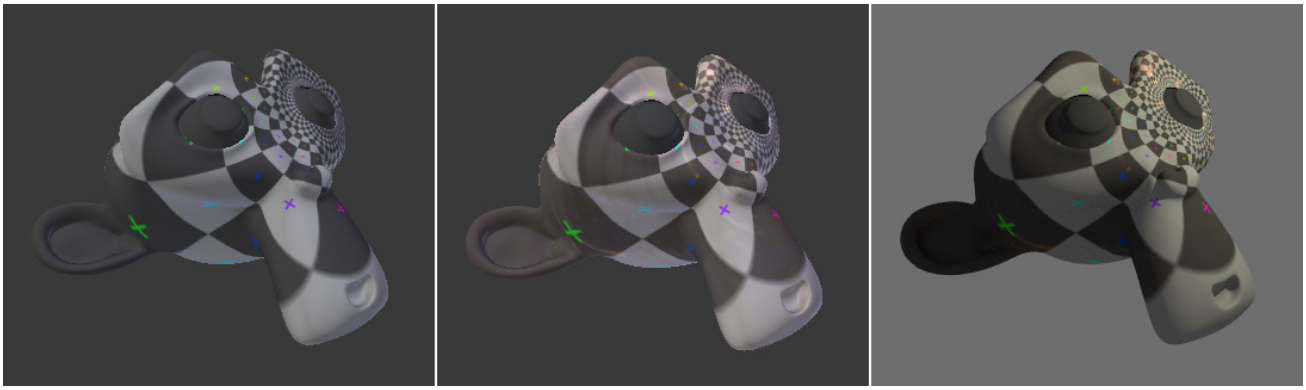
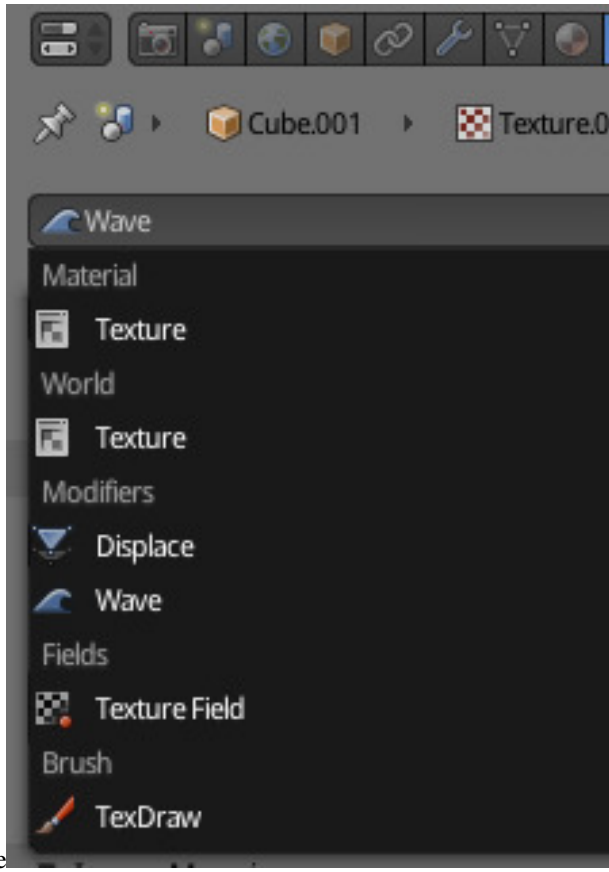


Fig. 2.1929: Material draw modes (Texture, Material, Rendered).

In the texture properties, the texture can now be selected from a list that contains all texture nodes from the world, lamps and materials, but also from e.g. modifiers, brushes and physics fields.

For shading nodes, the available textures are Cycles textures. For others, Blender textures are still used, but this will change in the future.



None

Painting & UV Editing

For texture paint mode, the image that is painted on is taken from the active image texture node. This can be selected in the node editor or the texture properties, and it is indicated as blue in the material properties.

For UV mapping, the active UV map as specified in the mesh properties is used. Assigning images in the UV/Image editor also affects the active image texture node.

Nodes

Introduction

Materials, lights and backgrounds are all defined using a network of shading nodes. These nodes output values, vectors, colors and shaders.

Shaders

An important concept to understand when building node setups is that of the *shader socket*. The output of all surface and volume shaders is a shader, describing lighting interaction at the surface or of the volume, rather than the color of the surface.

There are a few types of shaders available as nodes:

BSDF shader Describe light reflection, refraction and absorption at an object surface.

Emission shader Describe light emission at an object surface or in a volume.

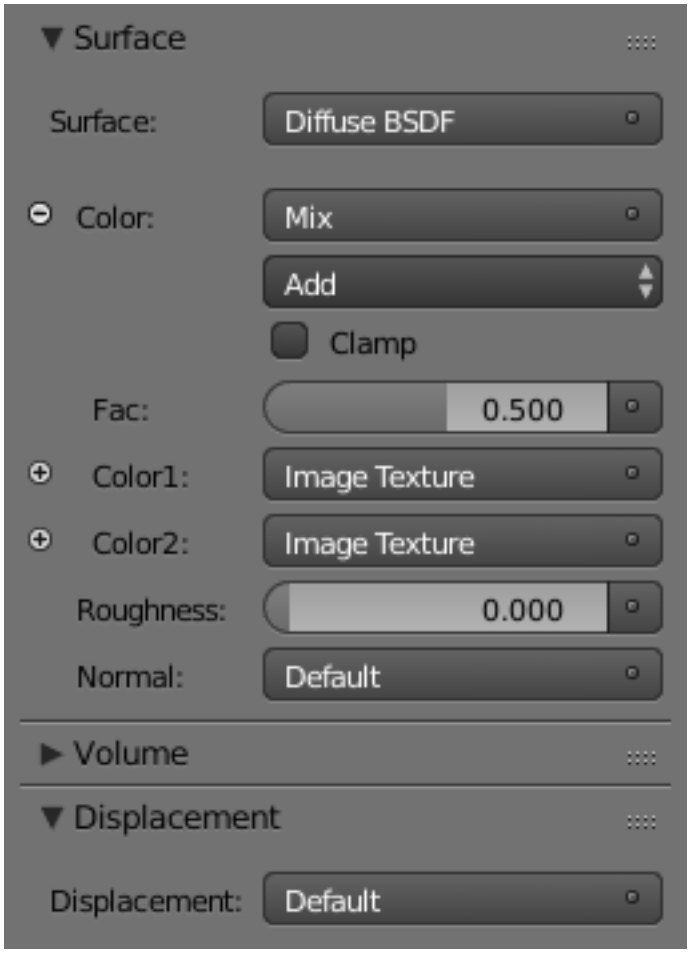
Volume shader Describe light scattering inside a volume.

Background shader Describe light emission from the environment.

Each shader node has a color input, and outputs a shader. These can then be mixed and added together using Mix and Add Shader nodes. No other operations are permitted. The resulting output can then be used by the render engine to compute all light interactions, for direct lighting or global illumination.

See also:

Shaders



Textures

Each texture type in Cycles corresponds to a node, with a texture coordinate and various parameters as input, and a color or value as output. No texture data-blocks are needed; instead node groups can be used for reusing texture setups.

For UV mapping and texture painting in the viewport, the Image texture node must be used. When setting such a node as active, it will be drawn in Textured draw mode, and can be painted on in texture paint mode.

The default texture coordinates for all nodes are Generated coordinates, with the exception of Image textures that use UV coordinates by default. Each node includes some options to modify the texture mapping and resulting color, and these can be edited in the texture properties.

See also:

Textures.

More

Nodes for geometric data, texture coordinates, layering shaders and non-physically based tricks can be found in:

- *Vector Nodes*,
- *Color Nodes*,
- *Converter Nodes*

Open Shading Language

Custom nodes can be written using the Open Shading Language.

See also:

Open Shading Language.

Open Shading Language

It is also possible to create your own nodes using Open Shading Language (OSL). Note that these nodes will only work for CPU rendering; there is no support for running OSL code on the GPU.

To enable it, select *Open Shading Language* as the shading system in the render settings.

Note: On Linux, C/C++ compiler tools (in particular /usr/bin/cpp) must be installed to compile OSL scripts.

Script Node

OSL was designed for node-based shading, and *each* OSL shader corresponds to *one* node in a node setup. To add an OSL shader, add a script node and link it to a text data-block or an external file. Input and output sockets will be created from the shader parameters on clicking the update button in the node or the text editor.

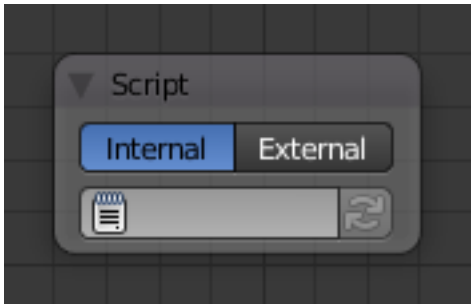


Fig. 2.1930: Script Node.

OSL shaders can be linked to the node in a few different ways. With the *Internal* mode, a text data-block is used to store the OSL shader, and the OSO bytecode is stored in the node itself. This is useful for distributing a blend-file with everything packed into it.

The *External* mode can be used to specify a `.osl` file on disk, and this will then be automatically compiled into a `.oso` file in the same directory. It is also possible to specify a path to a `.oso` file, which will then be used directly, with compilation done manually by the user. The third option is to specify just the module name, which will be looked up in the shader search path.

The shader search path is located in the same place as the scripts or configuration path, under:

Linux

`$HOME/.config/blender/2.78/shaders/`

MS-Windows

`C:\Users\$user\AppData\Roaming\Blender Foundation\Blender\2.78\shaders\`

macOS

`/Users/$USER/Library/Application Support/Blender/2.78/shaders/`

Tip: For use in production, we suggest to use a node group to wrap shader script nodes, and link that into other blend-files. This makes it easier to make changes to the node afterwards as sockets are added or removed, without having to update the script nodes in all files.

Writing Shaders

For more details on how to write shaders, see the OSL specification. Here is a simple example:

```
shader simple_material(
    color Diffuse_Color = color(0.6, 0.8, 0.6),
    float Noise_Factor = 0.5,
    output closure color BSDF = diffuse(N))
{
    color material_color = Diffuse_Color * mix(1.0, noise(P * 10.0), Noise_Factor);
    BSDF = material_color * diffuse(N);
}
```

Closures

OSL is different from, for example, RSL or GLSL, in that it does not have a light loop. There is no access to lights in the scene, and the material must be built from closures that are implemented in the render engine itself. This is more limited, but also makes it possible for the render engine to do optimizations and ensure all shaders can be importance sampled.

The available closures in Cycles correspond to the shader nodes and their sockets; for more details on what they do and the meaning of the parameters, see the *shader nodes manual*.

BSDF

- `diffuse(N)`
- `oren_nayar(N,roughness)`
- `diffuse_ramp(N,colors[8])`
- `phong_ramp(N,exponent,colors[8])`
- `diffuse_toon(N,size,smooth)`
- `glossy_toon(N,size,smooth)`
- `translucent(N)`
- `reflection(N)`
- `refraction(N,ior)`
- `transparent()`
- `microfacet_ggx(N,roughness)`
- `microfacet_ggx_aniso(N,T,ax,ay)`
- `microfacet_ggx_refraction(N,roughness,ior)`
- `microfacet_beckmann(N,roughness)`
- `microfacet_beckmann_aniso(N,T,ax,ay)`
- `microfacet_beckmann_refraction(N,roughness,ior)`
- `ashikhmin_shirley(N,T,ax,ay)`
- `ashikhmin_velvet(N,roughness)`

Hair

- `hair_reflection(N,roughnessu,roughnessv,T,offset)`
- `hair_transmission(N,roughnessu,roughnessv,T,offset)`

BSSRDF

- `bssrdf_cubic(N,radius,texture_blur,sharpness)`
- `bssrdf_gaussian(N,radius,texture_blur)`

Volume

- `henyey_greenstein(g)`
- `absorption()`

Other

- `emission()`
- `ambient_occlusion()`
- `holdout()`
- `background()`

Attributes

Some object, particle and mesh attributes are available to the built-in getattribute() function. UV maps and vertex colors can be retrieved using their name. Other attributes are listed below:

`geom:generated` Generated texture coordinates.

`geom:uv` Default render UV map.

`geom:dupli_generated` For instances, generated coordinate from duplicator object.

`geom:dupli_uv` For instances, UV coordinate from duplicator object.

`geom:trianglevertices` 3 vertex coordinates of the triangle.

`geom:numpolyvertices` Number of vertices in the polygon (always returns three currently).

`geom:polyvertices` Vertex coordinates array of the polygon (always three vertices currently).

`geom:name` Name of the object.

`geom:is_curve` Is object a strand or not.

`geom:curve_intercept` Point along the strand, from root to tip.

`geom:curve_thickness` Thickness of the strand.

`geom:curve_tangent_normal` Tangent Normal of the strand.

`path:ray_length` Ray distance since last hit.

`object:location` Object location.

`object:index` Object index number.

`object:random` Per object random number generated from object index and name.

`material:index` Material index number.

`particle:index` Particle instance number.

`particle:age` Particle age in frames.

`particle:lifetime` Total lifespan of particle in frames.

`particle:location` Location of the particle.

`particle:size` Size of the particle.

`particle:velocity` Velocity of the particle.

`particle:angular_velocity` Angular velocity of the particle.

Trace

We support the `trace(point pos,vector dir,...)` function, to trace rays from the OSL shader. The "shade" parameter is not supported currently, but attributes can be retrieved from the object that was hit using the `getmessage("trace",..)` function. See the OSL specification for details on how to use this.

This function cannot be used instead of lighting; the main purpose is to allow shaders to "probe" nearby geometry, for example to apply a projected texture that can be blocked by geometry, apply more "wear" to exposed geometry, or make other ambient occlusion-like effects.

Node Types

Input Nodes

Attribute Node

The *Attribute* node allows you to retrieve attributes attached to an object or mesh.

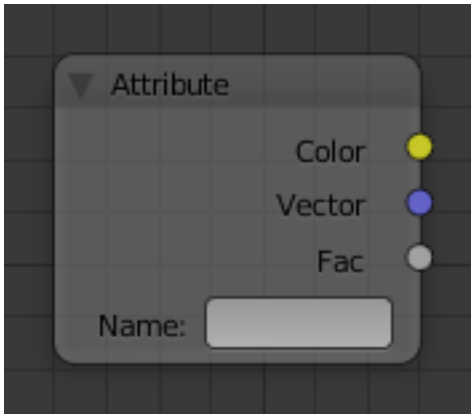


Fig. 2.1931: Attribute Node.

Inputs

This node has no inputs.

Properties

Name Name of the attribute. Currently, the following are the most important ones that you will need to know:

Vertex Color Layers These can be retrieved this by their names.

Density Gives a scalar defining the density of any smoke inside the *Smoke Domain*.

Flame Gives a scalar defining the density of any fire inside the *Smoke Domain*. All three outputs are the same.

Color Gives the color of the smoke inside the *Smoke Domain*. The color and vector outputs are the same. The Factor output is an average of the channels.

Ocean Foam Gives a scalar define where foam might apear when using an *Ocean Modifier*. This depends on the name you give this property.

See also:

For a full list of options see This Tread on the Blender Stack Exchange.

Outputs

Color RGB color interpolated from the attribute.

Vector XYZ vector interpolated from the attribute.

Factor Scalar value interpolated from the attribute.

Camera Data Node

The *Camera Data* node is used for getting information about what the camera is view in order to achieve different effects.

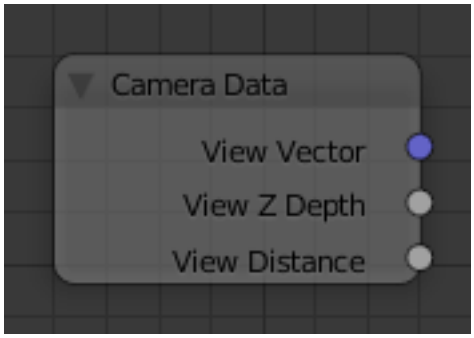


Fig. 2.1932: Camera Data Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

View Vector A Camera space vector from the camera to the shading point.

View Z Depth The distance each pixel is away from the camera.

View Distance Distance from the camera to the shading point.

Fresnel Node

The *Fresnel* or *Dielectric Fresnel* node computes how much light is reflected off a layer, where the rest will be refracted through the layer. The resulting weight can be used for layering shaders with the *Mix Shader* node. It is dependent on the angle between the surface normal and the viewing direction.

The most common use is to mix between two BSDFs using it as a blending factor in a mix shader node. For a simple glass material you would mix between a glossy refraction and glossy reflection. At grazing angles more light will be reflected than refracted as happens in reality.

For a two-layered material with a diffuse base and a glossy coating, you can use the same setup, mixing between a diffuse and glossy BSDF. By using the Fresnel as the blending factor you are specifying that any light which is refracted through the glossy coating layer would hit the diffuse base and be reflected off that.

Fig. 2.1933: Fresnel Node.

Inputs

IOR Index of refraction (*IOR*) of the material being entered.

Normal Todo.

Properties

This node has no properties.

Outputs

Factor Fresnel weight, indicating the probability with which light will reflect off the layer rather than passing through.

Geometry Node

The *Geometry* node gives geometric information about the current shading point. All vector coordinates are in *World Space*. For volume shaders, only the position and incoming vector are available.

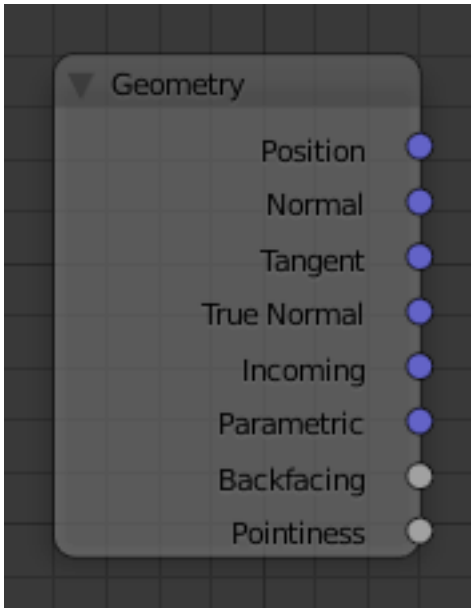


Fig. 2.1934: Geometry Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Position Position of the shading point.

Normal Shading normal at the surface (includes smooth normals and bump mapping).

Tangent Tangent at the surface.

True Normal Geometry or flat normal of the surface.

Incoming Vector pointing towards the point the shading point is being viewed from.

Parametric Parametric coordinates of the shading point on the surface.

Backfacing 1.0 if the face is being viewed from the back side, 0.0 for the front side.

Pointiness An approximation of the curvature of the mesh per-vertex. Lighter values indicate convex angles, darker values indicate concave angles.

Hair Info Node

The *Hair Info* node gives access to *Hair* information.

Fig. 2.1935: Hair Info Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Is Strand Returns 1 when the shader is acting on a strand, otherwise 0.

Intercept The point along the strand where the ray hits the strand (1 at the tip and 0 at the root).

Thickness The thickness of the strand at the point where the ray hits the strand.

Tangent Normal Tangent normal of the strand.

Layer Weight Node

The *Layer Weight* node outputs a weight typically used for layering shaders with the *Mix Shader* node.

Inputs

Blend Blend between the first and second shader.

Normal Input meant for plugging in bump or normal maps which will affect the output.

Properties

This node has no properties.

Outputs

Fresnel Dielectric Fresnel weight, useful for example for layering diffuse and glossy shaders to create a plastic material. This is like the Fresnel node, except that the input of this node is in the often more-convenient 0.0 to 1.0 range.

Facing Weight that blends from the first to the second shader as the surface goes from facing the viewer to viewing it at a grazing angle.

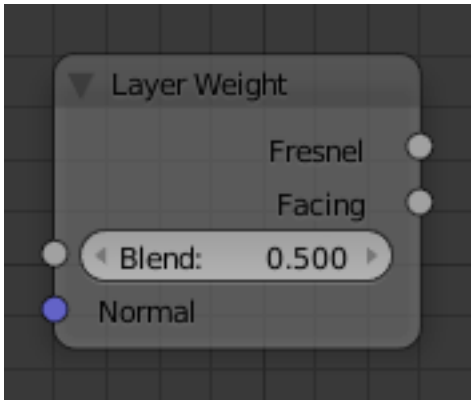


Fig. 2.1936: Layer Weight Node.

Light Path Node

The *Light Path* node is used to find out for which kind of incoming ray the shader is being executed; particularly useful for non-physically based tricks. More information about the meaning of each type is in the *Light Paths* documentation.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Is Camera Ray 1.0 if shading is executed for a camera ray, 0.0 otherwise.

Is Shadow Ray 1.0 if shading is executed for a shadow ray, 0.0 otherwise.

Is Diffuse Ray output 1.0 if shading is executed for a diffuse ray, 0.0 otherwise.

Is Glossy Ray 1.0 if shading is executed for a glossy ray, 0.0 otherwise.

Fig. 2.1937: Light Path Node.

Is Singular Ray 1.0 if shading is executed for a singular ray, 0.0 otherwise.

Is Reflection Ray 1.0 if shading is executed for a reflection ray, 0.0 otherwise.

Is Transmission Ray output 1.0 if shading is executed for a transmission ray, 0.0 otherwise.

Ray Length Distance traveled by the light ray from the last bounce or camera.

Ray Depth Number of times the ray has "bounced", i.e. been reflected or transmitted on interaction with a surface.

Note: Passing through a transparent shader *does not count as a normal "bounce"*.

Transparent Depth Returns the number of transparent surfaces passed through.

Transmission Depth Replace a Transmission lightpath after X bounces with another shader, e.g a Diffuse one. This can be used to avoid black surfaces, due to low amount of max bounces.

Object Info Node

The *Object Info* node gives information about the object instance. This can be useful to give some variation to a single material assigned to multiple instances, either manually controlled through the object index, based on the object location, or randomized for each instance. For example a Noise texture can give random colors or a Color ramp can give a range of colors to be randomly picked from.

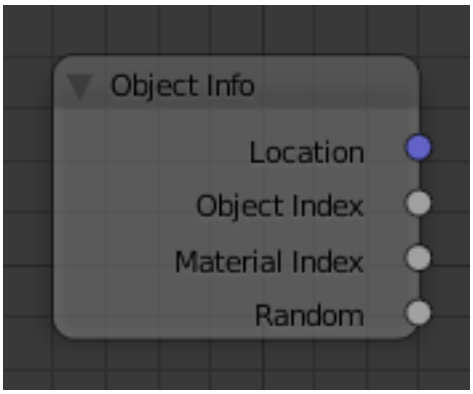


Fig. 2.1938: Object Info Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Location Location of the object in world space.

Object Index Object pass index, same as in the Object Index pass.transformed.

Material Index Material pass index, same as in the Material Index pass.

Random Random number unique to a single object instance.

Note: Note that this node only works for material shading nodes; it does nothing for lamp and world shading nodes.

Particle Info Node

The *Particle Info* node is for objects instanced from a *Particle System*, this node give access to the data of the particle that spawned the instance.

Note: This node currently only supports parent particles, info from child particles is not available.

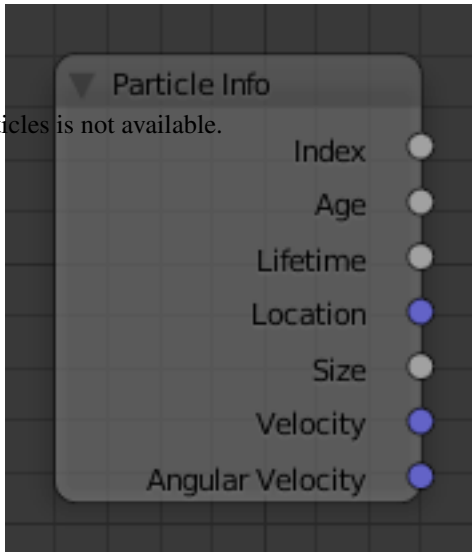


Fig. 2.1939: Particle Info Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Index Index number of the particle (from 0 to number of particles).

Age Age of the particle in frames.

Lifetime Total lifespan of the particle in frames.

Location Location of the particle.

Size Size of the particle.

Velocity Velocity of the particle.

Angular Velocity Angular velocity of the particle.

RGB Node

Inputs

This node has no input sockets.

Properties

The RGB node uses the *color picker widget*.

Outputs

Color / RGBA A single RGBA color value.

Fig. 2.1941: Tangent Node.

Tangent Node

The *Tangent* node generates a tangent direction for the Anisotropic BSDF.

Inputs

This node has no inputs.

Fig. 2.1940: RGB Node.

Properties

Direction Type The tangent direction can be derived from a cylindrical projection around the X, Y, or Z axis (Radial), or from a manually created UV Map for full control.

Outputs

Tangent The tangent direction vector.

Texture Coordinate Node

The *Texture Coordinate* node is commonly used for the coordinates of textures, typically used as inputs for the *Vector* input for texture nodes.

Inputs

This node has no inputs.

Properties

Object Specific object to use for object space coordinates. This only affects the *Object* output.

From Dupli If the material is applied to a dupli object, use texture coordinates from the parent object. This only affects the *Generated* and *UV* outputs.

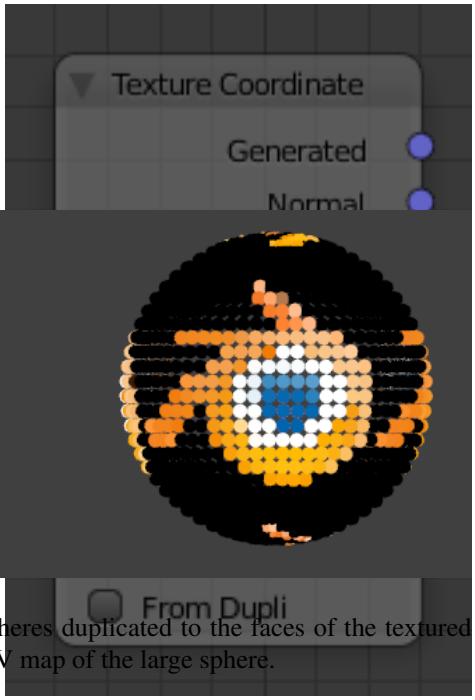


Fig. 2.1942: Texture Coordinate Node.

Fig. 2.1943: From left to right: Sphere with UV mapped texture. Small spheres duplicated to the faces of the textured sphere using *duplifaces*. Small spheres with *From Dupli* enabled, using the UV map of the large sphere.

Note: *From Dupli* only works with the UV output when the dupli object is instanced from faces, either with *particles* or *duplifaces*.

Outputs

Generated Automatically-generated texture coordinates from the vertex positions of the mesh without deformation, keeping them sticking to the surface under animation. Range from 0.0 to 1. 0 over the bounding box of the undeformed mesh.

Normal Object space normal, for texturing objects with the texture staying fixed on the object as it transformed.

UV UV texture coordinates from the active render UV map.

Object Position coordinate in object space.

Camera Position coordinate in camera space.

Window Location of shading point on the screen, ranging from 0.0 to 1. 0 from the left to right side and bottom to top of the render.

Reflection Vector in the direction of a sharp reflection, typically used for environment maps.

UV Map Node

The *UV Map* node is used to retrieve specific UV maps. Unlike the *Texture Coordinate Node* which only provides the active UV map, this node can retrieve any UV map belonging to the object using the material.

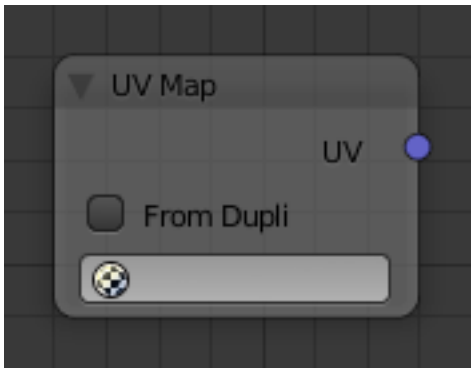


Fig. 2.1944: UV Map Node.

Inputs

This node has no inputs.

Properties

From Dupli See the *From Dupli* option of the *Texture Coordinate Node*.

UV Map UV map to use.

Outputs

UV UV mapping coordinates from the specified UV layer.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.

Inputs

This node has no input sockets.

Properties

Single numerical value (floating point).

Outputs

Value The value set in the options.

Fig. 2.1945: Value Node.

Example

In the example below the *Value Node* is used to control multiple values at once, this make the node a useful organizational tool.

Fig. 2.1946: Example of the *Value Node*.

Tip: From this you can also make different values proportional to each other by adding a *Math Node* in between the different links.

Wireframe Node

The *Wireframe* node is used to retrieve the edges of an object as it appears to cycles. As meshes are triangulated before being processed by cycles, topology will always appear triangulated when viewed with the *Wireframe node*.

Inputs

This node has no inputs.

Properties

Pixel Size When enabled, the size of edge lines are set in screen space.

Size Thickness of the edge lines.

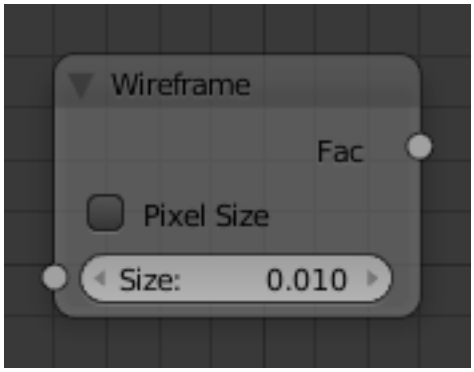


Fig. 2.1947: Wireframe Node.

Outputs

Factor Black and white mask showing white lines representing edges according to the object's *topology*.

Examples

Todo.

Output Nodes

Output nodes are the final node in every node tree. Although you can add more than one, only one will be used (indicated by a colored or darkened header). Output nodes are always preceded by *Shaders* except in the case of the *Displacement* of a Material Output.

Material Node

The *Material Output* node is used to output surface material information to a surface object.

Inputs

Surface The surface output of the material.

Volume Used to output of the different volume shaders.

> **See also:**
>
> The types of volume shaders are:
>
> - *Emission* shader.
> - *Volume Absorption* shader.
> - *Volume Scatter* shader.

Displacement Used to create bump mapping or actual subdivided *Displacement*.

Properties

This node has no properties.

Outputs

This node has no outputs.

Lamp Node

The *Lamp Output* node is used to output light information to a lamp object.

Inputs

Surface Not an actual surface, but the final output of a *Lamp* Object.

Properties

This node has no properties.

Outputs

This node has no outputs.

World Node

The *World Output* node is used to output light a color information to the scene's *World*.

Inputs

Surface The appearance of the environment, usually preceded by a *Background* shader.

Volume Used to add volumetric effects to the world. See the *Volume Absorption* and *Volume Scatter* for more information.

> **Note:** It is not possible to have and HDR and volumetric due to the fact that HDR's are assumed to be an infant distance from the the camera.

Properties

This node has no properties.

Outputs

This node has no outputs.

Shader Nodes

Add Node

The *Add* node is uses to add to *Shaders* together.

Inputs

Shaders Standard shader inputs.

Properties

This node has no properties.

Outputs

Shader Standard shader output.

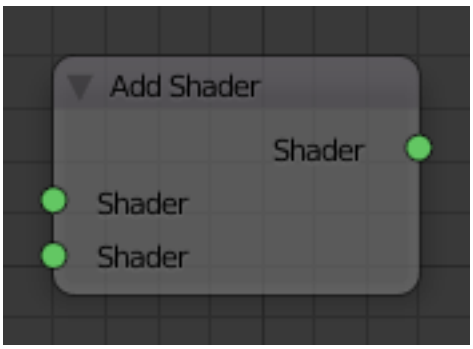


Fig. 2.1948: Add Node.

Fig. 2.1949: A mix of a glossy and a diffuse shader makes a nice ceramic material.

Example

Anisotropic Node

The *Anisotropic* BSDF node is used to add a glossy reflection, with separate control over U and V direction roughness. The tangents used for shading are derived from the active UV map. If no UV map is available, they are automatically generated using a sphere mapping based on the mesh bounding box.

Inputs

Color Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Roughness Sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

Anisotropy Amount of anisotropy in the reflection; 0.0 gives a round highlight. Higher values give elongated highlights orthogonal to the tangent direction; negative values give highlights shaped along the tangent direction.

Rotation Rotation of the anisotropic tangent direction. Value 0.0 equals 0° rotation, 0.25 equals 90° and 1.0 equals 360° = 0° . This can be used to texture the tangent direction.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

Tangent Tangent used for shading; if nothing is connected the default shading tangent is used.

Properties

Distribution Microfacet distribution to use. *Sharp* results in perfectly sharp reflections like a mirror, while *Beckmann*, *GGX* and *Ashikhmin-Shirley* can use the *Roughness* input for blurry reflections.

Fig. 2.1950: Anisotropic Node.

Outputs

BSDF output Standard shader output.

Examples

Fig. 2.1951: Anisotropic rotation on 0.

Fig. 2.1952: Anisotropic rotation on 0.25 (90°)

Ambient Occlusion Node

The *Ambient Occlusion* shader node gives per-material control for the amount of AO. When AO is enabled in the world, it affects all diffuse BSDFs in the scene. With this option it is possible to let only some materials be affected by AO, or to let it influence some materials more or less than others.

Fig. 2.1953: Ambient Occlusion Node.

Inputs

Color Surface reflection color.

Properties

This node has no properties.

Outputs

AO Standard shader output.

Example

Fig. 2.1954: White AO shader.

Background Node

The *Background* shader node is used to add background light emission. This node should only be used for the world surface output; it is ignored in other cases.

Inputs

Color Color of the emitted light.

Strength Strength of the emitted light.

Properties

This node has no properties.

Outputs

Background Standard shader output.

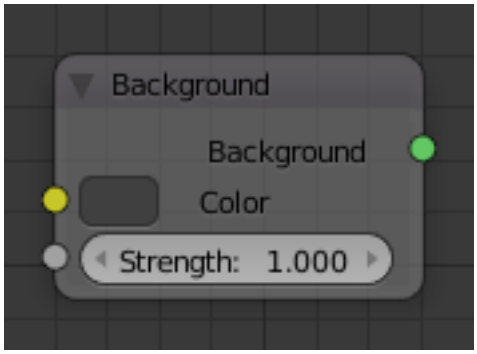


Fig. 2.1955: Background Node.

Examples

Todo.

Diffuse Node

The *Diffuse* BSDF node is used to add Lambertian and Oren-Nayar diffuse reflection.

Inputs

Color Color of the surface, or physically speaking, the probability that light is reflected or transmitted for each wavelength.

Roughness Surface roughness; 0.0 gives standard Lambertian reflection, higher values activate the Oren-Nayar BSDF.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

Fig. 2.1956: Diffuse Node.

Properties

This node has no properties.

Outputs

BSDF Standard shader output.

Examples

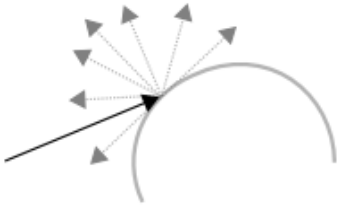
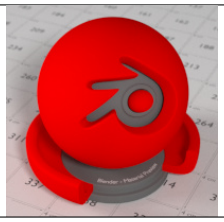
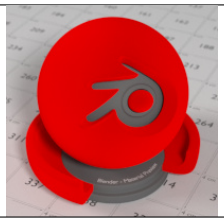


Fig. 2.1957: Diffuse behavior.





Emission Node

The *Emission* BSDF node is used to add Lambertian emission. This can for example, be used for material and lamp surface outputs.

Cycles uses a physically correct light falloff by default, whereas Blender Internal uses a smoothed falloff with a Distance parameter. A similar effect can be found by using the Light Falloff node with the Smooth parameter.

Lamp strength for point, spot and area lamps is specified in Watts. This means you typically need higher values than Blender Internal, as you could not use a 1W lamp to light a room; you need something stronger like a 100W lamp.

Sun lamps are specified in Watts/m^2, which require much smaller values like 1 W/m^2. This can be confusing, but specifying strength in Watts would not have been convenient; the real sun for example has strength 384.6×10^{24}W. Emission shaders on meshes are also in Watts/m^2.

Fig. 2.1958: Emission Node.

Inputs

Color Color of the emitted light.

Strength Strength of the emitted light. For point and area lamps, the unit is Watts. For materials, a value of 1.0 will ensure that the object in the image has the exact same color as the Color input, i.e. make it 'shadeless'.

Properties

This node has no properites.

Outputs

Emission Standard shader output.

Examples

Fig. 2.1959: Emission shader, with strength at 1.0 .

Fig. 2.1960: Emission shader, with strength at 3.0 .

Glass Node

The *Glass* BSDF node is used to add a Glass-like shader mixing refraction and reflection at grazing angles. Like the transparent shader, only pure white will make it transparent. The glass shader tends to cause noise due to caustics. Since the Cycles path tracing integrator is not very good at rendering caustics, it helps to combine this with a transparent shader for shadows; for *more details see here*.

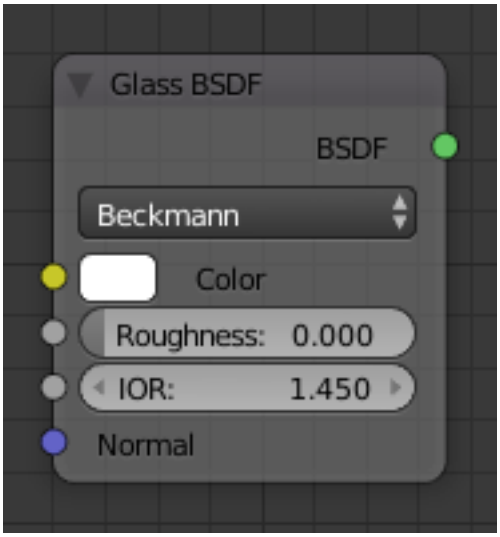


Fig. 2.1961: Glass Node.

Inputs

Distribution Microfacet distribution to use. *Sharp* results in perfectly sharp refractions like clear glass, while *Beckmann* and *GGX* can use the *Roughness* input for rough glass.

Color Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

Roughness Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

IOR Index of refraction (*IOR*) defining how much the ray changes direction. At 1. 0 rays pass straight through like transparent; higher values give more refraction.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

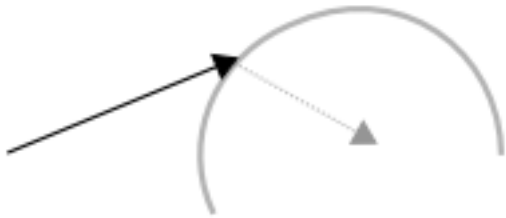
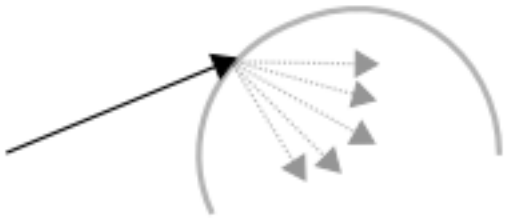
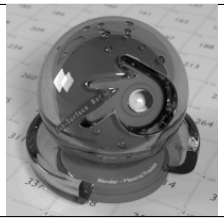
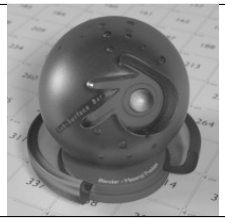
Properties

Distribution Microfacet distribution to use. *Sharp* results in perfectly sharp refractions like clear glass, while *Beckmann* and *GGX* can use the *Roughness* input for rough glass.

Outputs

BSDF Standard shader output.

Examples

Sharp Glass	Rough Glass
	
	

Glossy Node

The *Glossy* BSDF node is used to add reflection with microfacet distribution, used for materials such as metal or mirrors.

Fig. 2.1962: Glossy Node.

Inputs

Color Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Roughness Influences sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

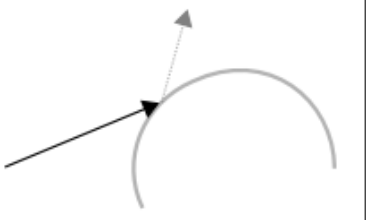
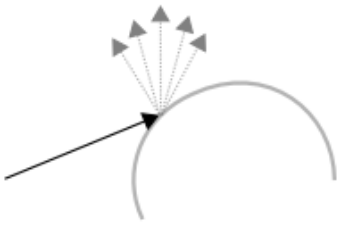
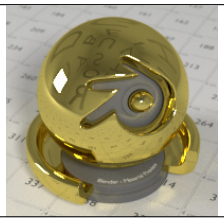
Properties

Distribution Microfacet distribution to use. *Sharp* results in perfectly sharp reflections like a mirror, while *Beckmann*, *GGX* and *Ashikhmin-Shirley* can use the *Roughness* input for blurry reflections.

Outputs

BSDF Standard shader output.

Examples

Sharp Glossy	Rough Glossy
	
	

Hair Node

The *Hair* BSDF node is used to add shading for *Hair*.

Fig. 2.1963: Hair Node.

Inputs

Color Color of the hair.

Offset Controls the way the light is rotated for the reflection/transmission.

Roughness U/V Controls the roughness in the direction light is skewed, and perpendicular to it.

Tangent Input tangent.

Properties

Component There are two components that can be used to control the look of the hair. Usually you are going to want each of these and use a *Mix Node*.

Reflection The light that bounces off the surface of the hair.

Transmission The light that passes through the hair and comes out the other side.

Outputs

BSDF Standard shader output.

Examples

Todo.

Holdout Node

The *Holdout* shader node is used to create a "hole" in the image with zero alpha transparency, which is useful for compositing (see *alpha channel*).

Note that the holdout shader can only create alpha when *Properties* → *Render* → *Film* → *Transparent* is enabled. If it is disabled, the holdout shader will be black.

Inputs

This node has no inputs.

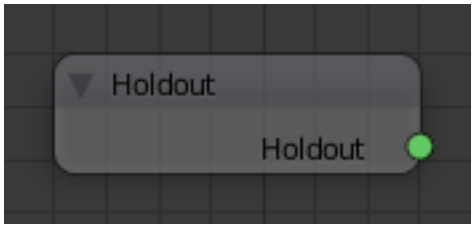


Fig. 2.1964: Holdout Node.

Properties

This node has no properties.

Outputs

Holdout Standard shader output.

Examples

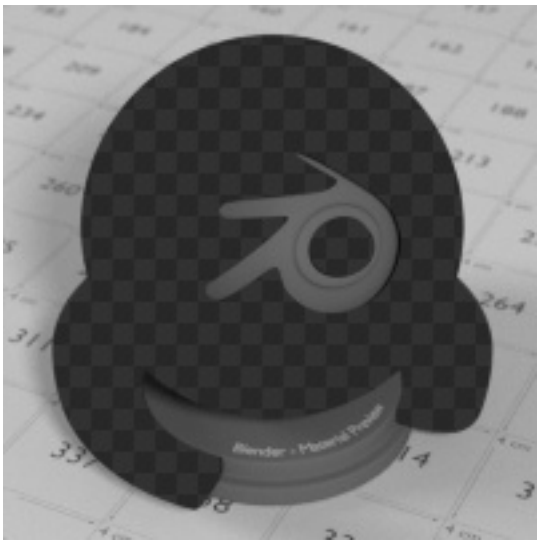


Fig. 2.1965: The checkered area is a region with zero alpha.

Mix Node

The *Mix* node is used to mix two shaders together. Mixing can be used for material layering, where the *Factor* input may, for example, be connected to a *Blend Weight* node.

Inputs

Shader Shaders to mix, such that incoming rays hit either with the specified probability in the *Factor* socket.

Factor Blend weight to use for mixing two shaders; at zero it uses the first shader entirely and at one the second shader.

Properties

This node has no properties.

Outputs

Shader Standard shader output.

Examples

Fig. 2.1967: A mix of a glossy and a diffuse shader makes a nice ceramic material.

Refraction Node

The *Refraction* BSDF node is used to add glossy refraction with sharp or microfacet distribution, used for materials that transmit light. For best results this node should be considered as a building block and not be used on its own, but rather mixed with a glossy node using a Fresnel factor. Otherwise it will give quite dark results at the edges for glossy refraction.

Inputs

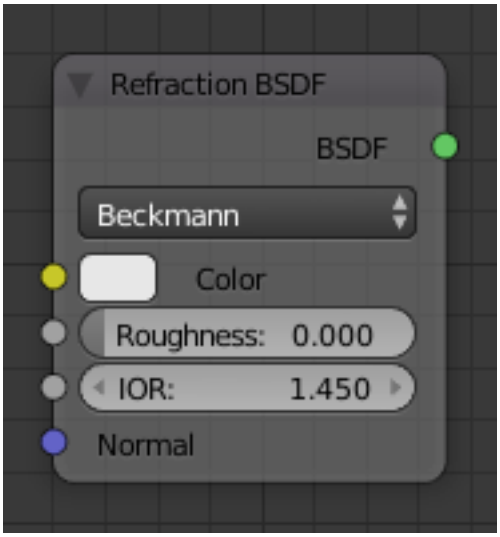


Fig. 2.1968: Refraction Node.

Color Color of the surface, or physically speaking, the probability that light is refracted for each wavelength.

Roughness Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Distribution Microfacet distribution to use. *Sharp* results in perfectly sharp refractions, while *Beckmann* and *GGX* can use the *Roughness* input for blurry refractions.

Outputs

BSDF Standard shader output.

Examples

Fig. 2.1969: Refraction Shader.

Subsurface Scattering Node

The *Subsurface Scattering* node is used to add simple subsurface multiple scattering, for materials such as skin, wax, marble, milk and others. For these materials, rather than light being reflect directly off the surface, it will penetrate the surface and bounce around internally before getting absorbed or leaving the surface at a nearby point.

How far the color scatters on average can be configured per RGB color channel. For example, for skin, red colors scatter further, which gives distinctive red-colored shadows, and a soft appearance.

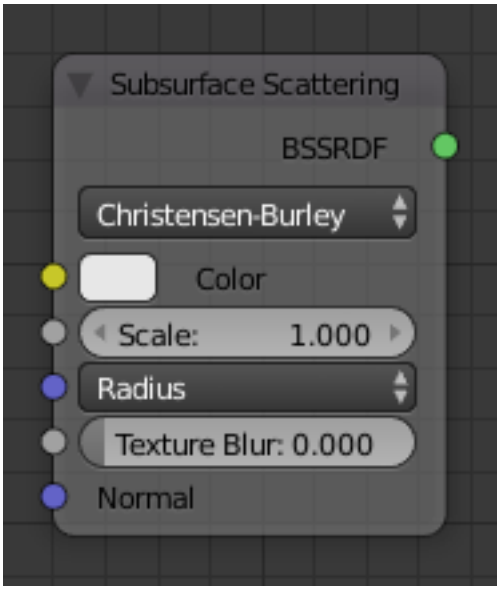


Fig. 2.1970: Subsurface Scattering Node.

Inputs

Color Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Scale Global scale factor for the scattering radius.

Radius Scattering radius for each RGB color channel, the maximum distance that light can scatter.

Sharpness Used only with *Cubic* falloff. Values increasing from 0 to 1 prevents softening of sharp edges and reduces unwanted darkening.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

Texture Blur How much of the texture will be blurred along with the lighting, mixing the texture at the incoming and outgoing points on the surface. Note that the right choice depends on the texture. Consider for example a texture created from a photograph of skin, in this cases the colors will already be pre-blurred and texture blur could be set to 0. Even for hand painted textures no or minimal blurring might be appropriate, as a texture artist would likely paint in softening already, one would usually not even know what an unblurred skin texture looks like, we always see it blurred. For a procedural texture on the other hand this option would likely have a higher value.

Properties

Falloff Lighting distance falloff function.

Cubic Is a sharp falloff useful for many simple materials. The function is $(radius - x)^3$.

Gaussian Gives a smoother falloff following a normal distribution, which is particularly useful for more advanced materials that use measured data that was fitted to one or more such Gaussian functions. The function is $e^{-8x^2/radius^2}$, such that the radius roughly matches the maximum falloff distance. To match a given measured variance *v*, set $radius = sqrt(16v)$.

Christensen-Burley Is an approximation to physically based volume scattering. Gives less blurry results than Cubic and Gaussian functions.

Outputs

BSSRDF BSSRDF (Bidirectional subsurface scattering distribution function) shader output.

Examples

Fig. 2.1971: A skin-toned SSS shader with color radius (1.0, 0.8, 0.5).

Toon Node

The *Toon* BSDF is used to create *Diffuse* and *Glossy* materials with cartoon light effects.

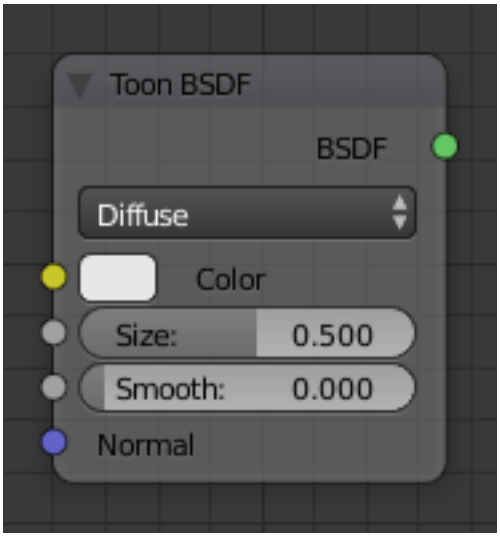


Fig. 2.1972: Toon Node.

Inputs

Color Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Size Parameter between 0.0 and 1.0 that gives an angle of reflection between 0° and 90°.

Smooth This value specifies an angle over which a smooth transition from full to no reflection happens.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

Properties

This node has no properties.

Outputs

BSDF Standard shader output.

Examples

Translucent Node

The *Translucent* BSDF is used to add Lambertian diffuse transmission.

Fig. 2.1973: Toon Shader.

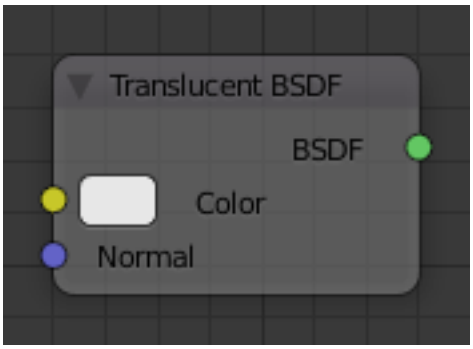
Inputs

Color Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

Properties

This node has no properties.

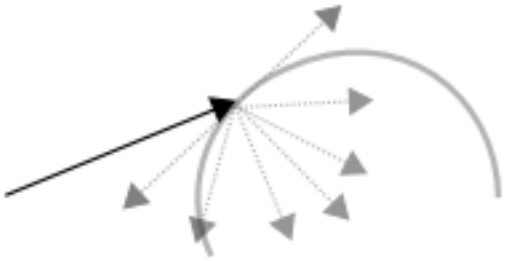


g. 2.1974: Translucent Node.

Outputs

BSDF output Standard shader output.

Examples



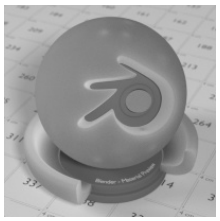


Fig. 2.1975: Translucent Shader.

Transparent Node

The *Transparent* BSDF node is used to add trancperency without refraction, passing straight through the surface, as if there were no geometry there. Useful with alpha maps, for example. This shader *affects light paths somewhat differently* than other BSDFs. Note that only pure white transparent shaders are completely transparent.

Fig. 2.1976: Transparent Node.

Inputs

Color Color of the surface, or physically speaking, the probability for each wavelength that light is blocked or passes straight through the surface.

Properties

This node has no properties.

Outputs

BSDF Standard shader output.

Examples

Fig. 2.1977: Transparent behavior.

Fig. 2.1978: Transparent Shader (pure white).

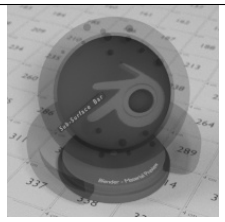
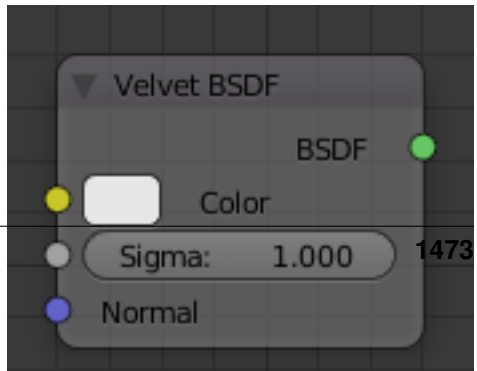


Fig. 2.1979: Transparent Shader (gray).

Velvet Node

The *Velvet* BSDF node is used to add reflection to materials such as cloth. It is meant to be used together with other shaders (such as a *Diffuse Shader*) and is not particularly useful on its own.



Inputs

Color Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Sigma Variance of the normal distribution, controlling the sharpness of the peak. It can be thought of as a kind of *roughness*.

Normal Normal used for shading; if nothing is connected the default shading normal is used.

Properties

This node has no properties.

Outputs

BSDF Standard shader output.

Examples

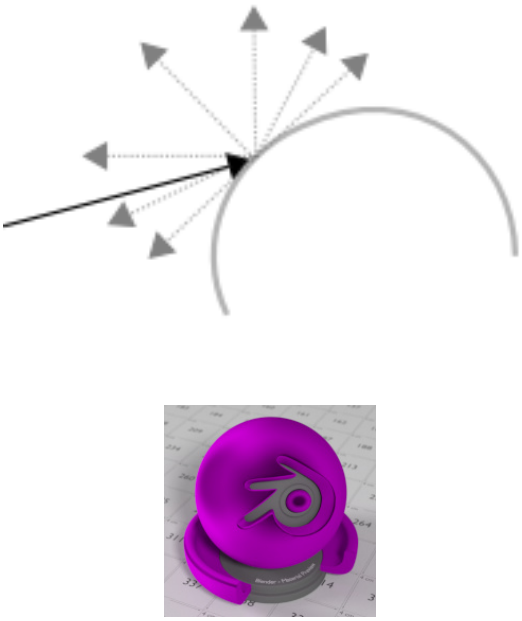


Fig. 2.1981: The Velvet Shader.

Volume Absorption Node

The *Volume Absorption* node allows light to be absorbed as is passes through it. Typical usage for this node would be water and glass. It can also be used with the *Volume Scatter* node to create smoke. This node must be plugged into the *Volume Input* of the *Material* output node.

Inputs

Color Color of the volume.

Density The density of the absorption effect.

Properties

This node has no properties.

Outputs

Volume Standard shader output.

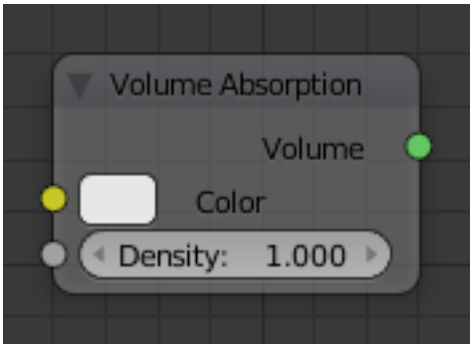


Fig. 2.1982: Volume Absorption Node.

Examples

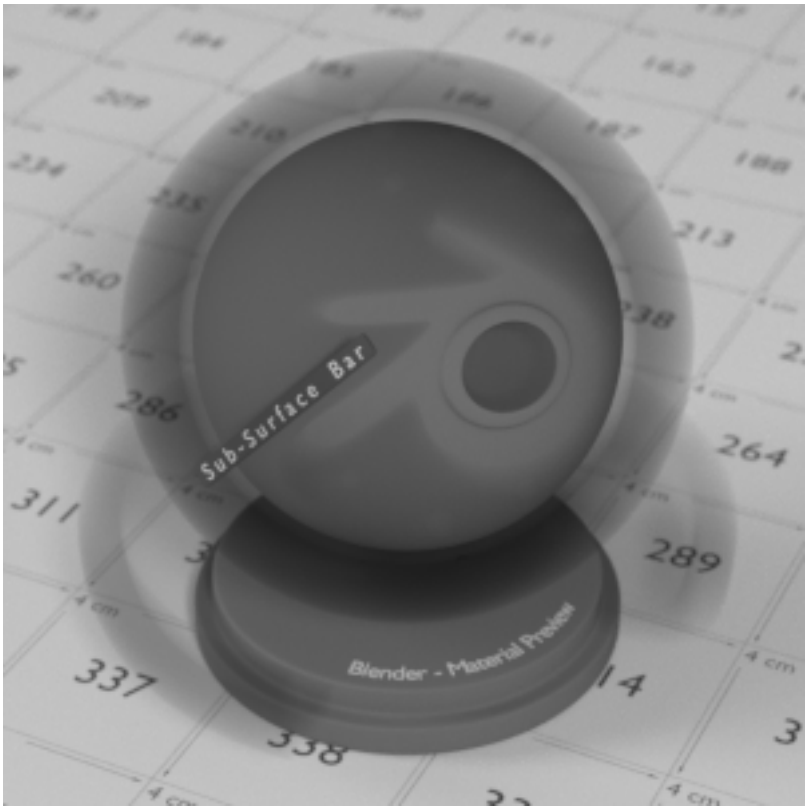
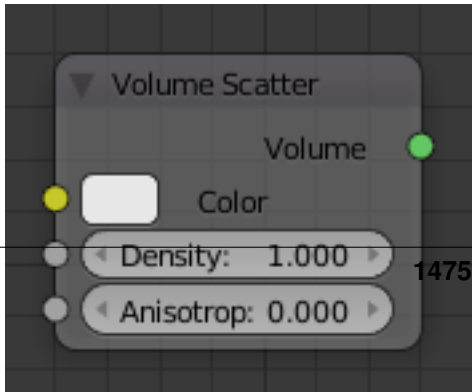


Fig. 2.1983: Example of Volume Absorption.

Volume Scatter Node

The *Volume Scatter* node allows light to be scattered scatter light as is passes through it. Typical usage would be to add fog to a scene. It can also be used with the *Volume Absorption* Node to create smoke. This node must be plugged into the *Volume Input* of the *Material* output node.



Inputs

Color Color of the volume.

Density The density of the scatter effect.

Anisotropy Controls the look of the scatter effect depending on the direction of the light passing through it.

Properties

Volume Standard shader output.

Examples

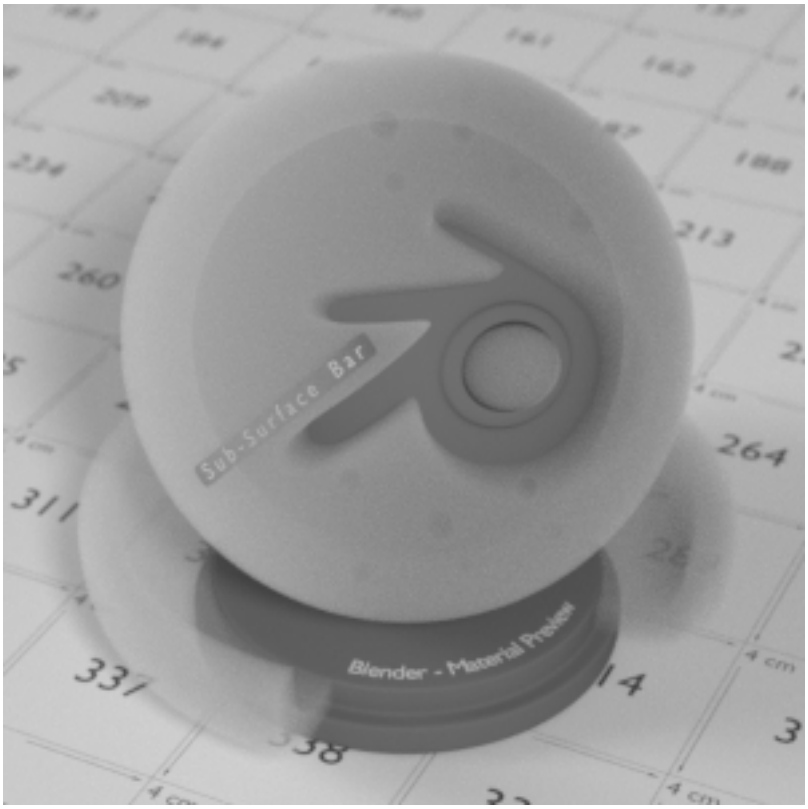


Fig. 2.1985: Example of Volume Scatter.

Texture Nodes

Brick Texture Node

The *Brick Texture* is used to add a procedural texture producing bricks.

Inputs

Color 1/2 and Mortar Color of the bricks and mortar.

Scale Overall texture scale.

Mortar Size The Mortar size; 0 means no Mortar.

Bias The color variation between *Color 1/2*. Values of -1 and 1 only use one of the two colors; values in between mix the colors.

Brick Width The width of the bricks.

Row Height The height of the brick rows.

Properties

Offset Determines the brick offset of the various rows.

Frequency Determines the offset frequency. A value of 2 gives an even/uneven pattern of rows.

Squash Amount of brick squashing.

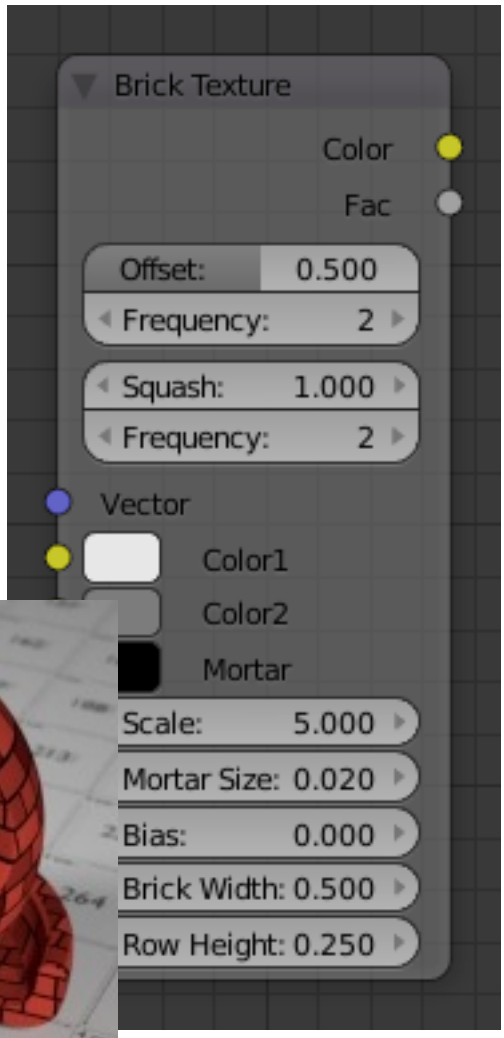
Frequency Brick squashing frequency.

Outputs

Color Texture color output.

Factor Mortar mask (1 = mortar).

Examples



2.1986: Brick Texture Node.

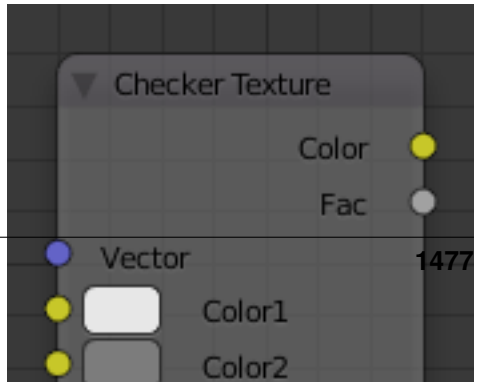
Fig. 2.1987: Brick texture: Colors changed, Squash 0.62, Squash Frequency 3.

Checker Texture Node

The *Checker Texture* is used to add a checkerboard texture.

Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.



Color1, Color 2 Color of the checkers.

Scale Overall texture scale. The scale is a factor of the bounding box of the face divided by the scale. For example, a scale of 15 will result in 15 alternate patterns over the overall UV bounding box. Different patterns could be achieved using other nodes to give different input patterns to this socket. For example, using the Math Node.

Properties

This node has no properties.

Outputs

Color Texture color output.

Factor Checker 1 mask (1 = Checker 1).

Examples

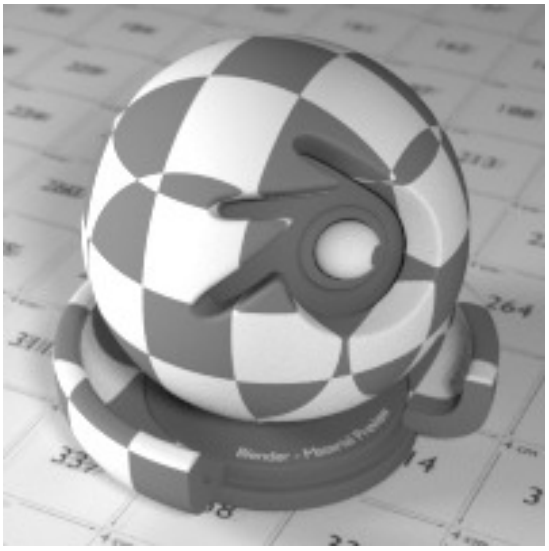


Fig. 2.1989: Default Checker texture.

Environment Texture Node

The *Environmental Texture* is used to light your scene using an environment map image file as a texture.

Inputs

Vector Texture coordinate for texture lookup. If this socket is left unconnected, the image is mapped as environment with the Z axis as up.

Properties

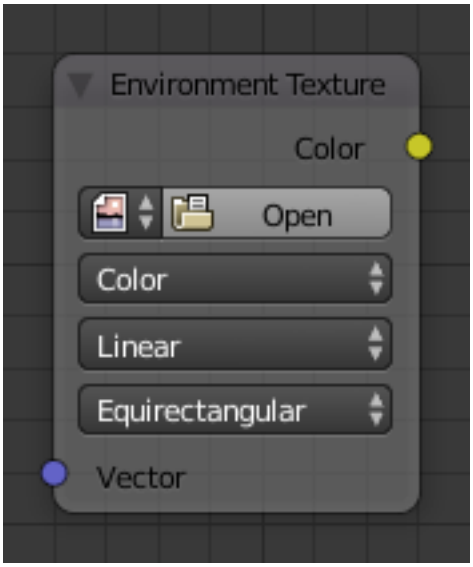


Fig. 2.1990: Environment Texture Node.

Image Data-Block Image data-block used as the image source.

Color Space Type of data that the image contains, either Color or Non-Color Data. For most color textures the default of Color should be used, but in case of e.g. a bump or alpha map, the pixel values should be interpreted as Non-Color Data, to avoid doing any unwanted color space conversions.

Texture Interpolation Interpolation method used for the environment texture. The following interpolations are available:

Linear Default.

Closest No interpolation.

Cubic Only available when rendering on the CPU.

Smart Bicubic when magnifying else Bilinear is used. This is only available for *OSL*.

Projection Method Allows you to use different types of environmental maps. The following methods are supported:

Equirectangular Projection from an Equirectangular photo.

Mirror Ball Projection from an orthographic photo or mirror ball.

Outputs

Color RGB color from the image.

Examples

Fig. 2.1991: HDR image from OpenFootage.net.

Gradient Texture Node

The *Gradient Texture* node is used a gradient texture.

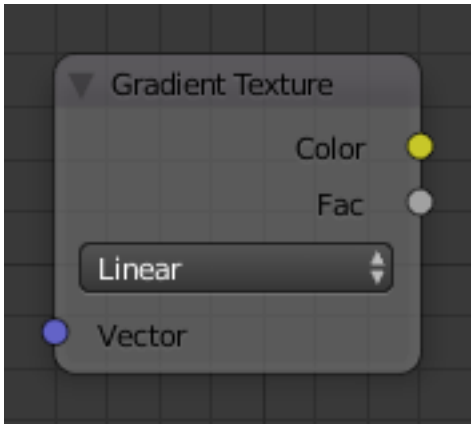


Fig. 2.1992: Gradient Texture Node.

Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Type The gradient can be *Linear*, *Quadratic*, *Easing*, *Diagonal*, *Spherical*, *Quadratic Sphere* or *Radial*.

Outputs

Color Texture color output.

Factor Texture intensity output.

Examples

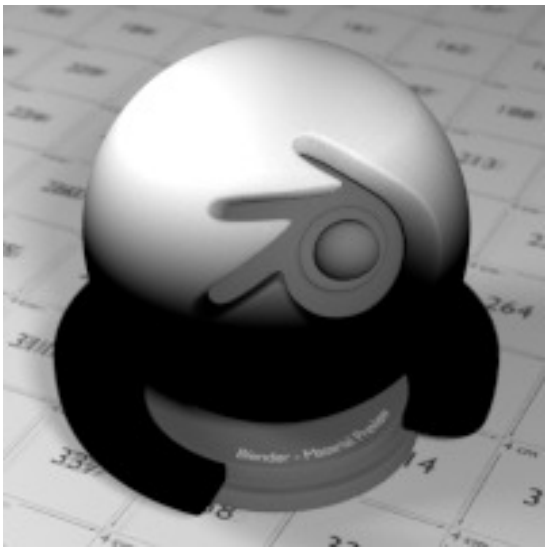


Fig. 2.1993: Gradient texture using object coordinates.

Image Texture Node

The *Image Texture* is used to add an image file as a texture.

Inputs

Vector Texture coordinate for texture lookup. If this socket is left unconnected, UV coordinates from the active UV render layer are used.

Properties

Image Data-Block Image data-block used as the image source. Currently not all images supported by Blender can be used by Cycles. In particular, generated, packed images or animations are not supported currently.

Color Space Type of data that the image contains, either Color or Non-Color Data. For most color textures the default of Color should be used, but in case of e.g. a bump or alpha map, the pixel values should be interpreted as Non-Color Data, to avoid doing any unwanted color space conversions.

Projection Projection to use for mapping the textures.

Flat Uses the XY coordinates for mapping.

Box Maps the image to the six sides of a virtual box, based on the normal, using XY, YZ and XYZ coordinates depending on the side.

Blend For Box mapping, the amount to blend between sides of the box, to get rid of sharp transitions between the different sides. Blending is useful to map a procedural-like image texture pattern seamlessly on a model. 0.0 gives no blending; higher values give a smoother transition.

Sphere Sphere mapping is the best type for mapping a sphere, and it is perfect for making planets and similar objects. It is often very useful for creating organic objects.

Tube Maps the texture around an object like a label on a bottle. The texture is therefore more stretched on the cylinder. This mapping is of course very good for making the label on a bottle, or assigning stickers to rounded objects. However, this is not a cylindrical mapping so the ends of the cylinder are undefined.

Extension Type Extension type defines how the image is extrapolated past the original bounds:

- *Repeat* will repeat the image horizontally and vertically giving tiled-looking result.
- *Extend* will extend the image by repeating pixels on its edges.
- *Clip* will set all the extended pixels values to transparent black.

Outputs

Color RGB color from image. If the image has alpha, the color is premultiplied with alpha if the Alpha output is used, and unpremultiplied or straight if the Alpha output is not used.

Alpha Alpha channel from image.

Examples

Magic Texture Node

The *Magic Texture* node is used to add psychedelic color texture.

Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale Scale of the texture.

Distortion Amount of distortion.

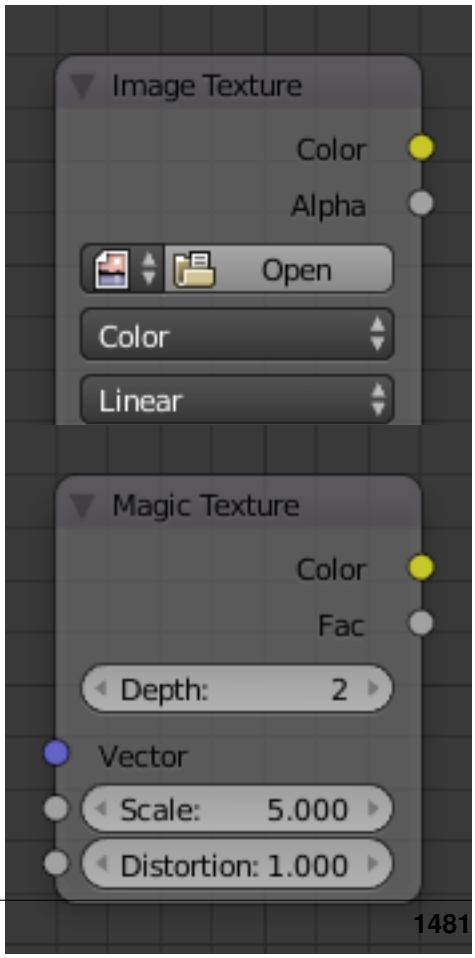


Fig. 2.1996: Magic Texture Node.

Fig. 2.1995: Image texture from GoodTextures.com.

Properties

Depth Number of iterations.

Outputs

Color Texture color output.

Factor Texture intensity output.

Examples

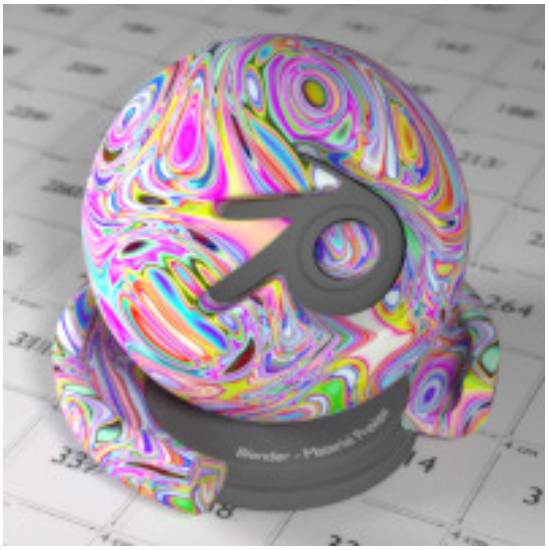


Fig. 2.1997: Magic texture: Depth 10, Distortion 2.0.

Musgrave Texture Node

The *Musgrave Texture* is used to add an advanced procedural noise texture.

Tip: The *Musgrave Texture* often needs some adjustments (multiplication and addition) in order to see more detail.

Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale Overall texture scale.

Detail Amount of noise detail.

Dimension The highest fractal dimension, specified as the highest scale for the steps of the intensity.

Lacunarity The space of the lacunarity, specified as a frequency factor.

Offset The offset of the fractal, specified between black and white values (Intensity).

Gain A multiplier for the gain input.

Properties

Type Multifractal, Ridged Multifractal, Hybrid Multifractal, fBM, Hetero Terrain.

Outputs

Color Texture color output.

Factor Texture intensity output.

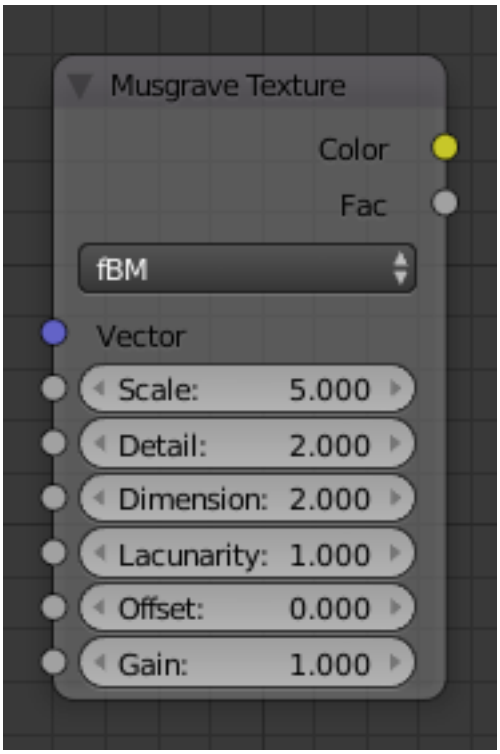
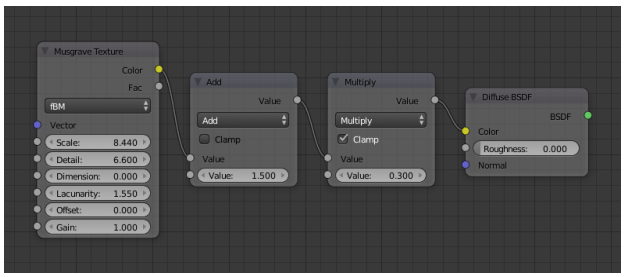
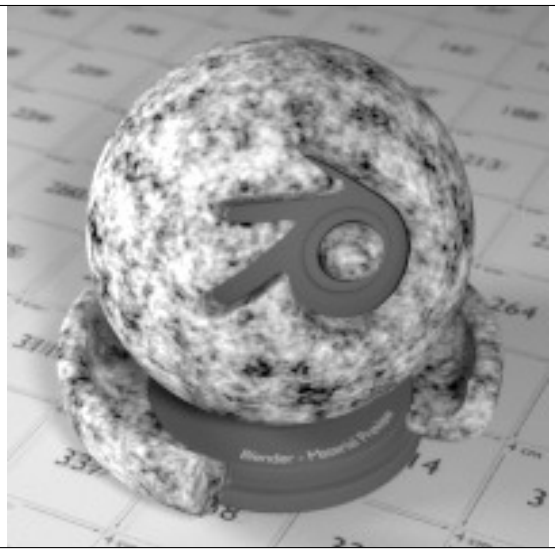


Fig. 2.1998: Musgrave Texture Node.

Examples

Table 2.101: Musgrave texture.





Noise Texture Node

The *Noise Texture* is used to add procedural Perlin noise texture, similar to the *Clouds* texture in *Blender Internal*.

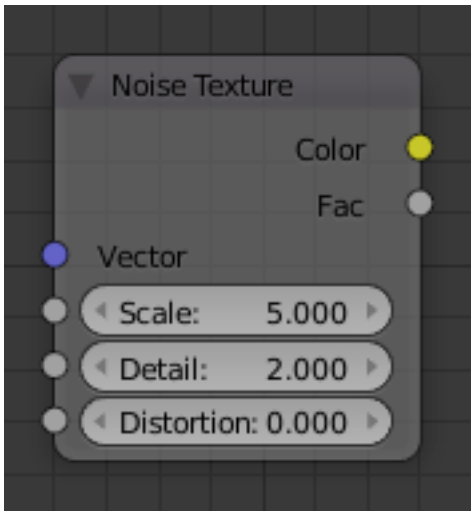


Fig. 2.2001: Noise Texture Node.

Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale Overall texture scale.

Detail Amount of noise detail.

Distortion Amount of distortion.

Properties

This node has no properties.

Outputs

Color Texture color output.

Factor Texture intensity output.

Examples

Point Density Node

The *Point Density* node is used to add volumetric points for each particle or vertex of another object.

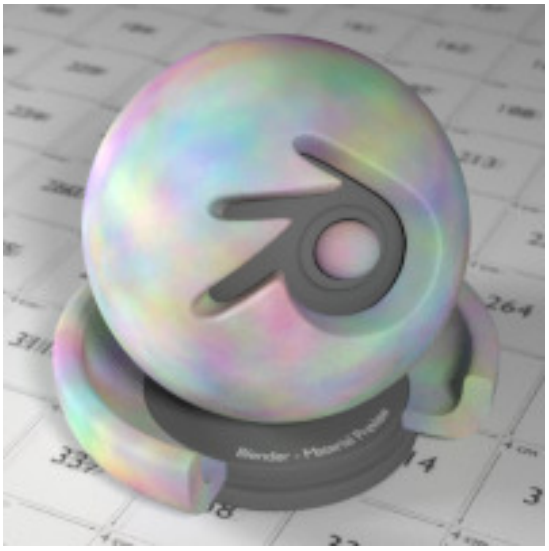


Fig. 2.2002: Noise Texture with high detail.

Inputs

Vector Texture coordinate to sample texture at; defaults to global position (*Position* output of *Geometry* node) if the socket is left unconnected.

Properties

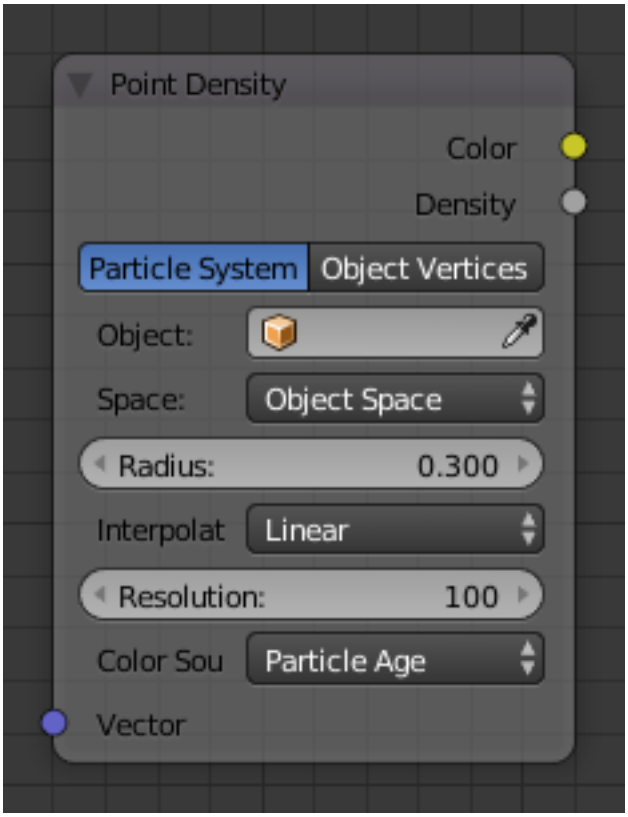


Fig. 2.2003: Point Density Node.

Point Data Where to get points from.

Particle System Use each particle position from the specified particle system.

Object Vertices Use each vertex position from the specified object.

Object Which object's vertices or particle system will be used.

Particle System Particle positions from this system will be used.

Space The coordinate system for mapping points.

World Space Map each point exactly where the source particle/vertex is.

Object Space Fit the points from the source particles/vertices inside the bounding box of the object with the point density texture.

Radius Radius from the shaded sample to look for points within.

Interpolation Texel filtering type.

Closest No interpolation, use nearest texel. Produces blocky looking points.

Linear Interpolate linearly between texels, producing soft, round points.

Cubic Use cubic falloff, producing very soft points. Useful when points are very densely packed.

Resolution The dimensions of the texture holding the point data.

Color Source Which attribute of the particle system or mesh is used to color the output.

Particle Color Sources

Particle Age Lifetime mapped as (0.0 - 1.0) intensity.

Particle Speed Particle speed (absolute magnitude of velocity) mapped as (0.0 - 1.0) intensity.

Particle Velocity XYZ velocity mapped to RGB colors.

Vertex Color Sources

Vertex Color Use a vertex color layer for coloring the point density texture.

Note: Vertex colors are defined per face corner. A single vertex can have as many different colors as faces it is part of. The actual color of the point density texture is averaged from all vertex corners.

Vertex Weight Use a weights from a vertex group as intensity values.

Vertex Normals Use object-space vertex normals as RGB values.

Outputs

Color Texture color output.

Density Density of volume.

Examples

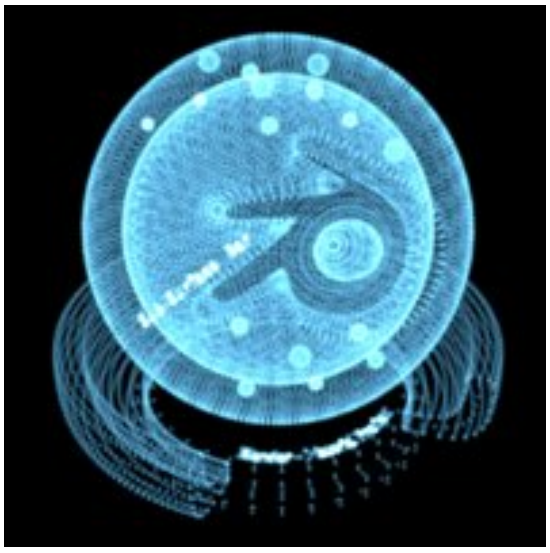


Fig. 2.2004: Domain object with Point Density texture using vertices from ball as points.

Sky Texture Node

The *Sky Texture* node adds a procedural Sky texture.

Fig. 2.2005: Sky Texture Node.

Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Sky Type Sky model to use (Preetham or Hosek/Wilkie).

Sun Direction Sun direction vector.

Turbidity Atmospheric turbidity. (2: Arctic like, 3: clear sky, 6: warm/moist day, 10: hazy day)

Ground Albedo Amount of light reflected from the planet surface back into the atmosphere. (RGB 0,0,0 is black, 1,1,1 is white).

Outputs

Color Texture color output.

Examples

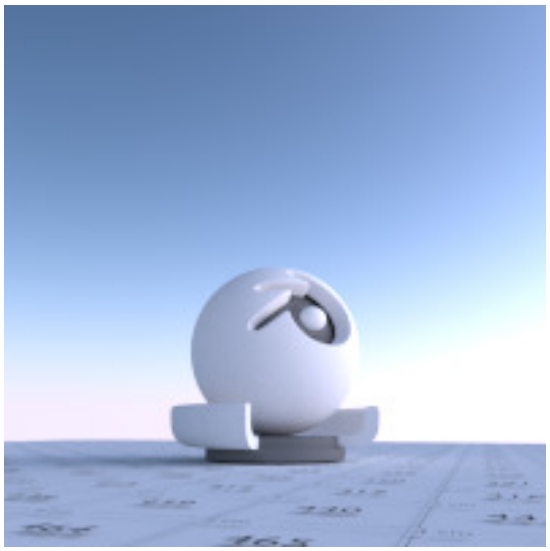


Fig. 2.2006: Sky Texture.

Voronoi Texture Node

The *Voronoi Texture* node adds a procedural texture producing Voronoi cells.

Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale Overall texture scale.

Properties

Coloring *Intensity* or *Cells* output.

Outputs

Color Texture color output.

Factor Texture intensity output.

Examples

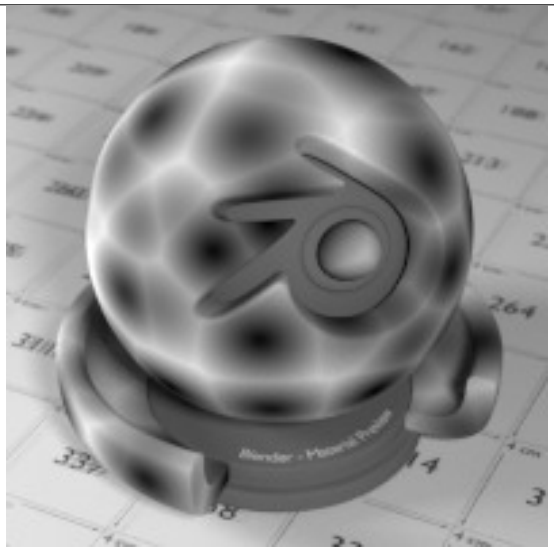


Fig. 2.2008: Voronoi texture, type: Intensity.

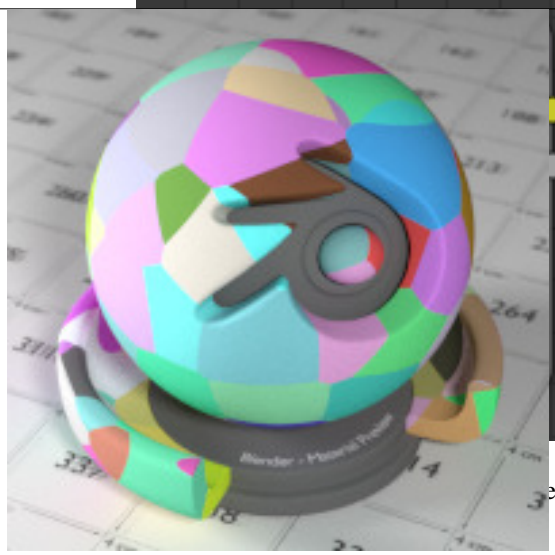
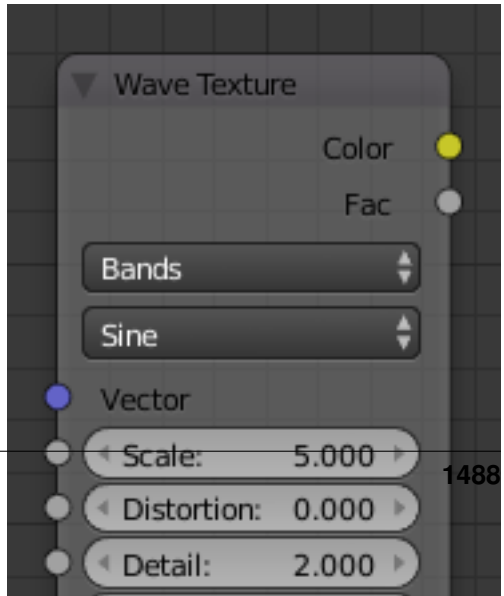


Fig. 2.2009: Voronoi texture, type: Cells.

Wave Texture Node

The *Wave Texture* node adds procedural bands or rings with noise distortion.



Inputs

Vector Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale Overall texture scale.

Distortion Amount of distortion of the wave (similar to the Marble texture in Blender Internal).

Detail Amount of distortion noise detail.

Detail Scale Scale of distortion noise.

Properties

Type *Bands* or *Rings* shaped waves.

Wave Profile Controls the look of the wave type.

Saw Uses a sawtooth profile.

Sine Uses the standard sine profile.

Outputs

Color Texture color output.

Factor Texture intensity output.

Examples

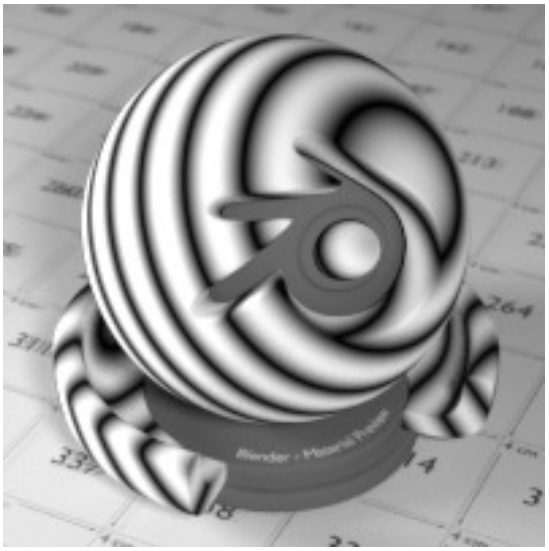


Fig. 2.2011: Default wave texture.

Color Nodes

Bright/Contrast Node

Inputs

Image Standard image input.

Bright A multiplier-type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast A scaling type factor by which to make brighter pixels brighter, but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

Properties

This node has no properties.

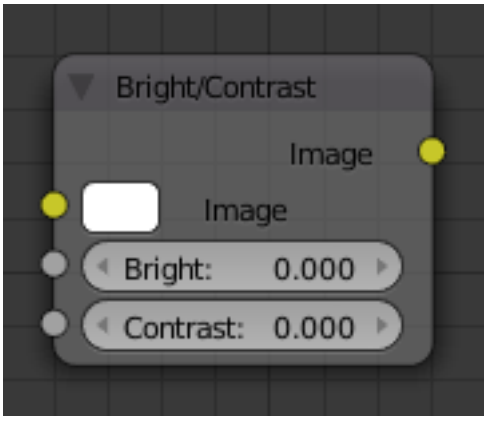


Fig. 2.2012: Bright/Contrast Node.

Outputs

Image Standard image output.

Notes

It is possible that this node will put out a value set that has values beyond the normal range, i. e. values greater than one and less than zero. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a Color Ramp node (with all normal defaults).

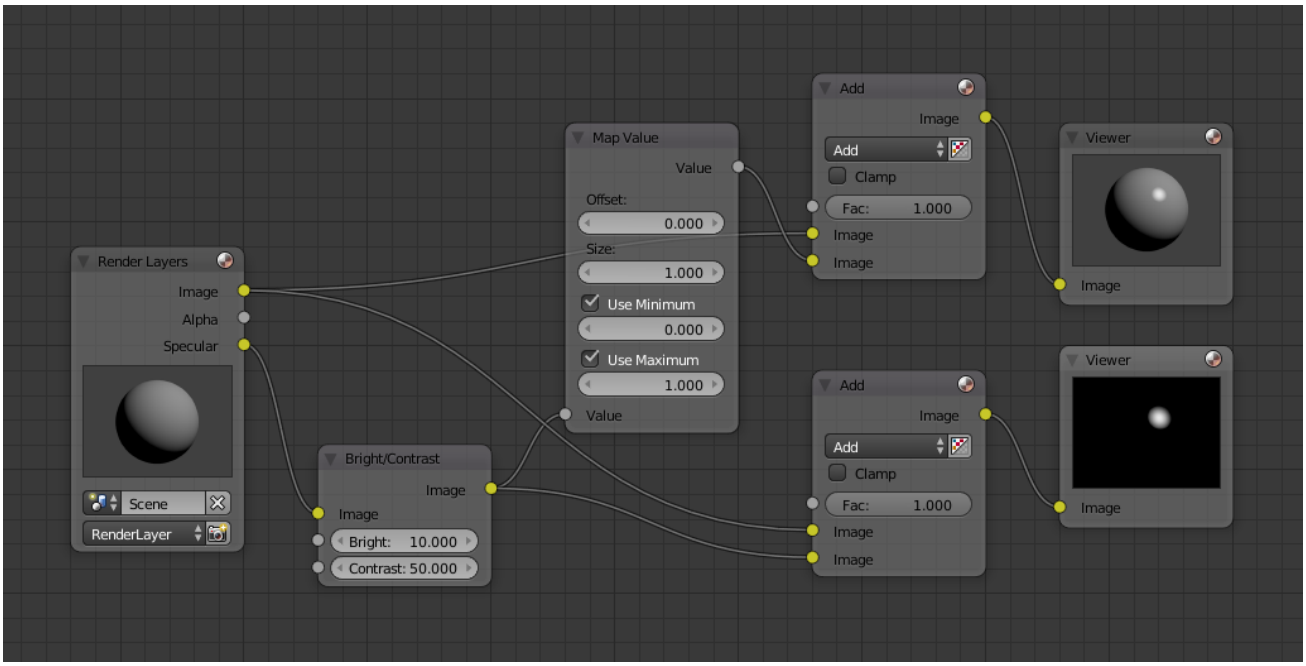


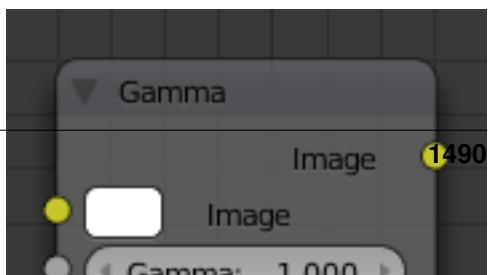
Fig. 2.2013: Clamp the values to normal range.

Either of these nodes will scale the values back to normal range. In the example image, we want to amp up the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has valued much less than one in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the Color Ramp node produces the desired effect.

Example

Gamma Node

Use this node to apply a gamma correction.



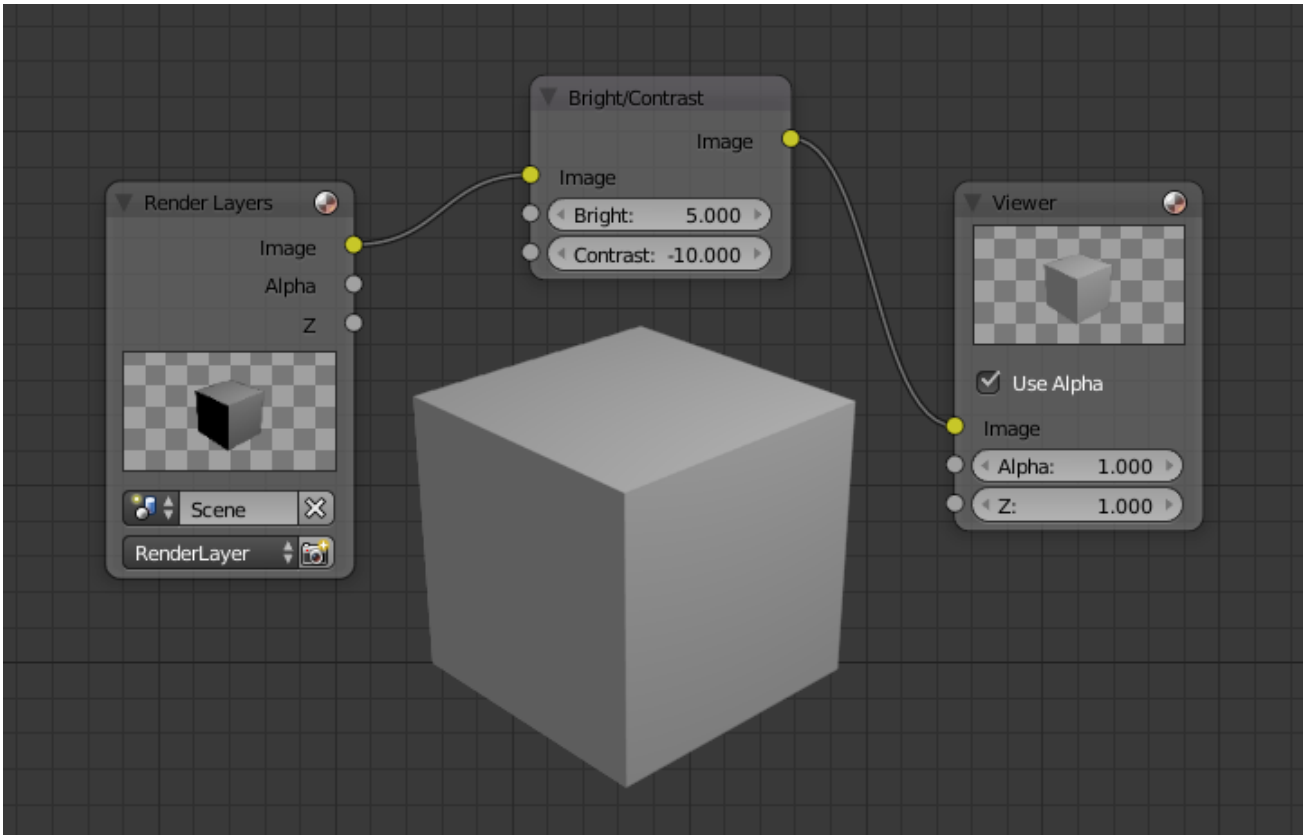


Fig. 2.2014: A basic example.

Inputs

Image Standard image input.

Gamma An exponential brightness factor.

Properties

This node has no properties.

Outputs

Image Standard image output.

Examples

Hue Saturation Value Node

This node applies a color transformation in the HSV color space. Called "Hue Saturation Value" in shader and texture context.

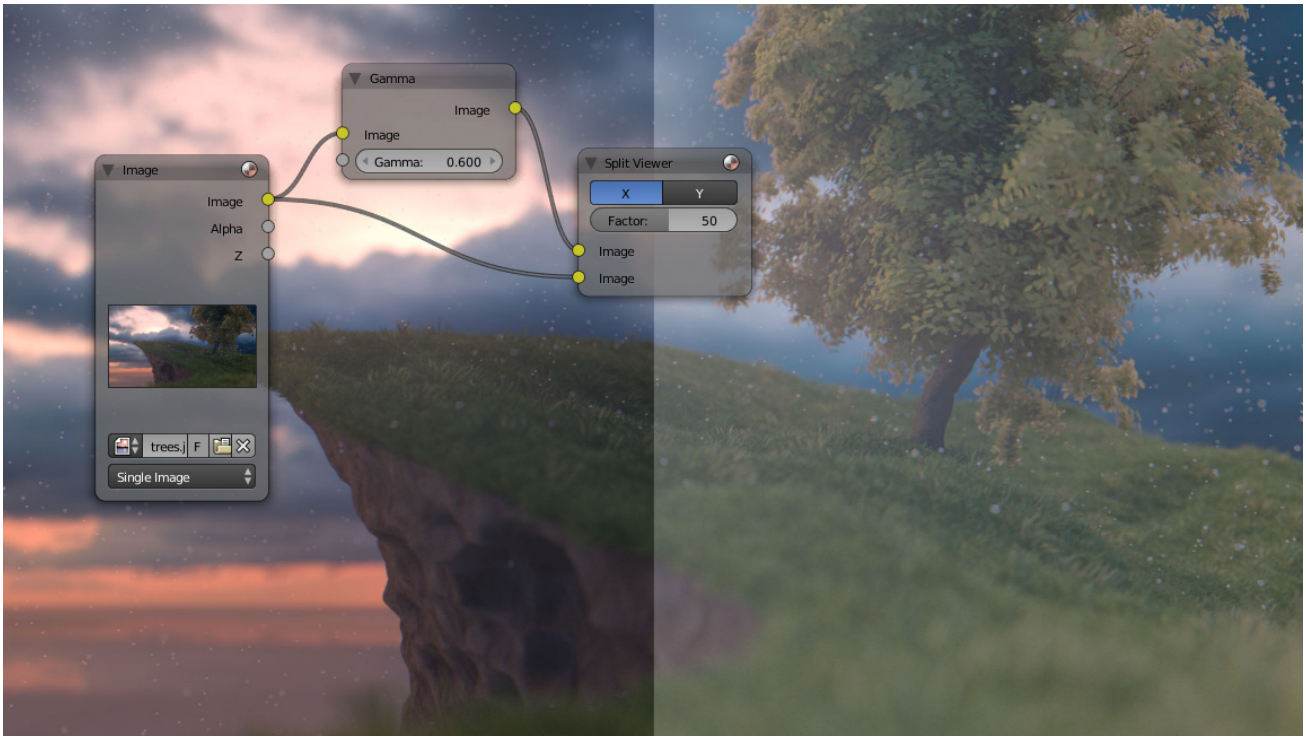


Fig. 2.2016: Example of Gamma node.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Properties

The transformations are relative shifts. In the shader and texture context the following properties are available as input sockets.

Hue Specifies how the hue rotation of the image. 360° are mapped to (0 to 1). The hue shift of 0 (-180°) and 1 (+180°) have the same result.

Saturation A saturation of 0 removes hues from the image, resulting in a grayscale image. A shift greater 1.0 increases saturation.

Value Value is the overall brightness of the image. De/Increasing values shift an image darker/lighter.

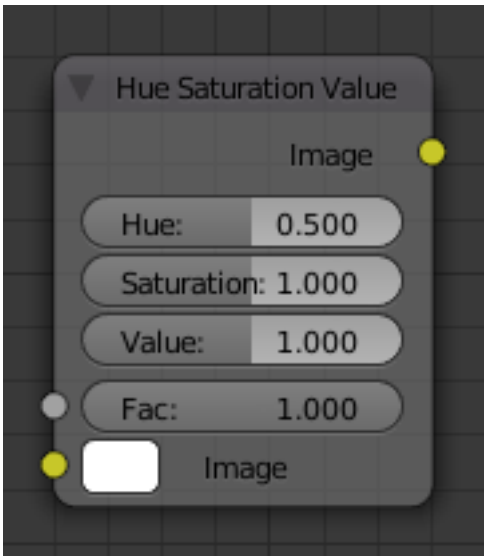


Fig. 2.2017: Hue Saturation Node.

Outputs

Image Standard image output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are vice versa A blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (recall that white, minus blue, equals yellow). A yellow image, with a Hue setting at 0 or 1, is blue.

Hue and Saturation work together. So, a Hue of 0.5 keeps the blues the same shade of blue, but *Saturation* can deepen or lighten the intensity of that color.

Gray & White are neutral hues A gray image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with *Value*. This applies to all shades of gray, from black to white; wherever the values are equal.

Changing the effect over time The Hue and Saturation values can be animated with a *Time Node* or by animating the property.

Note: Tinge

This HSV node simply shifts hues that are already there. To colorize a gray image, or to add a tint to an image, use a mix node to add in a static color from an RGB input node with your image.

HSV Example

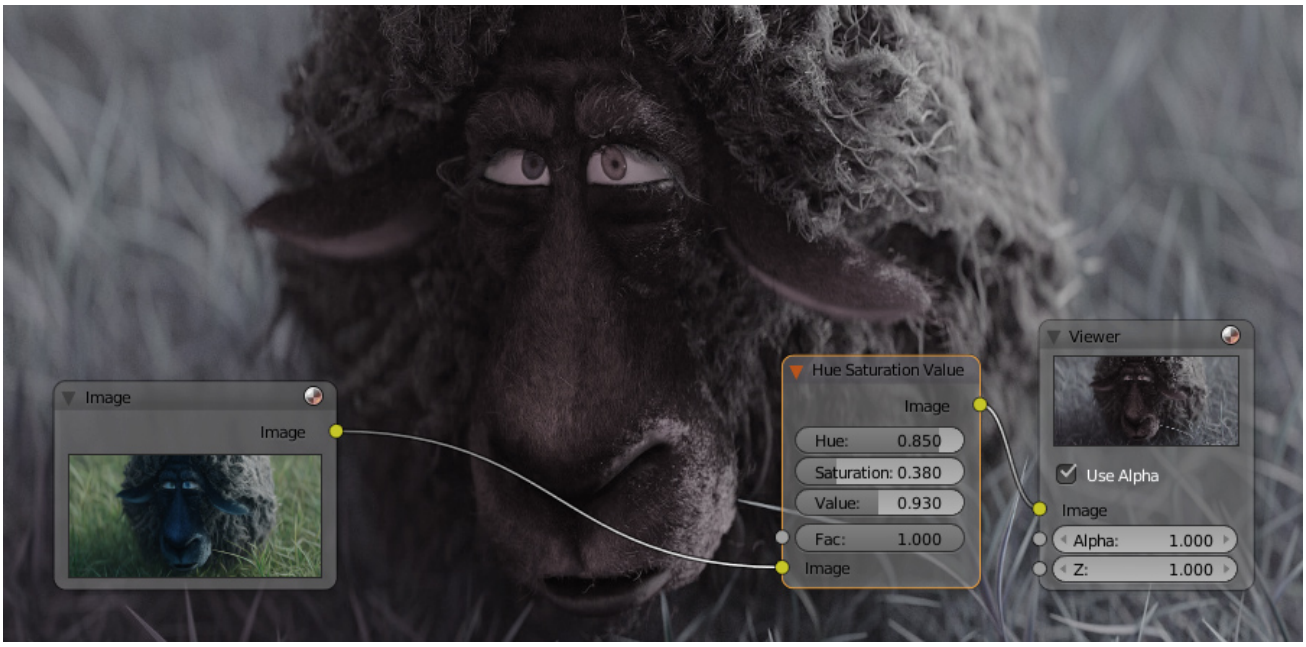
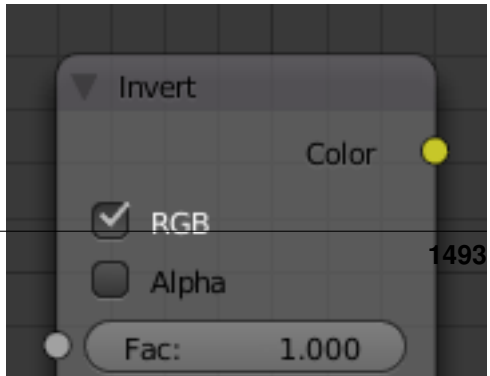


Fig. 2.2018: A basic example.

Invert Node

This node inverts the colors in the input image, producing a negative.



Inputs

Factor Controls the amount of influence the node exerts on the output image.

Color Standard image input.

Properties

In the compositing context this node has the following properties.

RGB De/activation of the color channel inversion.

Alpha De/activation of the alpha channel inversion.

Outputs

Color Standard image output.

Light Falloff Node

The *Light Falloff* node allows you to manipulate how light intensity decreases over distance. In reality light will always fall off quadratically; however, it can be useful to manipulate as a non-physically based lighting trick. Note that using Linear or Constant falloff may cause more light to be introduced with every global illumination bounce, making the resulting image extremely bright if many bounces are used.

Inputs

Strength Light strength before applying falloff modification.

Smooth Smooth intensity of light near light sources. This can avoid harsh highlights, and reduce global illumination noise. 0.0 corresponds to no smoothing; higher values smooth more. The maximum light strength will be strength/smooth.

Properties

This node has no properties.

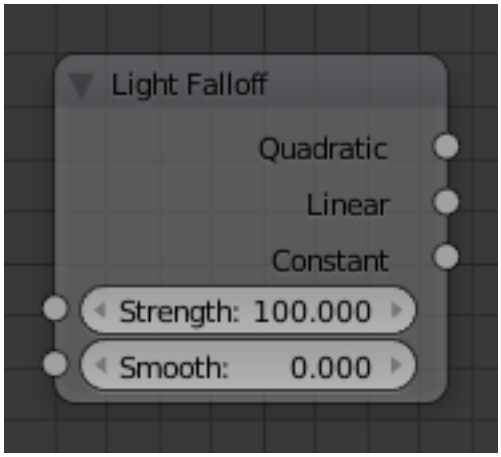


Fig. 2.2020: Light Falloff Node.

Outputs

Quadratic Quadratic light falloff; this will leave strength unmodified if smooth is 0.0 and corresponds to reality.

Linear Linear light falloff, giving a slower decrease in intensity over distance.

Constant Constant light falloff, where the distance to the light has no influence on its intensity.

Examples

Todo.

Mix Node

This node mixes images by working on the individual and corresponding pixels of the two input images. Called "MixRGB" in the shader and texture context.

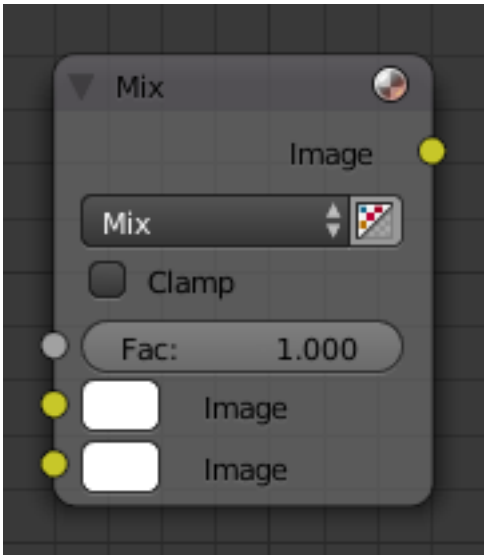


Fig. 2.2021: Mix Node.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image The background image. The image size and resolution sets the dimensions of the output image.

Image The foreground image.

Properties

Mix The Blend types could be selected in the select menu. See *Color Blend Modes* for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Dodge, Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Use Alpha If activated, by clicking on the *Color and Alpha* icon, the Alpha channel of the second image is used for mixing. When deactivated, the default, the icon background is a light gray. The alpha channel of the base image is always used.

Clamp Limit the highest color value to not exceed 1.

Outputs

Image Standard image output.

Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.

Some explanation of the mixing methods above might help you use the Mix node effectively:

Add adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.

Subtract Taking Blue away from white leaves Red and Green, which combined make Yellow. Taking Blue away from Purple leaves Red. Use this to desaturate an image. Taking away yellow makes an image bluer and more depressing.

Multiply Black (0.00) times anything leaves black. Anything times White (1.00) is itself. Use this to mask out garbage, or to colorize a black-and-white image.

Hue Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').

Mix Combines the two images, averaging the two.

Lighten Like bleach makes your whites whiter. Use with a mask to lighten up a little.

Difference Kinda cute in that it takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two verrry similar images to see what had been done to one to make it the other; sorta like a change log for images. You can use this to see a watermark (see *Watermark images*) you have placed in an image for theft detection.

Darken with the colors set here, is like looking at the world through rose-colored glasses.

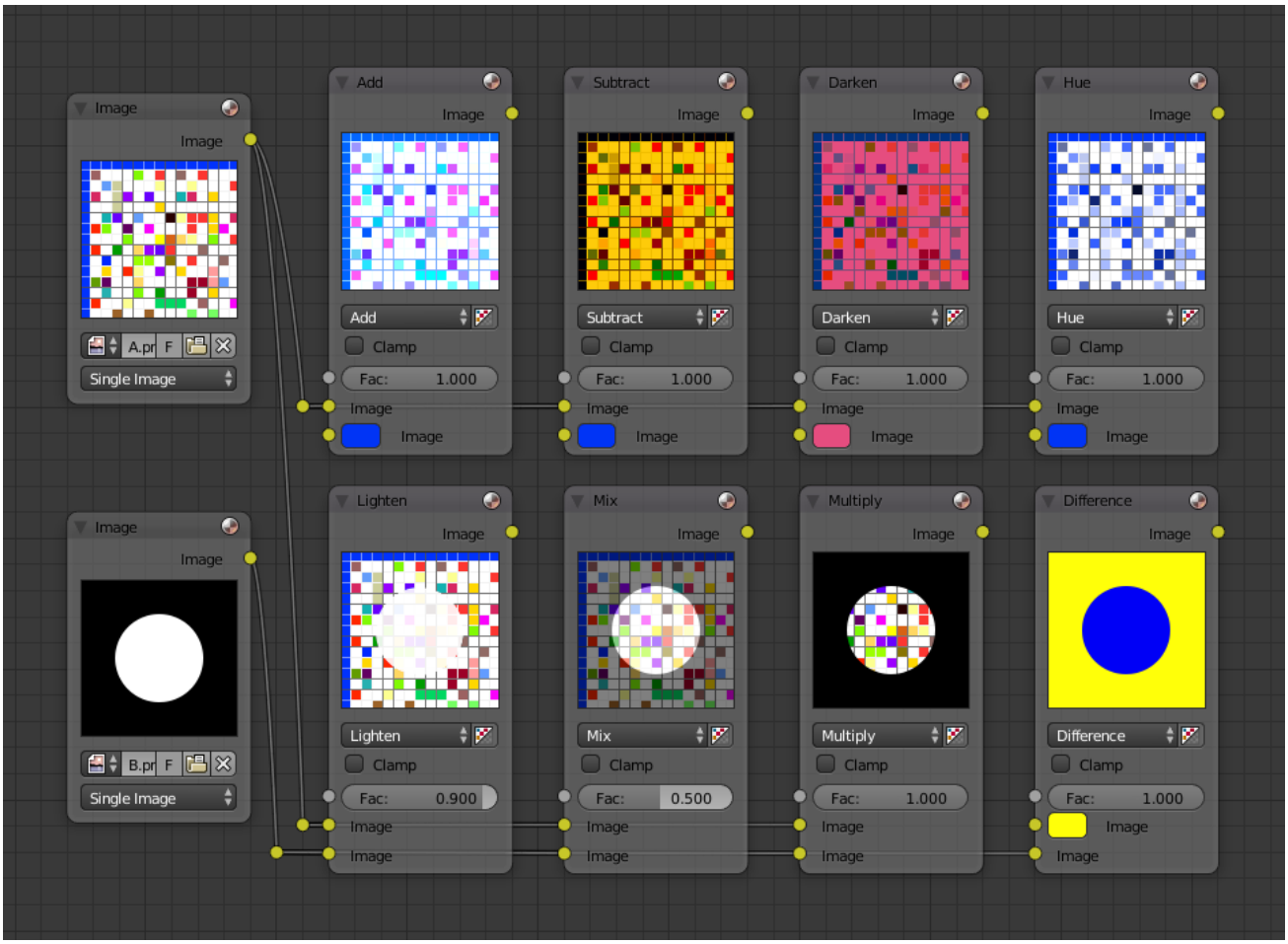


Image
Image
A.pr
F
Single Image
Add
Image
Add
Clamp
Fac: 1.000
Image
Image
Subtract
Image
Subtract
Clamp
Fac: 1.000
Image
Image
Darken
Image
Darken
Clamp
Fac: 1.000
Image
Image
Hue
Image
Hue
Clamp
Fac: 1.000
Image
Image
Image
Image
B.pr
F
Single Image
Lighten
Image
Lighten
Clamp
Fac: 0.900
Image
Image
Mix
Image
Mix
Clamp
Fac: 0.500
Image
Image
Multiply
Image
Multiply
Clamp
Fac: 1.000
Image
Image
Difference
Image
Difference
Clamp
Fac: 1.000
Image
Image

Contrast Enhancement

Here is a small map showing the effects of two other common uses for the RGB Curve: *Darken* and *Contrast Enhancement*. You can see the effect each curve has independently, and the combined effect when they are *mixed* equally.

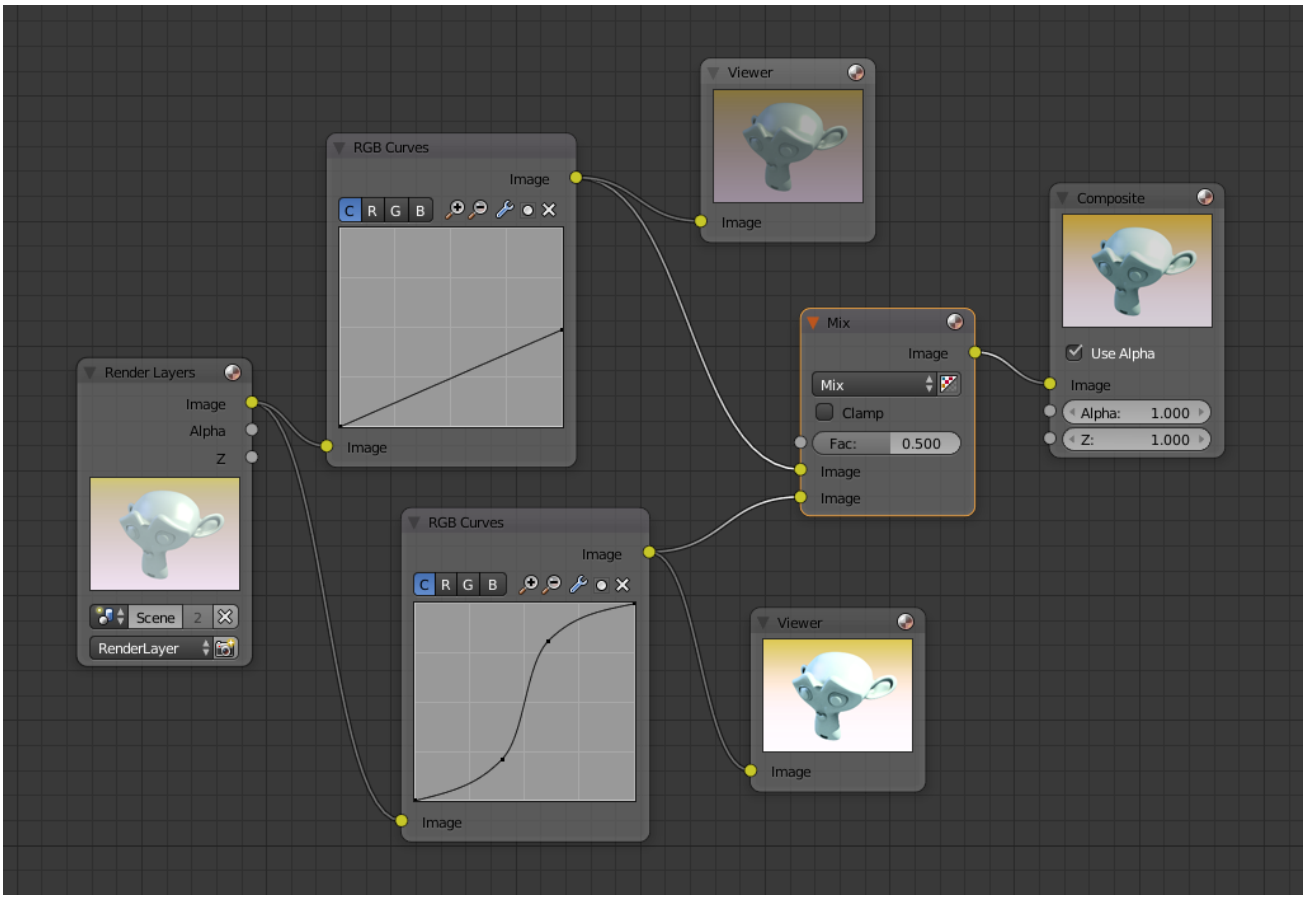


Fig. 2.2022: Example node setup showing “Darken”, “Enhance Contrast” and “Mix” nodes for composition.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast.

In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB (S shaped) curve scales the output such that middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus, whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better.

Watermark images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising. Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting service, or simply to track their image’s proliferation throughout the web. Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding Your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it is up to you. In the example below, we are encoding the watermark in a specific location in the image using the *Translate* node; this helps later because we only have to look at a specific location for the mark. We then use the RGB to BW node to convert the image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one-tenth of its original intensity. The Add node adds the corresponding pixels, make the ones containing the mark ever-so-slightly brighter.

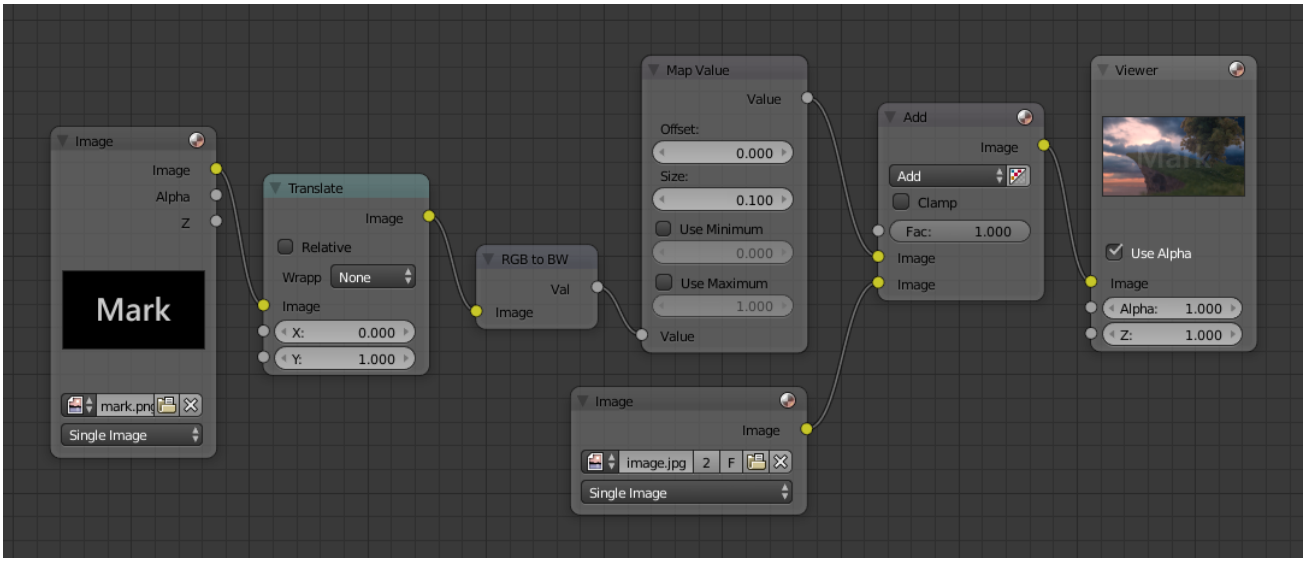


Fig. 2.2023: Embedding your mark in an Image using a Mark and Specific Position.

Of course, if you *want* people to notice your mark, do not scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Note: Additional uses

You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface, or as a special effect to make words appear on the possessed girl's forearm. yuk.

Decoding an Image for your Watermark

When you see an image that you think might be yours, use the node map below to compare it to your stock image (pre-watermarked original). In this map, the Mix node is set to Difference, and the Map Value node amplifies any difference. The result is routed to a viewer, and you can see how the original mark stands out, clear as a bell:

Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding map in one scene, and the decoding map in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to JPEG compression of 50%.

RGB Curves Node

This node allows color corrections for each color channel and levels adjustments in the compositing context.

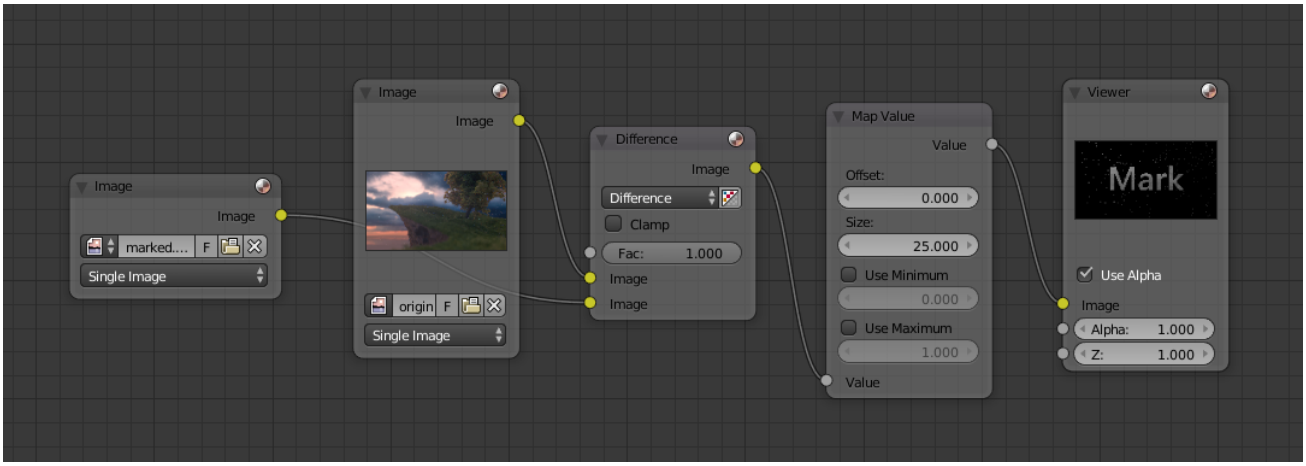


Fig. 2.2024: Checking an image for your watermark.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Black Level Defines the input color that is (linear) mapped to black.

White Level Defines the input color that is (linear) mapped to white.

Tip: To define the levels, use the *eye dropper* to select a color sample of a displayed image.

Properties

Channel Clicking on one of the channels displays the curve for each.

C (Combined RGB), R (Red), G (Green), B (Blue), L (Luminance)

Curve A Bézier curve that varies the input levels (x-axis) to produce an output level (y-axis). For the curve controls see: *Curve widget*.

Outputs

Image Standard image output.

Examples

Here are some common curves you can use to achieve desired effects:

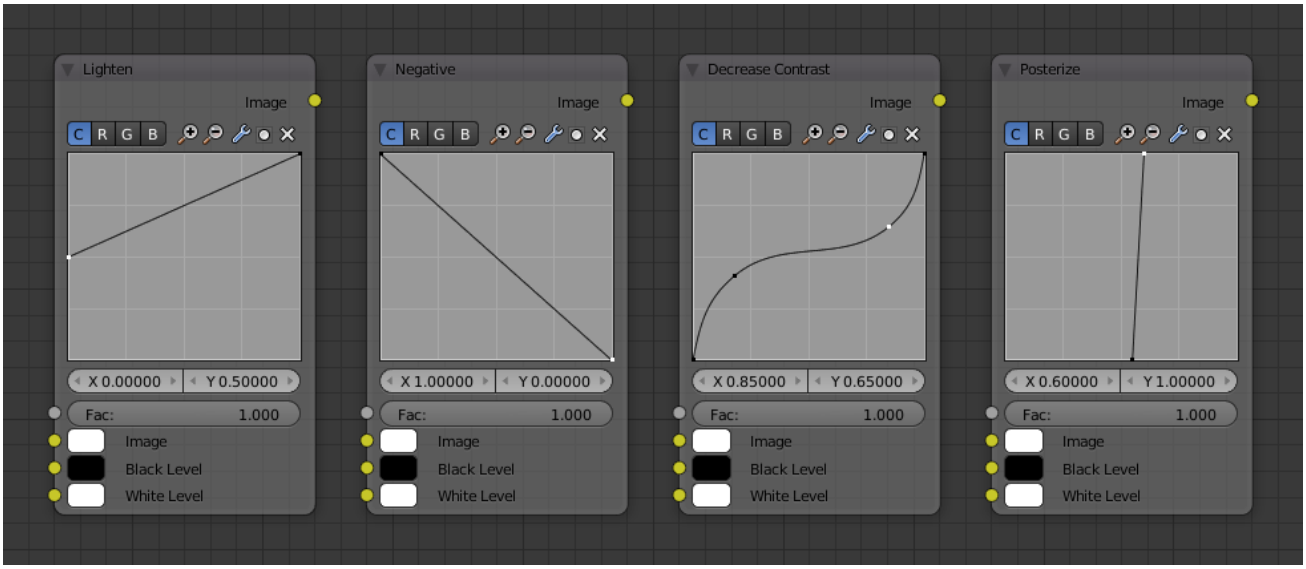


Fig. 2.2026: From left to right: 1. Lighten 2. Negative 3. Decrease Contrast 4. Posterize.

Color correction using Curves

In this example, the image has way too much red in it, so we run it through an RGB node and reduce the Red channel by about half.

We added a middle dot so we could make the line into a sideways exponential curve. This kind of curve evens out the amount of a color in an image as it reaches saturation. Also, read on for examples of the Darken and Contrast Enhancement curves.

Color correction using Black/White Levels

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the UV/Image editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side-effect of setting Black and White Levels.

Effects

Curves and Black/White Levels can also be used to completely change the colors of an image.

Note that e.g. setting Black Level to red and White Level to blue does not simply substitute black with red and white with blue as the example image might suggest. Levels do color scaling, not substitution, but depending on the settings they can result in the described color substitution.

(What really happens when setting Black Level to pure red and White Level to pure blue is that the red channel gets inverted, green gets reduced to zero and blue remains unchanged.)

Because of this, the results of setting arbitrary Black/White Levels or RGB curves is hard to predict, but can be fun to play with.

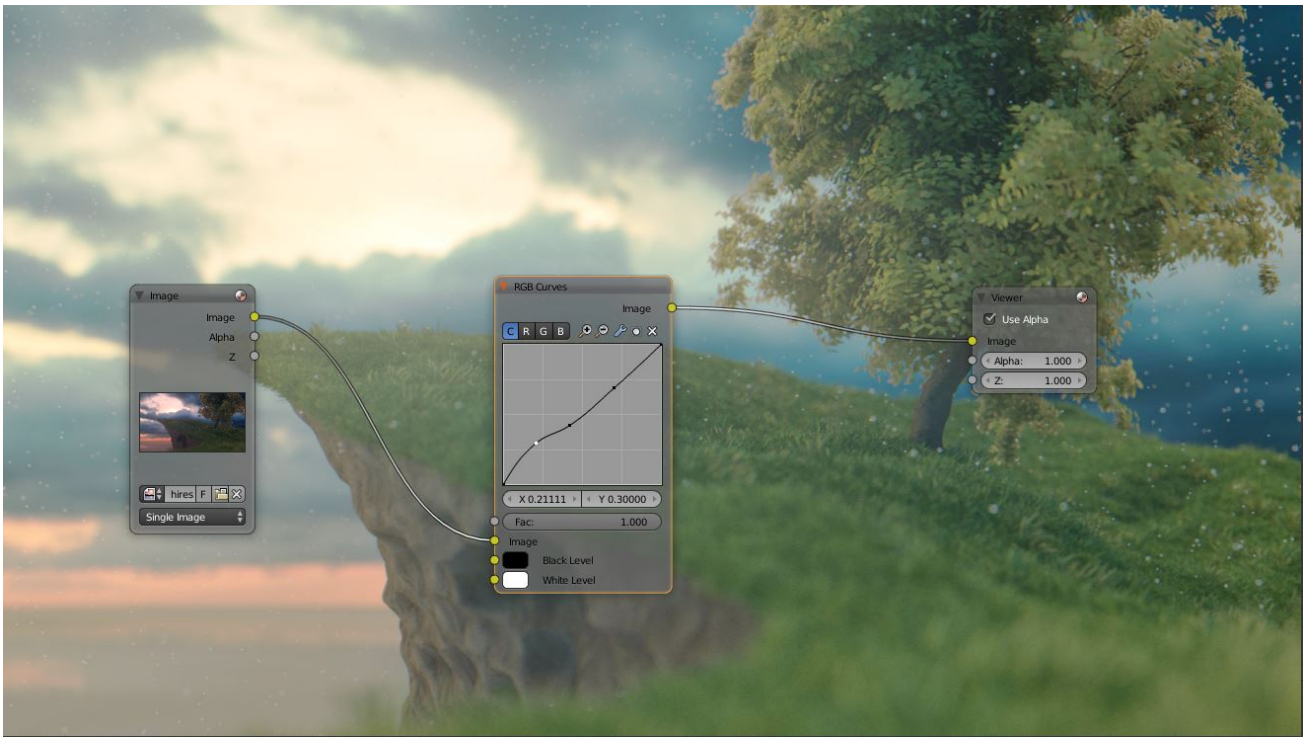


Fig. 2.2027: Color correction with curves.

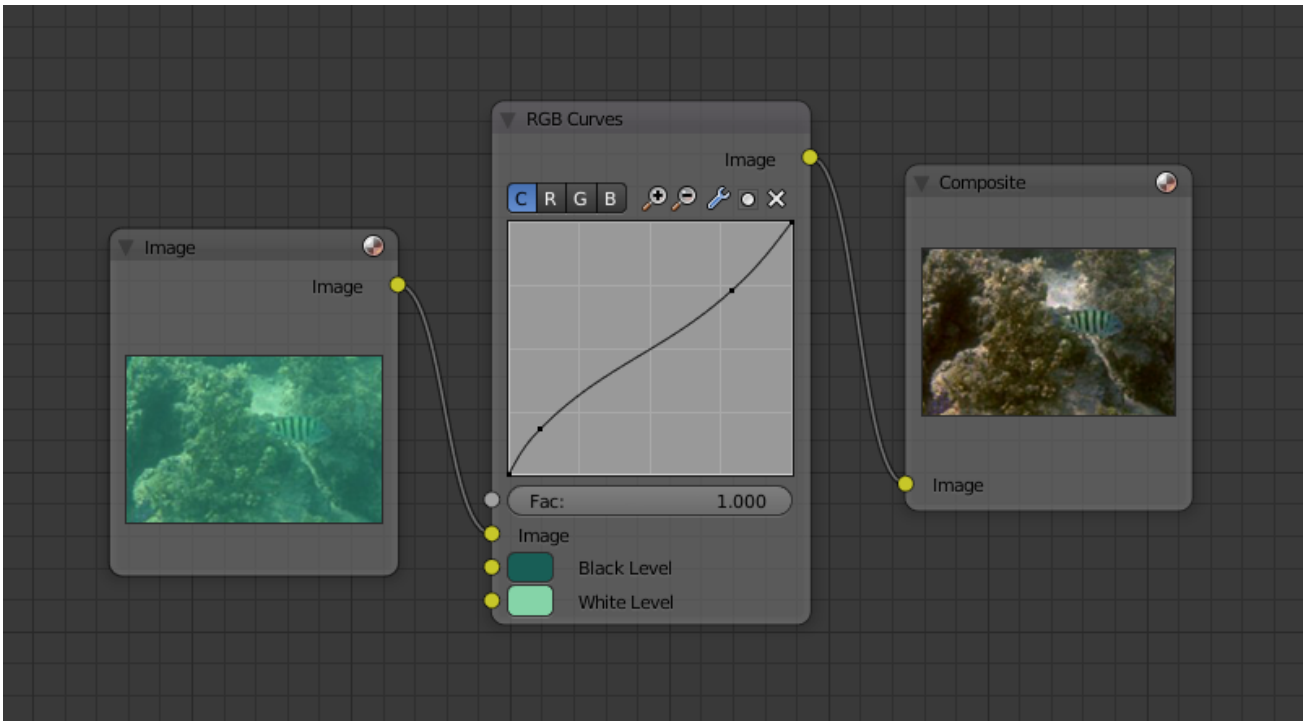


Fig. 2.2028: Color correction with Black/White Levels.

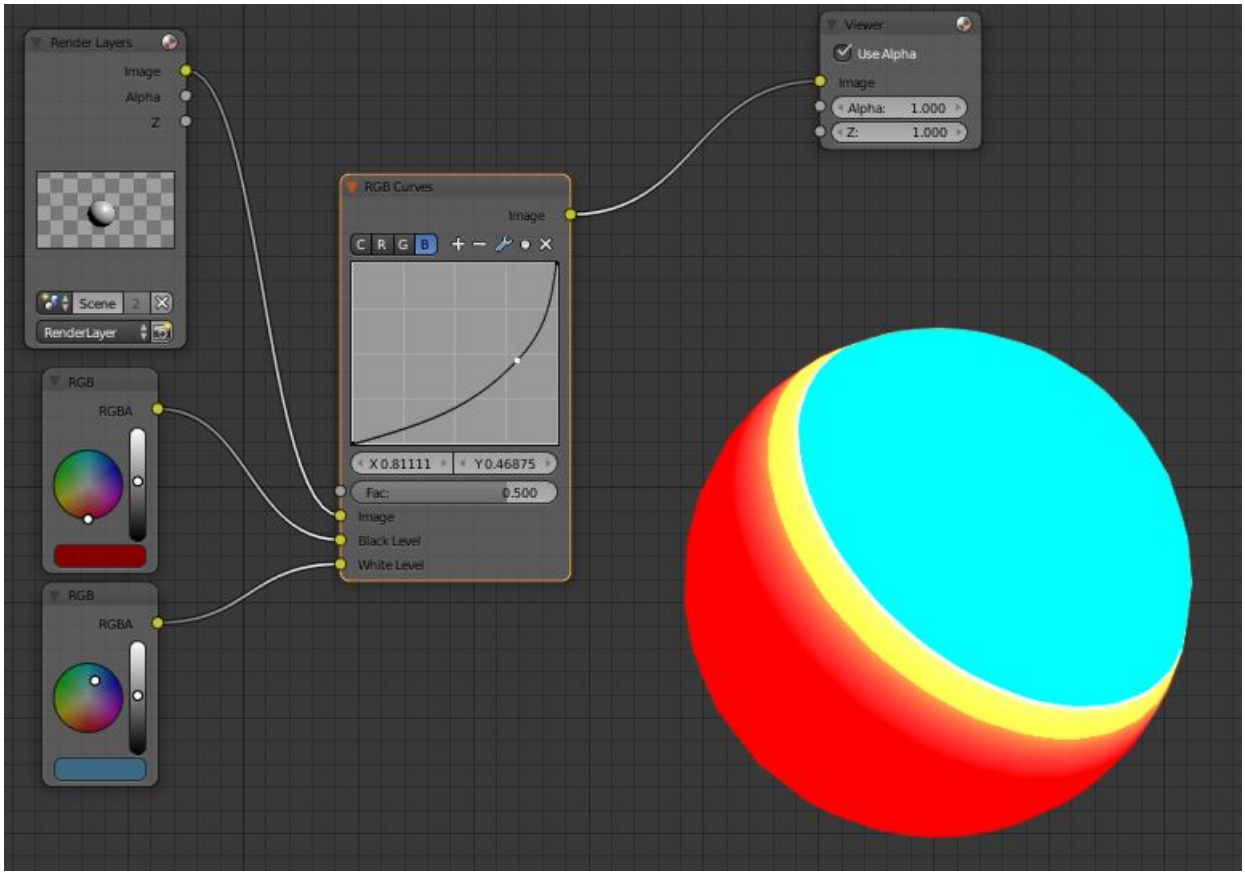


Fig. 2.2029: Changing colors.

Vector Nodes

Bump Node

The *Bump* node generates a perturbed normal from a height texture, for bump mapping. The height value will be sampled at the shading point and two nearby points on the surface to determine the local direction of the normal.

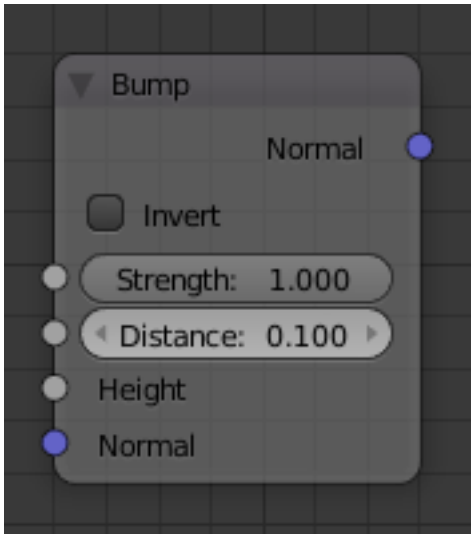


Fig. 2.2030: Bump Node.

Inputs

Strength Strength of the bump mapping effect, interpolating between no bump mapping and full bump mapping.

Distance Multiplier for the height value to control the overall distance for bump mapping.

Height Scalar value giving the height offset from the surface at the shading point; this is where you plug in textures.

Normal Standard normal input.

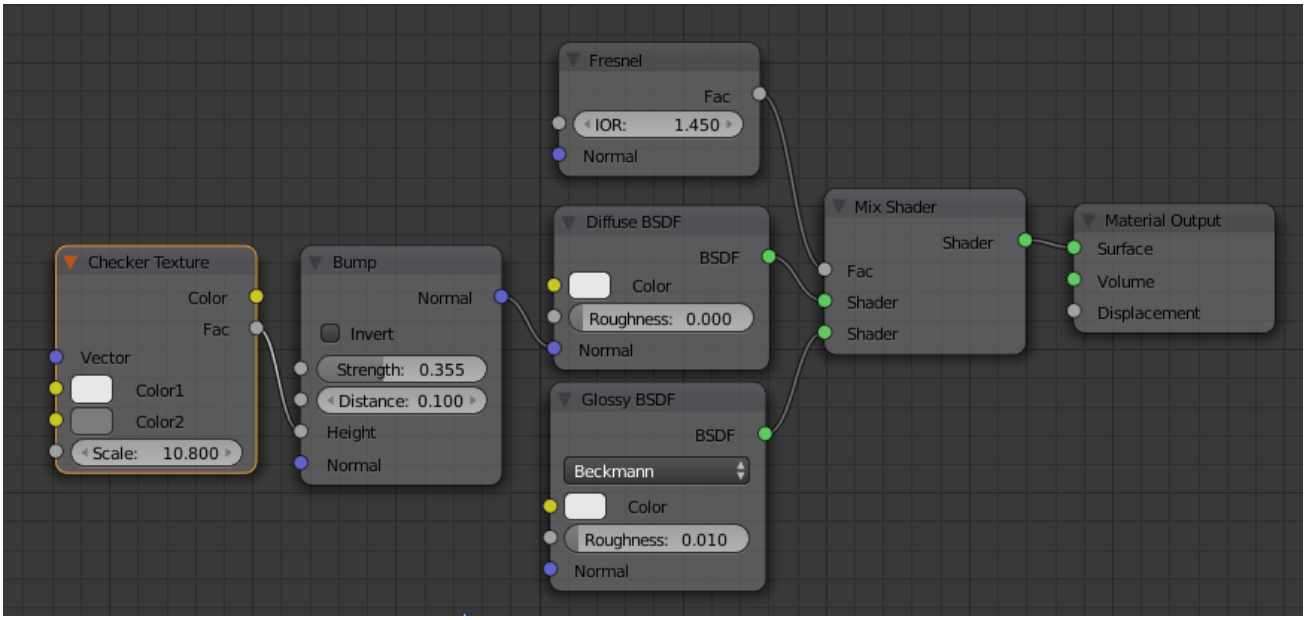
Properties

Invert Invert the bump mapping, to displace into the surface instead of out.

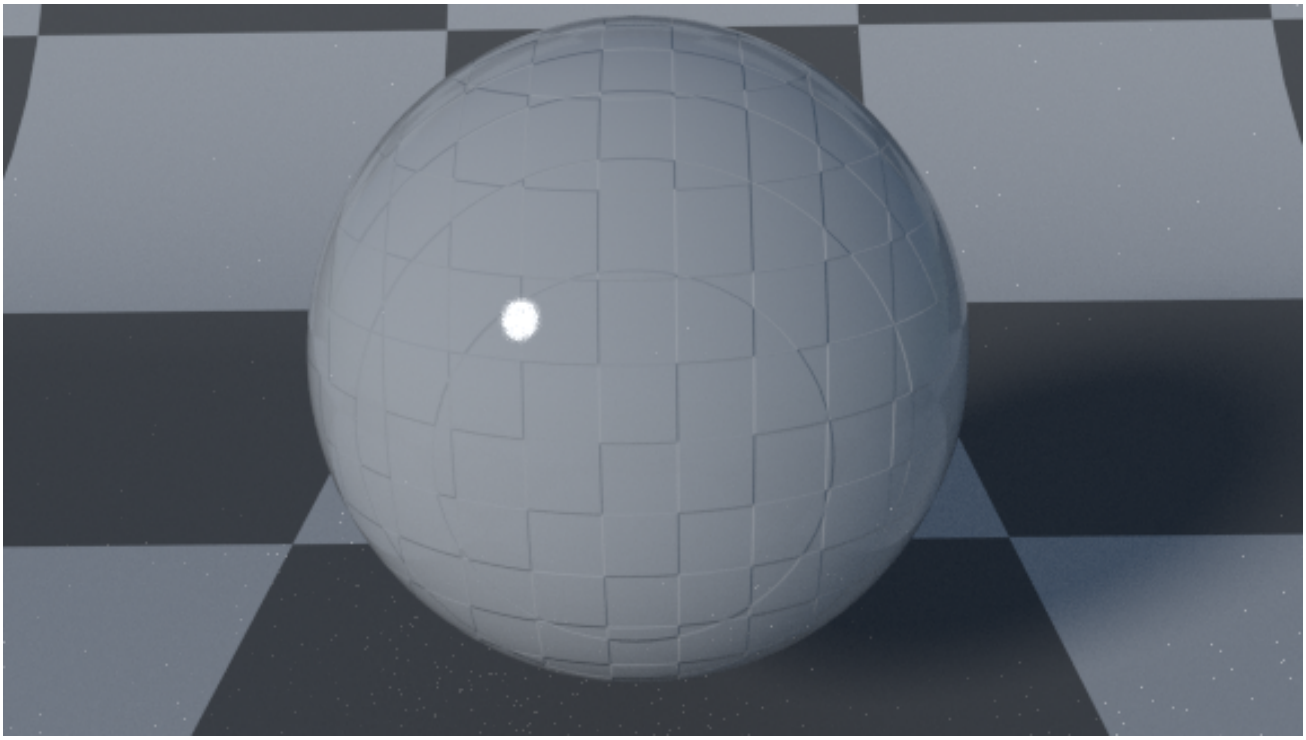
Outputs

Normal Standard normal output.

Examples



The above node setup will only bump the diffuse part of the shader, simulating a bumpy diffuse surface coated with a smooth glossy "glaze" layer.



Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.

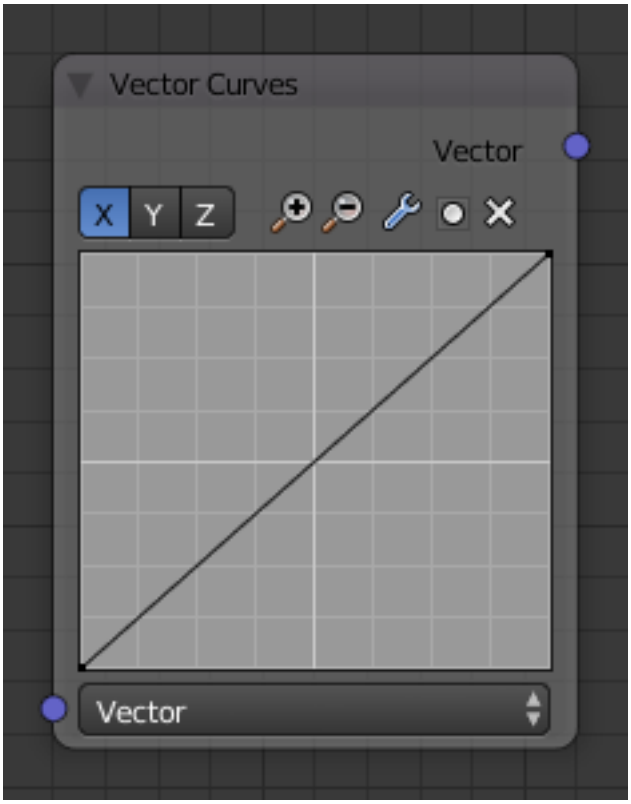


Fig. 2.2031: Vector Curves Node.

Inputs

In the shader context the node also has an additional Factor property.

Factor Controls the amount of influence the node exerts on the output vector.

Vector Standard vector input.

Properties

Channel X, Y, Z

Curve For the curve controls see: *Curve widget*.

Outputs

Vector Standard vector output.

Mapping Node

The *Mapping* nodes is used to transform a coordinate; typically used for modifying texture coordinates.

Inputs

Vector Vector to be transformed.

Properties

Vector type Todo.

Texture Todo.

Point Todo.

Vector Todo.

Normal Todo.

Location Vector translation.

Rotation Rotation of the vector along XYZ axes.

Scale Scale of the vector.

Min Todo.

Max Todo.

Outputs

Vector Transformed vector.

Examples

Todo.

Normal Node

The Normal node generates a normal vector and a dot product.

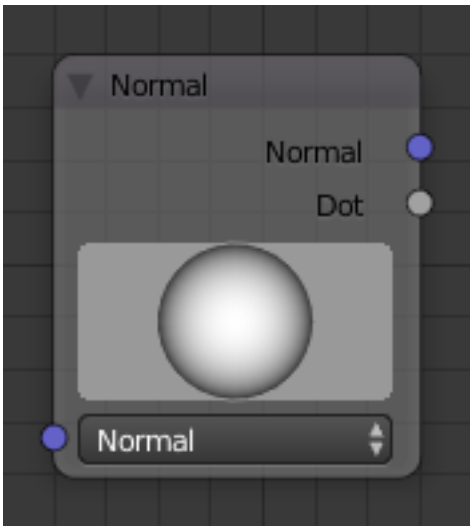


Fig. 2.2032: Normal Node.

Inputs

Normal Normal vector input.

Properties

Normal Direction To manually set a fixed normal direction vector. LMB click and drag on the sphere to set the direction of the normal.

Outputs

Normal Normal vector output.

Dot Dot product output. The dot product is a scalar value.

- If two normals are pointing in the same direction the dot product is 1.
- If they are perpendicular the dot product is zero (0).
- If they are antiparallel (facing directly away from each other) the dot product is -1.

Normal Map Node

The *Normal Map* node generate a perturbed normal from an RGB normal map image. This is usually chained with an *Image Texture* node in the color input, to specify the normal map image. For tangent space normal maps, the UV coordinates for the image must match, and the image texture should be set to *Non-Color* mode to give correct results.

Inputs

Strength Strength of the normal mapping effect.

Color RGB color that encodes the normal in the specified space.

Properties

Space The input RGB color can be in one of three spaces: Tangent, Object and World space. Tangent space normal maps are the most common, as they support object transformation and mesh deformations. Object space normal maps keep sticking to the surface under object transformations, while World normal maps do not.

UV Map Name of the UV map to derive normal mapping tangents from. When chained with an Image Texture node, this UV map should be the same as the UV map used to map the texture.

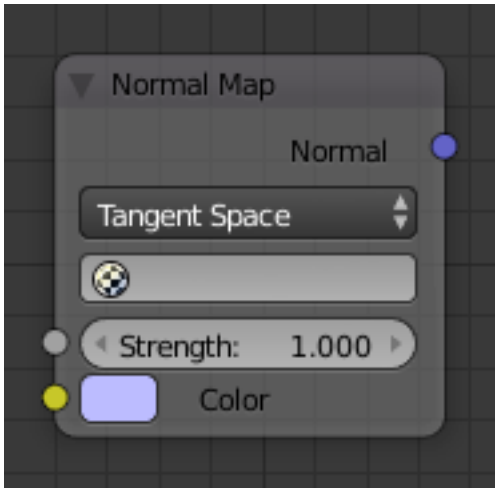


Fig. 2.2033: Normal Map Node.

Outputs

Normal Normal that can be used as an input to BSDF nodes.

Examples

Todo.

Vector Transform Node

The *Vector Transform* node allows converting a Vector, Point or Normal between World <=> Camera <=> Object coordinate space.

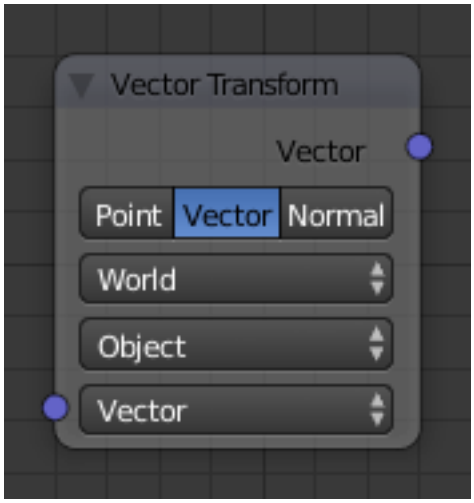


Fig. 2.2034: Vector Transform node.

Inputs

Vector Input Standard vector input.

Properties

Type Specifies the input/output type: Vector, Point or Normal.

Convert From Coordinate Space to convert from: World, Object or Camera.

Convert To Coordinate Space to convert to: World, Object or Camera.

Outputs

Vector Output The transformed output vector.

Examples

Todo.

Converter Nodes

Blackbody Node

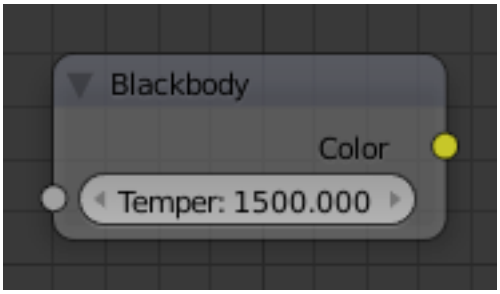
The *Blackbody* node converts a blackbody temperature to RGB value. This can be useful for materials that emit light at natural occurring frequencies.

Inputs

Temperature The temperature in Kelvin.

Properties

This node has no properties.



Outputs

Color RGB color output.

Examples

Fig. 2.2036: Example of the color ranges of the Blackbody node.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors with the use of a gradient.

Inputs

Factor The Factor input is used as an index for the color ramp.

Properties

Color Ramp For controls see *Color Ramp Widget*.

Outputs

Image Standard image output.

Alpha Standard alpha output.

Examples

Creating an Alpha Mask

A powerful but often overlooked feature of the Color Ramp is to create an Alpha Mask, or a mask that is overlaid on top of another image, and, like a mask, allows some of the background to show through. The example map below shows how to use the Color Ramp node to do this:

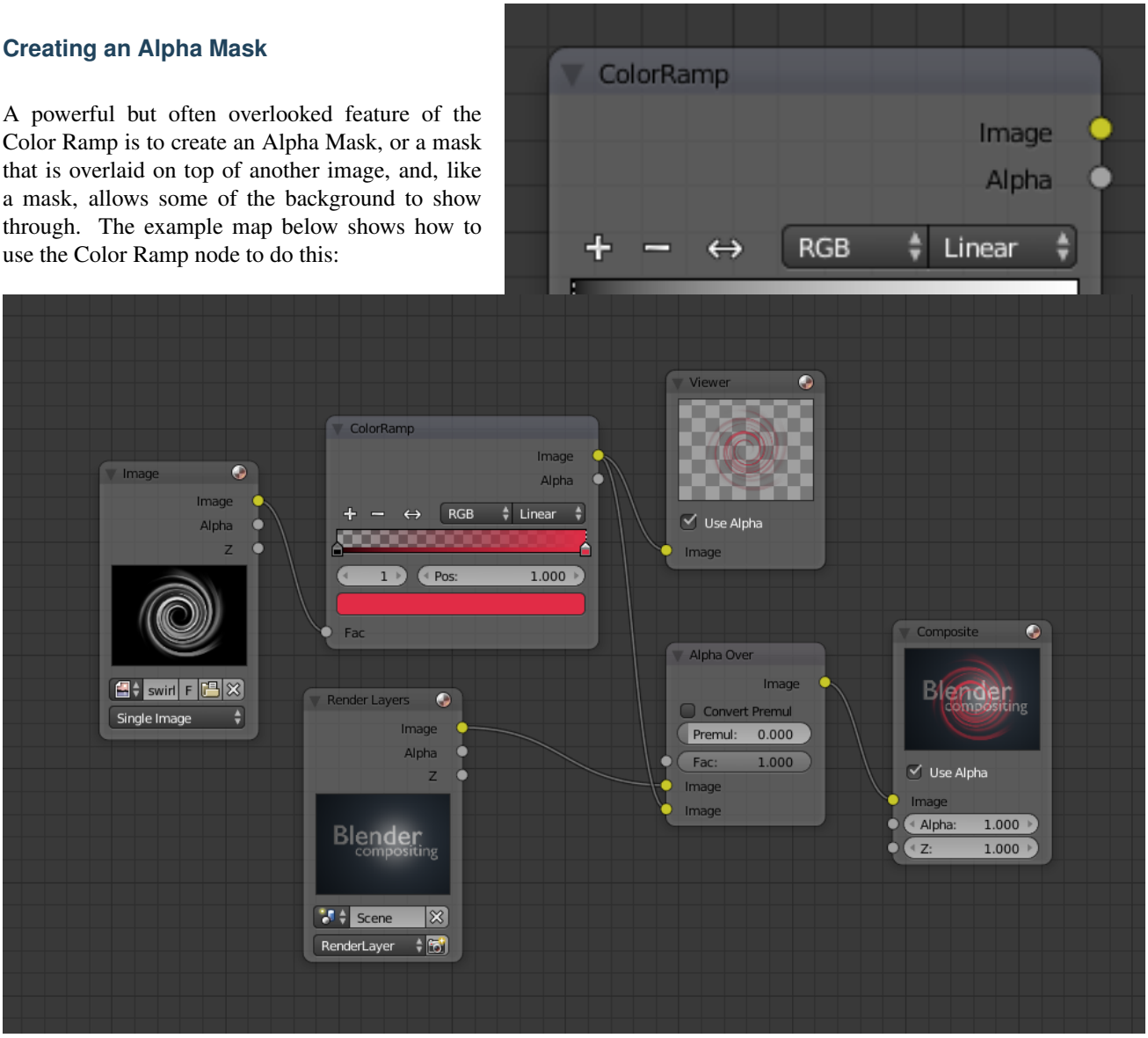


Fig. 2.2038: Using the Color Ramp node to create an alpha mask.

In the map above, a black and white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*. (Technically, we should have converted the image to a value using

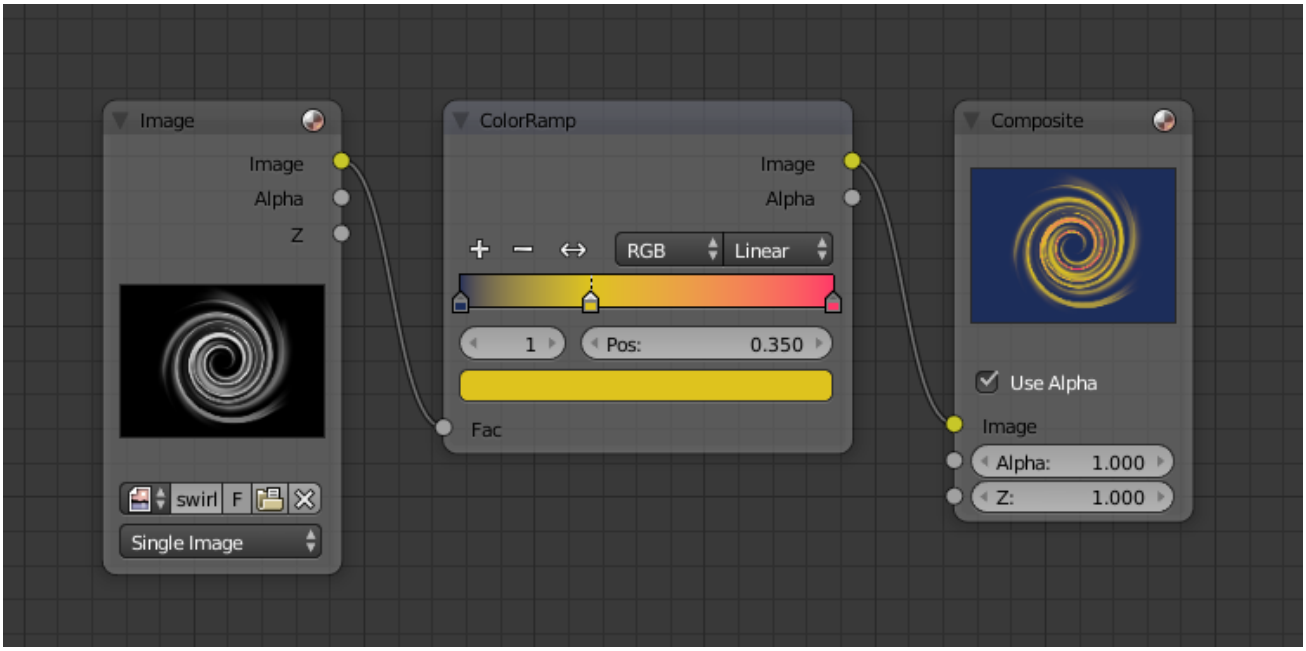
the RGB-to-BW node, buy hey, this works just as well since we are using a BW image as input.)

We have set the Color Ramp node to a purely transparent color on the left end of the spectrum, and a fully Red color on the right. As seen in the viewer, the Color Ramp node puts out a mask that is fully transparent where the image is black. Black is zero, so Color Ramp uses the color at the left end of the spectrum, which we have set to transparent. The Color Ramp image is fully red and opaque where the image is white (1.00).

We verify that the output image mask is indeed transparent by overlaying it on top of a other image.

Colorizing an Image

The real power of Color Ramp is that multiple colors can be added to the color spectrum. This example compositing map takes a boring BW image and makes it a flaming swirl!



In this example, we have mapped the shades of gray in the input image to three colors, blue, yellow, and red, all fully opaque (Alpha of 1.00). Where the image is black, Color Ramp substitutes blue, the currently selected color. Where it is some shade of gray, Color Ramp chooses a corresponding color from the spectrum (bluish, yellow, to reddish). Where the image is fully white, Color Ramp chooses red.

Combine/Separate Nodes

All of these nodes do essentially the same thing:

- Separate: Split out an image into its composite color channels.
- Combine: Re/combine an image from it is composite color channels.

This nodes could be use this to manipulate on each color channel independently. Each type is differentiate in the applied *color space*.

In compositing and texture context each node supports the Alpha channel. In the texture context only RGB color space is available. In the shading context of the Blender internal adds HSV and the Cycles shading context offers an additional pair of nodes to combine/separate a vector (XYZ).

The Combine nodes could also be used to input single color values. For RGBA and HSVA color spaces it is recommended to use the *RGB Node*. Some common operation could easier executed with the *Color Nodes*.

Separate/Combine RGBA Node

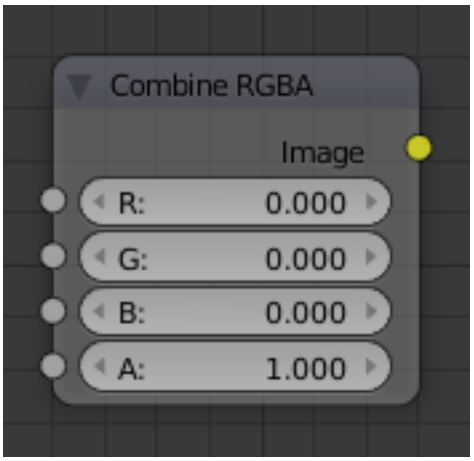


Fig. 2.2039: Combine RGBA Node.

Input/ Output

Image Standard image in/output.

- R (Red)
- G (Green)
- B (Blue)
- A (Alpha)

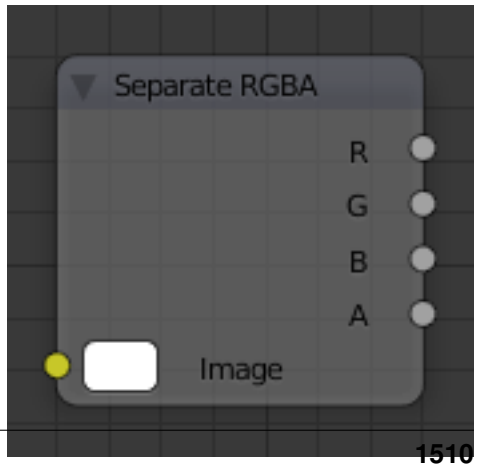
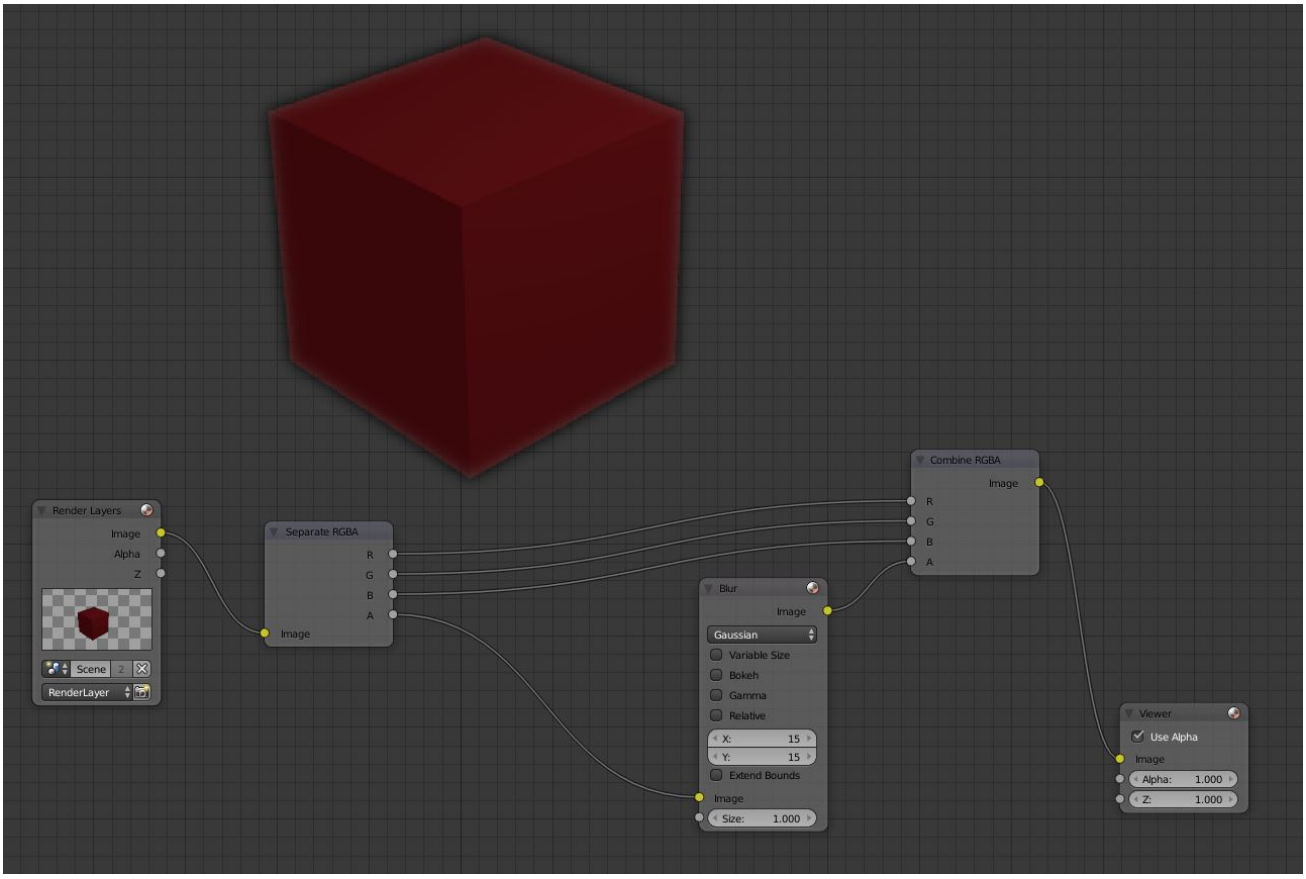


Fig. 2.2040: Separate RGBA Node.

Properties

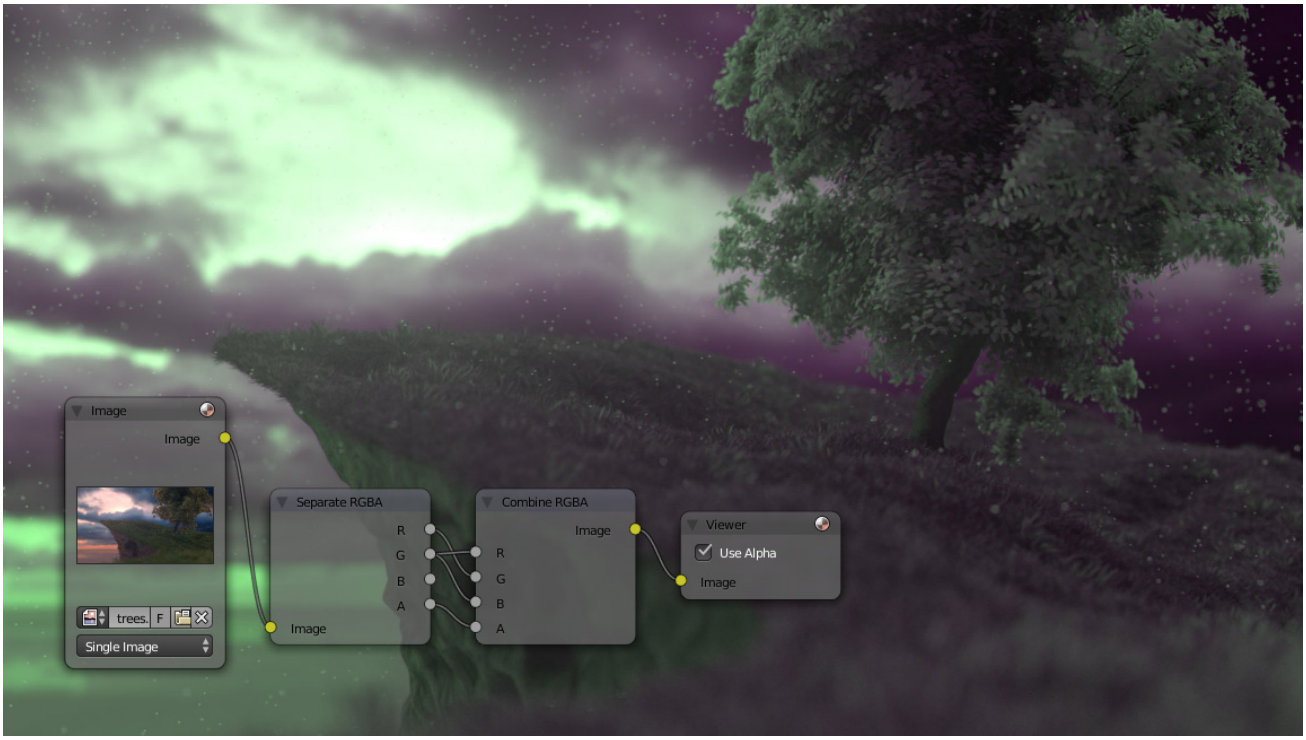
This node has no properties.

Examples



In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like anti-aliasing but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a broader scale will make the object appear to "phase" in and out, as an "out-of-phase" time-traveling sync effect.

In this node set up, we make all the reds become green, and all the green both Red and Blue, and remove Blue from the image completely.



Separate/Combine HSVA Nodes

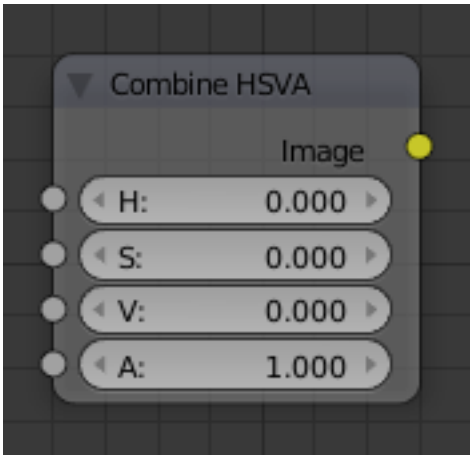


Fig. 2.2041: Combine HSVA Node.

Input/ Output

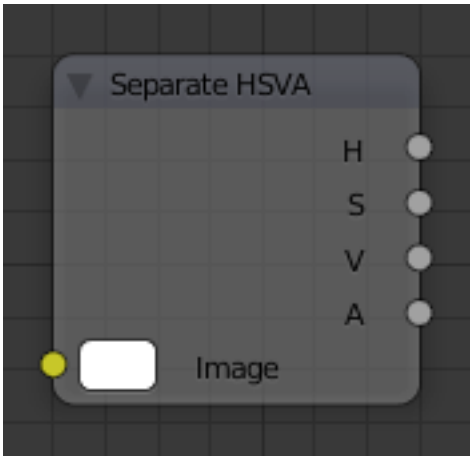


Fig. 2.2042: Separate HSVA Node.

Image Standard image in/output.

- H (Hue)
- S (Saturation)
- V (Value)
- A (Alpha)

Properties

This node has no properties.

Separate/Combine YUVA Node

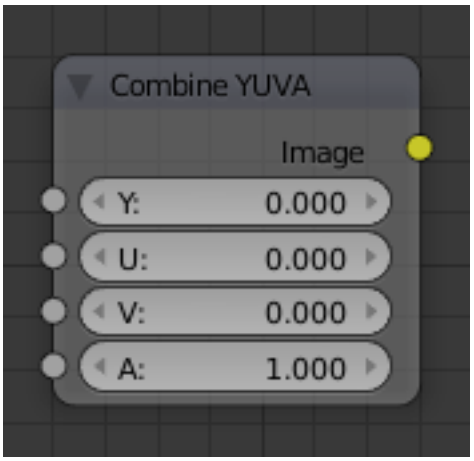


Fig. 2.2043: Combine YUVA Node.

Input/ Output

Image Standard image in/output.

- Y (Luminance)
- U (U chrominance)
- V (V chrominance)
- A (Alpha)

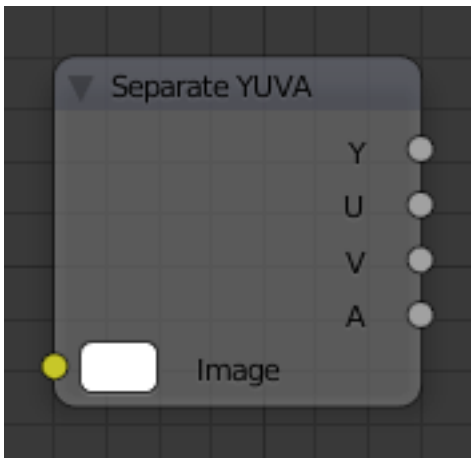


Fig. 2.2044: Separate YUVA Node.

Properties

This node has no properties.

Separate/Combine YCbCrA Node

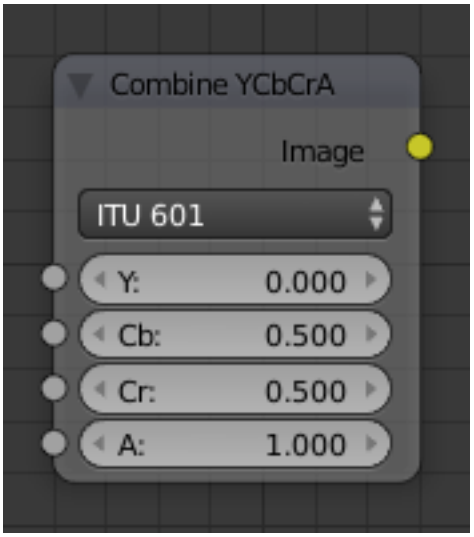


Fig. 2.2045: Combine YCbCrA Node.

Input/ Output

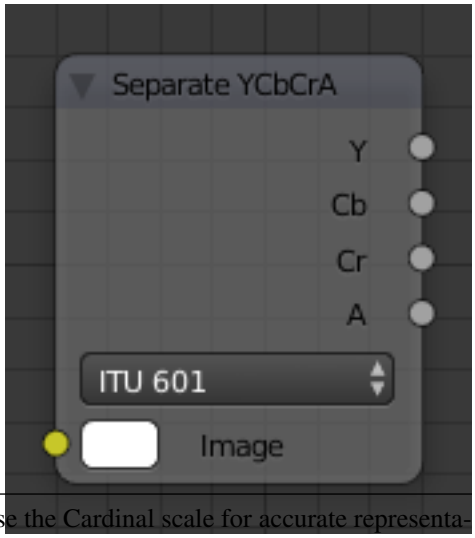


Fig. 2.2046: Separate YCbCrA Node.

Image Standard image in/output.

- Y (Luminance)
- Cb (Chrominance Blue)
- Cr (Chrominance Red)
- A (Alpha)

Properties

Mode ITU 601, ITU 709, Jpeg

Tip: If running these channels through a Color Ramp node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives high-contrast effect.

Math Node

This node performs math operations.

Inputs

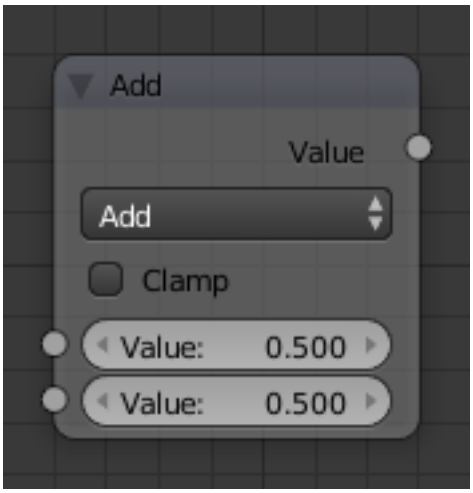


Fig. 2.2047: Math node.

Value First numerical value. The trigonometric functions accept values in radians.

Value Second numerical value. This value is **not** used in functions that accept only one parameter like the trigonometric functions, Round and Absolute.

Properties

Operation Add, Subtract, Multiply, Divide, Sine, Cosine, Tangent, Arcsine, Arccosine, Arctangent, Power, Logarithm, Minimum, Maximum, Round, Less Than, Greater Than, Modulo, Absolute.

Clamp Limits the output to the range (0 to 1). See *clamp*.

Outputs

Value Numerical value output.

RGB to BW Node

This node maps a RGB color image to a grayscale by the luminance.

Inputs

Image Color image input.

Properties

This node has no properties.

Outputs

Value Grayscale value output.

Vector Math Node

The *Vector Math* node performs the selected math operation on vectors. Select the math function by clicking the up-down selector where the "Add" selection is shown.

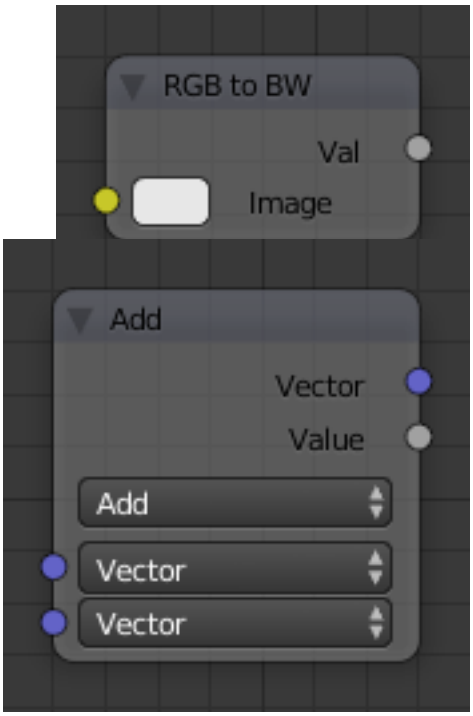


Fig. 2.2049: Vector Math Node.

Inputs

Vector Input vector 1 (upper). The value can be provided by another node or set manually.

Vector Input vector 2 (lower). The value can be provided by another node or set manually.

Properties

Operation Selector the math function for conversion.

Add Adding input 1 and 2.

Subtract Subtracting input 1 and 2.

Average Averaging input 1 and 2.

Dot Product Algebraic operation that takes two equal-length sequences of vectors 1 and 2 and returns a single number. The Result is a scalar.

Cross Product Geometric binary operation on two vectors 1 and 2 in three-dimensional space. It results in a vector which is perpendicular to both and therefore normal to the plane containing them. The Result is a vector.

Normalize Normalizing input 1 and 2.

Outputs

Vector Output vector, converted by the node.

Value Output value, converted by the node.

Example

Todo.

Wavelength Node

The *Wavelength* node converts a wavelength value to a RGB value. This can be used to achieve a specific color on the light spectrum.

Fig. 2.2050: Wavelength Node.

Inputs

Wavelength The color wavelength from 380 to 780 nanometers.

Properties

This node has no inputs.

Outputs

Color RGB color output.

Examples

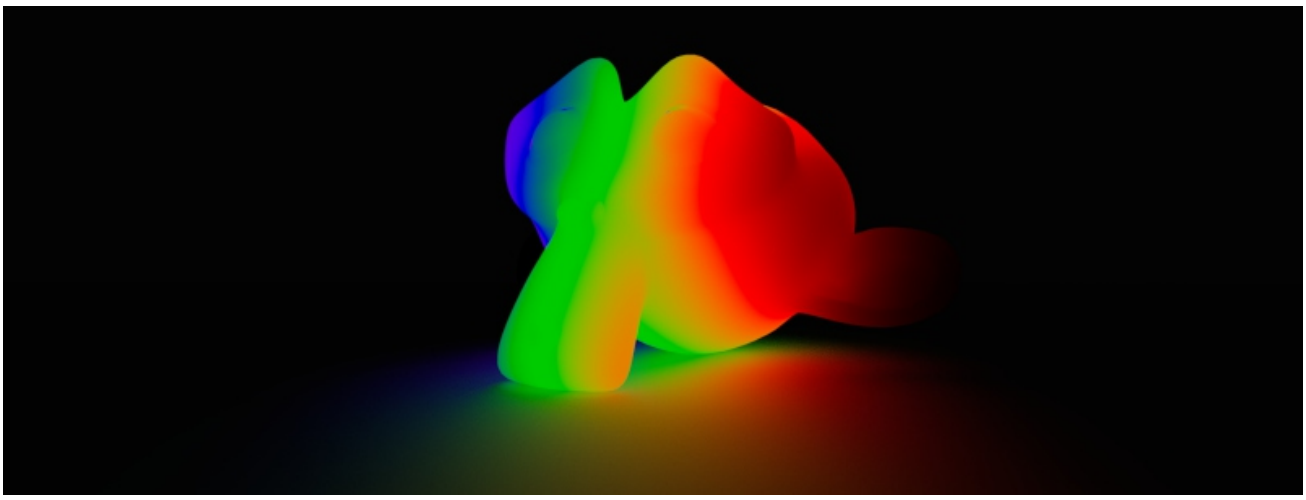


Fig. 2.2051: Example of Wavelength node.

Script Node

OSL was designed for node-based shading, and *each* OSL shader corresponds to *one* node in a node setup. To add an OSL shader, add a script node and link it to a text data-block or an external file. Input and output sockets will be created from the shader parameters on clicking the update button in the node or the text editor.

OSL shaders can be linked to the node in a few different ways. With the *Internal* mode, a text data-block is used to store the OSL shader, and the OSO bytecode is stored in the node itself. This is useful for distributing a blend-file with everything packed into it.

The *External* mode can be used to specify a `.osl` file on disk, and this will then be automatically compiled into a `.oso` file in the same directory. It is also possible to specify a path to a `.oso` file, which will then be used directly, with compilation done manually by the user. The third option is to specify just the module name, which will be looked up in the shader search path.

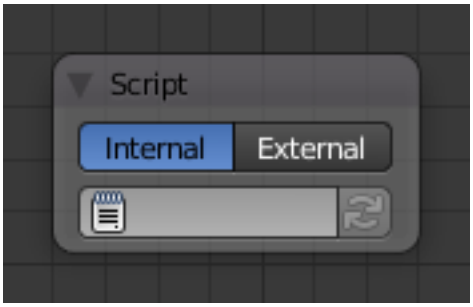


Fig. 2.2052: Script Node.

The shader search path is located in the same place as the scripts or configuration path, under:

Linux

```
$HOME/.config/blender/2.78/shaders/
```

MS-Windows

```
C:\Users\$user\AppData\Roaming\Blender Foundation\Blender\2.78\shaders\
```

macOS

```
/Users/$USER/Library/Application Support/Blender/2.78/shaders/
```

Tip: For use in production, we suggest to use a node group to wrap shader script nodes, and link that into other blend-files. This makes it easier to make changes to the node afterwards as sockets are added or removed, without having to update the script nodes in all files.

Writing Shaders

For more details on how to write shaders, see the OSL specification. Here is a simple example:

```
shader simple_material(
    color Diffuse_Color = color(0.6, 0.8, 0.6),
    float Noise_Factor = 0.5,
    output closure color BSDF = diffuse(N))
{
    color material_color = Diffuse_Color * mix(1.0, noise(P * 10.0), Noise_Factor);
    BSDF = material_color * diffuse(N);
}
```

Closures

OSL is different from, for example, RSL or GLSL, in that it does not have a light loop. There is no access to lights in the scene, and the material must be built from closures that are implemented in the render engine itself. This is more limited, but also makes it possible for the render engine to do optimizations and ensure all shaders can be importance sampled.

The available closures in Cycles correspond to the shader nodes and their sockets; for more details on what they do and the meaning of the parameters, see the *shader nodes manual*.

BSDF

- `diffuse(N)`
- `oren_nayar(N,roughness)`
- `diffuse_ramp(N,colors[8])`
- `phong_ramp(N,exponent,colors[8])`
- `diffuse_toon(N,size,smooth)`
- `glossy_toon(N,size,smooth)`
- `translucent(N)`
- `reflection(N)`
- `refraction(N,ior)`
- `transparent()`

- `microfacet_ggx(N,roughness)`
- `microfacet_ggx_aniso(N,T,ax,ay)`
- `microfacet_ggx_refraction(N,roughness,ior)`
- `microfacet_beckmann(N,roughness)`
- `microfacet_beckmann_aniso(N,T,ax,ay)`
- `microfacet_beckmann_refraction(N,roughness,ior)`
- `ashikhmin_shirley(N,T,ax,ay)`
- `ashikhmin_velvet(N,roughness)`

Hair

- `hair_reflection(N,roughnessu,roughnessv,T,offset)`
- `hair_transmission(N,roughnessu,roughnessv,T,offset)`

BSSRDF

- `bssrdf_cubic(N,radius,texture_blur,sharpness)`
- `bssrdf_gaussian(N,radius,texture_blur)`

Volume

- `henyey_greenstein(g)`
- `absorption()`

Other

- `emission()`
- `ambient_occlusion()`
- `holdout()`
- `background()`

Attributes

Some object, particle and mesh attributes are available to the built-in getattribute() function. UV maps and vertex colors can be retrieved using their name. Other attributes are listed below:

`geom:generated` Generated texture coordinates.

`geom:uv` Default render UV map.

`geom:dupli_generated` For instances, generated coordinate from duplicator object.

`geom:dupli_uv` For instances, UV coordinate from duplicator object.

`geom:trianglevertices` 3 vertex coordinates of the triangle.

`geom:numpolyvertices` Number of vertices in the polygon (always returns three currently).

`geom:polyvertices` Vertex coordinates array of the polygon (always three vertices currently).

`geom:name` Name of the object.

geom:is_curve Is object a strand or not.

geom:curve_intercept Point along the strand, from root to tip.

geom:curve_thickness Thickness of the strand.

geom:curve_tangent_normal Tangent Normal of the strand.

path:ray_length Ray distance since last hit.

object:location Object location.

object:index Object index number.

object:random Per object random number generated from object index and name.

material:index Material index number.

particle:index Particle instance number.

particle:age Particle age in frames.

particle:lifetime Total lifespan of particle in frames.

particle:location Location of the particle.

particle:size Size of the particle.

particle:velocity Velocity of the particle.

particle:angular_velocity Angular velocity of the particle.

Trace

We support the `trace(point pos,vector dir,...)` function, to trace rays from the OSL shader. The "shade" parameter is not supported currently, but attributes can be retrieved from the object that was hit using the `getmessage("trace",..)` function. See the OSL specification for details on how to use this.

This function cannot be used instead of lighting; the main purpose is to allow shaders to "probe" nearby geometry, for example to apply a projected texture that can be blocked by geometry, apply more "wear" to exposed geometry, or make other ambient occlusion-like effects.

World

The world environment can emit light, ranging from a single solid color, physical sky model, to arbitrary textures.

Fig. 2.2053: Lighting with an HDR image.

Surface Shader

The surface shader defines the light emission from the environment into the scene. The world surface is rendered as if it is very distant from the scene, and as such there is no two-way interacting between objects in the scene and the environment, only light coming in. The only shader accepted is the Background node with a color input and strength factor for the intensity of the light.

Image Based Lighting

For image based lighting, use the Environment Texture node rather than the Image Texture node for correct mapping. This supports *Equirectangular* (also known as Lat/Long) for environment maps, and *Mirror Ball* mapping for converting photos of mirror balls to environment maps.

Volume Shader

A volume shader can be applied to the entirely world, filling the entire space.

Currently this is most useful for night time or other dark scenes, as the world surface shader or sun lamps will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However, for modeling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Ambient Occlusion

Ambient occlusion is a lighting method based on how much a point on a surface is occluded by nearby surfaces. This is a trick that is not physically accurate, but it is useful to emphasize shapes of surfaces, or as a cheap way to get an effect that looks a bit like indirect lighting.

Factor The strength of the ambient occlusion; value 1.0 is like a white world shader.

Distance Distance from shading point to trace rays. A shorter distance emphasizes nearby features, while longer distances make it also take objects further away into account.

Lighting from ambient occlusion is only applied to diffuse reflection BSDFs; glossy or transmission BSDFs are not affected. Transparency of surfaces will be taken into account, i.e. a half-transparent surface will only half occlude.

An alternative method of using Ambient Occlusion on a per-shader basis is to use the *Ambient Occlusion* shader.

World Settings

Surface

Multiple Importance Sample Enabling this will sample the background texture such that lighter parts are favored, producing less noise in the render. It is almost always a good idea to enable this when using an image texture to light the scene, otherwise noise can take a very long time to converge.

Below is a comparison between *Multiple Importance Sample* off and on. Both images are rendered for 25 seconds (Off: 1500 samples, On: 1000 samples).

Fig. 2.2054: Multiple Importance Sample Off.

Fig. 2.2055: Multiple Importance Sample On.

Map Resolution Sets the resolution of the 'Multiple Importance Sample' map. Higher values may produce less noise when using high-res images, but will take up more memory and render slightly slower.

Max Bounces Maximim number of bounces the background light will contribute to the render.

See also:

See *Reducing Noise* for more information on how to reduce noise.

Volume

Sampling Method Options are *Multiple Importance*, *Distance*, or *Equiangular*. - If you have got a pretty dense volume that is lit from far away then distance sampling is usually more efficient. - If you have got a light inside or near the volume then equiangular sampling is better. - If you have a combination of both, then the multiple importance sampling will be better.

Interpolation Interpolation meathod to use for world volumes.

Linear Good smoothness and speed.

Cubic Smoothed high quality interpolation, but slower.

Homogeneous Volume Assume volume has the same density everywhere (not using any textures), for faster rendering. For example absorption in a glass object would typically not have any textures, and by knowing this we can avoid taking small steps to sample the volume shader.

Ray Visibility

As with other objects, *Ray Visibility* allows you to control which other shaders can "see" the environment.

Tricks

Sometimes it may be useful to have a different background that is directly visible versus one that is indirectly lighting the objects. A simple solution to this is to add a Mix node, with the Blend Factor set to Is Camera Ray. The first input color is then the indirect color, and the second the directly visible color. This is useful when using a high-res image for the background and a low-res image for the actual lighting.

Similarly, adding the *Is Camera* and *Is Glossy* rays will mean that the high-res image will also be visible in reflections.

Lamps

Next to lighting from the background and any object with an emission shader, lamps are another way to add light into the scene. The difference is that they are not directly visible in the rendered image, and can be more easily managed as objects of their own type.

Common Settings

Type Currently *Point*, *Spot*, *Area* and *Sun* lamps are supported. *Hemi* lamps are not supported, and will be rendered as sun lamps.

Size Size of the lamp in Blender Units; increasing this will result in softer shadows and shading.

Samples For the branch path tracing integrator, this specifies the number of direct light samples per AA sample. Point lamps might need only one sample, while area lamps typically need more.

Max Bounces Maximum number of times light from the lamp is allowed to *bounce*. Limited by *scene-wide bounce settings*

Cast Shadow By disabling this option, light from lamps will not be blocked by objects in-between. This can speed up rendering by not having to trace rays to the light source.

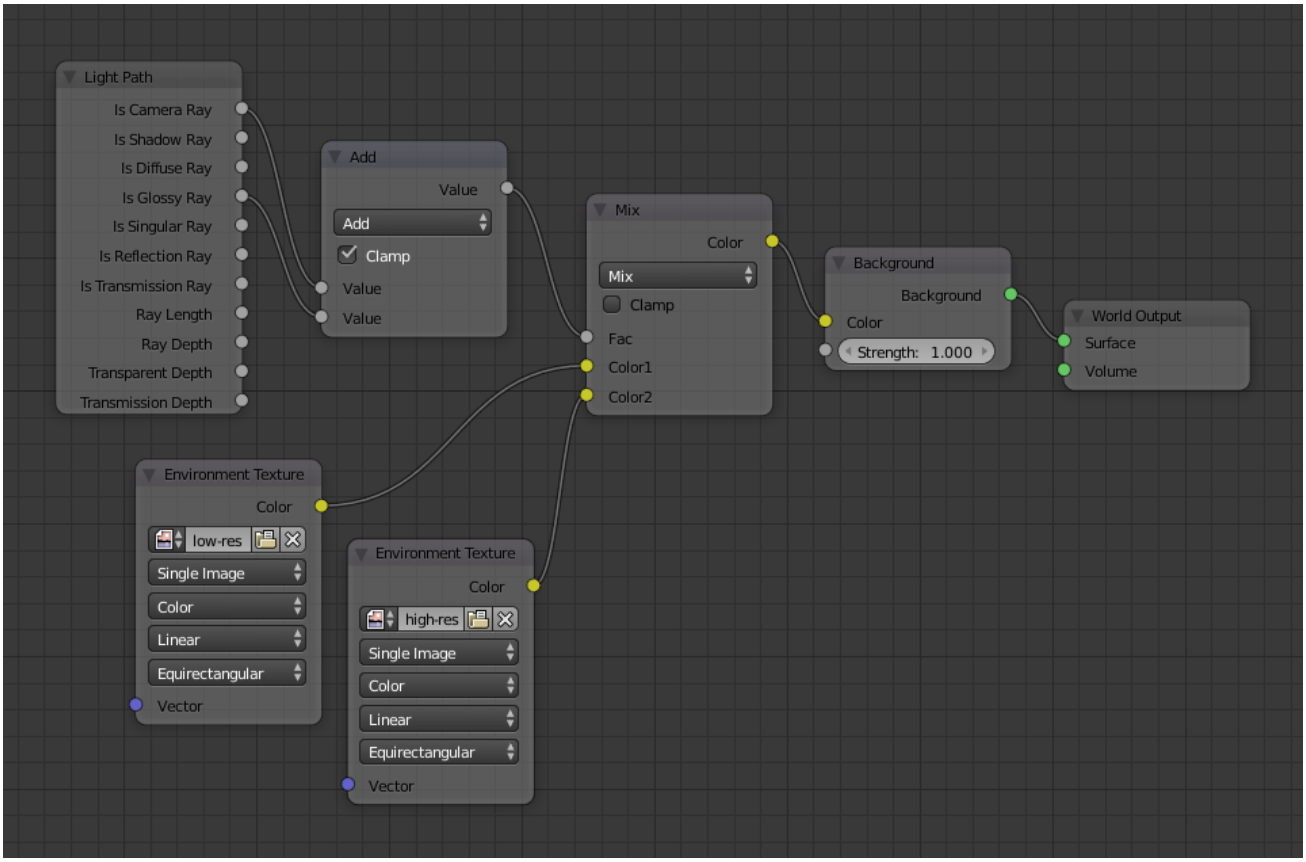


Fig. 2.2056: Nodes for the trick above.

Multiple Importance Sample By default lamps use only direct light sampling. For area lights and sharp glossy reflections, however, this can be noisy, and enabling this option will enable indirect light sampling to be used in addition to reduce noise.

Lamp Types

Point Lamp

Point lamps emit light equally in all directions. By setting the *Size* larger than zero, they become spherical lamps, which give softer shadows and shading. The strength of point lamps is specified in Watts.

Spot Lamp

Spot lamps emit light in a particular direction, inside a cone. By setting the *Size* larger than zero, they can cast softer shadows and shading. The size parameter defines the size of the cone, while the blend parameter can soften the edges of the cone.

Area Lamp

Area lamps emit light from a square or rectangular area with a Lambertian distribution.

Shape Shape of the lamp.

Rectangle The shape of the lamp can be represented as a rectangle and changed with the "X" and "Y" values.

Square The shape of the lamp can be represented as a square and changed with the *Size* property.

Light Portals

Area lamps can also function as light portals to help sample the environment light, and significantly reduce noise in interior scenes. Note that rendering with portals is usually slower, but as it converges more quickly, less samples are required.

Light portals work by enabling the *Portal* option, and placing areas lamps in windows, door openings, and any place where light will enter the interior.

In outdoor scenes most rays do not bounce much and just fly off into the sky and therefore, light portals are not helpful for outdoor scenes.

Fig. 2.2057: White Room model by Jay Hardy.

Sun Lamp

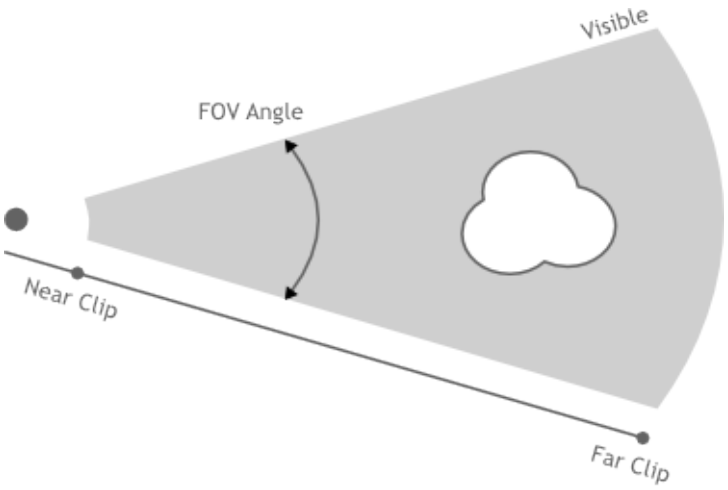
Sun lamps emit light in a given direction. Their position is not taken into account; they are always located outside of the scene, infinitely far away, and will not result in any distance falloff.

Because they are not located inside the scene, their strength uses different units, and should typically be set to lower values than other lights.

Camera

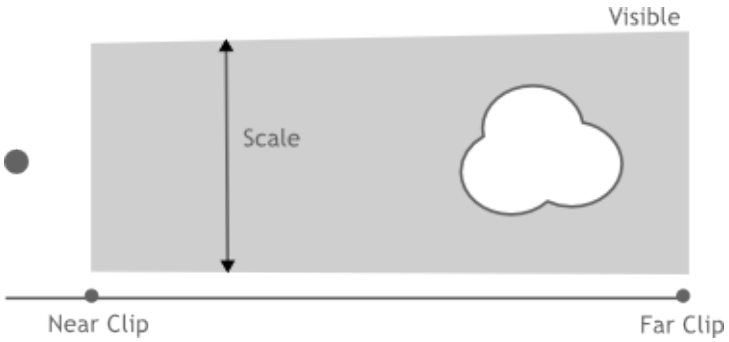
Perspective

Lens Size and Angle Control the field of view angle.



Orthographic

Scale Controls the size of objects projected on the image.



Panoramic

Cycles supports Equirectangular and Fisheye panoramic cameras. Note that these cannot be displayed with OpenGL rendering in the view-port; they will only work for rendering.

Equirectangular

Render a panoramic view of the scenes from the camera location and use an equirectangular projection, always rendering the full 360° over the X-axis and 180° over the Y-axis.

This projection is compatible with the environment texture as used for world shaders, so it can be used to render an environment map. To match the default mapping, set the camera object rotation to (90, 0, -90) or pointing along the positive X-axis. This corresponds to looking at the center of the image using the default environment texture mapping.

Fisheye

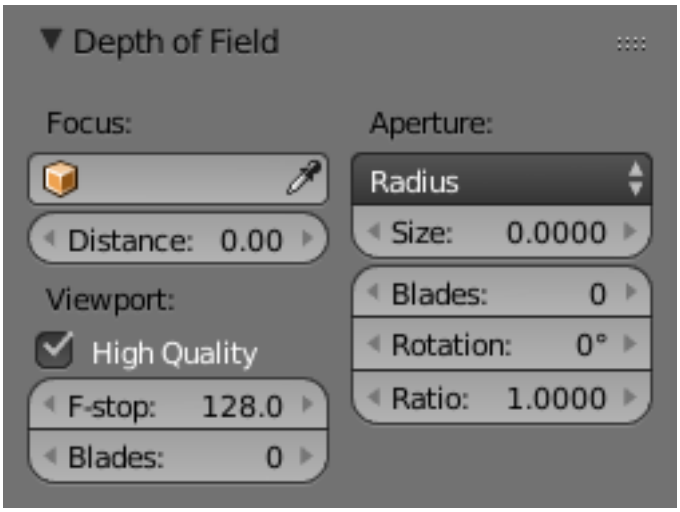
Fisheye lenses are typically wide angle lenses with strong distortion, useful for creating panoramic images for e.g. dome projection, or as an artistic effect. The *Fisheye Equisolid* lens will best match real cameras. It provides a lens focal length and field of view angle, and will also take the sensor dimensions into account.

The *Fisheye Equidistant* lens does not correspond to any real lens model; it will give a circular fish-eye that does not take any sensor information into account but rather uses the whole sensor. This is a good lens for full dome projection.

Lens Lens focal length in millimeter.

Field of View Field of view angle, going to 360 and more to capture the whole environment.

Depth of Field



Focus Set an object to be used as a focal point by the camera, causing the camera to focus on the selected object.

Distance When an object is not used, the camera can be set to focus on an area in 3D space set by the distance from the camera. Using the *Limit* Display option, you are able to view the distance in the 3D space.

High Quality Enables the High Quality *view-port* depth of field, giving a more accurate representation of *depth of field*. This allows the view-port depth of field to be closely represented to that of the render and render preview depth of field.

F-Stop Viewport depth of field aperture measured in F-Stops. Smaller numbers will cause more blur in the view-port, OpenGL renders, and sequencer.

Blades The number of polygonal sides to give blurred objects in the view-port. The minimum number of blades needed to enable the bokeh effect is 3 (triangle). (Only available with High Quality).

Aperture Use F-Stop or Radius to set the aperture for the render, and render preview. F-Stop is the focal ratio, where Radius is the the raidus of the focal point.

Size/Number Aperture radius *size*, or F-Stop *number* used for the render, and render preview. Using the F-Stop with a low number, or Radius with a large size will result in a strong blur, also allowing the use of the *bokeh effect*.

Blades Total number of polygonal blades used to alter the shape of the blurred objects in the render, and render preview. As with the view-port, the minimum amount of blades to enable the bokeh effect is 3, resulting in a triangle shaped blur.

Rotation Rotate the polygonal blades along the facing axis, and will rotate in a clockwise, and counter-clockwise fashion.

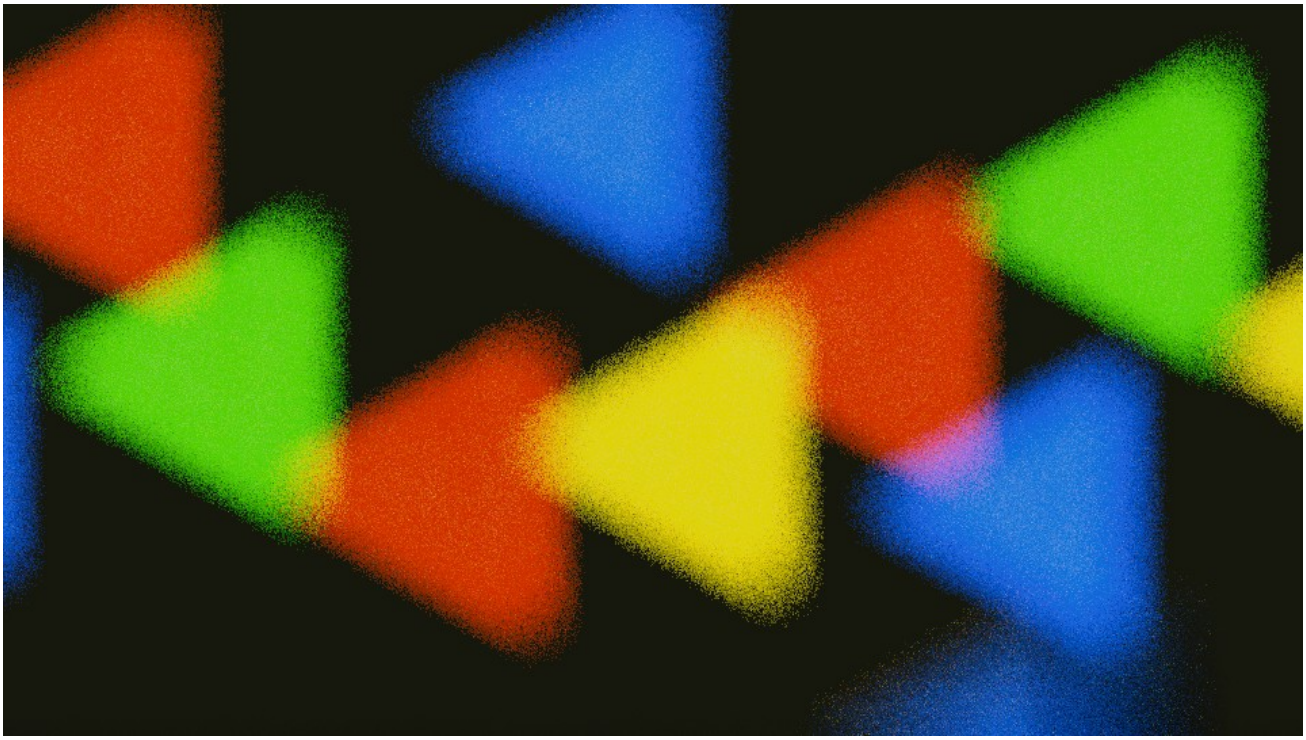
Ratio Change the amount of distortion to simulate the anamorphic bokeh effect. A setting of 1.0 shows no distortion, where a number below 1.0 will cause a horizontal distortion, and a higher number will cause a vertical distortion.

Clipping

Clip Start and End The interval in which objects are directly visible, Any objects outside this range still influence the image indirectly, as further light bounces are not clipped.

See also:

Camera Clipping.



Features

This page offers a comparison of available features on CPU, CUDA and OpenCL.

Feature	CPU	CUDA (NVIDIA GPU)	OpenCL (AMD GPU)
BASIC SHADING (Includes Node Shaders and Textures, Ambient Occlusion, Global Illumination...)			
Transparent Shadows			
Motion Blur			
Hair			
Volume			
Smoke / Fire			
Subsurface Scattering			
Open Shading Language			
CMJ sampling			
Branched Path integrator			
Displacement/Subdivision	(experimental)	(experimental)	(experimental)

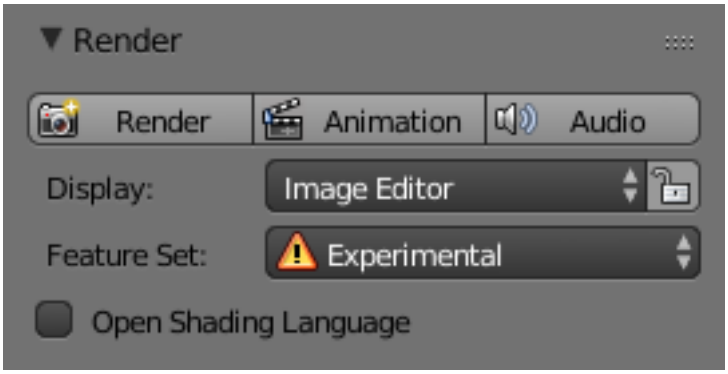
Experimental Features

Experimental features are disabled / hidden by default, but can be enabled by setting *Feature Set* to *Experimental* in the Render properties. Enabling the *Experimental Feature Set* will use experemental and incomplete features that might be broken or change in the future.

GPU Rendering

Introduction

GPU rendering makes it possible to use your graphics card for rendering, instead of the CPU. This can speed up rendering, because modern GPUs are designed to do quite a lot of number crunching. On the other hand, they also have some



limitations in rendering complex scenes, due to more limited memory, and issues with interactivity when using the same graphics card for display and rendering.

Cycles has two GPU rendering modes: *CUDA*, which is the preferred method for Nvidia graphics cards; and *OpenCL*, which supports rendering on AMD graphics cards.

Configuration

To enable GPU rendering, go into the User Preferences, and under the System tab, select the Compute Device(s) to use. Next, for each scene, you can configure to use CPU or GPU rendering in the Render properties.

CUDA

Nvidia CUDA (Compute Unified Device Architecture) is supported for GPU rendering with *Nvidia* graphics cards. We support graphics cards starting from GTX 4xx (computing capability 2.0).

Cycles requires recent Nvidia drivers to be installed, on all operating systems.

List of CUDA cards with shader model.

OpenCL

OpenCL (Open Computing Language) is supported for GPU rendering with *AMD* graphics cards. We only support graphics cards with GCN (Graphics Core Next) architecture (HD 7xxx and above). Not all HD 7xxx cards are GCN cards though, you can check if your card is here.

Cycles requires recent AMD drivers to be installed, on all operating systems.

Supported Features and Limitations

For an overview of supported features, check the comparison in the *Features*.

CUDA limitations: The maximum amount of individual textures is limited to 88 byte-image textures (`PNG`, `JPEG`, ..) and 5 float-image textures (`OpenEXR`, 16 bit `TIFF`, ..) on GTX 4xx/5xx cards. Newer cards do not have this limit.

Frequently Asked Questions

Why is Blender unresponsive during rendering?

While a graphics card is rendering, it cannot redraw the user interface, which makes Blender unresponsive. We attempt to avoid this problem by giving back control over the GPU as often as possible, but a completely smooth interaction cannot

be guaranteed, especially on heavy scenes. This is a limitation of graphics cards for which no true solution exists, though we might be able to improve this somewhat in the future.

If possible, it is best to install more than one GPU, using one for display and the other(s) for rendering.

Why does a scene that renders on the CPU not render on the GPU?

There maybe be multiple causes, but the most common is that there is not enough memory on your graphics card. We can currently only render scenes that fit in graphics card memory, and this is usually smaller than that of the CPU. Note that, for example, 8k, 4k, 2k and 1k image textures take up respectively 256MB, 64MB, 16MB and 4MB of memory.

We do intend to add a system to support scenes bigger than GPU memory, but this will not be added soon.

Can multiple GPUs be used for rendering?

Yes, go to *User Preferences* → *System* → *Compute Device Panel*, and configure it as you desire.

Would multiple GPUs increase available memory?

No, each GPU can only access its own memory.

What renders faster, Nvidia or AMD, CUDA or OpenCL?

Currently Nvidia with CUDA is rendering faster. There is no fundamental reason why this should be so, because we do not use any CUDA specific features, but the compiler appears to be more mature, and can better support big kernels. OpenCL support is still in an early stage and has not been optimized as much.

Error Messages

Unsupported GNU version! gcc 4.7 and up are not supported!

On Linux, depending on your GCC version you might get this error.

If so, delete the following line in `/usr/local/cuda/include/host_config.h`

```
#error -- unsupported GNU version! gcc 4.7 and up are not supported!
```

CUDA Error: Invalid kernel image

If you get this error on MS-Windows 64-bit, be sure to use the 64-bit build of Blender, not the 32-bit version.

CUDA Error: Kernel compilation failed

This error may happen if you have a new Nvidia graphics card that is not yet supported by the Blender version and CUDA toolkit you have installed. In this case Blender may try to dynamically build a kernel for your graphics card and fail.

In this case you can:

1. Check if the latest Blender version (official or experimental builds) supports your graphics card.
2. If you build Blender yourself, try to download and install a newer CUDA developer toolkit.

Normally users do not need to install the CUDA toolkit as Blender comes with precompiled kernels.

CUDA Error: Out of memory

This usually means there is not enough memory to store the scene on the GPU. We can currently only render scenes that fit in graphics card memory, and this is usually smaller than that of the CPU. See above for more details.

The Nvidia OpenGL driver lost connection with the display driver

If a GPU is used for both display and rendering, MS-Windows has a limit on the time the GPU can do render computations. If you have a particularly heavy scene, Cycles can take up too much GPU time. Reducing Tile Size in the Performance panel may alleviate the issue, but the only real solution is to use separate graphics cards for display and rendering.

Another solution can be to increase the timeout, although this will make the user interface less responsive when rendering heavy scenes. Learn More Here.

CUDA error: Unknown error in cuCtxSynchronize()

An unknown error can have many causes, but one possibility is that it is a timeout. See the above answer for solutions.

Render Baking

Refer to the Blender Render page for *general baking guidelines*

Cycles uses the render settings (samples, bounces, ...) for baking. This way the quality of the baked textures should match the result you get from the rendered scene.

The baking happens into the respective active textures of the object materials. The active texture is the last selected Image Texture node of the material node tree. That means the active object (or the selected objects, when not baking 'Selected to Active') needs a material, and that material needs at least an Image Texture node, with the image to be used for the baking. Note, the node does not need to be connected to any other node. The active texture is what projection painting and the viewport use as a criteria to which image to use. This way after the baking is done you can automatically preview the baked result in the Texture mode.

Options

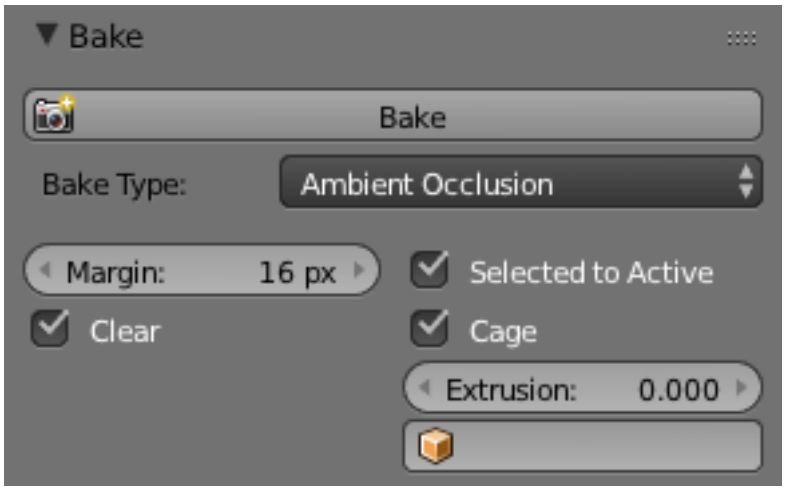


Fig. 2.2058: Ambient Occlusion Pass.

Bake Mode

Combined Bakes all materials, textures, and lighting except specularity.

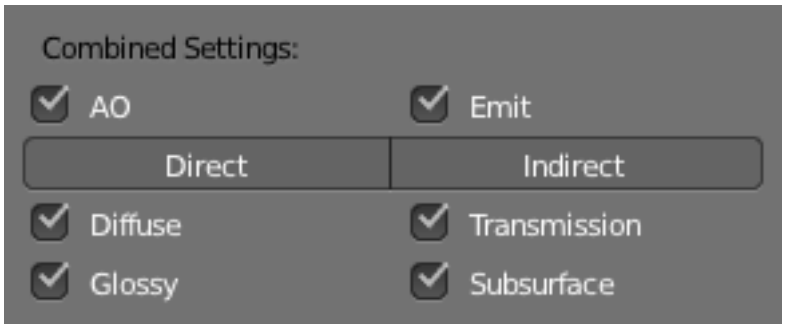


Fig. 2.2059: Combined Pass Options.

The passes that contribute to the combined pass can be toggled individually to form the final map.

Ambient Occlusion Bakes ambient occlusion as specified in the World panels. Ignores all lights in the scene.

Shadow Bakes shadows and lighting.

Normals Bakes normals to an RGB image.

Fig. 2.2060: Normal Pass Options.

Normal Space Normals can be baked in different spaces:

Object space Normals in object coordinates, independent of object transformation, but dependent on deformation.

Tangent space Normals in tangent space coordinates, independent of object transformation and deformation. This is the default, and the right choice in most cases, since then the normal map can be used for animated objects too.

Normal Swizzle Axis to bake into the red, green and blue channel.

For materials the same spaces can be chosen in the image texture options next to the existing *Normal Map* setting. For correct results, the setting here should match the setting used for baking.

UV Bakes colors of materials and textures only, without shading.

Emit Bakes Emission, or the Glow color of a material.

Environment Bakes the environment as seen from the center of the object.

Diffuse/Glossy/Transmission/Subsurface Bakes the diffuse, glossiness, transmission of subsurface pass of a material.

If only color is selected you get the pass color, which is a property of the surface and independent of sampling refinement.

If color is not selected, you get the direct and/or indirect contributions in grayscale.

If color and either direct or indirect is selected you get the direct and/or indirect contributions colored.

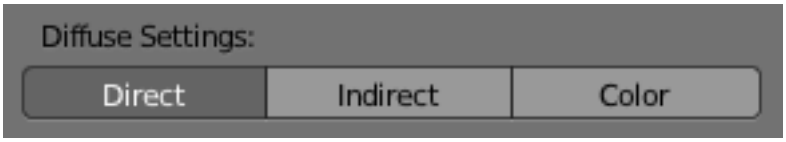


Fig. 2.2061: Diffuse Pass Options.

Additional Options

Margin Baked result is extended this many pixels beyond the border of each UV "island," to soften seams in the texture.

Clear If selected, clears the image before baking render.

Select to Active Bake shading on the surface of selected objects to the active object. The rays are cast from the lowpoly object inwards towards the highpoly object. If the highpoly object is not entirely involved by the lowpoly object, you can tweak the rays start point with *Ray Distance* or *Cage Extrusion* (depending on whether or not you are using cage). For even more control you can use a *Cage Object*.

Note: Memory Usage

There is a CPU fixed memory footprint for every object used to bake from. In order to avoid crashes due to lack of memory the highpoly objects can be joined before the baking process. The render tiles parameter also influence the memory usage, so the bigger the tile the less overhead you have, but the more memory it will take during baking (either in GPU or CPU).

Cage Cast rays to active object from a cage. A cage is a ballooned-out version of the lowpoly mesh created either automatically (by adjusting the ray distance) or manually (by specifying an object to use). When not using a cage the rays will conform to the mesh normals. This produces glitches on the edges, but it is a preferable method when baking into planes to avoid the need of adding extra loops around the edges.

Ray Distance Distance to use for the inward ray cast when using selected to active. Ray distance is only available when not using *Cage*.

Cage Extrusion Distance to use for the inward ray cast when using *Selected to Active* and *Cage*. The inward rays are casted from a version of the active object with disabled Edge Split modifiers. Hard splits (e.g., when the Edge Split modifier is applied) should be avoided because they will lead to non-smooth normals around the edges.

Cage Object to use as cage instead of calculating the cage from the active object with the *Cage Extrusion*.

Note: When the base mesh extruded does not give good results, you can create a copy of the base mesh and modify it to use as a *Cage*. Both meshes need to have the same *topology* (number of faces and face order).

Optimizing Renders

Render Settings

Integrator Panel

See the *integrator settings* for details.

Performance Panel

Threads

Auto-detect Automatically chooses the amount threads to match the number of logical processors on your computer.

Fixed Manually choose the amount threads to use for rendering. This can be useful for example, if you want to use your computer while rendering you can set the property to a thread count lower the the amount of logical processors on your computer.

Tiles

Tile Order Order of rendering tiles. This does not significantly affect performance.

Tile size X/Y The size of the tiles for rendering.

Depending on what device you are using for rendering, different tile sizes can give faster renders. For CPU rendering smaller tiles sizes (like 32 x 32) tend to be faster, while for *GPU rendering* larger tile sizes give better performance (like 256 x 256).

Progressive Refine Instead of rendering each tile until it has finished every sample, refine the whole image progressively. Note that progressive rendering is slightly slower than tiled rendering, but time can be saved by manually stopping the render when the noise level is low enough.

For rendering animations it is best to disable this feature, as stopping a frame early is not possible.

Save Buffers Save all render layers and passes to disk in the temp directory, and read them back after rendering has finished. This saves memory usage during rendering, particularly when using many render layers and passes.

Viewport

Viewport BVH Type

Dynamic BVH Objects can be transformed, added and deleted interactively, at the cost of slower renders.

Static BVH Object modifications require a complete *BVH* rebuild which reduces interactivity but renders faster.

Start Resolution Resolution to start rendering preview at, progressively increase it to the full viewport size.

Final Render

Persistent Images Keep image data in memory after rendering, for faster re-renders at the cost of extra memory usage when performing other tasks in Blender.

Acceleration Structure

Use Spatial Splits Spatial splits improve rendering performance in scenes with a mix of large and small polygons. The downsides are longer BVH build times and slightly increased memory usage.

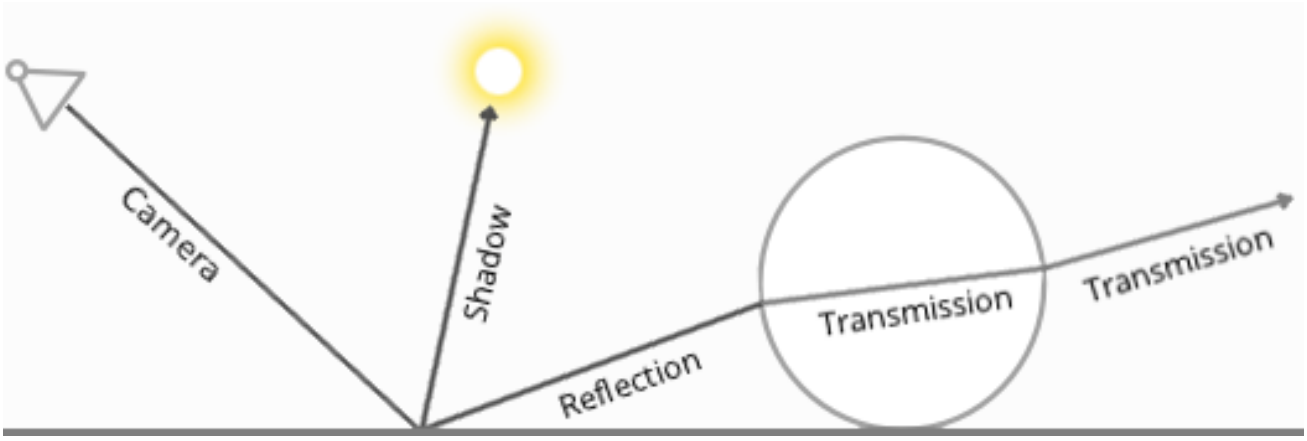
Use Hair BVH Use a special type of *BVH* for rendering hair. Disabling this option will reduce memory, at the cost of increasing hair render time.

Reducing Noise

When performing a final render, it is important to reduce noise as much as possible. Here we will discuss a number of tricks that, while breaking the laws of physics, are particularly important when rendering animations within a reasonable time. Click to enlarge the example images to see the noise differences well.

Path Tracing

Cycles uses path tracing with next event estimation, which is not good at rendering all types of light effects, like caustics, but has the advantage of being able to render more detailed and larger scenes compared to some other rendering algorithms. This is because we do not need to store, for example, a photon map in memory, and because we can keep rays relatively coherent to use an on-demand image cache, compared to e.g. bidirectional path tracing.



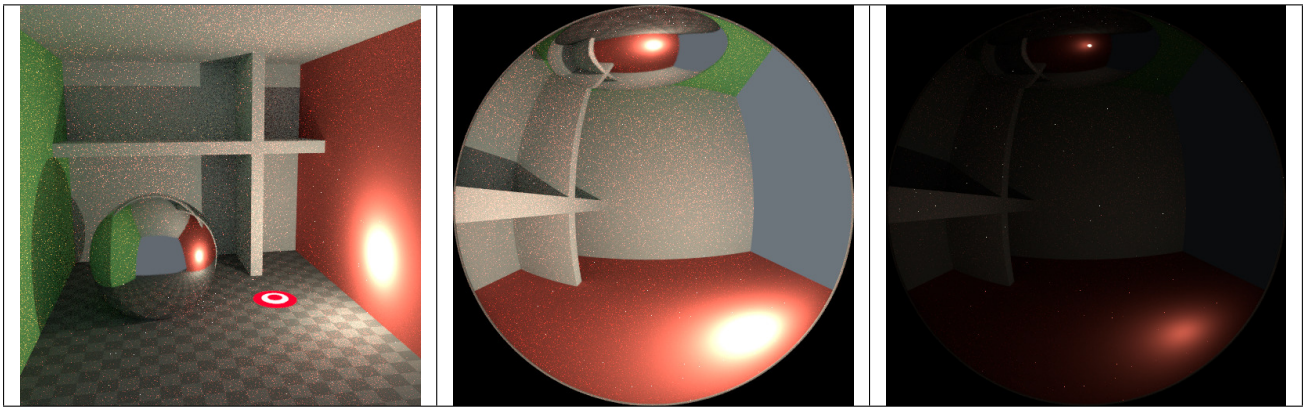
We do the inverse of what reality does, tracing light rays from the camera into the scene and onto lights, rather than from the light sources into the scene and then into the camera. This has the advantage that we do not waste light rays that will not end up in the camera, but also means that it is difficult to find some light paths that may contribute a lot. Light rays will be sent either according to the surface BRDF, or in the direction of known light sources (lamps, emitting meshes with Sample as Lamp).

See also:

For more details, see the *Light Paths* and *Integrator* documentation.

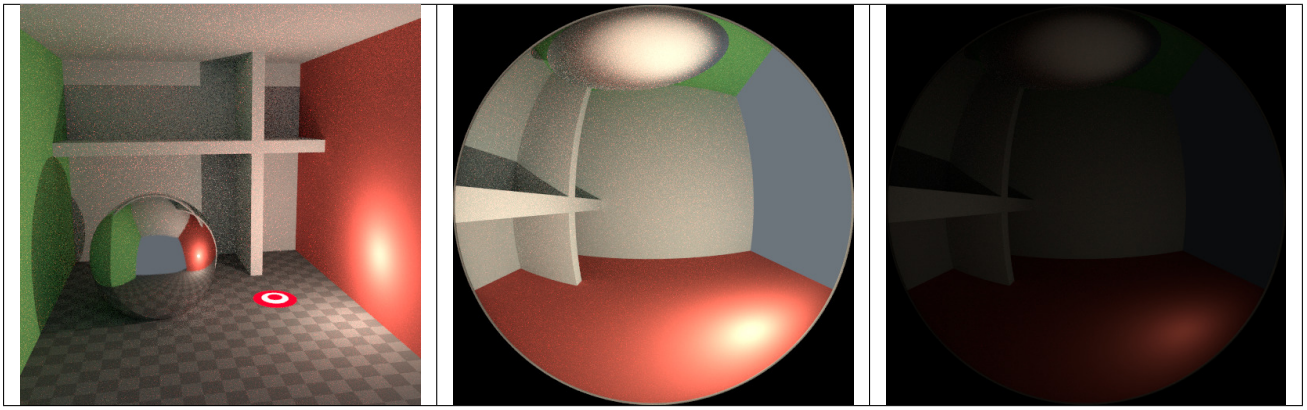
Where Noise Comes From

To understand where noise can come from, take for example this scene. When we trace a light ray into the specified location, this is what the diffuse shader "sees". To find the light that is reflected from this surface, we need to find the average color from all these pixels. Note the glossy highlight on the sphere, and the bright spot the lamp casts on the nearby wall. These hotspots are 100x brighter than other parts of the image and will contribute significantly to the lighting of this pixel.



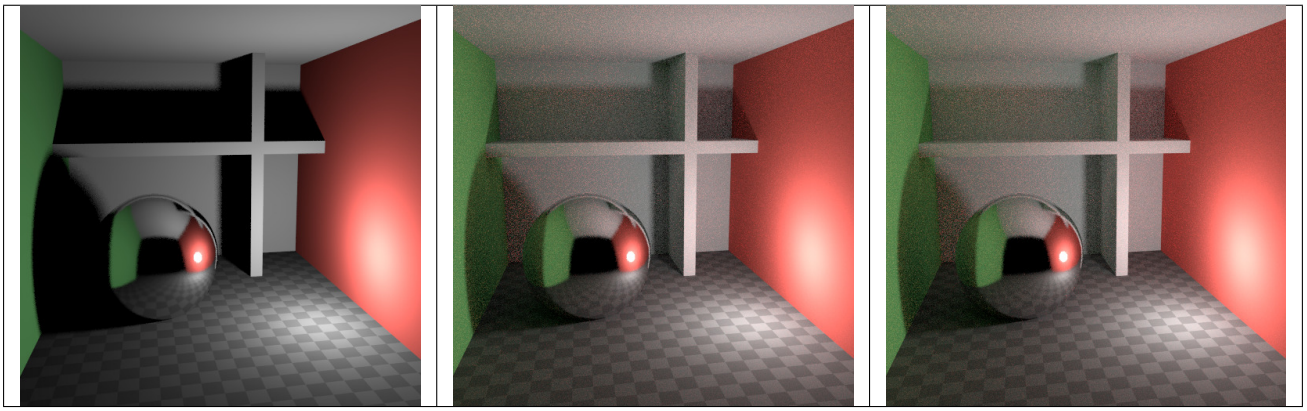
The lamp is a known light source, so it will not be too hard to find, but the glossy highlight(s) that it causes are a different matter. The best we can do with path tracing is to distribute light rays randomly over the hemisphere, hoping to find all the important bright spots. If for some pixels we miss some bright spot, but we do find it for another, that results in noise. The more samples we take, the higher the probability that we cover all the important sources of light.

With some tricks we can reduce this noise. If we blur the bright spots, they become bigger and less intense, making them easier to find and less noisy. This will not give the same exact result, but often it's close enough when viewed through a diffuse or soft glossy reflection. Below is an example of using Filter Glossy and Smooth Light Falloff.



Bounces

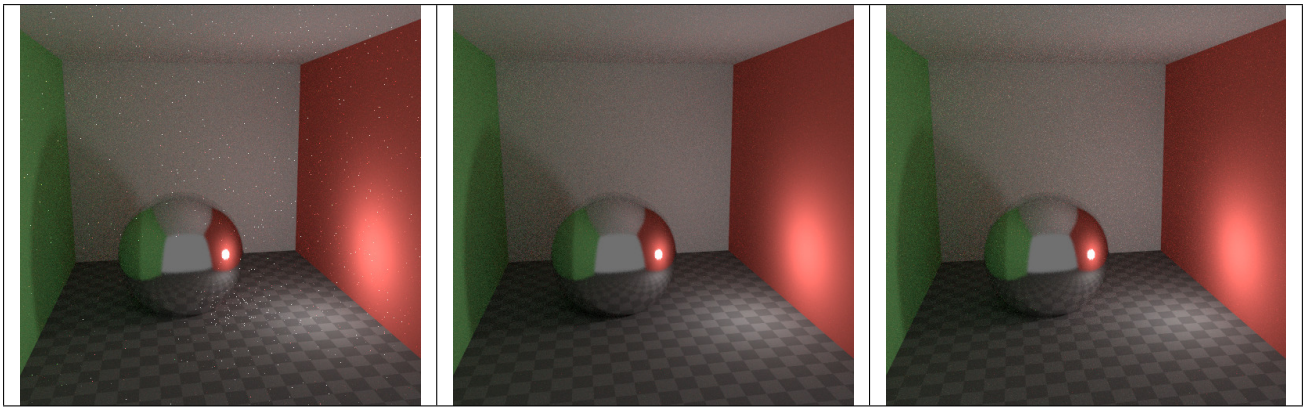
In reality light will bounce a huge number of times due to the speed of light being very high. In practice more bounces will introduce more noise, and it might be good to use something like the Limited Global Illumination preset that uses *fewer* bounces for different shader types. Diffuse surfaces typically can get away with fewer bounces, while glossy surfaces need a few more, and transmission shaders such as glass usually need the most.



Also important is to use shader colors that do **not** have components of value 1.0 or values near that; try to keep the maximum value to 0.8 or less and make your lights brighter. In reality, surfaces are rarely perfectly reflecting all light, but there are of course exceptions; usually glass will let most light through, which is why we need more bounces there. High values for the color components tend to introduce noise because light intensity then does not decrease much as it bounces off each surface.

Caustics and Filter Glossy

Caustics are a well-known source of noise, causing fireflies. They happen because the renderer has difficulty finding specular highlights viewed through a soft glossy or diffuse reflection. There is a *No Caustics* option to disable glossy behind a diffuse reflection entirely. Many render engines will typically disable caustics by default.

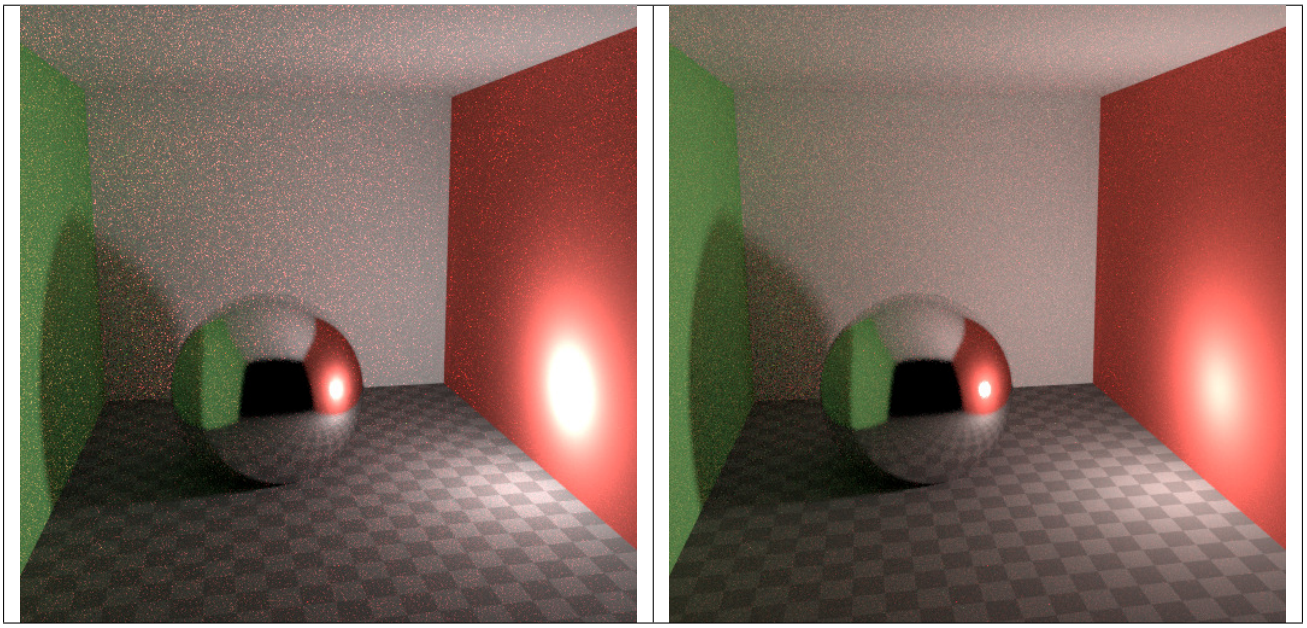


However, using No Caustics will result in missing light, and it still does not cover the case where a sharp glossy reflection is viewed through a soft glossy reflection. There is a *Filter Glossy* option to reduce the noise from such cases at the cost of accuracy. This will blur the sharp glossy reflection to make it easier to find, by increasing the shader Roughness.

The above images show default settings, no caustics, and filter glossy set to 1.0.

Light Falloff

In reality light in a vacuum will always fall off at a rate of 1/(distance^2). However, as distance goes to zero, this value goes to infinity and we can get very bright spots in the image. These are mostly a problem for indirect lighting, where the probability of hitting such a small but extremely bright spot is low and so happens only rarely. This is a typical recipe for fireflies.



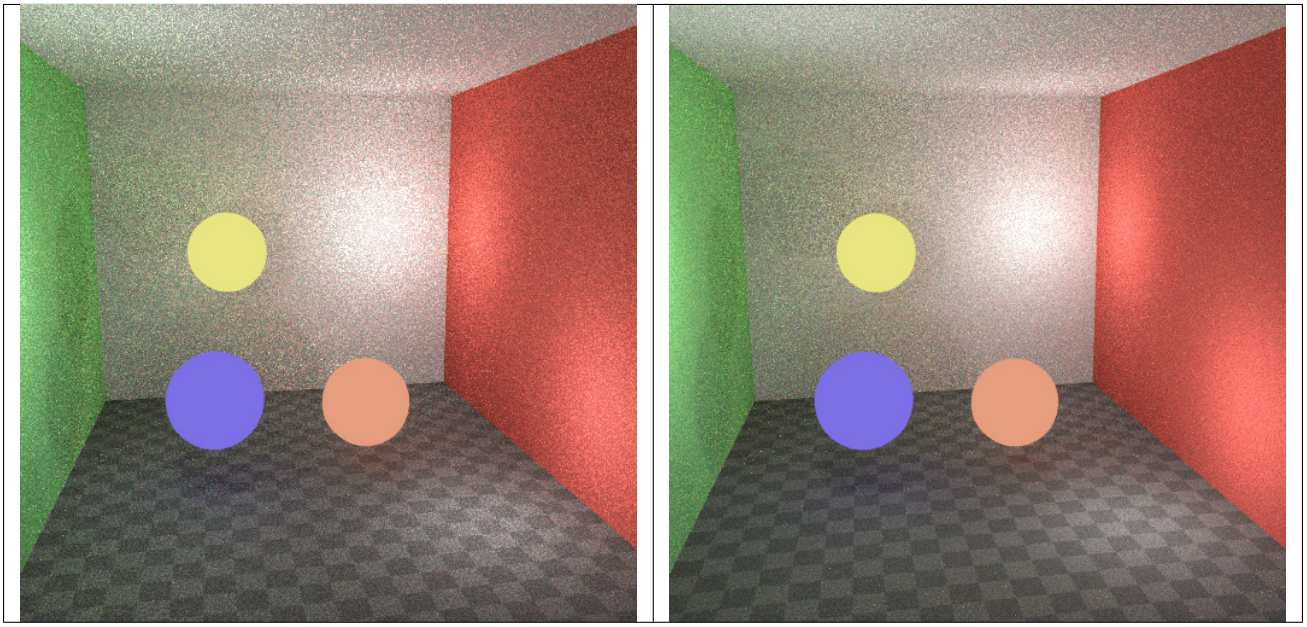
To reduce this problem, the *Light Falloff* node has a *Smooth factor*, that can be used to reduce the maximum intensity a light can contribute to nearby surfaces. The images above show default falloff and smooth value 1.0.

Sample as Lamp

Materials with emission shaders can be configured to be *sampled as lamp* (*Material Settings*). This means that they will get rays sent directly towards them, rather than ending up there based on rays randomly bouncing around. For very bright mesh light sources, this can reduce noise significantly. However, when the emission is not particularly bright, this will take samples away from other brighter light sources for which it is important to find them this way.

The optimal setting here is difficult to guess; it may be a matter of trial and error, but often it is clear that a somewhat glowing object may be only contributing light locally, while a mesh light used as a lamp would need this option enabled.

Here is an example where the emissive spheres contribute little to the lighting, and the image renders with slightly less noise by disabling Sample as Lamp on them.

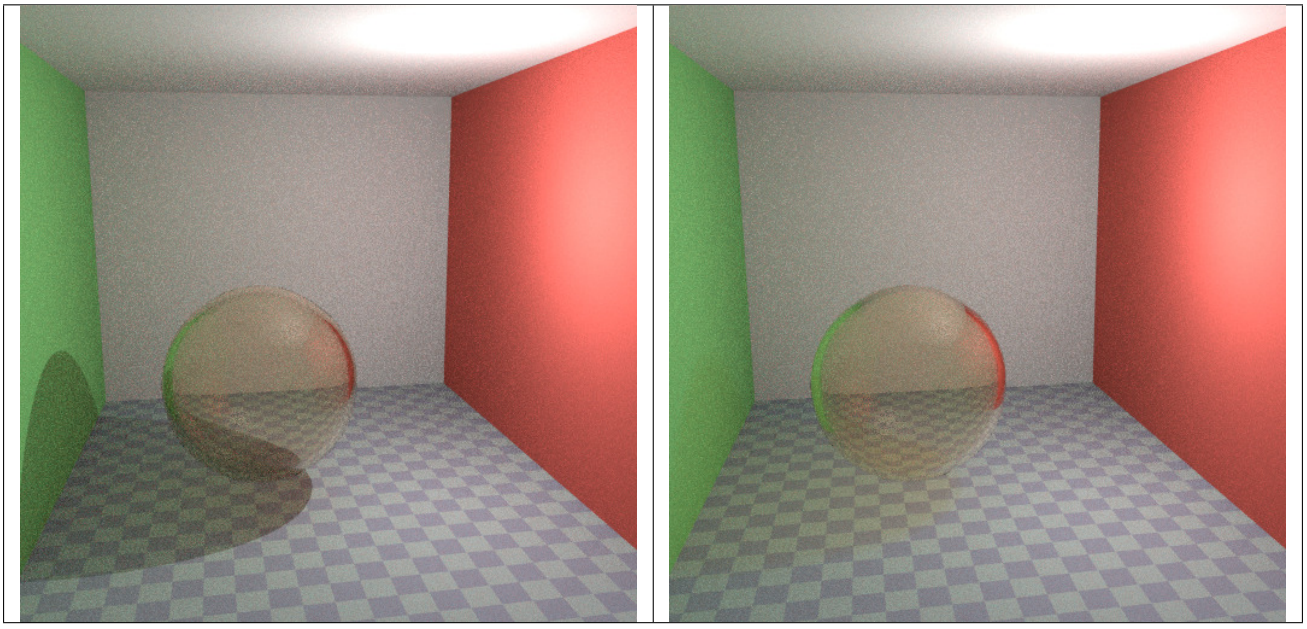


The world background also has a *Sample as Lamp* (*World Settings*) option. This is mostly useful for environment maps that have small bright spots in them, rather than being smooth. This option will then, in a preprocess, determine the bright spots, and send light rays directly towards them. Again, enabling this option may take samples away from more important light sources if it is not needed.

Glass and Transparent Shadows

With caustics disabled, glass will miss shadows, and with filter glossy they might be too soft. We can make a glass shader that will use a Glass BSDF when viewed *directly*, and a Transparent BSDF when viewed *indirectly*. The Transparent BSDF can be used for transparent shadows to find light sources straight through surfaces, and will give properly-colored shadows, but without the caustics. The Light Path node is used to determine when to use which of the two shaders.

Above we can see the node setup used for the glass transparency trick; on the left the render has too much shadow due to missing caustics, and on the right the render with the trick.



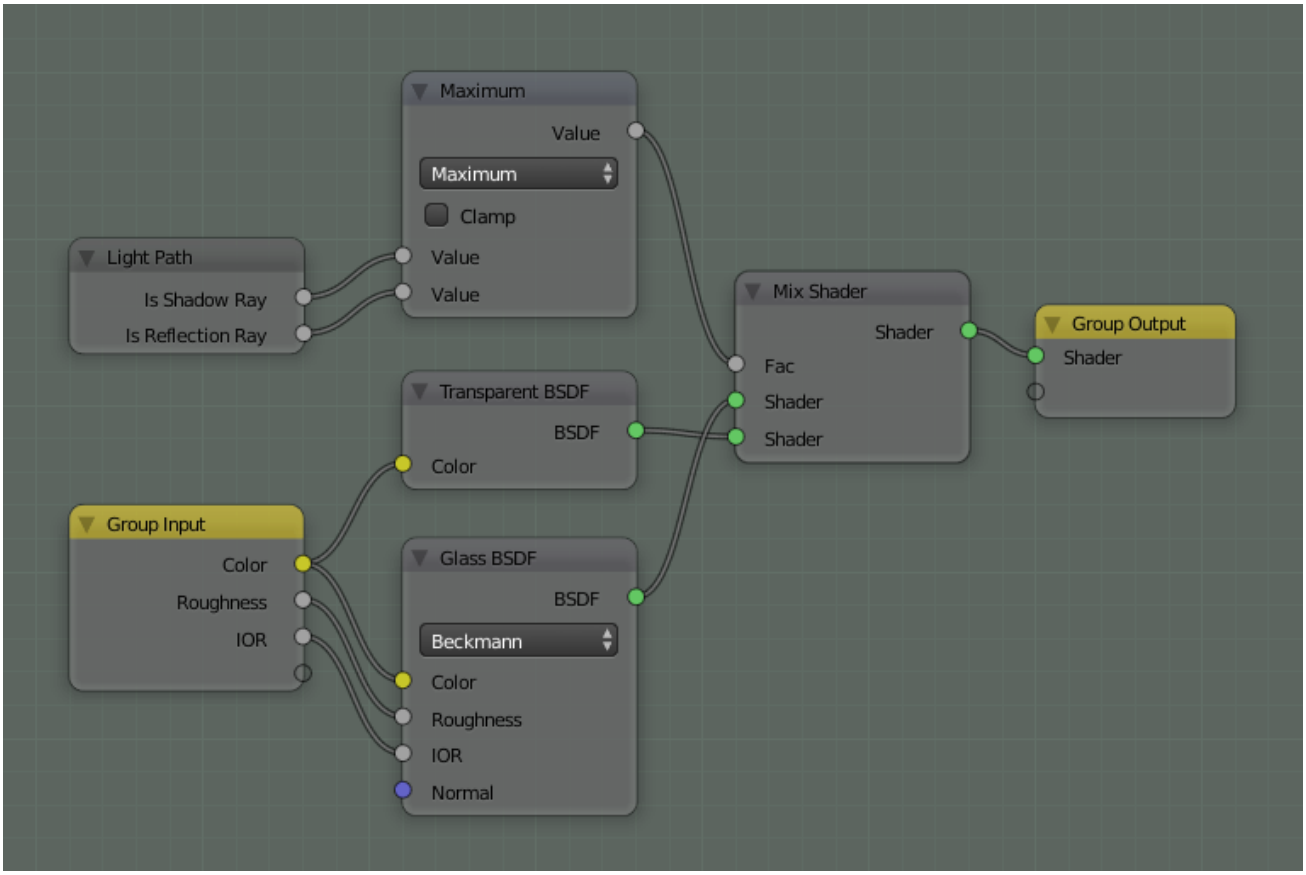


Fig. 2.2062: Optimized glass shader.

Light Portals

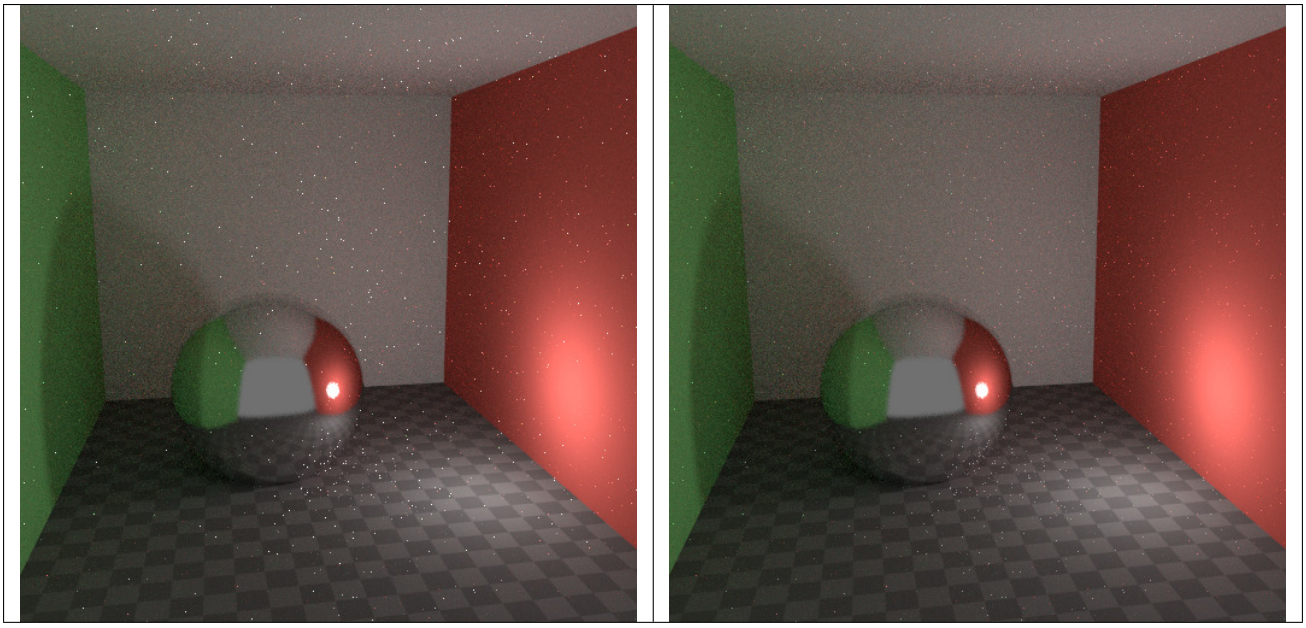
When rendering a daylight indoor scene where most of the light is coming in through a window or door opening, it is difficult for the integrator to find its way to them. To fix this, use *Light Portals*, these work by adding a *Area Lamp*. You then will need to modify its shape to match that of the opening that you are trying to fill.

Clamp Fireflies

Ideally with all the previous tricks, fireflies would be eliminated, but they could still happen. For that, the *intensity* that any individual light ray sample will contribute to a pixel can be *clamped* to a maximum value with the integrator *Clamp setting*.

If set too low this can cause missing highlights in the image, which might be useful to preserve for camera effects such as bloom or glare. To mitigate this conundrum it's often useful to clamp only indirect bounces, leaving highlights directly

visible to the camera untouched.



Shader Nodes

Cycles applies a number of shader node optimizations both at compile time and runtime. By exploiting them it is possible to design complicated "Uber Shader" style node groups that incur minimal render time overhead for unused features.

Node Optimizations

As the first step in preparing a node shader for execution, Cycles expands all node groups, as if using the Ungroup command, and discards UI only features like frames and reroute nodes.

After that, it applies some obvious transformations For example, it can (the list is not exhaustive):

- Replace the following nodes with the constant result of their evaluation, if all their inputs are determined to be constant:

  RGB, Value, Mix RGB, Math, Vector Math, RGB to BW, Gamma, Bright Contrast, Invert, Separate/Combine RGB/XYZ/HSV, Blackbody, RGB Curves, Vector Curves, Color Ramps.

- Detect Mix RGB, Math and Vector Math nodes that become no-op (without Clamp) or evaluate to 0 as a result of addition, subtraction, multiplication, division or dot/cross product with a known constant 0 or 1 input, and replace with the apropriate input link or constant result.

- Eliminate Mix RGB Mix (without Clamp) and Mix Shader nodes when Factor is known to be 0 or 1 by replacing with the appropriate input value or link.
- Eliminate no-op Mix RGB (except Burn, Dodge, Lighten, or enabled Clamp), Invert, RGB Curves and Vector Curves nodes with known zero Factor.
- Eliminate Emission and Background shader nodes that do not emit any light, and Add Shader nodes with one or both input arguments missing.
- Eliminate Bump with constant Height input, using its Normal input or Geometry Normal instead.
- Combine multiple copies of the same node with the same inputs into only one instance.

Finally, any nodes that end up not connected either directly or indirectly to the output node are removed.

Runtime Optimizations

When executing shaders, a special optimization is applied to Mix Shader nodes. If Factor evaluates to 0 or 1, any nodes that are only reachable via the unused branch of the mix are not evaluated.

This can substantially reduce the performance cost of combining multiple materials in one shader with vertex color, texture, or other input used as a switch.

Open Shading Language

If Open Shading Language is chosen as the rendering back-end, node shaders are translated to OSL code and then compiled and executed by the OSL runtime. In the process it applies its own extensive set of optimizations, both at compile time and runtime.

Open Shading Language can optimize out Script nodes if their outputs are unused or constant, even if their OSL shaders have side effects like debug tracing and message passing, which may be confusing. For that reason message passing with `setmessage` and `getmessage` should generally not be used for passing information forward in the graph; explicitly passing information through sockets should be preferred.

2.9.4 Render Output

Render Panel

Render **`F12`** Starts rendering a still image of the current frame.

Animation **`Ctrl-F12`** Starts rendering an animation. See *Rendering Animations* for more detail.

Audio Mixes all the the audio found in a scene and mixes into one file. See *Audio Rendering*.

By default the biggest area is replaced with the UV/Image Editor and the render appears.

To chancel the rendering process click the chancel button `X` besides the progressbar in the Info Editor, or press `Esc`.

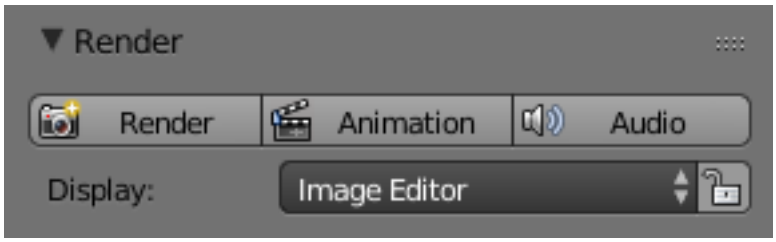


Fig. 2.2063: Render panel.

Display

Renders are displayed in the UV/Image Editor. You can set the way this is displayed to several different options in the Display menu:

Display

Keep UI The image is rendered to the UV/Image Editor, but the UI remains the same. You will need to open the UV/Image Editor manually to see the render result.

New Window A new floating window opens up, displaying the render.

Image Editor One of the existing editors is replaced with the UV/Image Editor, showing the render.

Full Screen The UV/Image Editor replaces the UI, showing the render.

Lock Interface Lock interface during rendering in favor of giving more memory to the renderer.

Output Options

The first step in the rendering process is to determine and set the output options. This includes render size, frame rate, pixel aspect ratio, output location, and file type.

Dimensions panel

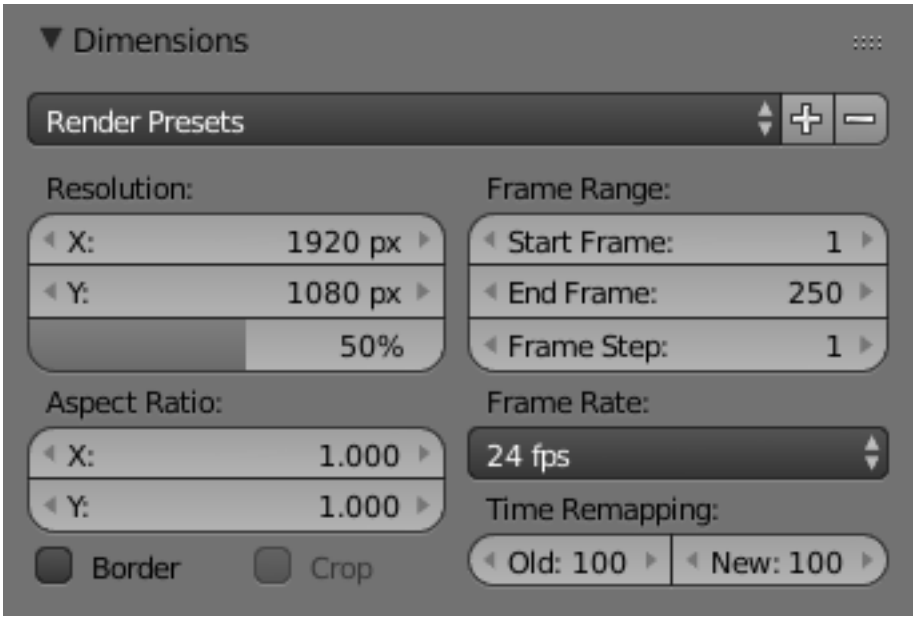


Fig. 2.2064: Dimensions Panel.

Render Presets Common format presets for TVs and screens.

Resolution

X/Y The number of pixels horizontally and vertically in the image.

Percentage Slider to reduce or increase the size of the rendered image relative to the X/Y values above. This is useful for small test renders that are the same proportions as the final image.

Aspect Ratio Older televisions may have non-square pixels, so this can be used to control the shape of the pixels along the respective axis. This will *pre-distorted* the images which will look stretched on a computer screen, but which will display correctly on a TV set. It is important that you use the correct pixel aspect ratio when rendering to prevent re-scaling, resulting in lowered image quality.

See *Video Output* for details on pixel aspect ratio.

Border You can render just a portion of the view instead of the entire frame. While in Camera View, press `Ctrl-B` and drag a rectangle to define the area you want to render. `Ctrl-Alt-B` is the shortcut to disable the border.

Note: This disables the *Save Buffers* option in *Performance* and *Full Sample* option in *Anti-Aliasing*.

Enabling *Crop* will crop the rendered image to the *Border* size, instead of rendering a black region around it.

Frame Range Set the *Start* and *End* frames for *Rendering Animations*. *Step* controls the number of frames to advance by for each frame in the timeline.

Frame Rate For an *Animation* the frame rate is how many frames will be displayed per second.

Time Remapping Use to remap the length of an animation.

Output Panel

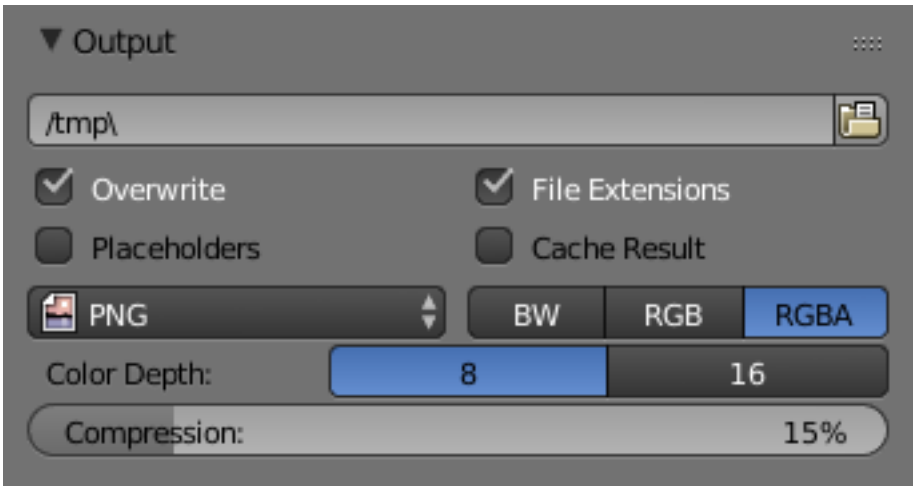


Fig. 2.2065: Output panel.

This panel provides options for setting the location of rendered frames for animations, and the quality of the saved images.

File Path Choose the location to save rendered frames.

When rendering an animation, the frame number is appended at the end of the file name with four padded zeros (e.g. `image0001.png`). You can set a custom padding size by adding the appropriate number of `#` anywhere in the file name (e.g. `image_##_test.png` translates to `image_01_test.png`).

This settings expands *relative paths* where a `//` prefix represents the directory of the current blend-file.

Overwrite Overwrite existing files when rendering

Placeholders Create empty placeholder frames while rendering

File Extensions Adds the correct file extensions per file type to the output files

Cache Result Saves the rendered image to your hard drive. This is helpful for heavy compositing.

Output Format Choose the file format to save to. Based on which format is used, other options such as channels, bit-depth and compression level are available.

Hint: Primitive Render-Farm

An easy way to get multiple machines to share the rendering workload is to:

- Set up a shared directory over a network file-system.

- Disable *Overwrite*, enable *Placeholders* in the Render *Output* panel.
- Start as many machines as you wish rendering to that directory

Video Output

Preparing your work for video

Once you have mastered the trick of animation you will surely start to produce wonderful animations, encoded with your favorite codecs, and possibly you will share them on the Internet with the rest of the community.

Sooner or later you will be struck with the desire to build an animation for television, or maybe burn your own DVDs. To spare you some disappointment, here are some tips specifically targeted at Video preparation. The first and principal one is to remember the double-dashed white lines in the camera view!

If you render for PC then the whole rendered image which lies within the *outer* dashed rectangle will be shown. For television, some lines and some part of the lines will be lost due to the mechanics of the electron beam scanning in your TV's cathode ray tube. You are guaranteed that what is within the *inner* dashed rectangle in camera view will be visible on the screen. Everything within the two rectangles may or may not be visible, depending on the given TV set that your audience watches the video on.

Color Saturation

Most video tapes and video signals are not based on the RGB model but on the YCrCb model: more precisely, the YUV in Europe (PAL), and the YIQ in the USA (NTSC), the latter being quite similar to the former. Hence some knowledge of this is necessary too.

The YCrCb model sends information as 'Luminance', or intensity (Y) and two 'Crominance' signals, red and blue (Cr and Cb). Actually a Black and White TV set shows only luminance, while color TV sets reconstruct color from Crominances (and from luminance). Construction of the YCrCb values from the RGB ones takes two steps (the constants *in italics* depend on the system: PAL or NTSC):

First, the Gamma correction (g varies: 2.2 for NTSC, 2.8 for PAL):

- $R' = R^{1/g}$
- $G' = G^{1/g}$
- $B' = B^{1/g}$

Then, the conversion itself:

- $Y = 0.299R' + 0.587G' + 0.114B'$
- $Cr = a_1 (R' - Y) + b_1 (B' - Y)$
- $Cb = a_2 (R' - Y) + b_2 (B' - Y)$

Whereas a standard 24 bit RGB picture has 8 bits for each channel, to keep bandwidth down, and considering that the human eye is more sensitive to luminance than to chrominance, the luminance signal is sent with more bits than the two chrominance signals. This bit expansion results in a smaller dynamic of colors in video, than what you are used to on monitors. You hence have to keep in mind that not all colors can be correctly displayed.

A rule of thumb is to keep the colors as 'grayish' or 'unsaturated' as possible; this roughly means keeping the dynamics of your colors within 80% of one another. In other words, the difference between the highest RGB value and the lowest RGB value should not exceed 0.8 (0 - 1 range) or 200 (0 - 255 range).

This is not strict, something more than 0.8 is acceptable, but an RGB display with color contrast that ranges from 0.0 to 1.0 will appear to be very ugly (over-saturated) on video, while appearing bright and dynamic on a computer monitor.

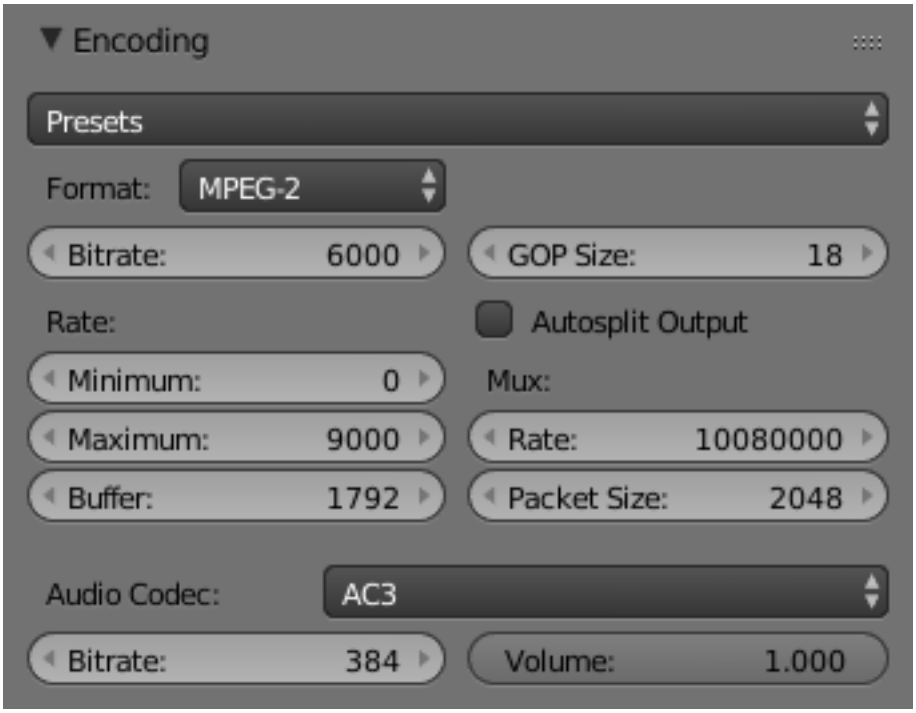


Fig. 2.2066: Encoding panel.

Encoding Panel

Here you choose which video codec you want to use, and compression settings. With all of these compression choices, there is a tradeoff between file size, compatibility across platforms, and playback quality.

When you view the *System Console*, you can see some of the output of the encoding process. You will see even more output if you execute Blender as `blender -d`.

Presets You can use the presets, which choose optimum settings for you for that type of output.

Format Video container or file type. For a list of all available options see *video formats*.

Codec Chooses the method of compression and encoding. For a list of all available options see *video formats*.

Lossless Output Allows the ability to perfectly reconstruct compressed data from compressed data.

Bitrate Set the average bitrate (quality), which is the count of binary digits per frame. See also: FFmpeg -b:v.

GOP Size The number of pictures per Group of Pictures. Set to 0 for "intra_only", which disables inter-frame video. From FFmpeg docs: "For streaming at very low bitrate application, use a low frame rate and a small GOP size. This is especially true for RealVideo where the Linux player does not seem to be very fast, so it can miss frames".

Autosplit Output If your video is HUGE and exceeds 2Gig, enable Autosplit Output. The main control over output filesize is the GOP or keyframe interlace. A higher number generally leads to a smaller file but needs a higher-powered device to replay it.

Mux Multiplexing settings.

Rate Maximum bit rate of the multiplexed stream.

Packet Size Reduces data fragmentation or muxer overhead depending on the source.

Note: Standards

Some codecs cannot encode off-the-wall video sizes, so stick to the XY sizes used in the presets for standard TV sizes.

Rate The bitrate control also includes a *Minimum* and a *Maximum*.

Buffer The decoder bitstream buffer size.

Audio Codec Audio conainer used, For a list of all available options see *video formats*.

Bitrate For each codec, you can control the bitrate (quality) of the sound in the movie. Higher bitrates are bigger files that stream worse but sound better. Use powers of 2 for compatibility.

Volume Sets the output volume of the audio.

Tips

Choosing which format to use depends on what you are going to do with the image.

If you are animating a movie and are not going to do any post-processing or special effects on it, use either `AVI-JPEG` or `AVI Codec` and choose the XviD open codec. If you want to output your movie with sound that you have loaded into the VSE, use M-PEG.

If you are going to do post-processing on your movie, it is best to use a frameset rendered as "OpenEXR" images; if you only want one file, then choose "AVI Raw". While AVI Raw is huge, it preserves the exact quality of output for post-processing. After post-processing (compositing and/or sequencing), you should compress the video.

Tip: You do not want to post-process a compressed file because the compression artifacts might throw off what you are trying to accomplish with the post-processing.

Note that you might not want to render directly to a video format. If a problem occurs while rendering, you have to re-render all frames from the beginning. If you first render out a set of static images (such as the default PNG, or the higher-quality OpenEXR), you can stitch them together with an Image Strip in the *Video Sequence Editor*. This way, you can easily:

- Restart the rendering from the place (the frame) where the problem occurred.
- Try out different video options in seconds, rather than minutes or hours.
- Enjoy the rest of the features of the VSE, such as adding Image Strips from previous renders, audio, video clips, etc.

Metadata

The *Metadata* panel includes options for writing meta-data into render output.

Note: Only some image formats support metadata: See *image formats*.

Stamp Output Add metadata has text to the render.

Stamp Text Color Set the color and alpha of the stamp text.

Stamp Background Set the color and alpha of the color behind the text.

Font Size Set the size of the text.

Draw Lables Draws the lables before the metadata text. For example, "Camera" infront of camera name ect...

Enabled Metadata

Stamping can include the following data.

Time Includes the current scene time and render frame as `HH:MM:SS.FF`

Date Includes the current date and time.

Render Time Includes the render time.

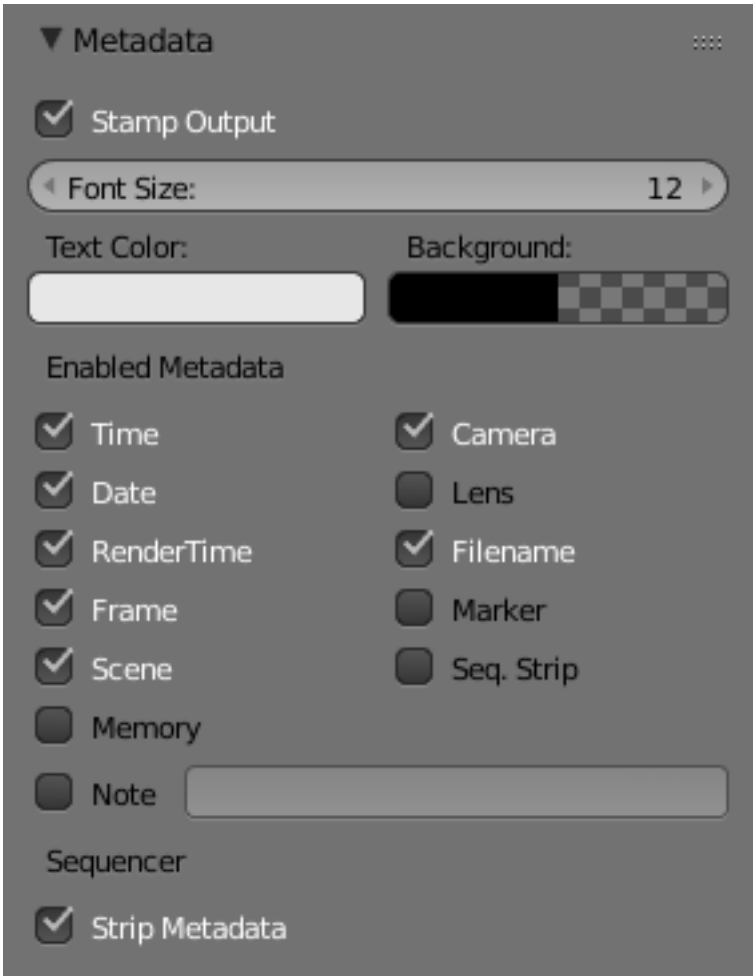


Fig. 2.2067: Metadata panel.

Frame Includes the frame number.

Scene Includes the name of the active scene.

Memory Includes the peak memory usage.

Note Includes a custom note.

> **Hint:** It can be useful to use the *Note* field if you are setting up a render-farm.
>
> Since you can script any information you like into it, such as an identifier for the render-node or the job-number.
>
> For details on stamping arbitrary values, see: this page.

Camera Includes the name of the active camera.

Lens Includes the name of the active camera's lens value.

Filename Includes the filename of the blend-file.

Marker Includes the name of the last marker.

Seq. Strip Includes the name of the foreground sequence strip.

Sequencer

Strip Metadata Use metadata from the strips in the sequencer.

Animation Player

The *Info Editor* → *Render* → *Play Rendered Animation* menu will play back the rendered animation in a new window.

You can also drop images or movie files in a running animation player. It will then restart the player with the new data.

Shortcuts

The following table shows the available hotkeys for the animation player.

Hotkey	Action
A	Toggle frame skipping.
P	Toggle ping-pong.
F	Flip drawing on the X axis.
Shift-F	Flip drawing on the Y axis.
Return	Start playback (when paused).
Numpad0	Toggle looping.
NumpadPeriod	Manual frame stepping.
Left	Step back one frame.
Right	Step forward one frame.
Down	Step back 10 frames.
Up	Step forward 10 frames.
Shift-Down	Use backward playback.
Shift-Up	Use forward playback.
Shift	Hold to show frame numbers.
LMB	Scrub in time.
Ctrl-Plus	Zoom in
Ctrl-Minus	Zoom out
Esc	Quit
Numpad1	60 fps

Continued on next page

Table 2.102 – continued from previous page

Hotkey	Action
Numpad2	50 fps
Numpad3	30 fps
Numpad4	25 fps
Shift-Numpad4	24 fps
Numpad5	20 fps
Numpad6	15 fps
Numpad7	12 fps
Numpad8	10 fps
Numpad9	6 fps
NumpadSlash	5 fps
Minus	Slow down playback.
Plus	Speed up playback.

A external player can also be used instead of the one included in Blender. To do this, select it in the *User Preferences*.

2.9.5 Post Processing

There are several effects you can enable in the Render Settings that add visual elements to rendered images, after the rendering has completed. These are not done in camera, but rather composited on top of the image.

Render Layers

Reference

Editor: Properties

Render layers allow you to render your scene in separate layers, usually with the intension of compositing them back together afterwards.

This can be useful for several purposes, such as color correcting certain elements differently, blurring the foreground as a fast manual method of creating DoF, or reducing the render quality for unimportant objects.

Using Render Layers can also save you from having to re-render your entire image each time you change something, allowing you to instead re-render only the layer(s) that you need.

Layer List

This is a list of all the Render Layers in the current scene.

Only layers which are enabled (checkbox on right is ticked) will be rendered. If the *Pin* icon at the bottom right of the list is enabled, only the active (highlighted) layer will be rendered.

Render Layers can be added and removed using the + and − buttons on the right, and existing layers can be renamed by double clicking on their name.

Layer Panel

The Layer Panel shows the settings of the active Render Layer from the list above.

You can select multiple layers using Shift-LMB.

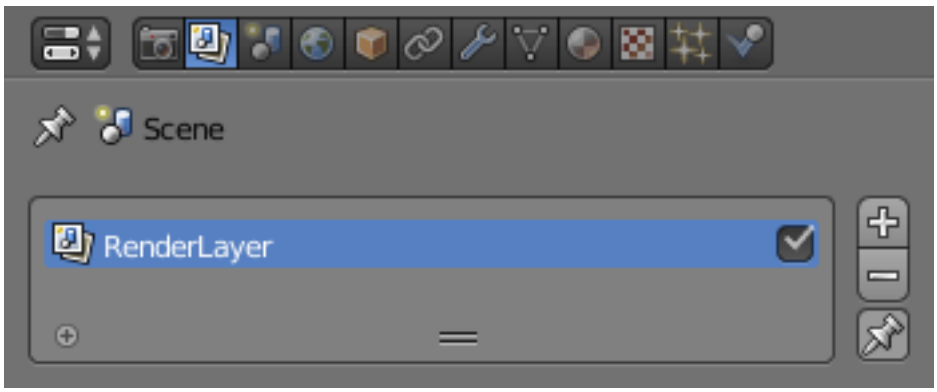


Fig. 2.2068: Layer list.

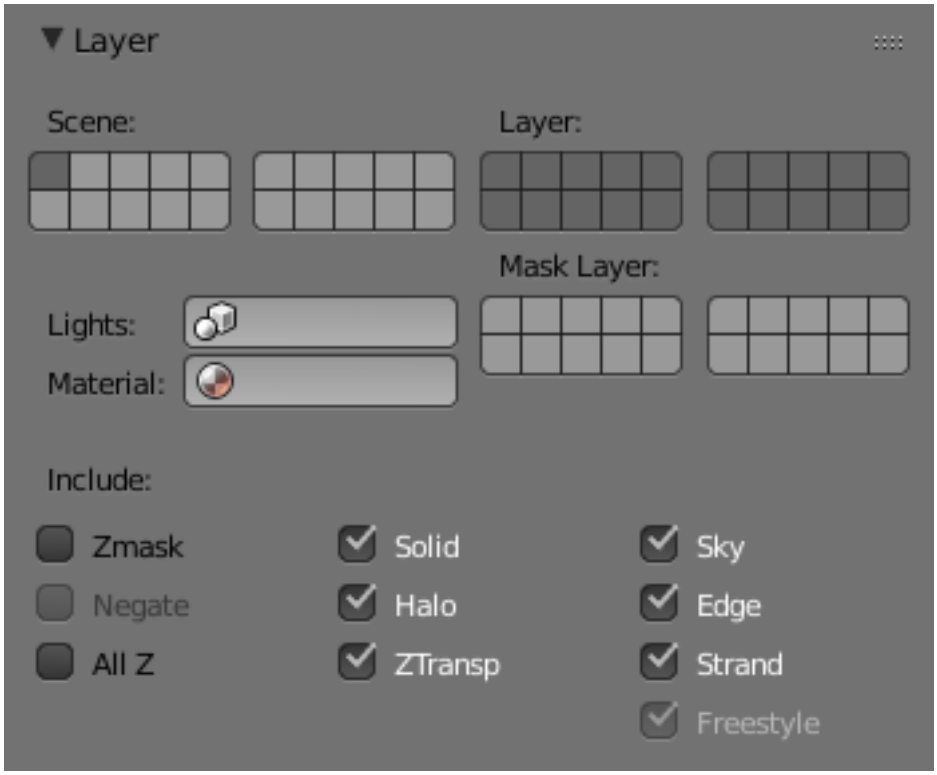


Fig. 2.2069: Layer panel.

Scene The Scene Layers, showing which are currently visible and will be rendered.

Layer The Scene Layers which are associated with the active Render Layer. Objects in those Scene Layers will be rendered in that Render Layer. When an object is in the Scene Layers but not the Render Layer, it will still cast shadows and be visible in reflections, so it is still indirectly visible.

Mask Layer Objects on these will mask out other objects appearing behind them.

Material Override Overrides all material settings to use the Material chosen here.

Examples of where this might be used:

- To check lighting by using a plain diffuse material on all objects
- Render a wireframe of the scene
- Create a custom render pass such as an anti-aliased matte or global coordinates.

See also:

Additional options shown in this panel are different for each render engine. See these options for:

- *Blender Render*
- *Cycles*

Using Render Layers

Each Render Layer has an associated set of *Scene Layers*. Objects which are on one of the associated Scene Layers are shown in that Render Layer, as long as that Scene Layer is also visible.

Warning: Only the objects in visible Scene Layers will be rendered. So, if only Scene Layer 1 is visible and your Render Layer set specifies to render only Layers 2 and 3, nothing will be rendered.

Post Processing Panel

The Post Processing panel is used to control different options used to process your image after rendering. It can be found in the *Render* tab of the *Properties Editor*.

Sequencer Renders the output of the sequence editor, instead of the view from the 3D scene's active camera. If the sequence contains scene strips, these will also be rendered as part of the pipeline. If *Compositing* is also enabled, the Scene strip will be the output of the Compositor.

Compositing Renders the output from the compositing node setup, and then pumps all images through the Composite node map, displaying the image fed to the Composite Output node.

Fig. 2.2070: Post Processing Panel.

Dithering

Dithering is a technique for blurring pixels to prevent banding that is seen in areas of gradients, where stair-stepping appears between colors. Banding artifacts are more noticeable when gradients are longer, or less steep. Dithering was developed for graphics with low bit depths, meaning they had a limited range of possible colors.

Dithering works by taking pixel values and comparing them with a threshold and neighboring pixels then does calculations to generate the appropriate color. Dithering creates the perceived effect of a larger color palette by creating a sort of visual color mixing. For example, if you take a grid and distribute red and yellow pixels evenly across it, the image would appear to be orange.

The *Dither* value ranges from 0 to 2.

Note: When using *Blender Internal* Render you get a few more options and these are discussed *here*.

Color Management

OpenColorIO is integrated into Blender, meaning many color spaces are supported with fine control over which color transformations are used.

Scene Linear Color Space

For correct results, different color spaces are needed for rendering, display and storage of images. Rendering and compositing is best done in scene *linear* color space, which corresponds more closely to nature, and makes computations more physically accurate.

Fig. 2.2071: Different views and exposures of the same render.

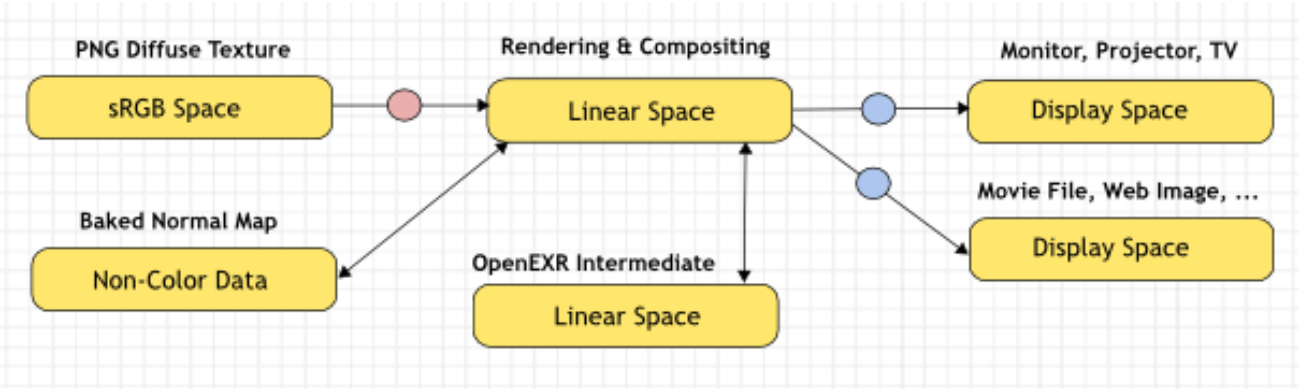


Fig. 2.2072: Example linear workflow.

If the colors are linear, it means that if in reality we double the amount of photons, the color values are also doubled. Put another way, if we have two photos/renders each with one of two lights on, and add those images together, the result would be the same as a render/photo with both lights on. It follows that such a radiometrically linear space is best for photo-realistic rendering and compositing.

However, these values do not di-

rectly cor- re- spond to hu- man per- cep- tion or the way dis- play de- vices work, and im- age files are of- ten stored in different color spaces, so we have to take care to do the right conversion into and out of this linear color space.

Settings

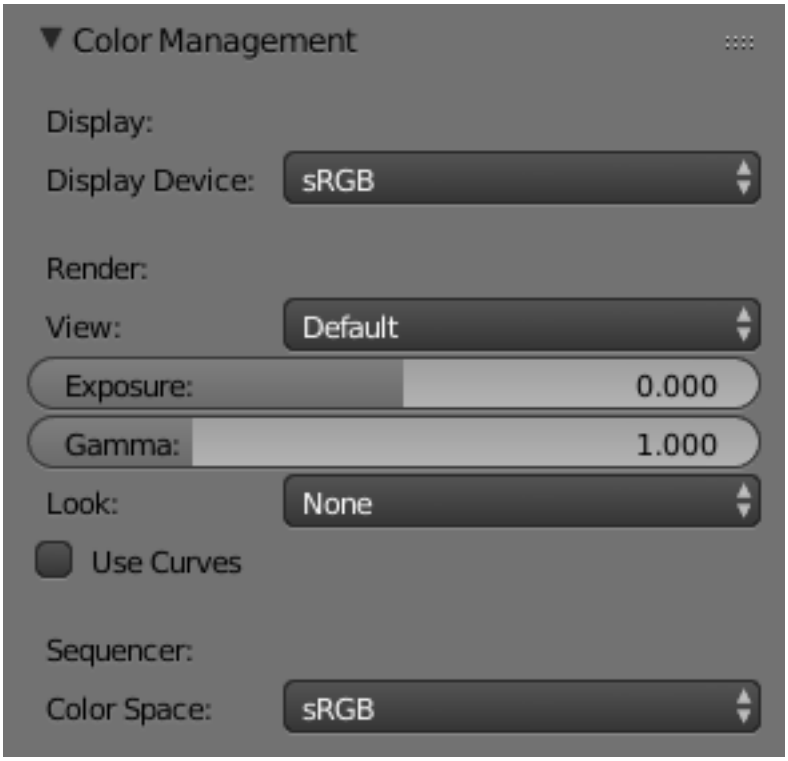


Fig. 2.2073: Scene settings for color management.

These

settings are found in the *Properties Editor* → *Scene* → *Color Management*

Display

Correct display of renders requires a conversion to the display device color space, which can be configured here. A computer monitor works differently from a digital cinema project or HDTV. The scene properties have these settings:

Display Device
The

device that the image is being viewed on.

Most computer monitors are configured for the sRGB color space, and so when working on a computer usually this option should just be left to the default. It would typically be changed when viewing the image on another display device connected to the computer, or when writing out image files intended to be displayed on another device.

Rec709 is commonly used for HDTVs, while XYZ and DCI-P3

are common for digital projectors.

Color management can be disabled by setting the device to None.

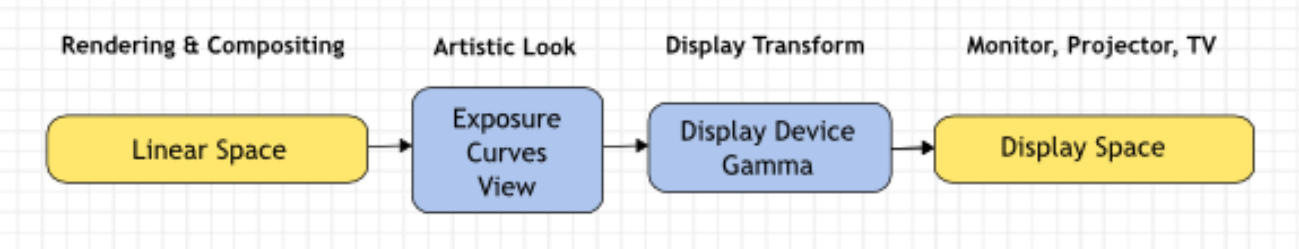


Fig. 2.2074: Conversion from linear to display device space.

Render

There is also an artistic choice to be made for renders. Partially that is

because display devices cannot display the full spectrum of colors and only have limited brightness, so we can squeeze the colors to fit in the gamut of the device. Besides that it can also be useful to give the renders a particular look, e.g. as if they have been printed on real film.

Another common use case is when you want to inspect renders, to see details in dark shadows or bright highlights, or identify render errors. Such settings would be only used temporarily and not get used for final renders.

View

These are different ways

to view the image on the same display device.

Default
Does no extra conversion besides the conversion for the display device.

RRT
Uses the ACES Reference Rendering Transform, to simulate a film-like look.

Film
This option is

another film-like look.

Raw and Log
Intended for inspecting the image but not for final export. Raw gives the image without any color space conversion, while Log gives a more "flat" view of the image without very dark or light areas.

Exposure
Used to control the image brightness (in stops) applied before color space con-

version.

Gamma
Extra gamma correction applied after color space conversion. Note that the default sRGB or Rec709 color space conversions already include a gamma correction of approximately 2.2 (except the Raw and Log views), so this would be applied in addition to that.

Look
Choose an artistic effect from set of measured film response data which roughly emulates the

look of certain film types. Applied before color space conversion.

Use Curves
Adjust RGB Curves to control image colors before color space conversion. Read more about using the *Curve Widget*.

Sequencer

Color Space
The color space that the sequencer operates in. By default the

sequencer operates in sRGB space, but it can also be set to work in Linear space like the Compositing nodes, or another color space. Different color spaces will give different results for color correction, cross fades, and other operations.

Image Files

When loading and saving media formats it is important to color management in mind. File formats such as PNG or JPEG will typically store colors in a color space ready for display, not in a linear space. When they are, for example, used as textures in renders, they need to be converted to linear first, and when saving renders for display on the web, they also need to be converted to a display space. Other file formats like OpenEXR store linear color spaces and as such are useful as

intermediate files in production.

When working with image files, the default color space is usually the right one. If this is not the case, the color space of the image file can be configured in the image settings. A common situation where manual changes are needed is when working with or baking normal maps or displacement maps, for example. Such maps do not actually store colors, just data encoded as colors. In such cases they should be marked as *Non-Color Data*.

Image datablocks will always store float buffers in memory in the scene linear color space, while a

byte buffer in memory and files on disk are stored in the color space specified with this setting:

Color Space
The color space of the image on disk. This depends on the file format, for example PNG or JPEG images are often stored in sRGB, while OpenEXR images are stored in a linear color space. Some images such as normal, bump or stencil maps do not strictly contain 'colors', and on such values no color space conversion should ever be applied. For such images the color space should be set to None.

By default only renders are displayed and

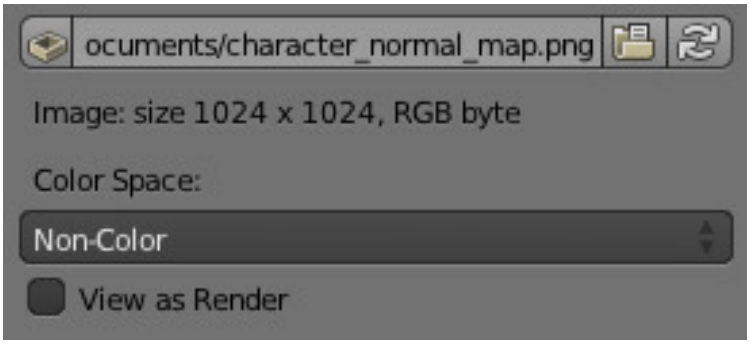


Fig. 2.2075: Image settings for color management.

saved with the render view transformations applied. These are the Render Result and Viewer image data-blocks, and the files saved directly to disk with the Render Animation operator. However, when loading a render saved to an intermediate OpenEXR file, Blender cannot detect automatically that this is a render (it could be e.g. an image texture or displacement map). We need to specify that this is a render and that we want the transformations applied, with these two settings:

View as Render
Display the image data-block (not only renders) with view transform, exposure

sure, gamma, RGB curves applied. Useful for viewing rendered frames in linear OpenEXR files the same as when rendering them directly.

Save as Render
Option in the image save operator to apply the view transform, exposure, gamma, RGB curves. This is useful for saving linear OpenEXR to e.g. PNG or JPEG files in display space.

OpenColorIO Configuration

Blender comes with a standard OpenColorIO configuration that contains a number of useful display devices and view transforms. The reference linear color space used is the linear color space with Rec. 709 chromaticities and D65 white point.

However, OpenColorIO was also designed to give a consistent user expe-

rience across multiple applications, and for this a single shared configuration file can be used. Blender will use the standard OCIO environment variable to read an OpenColorIO configuration other than the default Blender one. More information about how to set up such a workflow can be found on the OpenColorIO website.

We currently use the following color space roles:

scene_linear
color space used for rendering, compositing, and storing all float precision images in memory.

default_sequencer
default color space

for sequencer, *scene_linear* if not specified

default_byte
default color space for byte precision images and files, *texture_paint* if not specified.

default_float
default color space for float precision images and files, *scene_linear* if not specified.

The standard Blender configuration

also includes some support for ACES (code and documentation), even though we have a different linear color space. It is possible to load and save EXR files with the Linear ACES color space, and the RRT view transform can be used to view images with their standard display transform. However, the ACES gamut is larger than the Rec. 709 gamut, so for best results an ACES specific configuration file should be used. OpenColorIO provides an ACES configuration, though it may need a few more tweaks to be usable in production.

2.9.6 Freestyle

Introduction

What is FreeStyle?

Freestyle is an edge-

and line-based non-photorealistic (NPR) rendering engine. It relies on mesh

data and z-depth information to draw lines on selected edge types. Various line styles can be added to produce artistic (“hand drawn”, “painted”, etc.) or technical (hard line) looks.

The two operating modes: *Python Scripting* and *Parameter Editor* – allow a powerful diversity of line styles and results. Line styles such as Japanese big brush, cartoon, blueprint, thickness-with-depth are already pre-scripted in Python. The Parameter Editor mode allows intuitive editing of features such as dotted lines and easy setup of multiple line types and edge definitions. On top of all of that, with the introduction of line style modifiers, the sky is the limit!

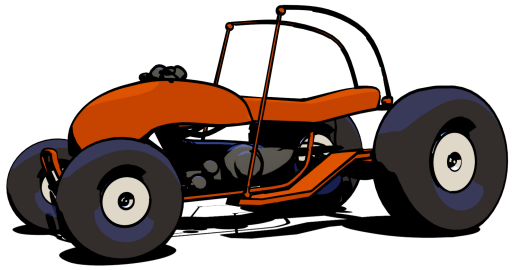


Fig. 2.2076: ATV buggy by Rylan Wright (RONIN). CC BY. (File:AtvBuggy.zip)

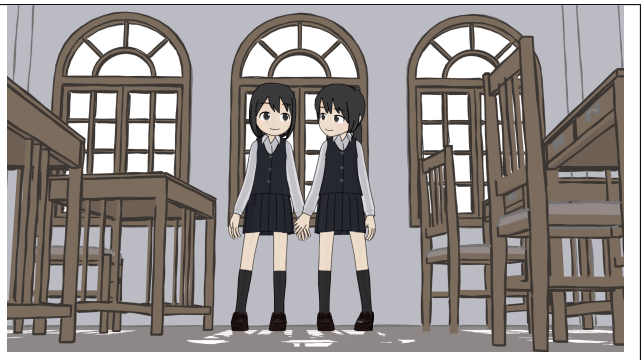


Fig. 2.2077: By mato.sus304. CC BY-SA. (File:Mato_sus304_cut02.zip)

Fig. 2.2078: A cartoon scene from OHA Studio © Mechanimotion Entertainment. (the blend-file).

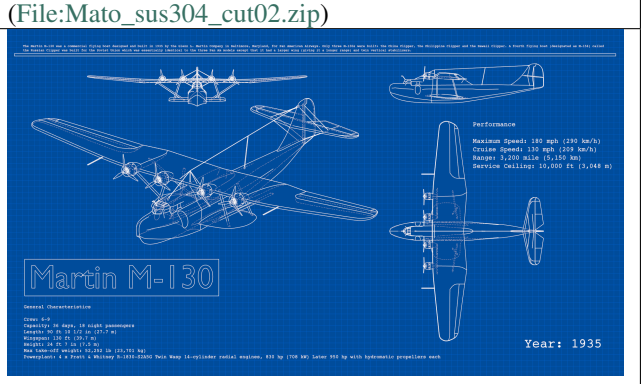


Fig. 2.2079: Blueprint render of Martin M-130 from 1935 by LightBWK. CC0. Warning: heavy file! designed for stress test Blender to the limits and may crash Blender. (File:M-130Blueprint.zip)

More artwork can be found at Release Note Artwork Showcase.

The Big Picture

- Activate FreeStyle by *Properties*

Editor → *Render tab* → *FreeStyle* panels checkbox. Please note that FreeStyle is only available for the Blender Internal renderer.

- Freestyle settings are located in the new *Render Layers* tab.
- One render layer can only have one viewmap. A viewmap holds the edge de-

tection settings (Crease Angle, Culling toggle, Face Smoothness toggle, Material Boundaries toggle, Sphere Radius and Kr Derivative Epsilon advanced options).

- A viewmap can have multiple line sets.
- A line set controls which line types and selections will be rendered, from lines based on your scene.
- Each line set uses one line

style (which can be shared between multiple line sets).

- A line style tells Freestyle how to render the linked line sets in terms of color, alpha, thickness and other aspects.

Known Limitations

- Highly memory demanding: All mesh objects in

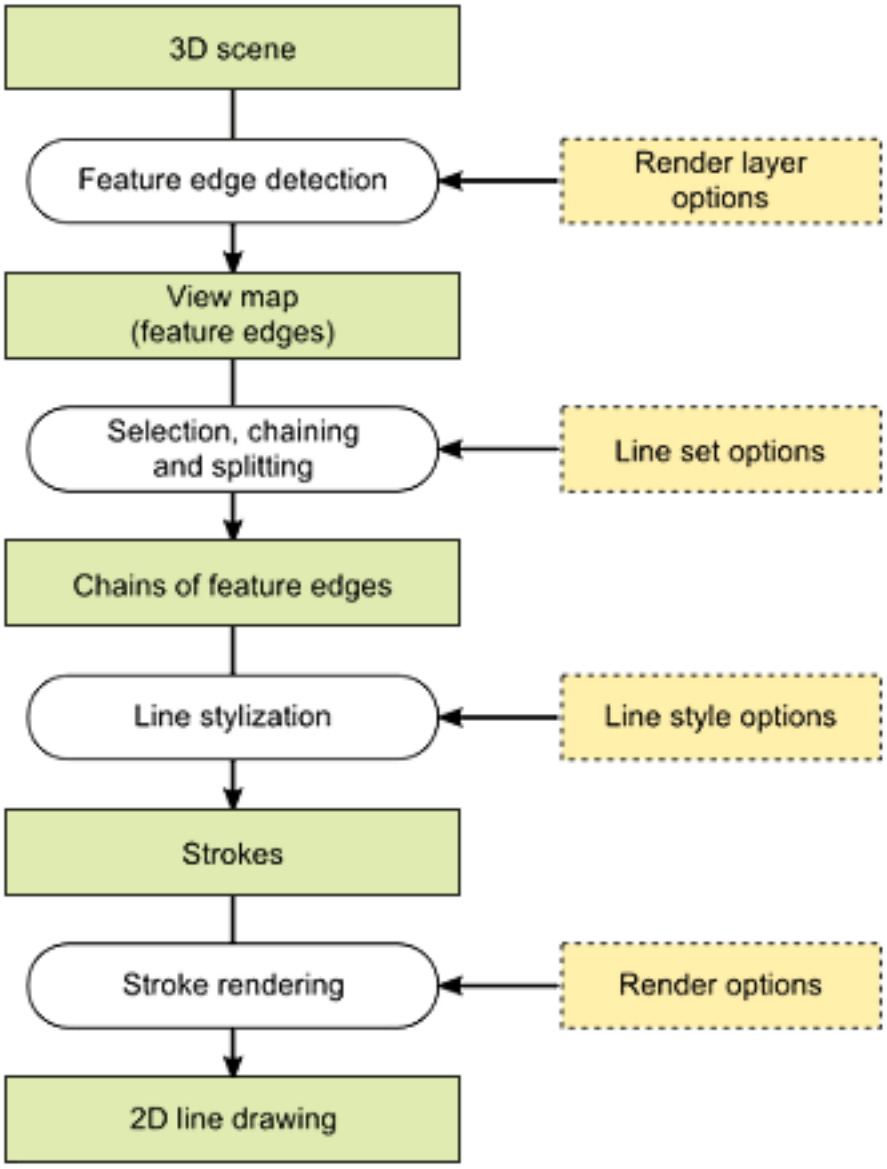


Fig. 2.2080: Block diagram of Freestyle view map and processes.

a render layer are loaded at once.

- Only faced mesh objects are supported. The following kinds of meshes are ignored:
  - Mesh faces with wire materials.
  - Mesh faces with completely transparent materials.
  - Mesh faces with the *Cast Only*

material option enabled.

- Transparent faces are treated as opaque faces.
- No edges at face intersections are detected yet.
- Layer masks do not work with Freestyle.
- Freestyle rendering results do not have any Z depth information.
- Panoramic cam-

eras are not supported.

Core Options

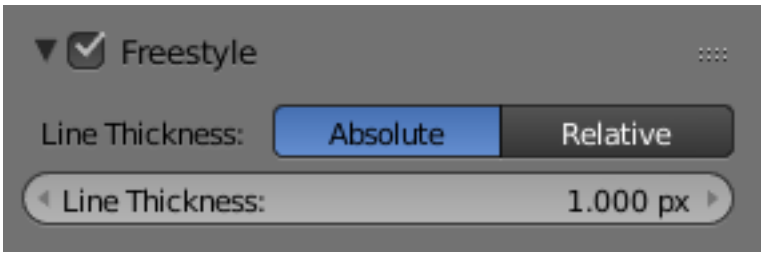


Fig. 2.2081: Freestyle core options.

Activating Freestyle in the *Render* tab of the Properties Editor will give you the following options:

Line Thickness
There are two different modes for defining the

base line thickness:

Absolute
The line thickness is given by a user-specified number of pixels. The default value is 1.0.

Relative
The unit line thickness is scaled by the proportion of the present vertical image resolution to 480 pixels. For in-

stance,
the *unit line thickness* is 1.0 with the image height set to 480 px, 1.5 with 720 px and 2.0 with 960 px.

Line Thickness
Only for *Absolute* line thickness: base line thickness in pixels is 1.0 by default.

Viewmaps

There is only one viewmap per render layer. It controls the edge detection parameters. Which detected edges are actually rendered,

and
how, can be controlled either through the user-friendly *parameter editor*, or powerful but complex *Python scripting*.

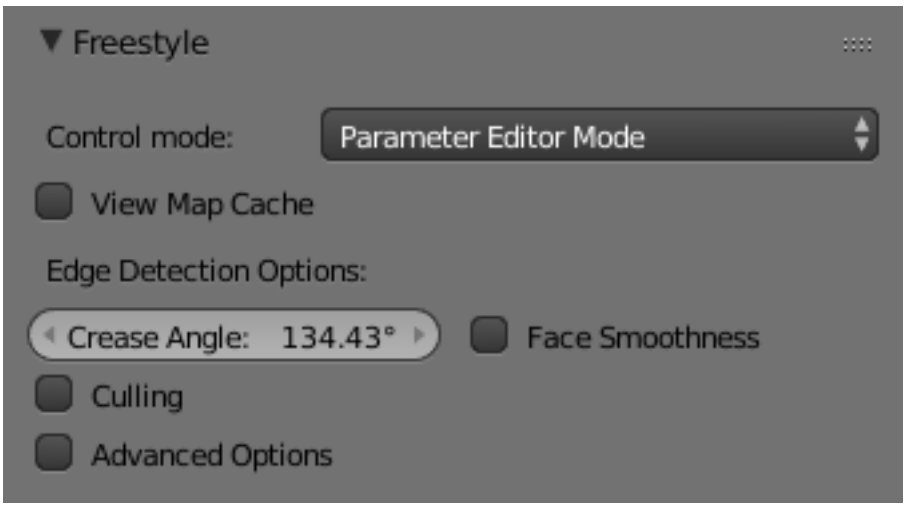


Fig. 2.2082: Freestyle panel.

View Map Cache
TODO.
See
wiki.

Face Smoothness
When
en-
abled,
Smooth
Shad-
ing
will
be
taken
into
ac-
count
for
edges
cal-
cu-
la-
tion.

Crease Angle
If
two
ad-
ja-
cent
faces
form
an
an-
gle
less

than the defined *Crease Angle*, the edge between them will be rendered when using *Crease* edge type selection in a line set. The value also affects *Silhouette* edge type selection.

Culling
Ignore the edges that are out of view (saves some processing time and memory, but may reduce the quality of the result in some cases).

Advanced Options

Fig. 2.2083: Advanced Options enabled.

Sphere Radius
It affects the calculation of curvatures for *Ridge*, *Valley* and *Suggestive Contour* edge type selection in a line set.

Kr Derivative Epsilon
It provides you with control over the output

of *Suggestive Contour* and *Silhouette* edge type selection (further information in this pdf).

Parameter Editor

Introduction to Parameter Editor

The Freestyle *Parameter Editor Mode* is a user-friendly interface to define

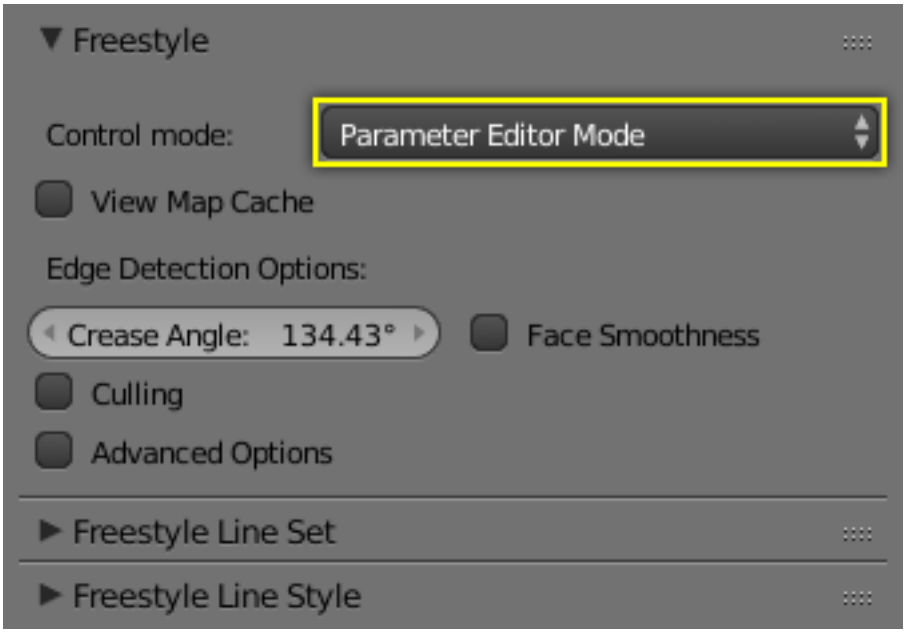


Fig. 2.2084: Parameter Editor Mode.

and control line sets and line styles.

Line Sets
Control which of the edges detected by Freestyle will actually be used (rendered).

Line Styles
Control how the selected edges are

rendered.

A view map (hence a render layer) can have multiple line sets, and each line set is linked to one line style.

Line Set

A line set selects, among the lines (edges) detected by Freestyle, which ones will be rendered using its attached *line style*,

through various methods.

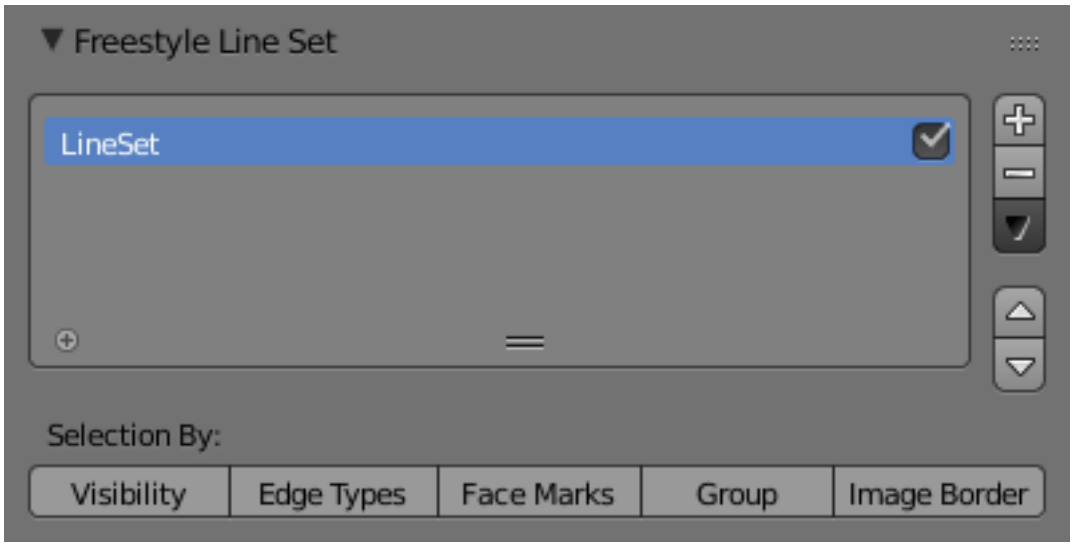


Fig. 2.2085: Freestyle Line Set panel

Selection By

Visibility

There are three choices for selecting edges by visibility.

Visible
Only lines occluded by no surfaces

are rendered.

Hidden
Lines occluded by at least one surface are rendered.

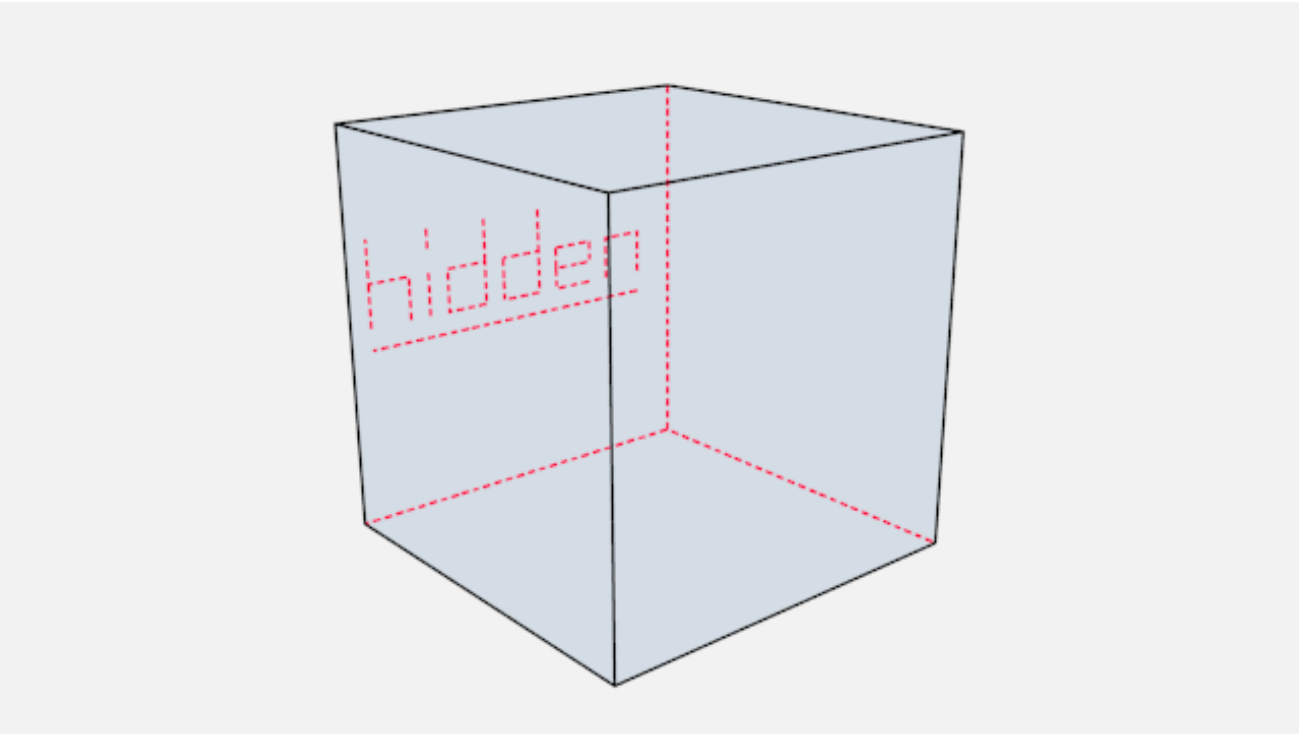


Fig. 2.2086: Proof of concept of visible and hidden edges by LightBWK (Sample blend-file)

QI Range
QI stands for *Quantitative Invisibility*. Lines oc-

cluded by a number of surfaces in the given range are rendered.

Start and End
Min/max number of occluding surfaces for a line to be rendered.

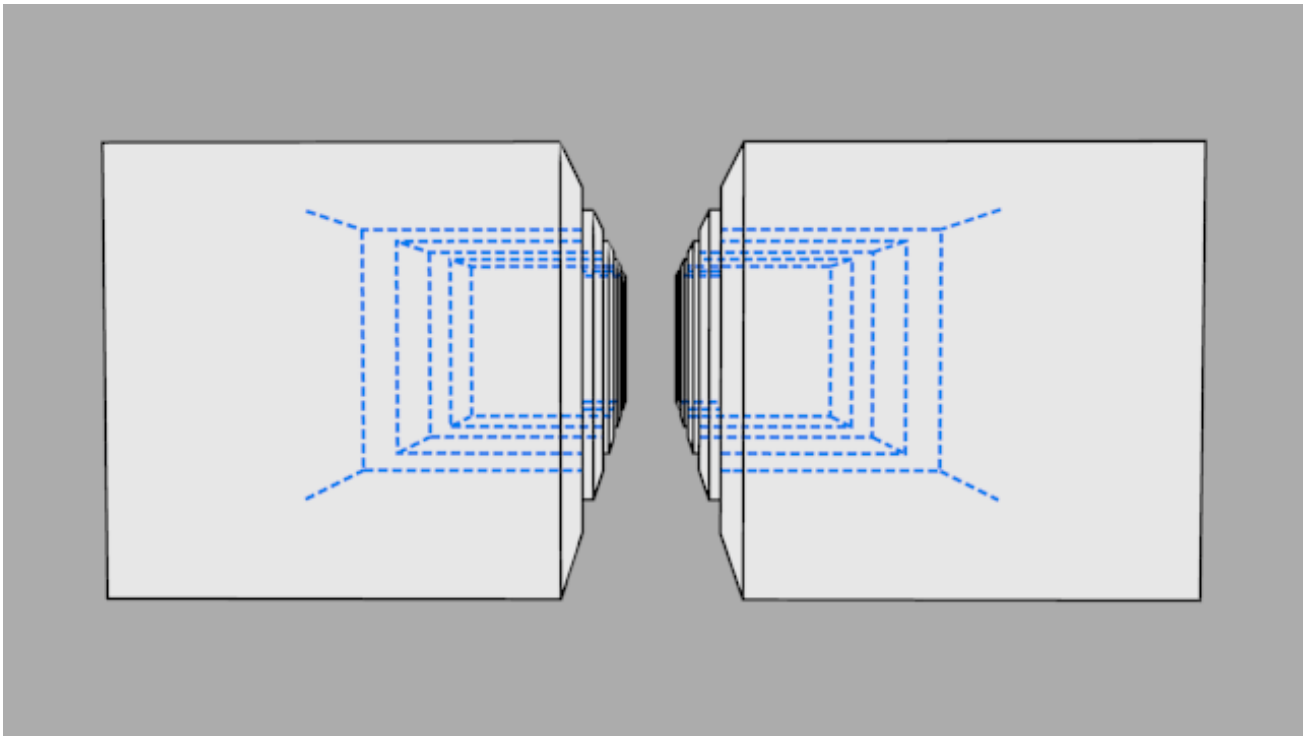


Fig. 2.2087: QI Range proof of concept demo, Start: 3, End: 7, by LightBWK (Sample blend-file)

Edge Types

Edge types are basic algorithms for the selection of lines from geometry. When using the parameter editor you have to choose at least one edge type in order to get a render output, but several edge types can be combined in one line set. Edge types can also be excluded from calculation by pressing the *X* next to them.

Silhouette
Draws silhouettes around your closed objects; it is often good for organic objects

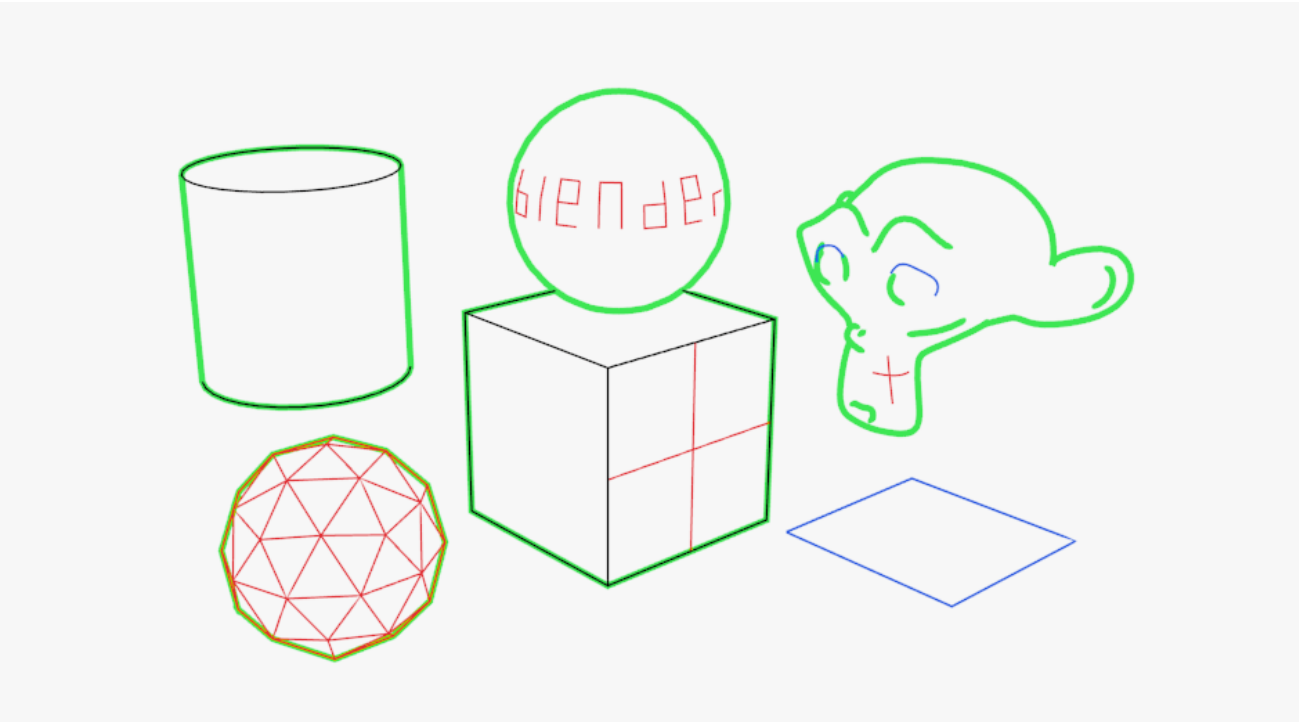


Fig. 2.2088: Examples of some basic edge types: Silhouette (green), Crease (black), Border (blue) and Edge Marks (red) (File:EdgeType.zip by LightBWK)

(like Suzanne & Sphere), and bad for sharp edges, like a box.
It cannot render open mesh objects like open cylinders and flat planes. The output is affected by the *Kr Derivative Epsilon* viewmap setting.

Crease
Shows only edges whose adjacent faces form an angle greater than the

de-
fined
viewmap's
Crease
An-
gle.

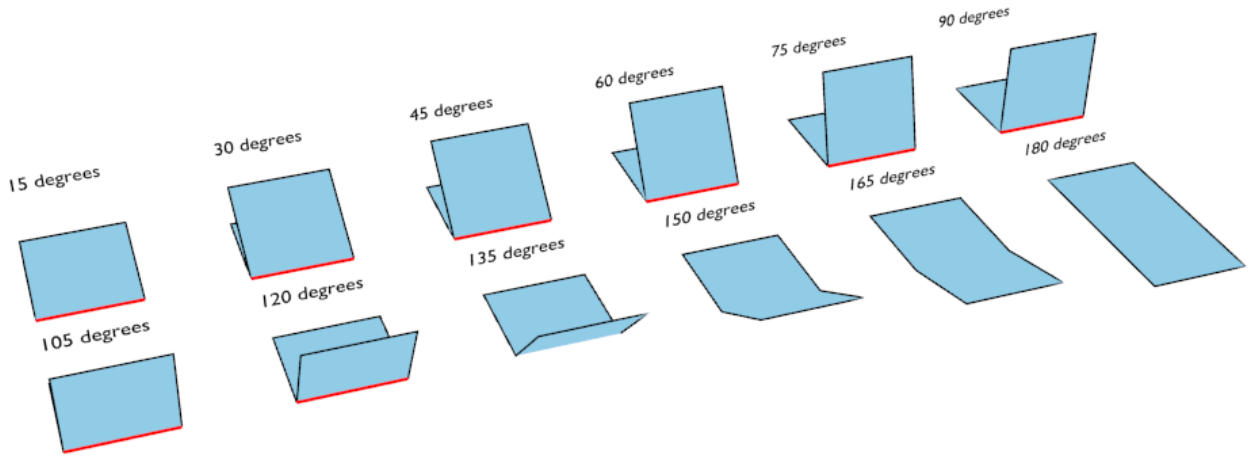


Fig. 2.2089: Crease Angle proof of concept for 121° by LightBWK (the blend-file)

Border
Border
is
for
open/unclosed
edge
meshes;
an
open
cylin-
der
has
an
open
edge
at
the
top
and
bot-
tom,
and
a
plane
is
open

all around. Suzanne's eye socket is an open edge. All open edges will have lines rendered. This depends on the mesh structure.

Edge Marks
Renders marked edges. See *Edge Marks* for details.

Contour
Draws the outer edges and inner open border.

External Contour
Draws the contour lines, but only on the outer edges.

Suggestive Contour
Draws some lines which would form the contour of the mesh if the view-

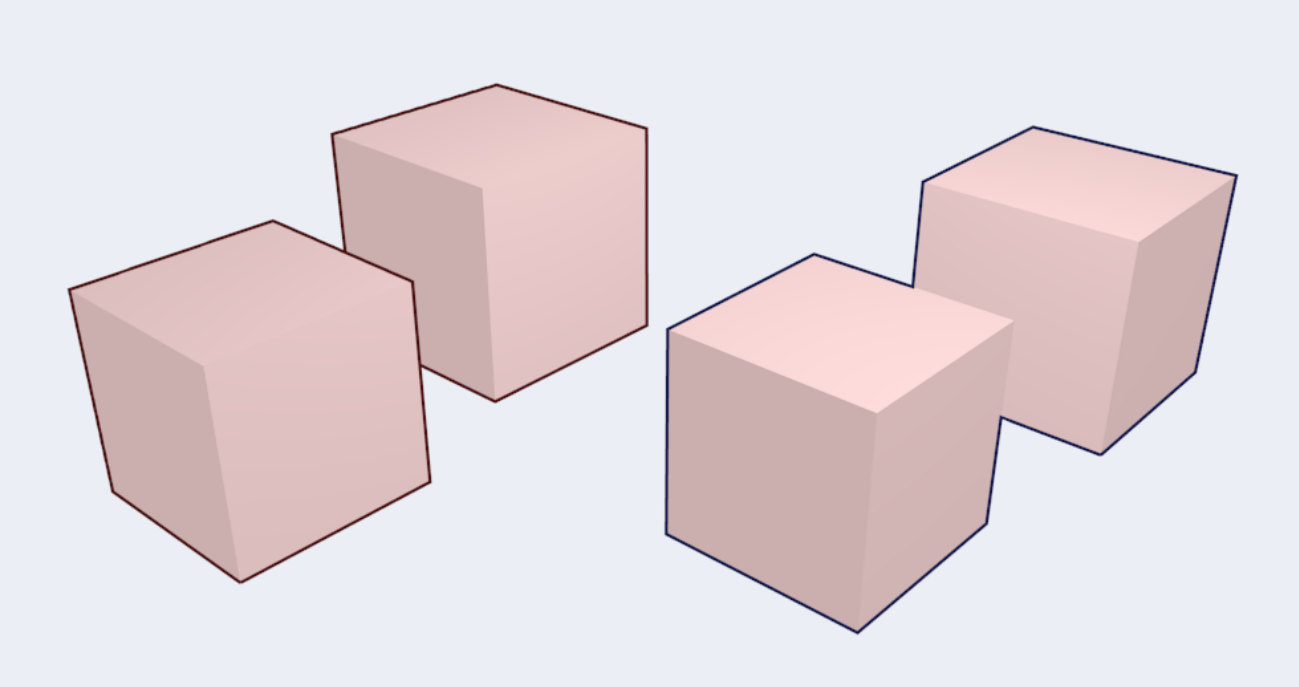


Fig. 2.2090: Left pair: Contour; Right pair: External Contour.

port was shifted. Depends on your viewmap settings for *Kr Derivative Epsilon* and *Sphere Radius* (further information: File:Manual-2.6-Render-Freestyle-PrincetownLinestyle.pdf).

Material Boundary
Draws lines where two materials meet on the same object.

Ridge & Valley
Draws ridges and valleys. Depends on your *Sphere Radius* viewmap settings.

Edge Marks

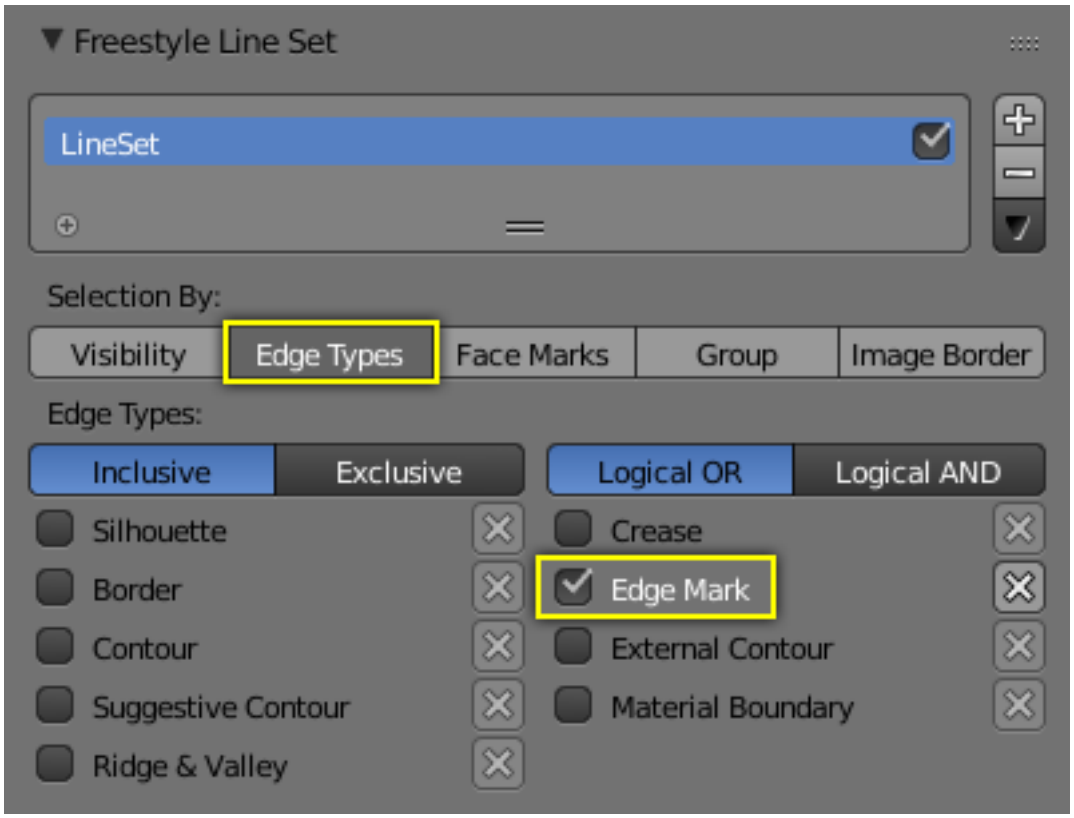


Fig. 2.2091: Edge Mark setting in the Line Sets tab.

In edit mode you can mark "Freestyle Edges" in

the same manner you can mark "Seams" for UV unwrapping or "Sharp" for edge split. These marked edges are available to render when you select *Edge Mark*.

This is done as follows:

- Select your mesh and tab into *Edit Mode*.
- Select the edges you want to be marked.
- Press `Ctrl-E` and select *Mark Freestyle*

Edge.

Edge marks are useful when you want to draw lines along particular mesh edges. The examples below explain the use of edge marks.

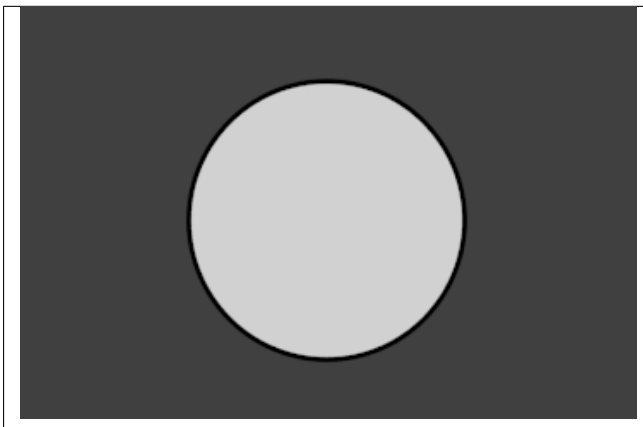


Fig. 2.2093: Render without Edge Marks.

Fig. 2.2094: Render with Edge Marks enabled.

With edge marks enabled, the previously-marked lines

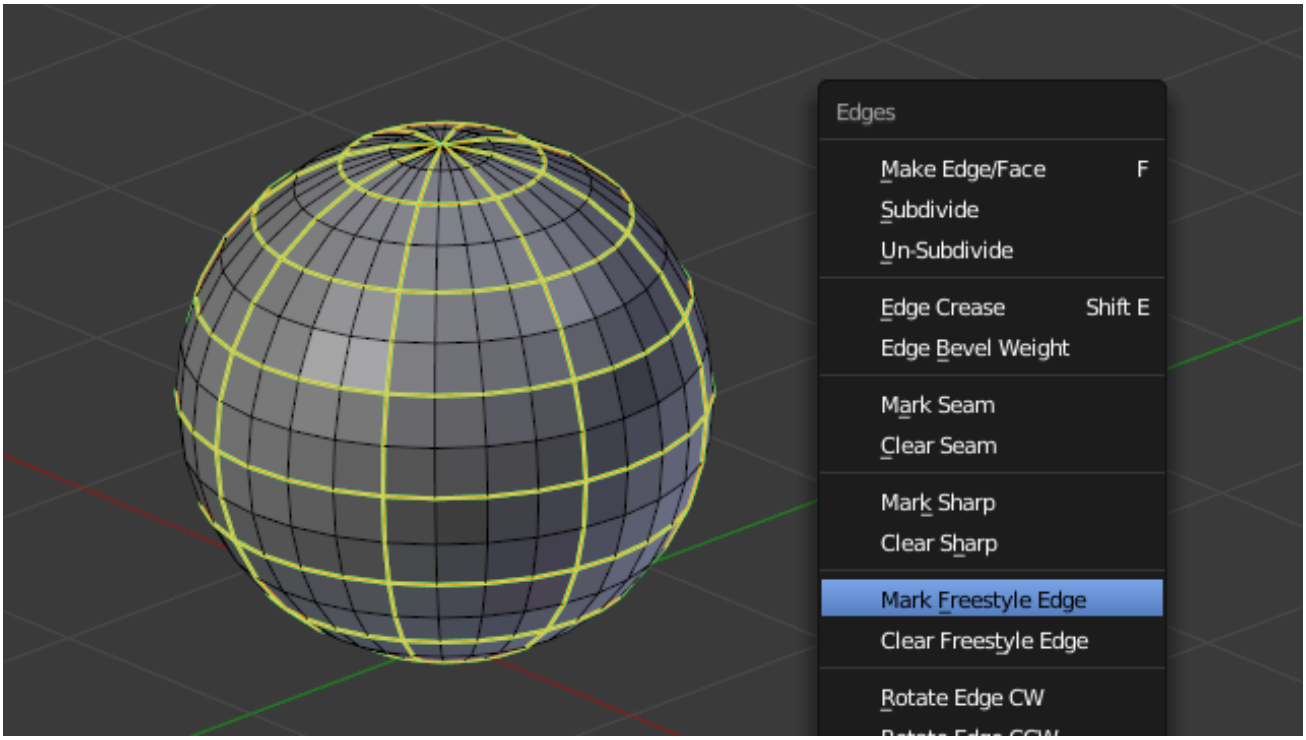


Fig. 2.2092: Marking Freestyle Edges in edit mode.
The edge marks are highlighted in green.

are always rendered. You can see the black contour lines and the blue lines that are made with edge marks.

What are edge marks good for?

- When you need to render marks on an almost-flat plane, when other edge types cannot detect any line.
- When you want full control of edge rendering. Often used for edges of squarish shapes.
- Mark the whole base mesh to be rendered for base mesh

pre-
view.

What
are
edge
marks
not
good
for?

- Round
outer
edges
(use
in-
stead
*Con-
tour/External
Con-
tour/Silhouette*).

Face Marks

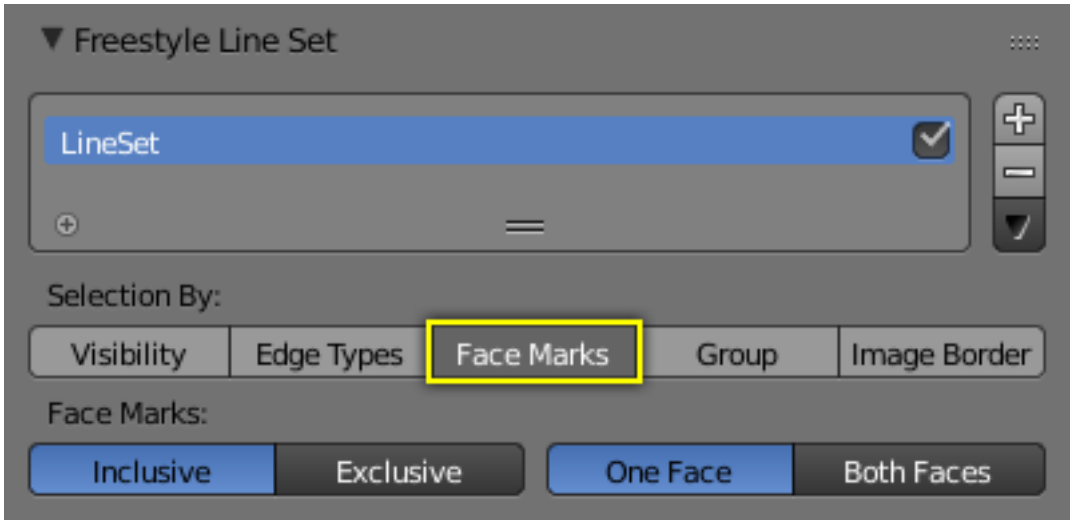


Fig. 2.2095: Face mark options.

To
set
a
face
mark:

- Select
a
mesh
and
tab
into

Edit Mode.

- Select the faces you want to be marked.
- Press `Ctrl-F` and select *Mark Freestyle Face*.

  Face marks are useful for removing lines from certain areas of a mesh.

  In this example, two faces of the default cube are marked like the image

on the left. On the right is a render without face marks activated.

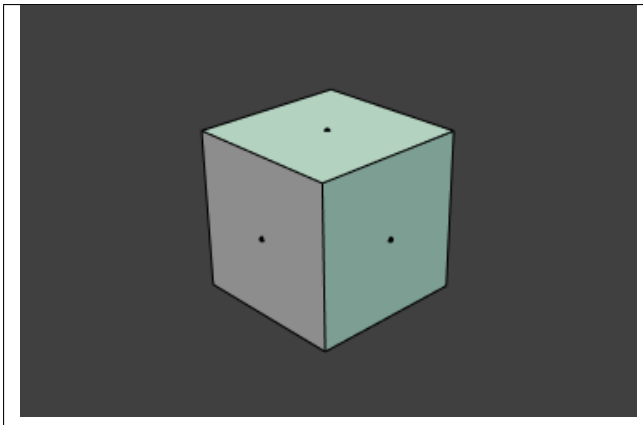


Fig. 2.2096: Marked Faces (Edit Mode).

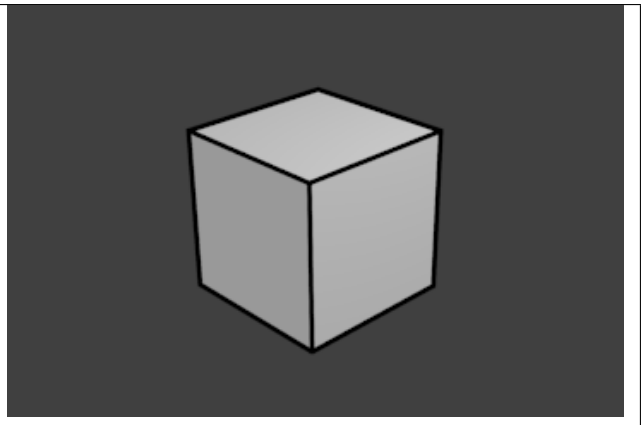


Fig. 2.2097: Render Output.

The line selection can be controlled via inclusion and faces options:

Inclusive/Exclusive
Whether to include or exclude edges match-

ing
de-
fined
face
mark
con-
di-
tions
from
the
line
set.

One Face
(De)select
all
edges
which
have
one
or
both
neigh-
bor
faces
marked.

Both Faces
(De)select
all
edges
which
have
both
of
their
neigh-
bor
faces
marked.

The
im-
age
be-
low
shows
the
re-
sult-
ing
com-
bi-
na-
tions.

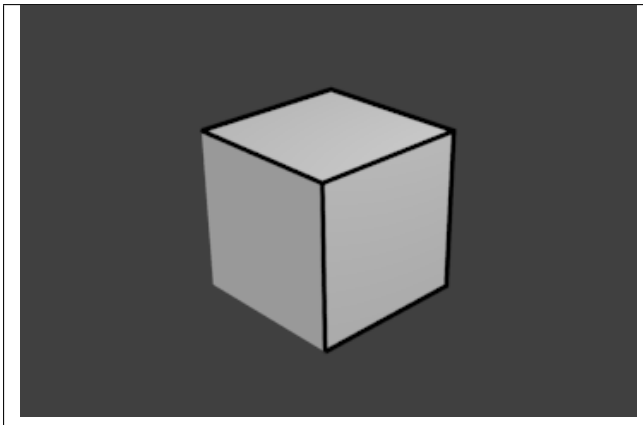


Fig. 2.2098: Inclusive, One Face.

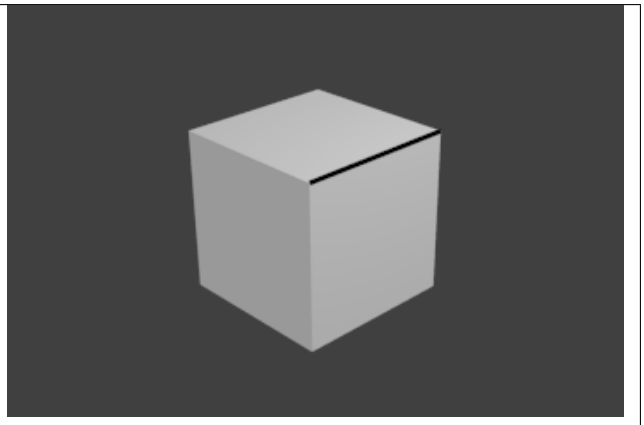


Fig. 2.2099: Inclusive, Both Faces.

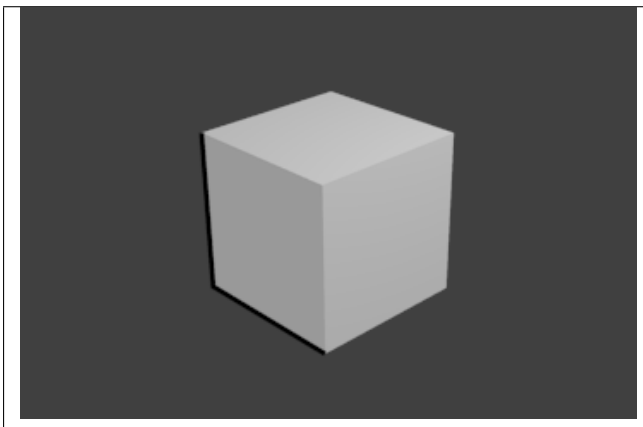


Fig. 2.2100: Exclusive, One Face.

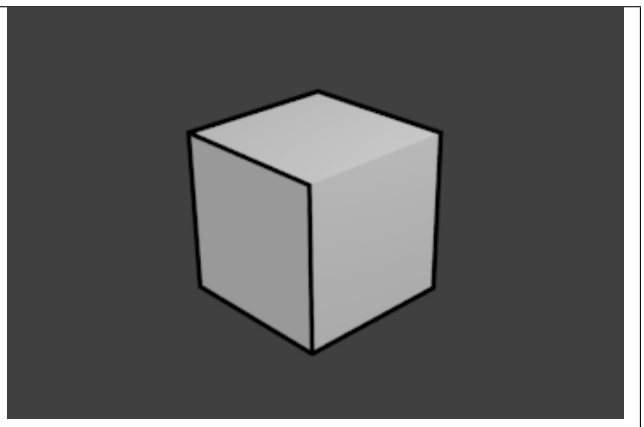


Fig. 2.2101: Exclusive, Both Faces.

Group

You can include or exclude objects for line calculation, based on their belonging to a group.

Group
The name of the object group to use.

Inclusive/Exclusive
Whether to include or exclude lines from those objects in this line set.

Image Border

If enabled, Freestyle only takes geometry within the image border into consideration for line cal-

culation. This reduces render times but increases continuity problems when geometry is moved out of and into camera view.

Line Style & Modifiers

Introduction

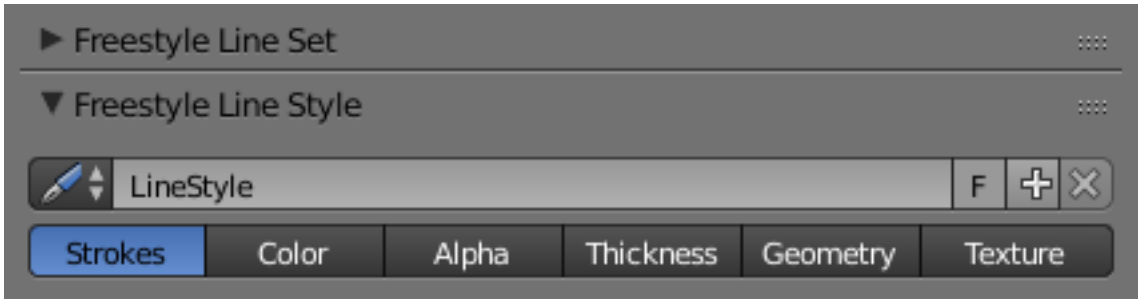


Fig. 2.2102: Line Style UI.

In Freestyle, the line style settings define the appearance of a line set using five main aspects:

- *Stroke*
- *Color*

- *Alpha*
- *Thickness*
- *Geometry*
- *Texture*

These allow you to get many different styles of renders (technical draw, rough sketch, cartoon, oriental calligraphy, etc.).

You can create as many line styles as you wish, and reuse a given line style for several line sets

by selecting it from the select menu next to its name.

Note: Length Unit

Unless otherwise specified, all lengths in line style settings are in pixels (either relative or absolute, as specified in the *core options*).

Stroke

Strokes are the final ren-

Fig. 2.2103: Line Style demo File:LineStyles.zip.

dered lines. Yet you can tweaks them, for example, by removing the ones longer/shorter than some threshold, chaining lines into a single stroke or breaking a stroke into several ones based on angles, dashed pattern, etc.

Chaining

By default

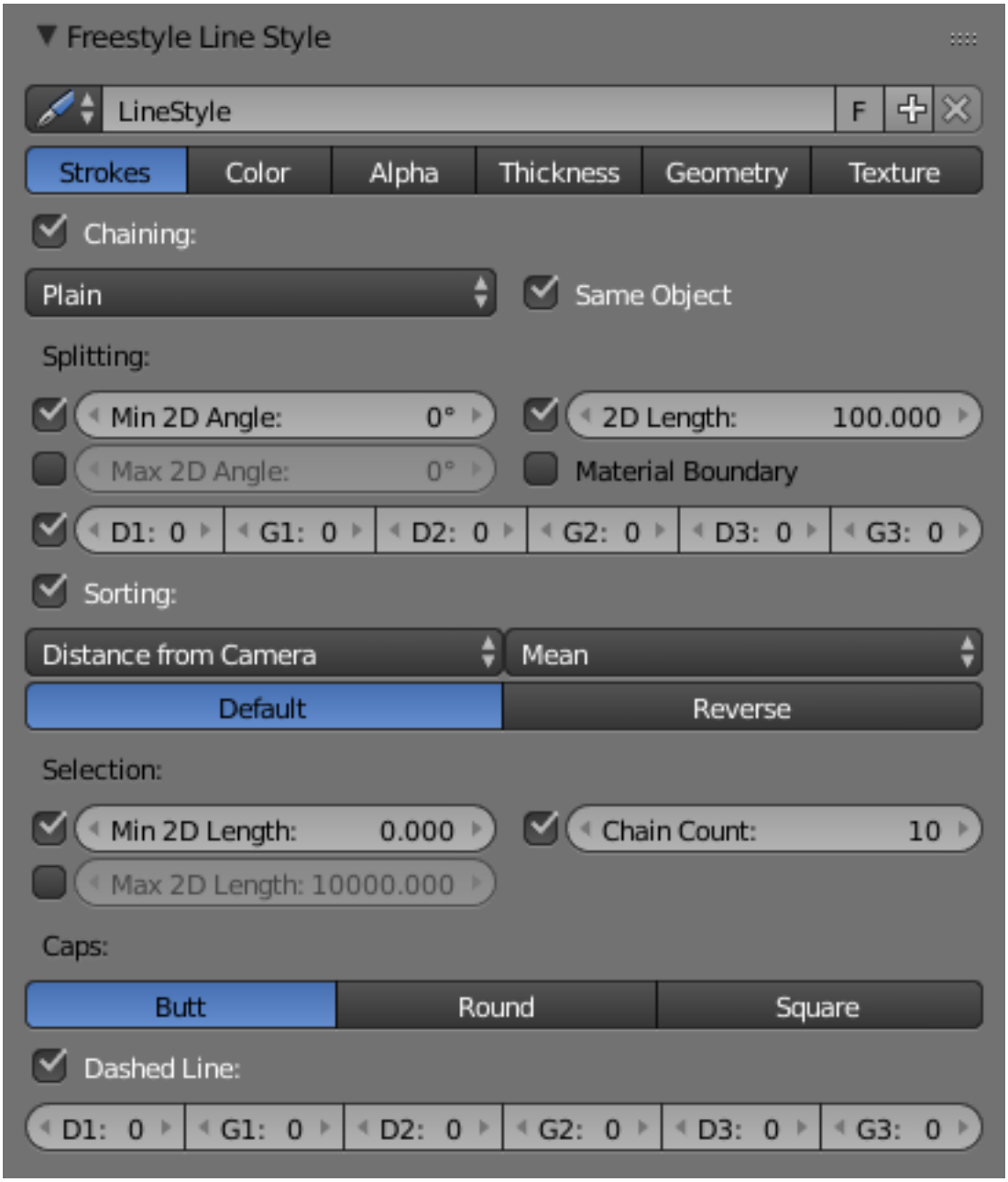


Fig. 2.2104: Stroke Line Style.

all retrieved lines from the line set are chained together. There are two basic chaining methods:

Plain
The default chaining method; it creates simple chains.

Sketchy
This chaining option allows for generating chains of feature edges with sketchy multiple

strokes. Basically, it generates *Round* strokes instead of a single one. It is only really useful if you use some random-driven modifiers in the line style!

Rounds
It specifies the number of rounds in sketchy strokes.

Chaining can also be turned off to render each line separately, which can be useful for line styles which depend on accurate representation of the line set.

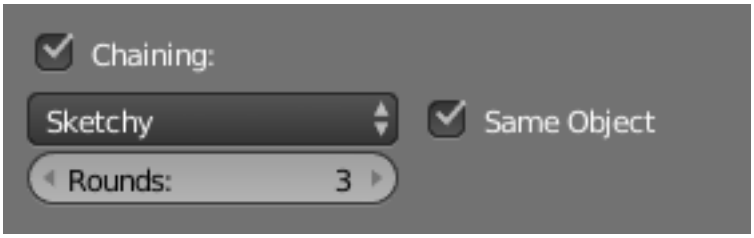


Fig. 2.2105: Chaining.

Splitting

You can split up chains of Freestyle lines by checking one of the following:

Material Boundary
Splits chains of feature edges if they cross from one material to another.

Min 2D Angle and Max 2D Angle
Splits chains of feature edges when they

make a 2D angle above (or below) a minimum (or maximum) threshold.

Fig. 2.2106: Splitting.

2D Length
Splits chains when they are longer than the given value.

D1/G1/D2/G2/D3/G3
Splits the chains using the given dashed pattern ("D" stands for "dash",

"G" stands for "gap"; see also *Dashed Line*).

Sorting

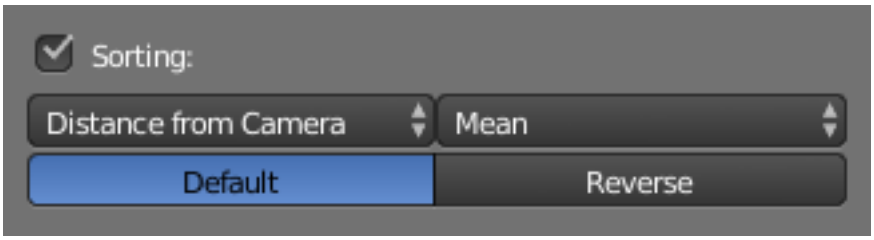


Fig. 2.2107: Sorting.

You can sort the order of your strokes, allowing the lines to stack in the order given.

Sort key
Choose which way you would like to sort your strokes.

Integration Type
Use in

tandem with the Sort Key to determine the range for sorting.

Sort Order
With the given result you can choose to "Reverse" the sort order.

Selection

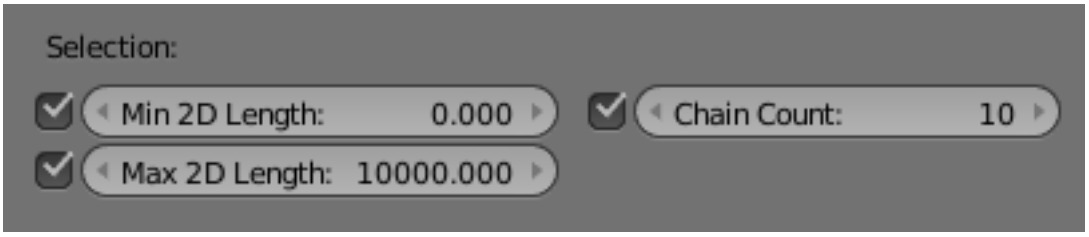


Fig. 2.2108: Selection.

You can also choose to only select (i.e. render) chains

longer than *Min 2D Length* and/or shorter than *Max 2D Length*.

Caps

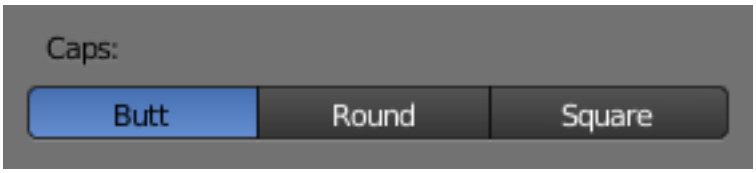


Fig. 2.2109: Line tip caps.

You can choose between three types of line caps:

Butt
Flat cap, exactly at the point the line ends.

Round
A half circle centered on the end point of

the line.

Square
A square centered on the end point of the line (hence, like the circle, the drawn end of the line is slightly extended compared to its computed value).

Dashed Line

By enabling the *Dashed Line* check box, you can specify three pairs of dash and gap

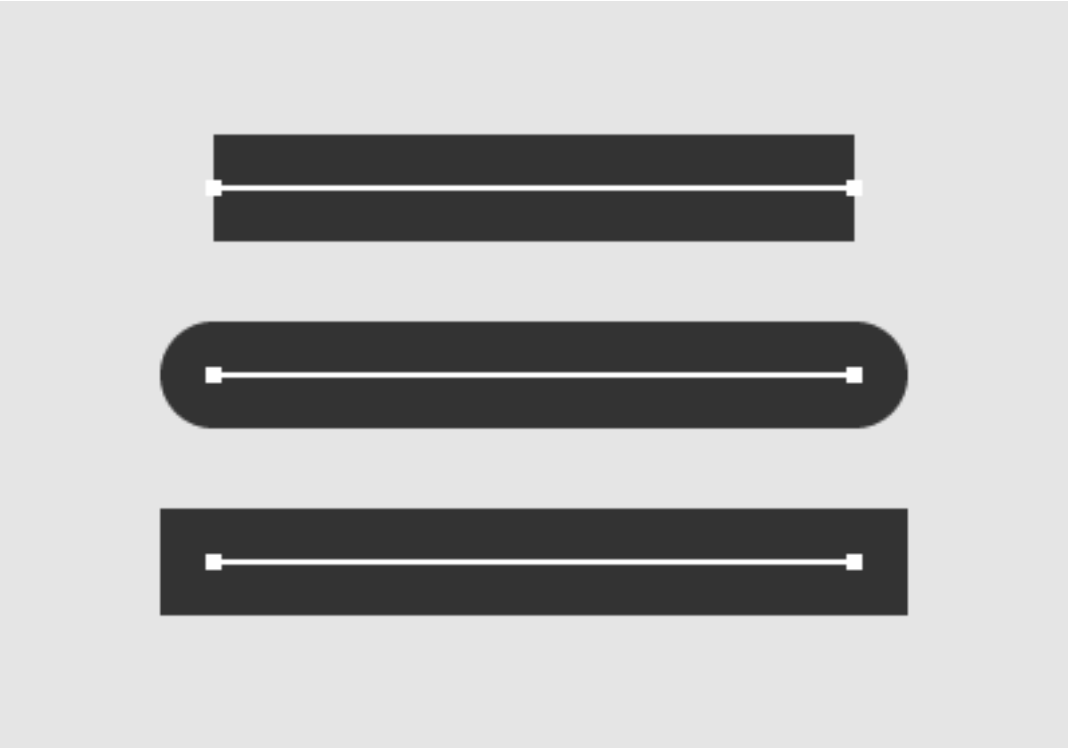


Fig. 2.2110: Line caps example.

Fig. 2.2111: Dashes Line UI.

lengths. Dash values define the lengths of dash strokes, while gap values specify intervals between two dashes.

If a zero gap is specified, then the corresponding dash is ignored even if it has a non-zero value.

Dashes are treated as separate strokes, meaning that you can apply line

caps, as well as color, alpha and thickness modifiers.

Color

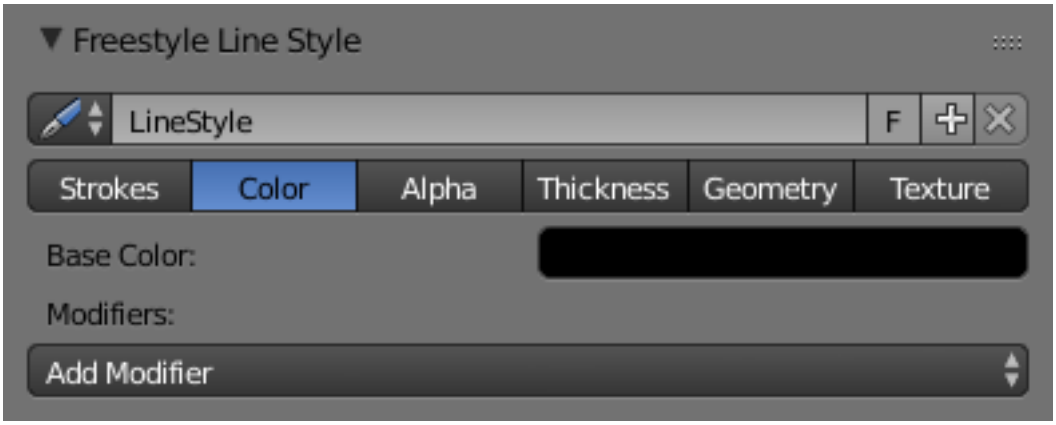


Fig. 2.2112: Line Style Color UI.

In this tab you control the color of your strokes.

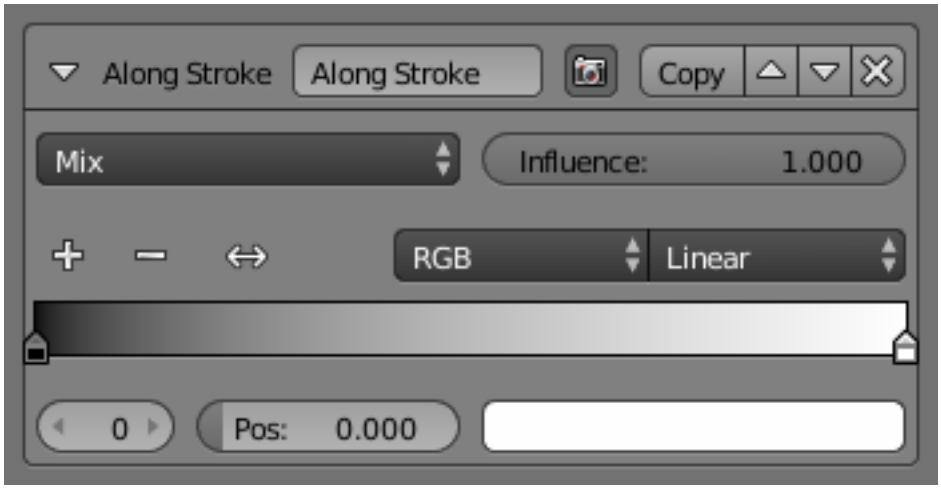
Base Color
The base color for this line style.

Modifiers

There are several color modifiers available, which can be mixed with the base color using the usual methods (see for example the *Mix compositing node* for further discussion of this topic). As with other modifier stacks in Blender, they are applied from top to bottom.

Influence
How much the result of this modifier affects the current color.

Along Stroke



The *Along Stroke* modifier alters the base color with a new one from a given color ramp mapped along each stroke's length. In other words, it applies a color ramp along each stroke.

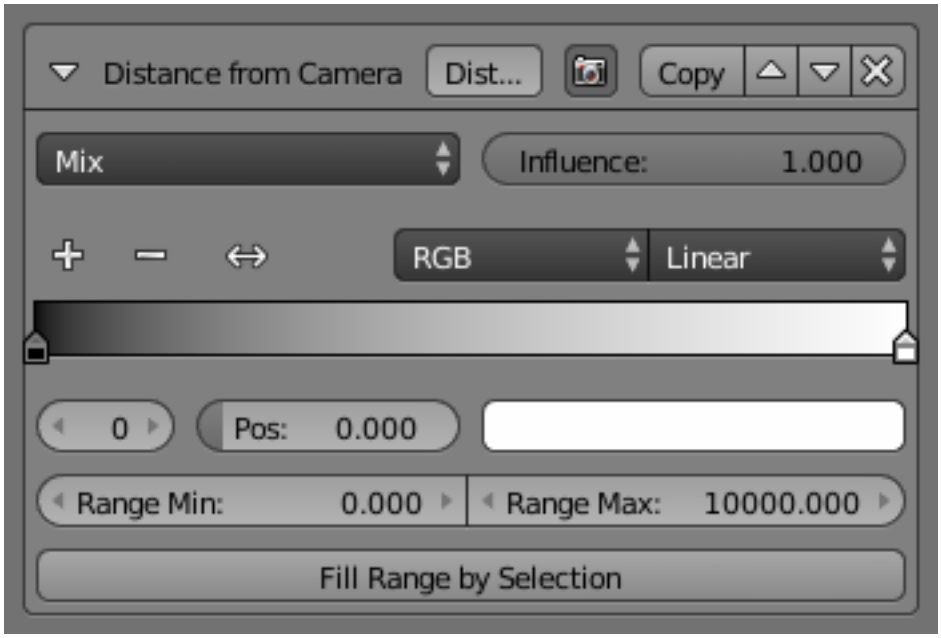
Color Ramp
A standard Blender

color ramp.

Distance from Camera

The *Distance from Camera* color modifier alters the base color with a new one from a given color ramp, using the distance to the active camera as the parameter.

Range Min and Range Max
The limits of the mapping from "distance to camera" to "color



in ramp". If the current point of the stroke is at *Range Min* or less from the active camera, it will take the start color of the ramp, and conversely, if it is at *Range Max* or more from the camera, it will take the end color of the ramp. These values are in the current scene's units, not in pixels!

Fill Range by Selection
Set the min/max range values from the distances between the current selected

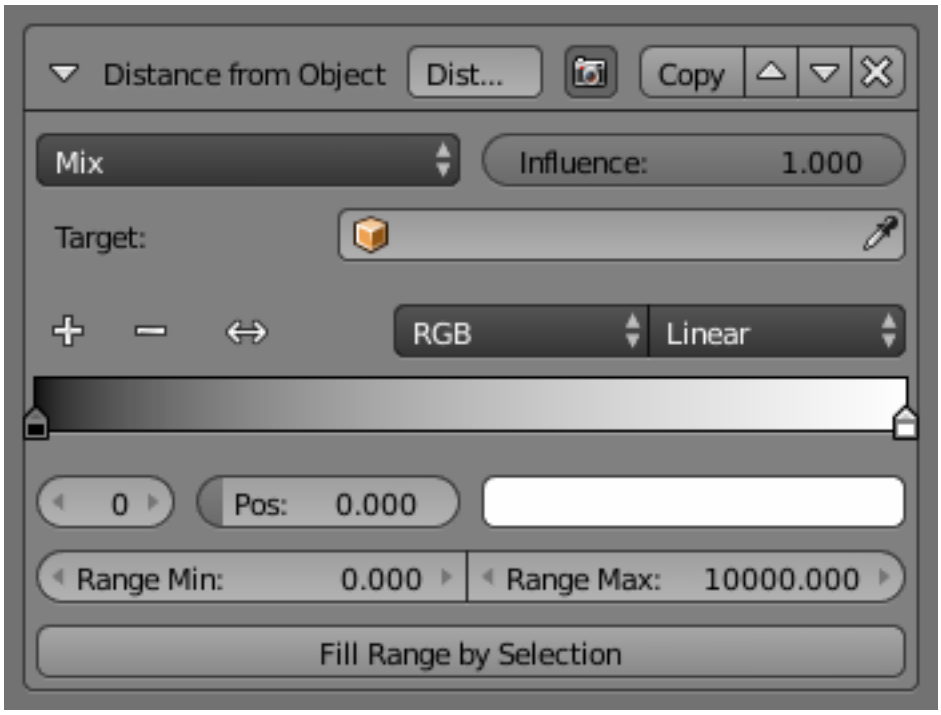
objects and the camera.

The other settings are those of the standard Blender color ramp!

Distance from Object

The *Distance from Object* color modifier alters the base color with a new one from a given color ramp, using the distance to a given

object as the parameter.



Target
The object to measure distance from.

Range Min and Range Max
The limits of the mapping from "distance to object" to "color in ramp". If the current

point of the stroke is at *Range Min* or less from the target, it will take the start color of the ramp, and conversely, if it is at *Range Max* or more from the target, it will take the end color of the ramp. These values are in the current scene's units, not in pixels!

Fill Range by Selection
Set the min/max range values from the distances between the current selected objects and the target.

The other settings are those of the standard Blender color ramp!

Material

The *Material*

color modifier alters the base color with a new one taken from the current material under the stroke.

You can use various properties of the materials, among which many are mono-component

(i.e. give B&W results). In this case, an optional color ramp can be used to map these grayscale values to colored ones.

If used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.

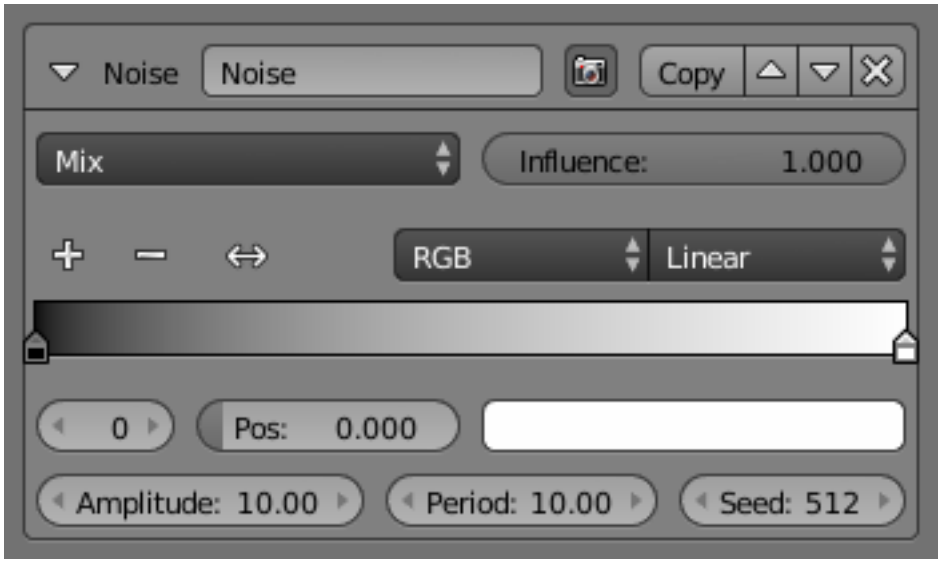
Noise

The *Noise* modifier uses a pseudo-random num-

Fig. 2.2113: Material modifiers demo by T.K. File:Lilies_Color_Material.zip.

ber gen- er- a- tor to vari- ably dis- tribute color along the stroke.

Amplitude
The max- i- mum value of the noise. A higher am- pli- tude means a



less transparent (more solid) stroke.

Period
The period of the noise. This means how quickly the color value can change. A higher value means a more smoothly changing color along the stroke.

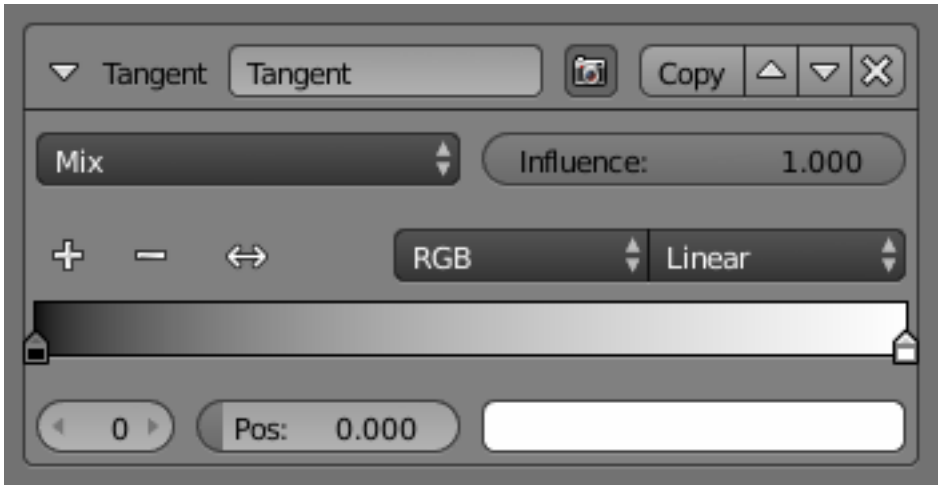
Seed
Seed

used by the pseudo-random number generator.

Color Ramp
A standard Blender color ramp that maps noise values to a stroke color.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke's vertices.



Color Ramp
A standard Blender color ramp that maps the traveling direction to a stroke color.

Min Angle and Max Angle
The range of input values to the mapping. Out-of-range input values

will be clamped by the Min and Max angles and their corresponding color values.

Curvature 3D

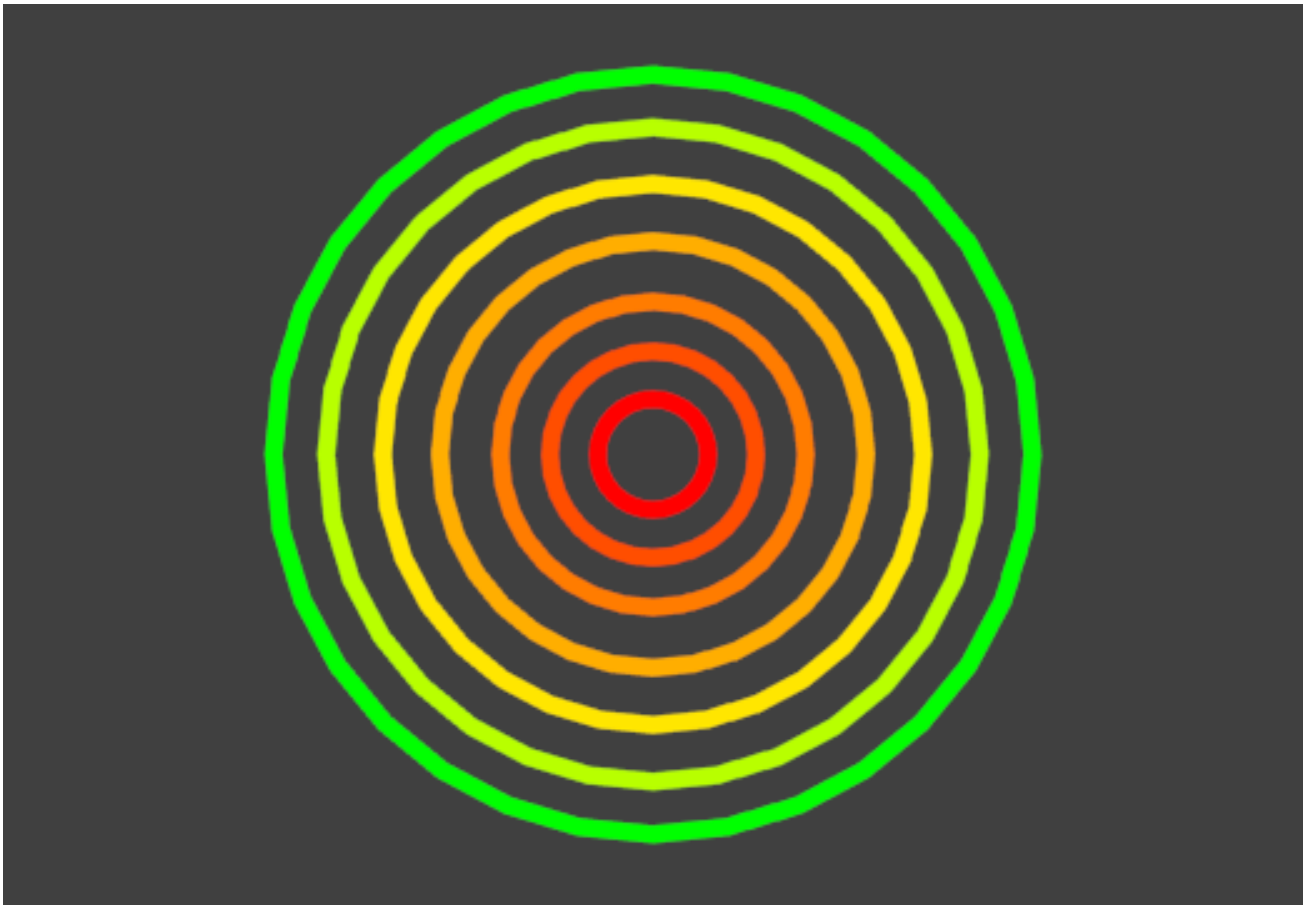


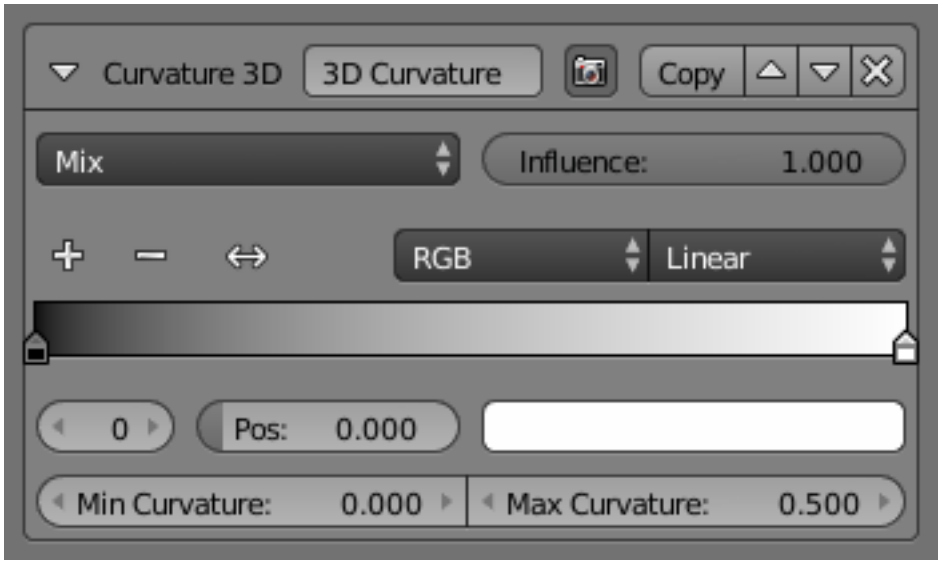
Fig. 2.2114: Curvature 3D modifier demo by T.K. File:Render_freestyle_modifier_curvature_3d.blend.

A modifier based on radial

curvatures of the underlying 3D surface. The curvature of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross-section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option

has to be turned on and the object needs to have *Smooth Shading*.



Color Ramp
A standard Blender color ramp that maps the radial curvature to a stroke color.

Min Curvature and Max Curvature
The limits of the color ramp. If the current point of the stroke

is at *Min Curvature* or less from the target, it will take the start color of the mapping, and conversely, if it is at *Max Curvature* or more from the target, it will take the end color of the mapping.

Crease Angle

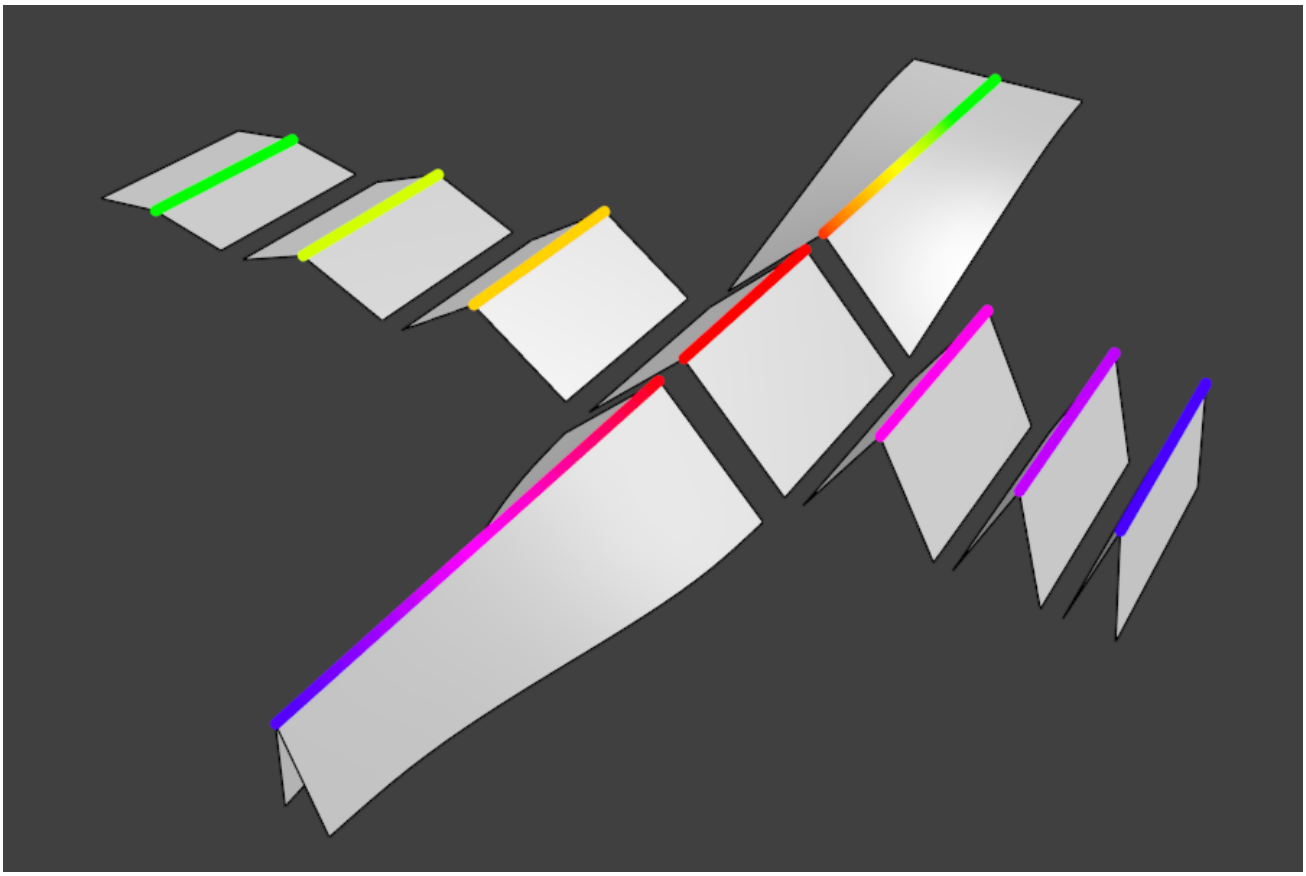
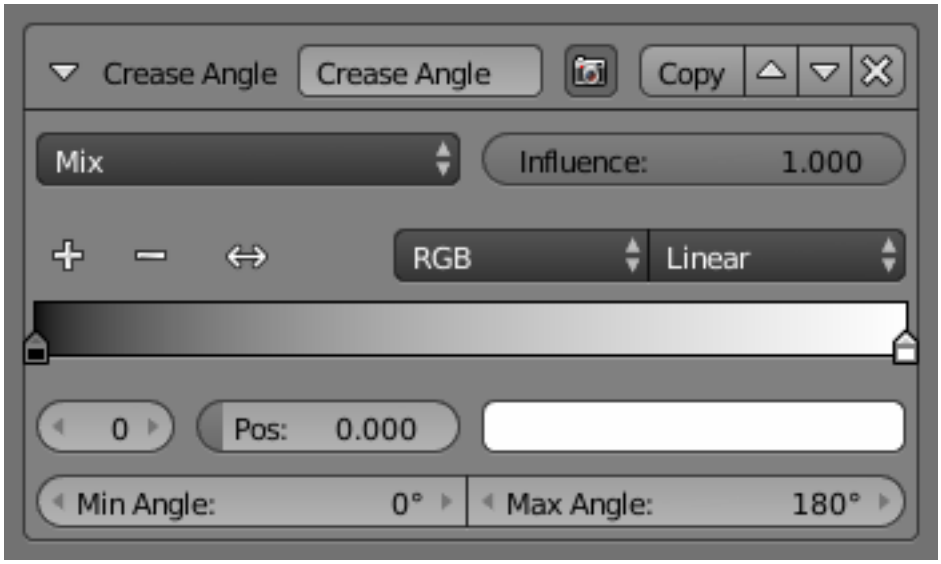


Fig. 2.2115: Crease Angle modifier demo by T.K. File:Render_freestyle_modifier_crease_angle.blend.

A modifier

based on the Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e., the edge does not have the *Crease Angle nature*, its color values are not touched by the modifier.



Color Ramp
A standard Blender color ramp that maps

the crease angle to a stroke color.

Min Angle and Max Angle
The range of input values to the mapping. Out-of-range crease angle values will be clamped by the Min and Max angles and their corresponding color values.

Alpha

In this tab you control the alpha (transparency) of your

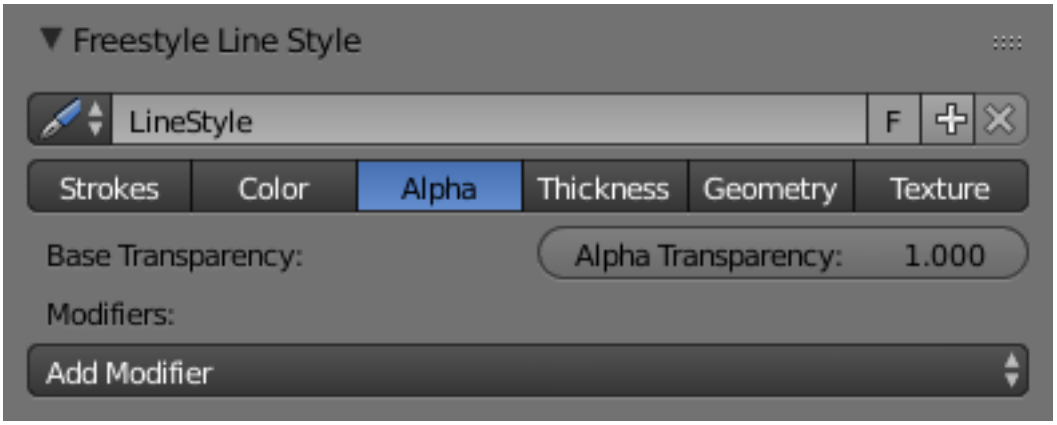


Fig. 2.2116: Line Style Alpha UI.

strokes.

Base Transparency
The base alpha for this line style.

Modifiers

There are four alpha modifiers available, which can be mixed with the base alpha using a subset of the usual

methods (see for example the *Mix compositing node* for further discussion of this topic). As with other modifier stacks in Blender, they are applied from top to bottom.

Influence
How much the result of this modifier affects the current transparency.

Along Stroke

The *Along Stroke* modifier alters the base alpha with a new one from

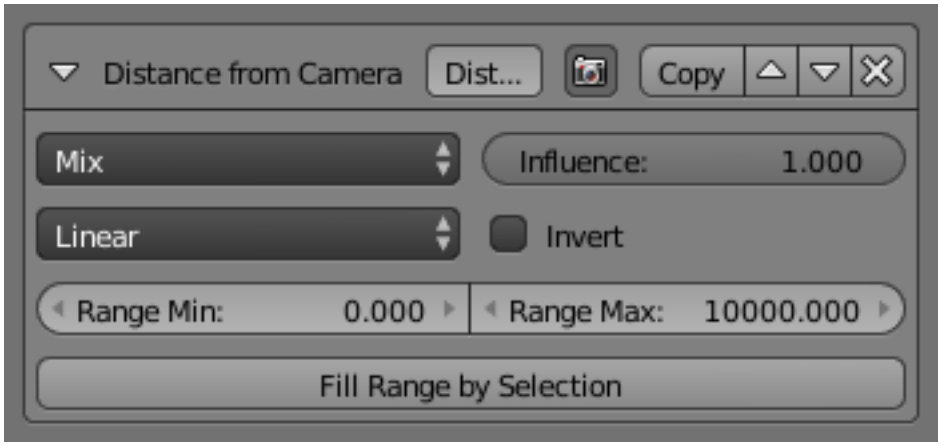
either a linear progression or a custom curve, mapped along each stroke's length. In other words, it applies the selected progression along each stroke.

Mapping
Either a linear progression (from `0.0` to `1.0`, which may be inverted with the *Invert* option), or a custom mapping curve.

Distance from Camera

The *Distance from Cam-*

era modifier alters the base alpha with a new one from either a linear progression or a custom curve, using the distance to the active camera as parameter.



Mapping
Either a linear progression (from `0.0` to `1.0`, which may

be inverted with the *Invert* option), or a custom mapping curve.

Range Min and Range Max
The limits of the mapping from "distance to camera" to "alpha in mapping". If the current point of the stroke is at *Range Min* or less from the active camera, it will take the start alpha of the mapping, and conversely, if it is at *Range Max* or more from the camera, it will take the end alpha of the mapping. These values are in the current scene's units, not in pixels!

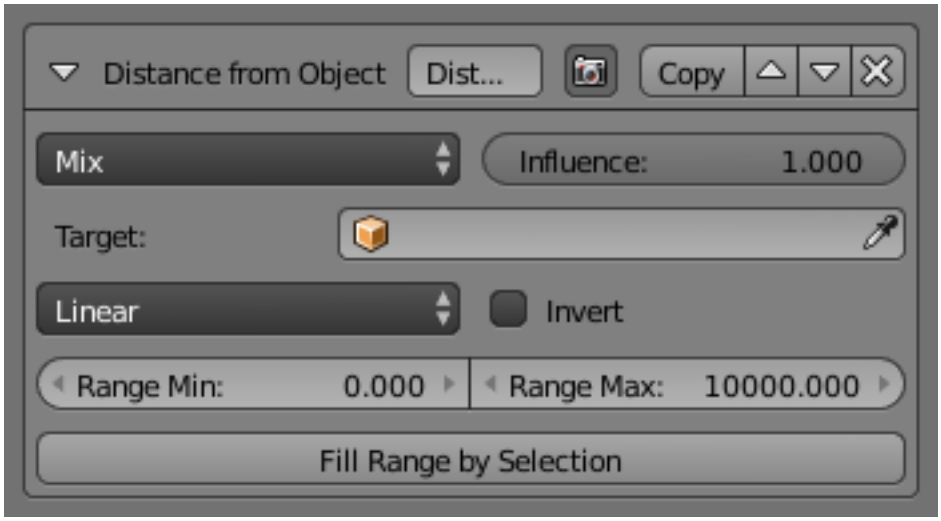
Fill Range by Selection
Set the min/max range values

from the distances between the current selected objects and the camera.

Distance from Object

The *Distance from Object* modifier alters the base alpha with a new one from either a linear progression or a custom curve, using the distance to a given object as parameter.

Target



The object to measure distance from.

Mapping
Either a linear progression (from `0.0` to `1.0`, which may be inverted with the *Invert* option), or a custom mapping curve.

Range Min and Range Max
The limits of the mapping from "distance to object" to "alpha in mapping". If the current point of the stroke is at *Range Min* or less from the target, it will take the start alpha of the mapping, and conversely, if it is at *Range Max* or more from the target, it will take the end alpha of the mapping. These values are in the current scene's units, not in pixels!

Fill Range by Selection
Set the min/max range values from the distances between the current selected objects and the target.

Material

The *Material* modifier alters the base alpha with a new one taken from the current material under the stroke.

You can use various properties of the materials, among which some are multi-components (i.e. give RGB results).

In that case, the mean value will be used.

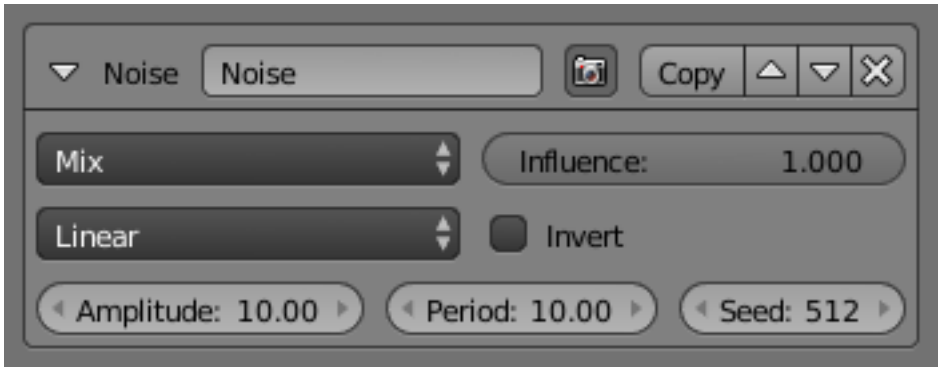
Mapping
Either a linear progression (from `0.0` to `1.0`, which may be inverted with the *Invert* option), or a custom mapping curve. Note the linear non-inverted option is equivalent to "do nothing", as original values from materials are already in the `[0.0,1.0]` range.

If used with

the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.

Noise

The *Noise* modifier uses a pseudo-random number generator to variably distribute transparency along the stroke.



Amplitude
: The maximum value of the noise. A higher amplitude means a less transparent (more solid) stroke.

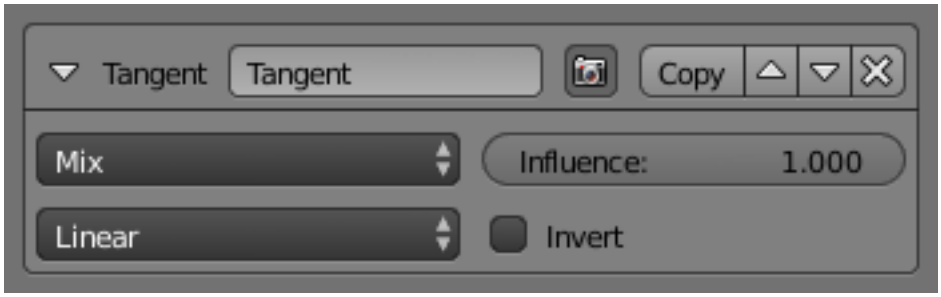
Period
: The period of the noise. This means how quickly the alpha value can change. A higher value means

a more smoothly changing transparency along the stroke.

Seed
Seed used by the pseudo-random number generator.

Mapping
Either a linear progression (from `0.0` to `1.0`, which may be inverted with the *Invert* option), or a custom mapping curve. Note the linear non-inverted option is equivalent to "do nothing", as original values from materials are already in the `[0.0,1.0]` range.

Tangent



This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke's vertices.

Mapping
Either a linear progression (from `0.0` to `1.0`, which may be inverted with

the *Invert* option), or a custom mapping curve. Note the linear non-inverted option is equivalent to "do nothing", as original values from materials are already in the `[0.0,1.0]` range.

Min Angle and Max Angle
The range of input values to the mapping. Out-of-range input values will be clamped by the Min and Max angles and their corresponding alpha values.

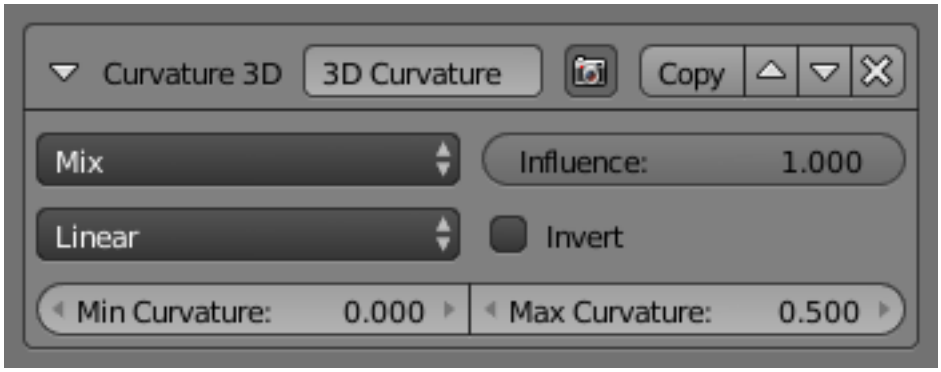
3D Curvature

A mod-

i-
fier
based
on
ra-
dial
cur-
va-
tures
of
the
un-
der-
ly-
ing
3D
sur-
face.
The
cur-
va-
ture
of
a
2D
curve
at
a
point
is
a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross-section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For
ra-
dial
cur-
va-
tures
to
be
cal-
cu-
lated
(and
there-
fore
for
this
mod-
i-
fier
to
have
any
ef-
fect),

the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.



Mapping

Either a linear progression (from 0.0 to 1.0, which may be inverted with the *Invert* option), or a custom mapping curve. Note the linear non-inverted option is equivalent to "do nothing", as original values from materials are already in the (0.0 to 1.0) range.

Min Curvature and Max Curvature

The limits of the mapping. If the current point of the stroke is at *Min Curvature* or less from the target, it will take the start alpha of the mapping, and conversely, if it is at *Max Curvature* or more from the target, it will take the end alpha of the mapping.

Crease Angle

A modifier based on the Crease Angle (angle between two adjacent faces).

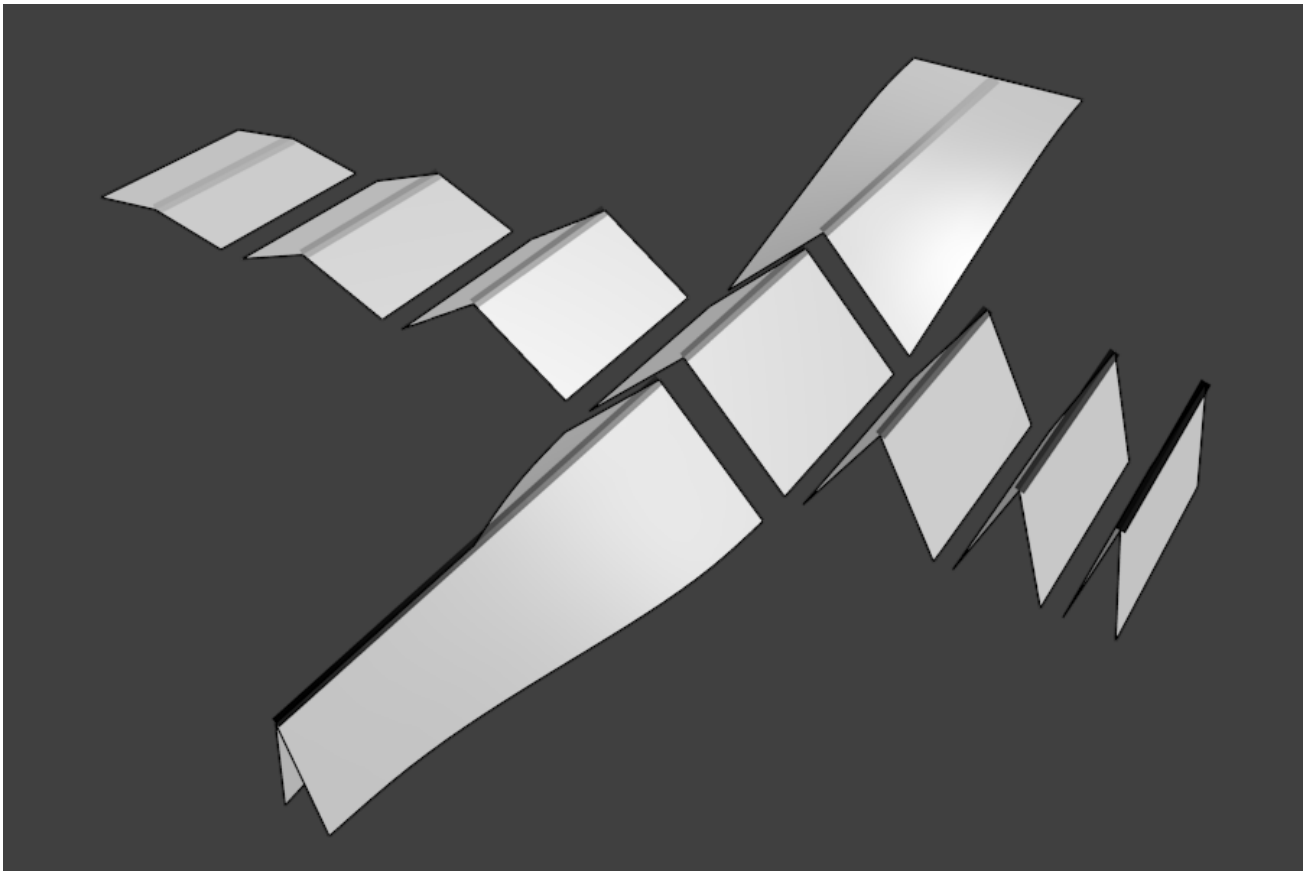
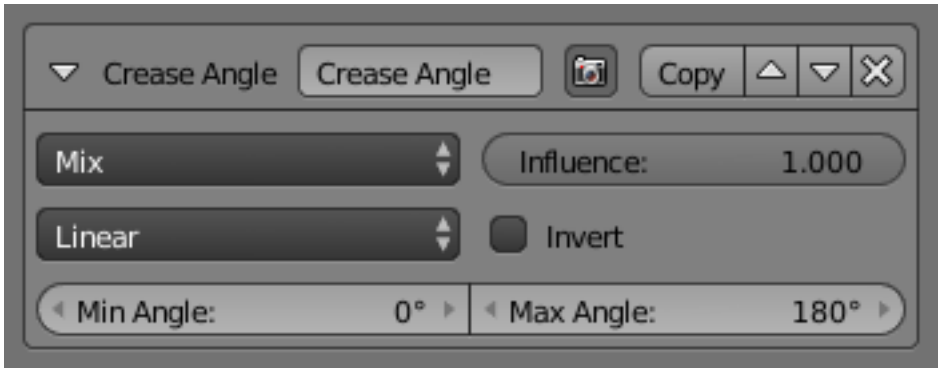


Fig. 2.2117: Crease Angle modifier demo by T.K. File:Render_freestyle_modifier_crease_angle.blend.

If a stroke segment does not lie on a crease (i.e., the edge does not have the *Crease Angle nature*, its alpha value is not touched by this modifier.



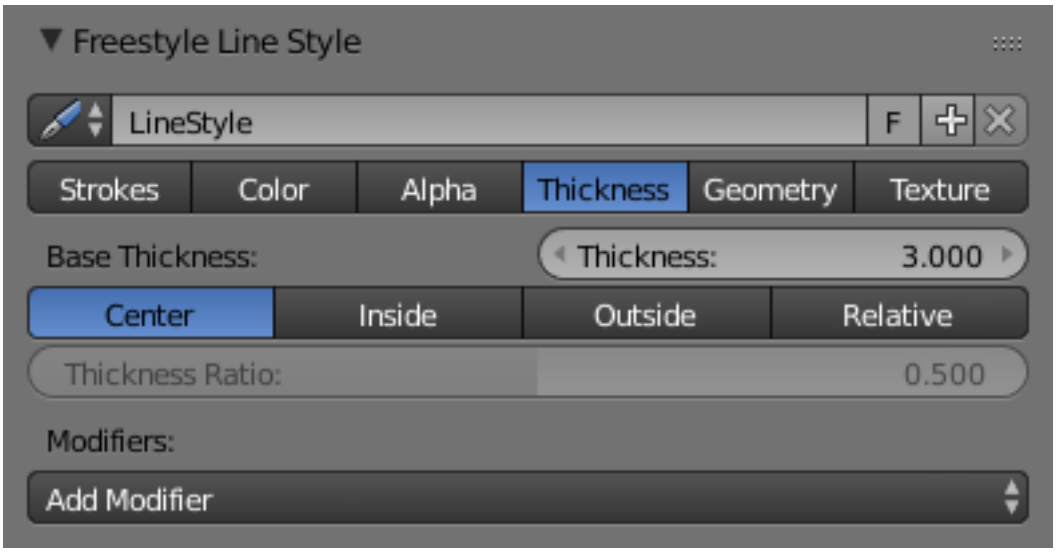
Mapping
Either a linear progression (from 0.0 to 1.0, which may be inverted with the *Invert* option), or a custom mapping curve. Note the linear non-inverted option is equivalent to "do nothing", as original values from materials are already in the (0.0 to 1.0) range.

Min Angle and Max Angle
The range of input values to the mapping. Out-of-range input values will be clamped by the

Min and Max angles and their corresponding alpha values.

Thickness

In this tab you control the thickness of your strokes.



Base Thickness
The base thickness for this line style.

Thickness Position
Control the

position of stroke thickness from the original (backbone) stroke geometry. There are four choices:

Center
The thickness is evenly split to the left and right side of the stroke geometry.

Inside
The strokes are drawn within object boundary.

Outside
The strokes are

drawn outside the object boundary.

Relative
This allows you to specify the relative position by a number between `0.0` (inside) and `1.0` (outside), in the *Thickness Ratio* number button just below.

The thickness position options are applied only to strokes of edge

types *Silhouette* and *Border*, since these are the only edge types defined in terms of the object boundary. Strokes of other edge types are always drawn using the *Center* option.

Modifiers

There are five thickness modifiers available, which can be mixed with the base thickness using a subset of the usual methods (see for example the *Mix compositing node* for further discussion of this topic). As with other modifier stacks in Blender, they are applied from top to bottom.

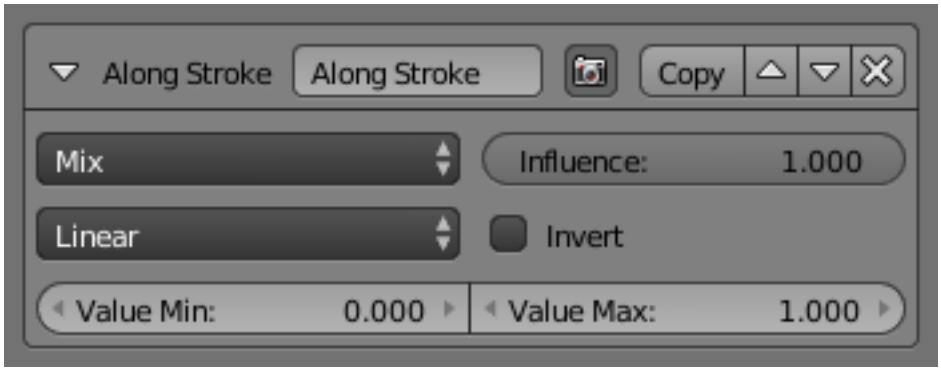
Influence
How much

the result of this modifier affects the current thickness.

Along Stroke

The *Along Stroke* modifier alters the base thickness with a new one from either a linear progression or a custom curve, mapped along each stroke's length. In other words, it applies the selected progression along each stroke.

Mapping
Either a linear

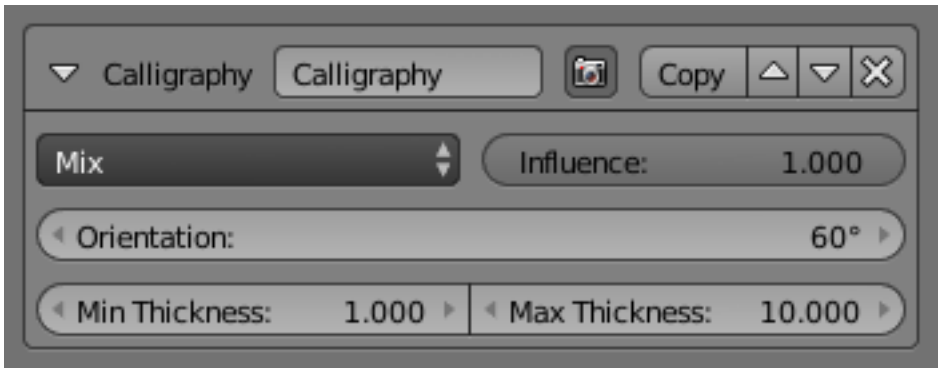


progression (from `0.0` to `1.0` which may be inverted with the *Invert* option), or a custom mapping curve.

Calligraphy

The *Calligraphy* modifier mimics some broad and flat pens for

cal-
lig-
ra-
phy.
It
gen-
er-
ates
dif-
fer-
ent
thick-
ness
based
on
the
orientation of the stroke.



Orientation
The
an-
gle
(ori-
en-
ta-
tion)
of
the
vir-
tual
draw-
ing
tool,
from
the
ver-
ti-
cal
axis
of
the
pic-
ture.
For
ex-

ample, an angle of `0.0` mimics a pen aligned with the vertical axis, hence the thickest strokes will be the vertical ones, and the thinnest, the horizontal ones.

Min Thickness and Max Thickness
The minimum and maximum generated thickness (as explained above, minimum is used when the stroke's direction is perpendicular to the main *Orientation*, and maximum, when aligned with it).

Distance from Camera

The *Distance from Camera* modifier alters

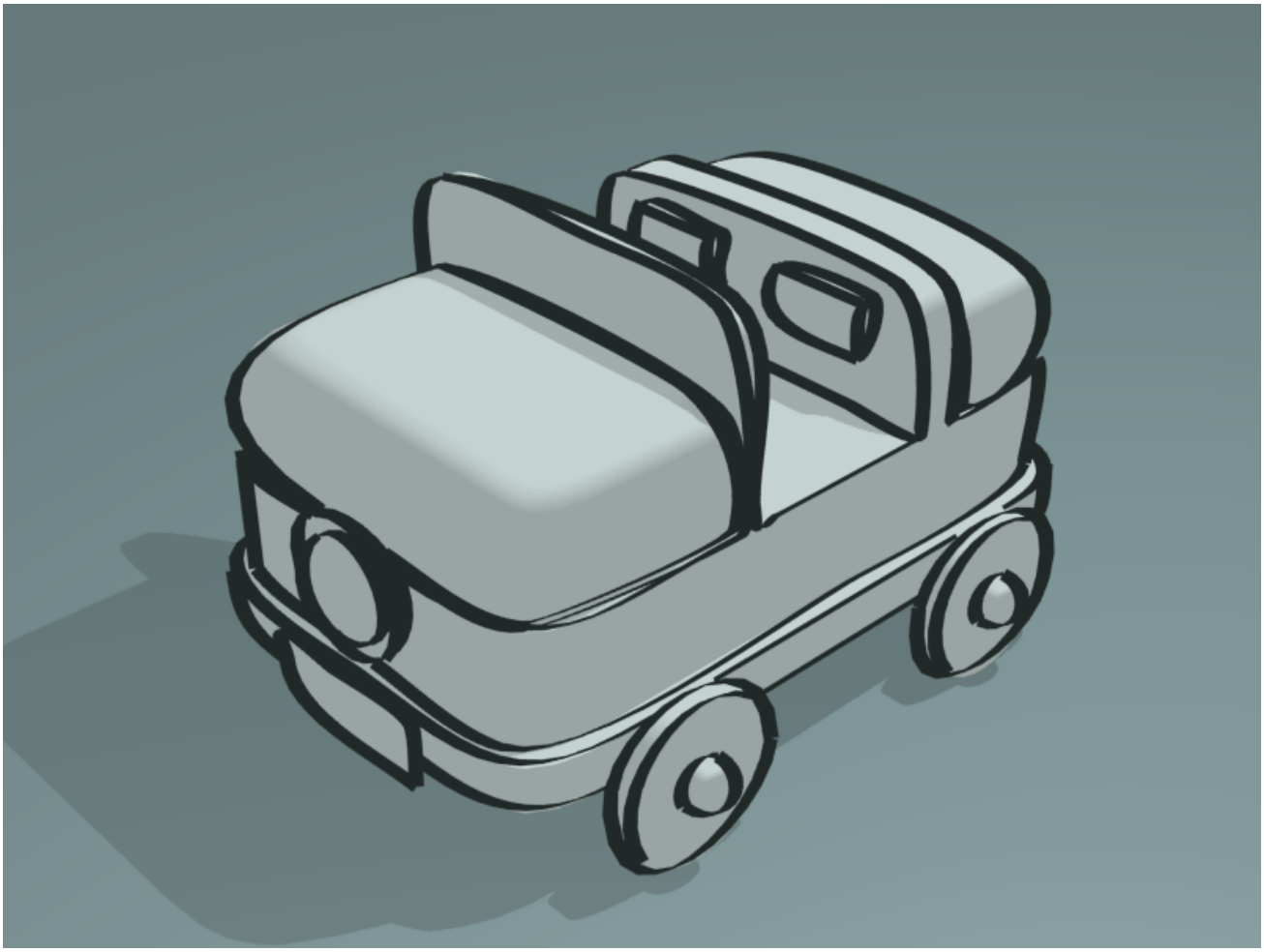
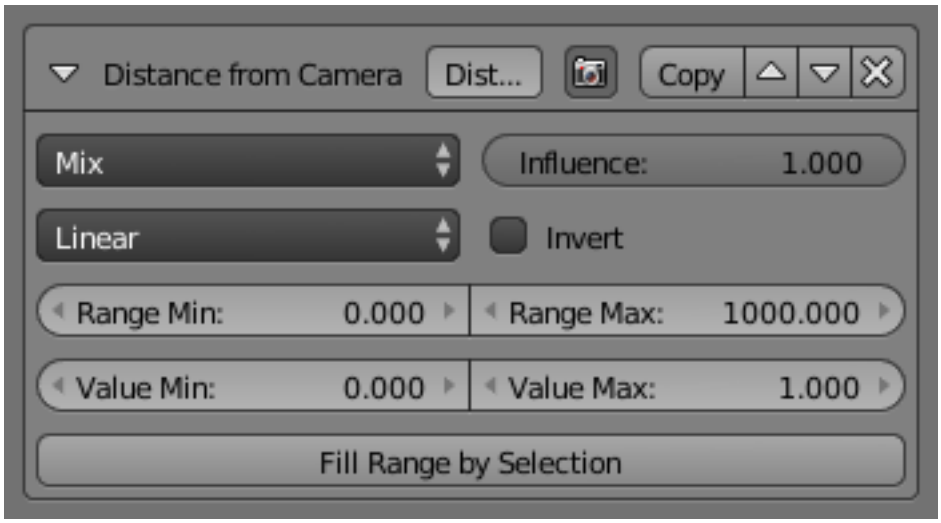


Fig. 2.2118: Calligraphy modifier demo by T.K. File:Toycar_Calligraphy.zip.

the base thickness with a new one from either a linear progression or a custom curve, using the distance to the active camera as the parameter.



Mapping
Either a linear progression (from `0.0` to `1.0` which may be inverted with

the *Invert* option), or a custom mapping curve.

Range Min and Range Max
The limits of the mapping from "distance to camera" to "thickness in mapping". If the current point of the stroke is at *Range Min* or less from the active camera, it will take the start thickness of the mapping, and conversely, if it is at *Range Max* or more from the camera, it will take the end thickness of the mapping. These values are in the current scene's units, not in pixels!

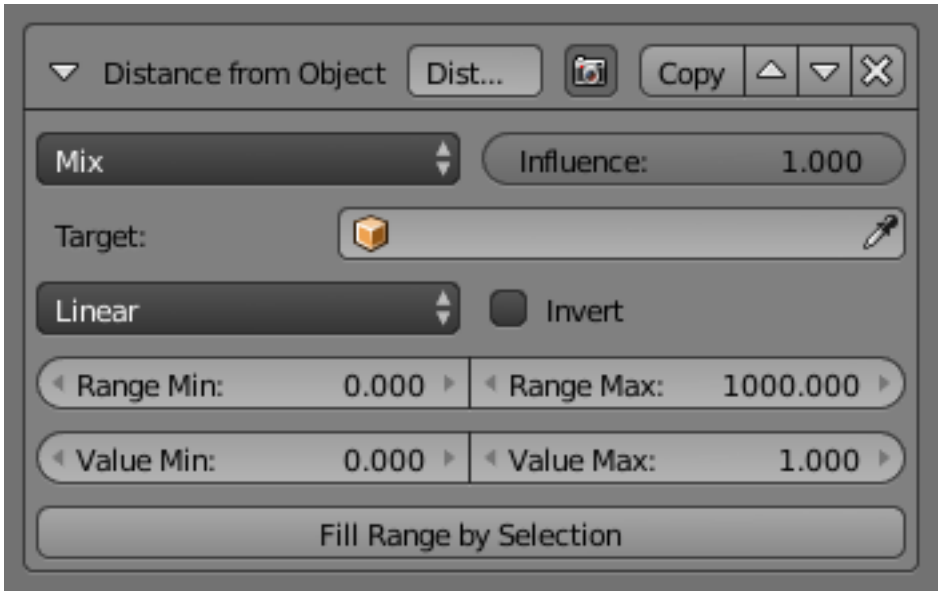
Fill Range by Selection
Set the min/max range values from the distances

between the current selected objects and the camera.

Distance from Object

The *Distance from Object* modifier alters the base thickness with a new one from either a linear progression or a custom curve, using the distance to a given object as parameter.

Target
The object to



measure distance from.

Mapping
Either a linear progression (from `0.0` to `1.0` which may be inverted with the *Invert* option), or a custom mapping curve.

Range Min and Range Max
The

limits of the mapping from "distance to object" to "alpha in mapping". If the current point of the stroke is at *Range Min* or less from the target, it will take the start thickness of the mapping, and conversely, if it is at *Range Max* or more from the target, it will take the end thickness of the mapping. These values are in the current scene's units, not in pixels!

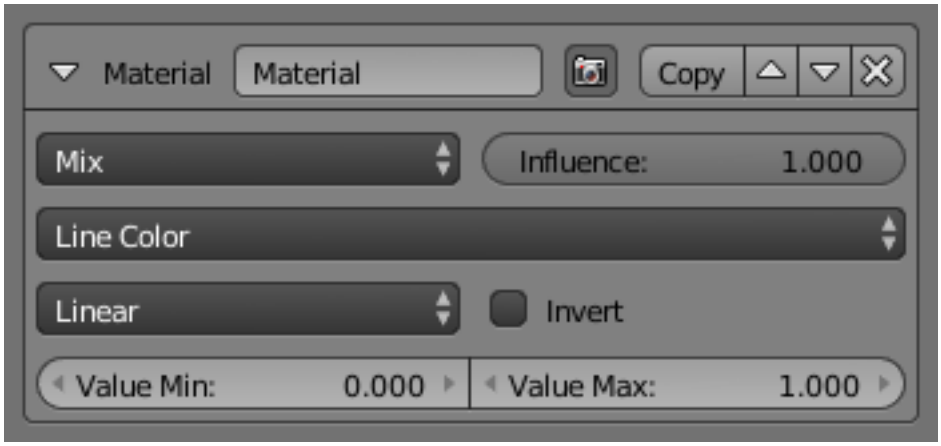
Fill Range by Selection
Set the min/max range values from the distances between the current selected objects and the target.

Material

The *Material* modifier alters the base thickness with a new one taken from the current material under the stroke.

You can use various properties of the materials, among which some are multi-components (i.e. give RGB results).

In that case, the mean value will be used.



Mapping
Either a linear progression (from `0.0` to `1.0` which may be inverted with the *Invert* option), or a custom mapping curve. Note the linear non-inverted option is equivalent to "do nothing", as original values from materials are already in the [0.0, 1.0] range...

If

used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.

Noise

The *Noise* modifier uses a pseudo-random number generator to variably distribute thickness along the

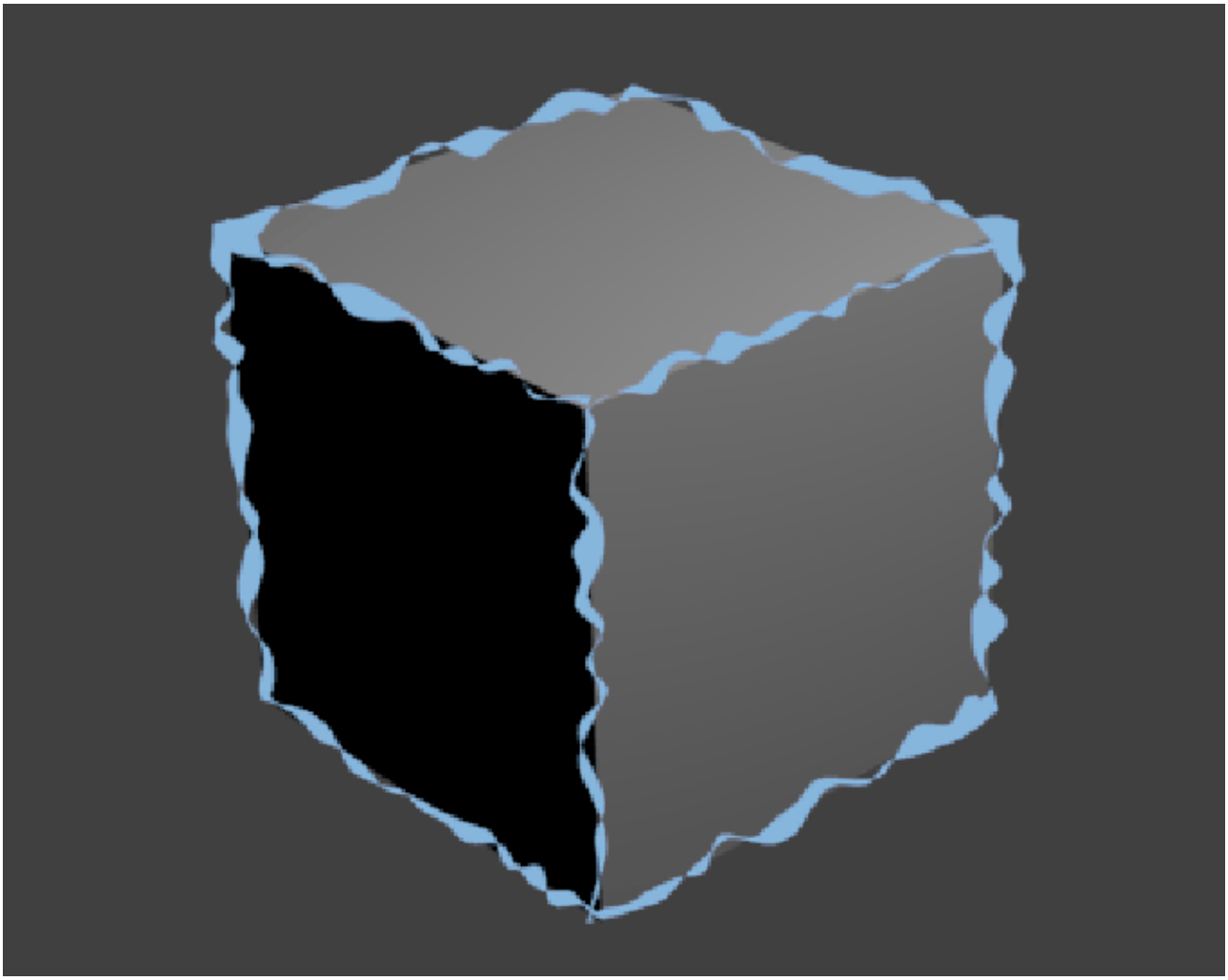
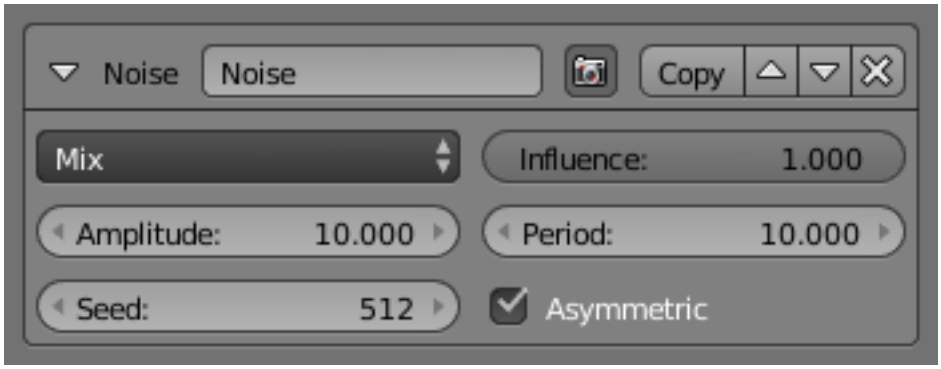


Fig. 2.2119: Effect generated with a noise thickness modifier using asymmetric thickness.

stroke.



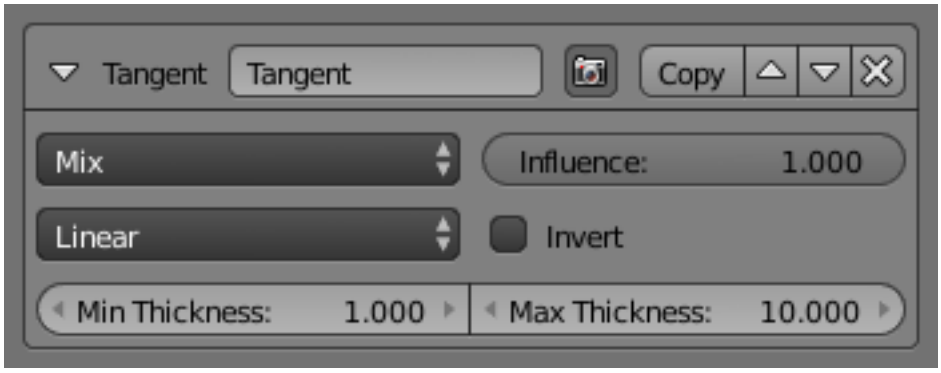
Min Thickness and Max Thickness
The minimum and maximum assigned thickness.

Asymmetric
Allows the thickness to be distributed unevenly at every point. Internally, the stroke is represented as a backbone with

a thickness to the right and left side. All other thickness shaders make sure that the left and right thickness values are equal. For the Noise shader however, a meaningful (and good-looking) result can be created by assigning different values to either side of the backbone.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke's vertices.



Min Thickness and Max Thickness
The minimum and

maximum assigned thickness.

Mapping
Either a linear progression (from *Min Thickness* to *Max Thickness*, which may be inverted with the *Invert* option), or a custom mapping curve (on the same range).

Min Angle and Max Angle
The range of input values to the mapping. Out-of-range input

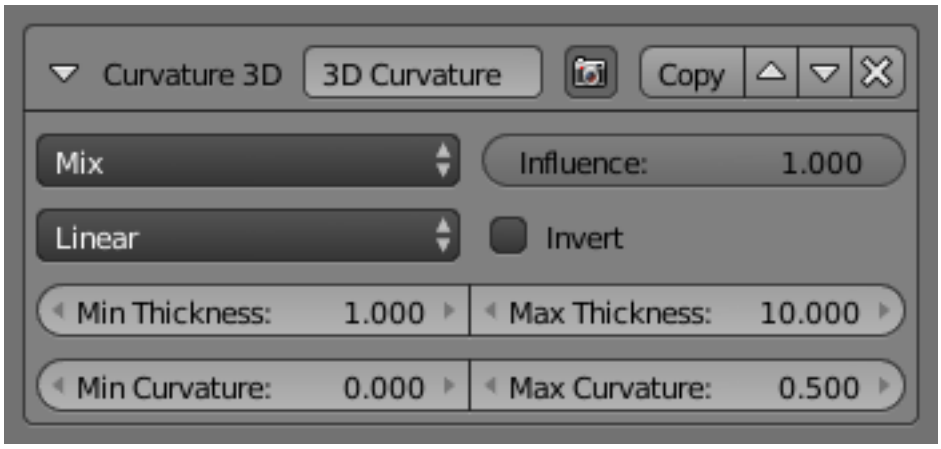
values will be clamped by the Min and Max angles and their corresponding thickness values.

3D Curvature

A modifier based on radial curvatures of the underlying 3D surface. The curvature of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross-section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of

the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.



Min Thickness and Max Thickness

The minimum and maxi-

mum assigned thickness.

Mapping
Either a linear progression (from *Min Thickness* to *Max Thickness*, which may be inverted with the *Invert* option), or a custom mapping curve (on the same range).

Min Curvature and Max Curvature
The limits of the mapping of the Min and Max Thickness. If the current

point of the stroke is at *Min Curvature* or less from the target, it will take the start thickness of the mapping, and conversely, if it is at *Max Curvature* or more from the target, it will take the end thickness of the mapping.

Crease Angle

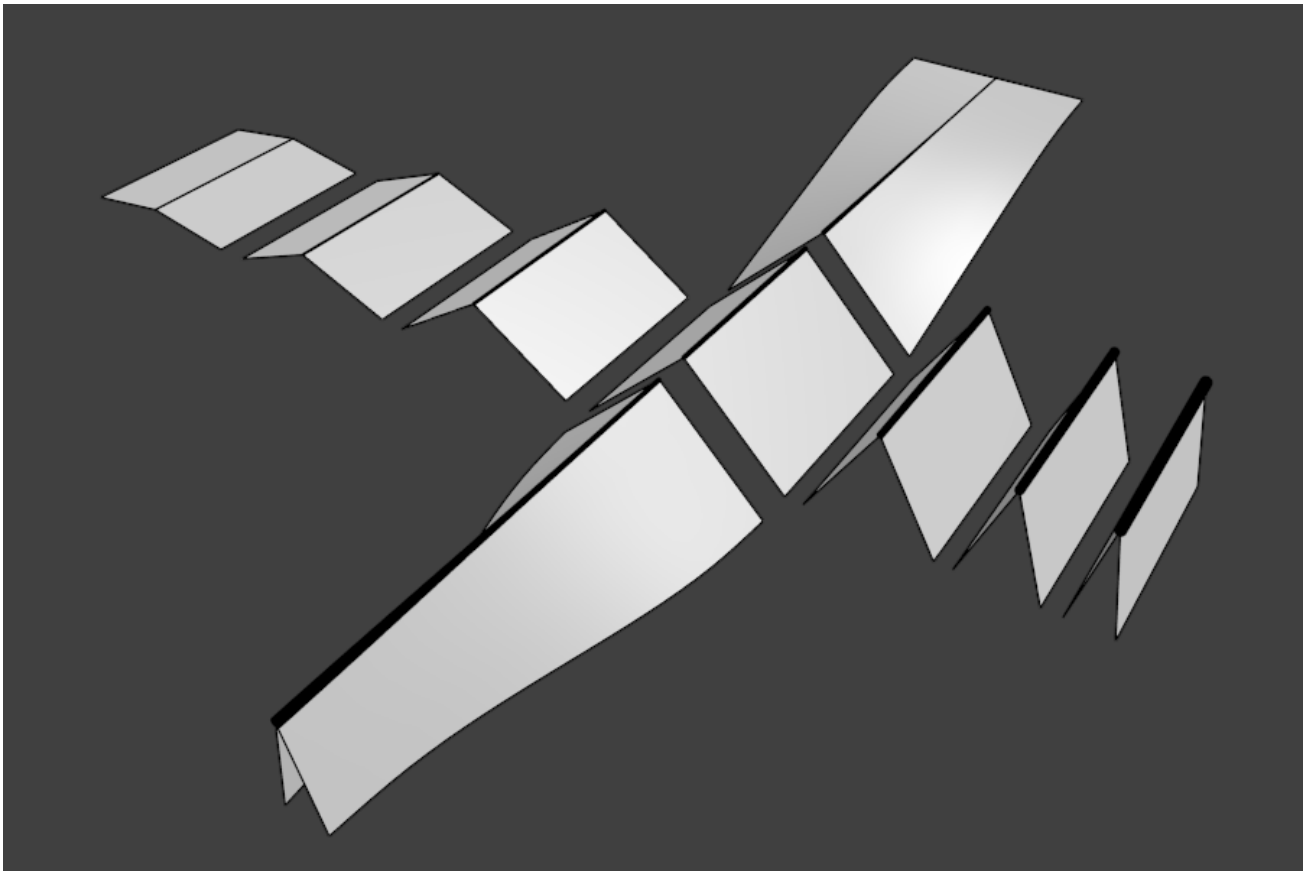
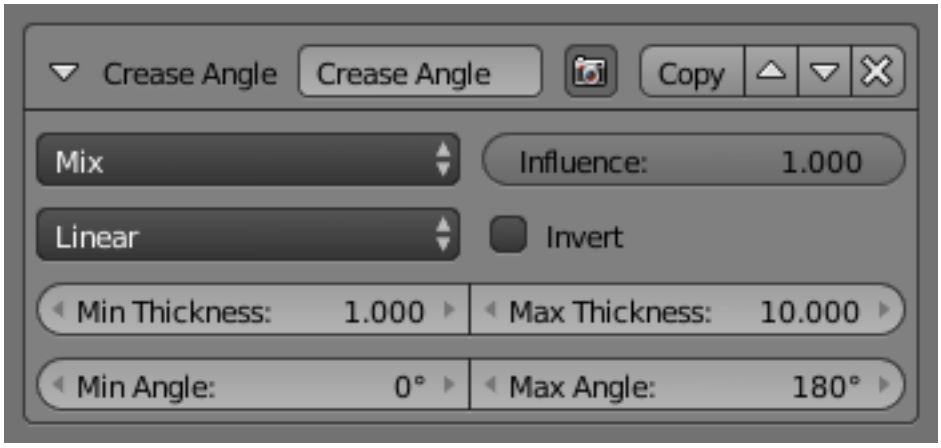


Fig. 2.2120: Crease Angle modifier demo by T.K. File:Render_freestyle_modifier_crease_angle.blend.

A modifier based on the

Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e., the edge does not have the *Crease Angle nature*, its thickness value is not touched by this modifier.



Min Thickness and Max Thickness
The minimum and maximum assigned thickness.

Mapping
Either a

linear progression (from *Min Thickness* to *Max Thickness*, which may be inverted with the *Invert* option), or a custom mapping curve (on the same range).

Geometry

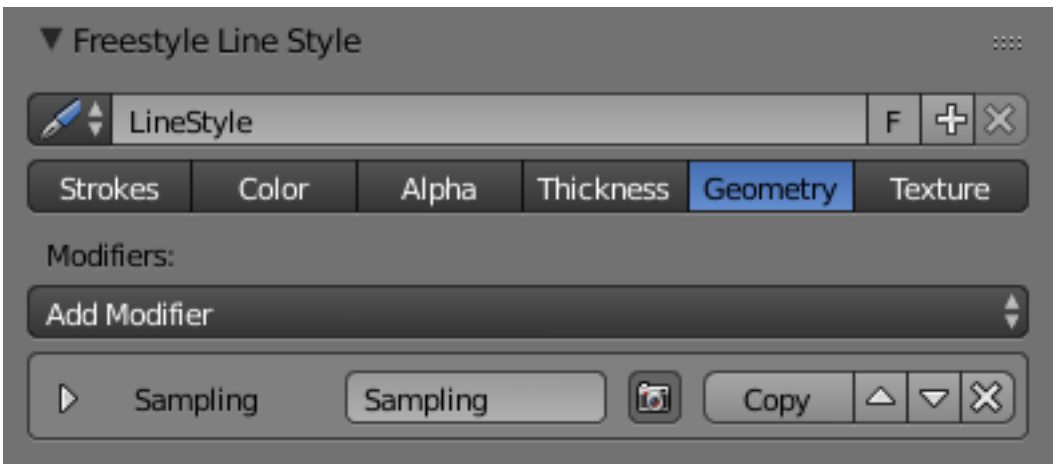


Fig. 2.2121: Line Style Geometry Overall UI.

In this tab you control the

geometry of your strokes.

Modifiers

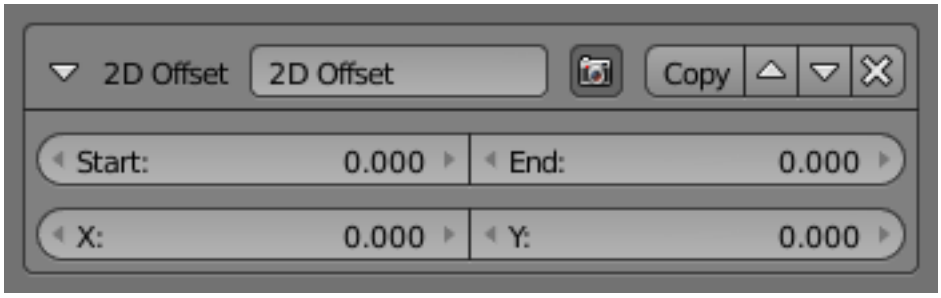
There are thirteen geometry modifiers available. These modifiers have no mix nor influence settings, as they always completely apply to the strokes' geometry (like object modifiers do). They take the resulting two-dimensional strokes from the Freestyle line set and displace or deform them in various ways.

As with other modifier stacks in Blender, they are applied

from top to bottom.

2D Offset

The *2D Offset* modifier adds some two-dimensional offsets to the stroke backbone geometry. It has two sets of independent options/effects:



Start and End
These two

options add the given amount of offset to the start (or end) point of the stroke, along the (2D) normal at those points. The effect is blended over the whole stroke, if you for example, set only *Start* to 50, the start of the stroke is offset 50 pixels along its normal, the middle of the stroke, 25 pixels along its own normal, and the end point is not moved.

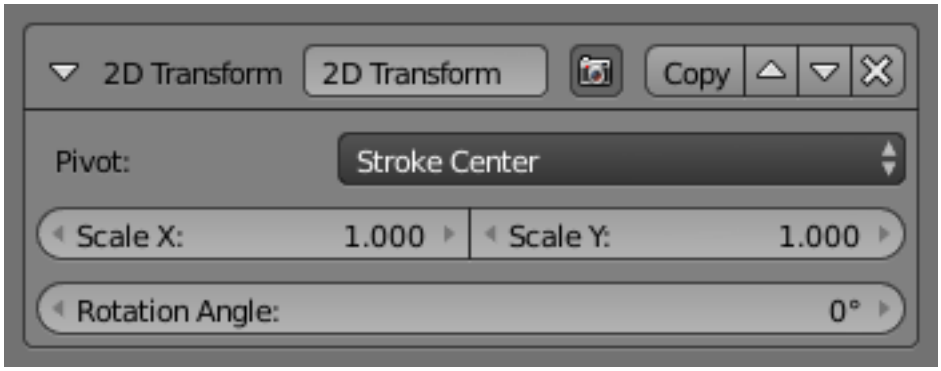
X and Y
These two options simply add a constant horizontal and/or vertical offset to the whole stroke.

2D Transform

The *2D Transform* modifier applies two-dimensional scaling and/or rotation to the stroke backbone geometry. Scale is applied before rotation.

The center (pivot point) of these 2D transformations can be:

Stroke Center
The median point



of the stroke.

Stroke Start
The beginning point of the stroke.

Stroke End
The end point of the stroke.

Stroke Point Parameter
The *Stroke Point Parameter* factor controls where along the stroke the pivot point is (start point if set

to 0.0; end point if set to 1.0).

Absolute 2D Point
The *Pivot X* and *Pivot Y* allows you to define the position of the pivot point in the final render (from the bottom left corner).

> **Warning:** Currently, you have to take into account the *real* render size, i.e. resolution **and** resolution percentage.

Scale X and Scale Y
The scaling factors, in their respective axes.

Rotation Angle
The rotation angle.

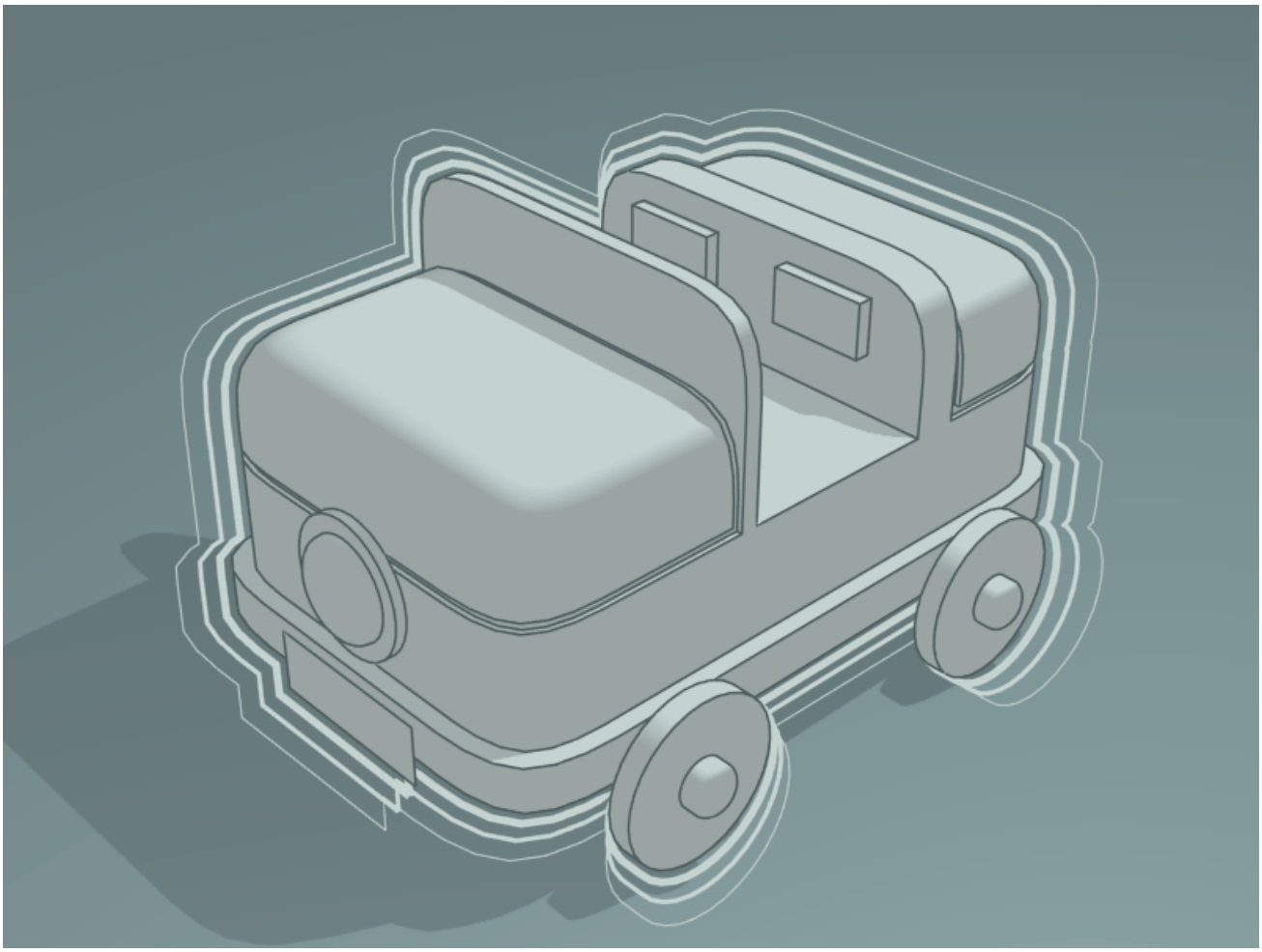
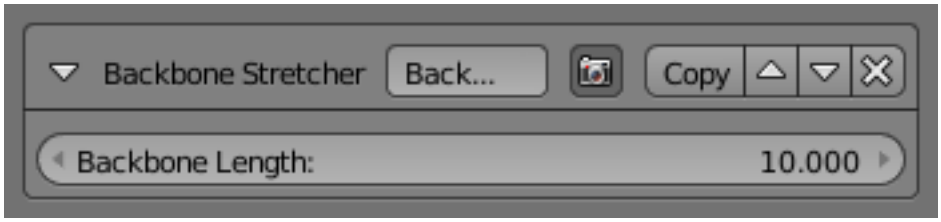


Fig. 2.2122: 2D Transform modifier File:Toycar_Three_Contours.zip.

Backbone Stretcher

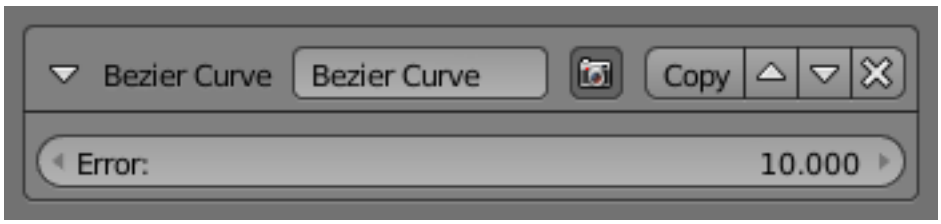


The *Backbone Stretcher*

modifier stretches (adds some length to) the beginning and end of the stroke.

Backbone Length
Length to add to the strokes' ends.

Bézier Curve



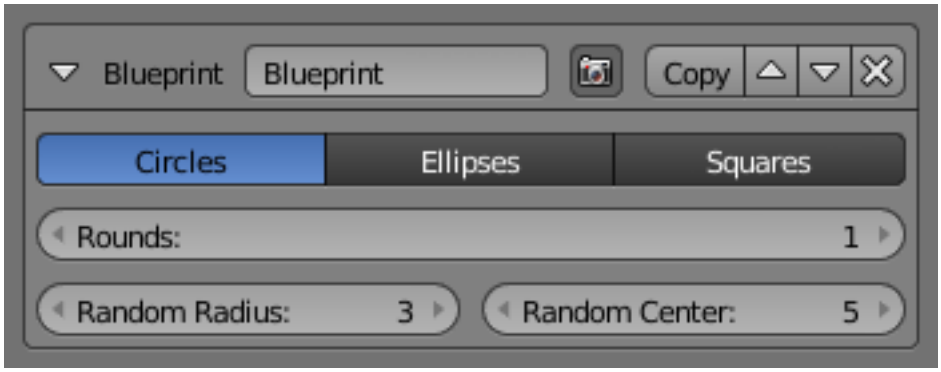
The *Bézier Curve* modifier replaces the stroke by a Bézier approximation of it.

Error
: The maximum distance allowed between the new Bézier curve and the original stroke.

Fig. 2.2123: Bézier Curve modifier demo by T.K. File:toycar_bezier.zip.

Blueprint

The *Blueprint* modifier produces blueprint-like strokes using either circular, elliptical, or square contours. A blueprint here refers to those lines drawn at the beginning of free-hand drawing to capture the silhouette of objects with a simple shape such as circles, ellipses and squares.



Shape
Which base shapes to use for this

blueprint: *Circles*, *Ellipses* or *Squares*.

Rounds
How many rounds are generated, as if the pen draws the same stroke several times (i.e. how many times the process is repeated).

Random Radius and Random Center
For the *Circles* and *Ellipses* shapes. Adds some randomness to each round in the rele-

vant aspect. Using more than one round with no randomness would be meaningless, as they would draw over each other exactly.

Backbone Length and Random Backbone

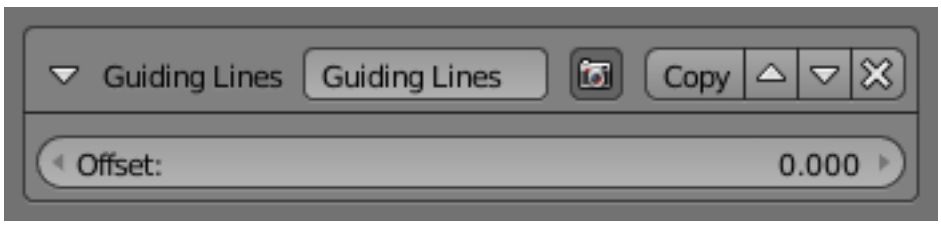
For the *Squares* shapes. The first adds some extra length to each edge of the generated squares (also affected by the second parameter). The second adds some randomness to the squares.

Note that the *Min 2D Length* feature from the *Strokes* set-

tings is quite handy here, to avoid the noise generated by small strokes...

Guiding Lines

The *Guiding Lines* modifier replaces a stroke by a straight line connecting both of its ends.



Offset
Offset the start and end

points along the original stroke, before generating the new straight one.

This modifier will produce reasonable results when strokes are short enough, because shorter strokes are more likely to be well approximated by straight lines. Therefore, it is recommended to use this modifier together with one of the splitting options (by 2D angle or by 2D length) from the *Strokes* panel.

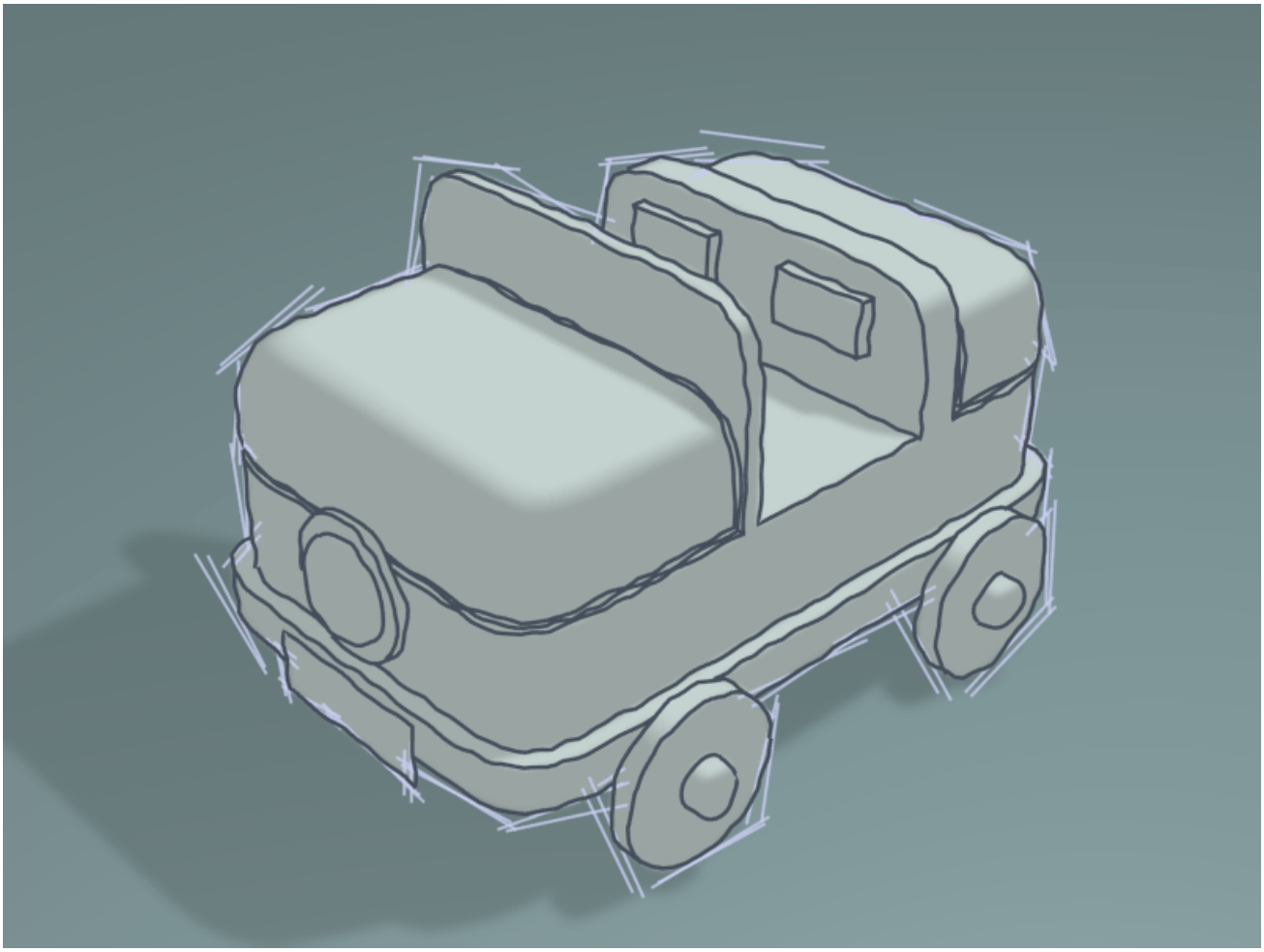


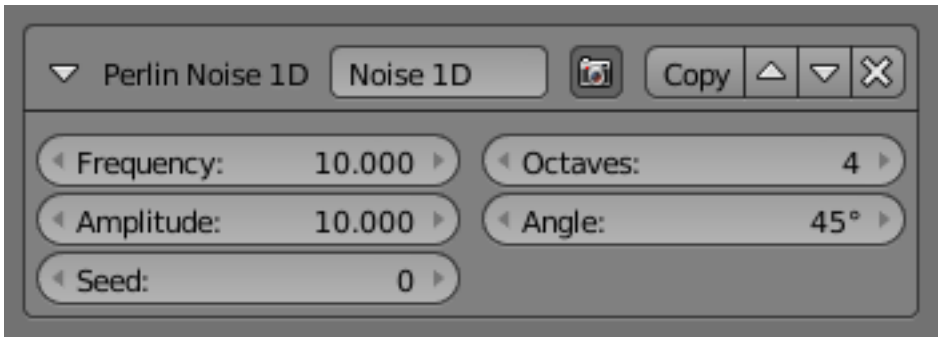
Fig. 2.2124: Guiding Lines modifier Demo by T.K. File:Toycar_Guiding_Line.zip.

Perlin Noise 1D

The *Perlin Noise 1D* modifier adds one-dimensional Perlin noise to the stroke. The curvilinear abscissa (value between 0 and 1 determined by a point's position relative to the first and last point of a stroke) is used as the input to the noise function to generate noisy displacements.

This means that this modifier will give an identical result for two strokes with

the same length and sampling interval.



Frequency
How dense the noise is (kind of a scale factor along the stroke).

Amplitude
How much the noise distorts the stroke in the *Angle* direction.

Seed
The seed

of the random generator (the same seed over a stroke will always give the same result).

Octaves
The "level of detail" of the noise.

Angle
In which direction the noise is applied (0.0 is fully horizontal).

Perlin Noise 2D

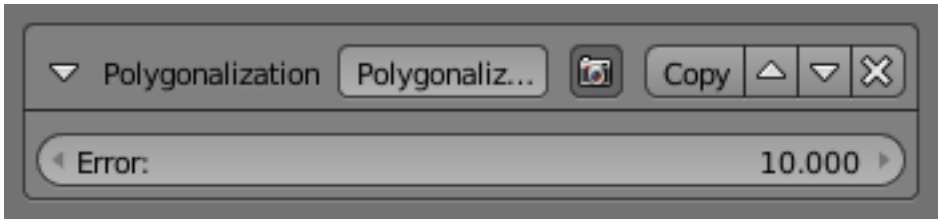
The *Per-*

lin Noise 2D modifier adds one-dimensional Perlin noise to the stroke. The modifier generates noisy displacements using 2D coordinates of stroke vertices as the input of the noise generator.

Its settings are exactly the same as the *Perlin Noise 1D* mod-

i-
fier.

Polygonization



The *Polygonization* modifier simplifies strokes as much as possible (in other words, it transforms smooth strokes into jagged polylines).

Error
: The maximum distance allowed between the new

simplified stroke and the original one (the larger this value is, the more jagged/approximated the resulting polylines are).

Sampling

The *Sampling* modifier changes the definition, precision of the stroke, for the following modifiers.

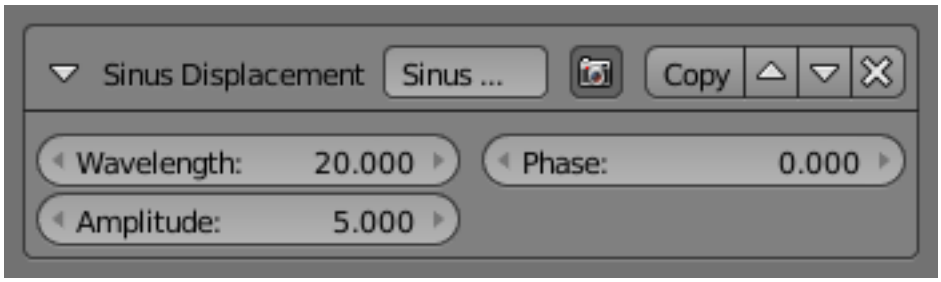
Sampling
The smaller this value, the more

precise are the strokes. Be careful; too small values will require a huge amount of time and memory during render!

Sinus Displacement

The *Sinus Displacement* modifier adds a sinusoidal displacement

to the stroke.



Wavelength
How wide the undulations are along the stroke.

Amplitude
How high the undulations are across the stroke.

Phase
Allows "offsetting" ("moving") the undulations along the stroke.

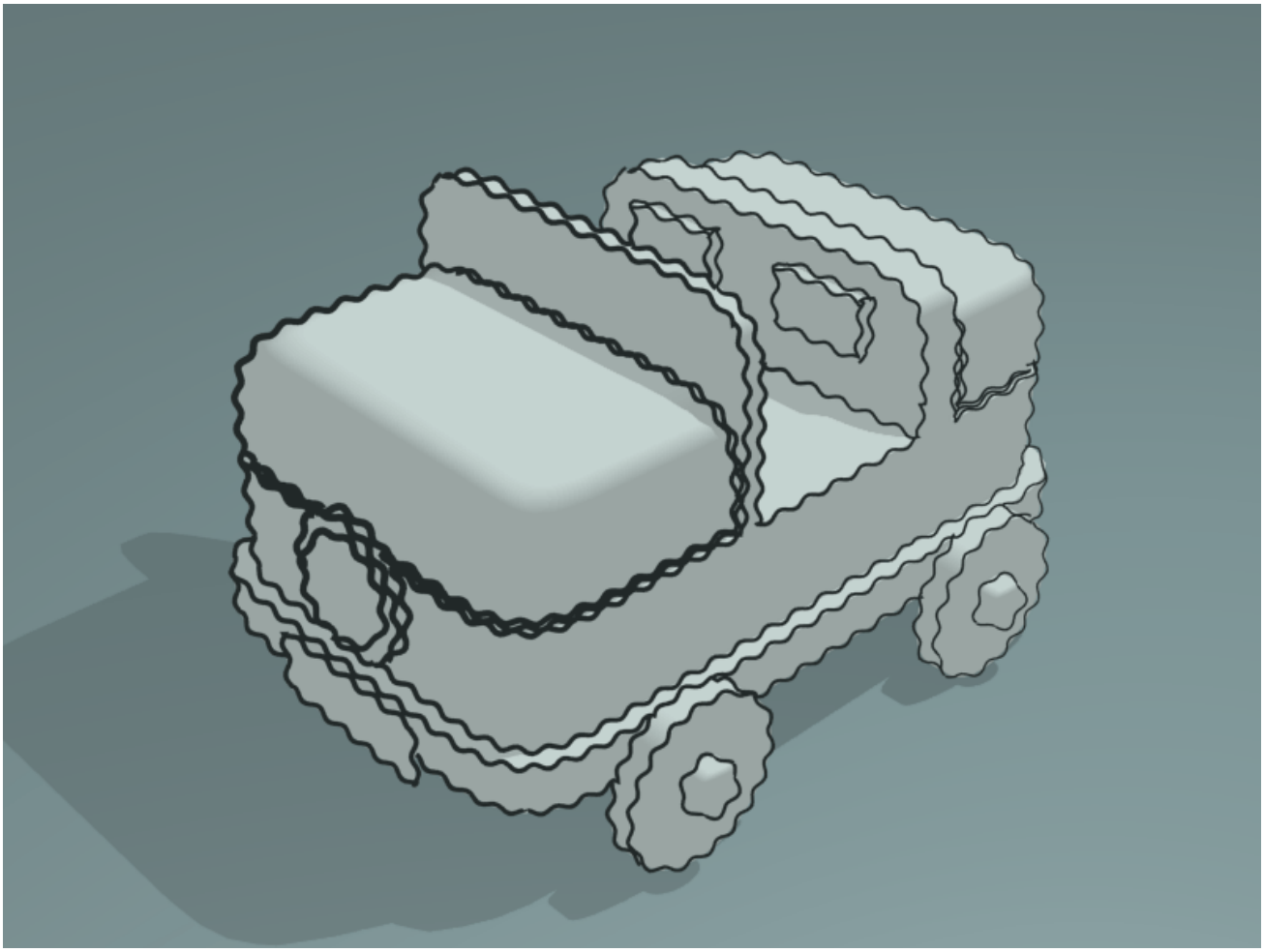
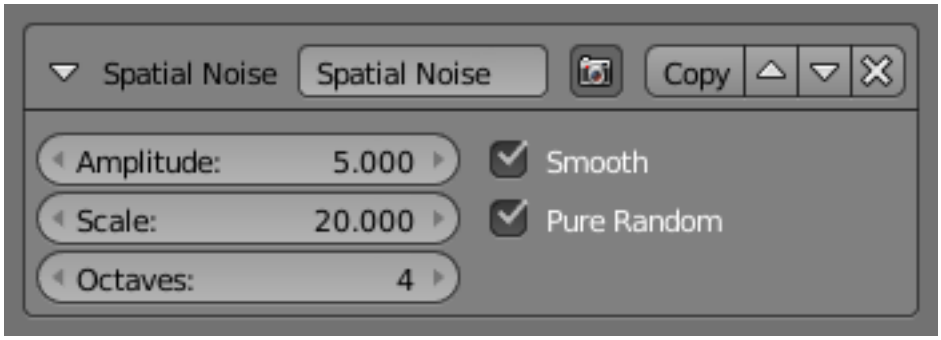


Fig. 2.2125: Sinus Displacement modifier demo by T.K. File:Toycar_Sinus.zip.

Spatial Noise

The *Spatial Noise* modifier adds some spatial noise to the stroke. Spatial noise displacements are added in the normal direction (i.e., the direction perpendicular to the tangent line) evaluated at each stroke vertex.



Amplitude
How much the noise distorts the stroke.

Scale
How wide the noise is along the stroke.

Octaves
The level of detail of the noise.

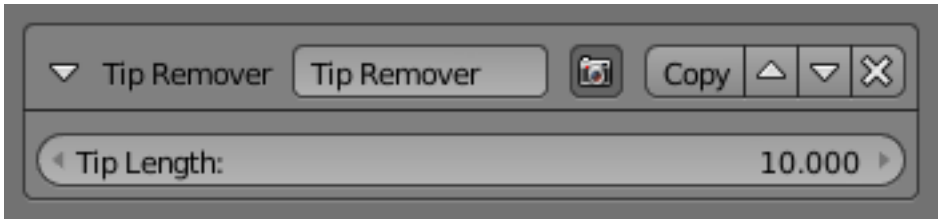
Smooth
When enabled, apply some smoothing over the generated noise.

Pure Random
When disabled, the next generated random value depends on the previous one; otherwise they

are completely independent. Disabling this setting gives a more "consistent" noise along a stroke.

Tip Remover

The *Tip Remover* modifier removes a piece of the stroke at its beginning and end.



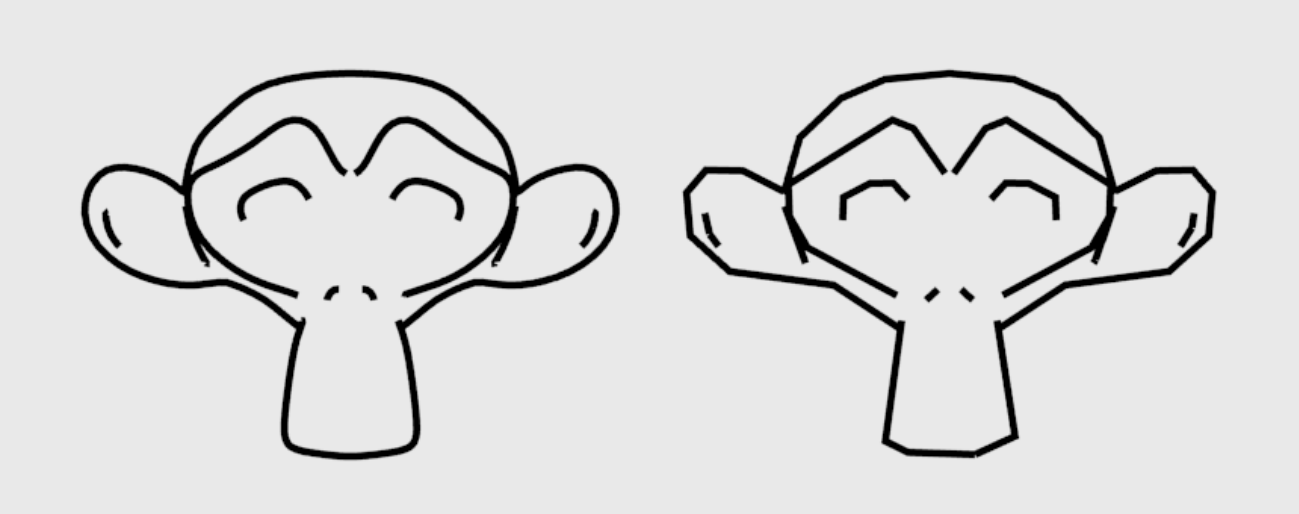
Tip Length
Length of stroke to remove at both of its tips.

Simplification

The *Simplification* modifier merges stroke vertices that lie close to one another, like the *Decimate* modifier for meshes.

Tolerance
Measure for how close points have to be to each other to be merged. A

higher tolerance means more vertices are merged.



Texture

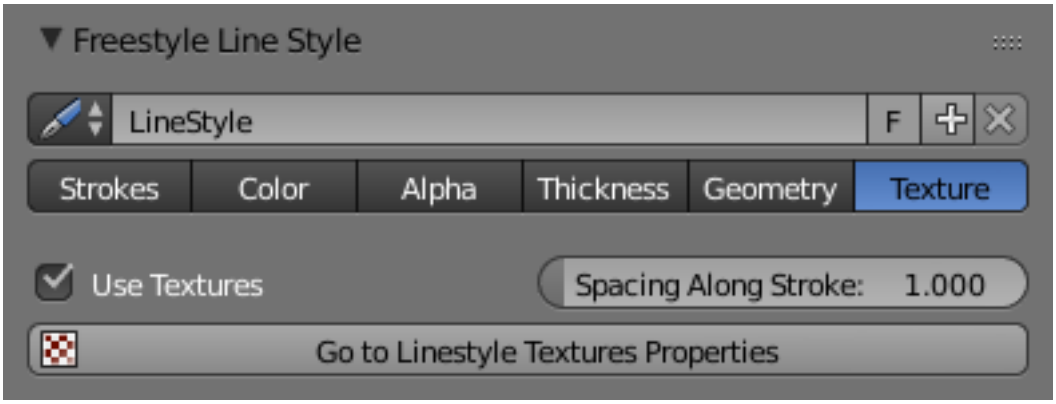


Fig. 2.2126: Line Style Texture.

TODO.
See wiki.

Python Scripting Mode

The Python

Scripting mode offers full programmability for line stylization. In this control mode, all stylization operations are written as Python scripts referred to as style modules in the Freestyle terminology. The input to a style module is a view map (i.e., a set of detected feature edges), and the output is a set of stylized strokes.

A style module is composed of successive calls of five basic operators: selection, chaining, split-

ting, sorting and stroke creation. The selection operator identifies a subset of input feature edges based on one or more user-defined selection conditions (predicates). The selected edges are processed with the chaining, splitting and sorting operators to build chains of feature edges. These operators are also controlled by user-supplied predicates and functions in order to determine how to transform the feature edges into chains. Finally, the chains are transformed into stylized strokes by the stroke creation operator, which takes a list of user-defined stroke shaders.

Python style modules are stored within blend-files as text data-blocks. External style module files first need to be loaded in the Text Editor. Then the select menu within an entry of the style module stack allows you to select a module from the list of loaded style modules.

Freestyle for Blender comes with a number of Python style mod-

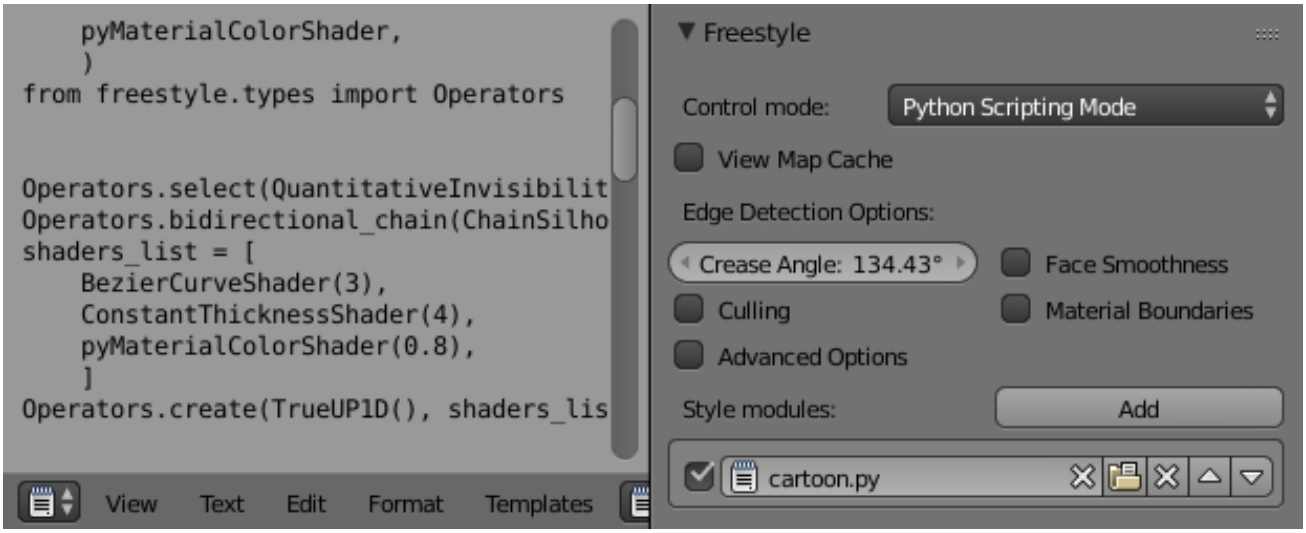


Fig. 2.2127: A screen capture of a style module (cartoon.py) loaded in the Text Editor (left), as well as Freestyle options in the Python Scripting mode in the Render Layers buttons (right).

ules that can serve as a starting point of your own style module writing. See also the section of the Freestyle Python API in the Blender Python API reference manual for the full detail of style module constructs.

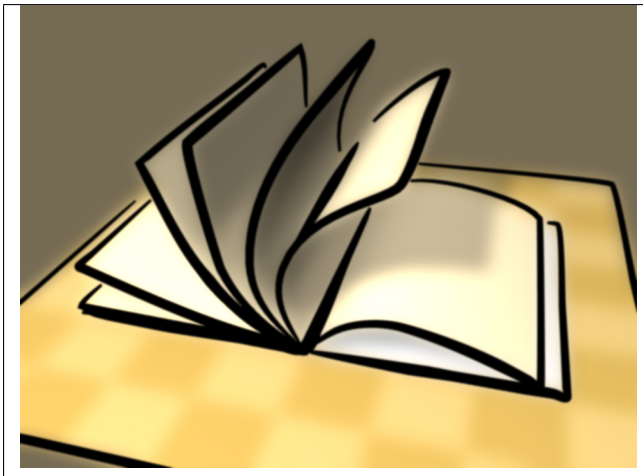


Fig. 2.2128: By T.K. using the Python Scripting mode. (File:Turning_Pages.zip, CC0)

Fig. 2.2129: By T.K. using the Python Scripting mode. (File:Lily_Broken_Topology.zip, CC0)

Writing Style Modules

A style module is a piece of code responsible for the stylization of Freestyle line drawing. The input of a style module is a set of feature edges called view map (ViewMap). The output is a set of stylized lines also referred to as strokes. A style module is structured as a pipeline of operations that allow for building strokes from the input edges within the view map.

There are five kinds of operations (listed with corresponding operator functions):

- Selection `Operators.select()`
- Chaining `Operators.chain()`,`Operators.bidirectional_chain()`
- Splitting `Operators.sequential_split()`,`Operators.recursive_split()`
- Sorting `Operators.sort()`
- Stroke creation `Operators.create()`

The input view map is populated with a set of ViewEdge objects. The

selection operation is used to pick up ViewEdges of interest to artists based on user-defined selection conditions (predicates). Chaining operations take the subset of ViewEdges and build Chains by concatenating ViewEdges according to user-defined predicates and functions. The Chains can be further refined by splitting them into smaller pieces (e.g., at points where edges make an acute turn) and selecting a fraction of them (e.g., to keep only those longer than a length threshold). The sorting operation is used to arrange the stacking order of chains to draw one line on top of another. The chains are finally transformed into stylized strokes by the stroke creation operation applying a series of stroke shaders to individual chains.

ViewEdges, Chains and Strokes are generically referred to as one-dimensional (1D) elements. A 1D element is a polyline that is a series of connected straight lines. Vertices of 1D elements are called 0D elements in general.

All the op-

erators act on a set of active 1D elements. The initial active set is the set of ViewEdges in the input view map. The active set is updated by the operators.

Selection

The selection operator goes through every element of the active set and keeps only the

ones satisfying a certain predicate. The `Operators.select()` method takes as the argument a unary predicate that works on any `Interface1D` that represents a 1D element. For example:

```
Operators.
↪select(QuantitativeInvisibilityUP1D(0))
```

This selection operation uses the `QuantitativeInvisibilityUP1D` predicate to select only the visible `ViewEdge` (more precisely, those whose quantitative invisibility is equal to 0). The selection operator is intended to selectively apply the style to a fraction of the active 1D elements.

It is noted that `QuantitativeInvisibilityUP1D` is a class imple-

menting the predicate that tests line visibility, and the `Operators.select()` method takes an instance of the predicate class as argument. The testing of the predicate for a given 1D element is actually done by calling the predicate instance, that is, by invoking the `__call__` method of the predicate class. In other words, the `Operators.select()` method takes as argument a functor which in turn takes an `Interface0D` object as argument. The Freestyle Python API employs functors extensively to implement predicates, as well as functions.

Chaining

The chaining operators act on the set of active `ViewEdge` objects and determine the topology of the future

strokes.
The
idea
is
to implement an iterator to traverse the ViewMap graph by marching along ViewEdges. The iterator defines a chaining rule that determines the next `ViewEdge` to follow at a given vertex (see `ViewEdgeIterator`). Several such iterators are provided as part of the Freestyle Python API (see `ChainPredicateIterator` and `ChainSilhouetteIterator`). Custom iterators can be defined by inheriting the `ViewEdgeIterator` class. The chaining operator also takes as argument a UnaryPredicate working on `Interface1D` as a stopping criterion. The chaining stops when the iterator has reached a `ViewEdge` satisfying this predicate during the march along the graph.

Chaining
can
be
either
unidirectional
`Operators.chain()`
or
bidirectional
`Operators.bidirectional_chain()`.
In
the
latter
case,
the
chaining
will
propagate
in
the
two
directions from the starting edge.

The
following
is
a
code
example
of
bidirectional

chain-
ing:

```
Operators.
 ↪bidirectional_
 ↪chain(
␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪ChainSilhouetteIterator(),
 ↪
␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪NotUP1D(QuantitativeInvisibilityUP1D(0)),
 ↪
␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪␣
 ↪)
```

The
chain-
ing
op-
er-
a-
tor
uses
the
`ChainSilhouetteIterator`
as
the
chain-
ing
rule
and
stops
chain-
ing
as
soon
as
the
it-
er-

a-
tor
has
come
to
an
in-
visible `ViewEdge`.

The
chain-
ing
op-
er-
a-
tors
pro-
cess
the
set
of
ac-
tive
`ViewEdge`
ob-
jects
in
or-
der.
The
ac-
tive
ViewEdges
can
be
pre-
vi-
ously
sorted
us-
ing
the `Operators.sort()` method (see below). It starts a chain with the first `ViewEdge` of the active set. All ViewEdges that have already been involved in the chaining process are marked (in the case of the example above, the time stamp of each `ViewEdge` is modified by default), in order not to process the same `ViewEdge` twice. Once the chaining reaches a `ViewEdge` that satisfies the stopping predicate, the chain is terminated. Then a new chain is started from the first unmarked `ViewEdge` in the active set. This operation is repeated until the last unmarked `ViewEdge` of the active set was processed. At the end of the chaining operation, the active set is set to the Chains that have just been constructed.

Splitting

The
split-
ting
op-
er-
a-
tion
is

used to refine the topology of each Chain. Splitting is performed either sequentially or recursively.
Sequential splitting `Operators.sequentialSplit()` in its basic form, parses the Chain at a given arbitrary resolution and evaluates a unary predicate (working on 0D elements) at each point along the Chain. Every time the predicate is satisfied, the chain is split into two chains. At the end of the sequential split operation, the active set of chains is set to the new chains.

```
Operators.
↪sequentialSplit(TrueUP0D(),
↪␣
↪2)
```

In this example, the chain is split every 2 units. A more elaborated version uses two predicates

instead of one: One to determine the starting point of the new chain and the other to determine its ending point. This second version can lead to a set of Chains that are disjoint or that overlap if the two predicates are different. (see `Operators.sequentialSplit()` for more details).

Recursive splitting `Operators.recursiveSplit()` evaluates a function on the 0D elements along the Chain at a given resolution and find the point that gives the maximum value for the function. The Chain is then split into two at that point. This process is recursively repeated on each of the two new Chains, until the input Chain satisfies a user-specified stopping condition.

```
func␣
↪=␣
↪Curvature2DAngleF0D()
Operators.
↪recursive_
↪split(func,
↪␣
↪NotUP1D(HigherLengthUP1D(5)),
↪␣
↪5)
```

In the code

example above, the Chains are recursively split at points of the highest 2D curvature. The curvature is evaluated at points along the Chain at a resolution of 5 units. Chains shorter than 5 units will not be split anymore.

Sorting

The sorting operator `Operators.sort()` arranges the stacking order of active 1D elements. It takes

as
ar-
gu-
ment
a
bi-
nary
pred-
icate used as a “smaller than” operator to order two 1D elements.

```
Operators.
↪sort(Length2DBP1D())
```

In
this
code
ex-
am-
ple,
the
sort-
ing
uses
the
`Length2DBP1D`
bi-
nary
pred-
i-
cate
to
sort
the
`Interface1D`
ob-
jects
in
the
as-
cend-
ing
or-
der
in
terms
of 2D length.

The
sort-
ing
is
par-
tic-
u-
larly
use-
ful
when
com-

bined with causal density. Indeed, the causal density evaluates the density of the resulting image as it is modified. If we wish to use such a tool to decide to remove strokes whenever the local density is too high, it is important to control the order in which the strokes are drawn. In this case, we would use the sorting operator to insure that the most "important" lines are drawn first.

Stroke creation

Finally, the stroke creation operator `Operators.create()` takes the active set of Chains as input and build Strokes. The operator takes

two arguments. The first is a unary predicate that works on `Interface1D` that is designed to make a last selection on the set of chains. A Chain that does not satisfy the condition will not lead to a Stroke. The second input is a list of shaders that will be responsible for the shading of each built stroke.

```
shaders_
↪list␣
↪=␣
↪[
␣
↪␣
↪␣
↪␣
↪SamplingShader(5.
↪0),
↪
␣
↪␣
↪␣
↪␣
↪ConstantThicknessShader(2),
↪
␣
↪␣
↪␣
↪␣
↪ConstantColorShader(0.
↪2,
↪0.
↪2,
↪0.
↪2,
↪1),
↪
␣
↪␣
↪␣
↪␣
↪]
Operators.
↪create(DensityUP1D(8,
↪0.
↪1,
↪␣
↪IntegrationType.
↪MEAN),
↪␣
↪shaders_
↪list)
```

In this example, the `DensityUP1D` predicate

is used to remove all Chains whose mean density is higher than 0.1. Each chain is transformed into a stroke by resampling it so as to have a point every 5 units and assigning to it a constant thickness of 2 units and a dark gray constant color.

User control on the pipeline definition

Style module writing offers different types of user control, even though individual

style modules have a fixed pipeline structure. One is the sequencing of different pipeline control structures, and another is through the definition of functor objects that are passed as argument all along the pipeline.

Different pipeline control structures can be defined by sequencing the selection, chaining, splitting, and sorting operations. The stroke creation is always the last operation that concludes a style module.

Predicates, functions, chaining iterators, and stroke

shaders can be defined by inheriting base classes and overriding appropriate methods. See the reference manual entries of the following base classes for more information on the user-scriptable constructs.

- UnaryPredicate0D
- UnaryPredicate1D
- BinaryPredicate0D
- BinaryPredicate1D
- UnaryFunction0DDouble
- UnaryFunction0DEdgeNature
- UnaryFunction0DFloat
- UnaryFunction0DId
- UnaryFunction0DMaterial
- UnaryFunction0DUnsigned
- UnaryFunction0DVec2f
- UnaryFunction0DVec3f
- UnaryFunction0DVectorViewShape
- UnaryFunction0DViewShape
- UnaryFunction1DDouble
- UnaryFunction1DEdgeNature
- UnaryFunction1DFloat
- UnaryFunction1DUnsigned

- UnaryFunction1DVec2f
- UnaryFunction1DVec3f
- UnaryFunction1DVectorViewShape
- UnaryFunction1DVoid
- ViewEdgeIterator
- StrokeShader

Freestyle SVG Exporter

SVG exporting for Freestyle is available through an add-on.

Fig. 2.2130: An example of a .svg result produced by the Freestyle SVG Exporter.

This add-on can be enabled via *User Preferences* → *Add-ons* → *Render* → *Freestyle SVG*

Exporter. The GUI for the exporter should now be visible in the render tab of the Properties editor. The exported .svg file is written to the default output path *Properties editor* → *Render* → *Output*.

Options

Mode
Option between Frame and Animation. Frame will render a single frame, Animation will bundle all rendered frames into a single `.svg` file.

Split at Invisible
By default the exporter will

not take invisible vertices into account and export them like they are visible. Some stroke modifiers, like Blueprint, mark vertices as invisible to achieve a certain effect. Enabling this option will make the paths split when encountering an invisible vertex, which leads to a better result.

Fill Contours
The contour of objects is filled with their material color.

Note:
This features is somewhat unstable – especially

with animations.

Stroke Cap Style
Defines the style the stroke caps will have in the SVG output.

Exportable Properties

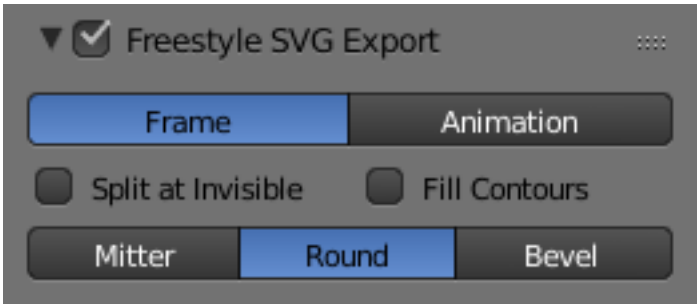


Fig. 2.2131: Freestyle SVG Export panel.

Because the representation of Freestyle strokes and SVG path objects is fundamentally different, an one-on-one translation between Freestyle and SVG is not possible. The main shortcoming of SVG compared to Freestyle is that Freestyle defines style per-point, where SVG defines it per-path. This means that Freestyle can produce much more complex results that are impossible to achieve in SVG.

The properties that can be exported are:

- Base color
- Base alpha
- Base thickness
- Dashes

Animations

The exporter supports the creation of SVG animations. When the Mode is set to Animation, all frames from a render – one when rendering a frame (`F12`) or all when rendering an animation (`Shift-F12`) – into a single file. Most modern browsers support the rendering of SVG animations.

Fig. 2.2132: An SVG animation rendered with the exporter.

Exporting Fills

Fills are colored areas extracted from a Freestyle render result. Specifically, they are defined by a combination of the Contour and External Contour edge type, combined with some predicates. The fill result can be unexpected, when the SVG renderer cannot correctly draw the path that the exporter has generated. This problem is extra apparent in animations.

Fig. 2.2133: An example of a .svg result produced by the Freestyle SVG Exporter. Model by Julien Deswaef.

Fills support holes and layering. When using layers, the exporter tries to render objects with the same material as the patch. The exporting of fills and especially the order in which they are layered is by no means perfect. In most cases, these problems can be easily solved in Inkscape or a text editor.

2.9.7 Workflows

Rendering Animations

While rendering stills will allow you to view and save the image from the render buffer when it is complete, animations are a series of images, or frames, and are automatically saved directly out to disk after being rendered.

After rendering the frames, you may need to edit the clips, or first use the Compositor to do green-screen masking, matting, color correction, DOF, and so on to the images. That result is then fed to the Sequencer where the strips are cut and mixed and a final overlay is done.

Finally you can render out from the Sequencer and compress the frames into a playable movie clip.

Workflow

Generally, you do a lot of intermediate renders of different frames in your animation to check for timing, lighting, placement, materials, and so on. At some point, you are ready to make a final render of the complete animation for publication.

There are two approaches you can use when making a movie, or animation, with or without sound. The approach you should use depends on the amount of CPU time you will need to render the movie. You can render a "typical" frame at the desired resolution, and then multiply by the number of frames that will ultimately go into the movie, to arrive at an total render time.

If the total render time is an hour or more, you want to use the "Frame Sequence" approach. For example,

if you are rendering an one-minute video clip for film, there will be (60 seconds per minute) X (24 frames per second) or 1440 frames per minute. If each frame takes 30 seconds to render, then you will be able to render two frames per minute, or need 720 minutes (12 hours) of render time.

Rendering takes all available CPU time; you should render overnight, when the computer is not needed, or set Blender to a low priority while rendering, and work on other things (be careful with the RAM space!).

Direct Approach

The Direct Approach, which is highly **not** recommended and not a standard practice, is where you set your output format to an AVI or MOV format, and click *Animation* to render your scene directly out to a movie file. Blender creates one file that holds all the frames of your animation. You can then use Blender's VSE to add an audio track to the animation and render out to an MPEG format to complete your movie.

Frame Sequence

The Frame Sequence is a much more stable approach, where you set your output format to a still format (such as JPG, PNG or MultiLayer), and click *Animation* to render your scene out to a set of images, where each image is a frame in the sequence.

Blender creates a file for each frame of the animation. You can then use Blender's compositor to perform any frame manipulation (post processing). You can then use Blender's VSE to load that final image sequence, add an audio track to the animation, and render out to an MPEG format to complete your movie. The Frame Sequence approach is a little more complicated and takes more disk space, but gives you more flexibility.

Here are some guidelines to help you choose an approach.

Direct Approach

- short segments with total render time < 1 hour
- stable power supply
- computer not needed for other uses

Frame Sequence Approach

- total render time > 1 hour

- **post-production work needed**
  - Color/lighting adjustment

- Green screen/matte replacement
- Layering/compositing
- Multiple formats and sizes of ultimate product

- intermediate frames/adjustments needed for compression/codec
- precise timing (e.g. lip-sync to audio track) needed in parts
- may need to interrupt rendering to use the computer, and want to be able to resume rendering where you left off.

Frame Sequence Workflow

1. First prepare your animation.
2. In the *Dimensions* panel, choose the render size, Pixel Aspect Ratio, and the Range of Frames to use, as well as the frame rate, which should already be set.
3. In the Output panel set up your animation to be rendered out as images, generally using a format that does not compromise any quality.
4. Choose the output path and file type in the Output panel as well, for example `//render/my-anim-`.
5. Confirm the range of your animation frame Start and End.
6. Save your blend-file.
7. Press the big *Animation* button. Do a long task [like sleeping, playing a video game, or cleaning your driveway] while you wait for your computer to finish rendering the frames.
8. Once the animation is finished, use your OS file explorer to navigate into the output folder ("`render` in this example). You will see lots of images (.png or .exr, etc... depending on the format you chose to render) that have a sequence number attached to them ranging from 0000 to a max of 9999. These are your single frames.
9. In Blender, now go into the *video sequence editor*.
10. Choose *Add Image* from the add menu. Select all the frames from your output folder that you want to include in your animation (Press A to Select All easily). They will be added as a strip to the sequence editor.
11. Now you can edit the strip and add effects or simply leave it like it is. You can add other strips, like an audio strip.
12. Scrub through the animation, checking that you have included all the frames.
13. In the Scene Render buttons, in the Post Processing panel, activate *Sequencer*.

14. In the Output panel, choose the container and codec you want (e.g. `MPEG H.264`) and configure them. The video codecs are described on the previous page: *Output Options*.

15. Click the *Animation* render button and Blender will render out the sequence editor output into your movie.

    Why go through all this hassle? Well, first of all, if you render out single frames you can stop the render at any time by pressing `Esc` in the render window or UV/image editor. You will not lose the frames you have already rendered, since they have been written out to individual files. You can always adjust the range you want to continue from where you left off.

    You can edit the frames afterwards and post-process them. You can add neat effects in the sequence editor. You can render the same sequence into different resolutions (640×480, 320×240, etc) and use different codecs (to get different file sizes and quality) with almost no effort whatsoever.

Hints

Your computer accidentally turns off in the middle of rendering your movie!
Unless your animation renders in a few minutes, it is best to render the animation as separate image files. Instead of rendering directly to a compressed movie file, use a loss-less format (i.e. `PNG`).

This allows you an easy recovery if there is a problem and you have to re-start the rendering, since the frames you have already rendered will still be in the output directory.

Just disable the *Overwrite* option to start rendering where you left off.

You can then make a movie out of the separate frames with Blender's sequence editor or using 3rd party encoding software.

Animation Preview It can be useful to render a subset of the animated sequence, since only part of an animation may have an error.

Using an image format for output, you can use the *Frame Step* option to render every *N'th* frame. Then disable *Overwrite* and re-render with *Frame Step* set to 1.

Command Line

In some situations we want to increase the render speed, access Blender remotely to render something or build scripts that use the command line.

One advantage of using the command line is that we do not need the X server (in the case of Linux) and consequently we can render remotely by SSH or telnet.

To see a list of available flags (for example to specify which scene to render, the end frame number, etc...), simply run:

```
blender --help
```

Note: Arguments are executed in the order they are given!

The following command will not work, since the output and extension are set after Blender is told to render:

```
blender␣
↪-b file.blend -a -x 1 -o //render
```

The following command will behave as expected:

```
blender␣
↪-b file.blend -x 1 -o //render -a
```

Always position `-f` or `-a` as the last arguments.

Platforms

How to actually execute Blender from the command line depends on the platform and where you have installed Blender. Here are basic instructions for the different platforms.

Linux

Open a terminal, then go to the directory where Blender is installed, and run Blender like this:

```
cd <blender installation directory>
./blender
```

If you have Blender installed in your `PATH` (usually when Blender is installed through a distribution package), you can simply run:

```
blender
```

macOS

Open the terminal application, go to the directory where Blender is installed, and run the executable within the app bundle, with commands like this:

```
cd /Applications/Blender
./blender.app/Contents/MacOS/blender
```

If you need to do this often, you can make an alias so that typing just `blender` in the terminal works. For that you can run a command like this in the terminal (with the appropriate path).

```
echo "alias blender=/
 ↪Applications/Blender/blender.app/
 ↪Contents/MacOS/blender" >> ~/.profile
```

If you then open a new terminal, the following command will work:

```
blender
```

MS-Windows

Open the Command Prompt, go to the directory where Blender is installed, and then run Blender:

```
cd c:\<blender installation directory>
blender
```

You can also add the Blender folder to your system `PATH` so that do you do not have to `cd` to it each time.

Examples

Single Image

```
blender -b file.blend -f 10
```

`-b` Render in the background (without UI).

`file.blend` Path to the blend-file to render.

`-f 10` Render only the 10th frame.

```
blender -b file.blend -o /project/
 ↪renders/frame_##### -F EXR -f -2
```

`-o /project/renders/frame_#####` Path of where to save the rendered image, using five padded zeros for the frame number.

`-F EXR` Override the image format specified in the blend-file and save to an OpenEXR image.

`-f -2` Render only the second last frame.

> **Warning:** Arguments are case sensitive! `-F` and `-f` are not the same.

Animation

```
blender -b file.blend -a
```

`-a` Render the whole animation using all the settings saved in the blend-file.

```
blender -b file.blend␣
 ↪-E BLENDER_RENDER -s 10 -e 500 -t 2 -a
```

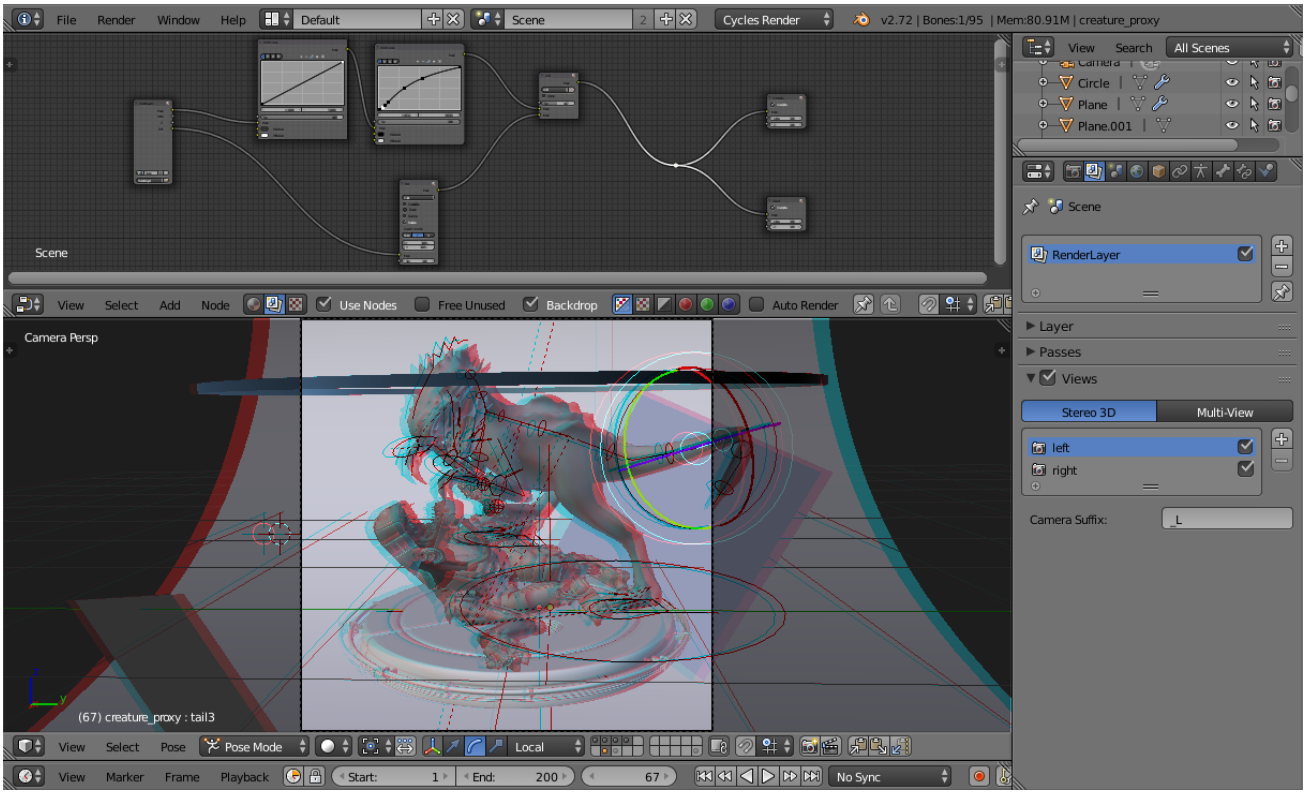
-E BLENDER_RENDER Use the "Blender Render" engine. For a list of available render engines, run `blender -E help`.

-s 10 -e 500 Set the start frame to `10` and the end frame to `500`.

-t 2 Use only two threads.

Multiview

Introduction



Since version 2.75, Blender has come with a new feature called Multiview. Multiview is a complete toolset for working with stereoscopic rendering in Blender. It works with both the Blender Internal and Cycles rendering engines and it also supports many different stereo 3D visualization types.

Note: If you have a real 3D display at some point you can change the 3D display mode in the Window menu, by calling the Stereo 3D operator. Be aware that some modes require a fullscreen editor to work, and this can be taxing on your CPU.

Usage

For example, we will take an existing blend file that was made for monoscopic rendering and transform it to be stereo 3D ready.

Fig. 2.2134: Creature Factory 2 by Andy Goralczyk Rendered in Stereo 3D (anaglyph).

Note: Multi-View drawing requires capable graphics card and drivers with *Triple Buffer* support. If the *Automatic* mode does not work, set the *Window Draw Method* in the *System User Preferences*.

Introduction

Start opening up your project file, in this case `turntable.blend` from the *Creature Factory 2* Open Movie Workshop series from the Blender Institute by Andy Goralczyk.

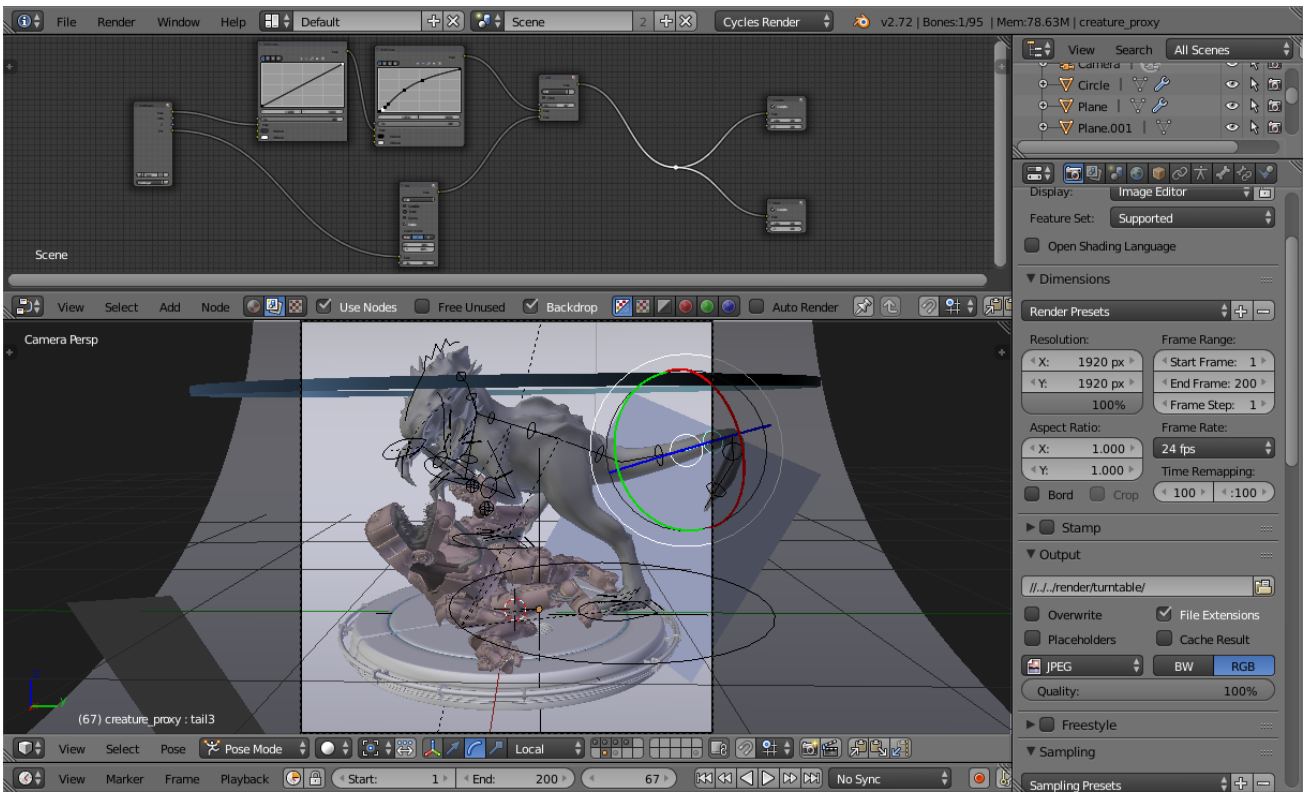


Fig. 2.2135: Turn Table Creature Factory 2.

Views Setup

Go to the *Render Layers* panel and enable *Views* for this scene.

Note: When you turn on *Views* in the scene, you get 3D preview in the viewport, as well as multiple panels that are now accessible all over the user interface.

Camera

To tweak the stereo 3D parameters, select the camera in the Outliner. In the Camera panel go to the Stereoscopy tab and change the *Convergence Distance*.

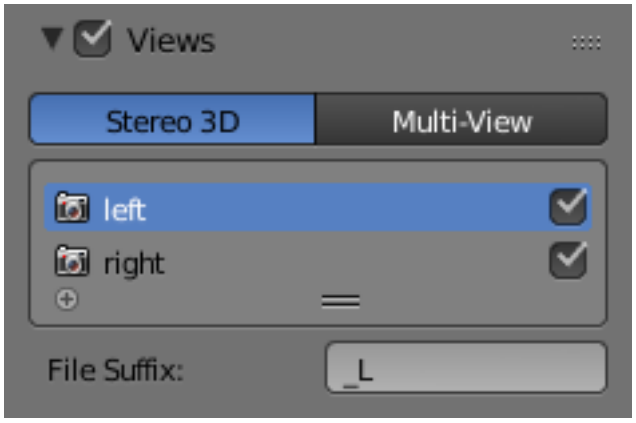


Fig. 2.2136: Scene Render Views.

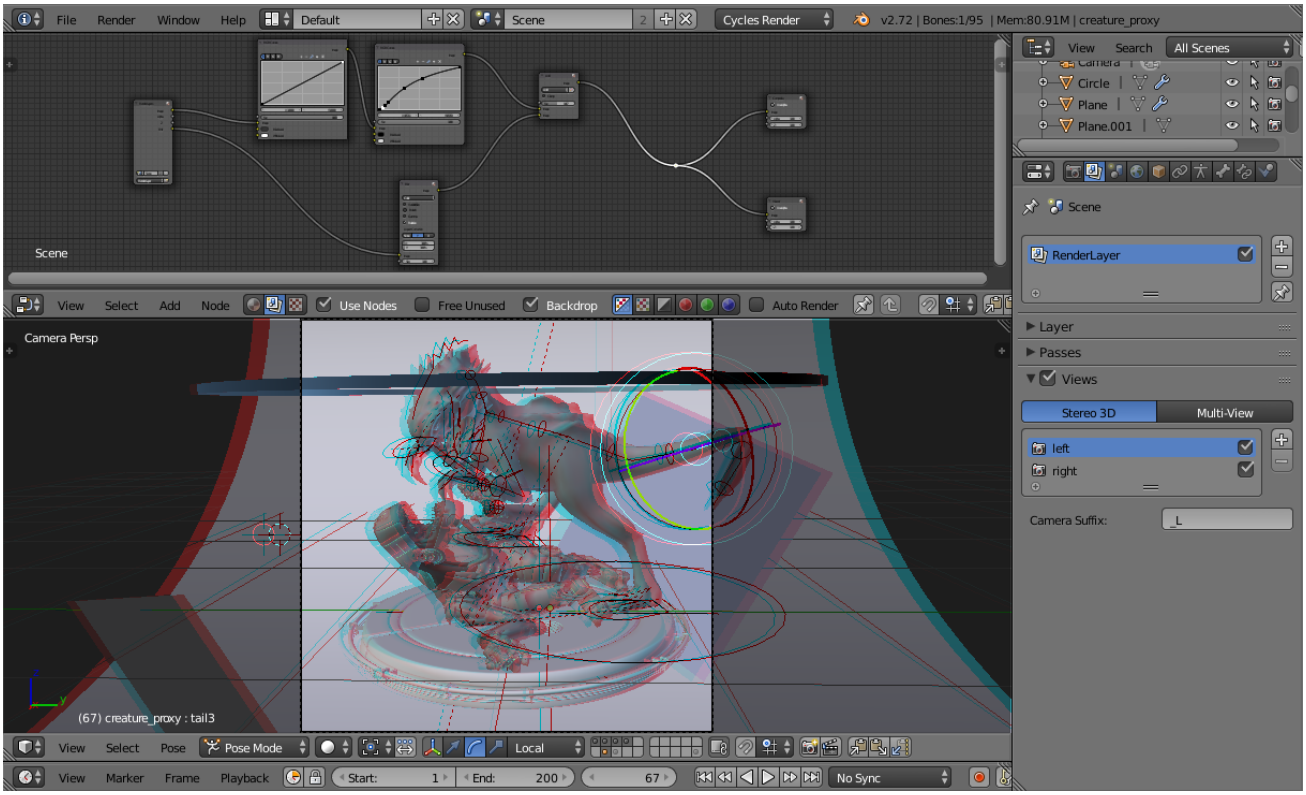


Fig. 2.2137: Viewport with 3D visualization.

The viewport will respond in real-time to those changes allowing you to preview the current depth value of the scene.

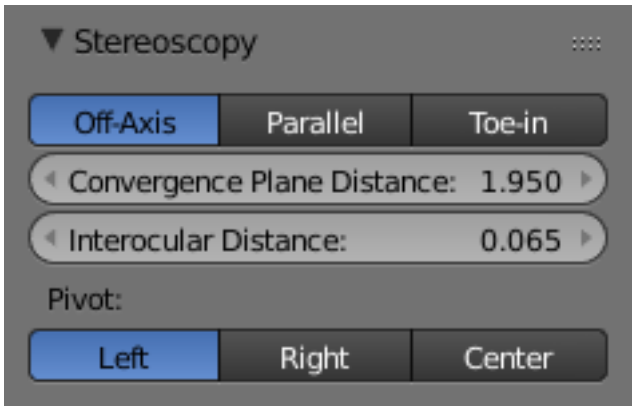


Fig. 2.2138: Stereo Convergence Distance.

Viewport

Before fine-tuning the camera parameters, you can set the convergence plane in the viewport based in your scene depth layout. Go outside the camera view and you will instantly see the convergence plane in front of the camera.

You can toggle this and other display settings in the Stereoscopy panel of the 3D Views properties region. In the following image, the cameras frustum volumes are also visible.

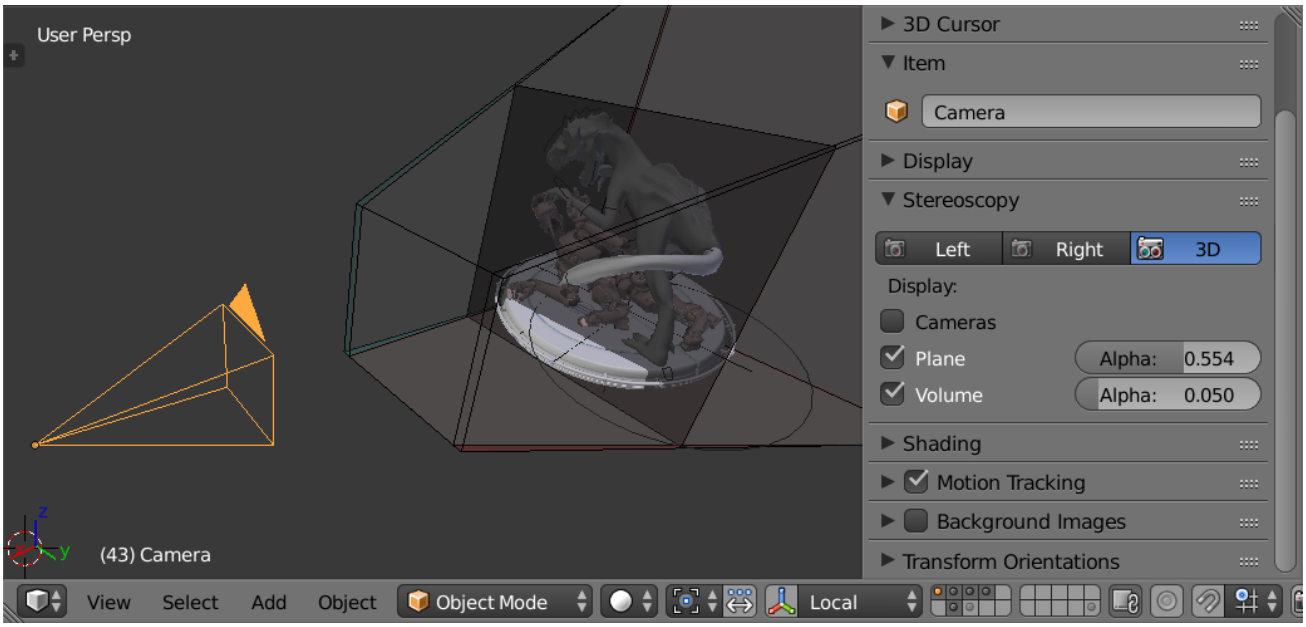


Fig. 2.2139: Viewport Plane and Volume Stereo Preview.

Stereo 3D Display

If you have a real 3D display at some point, you can change the 3D display mode in the Window menu, by calling the Stereo 3D operator. Be aware that some modes require a fullscreen editor to work.

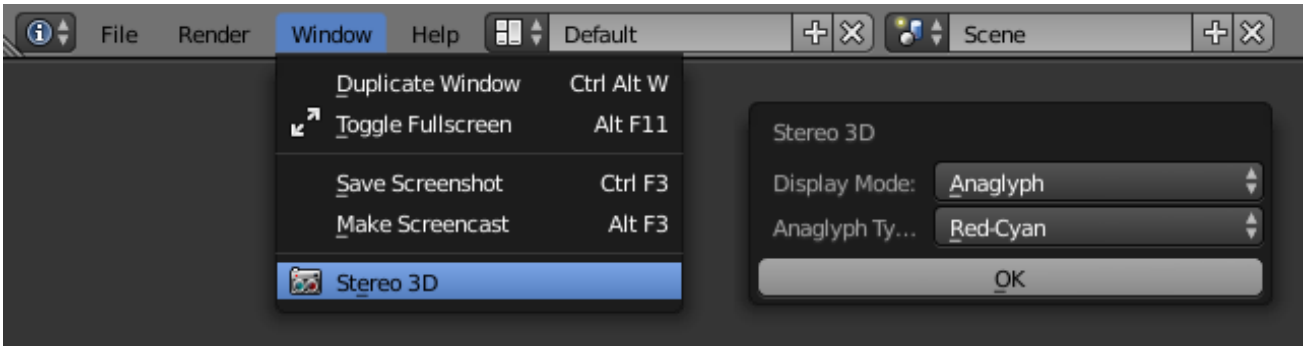


Fig. 2.2140: Window Menu, Stereo 3D Operator.

OpenGL Preview

Before rendering your scene you can save an OpenGL preview of the animation for testing in the final display. In the Render Output panel you can choose the output *Views Format*.

The options include individual files per view, top-bottom, anaglyph among others. Pick the one that fits your display requirements.

None

Fig. 2.2141: Turn Table OpenGL Rendering Preview.

Rendering and UV/Image Editor

Once you are happy with the results you can render out the final animation. In the UV/Image Editor you can inspect the individual views and the stereo result.

Image Formats

Your final animation can be saved in more robust formats than the ones used by the OpenGL render preview. In this example we saved as cross-eyed side-by-side stereo 3D.

Final Considerations

As this guide showed, there is more to stereo 3D rendering than just generate two images. The earlier the stereo pipeline is considered the smoother it will get. The following sections are a more in-depth view of the individual components we visited in the workflow.

Window Stereo 3D Display

An essential component of the Stereoscopy pipeline is the ability to display the stereo image in a proper display. Blender supports from high-end 3D displays to simple red-cyan glasses. On top of that you can set a different display mode for

Fig. 2.2142: Side by Side Cross-Eye Format.

each window.

The display mode can be changed via the Window menu or if you create your own shortcuts for the `wm.set_stereo_3d` operator.

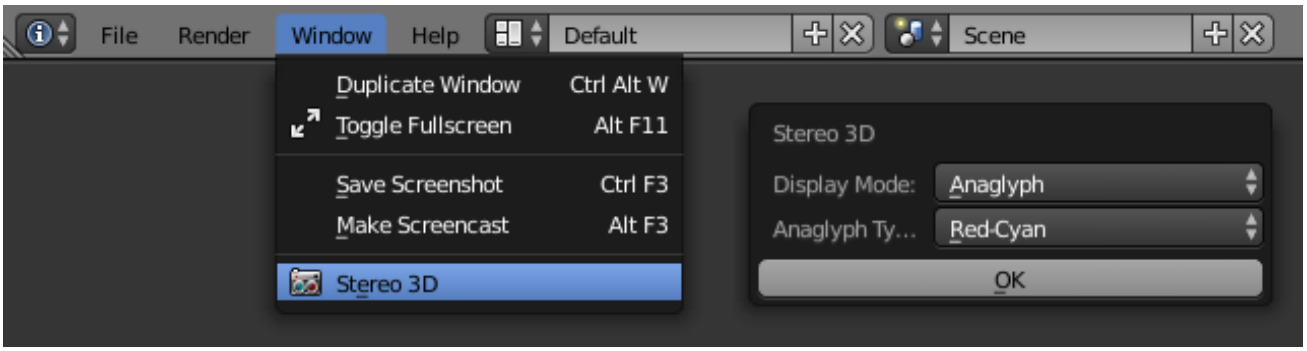


Fig. 2.2143: Window Menu, Stereo 3D Operator.

Display Mode

Anaglyph Render two differently filtered colored images for each eye. Anaglyph glasses are required. We support Red-Cyan, Green-Magenta and Yellow-Blue glasses.

Interlace Render two images for each eye into one interlaced image. A 3D-ready monitor is required. We support Row, Column and Checkerboard Interleaved. An option to Swap Left/Right helps to adjust the image for the screen. This method works better in fullscreen.

Time Sequential Renders alternate eyes. This method is also known as Page Flip. This requires the graphic card to support Quad Buffer and it only works in fullscreen.

Side-by-Side Render images for left and right eye side-by-side. There is an option to support Cross-Eye glasses. It works only in fullscreen, and it should be used with the Full Editor operator.

Top-Bottom Render images for left and right eye one above another. It works only in fullscreen, and it should be used with the Full Editor operator.

Note: Full Screen Stereo 3D Modes

If you have a 3D display most of the time you will use it to see in stereo 3D you will have to go to the fullscreen mode. In fact some modes will only work in the full window mode that hides most of the user interface from the work area. In this case it is recommended to work with two monitors, using the 3D screen for visualizing the stereo result while the other screen can be used for the regular Blender work.

Stereo 3D Camera

When using the Stereo 3D scene view setup a stereo pair is created on-the-fly and used for rendering and previsualization. For all the purposes this works as two cameras that share most parameters (focal length, clipping, ...). The stereo pair, however, is offsetted, and can have unique rotation and shift between itself.

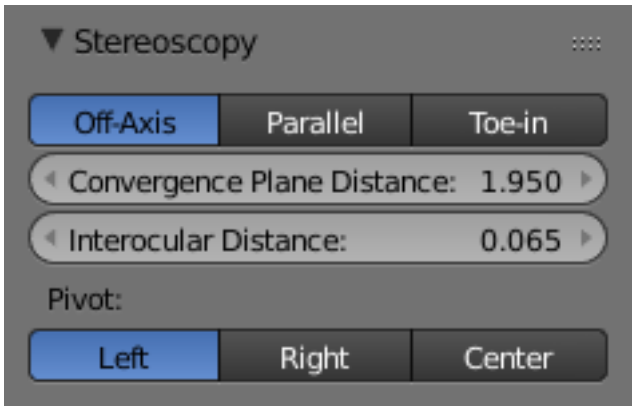


Fig. 2.2144: Stereo 3D Camera Settings.

Interocular Distance Set the distance between the camera pair. Although the convergence of a stereo pair can be changed in post-production, different interocular distances will produce different results due to the parts of the scene being occluded from each point of view.

Convergence Plane Distance The converge point for the stereo cameras. This is often the distance between a projector and the projection screen. You can visualize this in the 3D View.

Convergence Mode

Off-Axis The stereo camera pair is separated by the interocular distance, and shifted inwards so it converges in the convergence plane. This is the ideal format since it is the one closest to how the human vision works.

Parallel This method produces two parallel cameras that do not converge. Since this method needs to be manually converged it cannot be used for viewing. This method is common when combining real footage with rendered elements.

Toe-in A less common approach is to rotate the cameras instead of shifting their frustum. The Toe-in method is rarely used in modern 3D productions.

Pivot The stereo pair can be constructed around the active camera with a new camera built for each eye (Center Pivot) or using the existing camera and creating (Left or Right). The latter is what is used when only one eye needs to be rendered for an existing mono 2D project.

Viewport Stereo 3D

When you enable 'Views' in the Render Layer panel, a new area is available in the 3D View properties region. In this panel you can pick whether to see the stereo 3D in the viewport, or which camera to see. It also allow you to see the Cameras, the Plane and the Volume of the stereo cameras.

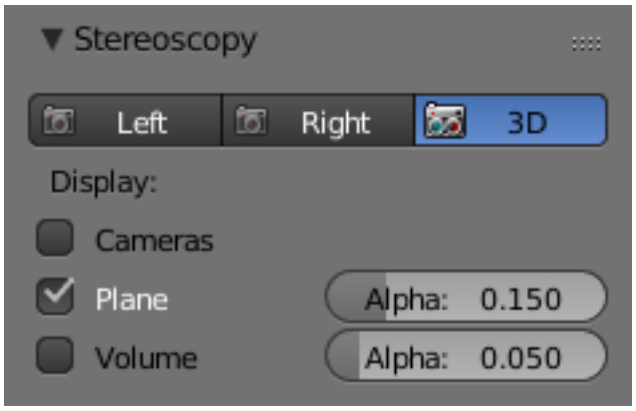


Fig. 2.2145: Viewport Stereo 3D Settings.

Cameras When working with the Stereo 3D Views setup you can inspect what each individual generated camera is looking or the combined result of them. In the Multi-View mode you can see the combined result of the left and right cameras (when available) or the current selected camera.

Plane The convergence plane represents the screen as it is perceived by the audience. Visualizing it in the 3D View allows you to layout your scene based on your depth script outside the camera view.

Volume The intersection of the stereo cameras frustums helps planning the show by avoiding elements being visible by only one camera. The volume is defined by the camera's start and end clipping distances. The areas that are in the frustum of one camera only are known as *retinal rivalry areas*. They are tolerated in the negative space (the region from the convergence plane into the image) but are to be avoided at all costs in the positive space (the area from the convergence plane to the camera).

Multi-View and Stereo 3D Image I/O

Multi-View and Stereo 3D Multi-View images can be saved in special formats according to the production requirements. By default the system saves each view as an individual file, thus generating as many files as views to be rendered. In stereo 3D productions, for the final deployment or even intermediary previews it is convenient to save stereo 3D images, that are ready to use with 3D displays or simple anaglyph glasses. The formats supported match the display modes available for the window.

Lossy-Formats Some stereo 3D formats represent a considerable loss of data. For example, the Anaglyph format will cap out entire color channels from the original image. The Top-Bottom compressed will discard half of your vertical resolution data. The Interlace will mash your data considerably. Once you export in those formats, you can still import the image back in Blender, for it to be treated as Stereo 3D. You will need to match the window stereo 3D display mode to the image stereo 3D format though.

Lossless Formats Some formats will preserve the original data, leading to no problems on exporting and importing the files back in Blender. The Individual option will produce separate images that (if saved in a lossless encoding such as `PNG` or `OpenEXR`) can be loaded back in production with no loss of data. For the Stereo 3D formats the only lossless options are *Top-Bottom* and *Side-by-Side* without the Squeezed Frame option.

Multi-View OpenEXR Another option is to use Multi-View OpenEXR files. This format can save multiple views in a single file and is backward compatible with old OpenEXR viewers (you see only one view though). Multi-View native support is only available to OpenEXR.

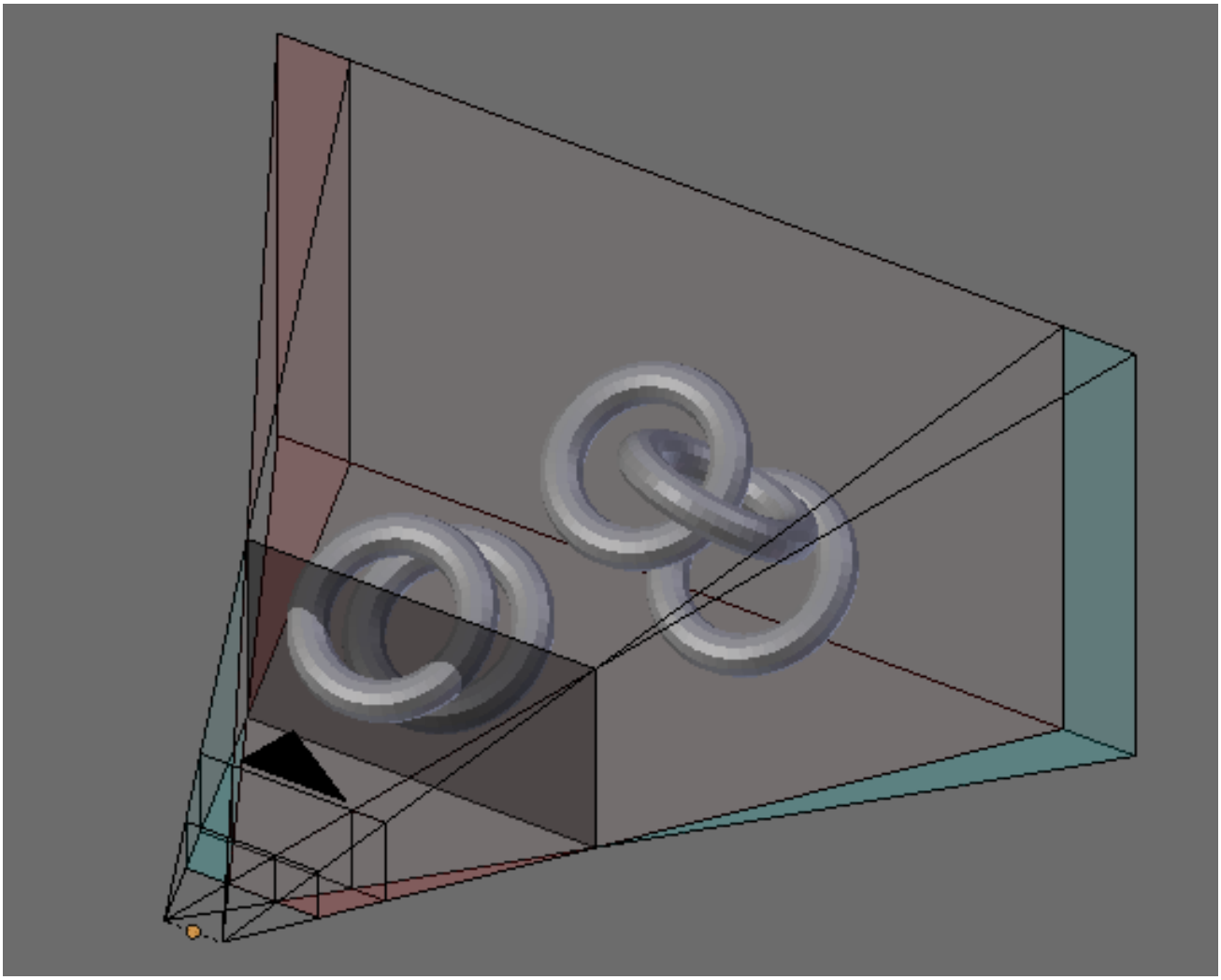


Fig. 2.2146: Viewport 3D: Convergence Plane and Volume Display.

Image Editor

View Menu After you render your scene with Stereo 3D you will be able to see the rendered result in the combined stereo 3D or to inspect the individual views. This works for Viewer nodes, render results or opened images.

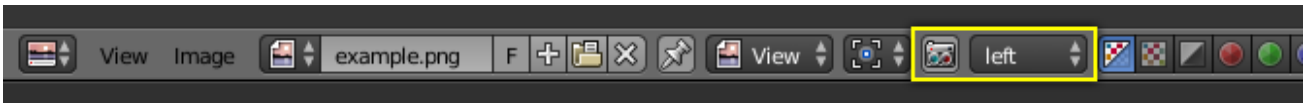


Fig. 2.2147: Stereo 3D and View menu.

Views Format When you drag and drop an image into the UV/Image Editor, Blender will open it as an individual images at first. If your image was saved with one of the Stereo 3D formats, you can change how Blender should interpret the image by switching the mode to Stereo 3D, turning on Use Multi-View and picking the corresponding stereo method.

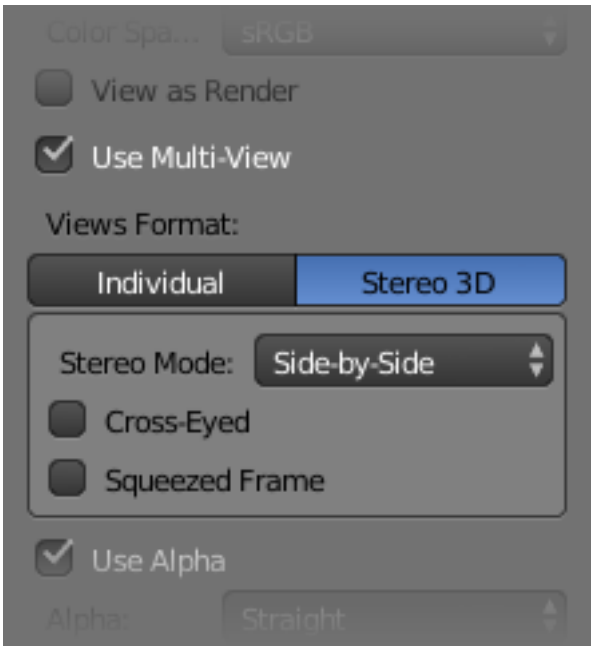


Fig. 2.2148: Views Formats and Stereo 3D.

Compositor

The compositor works smoothly with Multi-View. The compositing of a view is completed before the remaining views start to be composited. The pipeline is the same as the single-view workflow, with the difference that you can use Image, Movies or Image Sequences in any of the supported Multi-View formats.

The views to render are defined in the current scene views, in a similar way as you define the composite output resolution in the current scene render panel, regardless of the Image nodes resolutions or Render Layers from different scenes.

Note: Single-View Images

If the image from an Image Node does not have the view you are trying to render, the image will be treated as a single-view image.

Switch View Node If you need to treat the views separately you can use the *Switch View node* to combine the views before an output node.

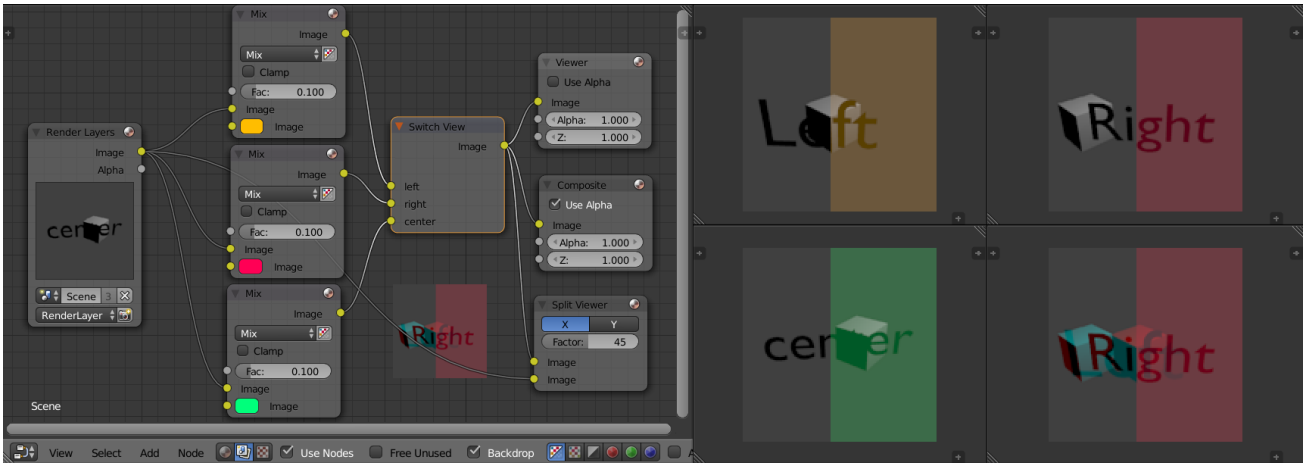


Fig. 2.2149: Compositor, Backdrop and Split Viewer Node.

Tip: Performance

By default, when compositing and rendering from the user interface all views are rendered and then composited. During test iterations you can disable all but one view from the Scene Views panel, and re-enable it after you get the final look.

2.9.8 OpenGL Render

OpenGL rendering uses the 3d View's drawing for quick *preview* renders.

This allows you to inspect your animatic (for object movements, alternate angles, etc.).

This can also be used to preview your animations – in the event your scene is too complex for your system to play back in real-time in the 3D View.

You can use OpenGL to render both images and animations.

Below is a comparison between the OpenGL render and a final render using the Cycles Render engine.

Table 2.103: Full Render.

Tip: Showing Only Rendered Objects

To access this option, enable the *Only Render* in the *Display Panel*.

While this option is not specific to OpenGL rendering, its often useful to enable, since it removes data such as rigs and empties that can be a distraction.

Settings

For the most part, *OpenGL Render* uses view-port, however, some render settings are used too:

- Render Dimensions
- Render Aspect
- Anti-Aliasing, Samples & Full Sample (for slower, higher quality output).
- Alpha Transparency Mode.
- File Format & Output (file-path, format, compression settings... etc).

Note: These options are only available in the *Render* properties when using Blender-Internal. When using other rendering engines, access these options from: *Render* → *OpenGL Render Options*.

Rendering

Activating OpenGL render from the menu will render from the active camera.

You can also render any view-port, from the header of the *3D View*, using the small button showing a *Camera*.

Fig. 2.2152: OpenGL Render buttons.

As with a normal render, you can abort it with `Esc`.

Render a Still Image Click on the small button showing a *camera* in the header of the 3D View.

Or from the menu: *Render* → *OpenGL Render Image* from the header of the *Info Editor*

Render an Animation Click on the small button showing a *slate* in the header of the 3D View.

Or from the menu: *Render* → *OpenGL Render Animation* from the header of the *Info Editor*

Render from the Sequencer Click on the small button showing a *slate* in the header of *Sequencer* preview region.

Using scene strips in the sequencer you can edit together scenes to quickly render an entire sequence of shots.

This can be activated using the render icons in the sequencer's playback header.

Known Limitations

OpenGL Anti-Aliasing Support

Some graphics cards do not support this feature (known as the frame-buffer multi-sample OpenGL extensions).

In this case rendering works but no anti-aliasing is performed.

Enabling *Full Sample*, can be used to workaround this limit, because it does not rely on hardware multi-sample support.

Hint: Exact extensions needed, as listed in output from *Save System Info* (OpenGL section):

- `GL_ARB_texture_multisample`
- `GL_EXT_framebuffer_blit`
- `GL_EXT_framebuffer_multisample_blit_scaled`

- `GL_EXT_framebuffer_multisample`

2.9.9 Audio Rendering

Audio Rendering

Audio can be rendered from the *Properties Editor* → *Render tab* → *Render* → *Audio*.

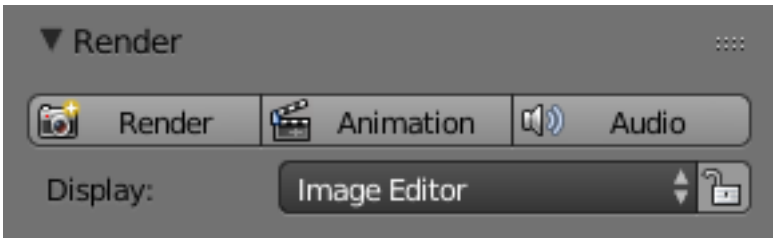


Fig. 2.2153: Render panel.

Options

Relative Path Select the file relative to the blend-file.

Accuracy Sample accuracy, important for animation data (the lower the value, the more accurate).

Audio Containers See *here*.

Codec Some *Audio Containers* also have option to choose a codec. For more information see *here*.

Split Channels Each audio channel will be rendered into a separate file.

See also:

- See *Scene Audio* settings.
- See *Audio Output Settings* settings.
- See *Audio Preferences*.

Speaker

The speaker object is used to give sound in the 3D View. After adding the object the various settings can be changed in the properties editor.

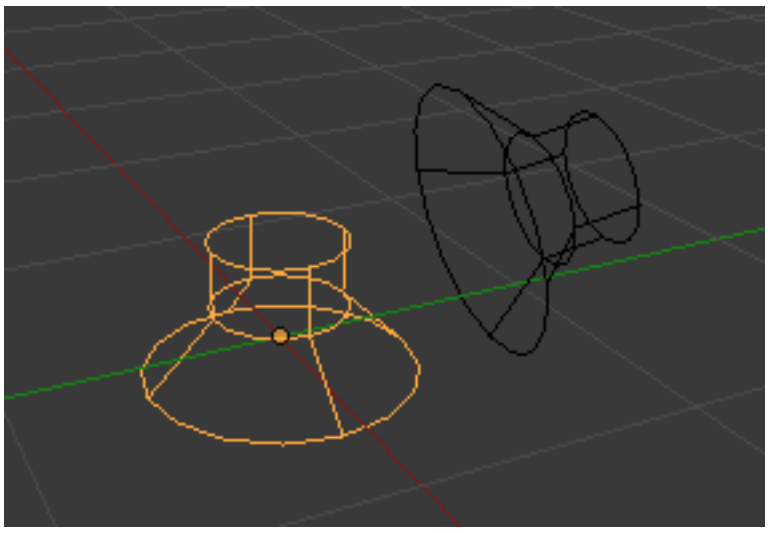


Fig. 2.2154: Speaker Objects.

Options

Sound

Mute Toggles whether or not the sound can be heard.

Volume Adjust the loudness of the sound.

Pitch Can be used to bend the pitch of the sound to be either deeper or higher.

Distance

Volume

Minimum Minimum volume, no matter how far the object is.

Maximum Maximum volume, no matter how far the object is.

Attenuation How strong the distance affects the volume.

Distance

Maximum Maximum distance for volume calculation.

Reference Reference distance at which volume is 100%.

Cone

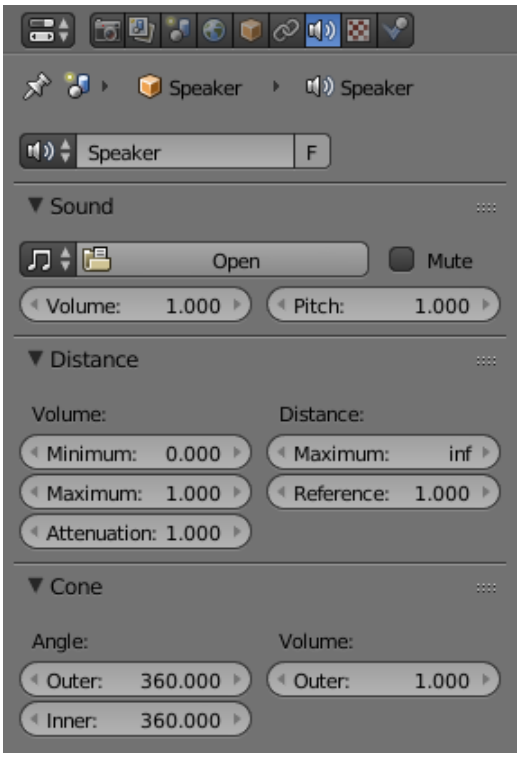
Angle

Outer Angle of the outer cone in degrees. Outside this cone the volume is the outer cone volume (see below). Between the inner and outer cone the volume is interpolated.

Inner Angle of the inner cone in degrees. Inside the cone the volume is 100%.

Volume

Outer Volume outside the outer cone.



2.10 Compositing

2.10.1 Introduction

Compositing Nodes allow you to assemble and enhance an image (or movie). Using composition nodes, you can glue two pieces of footage together and colorize the whole sequence all at once. You can enhance the colors of a single image or an entire movie clip in a static manner or in a dynamic way that changes over time (as the clip progresses). In this way, you use composition nodes to both assemble video clips together and enhance them.

Note: Term: Image

The term *Image* may refer to a single picture, a picture in a numbered sequence of images, or a frame of a movie clip. The Compositor processes one image at a time, no matter what kind of input you provide.

To process your image, you use nodes to import the image into Blender, change it, optionally merge it with other images, and finally, save it.

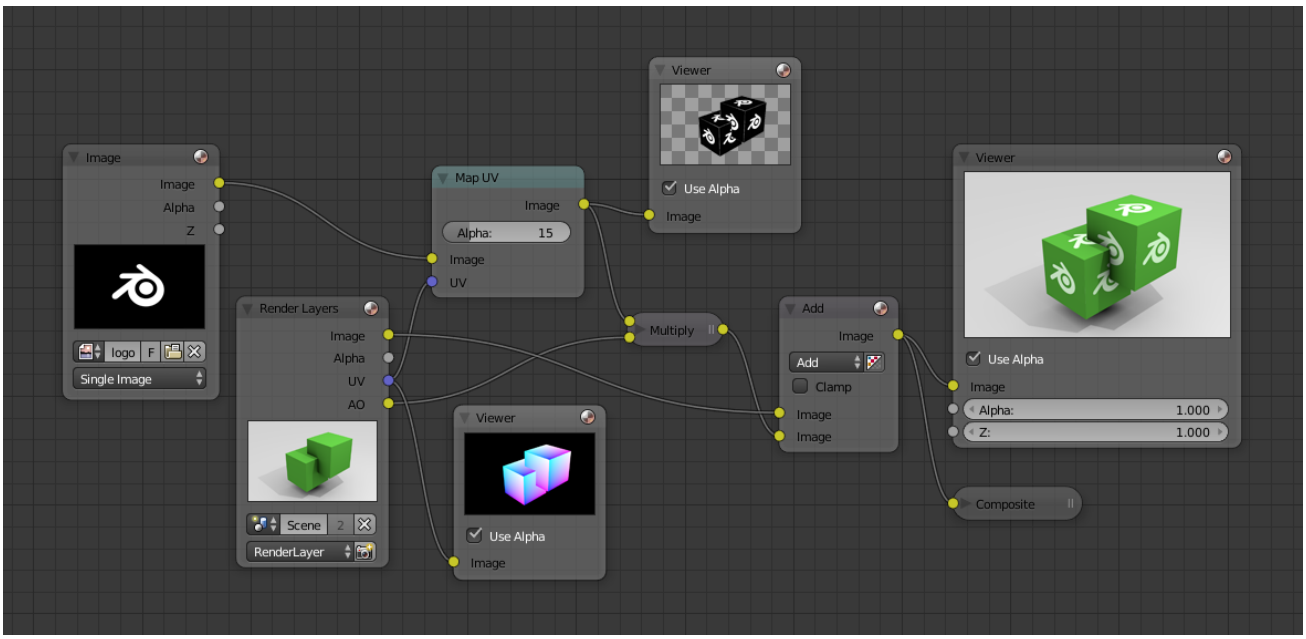


Fig. 2.2155: An example of Composition.

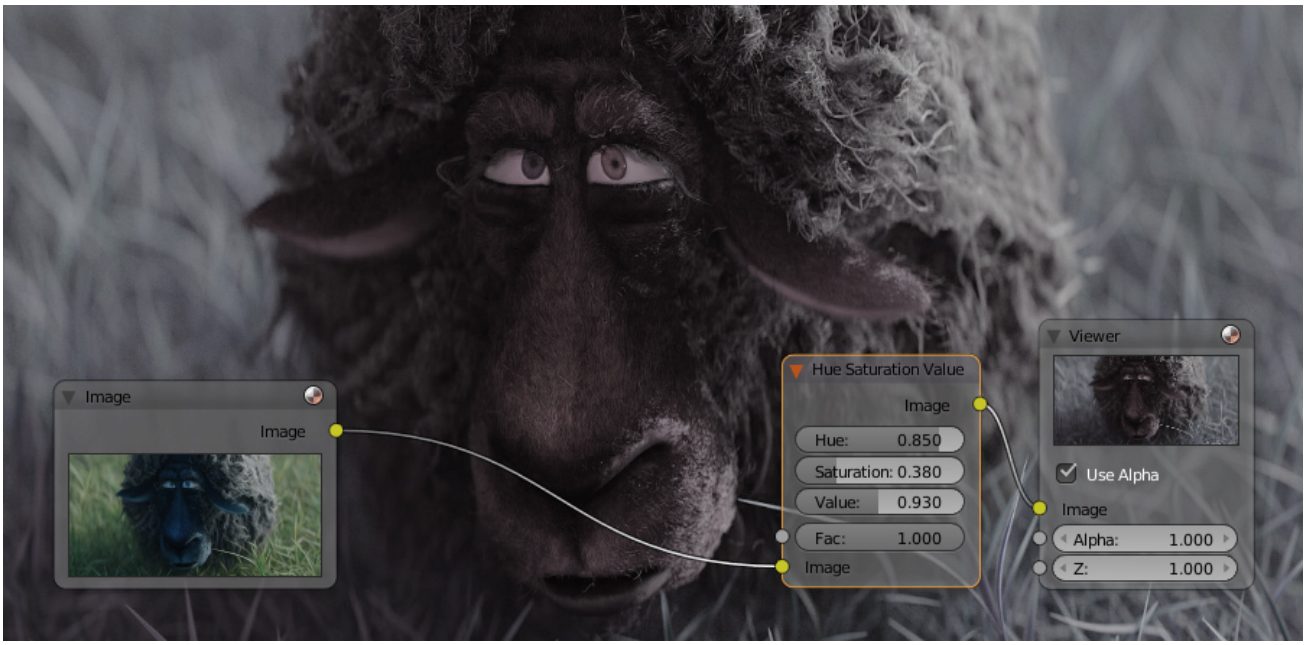


Fig. 2.2156: An example of color correction.

Getting Started

Access the *Node Editor* and enable *Composite Nodes* by clicking on the *Image* icon.

To activate nodes for compositing, click the *Use Nodes* checkbox (see *Options*).

Note: After clicking *Use Nodes* the Compositor is enabled, however, it can also be disabled in the *Post Processing Panel*.

You now have your first node setup, from here you can add and connect many types of *Compositing Nodes*, in a sort of map layout, to your heart's content (or physical memory constraints, whichever comes first).

Note: Nodes and node concepts are explained in more detail in the *Node Editor*.

Options

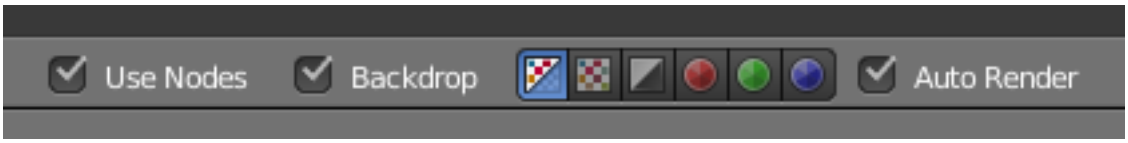


Fig. 2.2157: Compositing Specific Options.

Use Nodes Enables basic compositing set up with a *Render Layer Node* and a *Composite Node*.

Backdrop Enables the use of a backdrop using a *Viewer Node*.

Backdrop Channels See below.

Auto Render Re-render and composite changed layer when edits to the 3D scene are made.

Backdrop

Backdrop Channels Set the image to be displayed with *Color*, *Color and Alpha*, or just *Alpha*.

Zoom Sets how big the backdrop image is.

Offset Change the screen space position of the backdrop, or click the *Move* button, or shortcut `Alt-MMB` to manually move it.

Fit Automatically scales the backdrop to fit the size of the node editor.

Performance

Render Sets the quality when doing the final render.

Edit Sets the quality when making edits.

Chunk Size Max size of a title (smaller values give a better distribution of multiple threads, but more overhead).

OpenCL This allows the use of an OpenCL platform to aid in rendering. Generally, this should be enabled unless your hardware does not have good OpenCL support.

Buffer Groups Enables buffering of group nodes to increase the speed at the cost of more memory.

Two Pass Use two pass execution during editing: first calculate fast nodes, the second pass calculate all nodes.

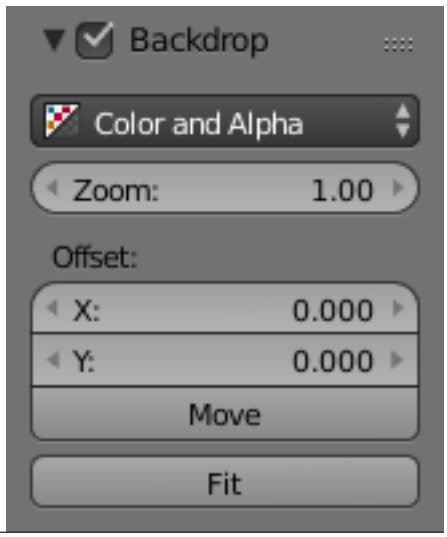


Fig. 2.2158: Backdrop Options.

Viewer Border This allows to set an area of interest for the backdrop and preview. The border is started by `Ctrl-B` and finished by selection of a rectangular area. `Ctrl-Alt-B` discards the border back to a full preview. This is only a preview option, final compositing during a render ignores this border.

Highlight Highlights the nodes that are being calculated.

Examples

You can do just about anything with images using nodes.

Raw footage from a foreground actor in front of a blue screen, or a rendered object doing something, can be layered on top of a background. Composite both together, and you have composited footage.

You can change the mood of an image:

- To make an image 'feel' colder, a blue tinge is added.
- To convey a flashback or memory, the image may be softened.
- To convey hatred and frustration, add a red tinge or enhance the red.
- A startling event may be sharpened and contrast-enhanced.
- A happy feeling – you guessed it – add yellow (equal parts red and green, no blue) for bright and sunny.
- Dust and airborne dirt are often added as a cloud texture over the image to give a little more realism.

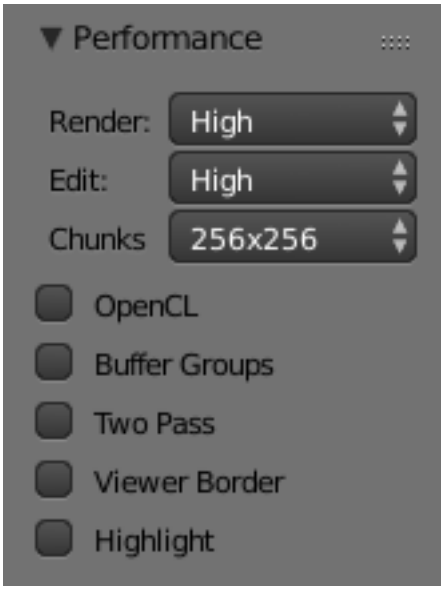


Fig. 2.2159: Performance Settings.

Image Size

It is recommended to pay attention to image resolution and color depth when mixing and matching images. Aliasing (rough edges), color *flatness*, or distorted images can all be traced to mixing inappropriate resolutions and color depths.

The compositor can mix images with any size, and will only perform operations on pixels where images have an overlap. When nodes receive inputs with differently sized Images, these rules apply:

- The first/top Image input socket defines the output size.
- The composite is centered by default, unless a translation has been assigned to a buffer using a *Translate* node.

So each node in a composite can operate on different sized images as defined by its inputs. Only the *Composite* output node has a fixed size, as defined by the settings in Properties Editor *Render* → *Dimensions*. The *Viewer* node always shows the size from its input, but when not linked (or linked to a value) it shows a small 320×256 pixel image.

Saving your Composite Image

The *Render* button renders a single frame or image. Save your image using *File* → *Save Image* or `F3`. The image will be saved using the image format settings on the Render panel.

To save a sequence of images, for example, if you input a movie clip or used a Time node with each frame in its own file, use the *Animation* button and its settings. If you might want to later overlay them, be sure to use an image format that supports an Alpha channel (such as `PNG`). If you might want to

later arrange them front to back or create a depth of field effect, use a format that supports a Z-depth channel (such as `EXR`).

To save a composition as a movie clip (all frames in a single file), use an `AVI` or `Quicktime` format, and use the *Animation* button and its settings.

2.10.2 Node Types

Input Nodes

Input nodes produce information from some source. For instance, an input could be:

- Taken directly from the active camera in a selected scene,
- from a `JPG`, `PNG`, etc. file as a static picture,
- a movie clip (such as an image sequence or video), or
- just a color or value.

These nodes generate the information that feeds other nodes. As such, they have no input-connectors; only outputs.

Bokeh Image Node

The *Bokeh Image* node generates a special input image for use with the *Bokeh Blur* filter node.

The *Bokeh Image* node is designed to create a reference image which simulates optical parameters such as aperture shape and lens distortions which have important impacts on bokeh in real cameras.

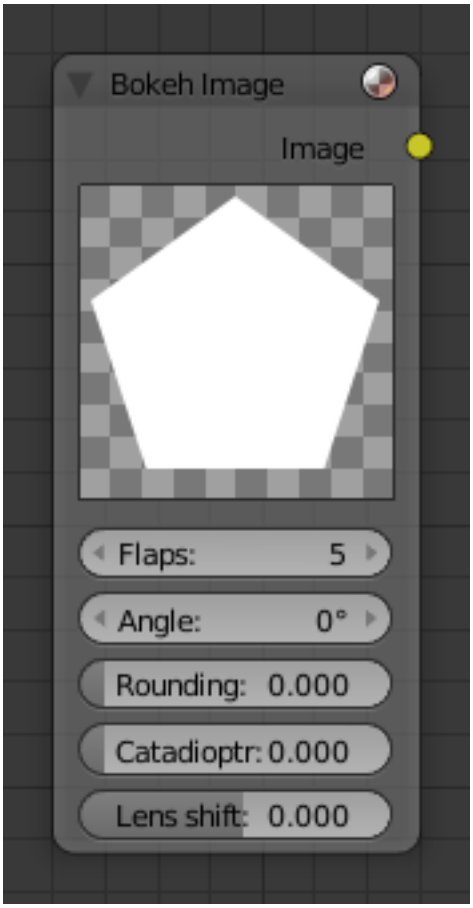


Fig. 2.2160: Bokeh Image Node.

Inputs

This node has no input sockets.

Properties

The first three settings simulate the aperture of the camera.

Flaps Sets an integer number of blades for the cameras iris diaphragm.

Angle Gives these blades an angular offset relative to the image plane

Rounding Sets the curvature of the blades with (0 to 1) from straight to bringing them to a perfect circle.

Catadioptric Provides a type of distortion found in mirror lenses and some telescopes. This can be useful to produce a visual complex bokeh.

Lens Shift Introduces chromatic aberration into the blur such as would be caused by a tilt-shift lens.

Outputs

Image The generated bokeh image.

Example

In the example below the *Bokeh Image* is used to define the shape of the bokeh for the *Bokeh Blur* node.

Fig. 2.2161: Example of *Bokeh* Image node.

Image Node

The *Image* node injects any image *format that is supported by Blender*.

Inputs

This node has no input sockets.

Properties

Image Selection of different types of media. For controls see *Data-Block Menu*. For the options see *Image Settings*.

Note: More options could be set in the properties region.

Outputs

The first two sockets are the minimum.

Image Standard image output.

Alpha Separate Alpha value.

Z Z-depth layer.

Note: MultiLayer format:

When a MultiLayer file format, like `EXR`, is loaded, each layer is made available as a socket.

Mask Node

The Mask node can be used to select a *Mask Datablock*. This node can be used with other nodes, for example to Invert, Multiply or Mix, or use as a factor input.

Inputs

This node has no input sockets.

Properties

Anti-Alias Create smooth mask edges rather than hard ones.

Feather Use or ignore feather points defined for splines see *Mask Feathers* for more details.

Size Scene Size will give an image the size of the render resolution for the scene, scaling along when rendering with different resolutions. Fixed gives a fixed size in pixels. Fixed/ Scene gives a size in pixels that still scales along when changing the render resolution percentage in the scene.

Motion Blur For animated masks, creating a motion blurred mask from the surrounding frames, with a given number of samples (higher gives better quality), and a camera shutter time in seconds.

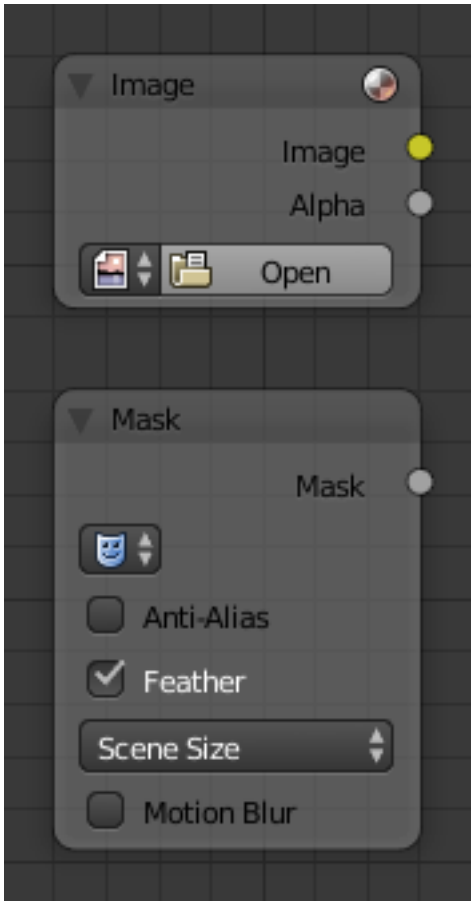


Fig. 2.2163: Mask Node.

Outputs

Mask The black and white output of the mask.

Example

In the example below the *Mask node* is used to define a rough outline of the island, where areas out side of the the island are dark, drawing the eye to the island.

Movie Clip Node

This node is a special node that uses some of the values taken from footage cameras and trackings and links them to the output. It is possible to load image sequences, but only Image and Alpha values will be available, because the other outputs will not have any values associated with them. When a tracked clip is chosen, Blender will fulfill the outputs using internal values taken from the tracking. So the controls for start and end frames will be defined at the movie clip editor.

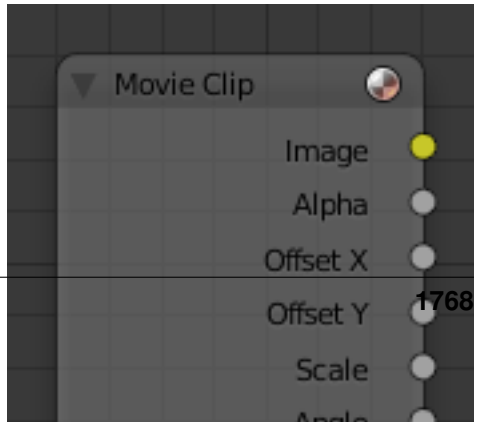


Fig. 2.2164: Example of the Mask Node.

Inputs

This node has no input sockets.

Properties

Movie Clip Used to select the movie clip. For controls see *Data-Block Menu*.

Outputs

The first two sockets are the minimum output.

Image Outputs the entire image at the specified color space.

Alpha The alpha value taken from the movie or image.

Offset X The X offset value from the footage camera or tracking.

Offset Y The Y offset value from the footage camera or tracking.

Scale The scale of the image taken from the footage camera or tracking.

Angle The lens angle taken from the footage camera or tracking.

Render Layers Node

This node is the starting place for getting a picture of your scene into the compositing node map.

Inputs

This node has no input sockets.

Properties

Scene Select the within your blend-file. The scene information taken is the raw footage (pre-compositing and pre-sequencing).

Hint: To use composited footage from another scene, it has to be rendered into a multilayer i.e. `OpenEXR` frameset as an intermediate file store and then imported with Image input node again.

Render layer A list of available *Render Layers*. The render button is a short hand to re-render the active scene.

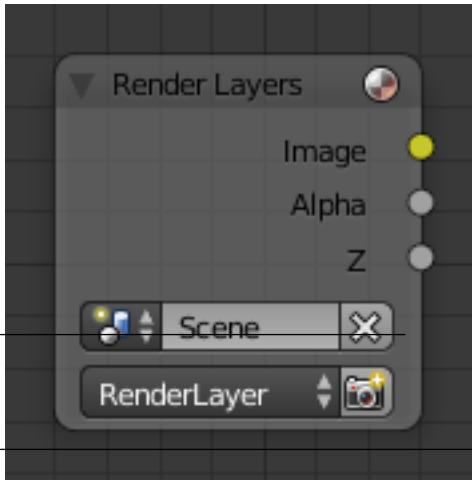


Fig. 2.2166: Render Layers Node.

Outputs

Image Rendered image.

Alpha Alpha channel.

Render passes sockets

Depending on the Render passes that are enabled, other sockets are available. See *Cycles render passes* or *Blender internal render passes*.

Z By default the Z depth pass is enabled.

RGB Node

Inputs

This node has no input sockets.

Properties

The RGB node uses the *color picker widget*.

Outputs

Color / RGBA A single RGBA color value.

Fig. 2.2167: RGB Node.

Texture Node

The Texture node makes 3D textures available to the compositor.

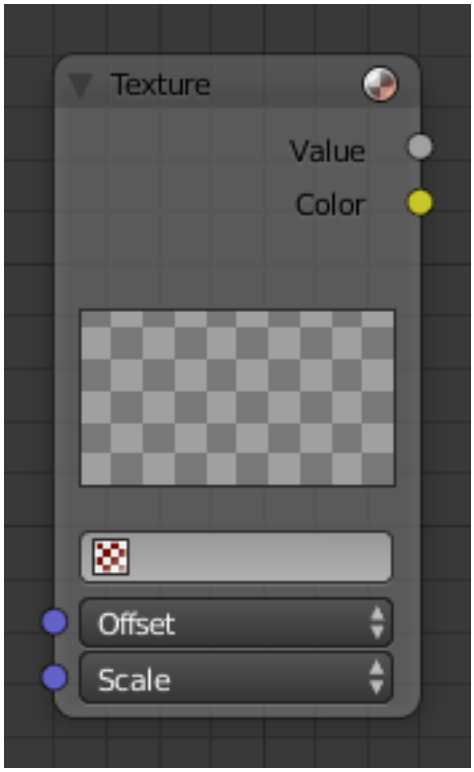


Fig. 2.2168: Texture Node.

Inputs

Offset A vector (XYZ) transforming the origin of the texture.

Scale A vector (XYZ) to scale the texture.

Properties

Texture The texture could be selected from a list of textures available in the current blend-file or link in textures. The textures themselves could not be edited in this note, but in the Texture panel.

Outputs

Value Gray scale color values.

Color Color values.

Time Node

The *Time node* generates a factor value (from 0.00 to 1.00) that changes according to the curve was drawn as time progresses through the *Timeline*.

Inputs

This node has no input sockets.

Properties

Curve The Y-value defined by the curve is the factor output. For the curve controls see: *Curve widget*.

> **Tip:** Flipping the curve around reverses the time input, but doing so is easily overlooked in the node setup.

Start, End Start frame and End frame of the range of time specifying the values the output should last. This range becomes the X-axis of the graph. The time input could be reversed by specifying a start frame greater than the end frame.

Outputs

Factor A speed of time factor (from 0.00 to 1.00) relative to the frame rate defined in the *Render Dimensions Panel*. The factor changes according to the defined curve.

Hint: Output values

The *Map Value* node can be used to map the output to a more appropriate value. With sometimes curves, it is possible that the Time node may output a number larger than one or less than zero. To be safe, use the Min/Max clamping function of the Map Value node to limit output.

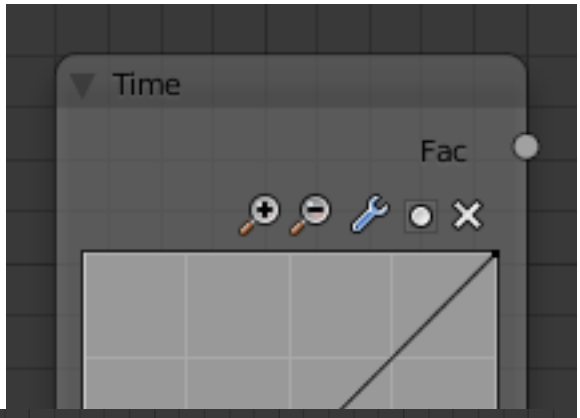


Fig. 2.2169: Time Node.

Example

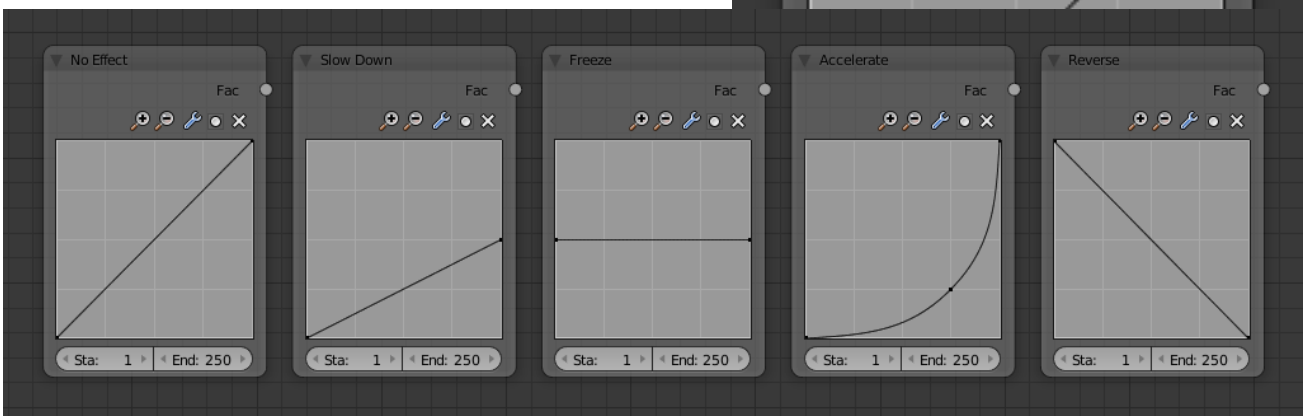


Fig. 2.2170: Time controls from left to right: no effect, slow down, freeze, accelerate, reverse

Track Position Node

The *Track Position node* is used to return information about a tracking marker to the compositor.

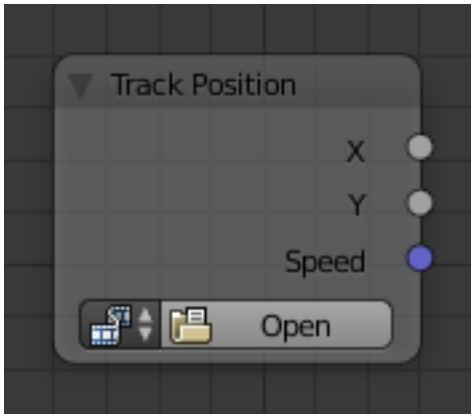


Fig. 2.2171: Track Position Node.

Inputs

This node as no inputs.

Properties

Movie Clip Used to select a Movie Clip data-block to use, for controls see *Data-Block Menu*.

Tracking Object Camera object to get track information from.

Track Name The name of the track to get track information from.

Position Which marker position to use for output.

Absolute Outputs a absolute position of a marker.

Relative Start Outputs the positions of a marker relative to the first marker of a track.

Relative Frame Outputs the positions of a marker relative to the markers of the given *Frame*.

Absolute Frame Outputs the absolute positions of a marker at the given *Frame*.

Outputs

X/Y The markers X and Y location.

Speed The velocity of the marker, measured in pixels per frame. This could be used to fake effects like motion blur by connecting it to the Vector Blur Node.

Examples

TODO.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.

Inputs

This node has no input sockets.

Properties

Single numerical value (floating point).

Fig. 2.2172: Value Node.

Outputs

Value The value set in the options.

Example

In the example below the *Value Node* is used to control multiple values at once, this make the node a useful organizational tool.

Tip: From this you can also make different values proportional to each other by adding a *Math Node* in between the different links.

Output Nodes

These nodes are used to output the composited result in some way.

Composite Node

The Composite node is where the actual output from the Compositor is connected to the renderer. This node is updated after each render, but also reflects changes in the node-tree (provided at least one finished input node is connected).

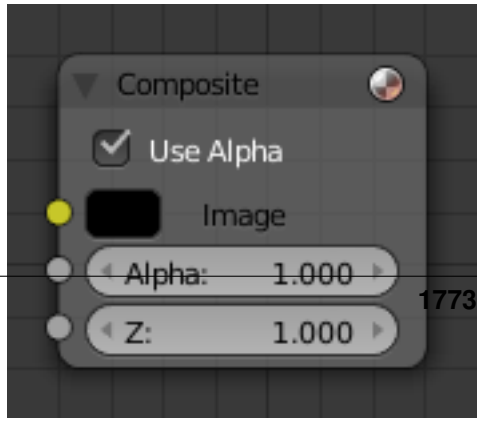


Fig. 2.2173: Example of the *Value Node*.

Inputs

Connecting a node to the Composite node will output the result of the prior tree of that node to the Compositor.

Image RGB image. The default is black, so leaving this node unconnected will result in a blank image.

Alpha Alpha channel.

Z Z-depth.

Properties

Use Alpha Premultiplied or straight.

Outputs

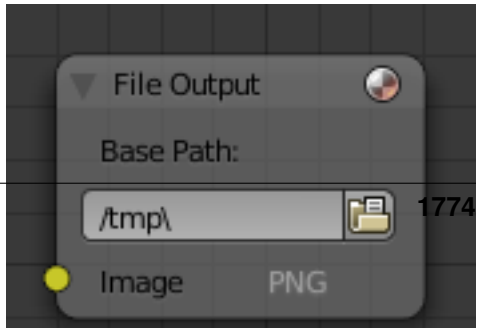
This node has no output sockets.

Note: If multiple Composite nodes are added, only the active one (last selected, indicated with a slightly darker header) will be used.

File Output Node

This node writes out an image, for each frame range specified, to the filename entered, as part of a frameset sequence.

This node can be used as a way to automatically save the image after a render; In addition, since this node can be hooked in anywhere in the node tree, it can also save intermediate images automatically.



Inputs

Image The image(s) will be saved on rendering, writing to the current frame. An entire sequence of images will be saved, when an animation is rendered.

Note: To support subsequent arrangement and layering of images, the node can supply a Z-depth map. However, please note that only the OpenEXR image formats save the Z information.

Properties

Base Path Unlike the render output filepath, this node uses a base directory and an image name, by default the output path is composed of: `{base path}/{file name}{frame number}.{extension}`.

Besides being split into two settings, in all other respects, this setting is treated the same as the *render output path*.

File Format label Shows the selected File Format.

Note: More options could be set in the properties region.

Outputs

This node has no output sockets.

Levels Node

The Levels Node read the inputs color channels and outputs analytical values.

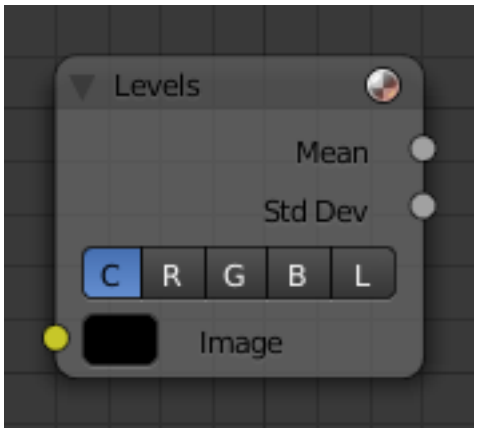


Fig. 2.2176: Levels Node.

Inputs

Image Standard image input.

Properties

Channel C (Combined RGB), R (Red), G (Green), B (Blue), L (Luminance)

Outputs

1D values based on the levels of an image.

Mean The mean is the average value of all image pixels in specified channel (combined, red, green, blue, luminance). It tells you how dark or bright the image is and can be used as such for setups that depend on how is input "bright" or "dark".

Standard deviation How much those pixel values differ from the mean. A low standard deviation indicates that the pixel values tend to be very close to the mean. A high standard deviation indicates that the values are spread out over a large range of values.

The visualization of such data is just a gray rectangle.

Split Viewer Node

The *Split Viewer* node takes two images and displays these side-by-side as backdrop or as a Viewer Node output.

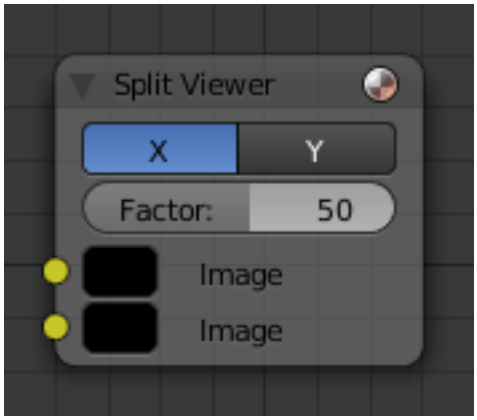


Fig. 2.2177: Split Viewer Node.

Inputs

Image Shown on the right or top half set by the axis.

Image And respectively the left or bottom half.

Properties

Axis X or Y used as the split axis.

Factor Percentage factor setting the space distribution between the two images.

Outputs

This node has no output sockets.

Hint: This node could be used to plan scene transitions by comparison of the end frame of one scene with the start frame of another to make sure that they align.

Examples

Viewer Node

The *Viewer* node is a temporary, in-process viewer. It could be plug in anywhere to inspect an image or value-map in your node-tree.

Select a view node with `LMB` to switch between multiple view nodes. It is possible to automatically plug a Viewer node to any other node by pressing `Shift-Ctrl-LMB` on it.

Fig. 2.2179: Viewer Node.

Inputs

see *Composite Node*.

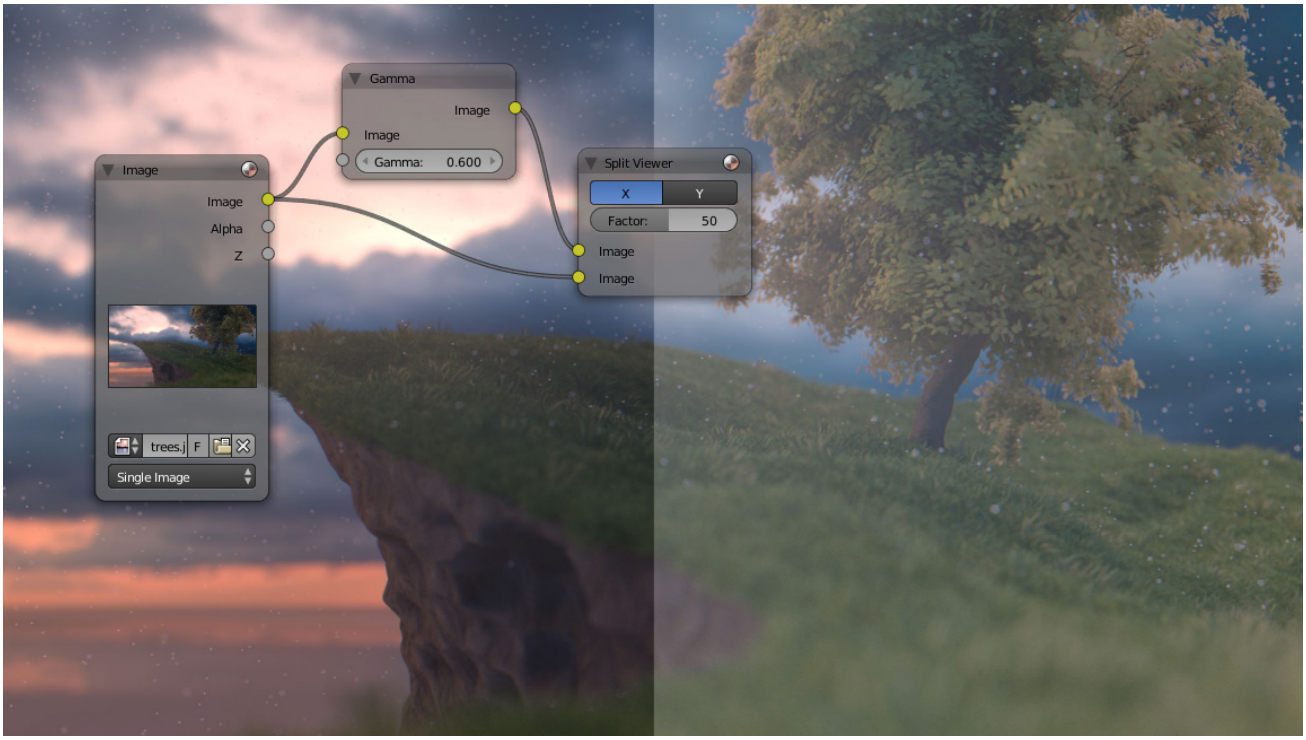


Fig. 2.2178: Example of Split Viewer node.

Properties

Tile order The tile order can be defined for the backdrop image, using the *Tile order* field in the properties of the viewer node (*Properties* panel in Properties region, with the viewer node selected):

Rule of thirds Calculates tiles around each of the nine zones defined by the *rule of thirds* .

Bottom up Tiles are calculated from the bottom up.

Random Calculates tiles in a non-specific order.

Center Calculates the tiles around a specific center, defined by X and Y fields.

X, Y

Outputs

This node has no output sockets.

Note: It is possible to add multiple Viewer nodes, though only the active one (last selected, indicated with a slightly darker header) will be shown on the backdrop or in the UV/Image editor.

Using the UV/Image Editor

The viewer node allows results to be displayed in the UV/Image Editor. The image is facilitated in the header by selecting *Viewer Node* in the linked *Image* data-block menu. The UV/Image Editor will display the image from the currently selected viewer node.

To save the image being viewed, use *Image* → *Save As Image*, F3 to save the image in a file.

The UV/Image Editor also has three additional options in its header to view Images with or without Alpha, or to view the Alpha or Z itself. Holding LMB in the Image display allows you to sample the values.

Color Nodes

These nodes adjust the image's colors, for example increasing the contrast, making it warmer, overlaying another image, etc.

Alpha Over Node

Use this node to layer images on top of one another. Alpha Over does not work on the colors of an image.

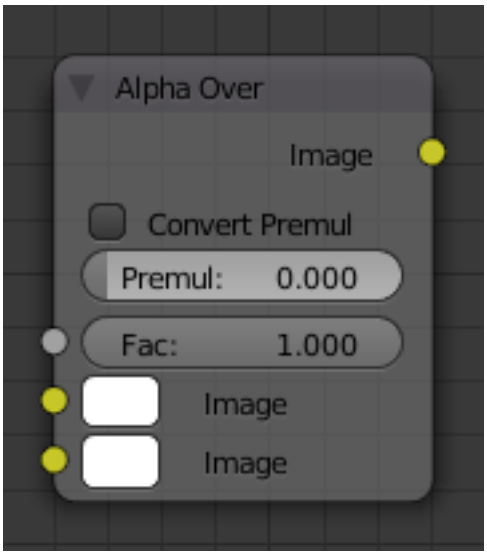


Fig. 2.2180: Alpha Over Node.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image The background image.

Image The foreground image. Where the image pixels has an alpha greater than 0, the background image will be overlaid.

Properties

Convert Premultiplied *Strange Halos or Outlines*.

Premultiply Mix Factor. See *Alpha Channel*.

Outputs

Image Standard image output.

Strange Halos or Outlines

This section clarifies the functionality of premultiplied-alpha button. An alpha channel has a value of between 0 and 1. To make an image transparent (to composite it over another one), the RGB pixel values are multiplied by the alpha values (making the image transparent (0) where the alpha is black (0), and opaque (1) where it is white (1)).

To composite image A over image B, the alpha of image A gets multiplied by image A, thus making the image part of A opaque and the rest transparent. Then the alpha channel of A is inverted and multiplied by image B, thus making image B transparent, where A is opaque and vice versa. To get the final composite the resultant images are added.

A pre-multiplied alpha is, when the image (RGB) pixels are already multiplied by the alpha channel, therefore, the above compositing operation does not work too well, and *Convert Premultiplied* has to be enabled. This is only an issue in semi-transparent area and edges usually. The issue normally occurs in a node setup, in which two images previously combined with alpha, then are combined again with yet another image. The previously combined image was already multiplied (pre-multiplied) and needs to be converted as such (hence, *Convert PreMul*).

If multiplied twice artifacts like a white or clear halo occur around where the image meet, since the alpha value is being squared or cubed. It also depends on whether or not the image has been rendered as a premultiplied, or as a straight RGBA image.

Examples

In this example, an image of a Cube is superimposed on a cliff background. Use the PreMultiply button, when the foreground image and background images have a combined Alpha that is greater than 1.00; otherwise, you will see an unwanted halo effect. The resulting image is a composite of the two source images.

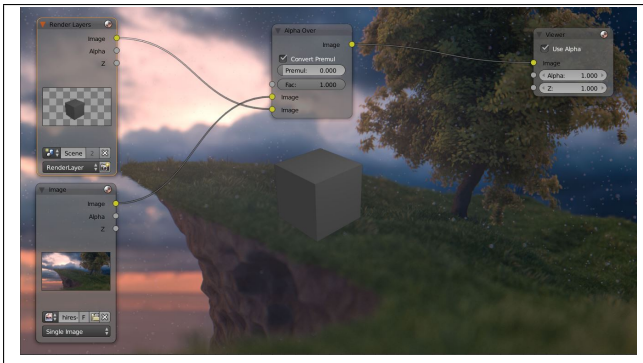


Fig. 2.2181: Assembling a composite Image using Alpha Over.

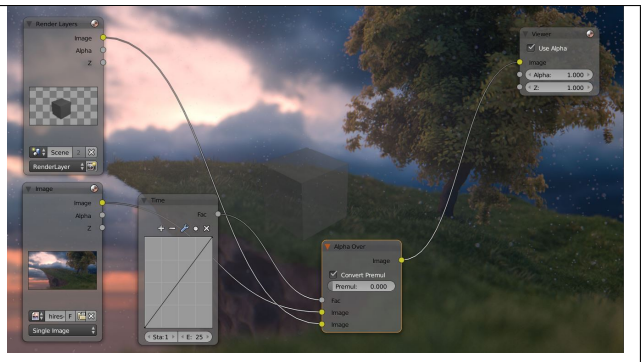


Fig. 2.2182: Animated See-Through/Sheer SFX using Alpha Over on Frame 11.

In this example, we use the Factor control to make a sheer cloth or onion-skin effect. This effect can be animate, allowing the observer to "see-through" walls (or any foreground object) by hooking up a Time node to feed the Factor socket as shown below. In this example, over the course of 30 frames, the Time node makes the Alpha Over node produce a picture that starts with the background cliff image, and slowly bleeds through the cube. This example shows frame 11 just as the cube starts to be revealed.

Bright/Contrast Node

Inputs

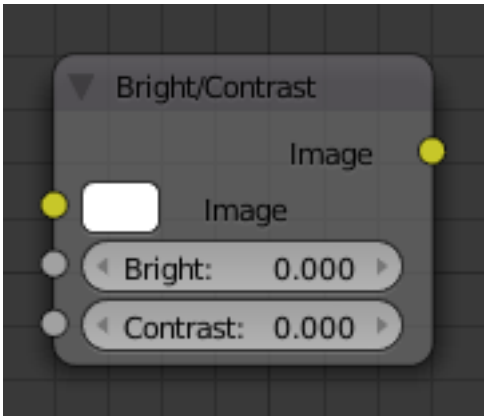


Fig. 2.2183: Bright/Contrast Node.

Image Standard image input.

Bright A multiplier-type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast A scaling type factor by which to make brighter pixels brighter, but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

Properties

This node has no properties.

Outputs

Image Standard image output.

Notes

It is possible that this node will put out a value set that has values beyond the normal range, i. e. values greater than one and less than zero. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a Color Ramp node (with all normal defaults).

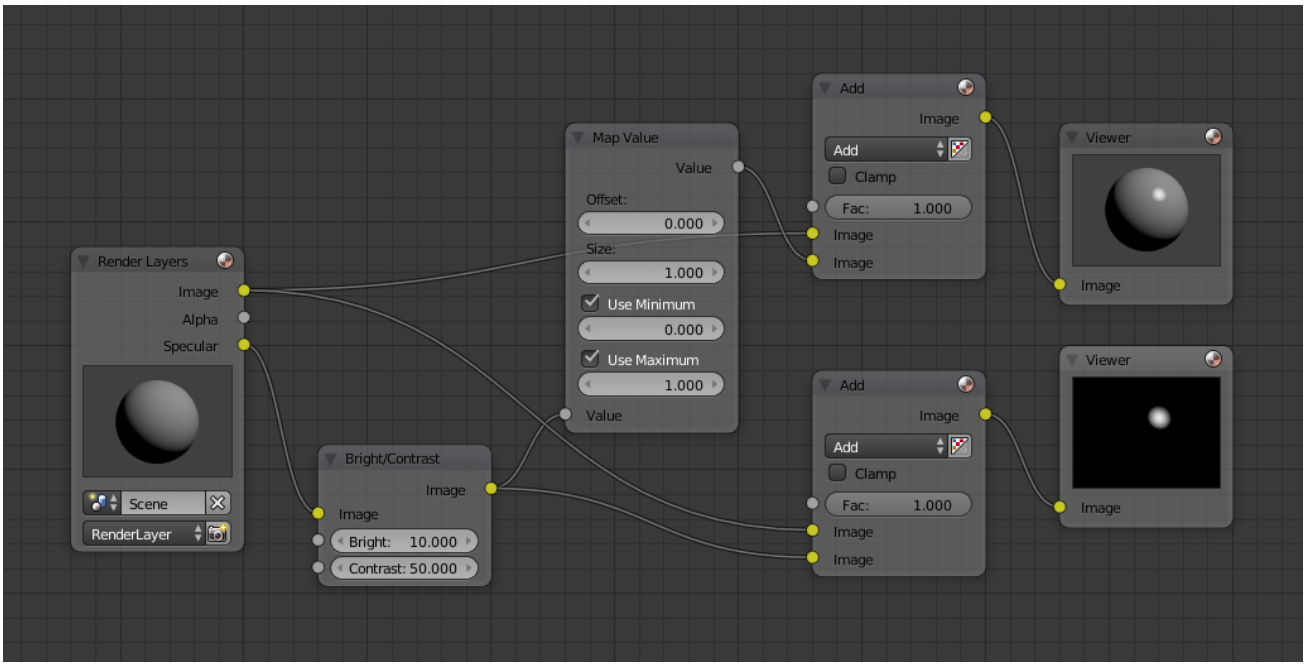


Fig. 2.2184: Clamp the values to normal range.

Either of these nodes will scale the values back to normal range. In the example image, we want to amp up the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has valued much less than one in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the Color Ramp node produces the desired effect.

Example

Color Balance Node

The Color Balance node can adjust the color and values of an image.

Inputs

Factor Controls the amount of influence the node exerts on the output image

Color Standard image input.

Properties

Two different correction formulas could be selected.

Lift/Gamma/Gain

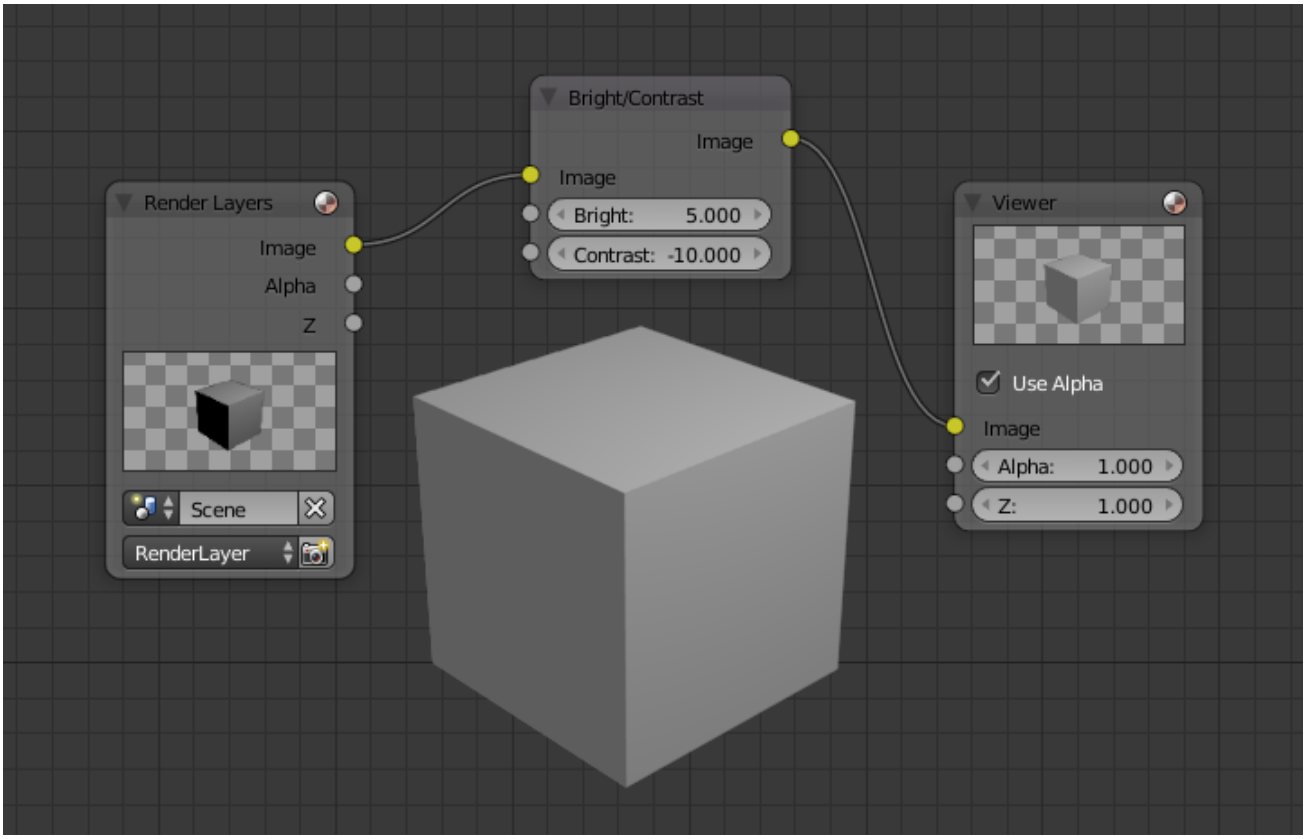


Fig. 2.2185: A basic example.

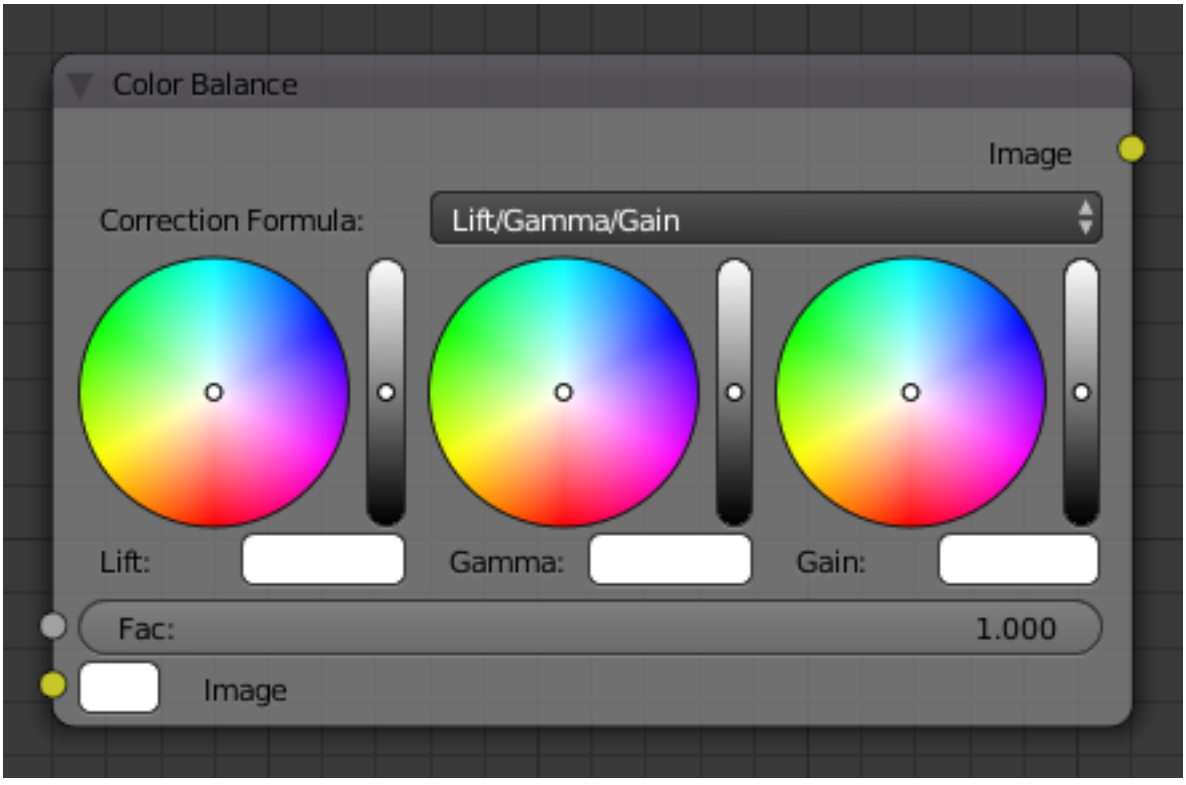


Fig. 2.2186: Bright/Contrast Node.

Lift Increases the value of dark colors.

Gamma Will adjust midtones.

Gain Adjusts highlights.

Offset/Power/Slope (ASC-CDL)

Offset Summand.

Power Over all exponent.

Slope Multiplier.

Outputs

Color Standard output image.

Advanced

The Offset/Power/Slope formula

$out = (i \times s + o)^p$

where:

- *out*: The color graded pixel code value.
- *i*: The input pixel code value (0 to 1) (black to white).
- *s*: Slope (any number 0 or greater, nominal value is 1.0).
- *o*: Offset (any number, the nominal value is 0).
- *p*: Power (any number greater than 0, nominal value is 1.0).

Color Correction Node

The Color Correction node can adjust the color of an image, separately in several tonal ranges (highlights, midtones and shadows) and only affect the necessary RGB channels.

Properties

Red, Green, Blue Specifies which RGB-channels will be affected by correction.

Correction tools (columns)

Saturation Adjusts the image's saturation.

Contrast Adjust image contrast.

Gamma Exponential gamma correction, affecting the midtones of the image. (Works like Power in the Color Balance node.).

Gain Multiplier, stronger influence on the highlights. (Works like Slope in the Color Balance node).

Lift This value (can be negative) will be added (+), linear lightens or darkens the image. (Works like *Offset* in the Color Balance node).

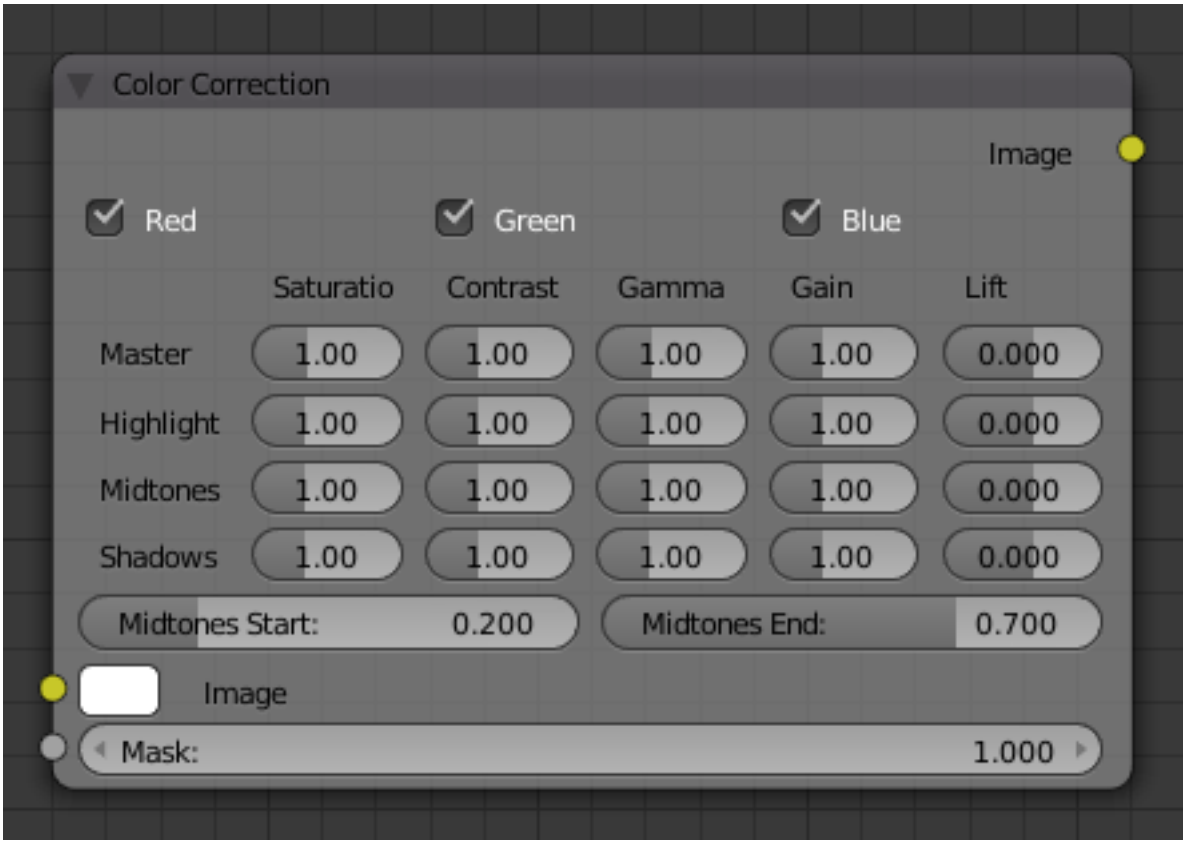


Fig. 2.2187: Color Balance Node.

Tonal ranges (rows)

Master these sliders affect the entire tonal range.

Highlights these sliders only affect the highlights.

Midtones these sliders only affect the midtones.

Shadows Affects the dark tones of an image often affecting the shadows.

Midtones Start, Midtones End Defines the start and the end of midtones range, i.e. values where the whole tonal range is divided into the highlights, midtones and shadows. (There is also a smooth transition between the ranges of width 0.2 units.)

Inputs

Image Standard image input.

Mask Controls the amount of influence the node exerts on the output image.

Outputs

Color Standard image output.

Gamma Node

Use this node to apply a gamma correction.

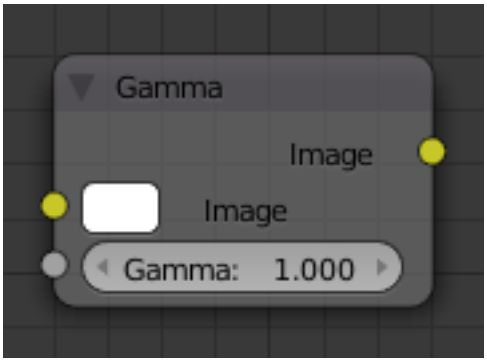


Fig. 2.2188: Gamma Node.

Inputs

Image Standard image input.

Gamma An exponential brightness factor.

Properties

This node has no properties.

Outputs

Image Standard image output.

Examples

Fig. 2.2189: Example of Gamma node.

Hue Correct Node

The Hue Correct node is able to adjust the Hue, Saturation, and Value of an image, with an input curve.

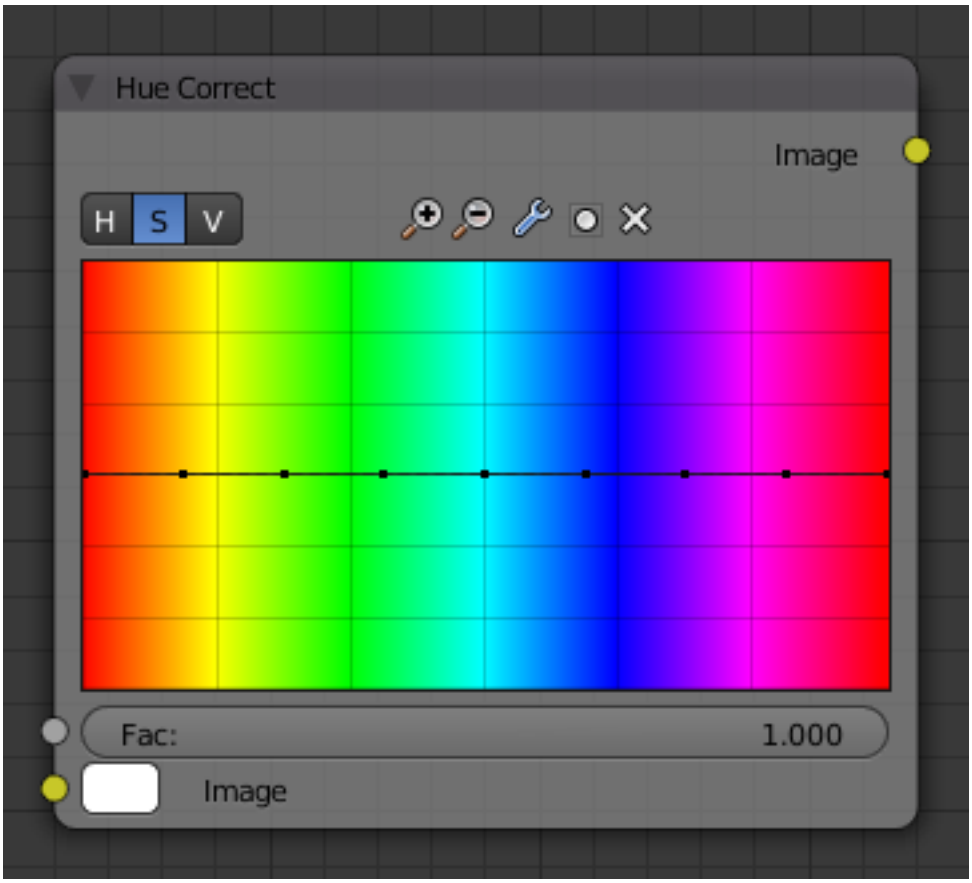


Fig. 2.2190: Color Balance Node.

Inputs

Factor Controls the amount of influence the node exerts on the output image

Image Standard image input.

Properties

Level H (Hue), S (Saturation), V (Value)

Curve For the curve controls see: *Curve widget*. By default, the curve is a straight line, meaning there is no change. The spectrum allows you to raise or lower HSV levels for each range of pixel colors. To change a H, S, or V level, move the curve points up or down. Pixels with hue values each point in the horizontal position of the graph will be changed depending on the shape of the curve.

Outputs

Image Standard image output.

Hue Saturation Value Node

This node applies a color transformation in the HSV color space. Called "Hue Saturation Value" in shader and texture context.

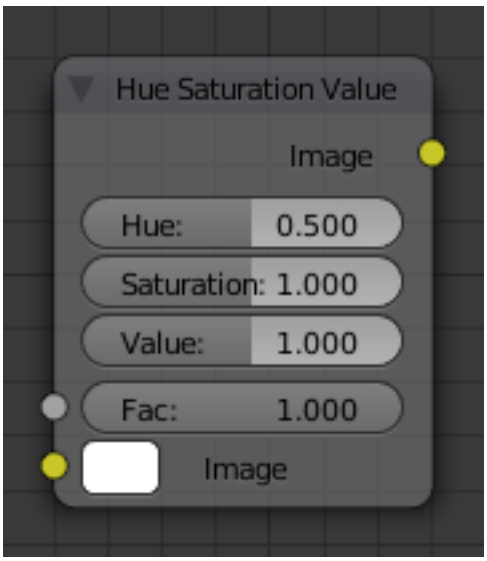


Fig. 2.2191: Hue Saturation Node.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Properties

The transformations are relative shifts. In the shader and texture context the following properties are available as input sockets.

Hue Specifies how the hue rotation of the image. 360° are mapped to (0 to 1). The hue shift of 0 (-180°) and 1 (+180°) have the same result.

Saturation A saturation of 0 removes hues from the image, resulting in a grayscale image. A shift greater 1.0 increases saturation.

Value Value is the overall brightness of the image. De/Increasing values shift an image darker/lighter.

Outputs

Image Standard image output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are vice versa A blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (recall that white, minus blue, equals yellow). A yellow image, with a Hue setting at 0 or 1, is blue.

Hue and Saturation work together. So, a Hue of 0.5 keeps the blues the same shade of blue, but *Saturation* can deepen or lighten the intensity of that color.

Gray & White are neutral hues A gray image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with *Value*. This applies to all shades of gray, from black to white; wherever the values are equal.

Changing the effect over time The Hue and Saturation values can be animated with a *Time Node* or by animating the property.

Note: Tinge

This HSV node simply shifts hues that are already there. To colorize a gray image, or to add a tint to an image, use a mix node to add in a static color from an RGB input node with your image.

HSV Example

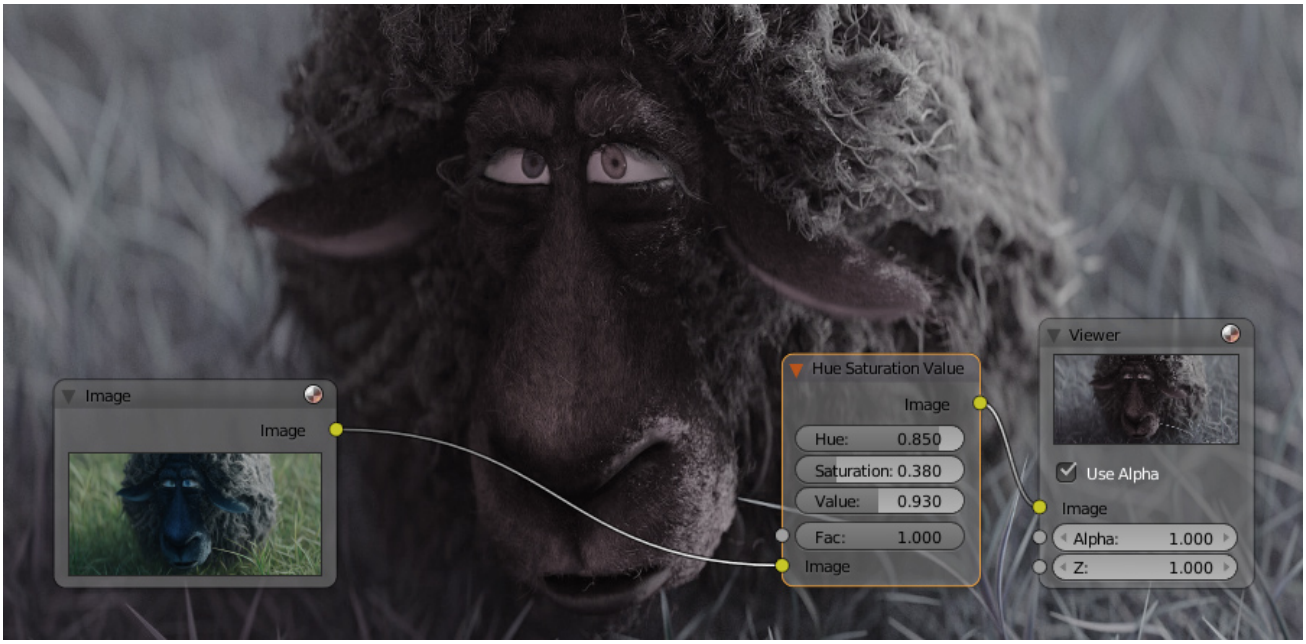
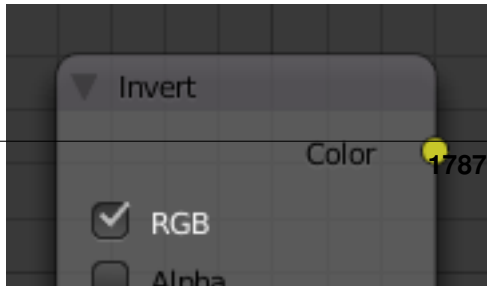


Fig. 2.2192: A basic example.

Invert Node

This node inverts the colors in the input image, producing a negative.



Inputs

Factor Controls the amount of influence the node exerts on the output image.

Color Standard image input.

Properties

In the compositing context this node has the following properties.

RGB De/activation of the color channel inversion.

Alpha De/activation of the alpha channel inversion.

Outputs

Color Standard image output.

Mix Node

This node mixes images by working on the individual and corresponding pixels of the two input images. Called "MixRGB" in the shader and texture context.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image The background image. The image size and resolution sets the dimensions of the output image.

Image The foreground image.

Properties

Mix The Blend types could be selected in the select menu. See *Color Blend Modes* for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Dodge, Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Use Alpha If activated, by clicking on the *Color and Alpha* icon, the Alpha channel of the second image is used for mixing. When deactivated, the default, the icon background is a light gray. The alpha channel of the base image is always used.

Clamp Limit the highest color value to not exceed 1.

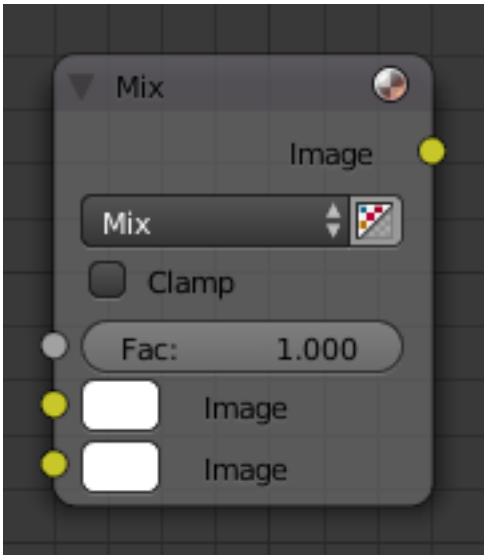


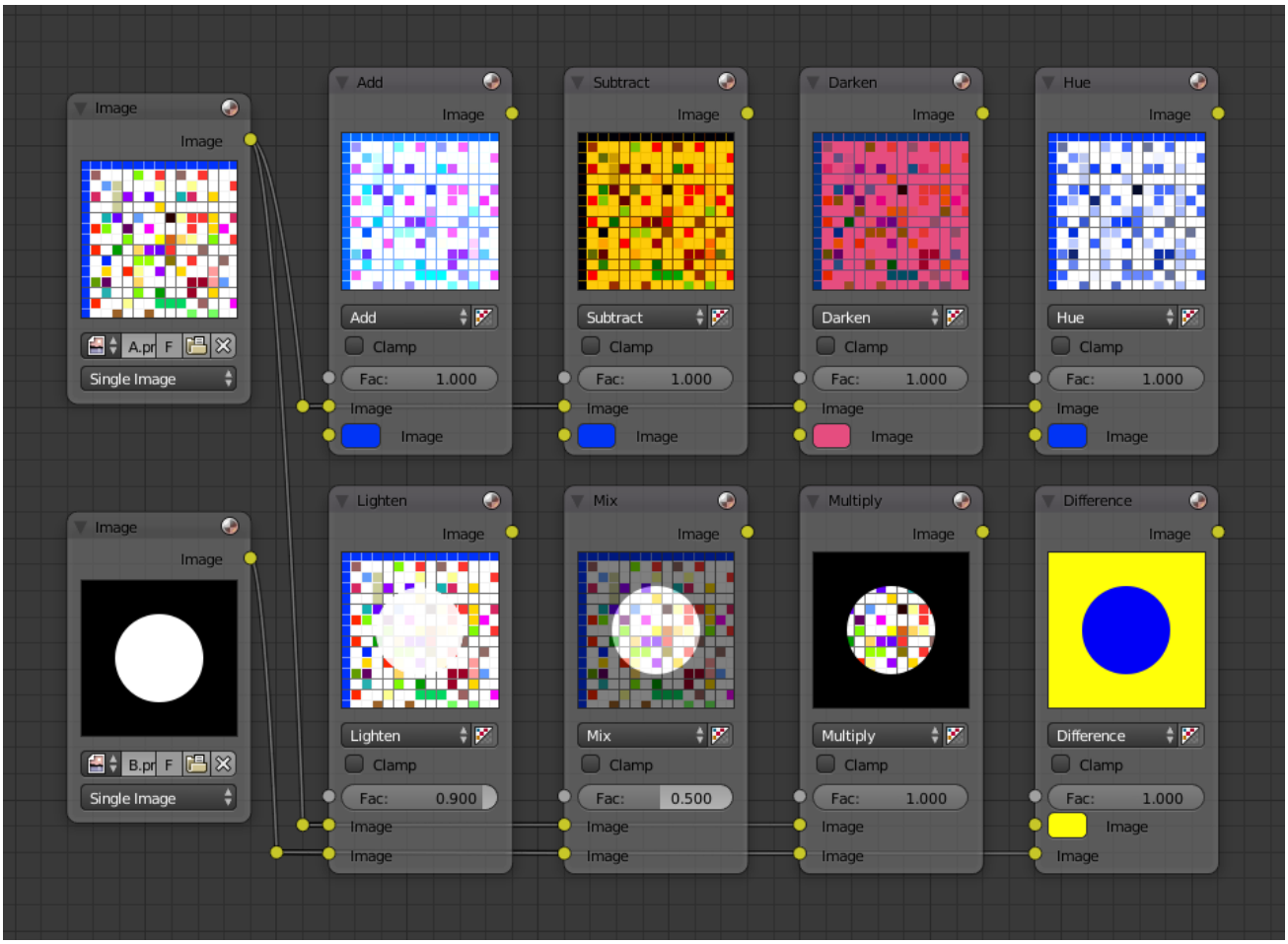
Fig. 2.2194: Mix Node.

Outputs

Image Standard image output.

Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.



Some explanation of the mixing methods above might help you use the Mix node effectively:

Add adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.

Subtract Taking Blue away from white leaves Red and Green, which combined make Yellow. Taking Blue away from Purple leaves Red. Use this to desaturate an image. Taking away yellow makes an image bluer and more depressing.

Multiply Black (0.00) times anything leaves black. Anything times White (1.00) is itself. Use this to mask out garbage, or to colorize a black-and-white image.

Hue Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').

Mix Combines the two images, averaging the two.

Lighten Like bleach makes your whites whiter. Use with a mask to lighten up a little.

Difference Kinda cute in that it takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two verrry similar images to see what had been done to one to make it the other; sorta like a change log for images. You can use this to see a watermark (see *Watermark images*) you have placed in an image for theft detection.

Darken with the colors set here, is like looking at the world through rose-colored glasses.

Contrast Enhancement

Here is a small map showing the effects of two other common uses for the RGB Curve: *Darken* and *Contrast Enhancement*. You can see the effect each curve has independently, and the combined effect when they are *mixed* equally.

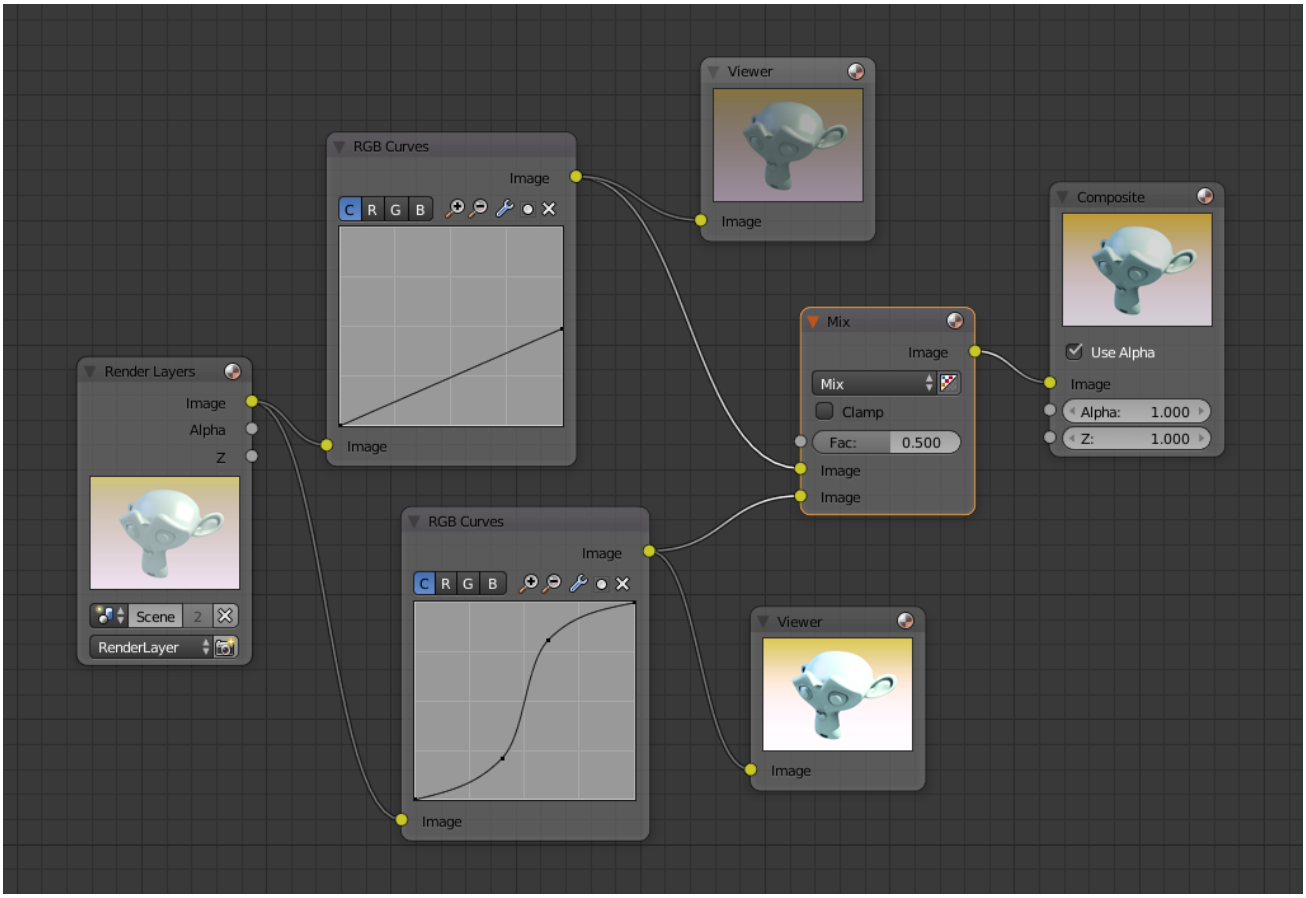


Fig. 2.2195: Example node setup showing “Darken”, “Enhance Contrast” and “Mix” nodes for composition.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast.

In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB (S shaped) curve scales the output such that middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus, whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is

added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better.

Watermark images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising. Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting service, or simply to track their image's proliferation throughout the web. Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding Your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it is up to you. In the example below, we are encoding the watermark in a specific location in the image using the *Translate* node; this helps later because we only have to look at a specific location for the mark. We then use the RGB to BW node to convert the image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one-tenth of its original intensity. The Add node adds the corresponding pixels, make the ones containing the mark ever-so-slightly brighter.

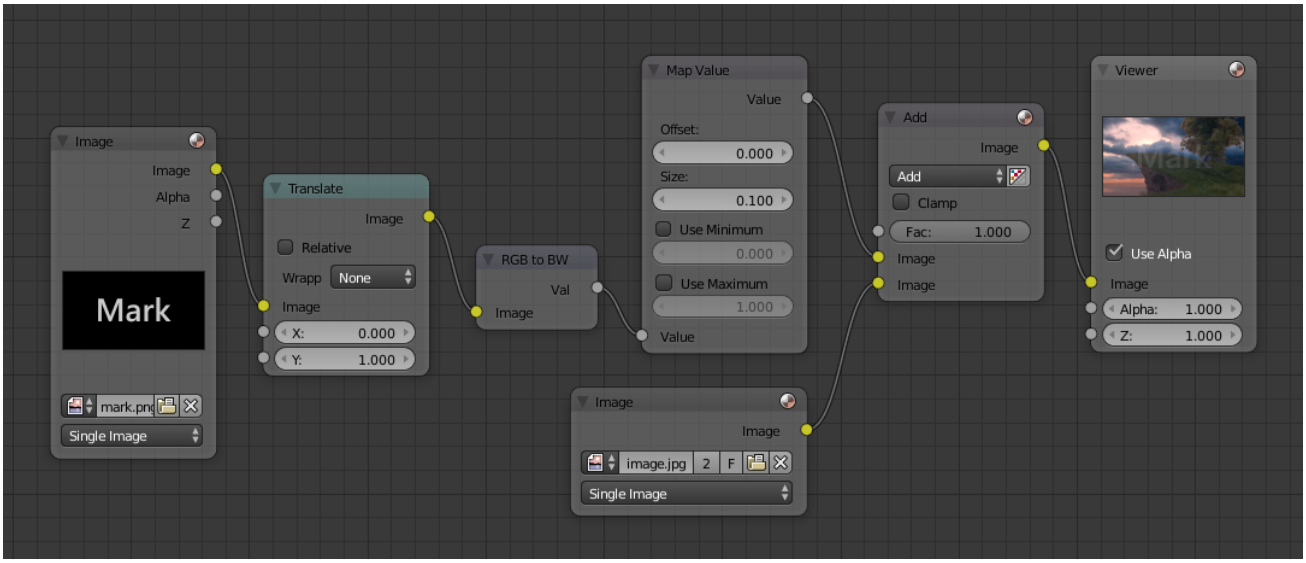


Fig. 2.2196: Embedding your mark in an Image using a Mark and Specific Position.

Of course, if you *want* people to notice your mark, do not scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Note: Additional uses

You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface,

or as a special effect to make words appear on the possessed girl's forearm.
yuk.

Decoding an Image for your Watermark

When you see an image that you think might be yours, use the node map below to compare it to your stock image (pre-watermarked original). In this map, the Mix node is set to Difference, and the Map Value node amplifies any difference. The result is routed to a viewer, and you can see how the original mark stands out, clear as a bell:

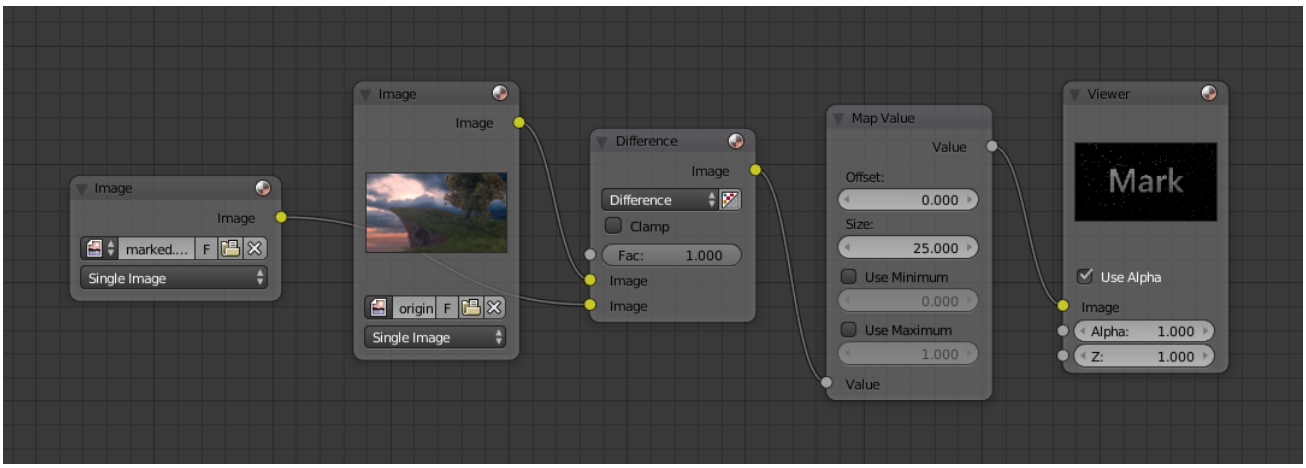
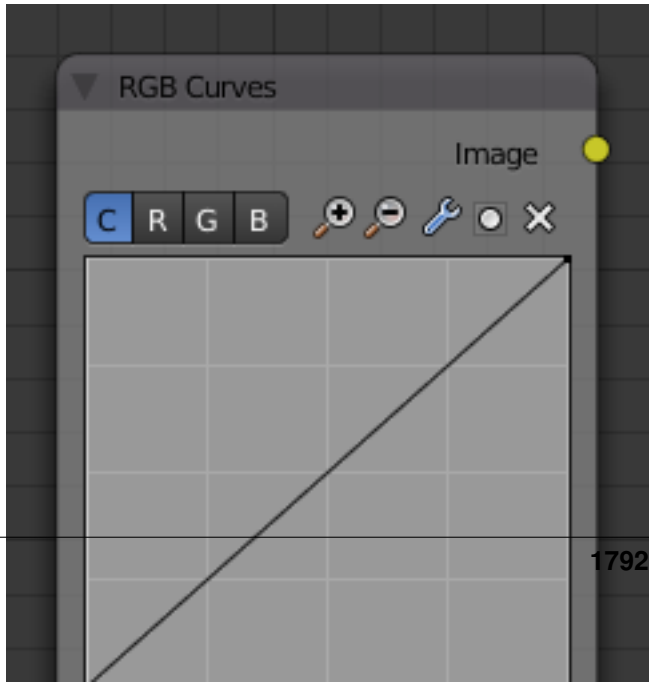


Fig. 2.2197: Checking an image for your watermark.

Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding map in one scene, and the decoding map in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to JPEG compression of 50%.

RGB Curves Node

This node allows color corrections for each color channel and levels adjustments in the compositing context.



Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Black Level Defines the input color that is (linear) mapped to black.

White Level Defines the input color that is (linear) mapped to white.

Tip: To define the levels, use the *eye dropper* to select a color sample of a displayed image.

Properties

Channel Clicking on one of the channels displays the curve for each.

C (Combined RGB), R (Red), G (Green), B (Blue), L (Luminance)

Curve A Bézier curve that varies the input levels (x-axis) to produce an output level (y-axis). For the curve controls see: *Curve widget*.

Outputs

Image Standard image output.

Examples

Here are some common curves you can use to achieve desired effects:

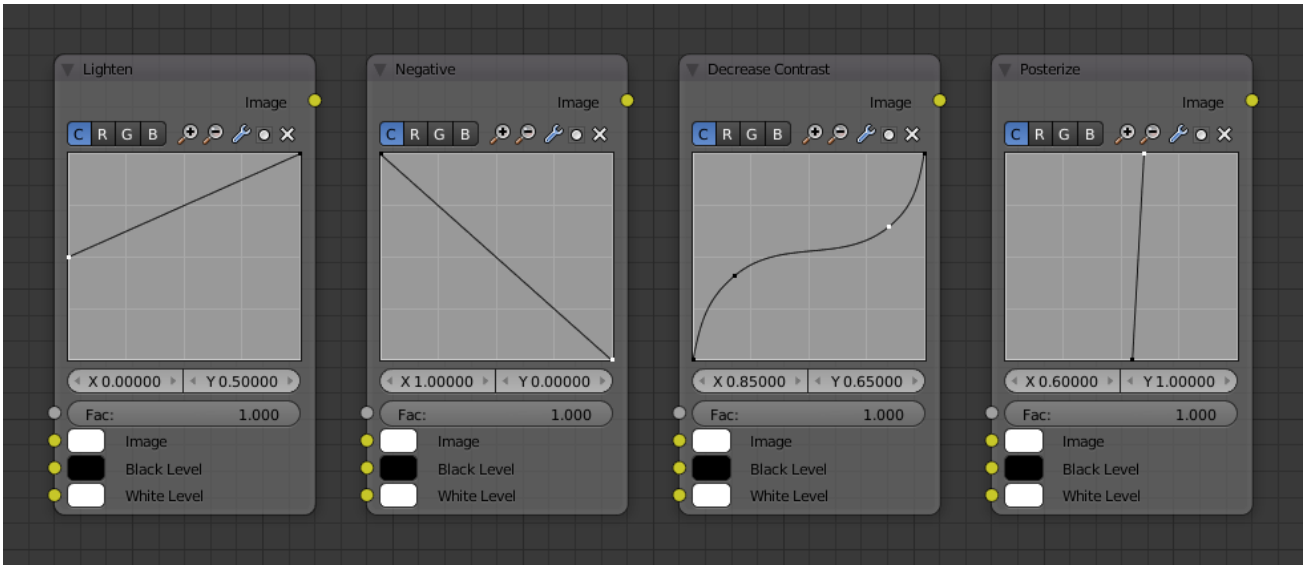


Fig. 2.2199: From left to right: 1. Lighten 2. Negative 3. Decrease Contrast 4. Posterize.

Color correction using Curves

In this example, the image has way too much red in it, so we run it through an RGB node and reduce the Red channel by about half.

We added a middle dot so we could make the line into a sideways exponential curve. This kind of curve evens out the amount of a color in an image as it reaches saturation.

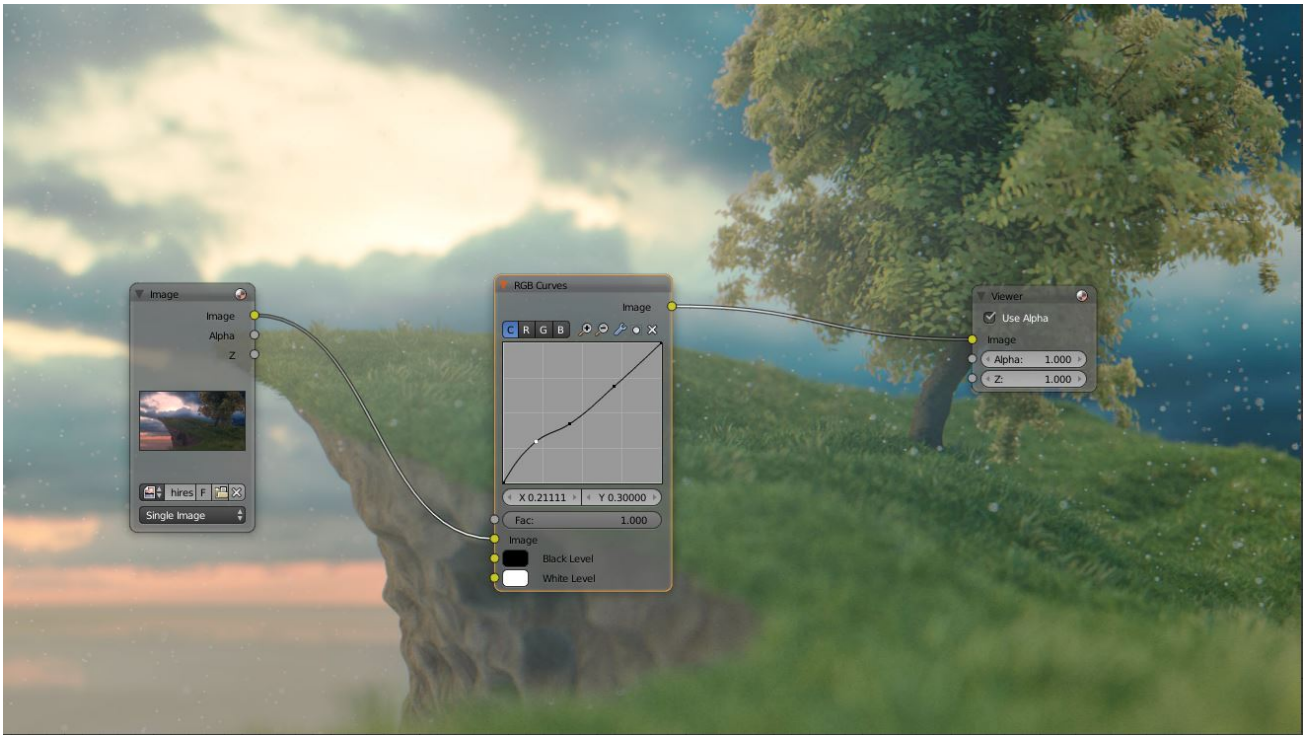


Fig. 2.2200: Color correction with curves.

Also, read on for examples of the Darken and Contrast Enhancement curves.

Color correction using Black/White Levels

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the UV/Image editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side-effect of setting Black and White Levels.

Effects

Curves and Black/White Levels can also be used to completely change the colors of an image.

Note that e.g. setting Black Level to red and White Level to blue does not simply substitute black with red and white with blue as the example image might suggest. Levels do color scaling, not substitution, but depending on the settings they can result in the described color substitution.

(What really happens when setting Black Level to pure red and White Level to pure blue is that the red channel gets inverted, green gets reduced to zero and blue remains unchanged.)

Because of this, the results of setting arbitrary Black/White Levels or RGB curves is hard to predict, but can be fun to play with.

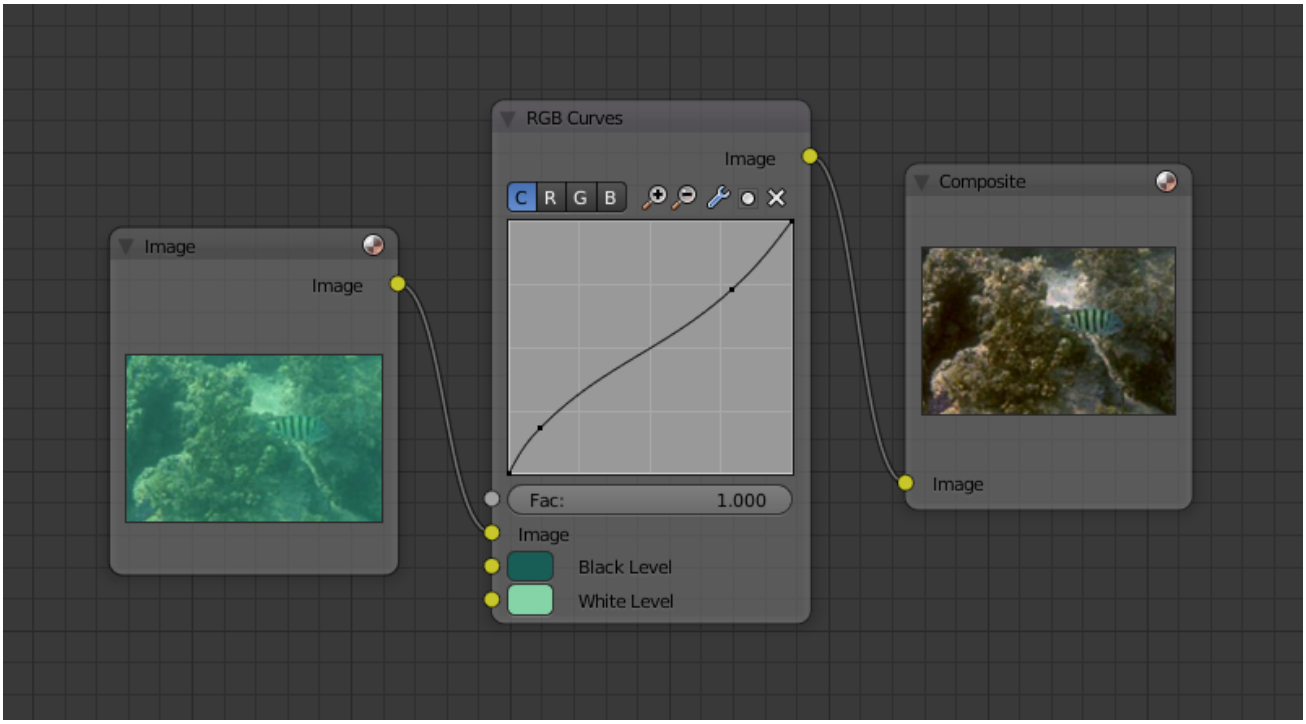


Fig. 2.2201: Color correction with Black/White Levels.

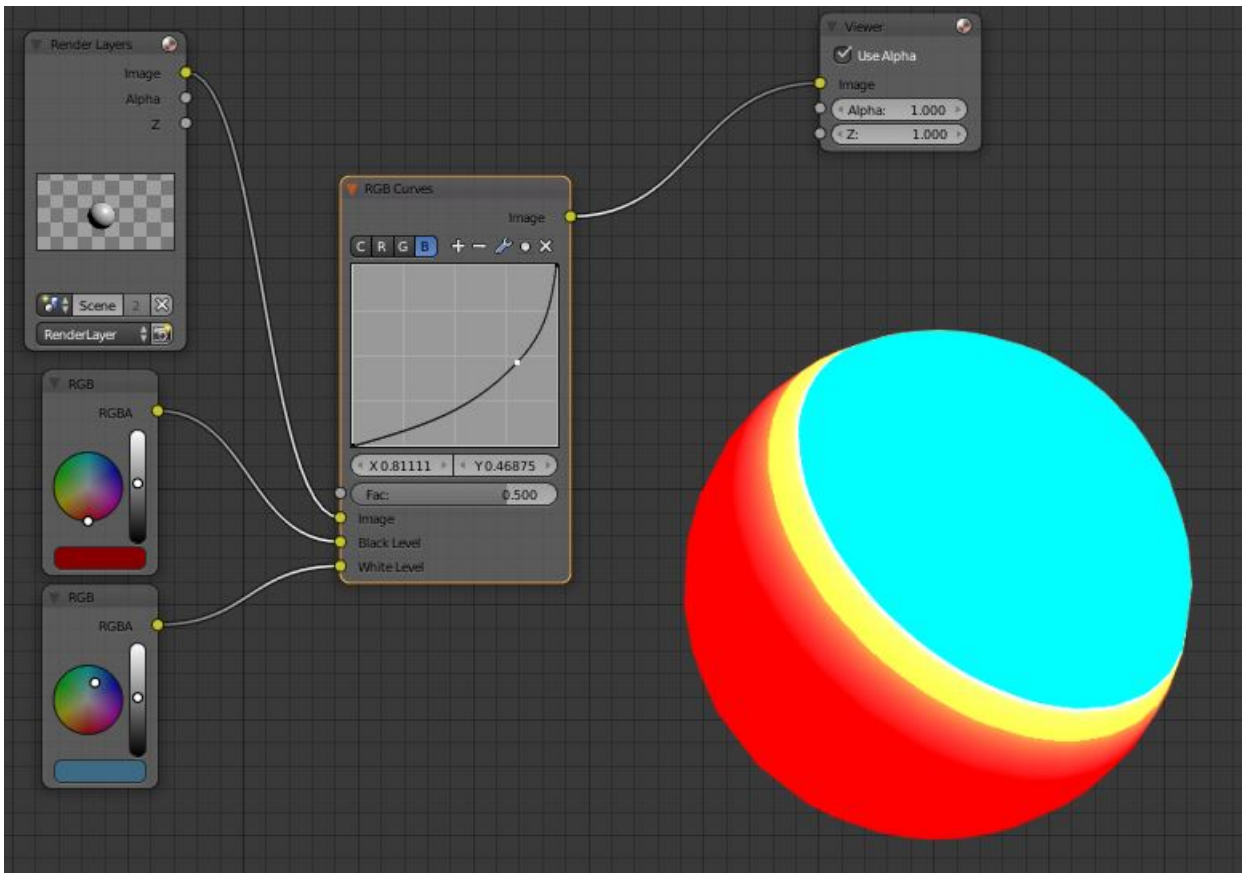


Fig. 2.2202: Changing colors.

Tone Map Node

Tone mapping is a technique used in image processing and computer graphics to map one set of colors to another in order to approximate the appearance of high dynamic range images in a medium that has a more limited dynamic range.

Essentially, tone mapping addresses the problem of strong contrast reduction from the scene values (radiance) to the displayable range, while preserving the image details and color appearance. This is important to appreciate the original scene content.

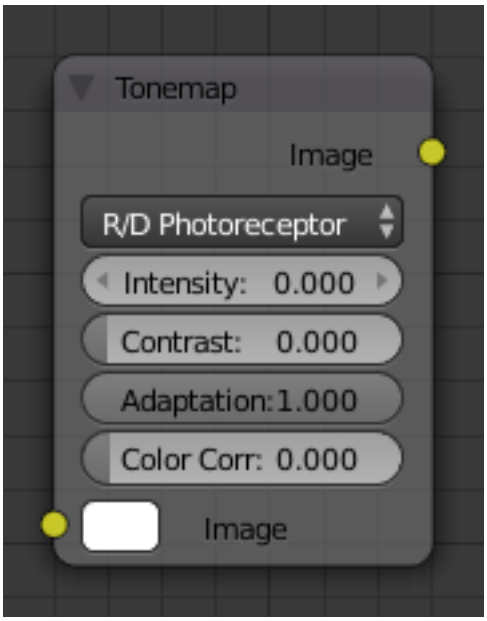


Fig. 2.2203: Tone Map Node.

Inputs

Image HDR (High Dynamic Range) image.

Properties

Type

Rh Simple

Key The value the average luminance is mapped to.

Offset Normally always 1, but can be used as an extra control to alter the brightness curve.

Gamma If not used, set to 1.

R/D Photoreceptor

Intensity If less than zero, darkens image; otherwise, makes it brighter.

Contrast Set to 0 to use estimate from input image.

Adaptation If 0, global; if 1, based on pixel intensity.

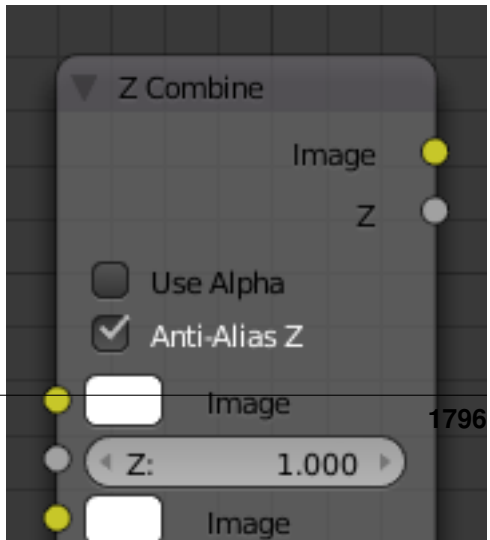
Color Correction If 0, same for all channels; if 1, each independent.

Outputs

Image LDR (Low Dynamic Range) image.

Z-Combine Node

The Z-Combine node combines two images based on their Z-depth maps. It overlays the images using the provided Z values to detect which parts of one image are in front of the other.



Inputs

Image The background image.

Z Z-depth of the background image.

Image The foreground image.

Z Z-depth of the foreground image.

Properties

Use Alpha The chosen Image pixel alpha channel is also carried over. If a pixel is partially or totally transparent, the result of the Z-Combine will also be partially transparent; in which case the background image will show through the foreground (chosen) pixel.

Anti-Alias Z Applies *Anti-Aliasing* to avoid artifacts at sharp edges or areas with a high contrast.

Outputs

Image If both Z values are equal, it uses the foreground image. Whichever Z-value is less decides which image pixel is used. See *Z-buffer*.

Z The combined Z-depth, which allows to thread multiple Z-combines together.

Examples

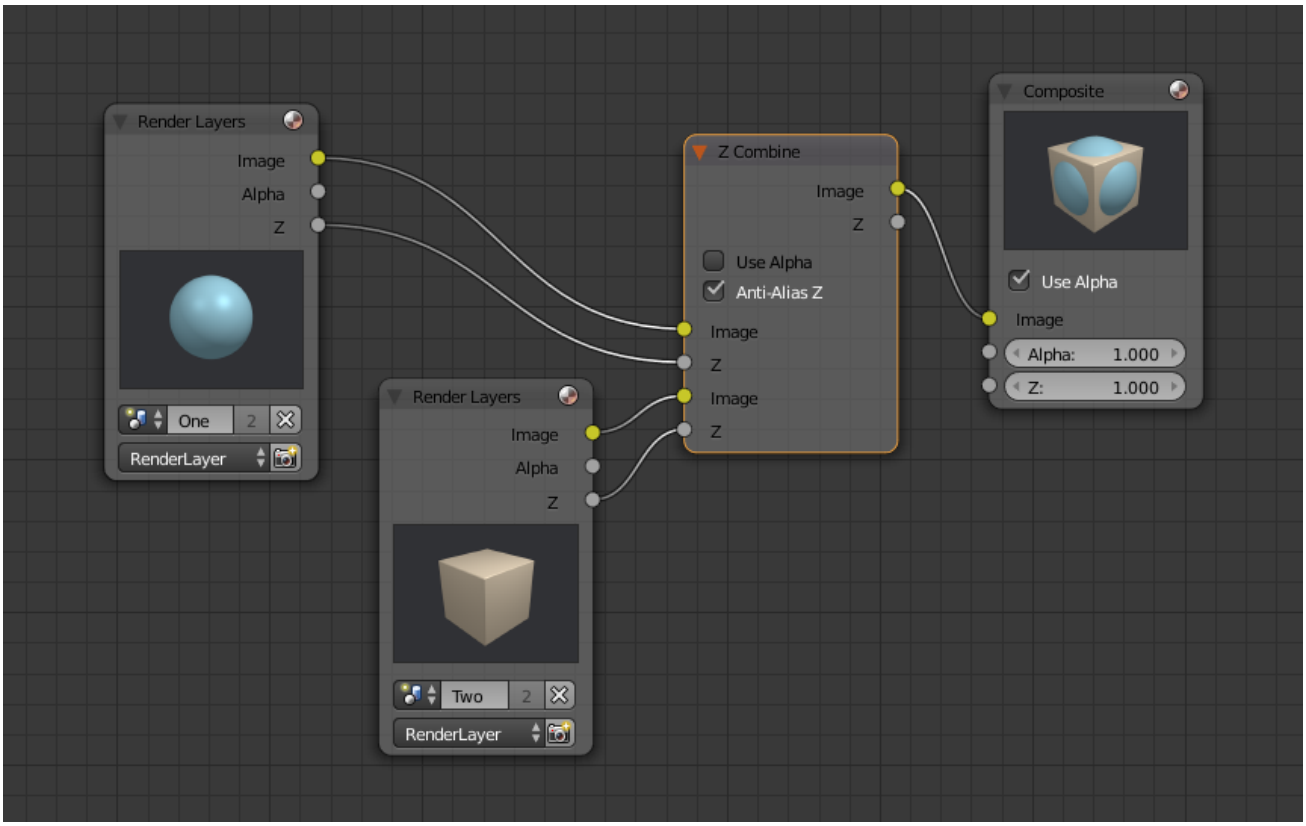


Fig. 2.2205: Choosing closest pixels.

In the example above, render output from two scenes are mixed using the Z-Offset node, one from a sphere of size 1.30, and the other a cube of size 1.00. The sphere and square are located at the same place. The cube is tipped forward, so the corner in the center is closer to the camera than the sphere surface; so Z-Offset chooses to use the cube's pixels. But the sphere is slightly larger (a size of 1.30 versus 1.00),

so it does not fit totally inside the cube. At some point, as the cube's sides recede back away from the camera, the sphere's sides are closer. When this happens, Z-offset uses the sphere's pixels to form the resulting picture.

This node can be used to combine a foreground with a background matte painting. Walt Disney pioneered the use of multi-plane mattes, where three or four partial mattes were painted on glass and placed on the left and right at different Z positions; minimal camera moves to the right created the illusion of depth as Bambi moved through the forest.

Note: Valid Input

Z Input Sockets do not accept fixed values; they must get a vector set (see Map Value node). Image Input Sockets will not accept a color since it does not have UV coordinates.

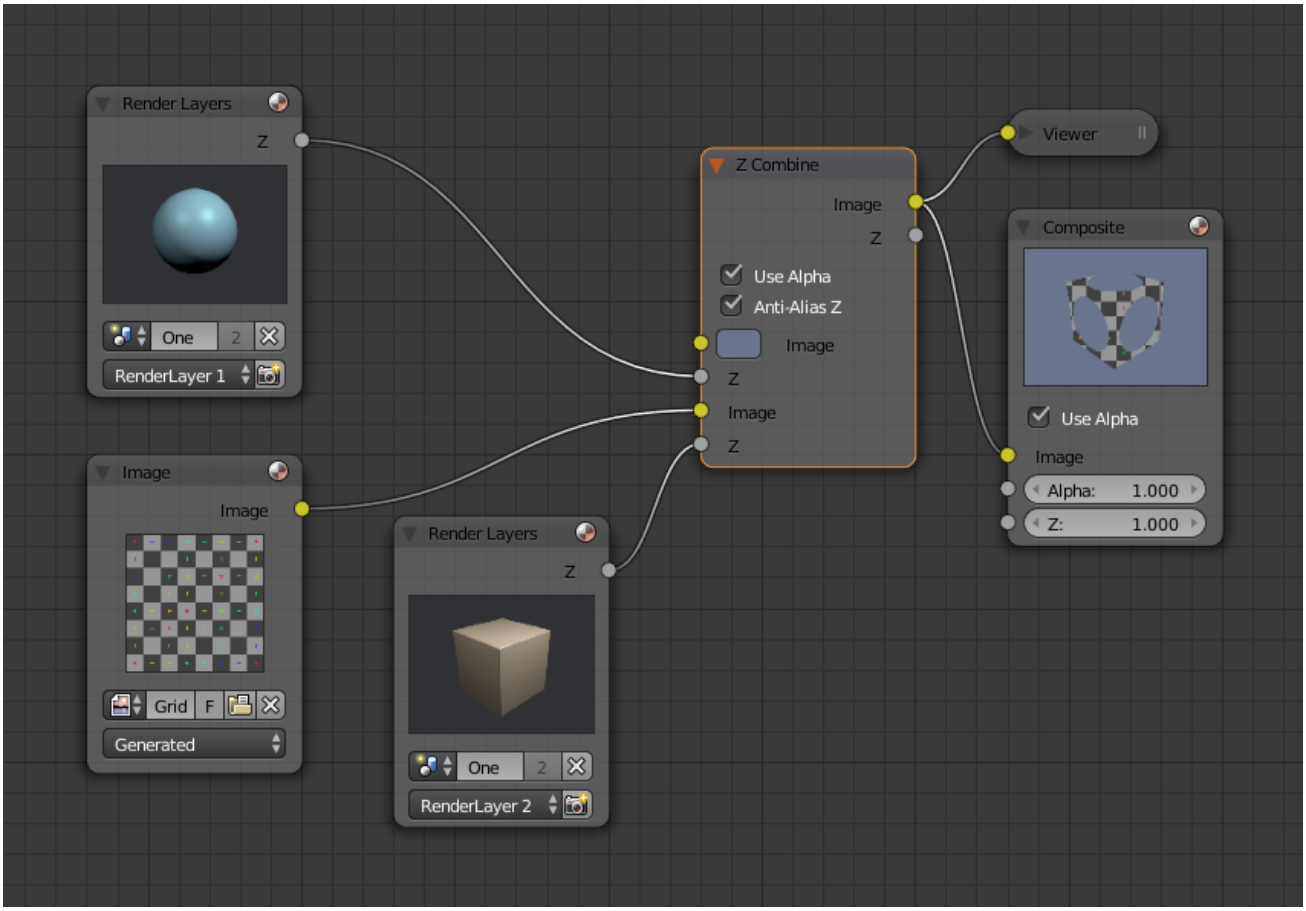


Fig. 2.2206: Mix and Match Images.

The Z-Combine can be used to merge two images as well, using the Z-values put out by two render layers. Using the Z-values from the sphere and cube scenes above, but threading different images, yields the example to the right.

In this node setup a render scene is mixed with a flat image. In the side view of the scene, the purple cube is 10 units away from the camera, and the gray ball is 20. The 3D cursor is about 15 units away from the camera. The image is Z-in at a location of 15, thus inserting it in-between the cube and the ball. The resulting image appears to have the cube on the table.

Note: Invisible Man Effect

If a foreground image with a higher Alpha than the background, is then mixed in the Z-combine with a slightly magnified background, the outline of the transparent area will distort the background, enough to make it look like seeing a part of the background through an invisible yet Fresnel-lens object.

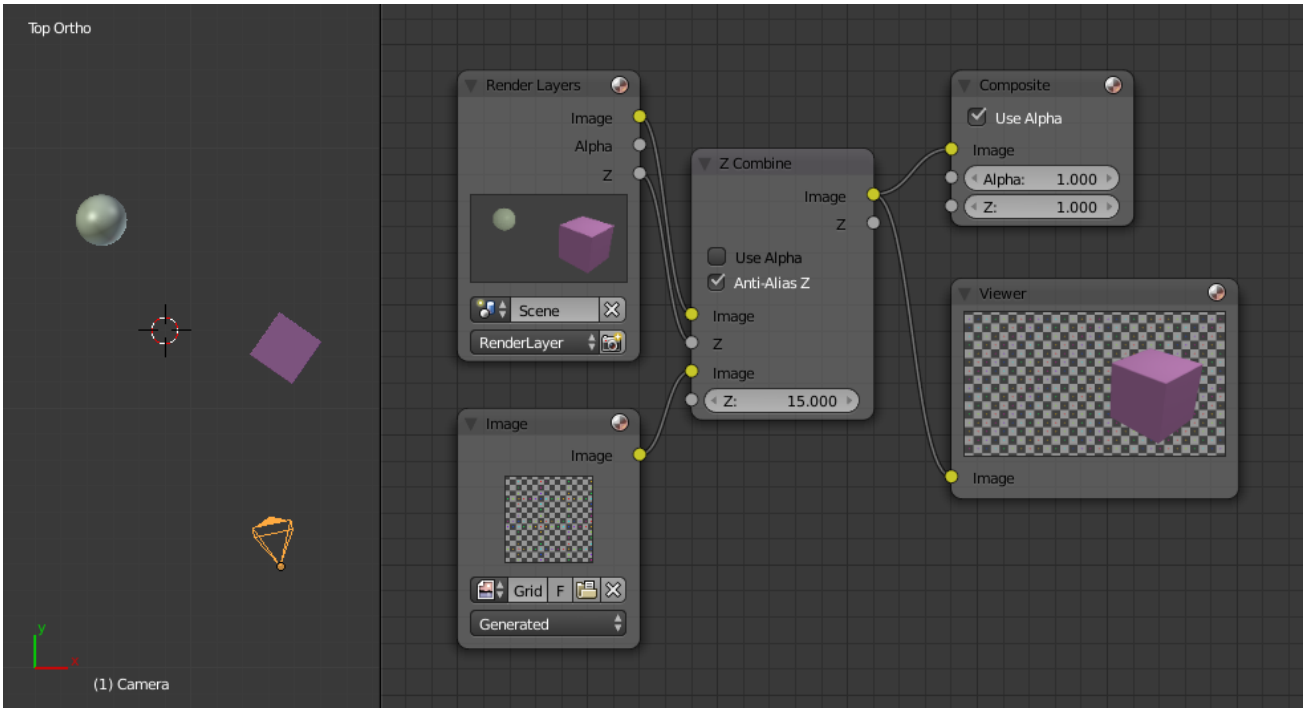


Fig. 2.2207: Z-Combine in action.

Converter Nodes

As the name implies, these nodes convert the colors or other properties of various data (e.g. transparency) in some way.

They also split out or re-combine the different color channels that make up an image, allowing you to work on each channel independently. Various color channel arrangements are supported, including traditional RGB, HSV and High Definition Media Interface (HDMI) formats.

Alpha Convert Node

This node converts the alpha channel interpretation of an image.

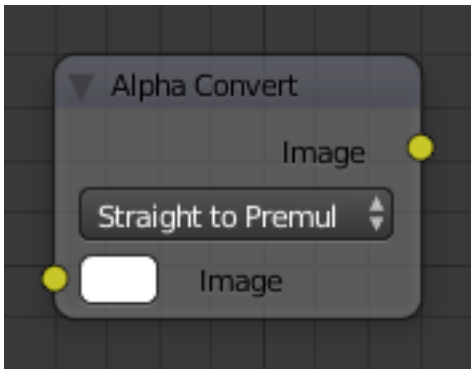


Fig. 2.2208: Alpha Convert Node.

Inputs

Image Standard image input.

Properties

Mapping

Straight to Premul, Premul to Straight Conversion in both directions. Premul. stands for Premultiplied. For details on the difference between both way to store alpha values see *Alpha Channel*.

Outputs

Image Standard image output.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors with the use of a gradient.

Inputs

Factor The Factor input is used as an index for the color ramp.

Properties

Color Ramp For controls see *Color Ramp Widget*.

Outputs

Image Standard image output.

Alpha Standard alpha output.

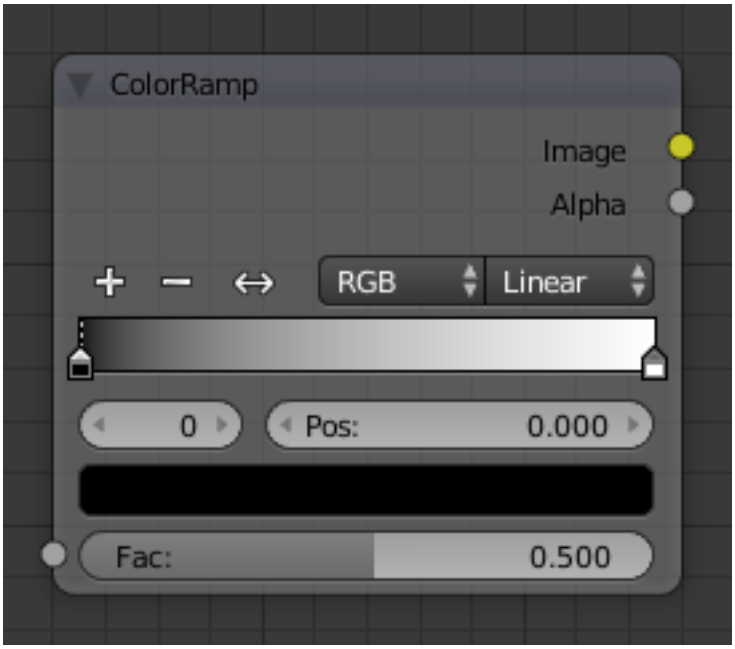


Fig. 2.2209: Color Ramp Node.

Examples

Creating an Alpha Mask

A powerful but often overlooked feature of the Color Ramp is to create an Alpha Mask, or a mask that is overlaid on top of another image, and, like a mask, allows some of the background to show through. The example map below shows how to use the Color Ramp node to do this:

In the map above, a black and white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*. (Technically, we should have converted the image to a value using the RGB-to-BW node, buy hey, this works just as well since we are using a BW image as input.)

We have set the Color Ramp node to a purely transparent color on the left end of the spectrum, and a fully Red color on the right. As seen in the viewer, the Color Ramp node puts out a mask that is fully transparent where the image is black. Black is zero, so Color Ramp uses the color at the left end of the spectrum, which we have set to transparent. The Color Ramp image is fully red and opaque where the image is white (1.00).

We verify that the output image mask is indeed transparent by overlaying it on top of a other image.

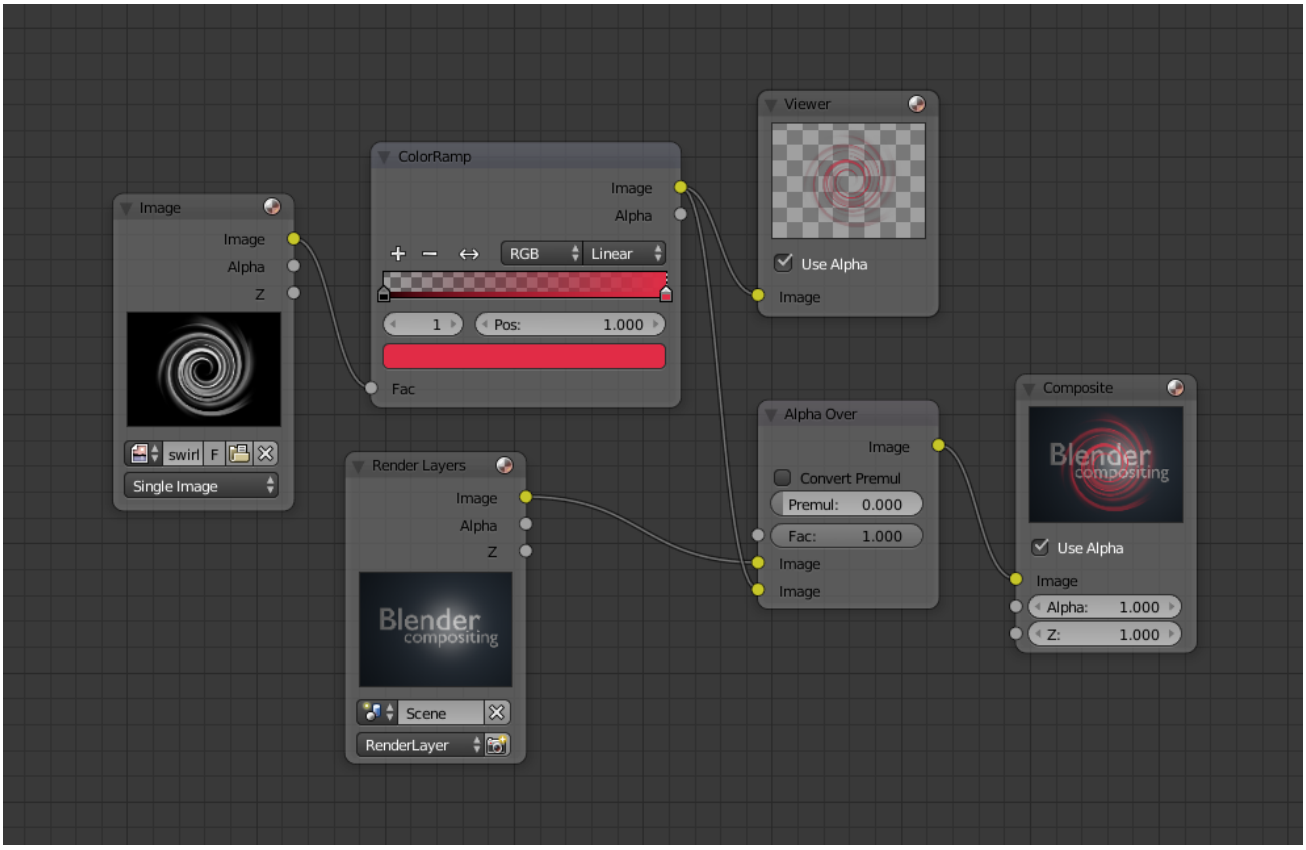


Fig. 2.2210: Using the Color Ramp node to create an alpha mask.

Colorizing an Image

The real power of Color Ramp is that multiple colors can be added to the color spectrum. This example compositing map takes a boring BW image and makes it a flaming swirl!

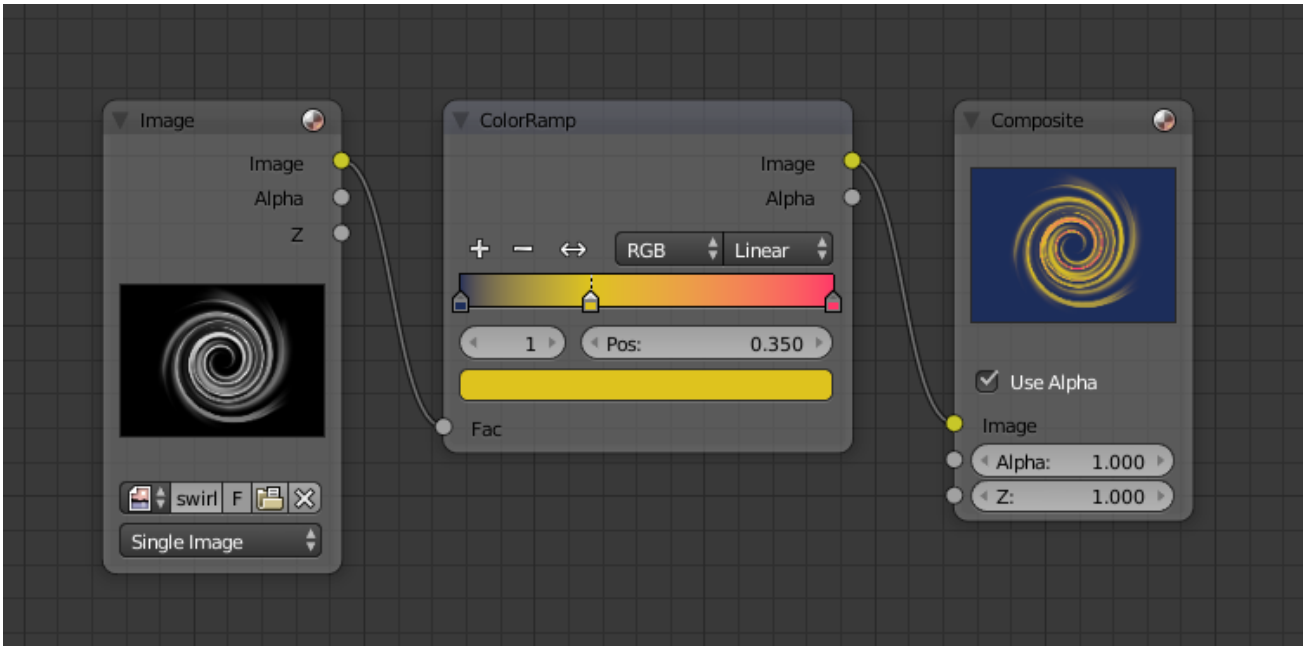
In this example, we have mapped the shades of gray in the input image to three colors, blue, yellow, and red, all fully opaque (Alpha of 1.00). Where the image is black, Color Ramp substitutes blue, the currently selected color. Where it is some shade of gray, Color Ramp chooses a corresponding color from the spectrum (bluish, yellow, to reddish). Where the image is fully white, Color Ramp chooses red.

Combine/Separate Nodes

All of these nodes do essentially the same thing:

- Separate: Split out an image into its composite color channels.
- Combine: Re/combine an image from it is composite color channels.

This nodes could be use this to manipulate on each color channel independently. Each type is differen-



tiate in the applied *color space*.

In compositing and texture context each node supports the Alpha channel. In the texture context only RGB color space is available. In the shading context of the Blender internal adds HSV and the Cycles shading context offers an additional pair of nodes to combine/separate a vector (XYZ).

The Combine nodes could also be used to input single color values. For RGBA and HSVA color spaces it is recommended to use the *RGB Node*. Some common operation could easier executed with the *Color Nodes*.

Separate/Combine RGBA Node

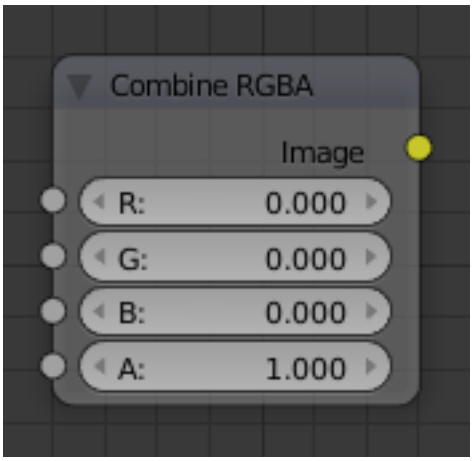


Fig. 2.2211: Combine RGBA Node.

Input/ Output

Image Standard image in/output.

- R (Red)
- G (Green)
- B (Blue)
- A (Alpha)

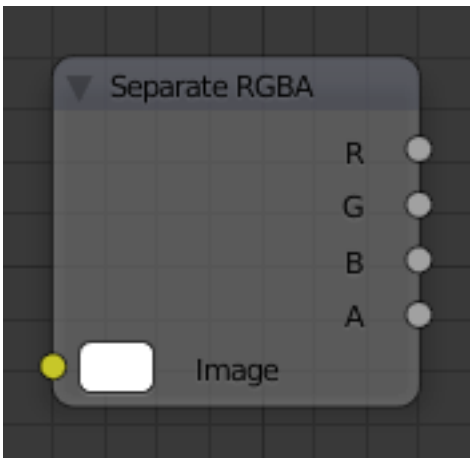


Fig. 2.2212: Separate RGBA Node.

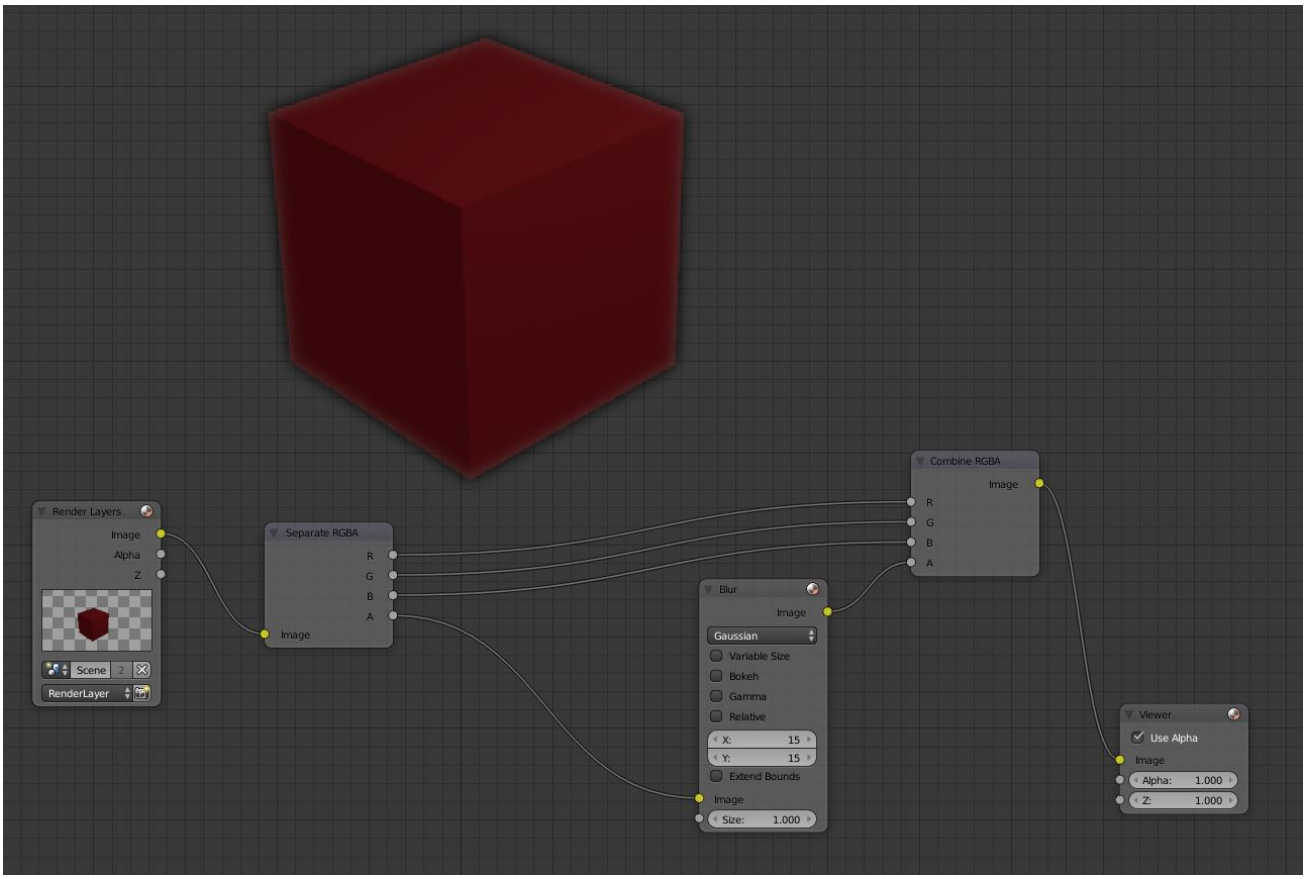
Properties

This node has no properties.

Examples

In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like anti-aliasing but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a broader scale will make the object appear to "phase" in and out, as an "out-of-phase" time-traveling sync effect.

In this node set up, we make all the reds become green, and all the green both Red and Blue, and remove Blue from the image completely.


Render Layers
Image
Alpha
Z
Scene
RenderLayer
Separate RGBA
R
G
B
A
Image
Combine RGBA
Image
R
G
B
A
Blur
Image
Gaussian
Variable Size
Bokeh
Gamma
Relative
X: 15
Y: 15
Extend Bounds
Image
Size: 1.000
Viewer
Use Alpha
Image
Alpha: 1.000
Z: 1.000

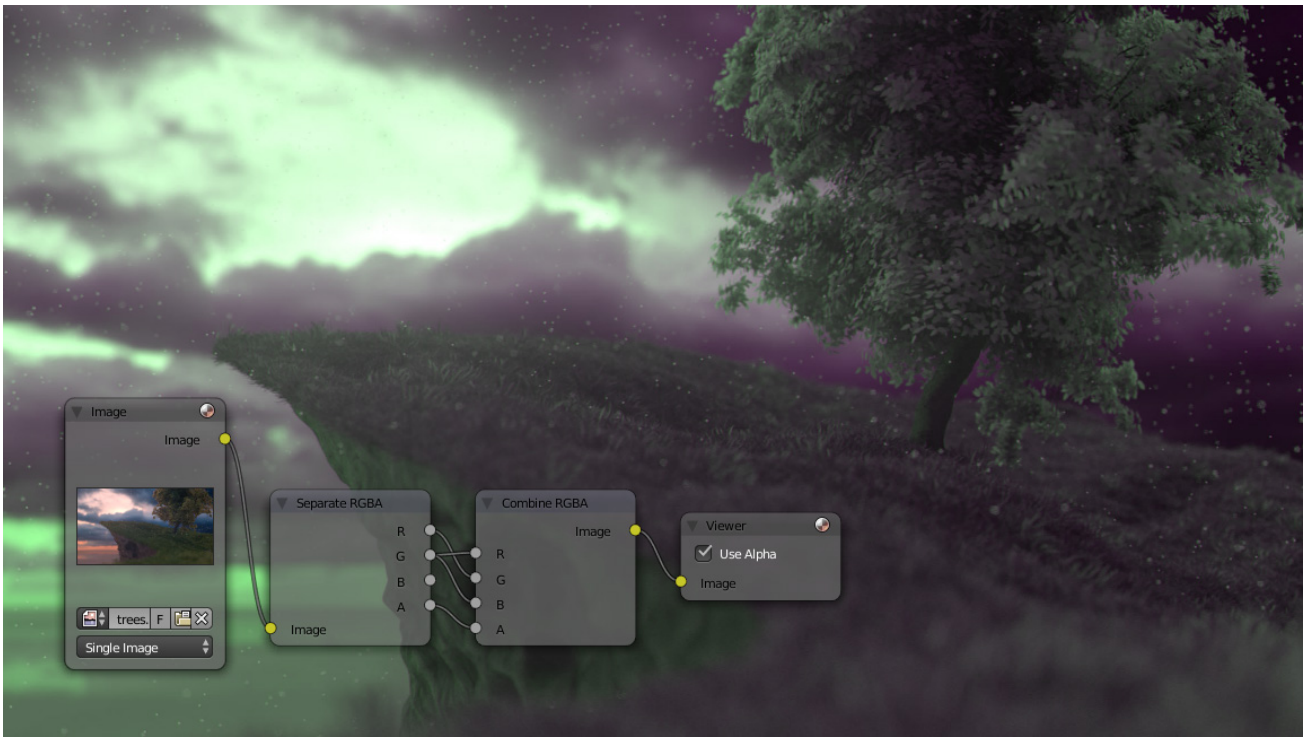


Image
Image
trees. F
Single Image
Separate RGBA
R
G
B
A
Image
Combine RGBA
Image
R
G
B
A
Viewer
Use Alpha
Image

Separate/Combine HSVA Nodes

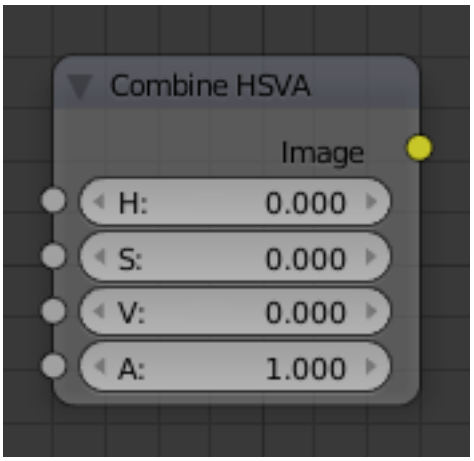


Fig. 2.2213: Combine HSVA Node.

Input/ Output

Image Standard image in/output.

- H (Hue)
- S (Saturation)
- V (Value)
- A (Alpha)

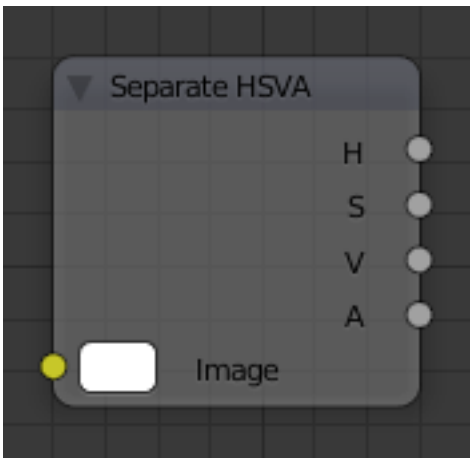


Fig. 2.2214: Separate HSVA Node.

Properties

This node has no properties.

Separate/Combine YUVA Node

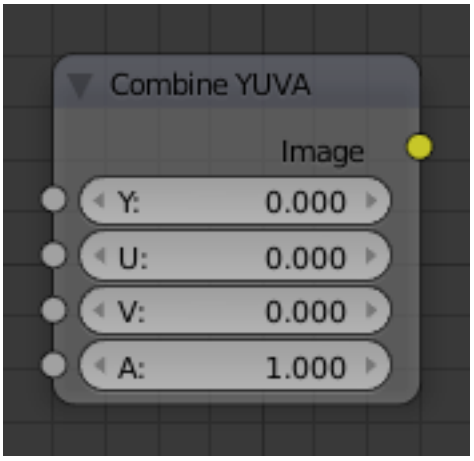


Fig. 2.2215: Combine YUVA Node.

Input/ Output

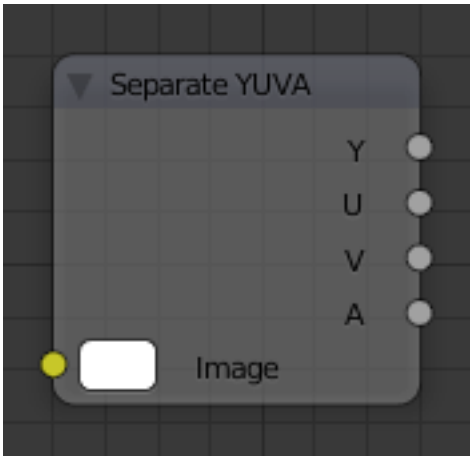


Fig. 2.2216: Separate YUVA Node.

Image Standard image in/output.

- Y (Luminance)
- U (U chrominance)
- V (V chrominance)
- A (Alpha)

Properties

This node has no properties.

Separate/Combine YCbCrA Node

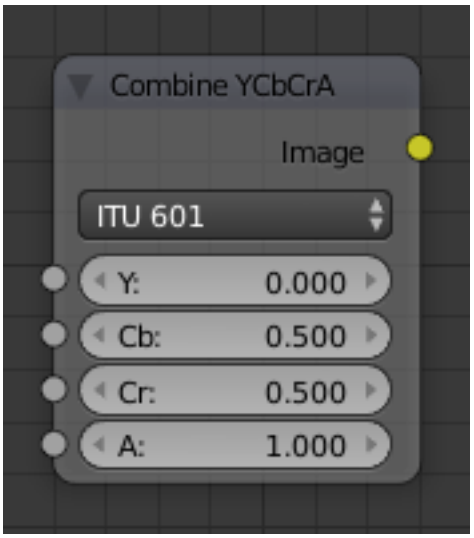


Fig. 2.2217: Combine YCbCrA Node.

Input/ Output

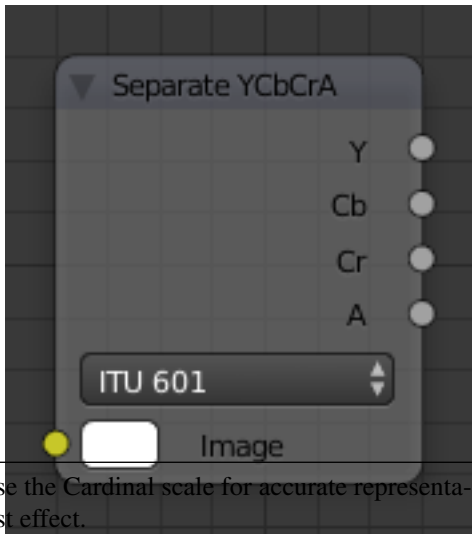


Fig. 2.2218: Separate YCbCrA Node.

Image Standard image in/output.

- Y (Luminance)
- Cb (Chrominance Blue)
- Cr (Chrominance Red)
- A (Alpha)

Properties

Mode ITU 601, ITU 709, Jpeg

Tip: If running these channels through a Color Ramp node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives high-contrast effect.

ID Mask Node

This node can be used to access an alpha mask per object or per material.

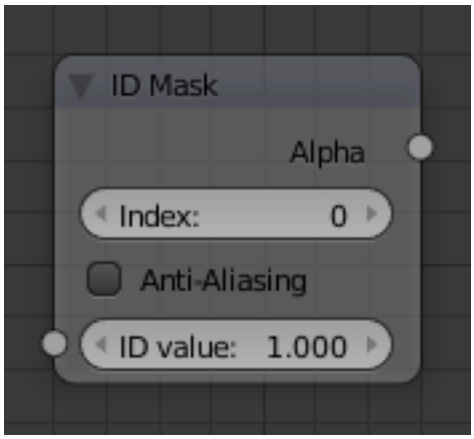


Fig. 2.2219: ID Mask Node.

Inputs

ID value Input for the *Object Index* or *Material Index* render pass. Which is an output of the *Render Layers node* or the *Image node* with a multilayer format.

Properties

Index Selection of the preciously specified index.

Anti-Aliased This post-process function refines the mask. See *anti-aliasing*.

Outputs

Alpha The mask is white where the object is and black where it is not. If the object is transparent, the alpha mask represent that with gray values.

Note: In Blender Internal if a transparent object in front of another, the mask will not reflect partial visibility of the object behind.

Setup

An index can be specify for any object or material in the scene. The Object Index can be set in Properties Editor *Object* → *Relations* → *Pass Index* and *Material* → *Options* → *Pass Index* for the Material Index. To be accessible after rendering, *Object Index* or *Material Index* render pass has to be enabled.

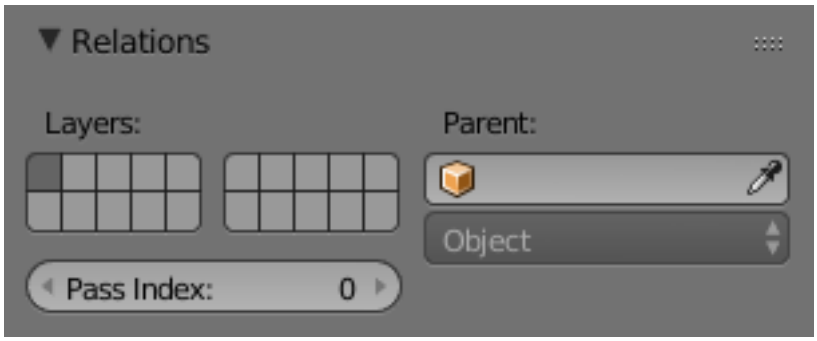


Fig. 2.2220: Object Pass Index.

Example

In this example, the left rear red cube is assigned Pass Index 1, and the right cube Pass Index 2. Where the two cubes intersect, there is going to be noticeable pixelation because they come together at a sharp angle and are different colors. Using the mask from object 1, which is smoothed (antialiased) at the edges, we use a *Mix Node* set on *Multiply* to multiply the smoothed edges of the image, thus removing those nasty lines, thus, being smoothed out.

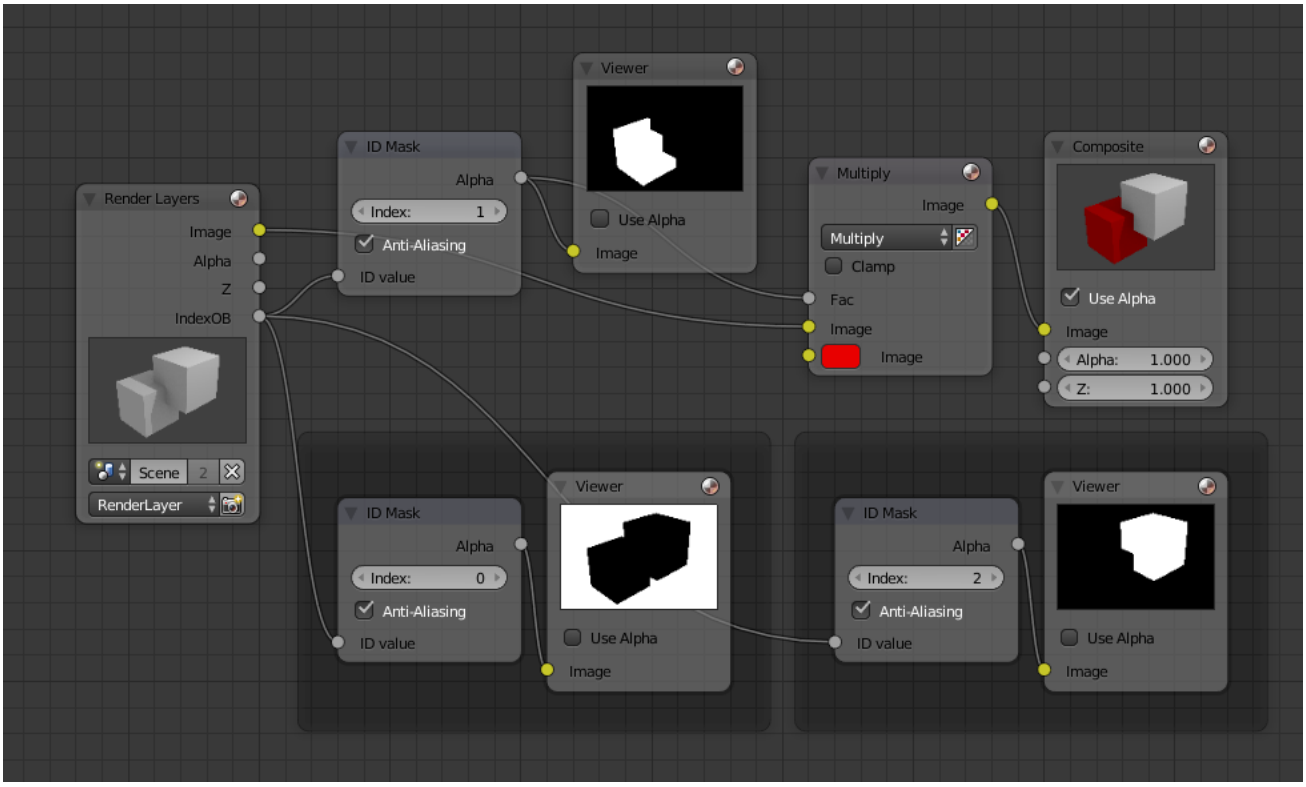


Fig. 2.2221: Id Mask node example.

Math Node

This node performs math operations.

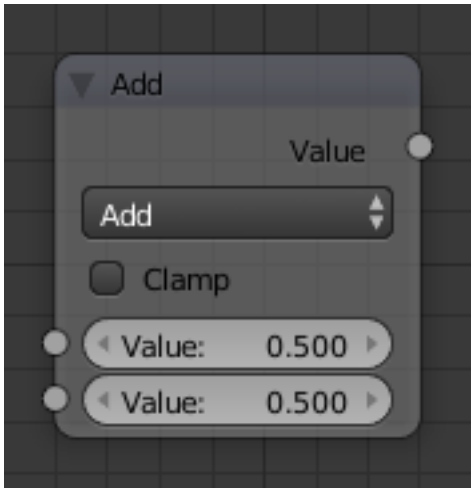


Fig. 2.2222: Math node.

Inputs

Value First numerical value. The trigonometric functions accept values in radians.

Value Second numerical value. This value is **not** used in functions that accept only one parameter like the trigonometric functions, Round and Absolute.

Properties

Operation Add, Subtract, Multiply, Divide, Sine, Cosine, Tangent, Arcsine, Arccosine, Arctangent, Power, Logarithm, Minimum, Maximum, Round, Less Than, Greater Than, Modulo, Absolute.

Clamp Limits the output to the range (0 to 1). See *clamp*.

Outputs

Value Numerical value output.

Examples

Manual Z-Mask

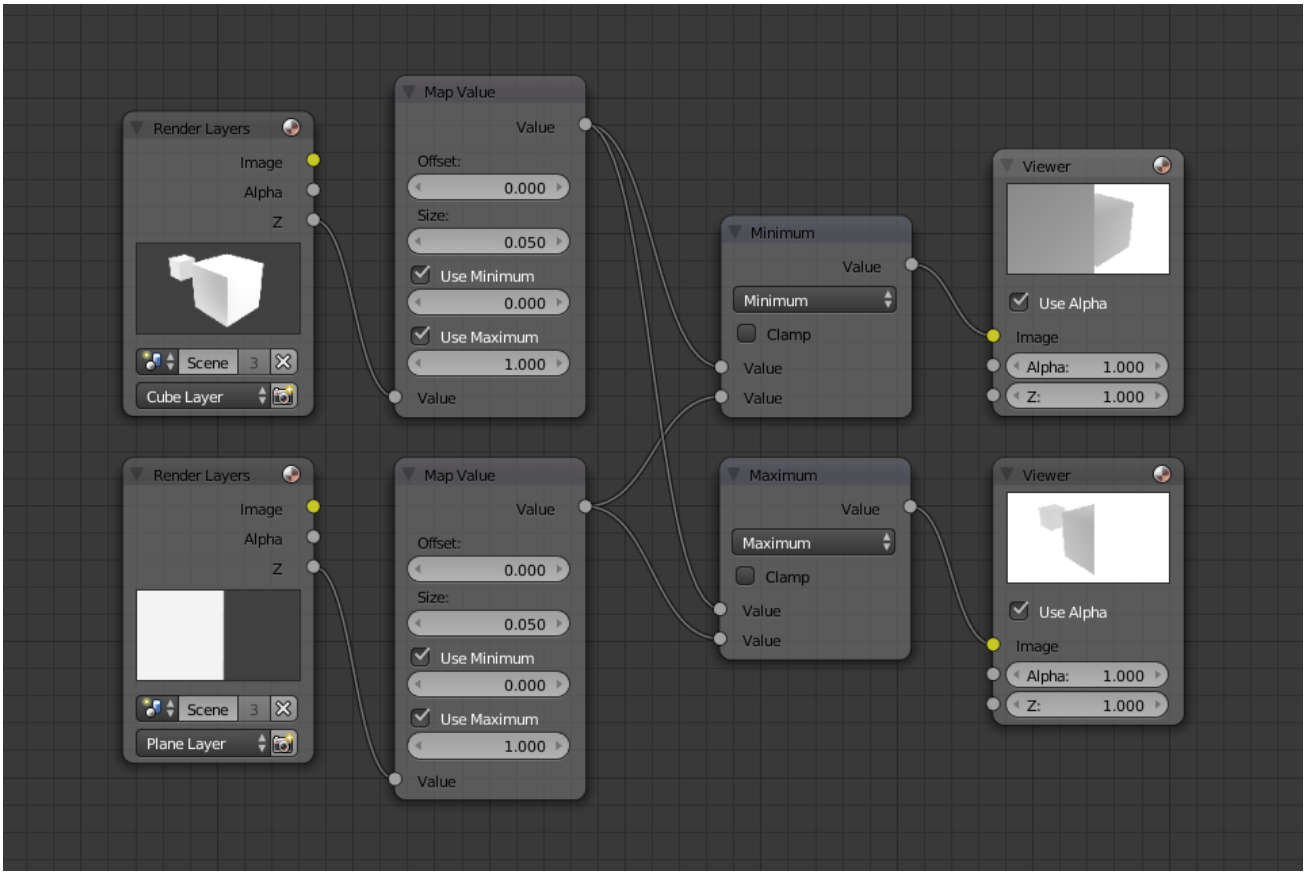


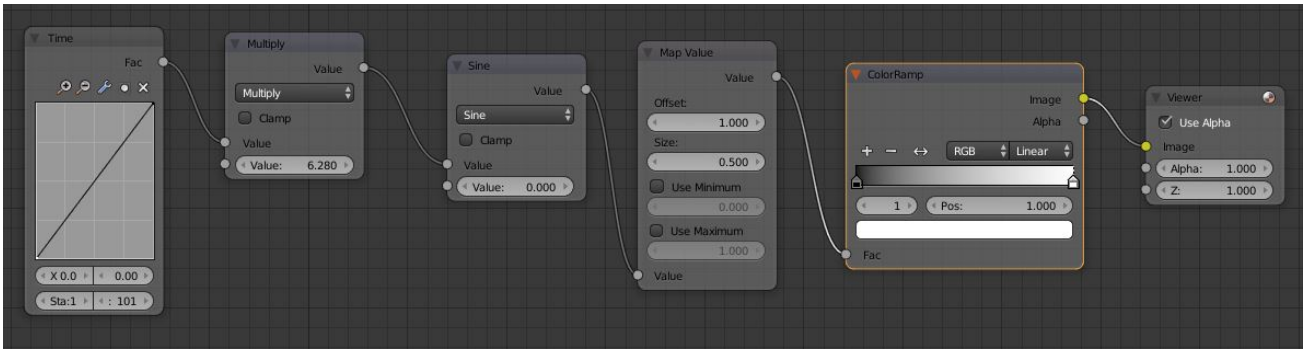
Fig. 2.2223: Example.

This example has one scene input by the top *Render Layer* node, which has a cube that is about 10 BU from the camera. The bottom Render Layer node inputs a scene (FlyCam) with a plane that covers the left half of the view and is 7 BU from the camera. Both are fed through their respective Map Value nodes to divide the Z buffer by 20 (multiply by 0.05, as shown in the Size field) and clamped to be a min/ max of 0.0/ 1.0 respectively.

For the minimum function, the node selects those Z values where the corresponding pixel is closer to the camera; so it chooses the Z values for the plane and part of the cube. The background has an infinite Z value, so it is clamped to 1.0 (shown as white). In the maximum example, the Z values of the cube are greater than the plane, so they are chosen for the left side, but the plane (FlyCam) Render layers Z are infinite (mapped to 1.0) for the right side, so they are chosen.

Using Sine Function to Pulsate

This example has a *Time* node putting out a linear sequence from 0 to 1 over the course of 101 frames. The green vertical line in the curve widget shows that frame 25 is being put out, or a value of 0.25. That value is multiplied by 2 × pi and converted to 1.0 by the Sine function, since we all know that $sin(2pi/4) = sin(pi/2) = +1.0$ Since the sine function can put out values between (-1.0 to 1.0), the *Map Value* node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by one-half (thus scaling the output between 0 to 1). The default *Color Ramp* converts those values to a grayscale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, $sin(pi/2) = 1.0$. Like having your own visual color calculator! Animating this node setup provides a smooth cyclic sequence through the range of grays.



Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move a scene in/out of focus. Alter a color channel value to make a color "pulse".

Brightening/Scaling a Channel

This example has a *Math: Multiply* node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a *Map Value node* with min() and max() enabled to clamp the output to valid values. With this approach, you could use a logarithmic function to make a high-dynamic range image. For this particular example, there is also a *Brighten/Contrast node* that might give simpler control over brightness.

Quantize/Restrict Color Selection

In this example, we want to restrict the color output to only 256 possible values. Possible use of this is to see what the image will look like on an 8-bit cell phone display. To do this, we want to restrict the R, G and B values of any pixel to be one of a certain value, such that when they are combined, will not result in more than 256 possible values. The number of possible values of an output is the number of channel values multiplied by each other, or Q = R × G × B.

Since there are three channels and 256 values, we have some flexibility how to quantize each channel, since there are a lot of combinations of R × G × B that would equal 256. For example, if {R, G, B} = {4, 4, 16}, then $4416 = 256$. Also, {6, 6, 7} would give 252 possible values. The difference in appearance between {4, 4, 16} and {6, 6, 7} is that the first set

(4, 4, 16} would have fewer shades of red and green, but lots of shades of blue. The set {6, 6, 7} would have a more even distribution of colors. To get better image quality with fewer color values, give possible values to the predominant colors in the image.

Theory

Two Approaches to Quantizing to six values.

To accomplish this quantization of an image to 256 possible values, let us use the set {6, 6, 7}. To split up a continuous range of values between 0 and 1 (the full Red spectrum) into six values, we need to construct an algorithm or function that takes any input value but only puts out six possible values, as illustrated by the image to the right. We want to include zero as true black, with five other colors in between. The approach shown produces {0, 0.2, 0.4, 0.6, 0.8, 1}. Dividing 1.0 by 5 equals 0.2, which tells how far apart each quantified value is from the other.

So, to get good even shading, we want to take values that are 0.16 or less and map them to 0.0; values between 0.16 and 0.33 get fixed to 0.2; color band values between 0.33 and 0.5 get quantized to 0.4, and so on up to values between 0.83 and 1.0 get mapped to 1.0.

Note: Function f(x)

An algebraic function is made up of primitive mathematical operations (add, subtract, multiply, sine, cosine, etc) that operate on an input value to provide the desired output value.

Spreadsheet showing a function.

The theory behind this function is scaled truncation. Suppose we want a math function that takes in a range of values between 0 and 1, such as 0.552, but only outputs a value of 0.0, 0.2, 0.4, etc. We can imagine then that we need to get that range 0 to 1 powered up to something 0 to 6 so that we can chop off and make it a whole number. So, with six divisions, how can we do that? The answer is we multiply the range by 6. The output of that first math Multiply Node is a range of values between 0 and 6. To get even divisions, because we are using the rounding function (see documentation above), we want any number plus or minus around a whole number will get rounded to that number. So, we subtract a half, which shifts everything over. The round() function then makes that range 0 to 5. We then divide by 5 to get back a range of numbers between 0 and 1 which can then be combined back with the other color channels. Thus, you get the function $f(x, n) = round(xn - 0.5)/(n - 1)$ where "n" is the number of possible output values, and "x" is the input pixel color and $f(x, n)$ is the output value. There is only one slight problem, and that is for the value exactly equal to 1, the formula result is 1.2, which is an invalid value. This is because the round function is actually a roundup function, and exactly 5.5 is rounded up to 6. So, by subtracting 0.501, we compensate and thus 5. 499 is rounded to 5. At the other end of the spectrum, pure black, or 0, when 0.501 subtracted, rounds up to 0 since the Round() function does not return a negative number.

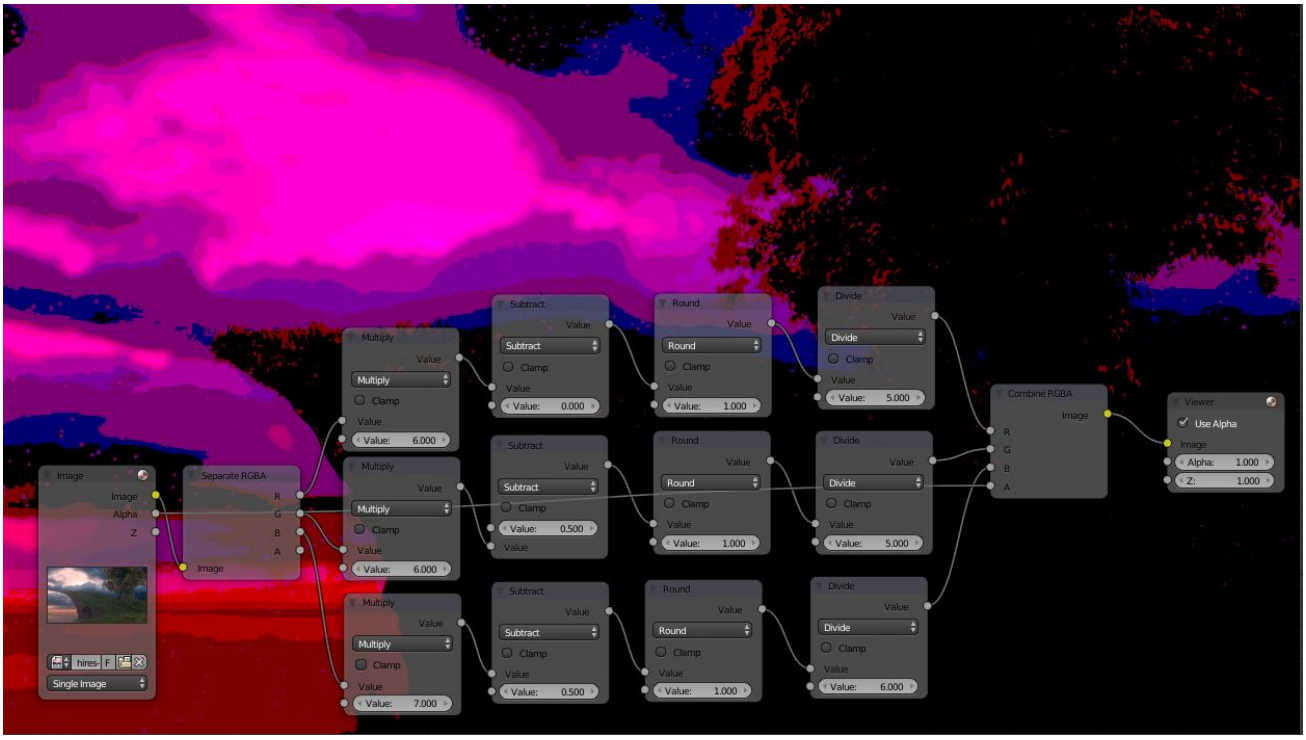
Sometimes using a spreadsheet can help you figure out how to put these nodes together to get the result that you want. Stepping you through the formula for $n = 6$ and $x = 0.70$, locate the line on the spreadsheet that has the 8-bit value 179 and R value 0.7. Multiplying by 6 gives 4.2 . Subtracting 1/2 gives 3.7, which rounds up to 4.4 divided by 5 = 0.8. Thus, f(0.7, 6) = 0.8 or an 8-bit value of 204. You can see that this same 8-bit value is output for a range of input values.

Reality

To implement this function in Blender, consider the node setup above. First, feed the image to the Separate RGB node. For the Red channel, we string the math nodes into a function that takes each red color, multiplies (scales) it up by the desired number of divisions (6), offsets it by 0.5, rounds the value to the nearest whole number, and then divides the image pixel color by 5. So, the transformation is {0 to 1} becomes {0 to 6}, subtracting centers the medians to {-0.5 to 5.5} and the rounding to the nearest whole number produces {0, 1, 2, 3, 4, 5} since the function rounds down, and then dividing by five results in six values {0.0, 0.2, 0.4, 0.6, 0.8, 1.0}.

The result is that the output value can only be one of a certain set of values, stair-stepped, because of the rounding function of the math node node setup. Copying this one channel to operate on Green and Blue gives the node setup below. To get the 6:6:7, we set the three Multiply Nodes to {6, 6, 7} and the divide nodes to {5, 5, 6}.

If you make this into a node group, you can easily re-use this setup from project to project. When you do, consider using a math node to drive the different values that you would have to otherwise set manually, just to error-proof your work.



Summary

Normally, an output render consists of 32- or 24-bit color depth, and each pixel can be one of the millions of possible colors. This node setup example takes each of the Red, Green and Blue channels and normalizes them to one of a few values. When all three channels are combined back together, each color can only be one of 256 possible values.

While this example uses the Separate/Combine RGB to create distinct colors, other Separate/Combine nodes can be used as well. If using the YUV values, remember that U and V vary between (-0.5 to +0.5), so you will have to first add on a half to bring the range between 0 and 1, and then after dividing, subtract a half to bring in back into standard range.

The `JPG` or `PNG` image format will store each of the colors according to their image standard for color depth (e.g. `JPG` is 24-bit), but the image will be very very small since reducing color depth and quantizing colors are essentially what the `JPEG` compression algorithm accomplishes.

You do not have to reduce the color depth of each channel evenly. For example, if blue was the dominant color in an image, to preserve image quality, you could reduce Red to 2 values, Green to 4, and let the blue take on $256/(24)$ or 32 values. If using the HSV, you could reduce the Saturation and Value to 2 values (0 or 1.0) by Multiply by 2 and Divide by 2, and restrict the Hue to 64 possible values.

You can use this node setup to quantize any channel; alpha, speed (vector), z-values, and so forth.

RGB to BW Node

This node maps a RGB color image to a grayscale by the luminance.

Inputs

Image Color image input.

Properties

This node has no properties.

Fig. 2.2224: RGB to BW Node.

Outputs

Value Grayscale value output.

Set Alpha Node

This node adds an alpha channel to an image.

Inputs

Image Standard image input.

Alpha The amount of Alpha can be set for the whole image by using the input field or per pixel by connecting to the socket.

Properties

This node has no properties.

Outputs

Image Standard image output.

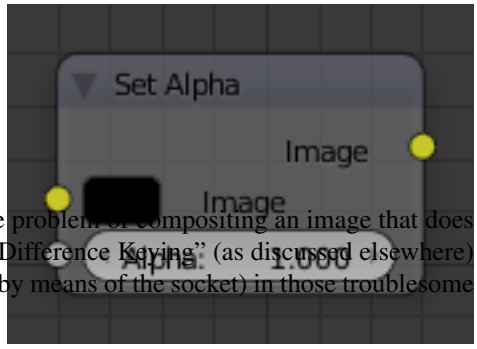


Fig. 2.2225: Set Alpha Node.

Note: This is not, and is not intended to be, a general-purpose solution to the problem of compositing an image that does not contain Alpha information. You might wish to use "Chroma Keying" or "Difference Keying" (as discussed elsewhere) if you can. This node is most often used (with a suitable input being provided by means of the socket) in those troublesome cases when you *cannot*, for some reason, use those techniques directly.

Example

Fade To Black

To transition the audience from one scene or shot to another, a common technique is to "fade to black". As its name implies, the scene fades to a black screen. You can also "fade to white" or whatever color you wish, but black is a good neutral color that is easy on the eyes and intellectually "resets" the viewer's mind. The node map below shows how to do this using the Set Alpha node.

In the example above, the alpha channel of the swirl image is ignored. Instead, a *time node* introduces a factor from 0.00 to 1.00 over 60 frames, or about 2 seconds, to the Set Alpha node. Note that the time curve is exponentially-shaped, so that the overall blackness will fade in slowly and then accelerate toward the end. The Set Alpha node does not need an input image; instead, the flat (shadeless) black color is used. The Set Alpha Node uses the input factor and color to create a black image that has an alpha set which goes from 0.00 to 1.00 over 60 frames, or completely transparent to completely opaque. Think of alpha as a multiplier for how vivid you can see that pixel. These two images are combined by our trusty Alpha Over node completely (a *Factor* of 1.00) to produce the composite image. The Set Alpha node will thus, depending on the frame being rendered, produce a black image that has some degree of transparency. Setup and Animate, and you have an image sequence that fades to black over a 2-second period.

Note: No Scene information used

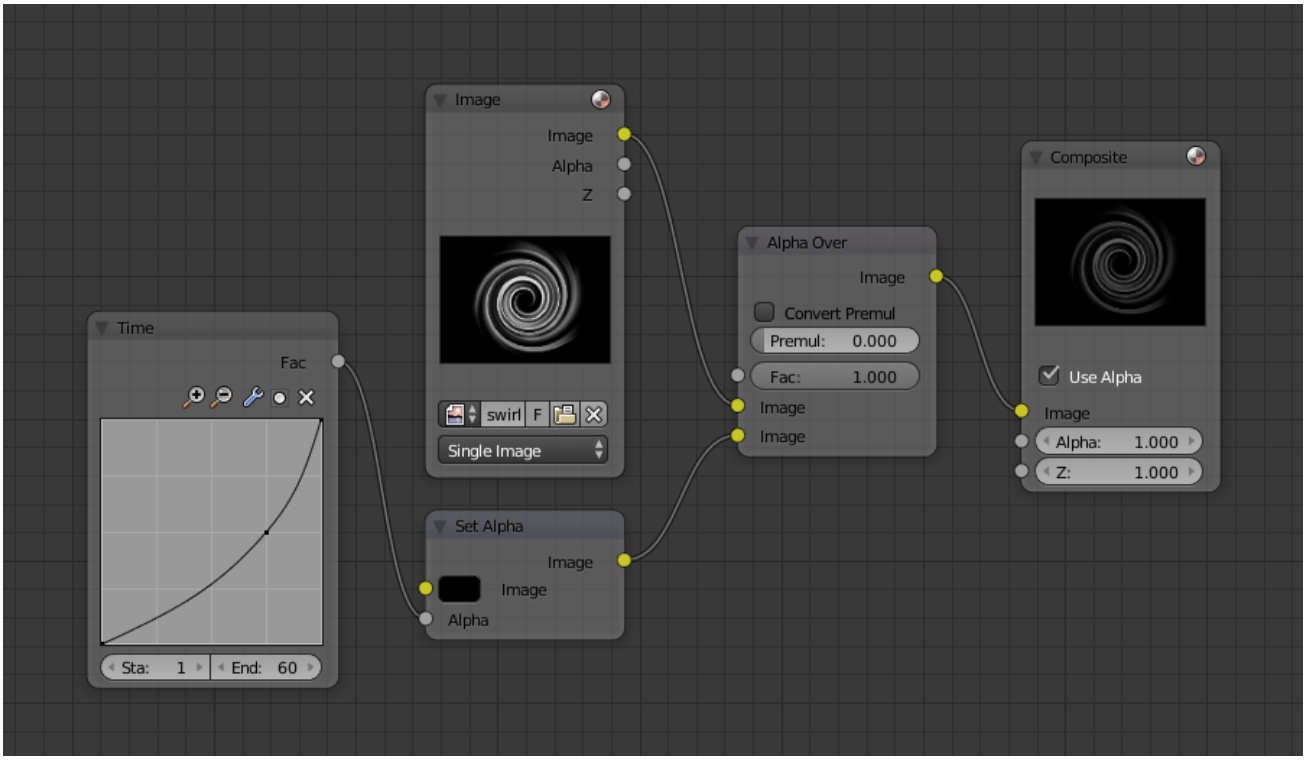


Fig. 2.2226: Fade To Black.

This example node map does not use the Render Layer node. To produce this 2-second animation, no Blender scene information was used. This is an example of using Blender's powerful compositing abilities separate from its modeling and animation capabilities. (A Render Layer could be substituted for the Image layer, and the "fade-network" effect will still produce the same effect).

Fade In a Title

To introduce your animation, you will want to present the title of your animation over a background. You can have the title fly in, or fade it in. To fade it in, use the Set Alpha node with the Time node as shown below.

In the above example, a Time curve provides the Alpha value to the input socket. The current Render Layer node, which has the title in view, provides the image. As before, the trusty Alpha Over node mixes (using the alpha values) the background swirl and the alpha title to produce the composite image. Notice the *Convert Premultiplied* - checkbox is **not** enabled; this produces a composite where the title lets the background image show through where even the background image is transparent, allowing you to layer images on top of one another.

Colorizing a BW Image

In the example above, notice how the blue tinge of the render input colors the swirl. You can use the Set Alpha node's color button with this kind of node map to add a consistent color to a BW image.

In the example map to the right, use the *Alpha* value of the Set Alpha node to give a desired degree of colorization. Thread the input image and the Set Alpha node into an Alpha Over node to colorize any black and white image in this manner. Note the *Convert Premultiplied* checkbox is enabled, which tells the Alpha Over node not to multiply the alpha values of the two images together.

Switch View Node

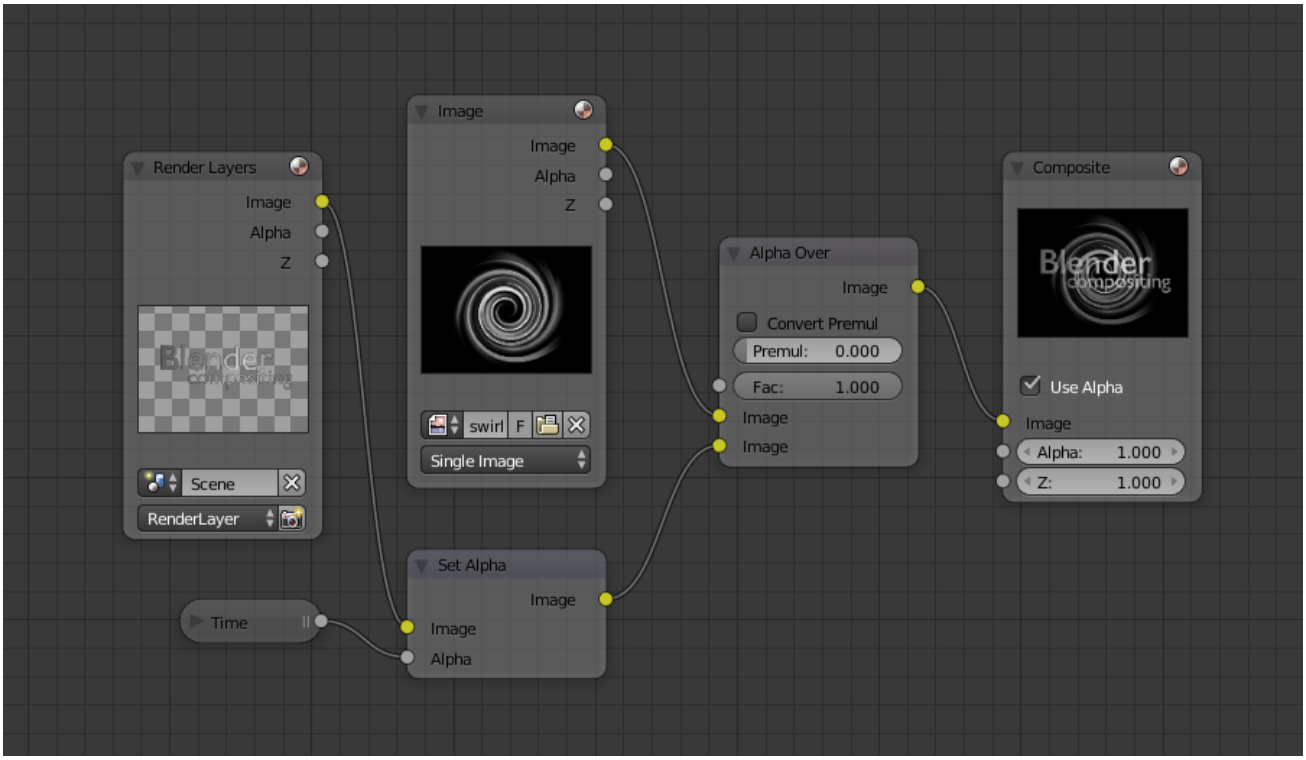


Fig. 2.2227: Using Set Alpha to Fade in a Title.

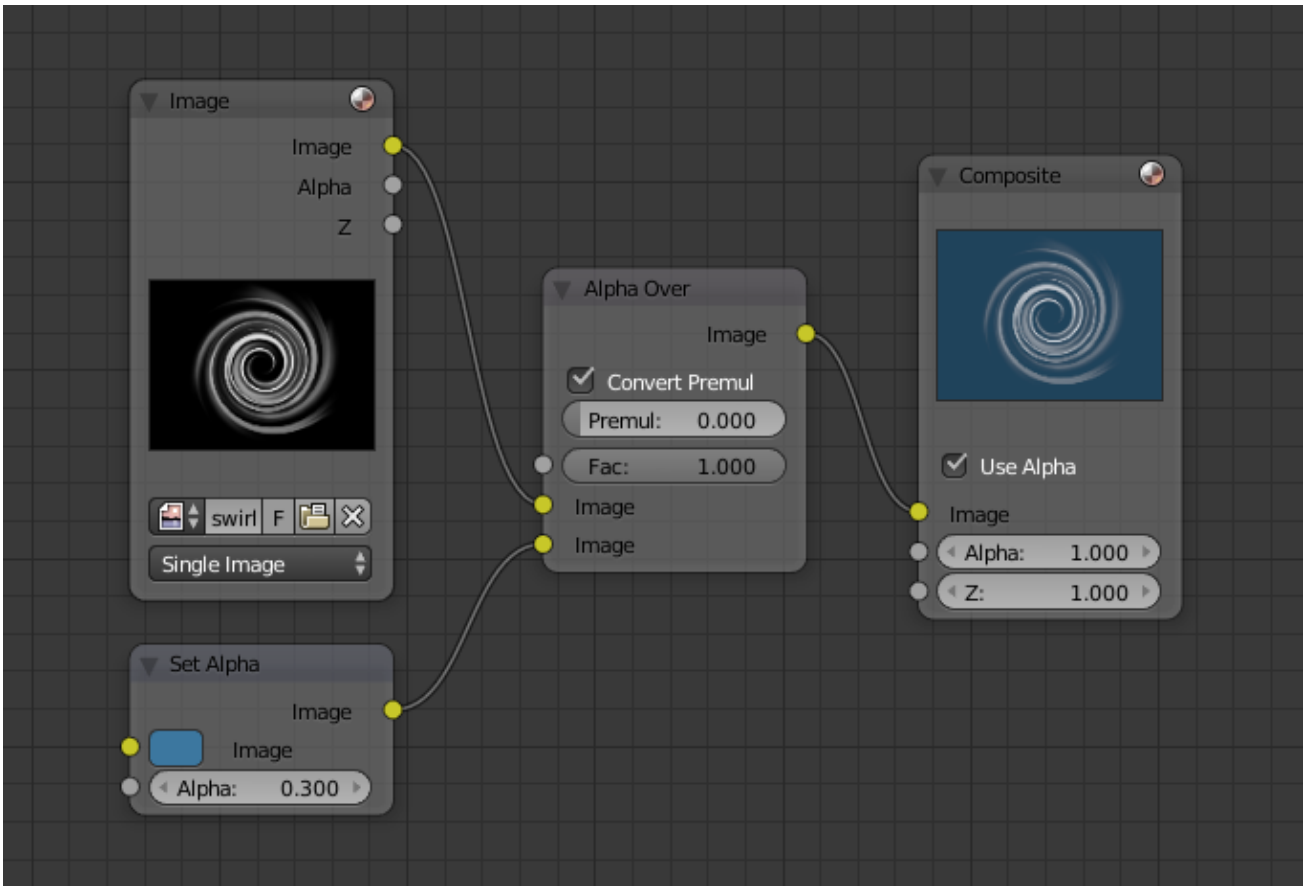


Fig. 2.2228: Using Set Alpha to Colorize an Image.

The *Switch View* node combines the *views* (left and right) into a single Stereo 3D output. This can be useful if for example, you need to treat the view as separate images by combining each of the views.

See also:

The Multi-View Workflow.

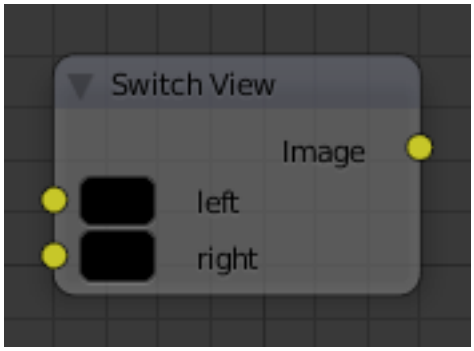


Fig. 2.2229: Switch View Node.

Inputs

Left Left eye image input.

Right Right eye image input.

Properties

This node has no properties.

Outputs

Image Stereo 3D image output.

Example

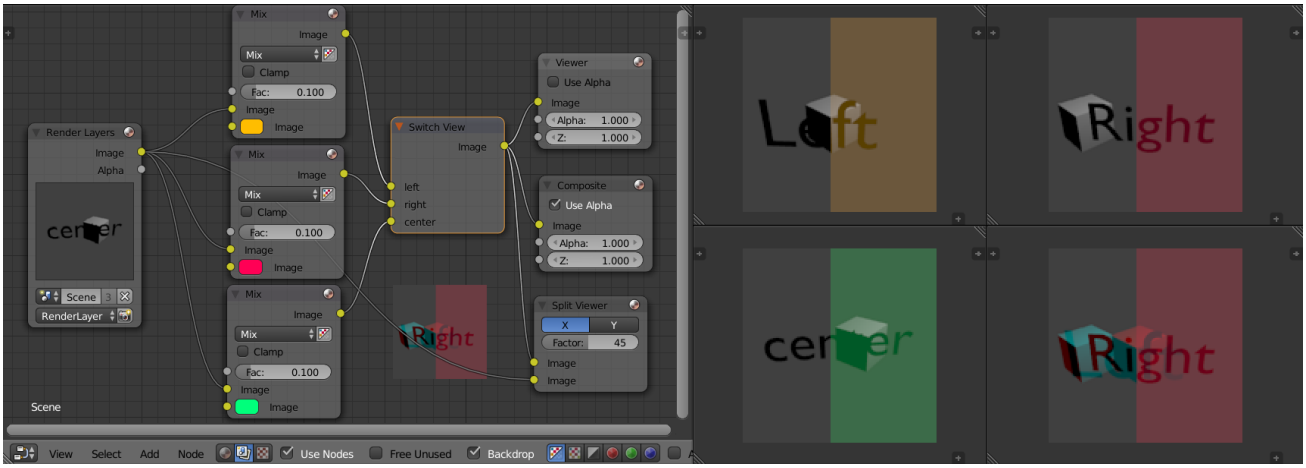


Fig. 2.2230: Compositor, Backdrop and Split Viewer Node.

The views to render are defined in the current scene views, in a similar way
as you define the composite output resolution in the current scene render panel, regardless of the Image nodes resolutions or Render Layers from different scenes.

Filter Nodes

Filters process the pixels of an image to highlight additional details or perform some sort of post-processing effect on the image.

Bilateral Blur Node

The Bilateral Blur node performs a high-quality adaptive blur on the source image.

It can be used for various purposes like: smoothing results from Blender's raytraced ambient occlusion smoothing results from various unbiased renderers, to fake some performance-heavy processes, like blurry refractions/reflections, soft shadows, to make non-photorealistic compositing effects.

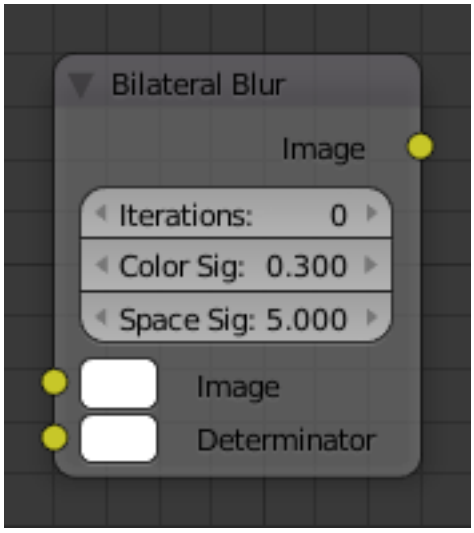


Fig. 2.2231: Bilateral Blur Node.

Inputs

Image Standard image input. If only the image input is connected, the node blurs the image depending on the edges present in the source image.

Determinator Which is non-obligatory and if the Determinator is connected, it serves as the source for defining edges/borders for the blur in the image. This has great advantage in case the source image is too noisy, but normals in combination with Z-buffer can still define exact borders/edges of objects.

Properties

Iterations Defines how many times the filter should perform the operation on the image. It practically defines the radius of blur.

Color Sigma Defines the threshold for which color differences in the image should be taken as edges.

Space Sigma A fine-tuning variable for blur radius.

Outputs

Image Standard image output.

Examples

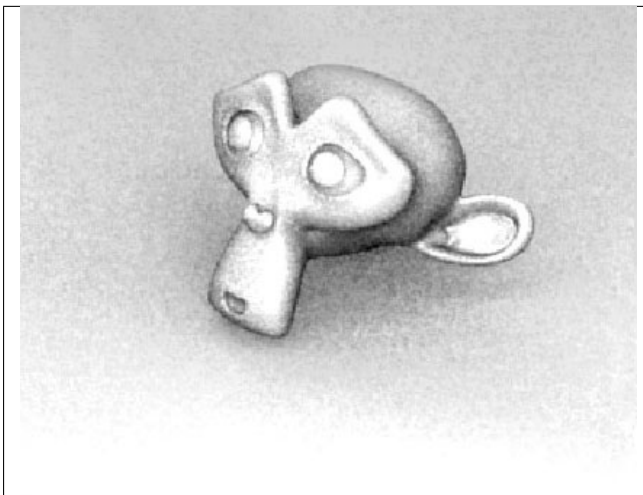


Fig. 2.2233: Render result.

Fig. 2.2234: Composite.

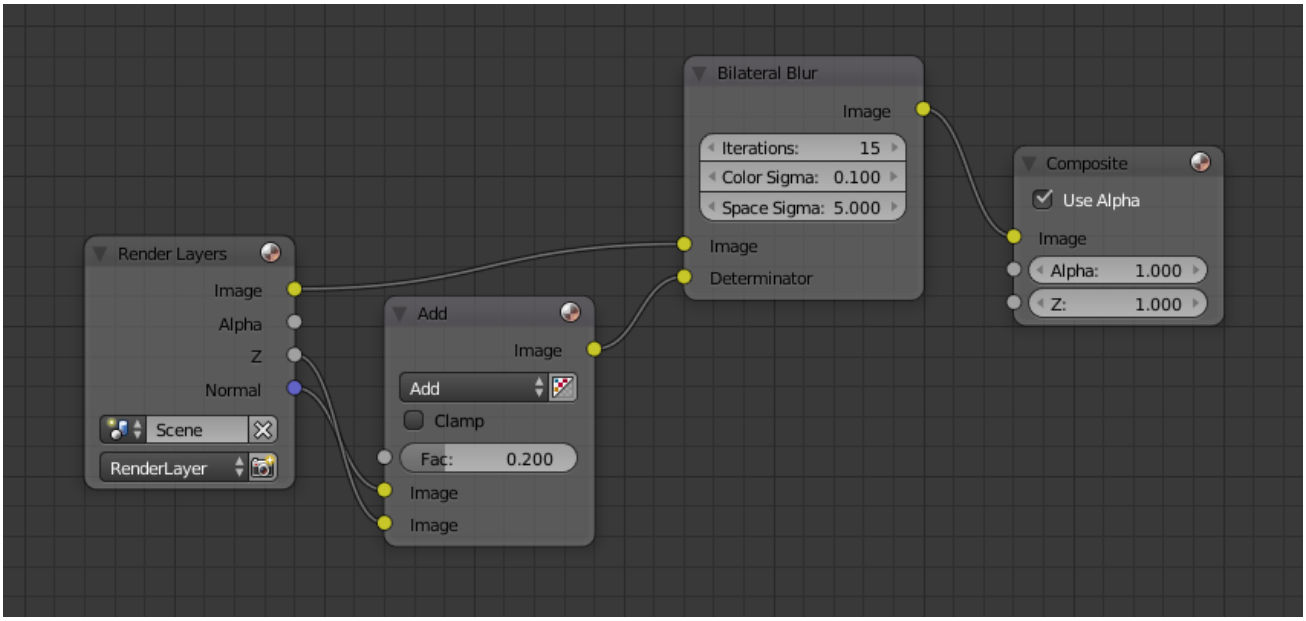


Fig. 2.2232: Bilateral smoothed AO.

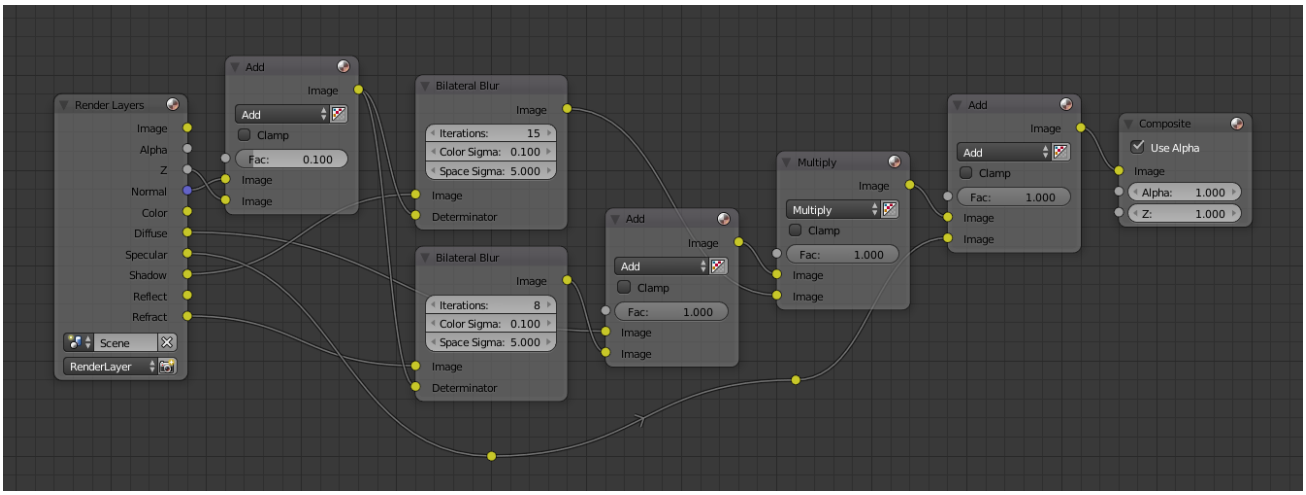


Fig. 2.2235: Bilateral faked blurry refraction and smoothed raytraced soft shadow.

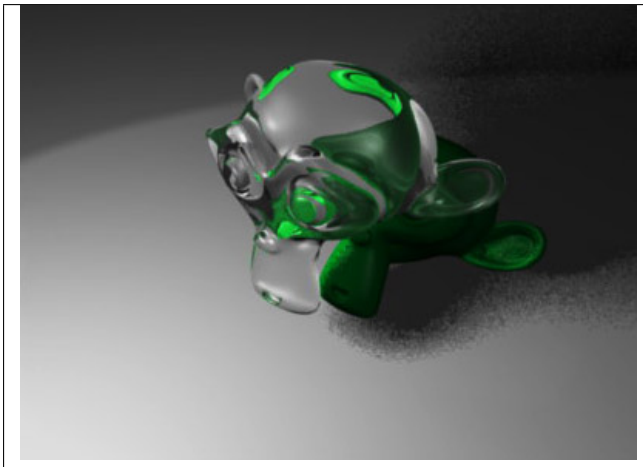


Fig. 2.2236: Render result.

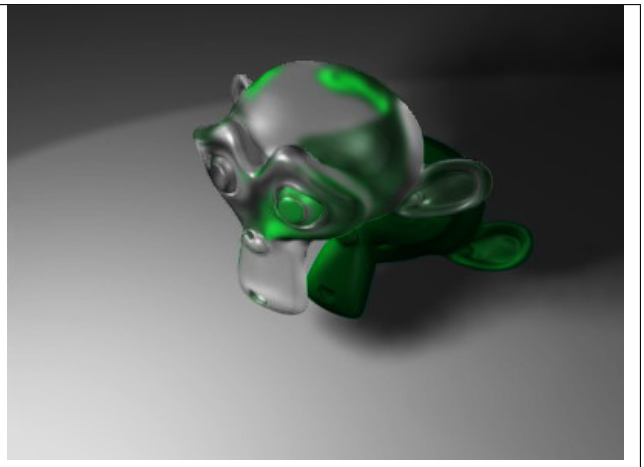


Fig. 2.2237: Composite.

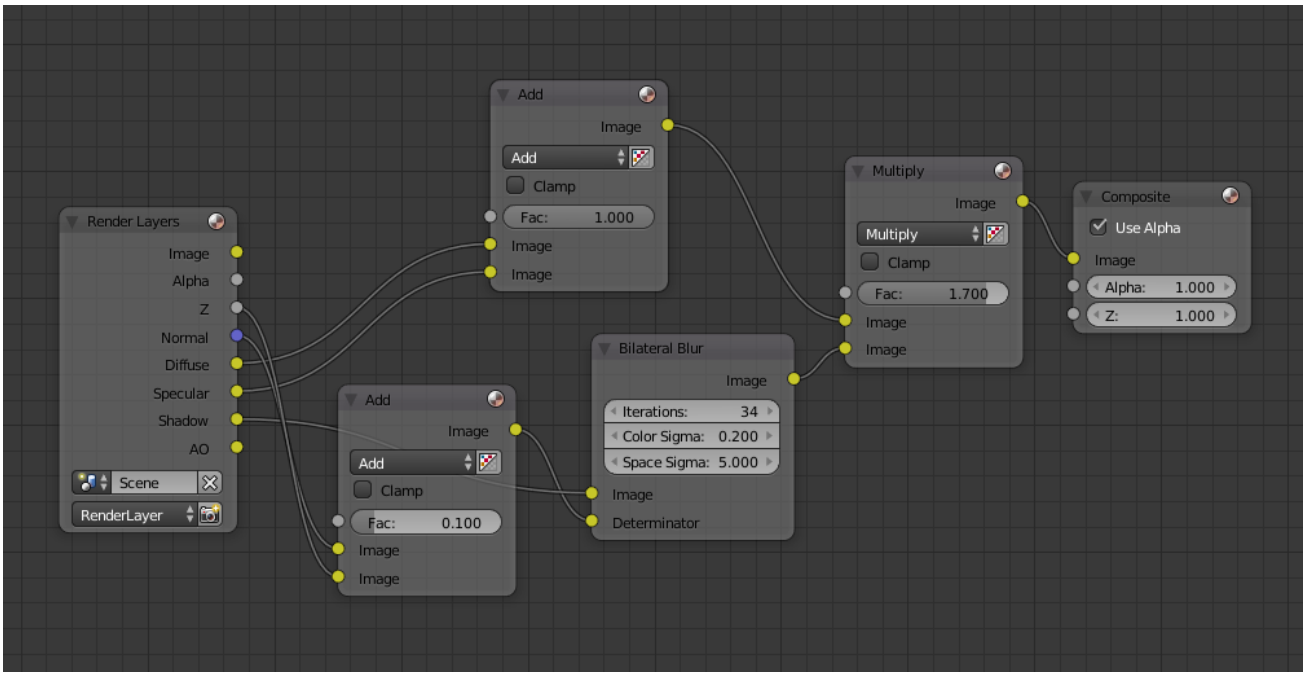


Fig. 2.2238: Bilateral smoothed buffered shadow.

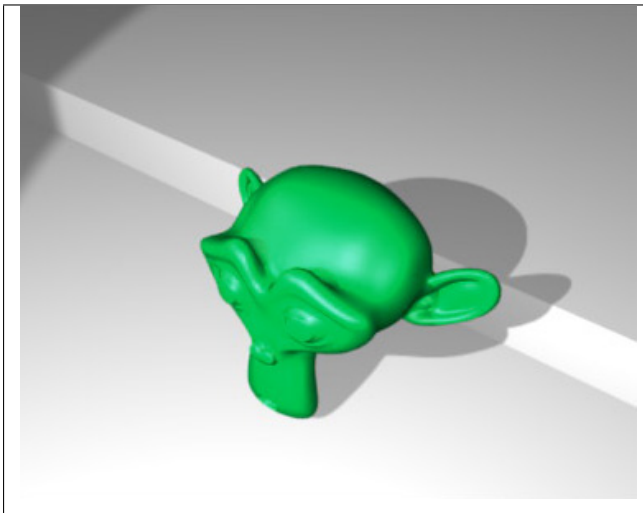


Fig. 2.2239: Render result.

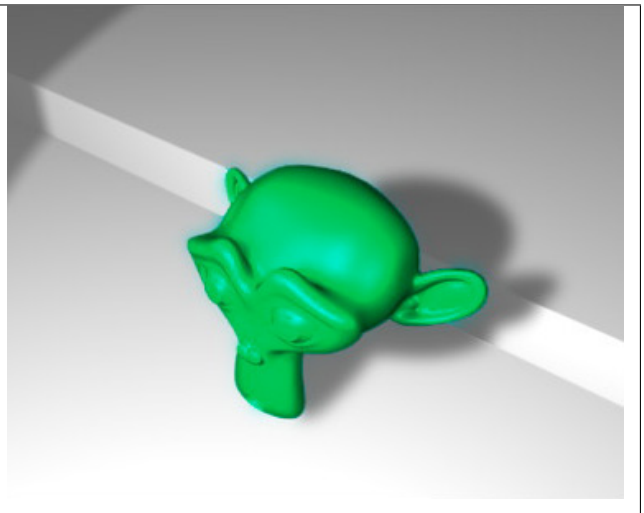


Fig. 2.2240: Composite.

Blur Node

The Blur node blurs an image, providing several blur modes.

Fig. 2.2241: Blur Node.

Inputs

Image Standard image input.

Size The optional Size input will be multiplied with the X and Y blur radius values. It accepts also a value image, to control the blur radius with a mask. The values should be mapped between (0 to 1) for an optimal effect.

Properties

Type The difference between the types is in the way they handle sharp edges, smooth gradients and preserve the highs and the lows.

Flat Simply blurs everything uniformly.

Tent Preserves the high and the lows better by making a linear falloff.

Quadratic TODO

Cubic Preserve the highs, but give an almost out-of-focus blur while smoothing sharp edges.

Gaussian TODO

Fast Gaussian An approximation of the Gaussian.

Catmull-Rom Catmull-Rom keeps sharp contrast edges crisp.

Mitch Preserve the highs, but give an almost out-of-focus blur while smoothing sharp edges.

Variable Size Allows a variable blur radius, if the size input is an image.

Bokeh The Bokeh button will force the blur node to use a circular blur filter. This gives higher quality results, but is slower than using a normal filter.

Gamma The Gamma button applies a gamma correction on the image before blurring it.

Relative Percentage Value of the blur radius relative to the image size.

Aspect Correction None, Y, X

X, Y Values set the ellipsoid radius in numbers of pixels over which to spread the blur effect.

Extend Bounds Allows the image, that is being blurred, to extend past its original dimension.

Outputs

Image Standard image output.

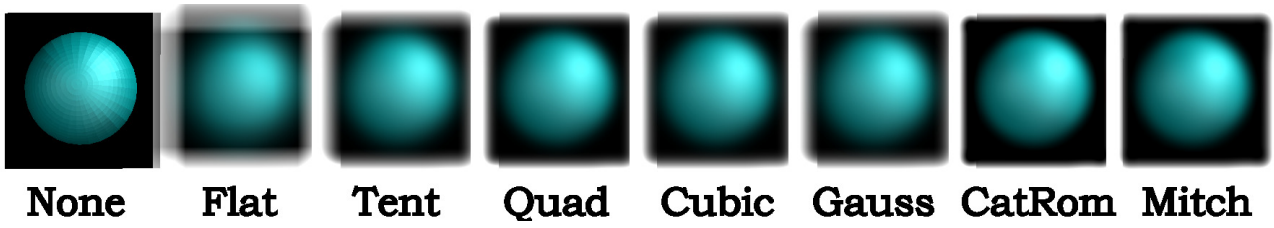


Fig. 2.2242: Blur node blur modes using 20% of image size as XY, no Bokeh/Gamma.

Example

An example blend-file, in fact, the one used to create the image above, is available here.. The blend-file takes one image from the Render Layer node "Blurs" and blurs it while offsetting it *Translate* and then combining it *Alpha Over* to build up the progressive sequence of blurs. Play with the Value and Multiply nodes to change the amount of blurring that each algorithm does.

Bokeh Blur Node

The Bokeh Blur node generates a bokeh type blur similar to Defocus. Unlike defocus an in-focus region is defined in the compositor. There is also more flexibility in the type of blur applied through the *Bokeh Image* node.

Several performance optimizations are also available such as OpenCL support, calculation area restriction and masking.

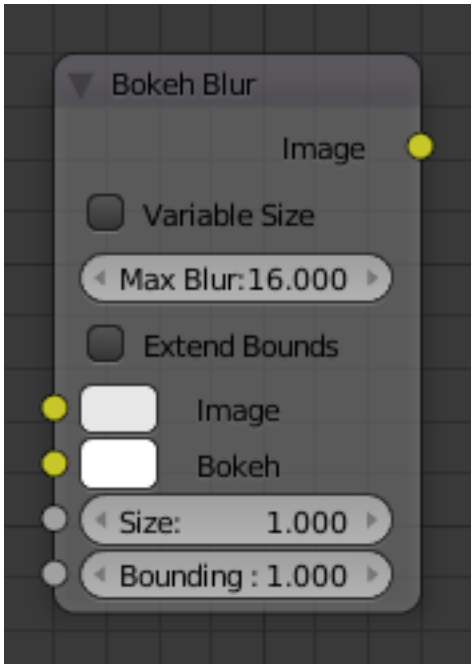


Fig. 2.2243: Bokeh Blur Node.

Inputs

Image Standard image input.

Bokeh This is an input for the *Bokeh Image* node.

Size Size controls the amount of blur. Size can either be a single value across the entire image or a variable value controlled by an input image. In order to use the latter, the Variable Size option must be selected. See the examples section below for more on how to use this.

Bounding Box This can be used with a *Box Mask* matte node or with a *Mask* input node to restrict the area of the image the blur is applied to. This could be helpful, for example, when developing a node system by allowing only a small area of the image to be filtered thus saving composite time each time adjustments are made.

Properties

Variable Size Allows a variable blur radius, if the Size input is an image.

Max blur Max blur is intended to act as an optimization tool by limiting the number of pixels across which the blur is calculated.

Outputs

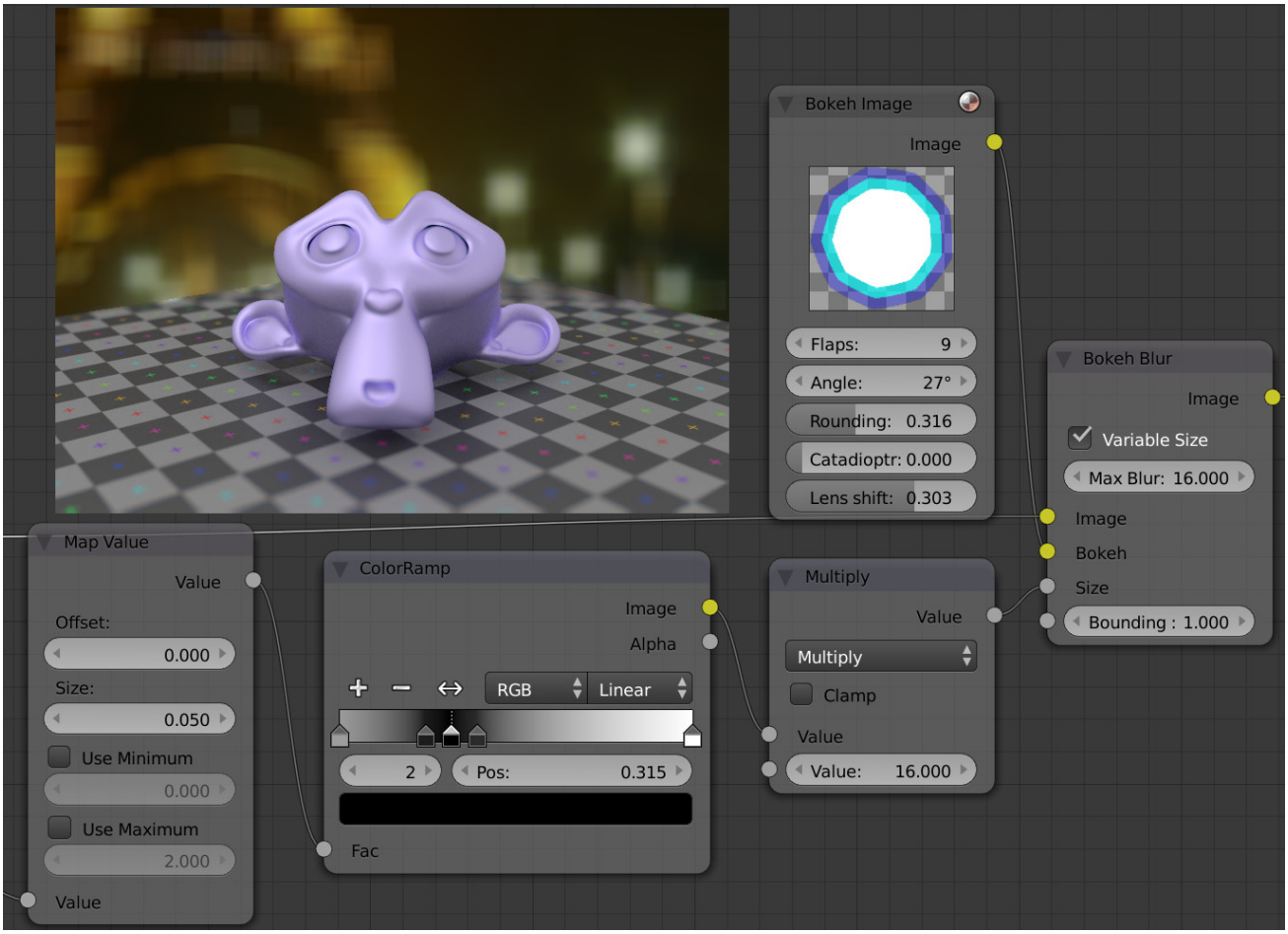
Image Standard image output.

Examples

Three examples of how the size input may be used follow.

An *ID masked* alpha image can be used so that a background is blurred while foreground objects remain in focus. To prevent strange edges the *Dilate Node* should be used.

The Z pass can be visualized using a *Map Value* node and *Color Ramp* node as described in *Render Layers*. A *multiply Math* node can be used following the color-ramp so that a blur value greater than one is used for objects outside the focal range.

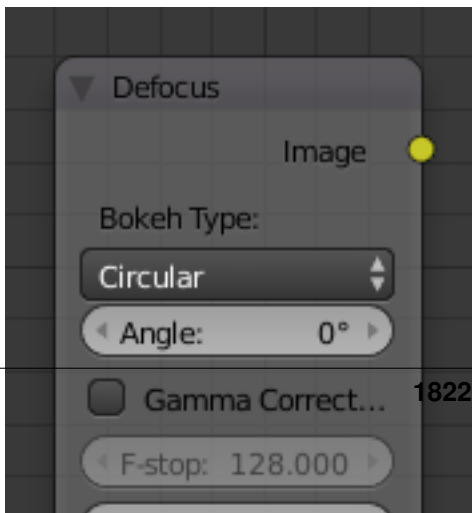


A manually created grayscale image can be used to define the sharp and blurry areas of a pre-existing image. Again, a Multiply Node can be used so that a blur value greater than one is used.

Defocus Node

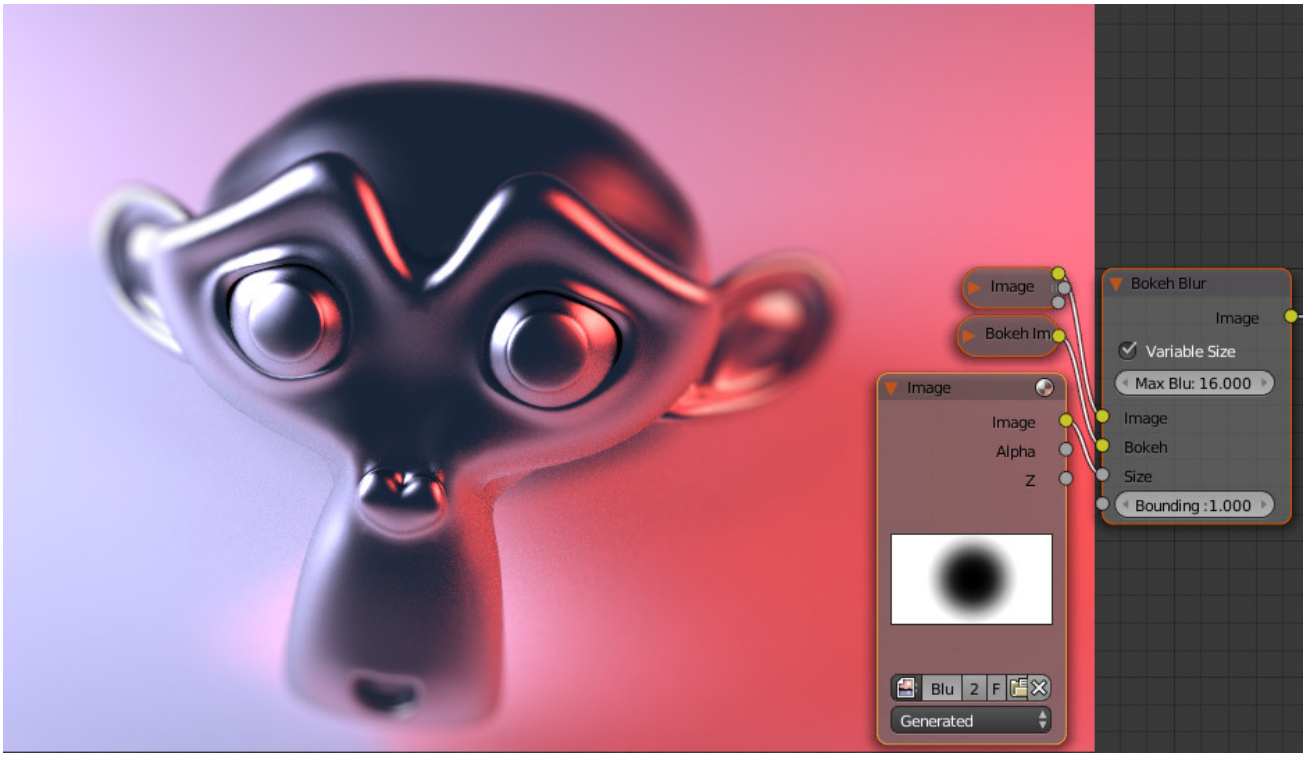
This node blurs areas of an image based on a map/mask input.

It is typically used to emulate depth of field (*DOF*) using a post-processing method with a Z-buffer input. But also allows to blur images that are not based on Z depth too.



Inputs

Image Standard image input.



Z Z-buffer input, but could also be a (grayscale) image used as a mask, or a single value input.

Properties

Bokeh Type The number of iris blades of the virtual camera's diaphragm.

Disk (to emulate a perfect circle) or Triangle (3 blades), Square (4 blades), Pentagon (5 blades), Hexagon (6 blades), Heptagon (7 blades) or Octagon (8 blades).

Angle This button is deactivated, if the Bokeh Type is set to Disk. It can be used to add a rotation offset to the Bokeh shape. The value is the angle in degrees.

Gamma Correction Applies a gamma correction on the image before and after blurring it.

F-Stop This option controls the amount of focal blur in the same way as a real camera. It simulates the aperture f of a real lens' iris, without modifying the luminosity of the picture. The default value 128 is assumed to be infinity: everything is in perfect focus. Half the value will double the amount of blur. This button is deactivated, if *No Z-buffer* is enabled.

Max Blur This value limits the amount of blur by setting a maximum blur radius. Could be used to optimize the performance. The default value of 0 means no limit.

Threshold Some artifacts, like edge bleed, may occur, if the blur difference between pixels is large. This value controls how large that blur difference considered to be safe.

Tip: Only change this value, if there is an occurring problem with an

in-focus object.

Preview If enabled a limited amount of (quasi-)random samples are used to render the preview. This way of sampling introduces additional noise, which will not show up in the final render.

Scene To select the linked scene.

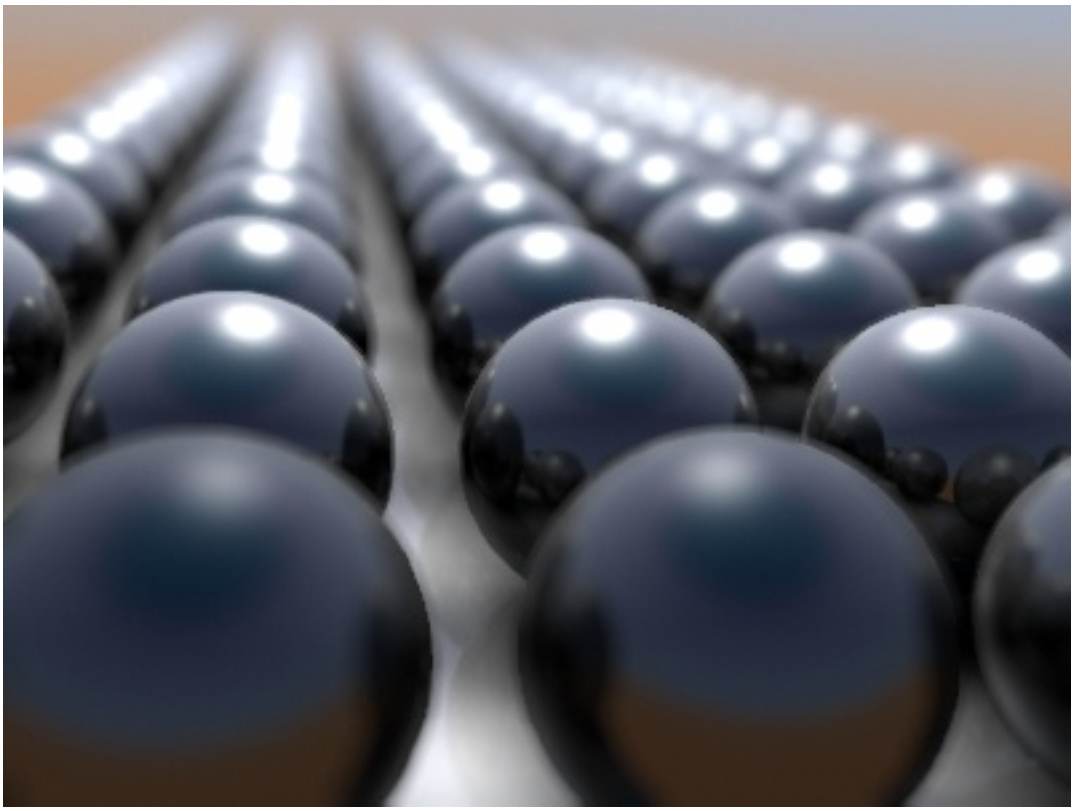
No Z-buffer Should be activate for a non Z-buffer in the Z input. No Z-buffer will be enabled automatically whenever a node that is not image based is connected to the Z input.

Z Scale Only active when *No Z-buffer* is enabled. When *No Z-buffer* is used, the input is used directly to control the blur radius (similar to *f-Stop* when using the Z-buffer). This parameter can be used to scale the range of the Z input.

Outputs

Image Standard image output.

Examples



In this blend-file example, the ball array image is blurred as if it was taken by a camera with a f-stop of 2.8 resulting in a fairly narrow depth of field centered on 7.5 Blender units from the camera. As the balls recede into the distance, they get blurrier. This node has no properties.

No Z-Buffer examples

Sometimes might want to have more control to blur the image. For instance, you may want to only blur one object while leaving everything else alone (or the other way around), or you want to blur the whole image uniformly all at once. The node, therefore, allows you to use something other than an actual Z-buffer as the Z input. For instance, you could connect an image node and use a grayscale image where the color designates how much to blur the image at that point, where white is the maximum blur and black is no blur. Or, you could use a Time node to uniformly blur the image, where the time value controls the maximum blur for that frame. It may also be used to obtain a possibly slightly better DoF blur, by using a fake depth shaded image instead of a Z-buffer. (A typical method to create the fake depth shaded image is by using a linear blend texture for all objects in the scene or by using the "fog/mist" fake depth shading method). This also has the advantage that the fake depth image can have anti-aliasing, which is not possible with a real Z-buffer.

The parameter *No Z-buffer*, becomes then the main blur control. The input has to be scaled, because usually the value of a texture is only in the numeric range 0.0 to 1.0.

Camera Settings

Fig. 2.2245: Distance setting in the Camera Depth of Field panel.

The *Defocus* node uses the actual camera data in your scene if supplied by a *Render Layer* node.

To set the point of focus, the camera now has a *Distance* parameter, which is shorthand for Depth of Field Distance. Use this camera parameter to set the focal plane of the camera (objects Depth of Field Distance away from the camera are in focus). Set *Distance* in the main *Camera* edit panel; the button is right below the *Depth of Field*.

To make the focal point visible, enable the camera *Limits* option, the focal point is then visible as a yellow cross along the view direction of the camera.

Hints

Preview In general, use preview mode, change parameters to your liking, only then disable preview mode for the final render. This node is computer intensive, so watch your console window, and it will give you status as it computes each render scan line.

Edge Artifacts For minimum artifacts, try to setup your scene such that differences in distances between two objects that may visibly overlap at some point are not too large.

"Focus Pull" Keep in mind that this is not real DoF, only a post-processing simulation. Some things cannot be done which would be no problem for real DoF at all. A typical example is a scene with some object very close to the camera, and the camera focusing on some point far behind it. In the real world, using shallow depth of field, it is not impossible for nearby objects to become completely invisible, in effect allowing the camera to see behind it. Hollywood cinematographers use this visual characteristic to to achieve the popular "focus pull" effect, where the focus shifts from a nearby to a distant object, such that the "other" object all but disappears. Well, this is simply not possible to do with the current post-processing method in a single pass. If you really want to achieve this effect, quite satisfactorily, here is how:

- Split up your scene into "nearby" and "far" objects, and render them in two passes.

- Now, combine the two the two results, each with their own "defocus" nodes driven by the same Time node, but with one of them inverted. (e.g. using a "Map Value" node with a Size of -1.) As the defocus of one increases, the defocus on the other decreases at the same rate, creating a smooth transition.

Aliasing at Low f-Stop Values At very low values, less than 5, the node will start to remove any oversampling and bring the objects at DoF Distance very sharply into focus. If the object is against a contrasting background, this may lead to visible stair-stepping (aliasing) which OSA is designed to avoid. If you run into this problem:

- Do your own OSA by rendering at twice the intended size and then scaling down, so that adjacent pixels are blurred together.
- Use the blur node with a setting of 2 for X and Y.
- Set DoF Distance off by a little, so that the object in focus is blurred by the tiniest bit.
- Use a higher f-Stop, which will start the blur, and then use the Z socket to a Map Value to a Blur node to enhance the blur effect.
- Rearrange the objects in your scene to use a lower-contrast background.

No Z-Buffer A final word of warning, since there is no way to detect if an actual Z-buffer is connected to the node, be **very** careful with the *No Z-Buffer* switch. If the *Z scale* value happens to be large, and you forget to set it back to some low value, the values may suddenly be interpreted as huge blur radius values that will cause processing times to explode.

Despeckle Node

The *Despeckle node* is used to smooth areas of an image in which noise is noticeable, while leaving complex areas untouched.

This works by the standard deviation of each pixel and its neighbors is calculated to determine if the area is one of high complexity or low complexity. If the complexity is lower than the threshold then the area is smoothed using a simple mean filter.

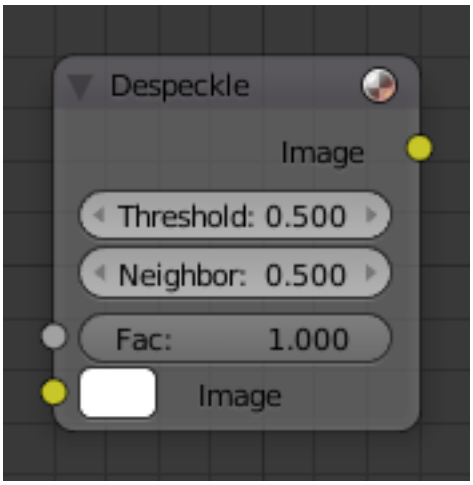


Fig. 2.2246: Despeckle Node.

Inputs

Factor Controls the amount the filter effects the image.

Image Standard image input.

Properties

Threshold The threshold to control high/low complexity.

Neighbor The threshold to control the number of pixels that must match.

Outputs

Image Standard image output.

Dilate/Erode Node

This node provides a morphology (mathematical shape analysis) filter.

Inputs

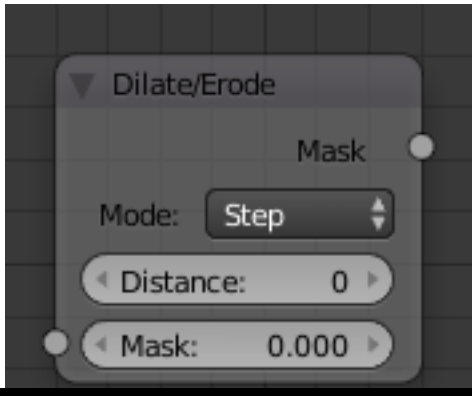
Mask Single color channel (or a black and white image) input.

Properties

Mode Step, Threshold, Distance, Feather

Distance The Distance is the filter radius. A *positive* value of Distance dilate (expands) the influence of a pixel on its surrounding pixels. A *negative* value erodes (shrinks) its influence.

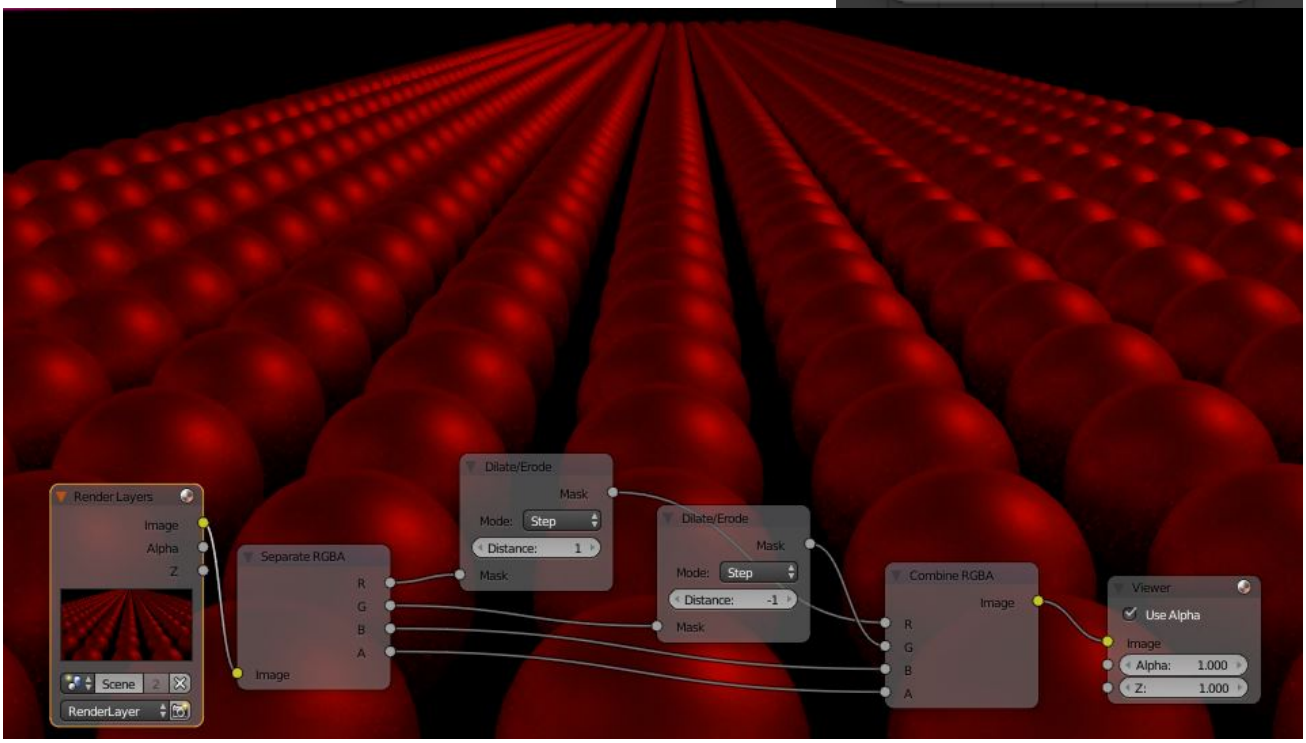
Outputs



Mask The filtered mask output.

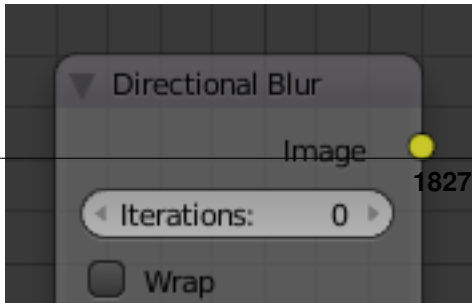
Example

In this example image, we wanted to take the rather boring array of ball bearings and spruce it up; make it hot, baby. So, we dilated the red and eroded the green, leaving the blue alone. If we had dilated both red and green... (hint: red and green make yellow). The amount of influence is increased by increasing the *Distance* values. Blend file available here..



Directional Blur Node

Blurs an image in a specified direction and magnitude. Can be used to fake motion blur.



Inputs

Image Standard image input.

Properties

Iterations Controls how may times the image is duplicated to create the blur effect. Higher values give smoother results.

Wrap Wraps the image on the X and Y axis to fill in areas, that become transparent from the blur effect.

Center X, Y Sets the position where the blur center is. This makes a difference if the angle, spin, and/or zoom are used.

Distance How large the blur effect is.

Angle Image is blurred at this angle from the center.

Spin Rotates the image each iteration to create a spin effect, from the center point.

Zoom Scales the image each iteration, creating the effect of a zoom.

Outputs

Image Standard image output.

Filter Node

The Filter node implements various common image enhancement filters.

Inputs

Factor Controls the amount of influence the node exerts on the output image.

Image Standard image input.

Properties

Type The Soften, Laplace, Sobel, Prewitt and Kirsch all perform edge-detection (in slightly different ways) based on vector calculus and set theory equations.

Soften Slightly blurs the image.

Sharpen Increases the contrast, especially at edges

Laplace Softens around edges

Sobel Creates a negative image that highlights edges

Prewitt Tries to do Sobel one better.

Kirsch Giving a better blending as Sobel or Prewitt, when approaching an edge.

Shadow Performs a relief emboss/ bumpmap effect, darkening outside edges.

Outputs

Image Standard image output.

Example

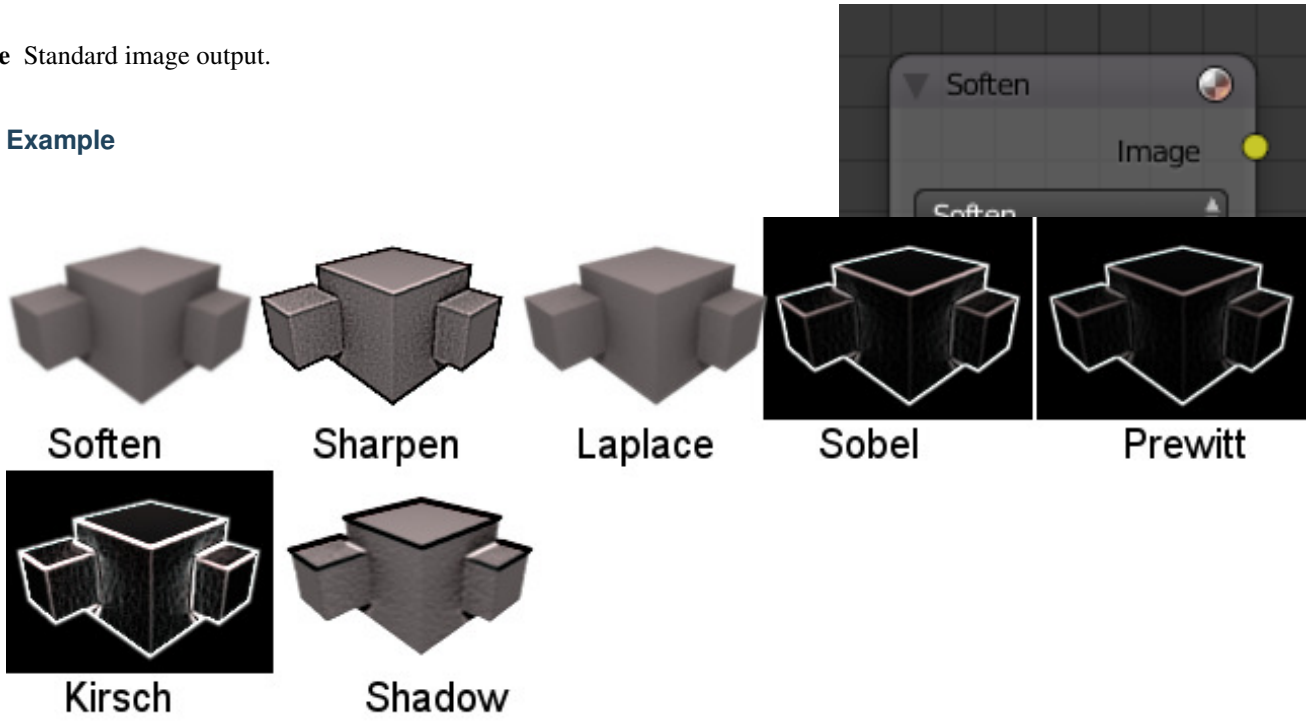


Fig. 2.2250: The Filter node has seven modes, shown here.

Glare Node

The *Glare node* is used add lens flares, fog, glows around exposed parts of an image an much more.

Fig. 2.2251: Glare Node.

Inputs

Image Standard image input.

Properties

Glare Type

Ghosts Creates a haze over the image.

Streaks Creates bright streaks used to simulate lens flares.

Streaks Total number of streaks.

Angle Offset The rotation offset factor of the streaks.

Fade Fade out factor for the streaks.

Fog Glow Looks similar to *Ghost* however, it is much smaller in size and gives more of a atmospheric haze or "glow" around the image.

Size Scale of the glow relative to the size of the original bright pixels.

Simple Star Works similar to *Streaks* but gives a simpler shape looking like a star.

Fade Fade out factor for the streaks.

Rotate 45 Rotate the streaks by 45°.

Common Options

Quality If not set to something other the *High*, then the glare effect will only be applied to a low resolution copy of the image. This can be helpful to save render times while only doing preview renders.

Iterations The number of times to run through the filter algorithm. Higher values will give more accurate results but will take longer to compute. Note, that this is not available for *Fog Glow* as it does not use an iterative based algorithm.

Color Modulation Used for *Streaks* and *Ghosts* to create a special dispersion effect. Johannes Itten describes this effect, Color Modulation, as subtle variations in tones and chroma.

Mix Value to control how much of the effect is added on to the image. A value of -1 would give just the original image, 0 gives a 50/50 mix, and 1 gives just the effect.

Threshold Pixels brighter than this value will be affected by the glare filter.

Outputs

Image Standard image output.

Example

TODO.

Inpaint Node

The *Inpaint node* is used to extend borders of an image into transparent or masked regions. This can be useful to solve problems like "wire removal" and holes created during chroma-keying.

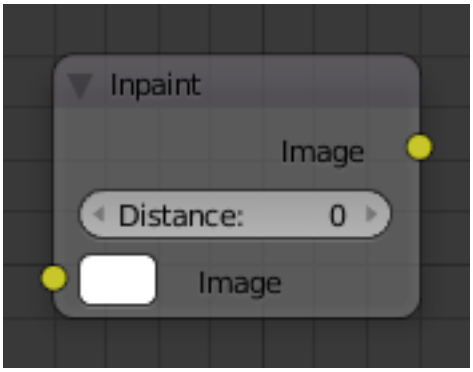


Fig. 2.2252: Inpaint Node.

Inputs

Image Standard image input.

Properties

Distance The number of times to extend the image.

Outputs

Image Standard image output.

Examples

In the left image shows the “wire” in place and after chroma-key has been applied you will see you’re left with a blank space – it’s shown as a black line here but it will be alpha in your Blender output.

Fig. 2.2253: Inpaint Node Example.

Inpainting fills in a couple of pixels using the surrounding image and voila... your wire is removed.

Note: The wider your “hole” is, the more noticeable this effect is! If you use more than a few pixels of infill, the effect is almost as irritating as the wire and your viewers won’t be impressed.

Inpainting can also cover up a multitude of other minor sins such as control points for motion capture: use it sparingly and it will amaze.

Pixelate Node

Add this node in front of a *scale node* to get a pixelated (non-smoothed) image from the resultant upscaled image.

Fig. 2.2254: Pixelate Node.

Inputs

Color Standard image input.

Properties

This node has no properties.

Outputs

Color Standard image output.

Example

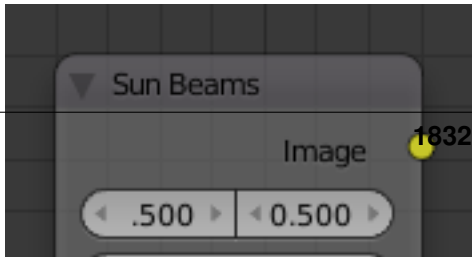
In the node editor, set the node tree to compositing in the header and check the *Use Nodes* checkbox. Add an input Image node and an output Viewer node. Connect the Input node to the viewer node and check the *Backdrop* checkbox in the header. Open an image you would like to pixelate using the open button on the image node. This image should now appear in the backdrop. Now add two scale nodes between the input and output *Add* → *Distort* → *Scale*. Change the values of X and Y to 0.2 in the first scale box and to 5 in the second. The background image will be unchanged.

Now add a Pixelate node between the two scale nodes.

Note: You can use `Alt-V` and `V` to zoom the backdrop in and out respectively.

Sun Beams Node

The Sun Beams node provides a computationally cheap way of creating the name giving effect based on the image brightness alone.



Sun Beams is a 2D effect for simulating the effect of bright light getting scattered in a medium (Crepuscular Rays). This phenomenon can be created by renderers, but full volumetric lighting is a rather arduous approach and takes a lot of render time.

Inputs

Image Standard image input.

Properties

Source width, height Source point of the rays as a factor of the image dimensions.

Ray length Length of the rays as a factor of the image size.

Outputs

Image Standard image output.

Example

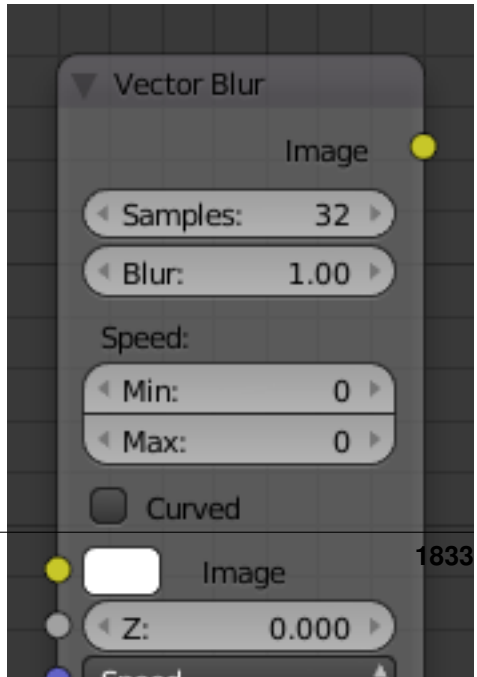
Usually, the first step is to define the area from which rays are cast. Any diffuse reflected light from surfaces is not going to contribute to such scattering in the real world, so should be excluded from the input data. Possible ways to achieve this are:

- Entirely separate image as a light source.
- Brightness/contrast tweaking to leave only the brightest areas.
- Muting shadow and midtone colors, which is a bit more flexible.
- Masking for ultimate control.

After generating the sun beams from such a light source image they can then be overlayed on the original image. Usually, a simple "Add" mix node is sufficient, and physically correct because the scattered light adds to the final result.

Vector (Motion) Blur Node

The Vector Blur node applies a **non** physically based method of simulating *Motion blur*. It uses the vector speed render pass to blur the image pixels in 2D.



Inputs

Image Standard image input.

Z Standard Z depth.

Speed Input for the "Vector" render pass. See *Cycles render passes* or *Blender internal render passes*.

Properties

Samples Quality factor.

Blur Scaling factor for the motion vector (actually the "shutter speed" in frames).

Speed The vector blur could produce artifacts like streaks, lines and other. To combat these problems, the filter applies clamping, which can be used to limit which pixels get blurred. The speed is set in pixel units.

Maximum Speed The maximum threshold. The majority of artifacts are caused by pixels moving too fast.

Minimum Speed The minimum threshold for moving pixels can separate the hardly moving pixels from the moving ones. Especially when the camera itself moves, the vector mask can become the entire image.

Outputs

Image Standard image output.

Hint: You can make vector blur results in a little smoother by passing the Speed pass through a blur node (but note that this can make strange results, so it is only really appropriate for still images with lots of motion blur).

Note: Does not work when reading from a multilayer OpenEXR sequence set

---

Vector Nodes

These nodes can be used to manipulate various types of vectors, such as surface normals and speed vectors.

Map Range Node

This node allows to convert (map) an input value range into a destination range. By default, values outside the specified input range will be proportionally mapped as well. This node is similar to *Map Value* node but provides a more intuitive way to specify the desired output range.

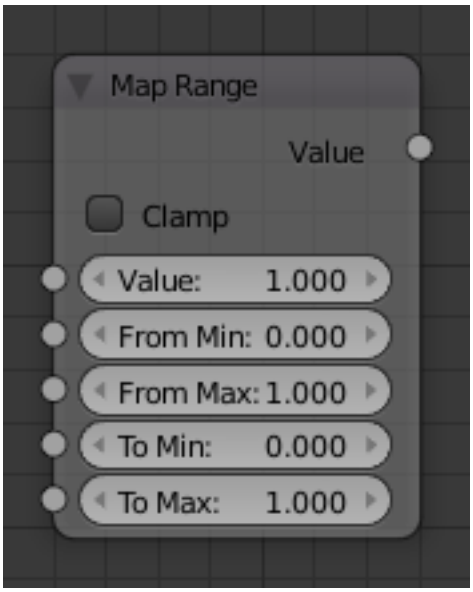


Fig. 2.2257: Map Range Node.

Inputs

Value Standard value input.

From Min/Max Start/End of the input value range.

To Min/Max Start/End of the destination range.

Properties

Clamp Clamps values to Min/Max of the destination range.

Outputs

Value Standard value output.

Usage

One important use case is to easily map the Z-depth channel from its original range to a more usable range (i.e.: 0.0 - 1.0) for use as a matte for colorization or filtering operations.

Map Value Node

Map Value node is used to scale, offset and clamp values.

Inputs

Value Standard Value input. (value refers to each vector in the set).

Properties

Offset Factor added to the input value.

Size Scales (multiply) the input value.

Use Minimum, Maximum Enable this to activate their related operation.

Min, Max Defines a range between minimum and maximum to *clamp* the input value to.

Outputs

Value Standard value output.

Example

Z Depth map

This is particularly useful in achieving a depth-of-field effect, where the Map Value node is used to map a Z value (which can be 20 or 30 or even 500 depending on the scene) to the range between (0 to 1), suitable for connecting to a Blur node.

Multiplying values

This node can also be used the map value node to multiply values to achieve a desired output value. In the mini-map to the right, the Time node outputs a value between 0.0 and 1. 00 evenly scaled over 30 frames. The *first* Map Value node multiplies the input by 2, resulting in an output value that scales from 0.0 to 2.0 over 30 frames. The *second* Map Value node subtracts 1 from the input, giving working values between (-1.00 to 1.0), and multiplies that by 150, resulting in an output value between (-150 to 150) over a 30-frame sequence.

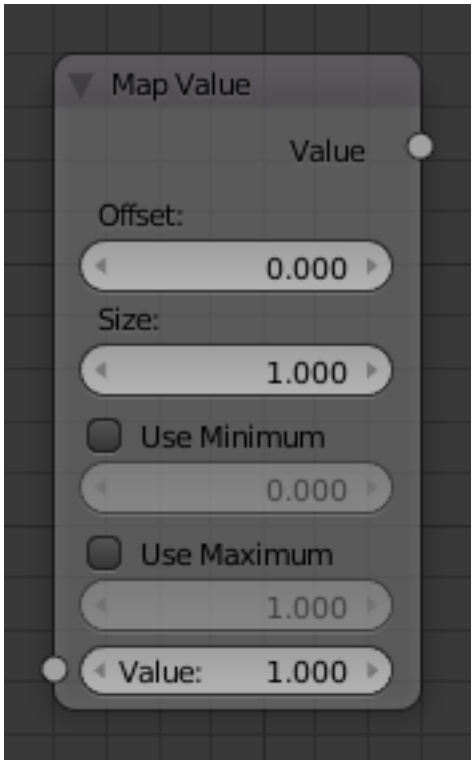


Fig. 2.2258: Map Value Node.

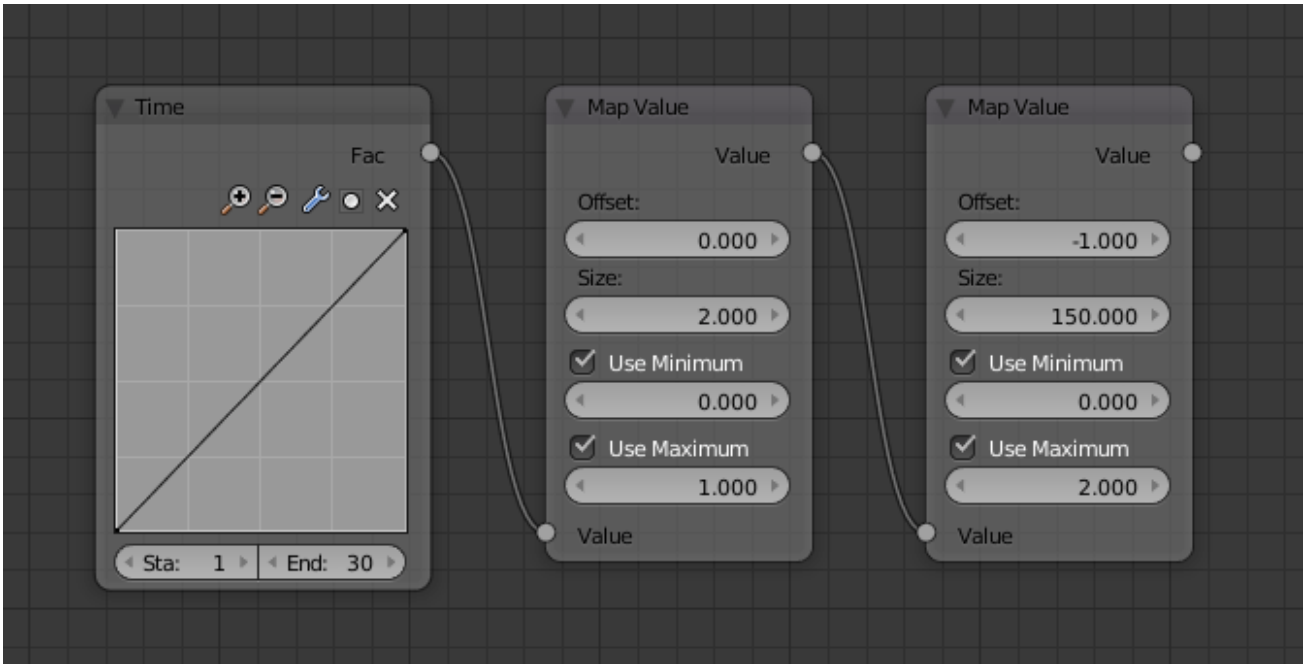


Fig. 2.2259: Using Map Value to multiply

Normal Node

The Normal node generates a normal vector and a dot product.

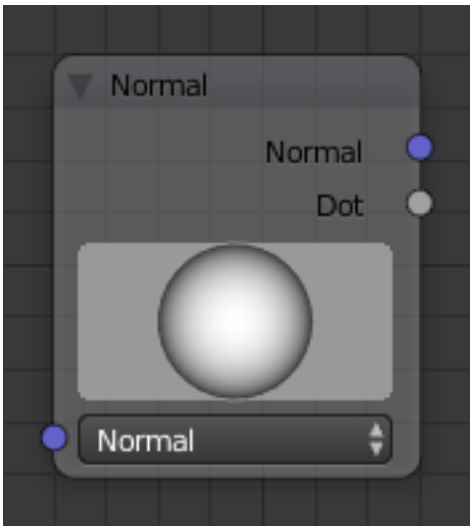


Fig. 2.2260: Normal Node.

Inputs

Normal Normal vector input.

Properties

Normal Direction To manually set a fixed normal direction vector. `LMB` click and drag on the sphere to set the direction of the normal.

Outputs

Normal Normal vector output.

Dot Dot product output. The dot product is a scalar value.

- If two normals are pointing in the same direction the dot product is 1.
- If they are perpendicular the dot product is zero (0).
- If they are antiparallel (facing directly away from each other) the dot product is -1.

Normalize Node

Normalizing a vector scales its magnitude, or length, to a value of 1, but keeps its direction intact.

Inputs

Value Standard value input.

Properties

This node has no properties.

Outputs

Value Standard value output.

Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.

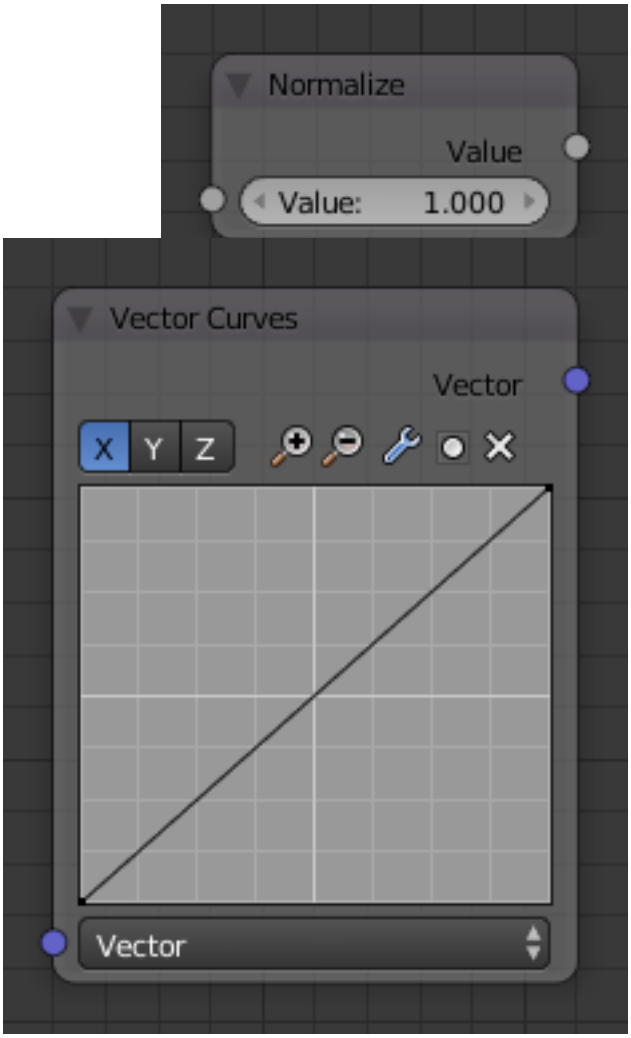


Fig. 2.2262: Vector Curves Node.

Inputs

In the shader context the node also has an additional Factor property.

Factor Controls the amount of influence the node exerts on the output vector.

Vector Standard vector input.

Properties

Channel X, Y, Z

Curve For the curve controls see: *Curve widget*.

Outputs

Vector Standard vector output.

Matte Nodes

These nodes give you the essential tools for creating a *Matte* for images that do not already have their own *Alpha Channel*. One usage scenario is blue-screen or green-screen footage, where live action is shot in front of a blue or green backdrop for replacement by a matte painting or virtual background.

In general, hook up these nodes to a viewer, set your UV/Image Editor to show the viewer node, and play with the sliders in real-time using a sample image from the footage, to get the settings right. In some cases, small adjustments can eliminate artifacts or foreground image degradation. Taking

out too much green can result in foreground actors looking flat or blueish/purplish.

You can and should chain these nodes together, improving your masking and color correction in successive refinements, using each node's strengths to operate on the previous node's output. *Keying Node* is the closest to a "does-it-all" node for green screens, but the best results stem from a combination of techniques.

Note: Garbage Matte is not a node, but a technique selecting what to exclude from an image. It is a *Mask* used to identify content to be removed from an image that cannot be removed by an automatic process like chroma keying. It is used either to select specific content to be removed, or it is the inverse of a rough selection of the subject; removing everything else.

Some nodes accept a garbage matte directly. For those that don't, you can still apply one by subtracting the garbage matte from the matte generated by the node.

Simple garbage mattes can be created with the *Box Mask* or *Ellipse Mask* More complicated matte shapes using *Double Edge Mask* or using a *Mask*.

Box Mask Node

The *Box Mask* node creates an image suitable for use as a simple matte.

Inputs

Mask An optional mask to use as the base for mask operations.

Value Intensity of the generated mask.

Properties

X, Y Position of the center of the box as a fraction of the total width or height. (0.5, 0.5 creates a centered box; 0.0, 0.0 creates a box in the lower left).

Width Width of the box as a fraction of the total image width.

Height Height of the box as a fraction of the total image *width*, not height.

Rotation Rotation of the box around its center point.

Mask Type Operation to use against the input mask.

Add This yields the *union* of the input mask and the generated mask: Areas covered by the generated mask are set to the specified *Value*. Other parts of the input masked are passed through unchanged, or set to black if there is no input mask.

Subtract Values of the input mask have the specified *Value* subtracted from them.

Multiply This yields the *intersection* of this generated mask and the input mask: Values of the input mask are multiplied by the specified *Value* for the area covered by the generated mask. All other areas become black.

Not Any area covered by both the input mask and the generated mask becomes black. Areas covered by the generated mask that are black on the input mask become the specified *Value*. Areas uncovered by the generated mask remain unchanged.

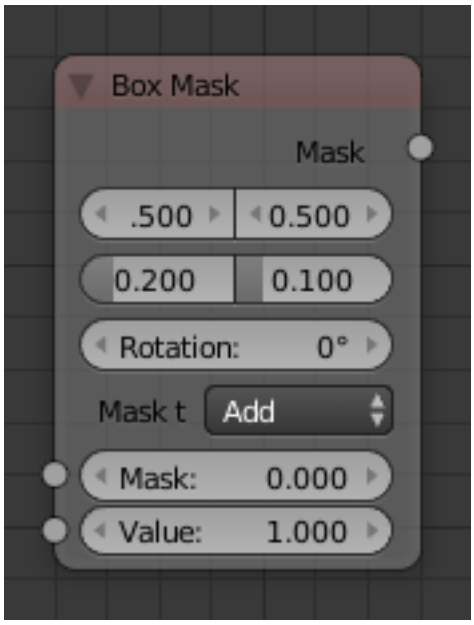


Fig. 2.2263: Box Mask Node.

Outputs

Mask A generated rectangular mask merged with the input mask. The created mask is the size of the current scene render dimensions.

Tip: For soft edges, pass the output mask through a slight *blur node*.

Channel Key Node

The *Channel Key* node determines background objects from foreground objects by the difference in the selected channel's levels.

For example in YUV color space, this is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.

Fig. 2.2264: Channel Key Node.

Inputs

Image Standard image input.

Properties

Color Space This button selects what color space the channels will represent.

RGB, HSV, YUV, YCbCr

Channel This button selects the channel, defined by the Color Space, to use to determine the matte.

Algorithm Max., Single

Limit It is possible to have a separation between the two values to allow for a gradient of transparency between foreground and background objects.

High Determines the lowest values that are considered foreground. (which is supposed to be – relatively – height values: from this value to 1.0).

Low Determines the highest values that are considered to be background objects. (which is supposed to be – relatively – low values: from 0.0 to this value).

Outputs

Image Image with an alpha channel adjusted for the keyed selection.

Matte A black and white alpha mask of the key.

Chroma Key Node

The *Chroma Key* node determines if a pixel is a foreground or background (and thereby should be transparent) based on its chroma values.

Use this, for example, to composite images that have been shot in front of a green or blue screen.

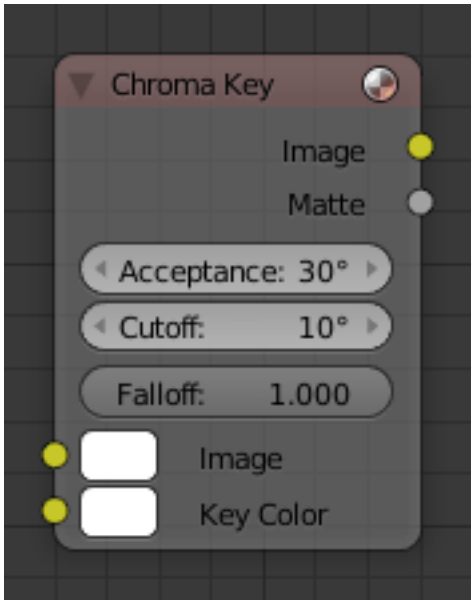


Fig. 2.2265: Chroma Key Node.

Inputs

Image Standard image input.

Key Color The background color usually selected using the color picker and the original image.

Properties

Acceptance An angle on the color wheel that represents how tolerant the keying color is. Larger angles allow for larger variation in the keying color to be considered background pixels.

Cutoff Controls the level that is considered the pure background. Higher cutoff levels mean more pixels will be 100% transparent if they are within the angle tolerance.

Falloff Increase to make nearby pixels partially transparent producing a smoother blend along the edges.

Outputs

Image Image with its alpha channel adjusted for the keyed selection.

Matte A black and white alpha mask of the key.

Color Key Node

The color key node creates a matte based on a specified color of the input image.

Inputs

Image Standard image input.

Properties

Color The sliders represent threshold values. Higher values in this node's context mean a wider range of colors from the specified will be added to the matte.

Hue, Saturation, Value

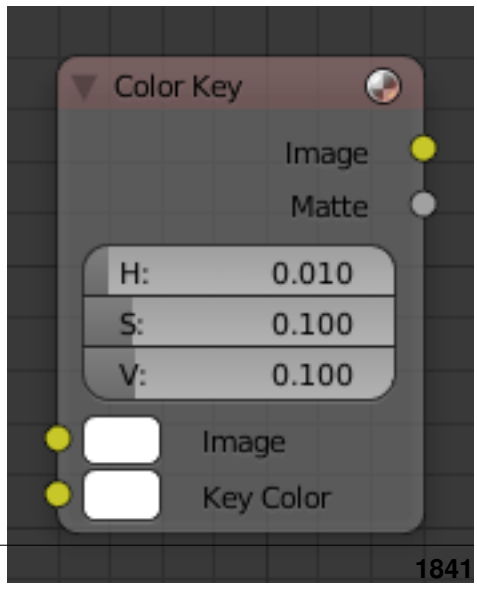


Fig. 2.2266: Color Key Node.

Outputs

Image Image with its alpha channel adjusted for the keyed selection.

Matte A black and white alpha mask of the key.

Color Spill Node

The *Color Spill* node reduces one of the RGB channels so that it is not greater then any of the others.

This is common when compositing images that were shot in front of a green or blue screen. In some cases, if the foreground object is reflective, it will show the green or blue color; that color has "spilled" onto the foreground object. If there is light from the side or back, and the foreground actor is wearing white, it is possible to get "spill" green (or blue) light from the background onto the foreground objects, coloring them with a tinge of green or blue. To remove the green (or blue) light, you use this fancy node.

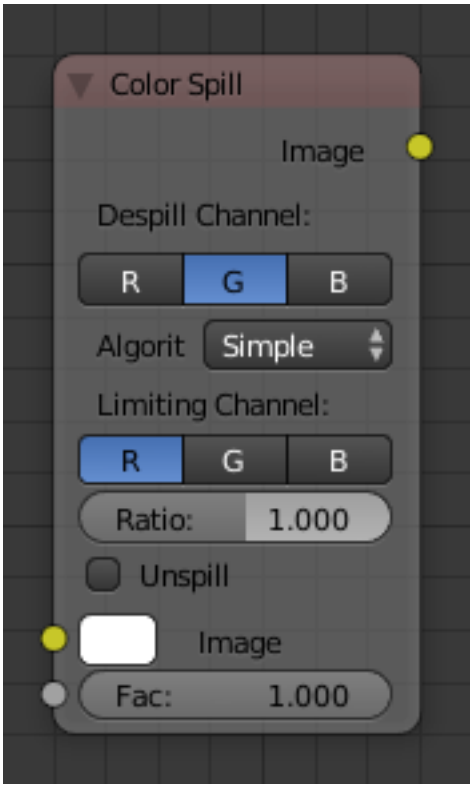


Fig. 2.2267: Color Spill Node.

Inputs

Image Standard image input.

Factor Standard Factor.

Properties

Despill Channel R, G, B

Algorithm Simple, Average

Limiting Channel R, G, B

Ratio Scale limit by value

Unspill Allows you to reduce the selected channel's input to the image greater than the color spill algorithm normally allows. This is useful for exceptionally high amounts of the color spill.

R, G, B

Outputs

Image The image with the corrected channels.

Example

Results with the nodes applied to an image from the Mango Open Movie.

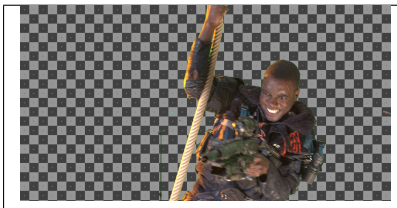


Fig. 2.2268: Before: green border and green reflections.

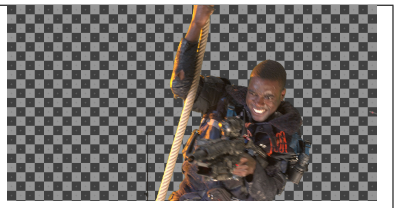


Fig. 2.2269: After: no unwanted geen.

Difference Key Node

This node produces a matte that isolates foreground content by comparing it with a reference background image.

Inputs

Image Contains foreground content against the background that is to be removed.

Image The reference background image.

Properties

Tolerance Where pixels match the reference background to within the specified threshold, the matte is made transparent.

Falloff Increase to make nearby pixels partially transparent producing a smoother blend along the edges.

Outputs

Image Image with its alpha channel adjusted for the keyed selection.

Matte A black and white alpha mask of the key.

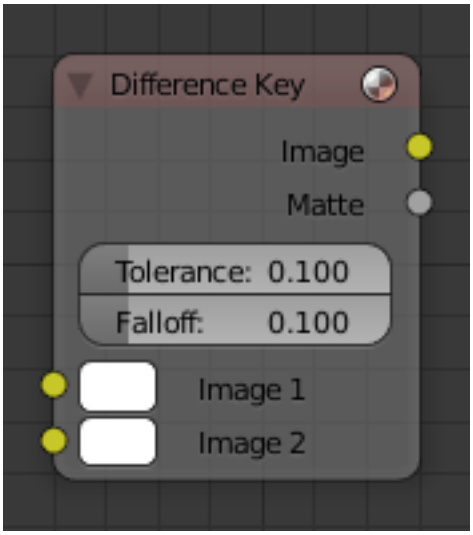


Fig. 2.2270: Difference Key Node.

Distance Key Node

The Distance Key node determines a pixel's alpha value based on the three-dimensional distance between the image pixel color and the key color in a 3D color space.

This key works well when trying to single out a specific color in a background (not necessarily green).

Inputs

Image Standard image input.

Key Color The color that is to be keyed.

Properties

Tolerance A threshold what the node considers a match between the key color and the foreground pixel. The tolerance affects how close a pixel needs to be to the background pixel to be considered an absolute match.

Falloff When the Falloff value is high, pixels that are close to the Key Color are more transparent than pixels that are not as close to the Key Color (but still considered close enough to be keyed). When the Falloff value is low, it does not matter how close the pixel color (Image) is to the Key Color, it is transparent.

Color Space It is also possible to work with YCbCr color space, but only the Cb and Cr channels are taken into consideration for determining the distance between the foreground and background pixels.

RGB, YCC

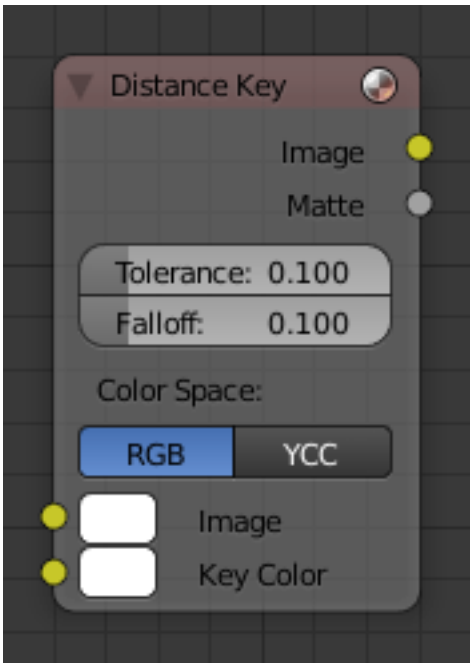


Fig. 2.2271: Distance Key Node.

Outputs

Image The image with an alpha channel adjusted for the keyed selection

Matte A black and white alpha mask of the key.

Double Edge Mask Node

The *Double Edge Mask* node creates a gradient between two masks.

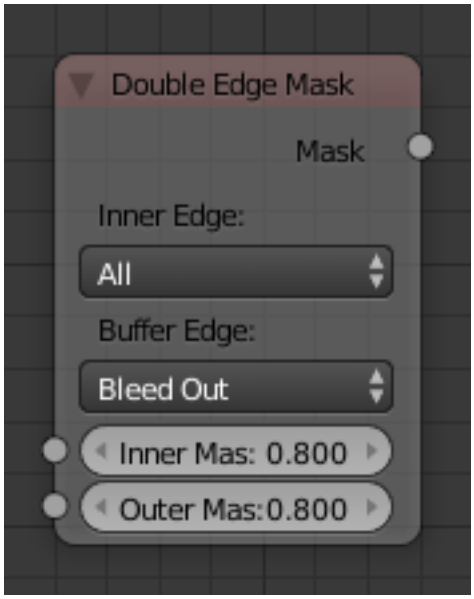


Fig. 2.2272: Double Edge Mask Node.

Inputs

Inner Mask A mask representing the inside shape, which will be fully white.

Outer Mask A mask representing the outside shape, which will fade from black at its edges to white at the *Inner Mask*.

Properties

Inner Edge

All All shapes in the *Inner Mask* contribute to the gradient, even ones that do not touch the *Outer Mask* shape.

Adjacent Only Only shapes in the *Inner Mask* that overlap with the *Outer Mask* contribute to the gradient.

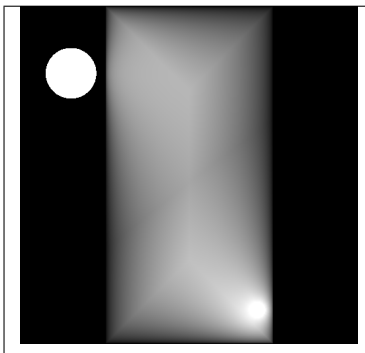


Fig. 2.2273: All.

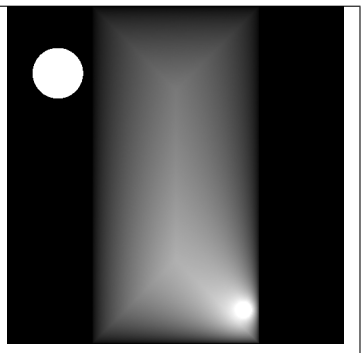


Fig. 2.2274: Adjacent Only.

Buffer Edge

Keep In Parts of the *Outer Mask* that touch the edge of the image are treated as if they stop at the edge.

Bleed Out Parts of the *Outer Mask* that touch the edge of the image are extended beyond the boundary of the image.

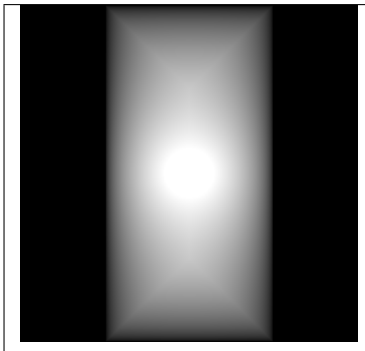


Fig. 2.2275: Keep In.

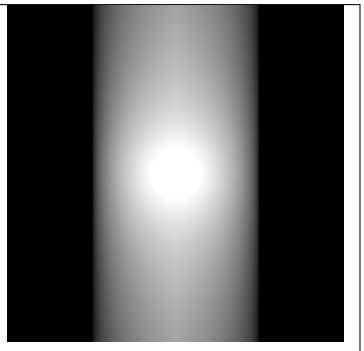


Fig. 2.2276: Bleed Out.

Outputs

Mask Standard mask output.

Demo Video

Ellipse Mask Node

The *Ellipse Mask* node creates an image suitable for use as a simple matte or vignette mask.

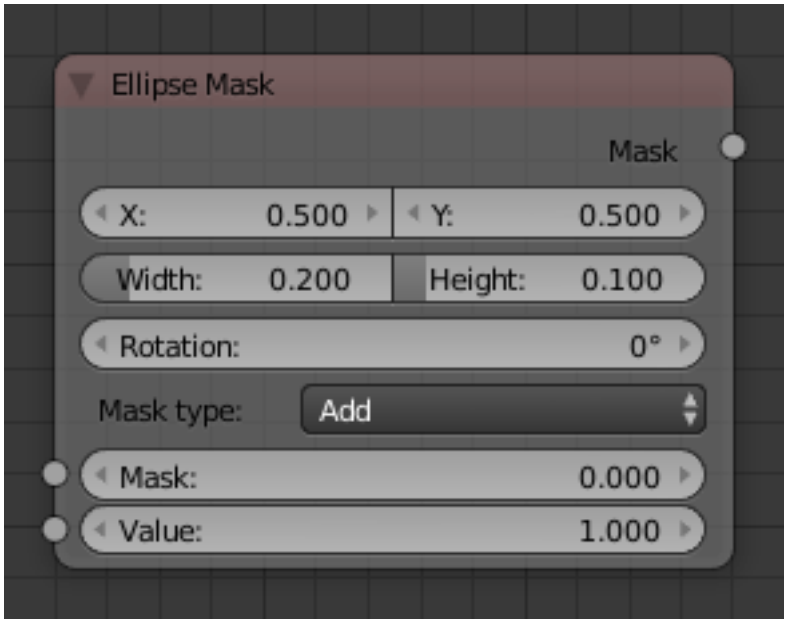


Fig. 2.2277: Ellipse Mask Node.

Inputs

Mask An optional mask to use as the base for mask operations.

Value Intensity of the generated mask.

Properties

X, Y Position of the center of the ellipse as a fraction of the total width or height. (0.5, 0.5 creates a centered ellipse; 0.0, 0.0 creates an ellipse with its center in the lower left).

Width Width of the ellipse as a fraction of the total image width.

Height Height of the ellipse as a fraction of the total image *width*, not height. Equal *Width* and *Height* values with produce a circle.

Rotation Rotation of the ellipse around its center point.

Mask Type Operation to use against the input mask.

Add This yields the *union* of the input mask and the generated mask: Areas covered by the generated mask are set to the specified *Value*. Other parts of the input masked are passed through unchanged, or set to black if there is no input mask.

Subtract Values of the input mask have the specified *Value* subtracted from them.

Multiply This yields the *intersection* of this generated mask and the input mask: Values of the input mask are multiplied by the specified *Value* for the area covered by the generated mask. All other areas become black.

Not Any area covered by both the input mask and the generated mask becomes black. Areas covered by the generated mask that are black on the input mask become the specified *Value*. Areas uncovered by the generated mask remain unchanged.

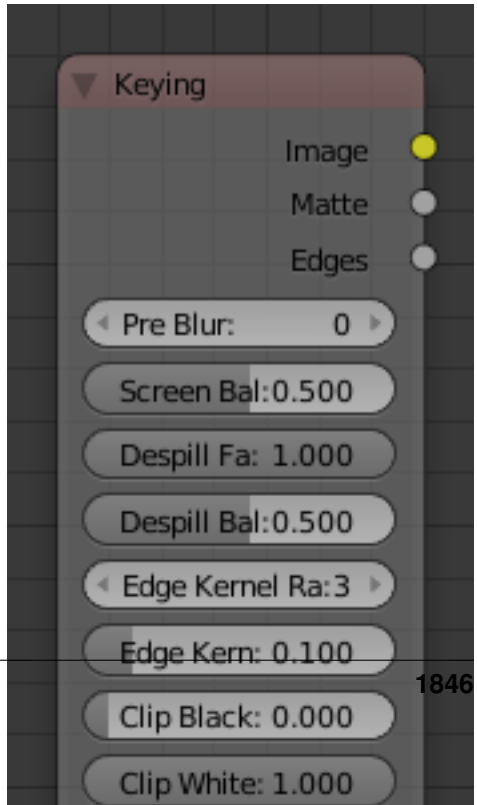
Outputs

Mask A generated elliptical mask merged with the input mask. The created mask is the size of the current scene render dimensions.

Tip: For soft edges, pass the output mask through a slight *blur node*. For a vignette, pass the output of this through a heavy blur.

Keying Node

The *Keying* node is an one-stop-shop for "green screen" / "blue screen" removal. It performs both chroma keying to remove the backdrop and despill to correct color cast from the backdrop. Additionally, you can perform common operations used to tweak the resulting matte.



Inputs

Image Standard image input.

Key Color The color of content to be removed. This may be a single color using the, or a reference image such as generated by the *Keying Screen Node*.

Garbage Matte An optional mask of area(s) to always *exclude* from the output. This is removed from the chroma key generated matte.

Core Matte An optional mask of area(s) to always *include* in the output. This is merged with the chroma key generated matte.

Properties

Pre Blur Reduce the effects of color noise in the image by blurring only color by the given amount, leaving luminocity intact. This will affect matte calculation only, not the result image.

Screen Balance This is the balance between color channels compared with the key color. 0.5 will average the other channels (red and blue in the case of a green screen).

This may be tweaked in tandem with *Clip Black* and *Clip White* while checking the *Matte* output to create a mask with optimal separation.

Despill Factor Controls how much color bleed from the key color is removed from the input image: 0 means no despilling, 1 means all possible spilling will be removed. The underlying implementation is the same as adjusting the *Unspill* amount of the *Color Spill Node*.

Despill Balance This controls how the color chanels are compared when computing spill, affecting the hue and shade of the corrected colors. It is similar to setting the *Limiting Channel* in the *Color Spill Node*.

Edge Kernel Radius Defines the radius in pixel used to detect an edge.

Edge Kernel Tolerance Defines threshold used to check if pixels in radius are the same as current pixel: If the difference between pixel colors is higher than this threshold then the point will be considered an edge.

Clip Black This sets the threshold for what becomes fully transparent in the output (black in the matte). It should be set as low as possible. Uneven backdrops will require this value to be increased. Use of the *Keying Screen Node* can help keep this value low. You may also use a *Garbage Matte* to exclude problematic areas.

This value does not impact areas detected as edges to ensure edge detail is preserved.

Clip White This sets the threshold for what becomes fully opaque in the output (white in the matte). It should be set as high as possible. Colors close to green in the foreground may require lowing this and/or adjusting the *Screen Balance*. Particularly problematic parts can fixed with a *Core Matte* instead of a low *Clip White*.

This value does not impact areas detected as edges to ensure edge detail is preserved.

Dilate/Erode Enlarge (positive numbers) or shrink (negative numbers) the matte by the specified number of pixels. This is similar to using the *Dilate/Erode Node* on the matte.

This a simple way to include more or less along the edges of the matte, particularly combined with *Post Blur*.

Feather Falloff The rate of fall off at the edges of the matte when feathering, to manage edge detail.

Feather Distance Controls how much the matte is feathered inwards (negative number) or outwards (positive number).

Post Blur Make the matte less sharp, for smoother transitions to the background and noise reduction.

Outputs

Image Processed image with the *Matte* applied to the images's *alpha channel*.

Matte Output matte to use for checking the quality of the key, or to manually apply using a *Set Alpha Node* or *Mix Node*.

Edges Shows what edges were detected on the matte. Useful for adjusting the *Edge Kernel Radius* and *Edge Kernel Tolerance*.

Tip: If there are problems with the edges of the matte, it may help to start with adjusting the *Edge Kernel* parameters before adjusting feathering. Detected edges are not subject to *Clip Black* / *Clip White* thresholds to preserve fine edge detail. You can check edge detection by connecting a *Viewer Node* to the *Edges* output.

Sharper detected edges (smaller *Edge Kernel Radius*, like 2 / larger *Edge Kernel Tolerance*, like 0.4) will create a sharper matte, but may loose some detail like stray hairs. A sharp matte is good, but disappearing or flickering hairs are distracting.

Fat edges (larger *Edge Kernel Radius*, like 8 / smaller *Edge Kernel Tolerance*, like 0.05) will capture more edge detail, but may also produce a halo around the subject. The halo can be adjusted with *Feather* controls along with *Dilate/Erode*.

Keying Screen Node

The *Keying Screen* node creates plates for use as a color reference for keying nodes. It generates gradients from sampled colors on motion tracking points on movie clips.

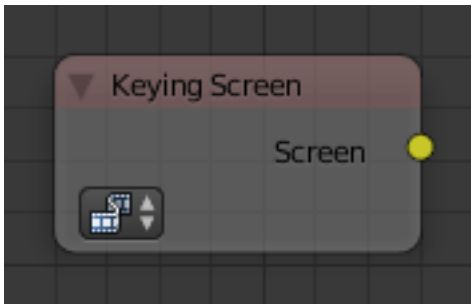


Fig. 2.2279: Keying Screen Node.

Example

Consider a node setup for green screen removal, using a *Color Key*:

Often, lighting is uneven across the backdrop.

That can result in a bad matte.

If you increase the tolerances on the keying node, it will accept mores shades of green to mask out. But it may also incorrectly mask out more of the foreground.

Instead of increasing the range of accepted shades to be masked out, the *Keying Screen* node lets you change what shade of green (or other color) to use for different parts of the image.

Start in the *Movie Clip Editor*. Open the Properties Region and Tool Shelf to show tracking configuration. Tracks used for gradients are not useful for camera solving, because they do not track well. So create a new object track in the *Objects* selector. Place tracking markers on the clip to sample different parts of the backdrop.

These tracks may be tracked or moved manually, so gradients can be updated over time. If the marker is not enabled for a frame, it will not be used creating the gradient. (Such as the red-colored marker on the arm in the screen shot above)

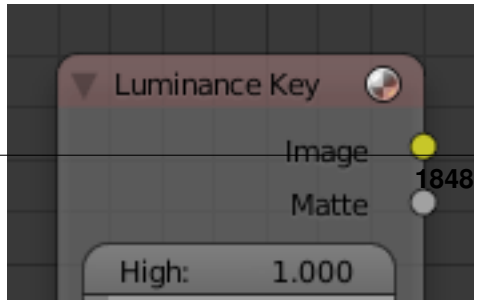
Once the tracks are created, add the node to your compositing setup, and select the tracking object used for the backdrop.

The resulting image now has a better matte.

Luminance Key Node

The *Luminance Key* node determines background objects from foreground objects by the difference in the luminance (brightness) levels.

Stock footage of explosions, smoke or debris are normally shot against a solid, dark background rather than a green screen. This node can separate the



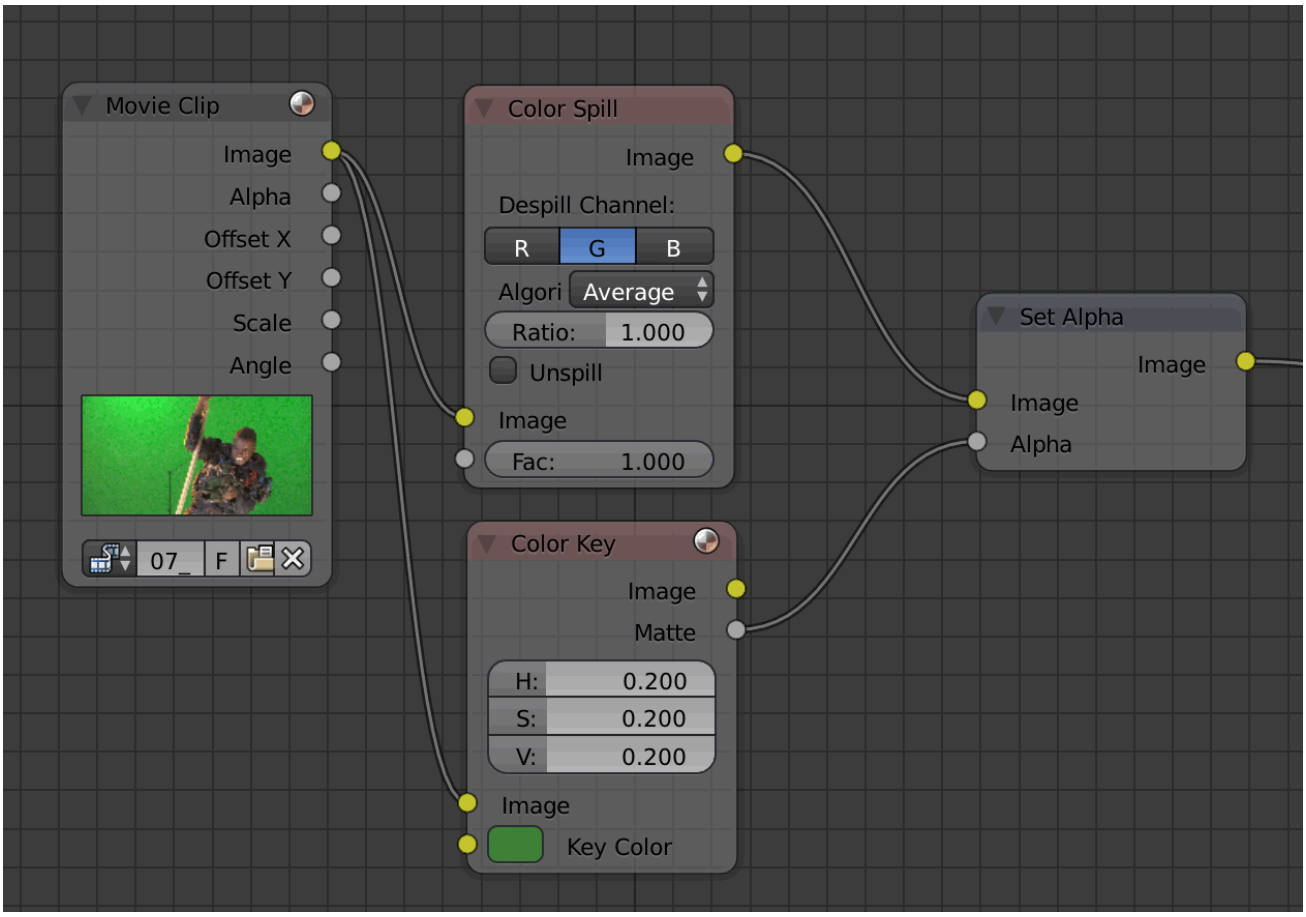


Fig. 2.2280: Example from the Mango Open Movie, Tears of Steel.

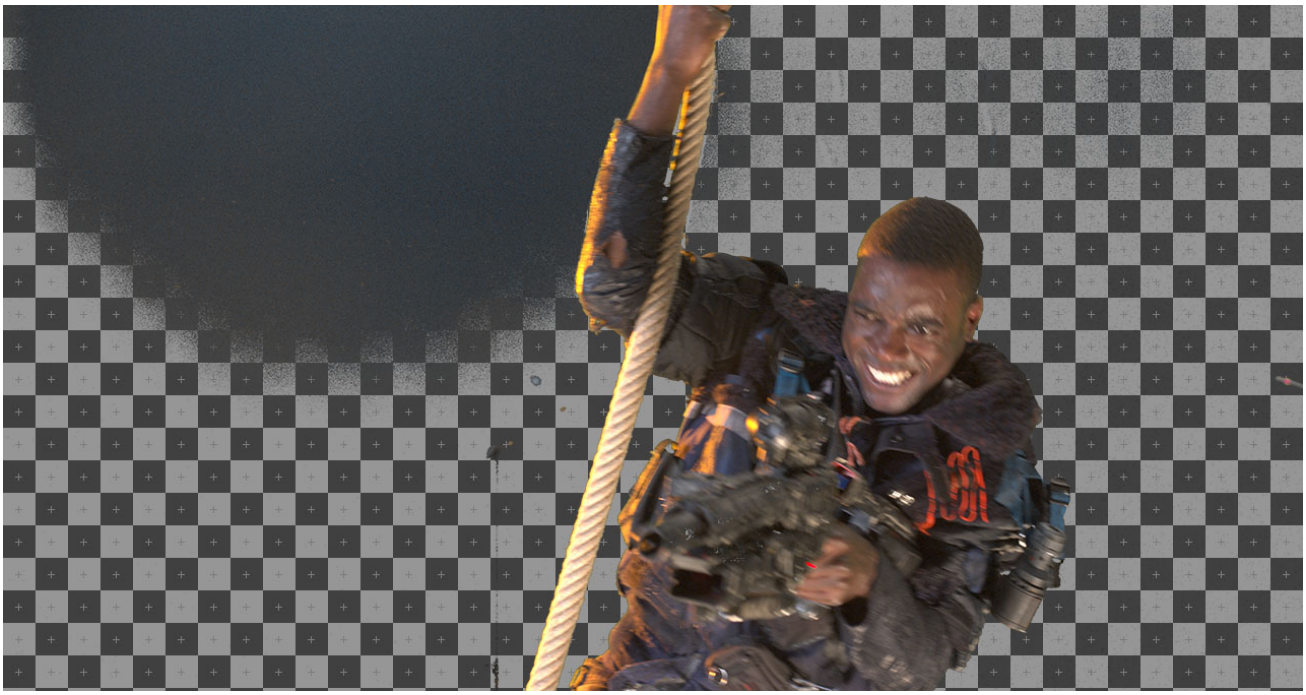


Fig. 2.2281: Exanple of a poor mask: Some of the backdrop is opaque, and some parts of the gun in the foreground are transparent.

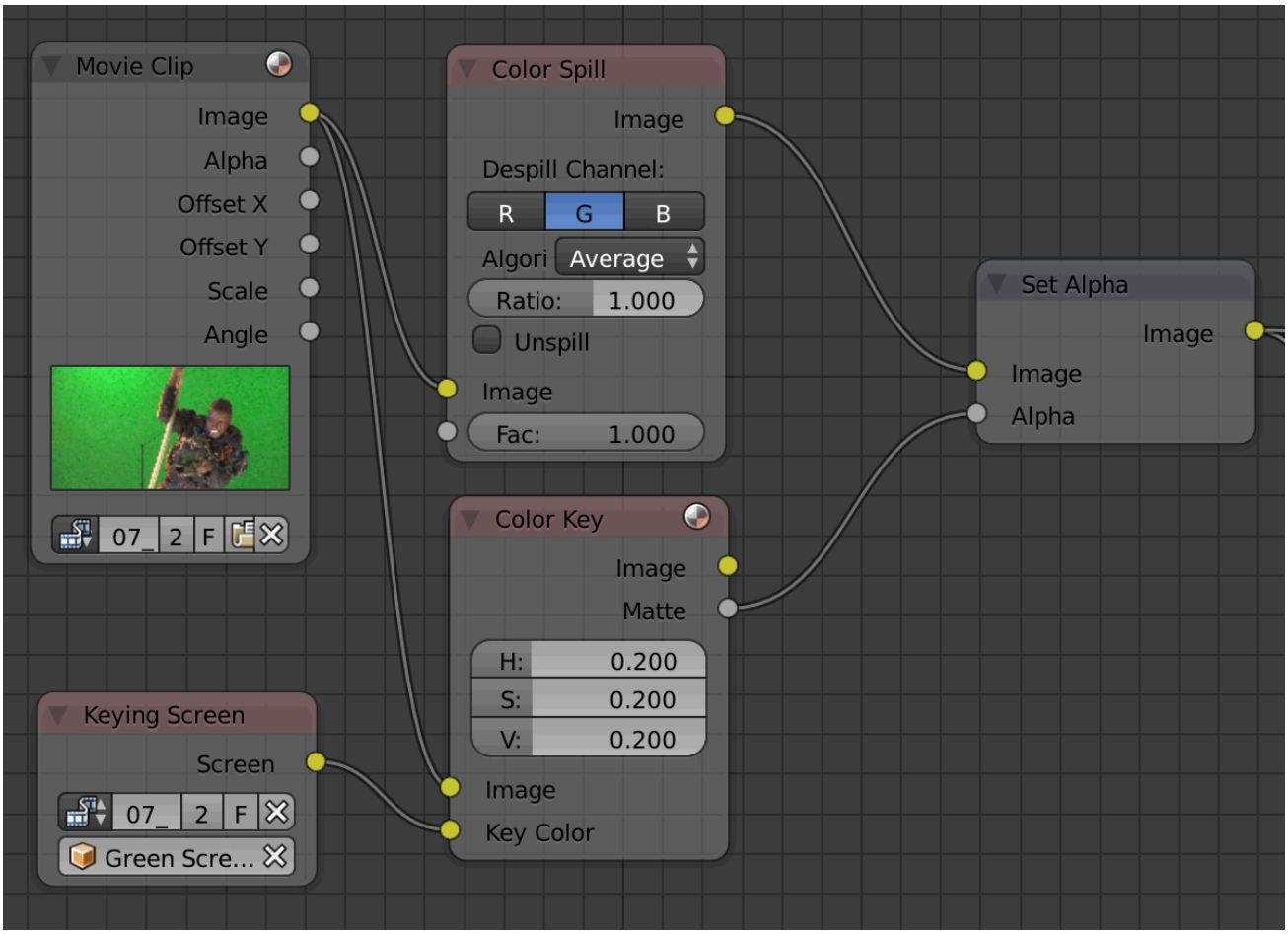
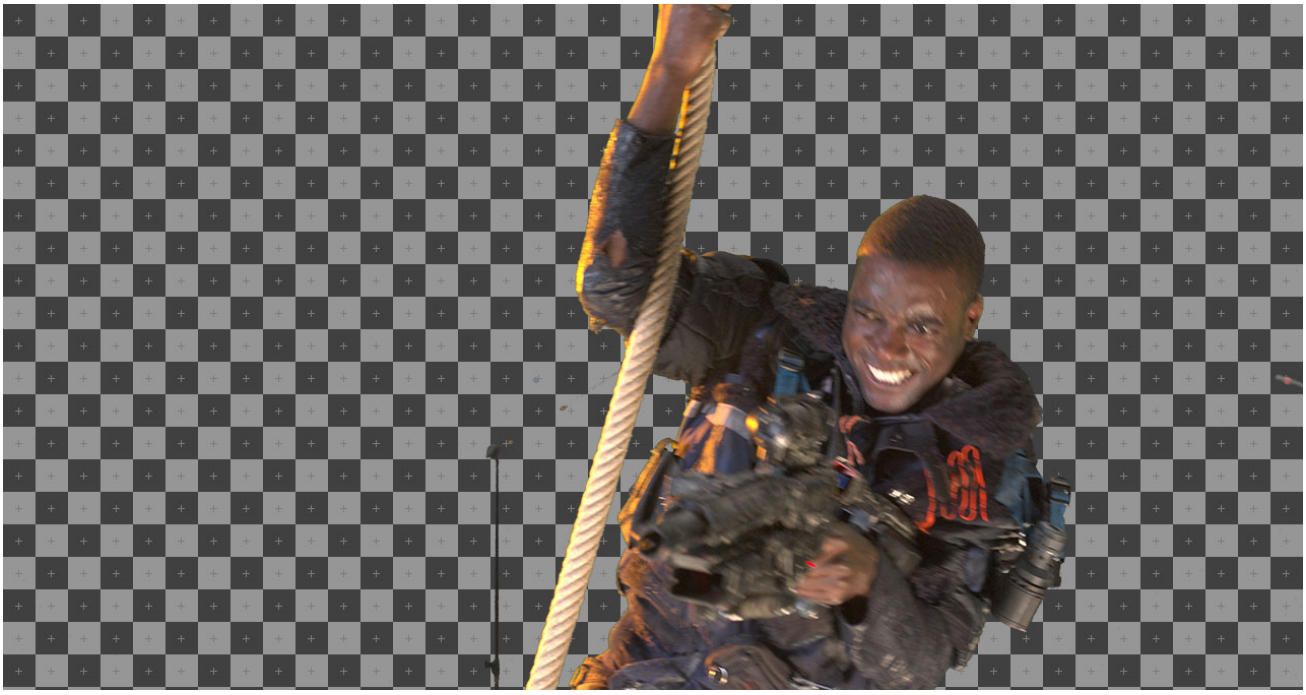


Fig. 2.2282: Node configuration with *Keying Screen*'s generated gradient plate connected to the Color input of the Keying node.

Fig. 2.2283: Gradient plate generated by *Keying Screen*.



foreground effect from the background. It can also be used for sky replacement for over-exposed or gray skies that aren't suitable for chroma keying.

Tip: When compositing footage of something that emits light and has a dark background, like fire, a *Mix Node* using a *Screen* or *Add* operator will produce better results.

Inputs

Image Standard image input.

Properties

Limit

High Determines the lowest values that are considered foreground. (which is supposed to be – relatively – light: from this value to 1.0).

Low Determines the highest values that are considered to be background objects. (which is supposed to be – relatively – dark: from 0.0 to this value).

Note: Brightness levels between the two values form a gradient of transparency between foreground and background objects.

Outputs

Image Image with an alpha channel adjusted for the keyed selection.

Matte A black and white alpha mask of the key.

Example

For this example the model was shot against a *white* background. Using the Luminance Key node, we get a matte out where the background is white, and the model is black; the opposite of what we want. If we wanted to use the matte, we have to switch the white and the black. How to do this? Color Ramp node to the rescue – we set the left color White Alpha 1.0, and the right color to be Black Alpha 0.0. Thus, when the Color Ramp gets in black, it spits out white, and vice versa. The reversed mask is shown; her white outline is usable as an alpha mask now.

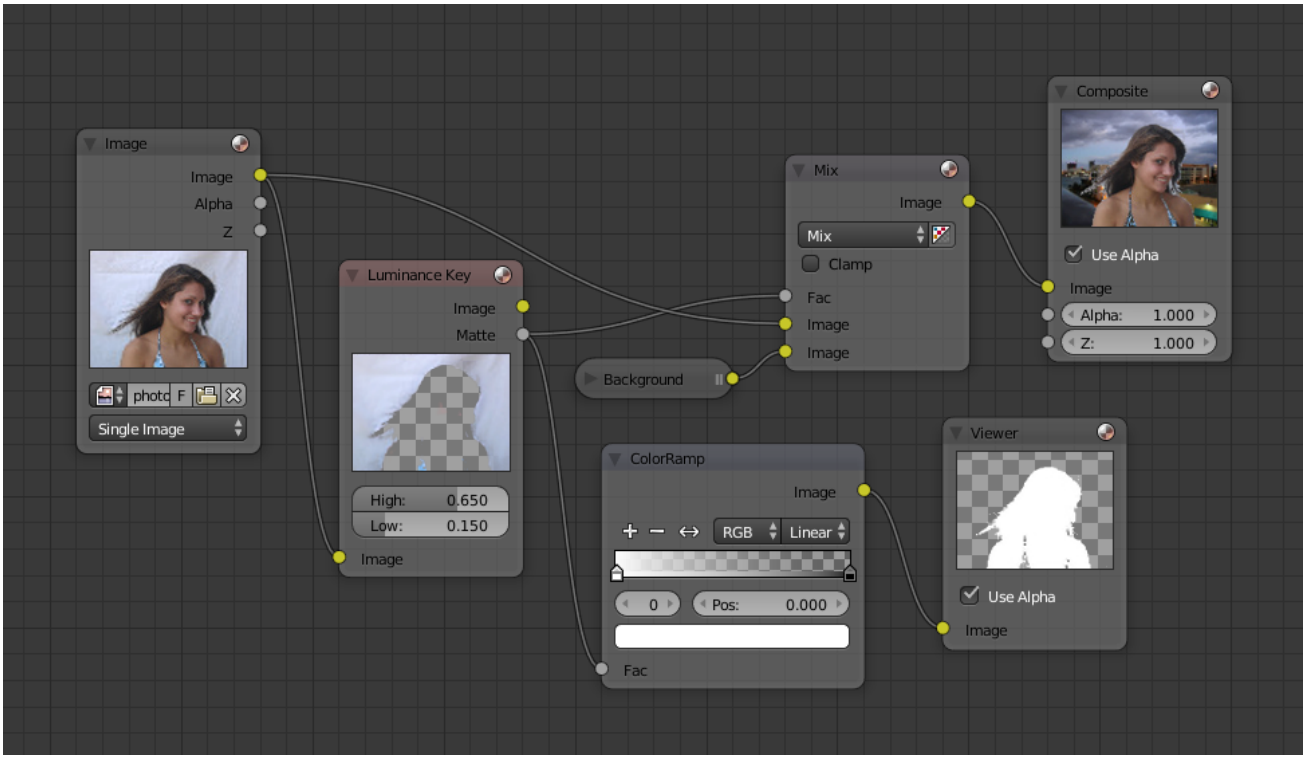


Fig. 2.2285: Using Luma Key with a twist.

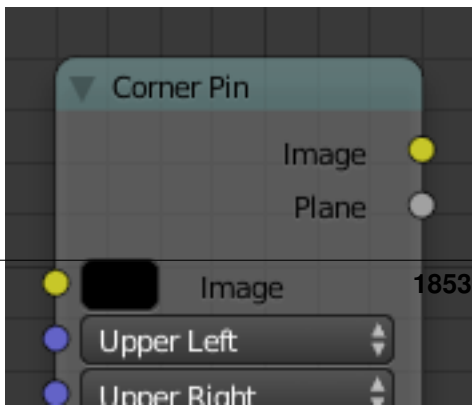
Now to mix, we do not really need the *Alpha Over* node; we can just use the mask as our Factor input. In this kinda weird case, we can use the matte directly; we just switch the input nodes. As you can see, since the matte is white (1.0) where we do not want to use the model picture, we feed the background photo to the bottom socket (recall the mix node uses the top socket where the factor is 0.0, and the bottom socket where the factor is 1.0). Feeding our original photo into the top socket means it will be used where the Luminance Key node has spit out Black. Voila, our model is teleported from Atlanta to aboard a cruise ship docked in Miami.

Distort Nodes

These nodes distort the image in some fashion, operating either uniformly on the image, or by using a mask to vary the effect over the image.

Corner Pin Node

The Corner Pin node uses explicit corner values for a plane warp transformation. It works like the Plane Track Deform node, but without using "plane track" data from the Movie Clip Editor.



Inputs

Image Standard image input.

Corners Four vector inputs to define the plane warping. (Z-component of vector inputs is ignored.)

Properties

This node has no properties.

Outputs

Image Standard image output. (The image after distorting.)

Plane A black and white alpha mask of the plane.

Example

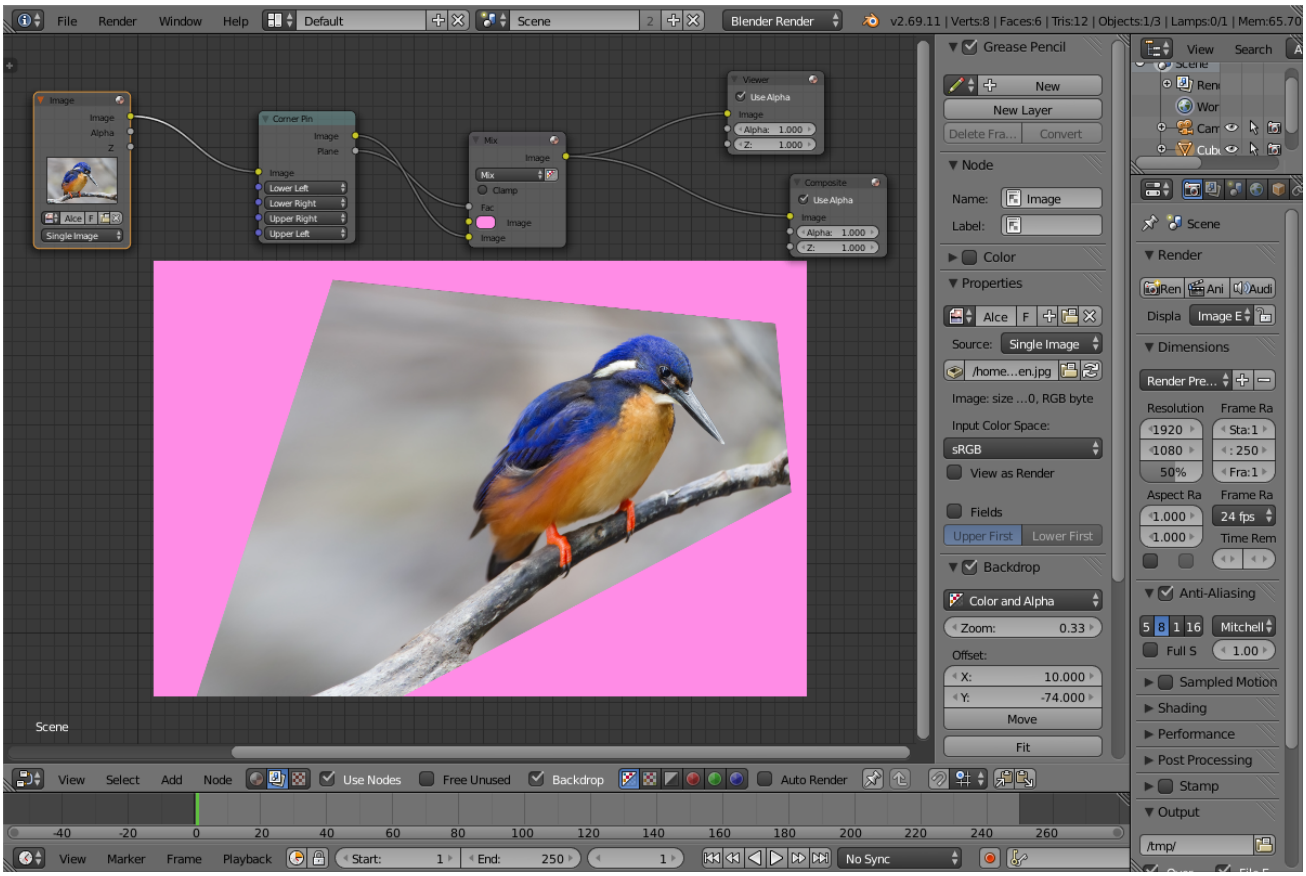


Fig. 2.2287: An example of the Conner Pin node.

In the example above, the image of the bird is distorted by the vectors specified by the Corner Pin node.

Crop Node

The Crop Node takes an input image and crops it to a selected region.

Inputs

Image Standard image input.

Properties

Crop Image Size When enabled, the image size is cropped to the specified region. When disabled, the image remains the same size, and uncropped areas become transparent pixels.

Relative When enabled, crop dimensions are a percentage of the image's width and height. When disabled, the range of the *Crop Region Values* are the width and height of the image in pixels.

Crop Region Values Define borders of the crop region.

lower, upper, left, right

Outputs

Image Standard image output.

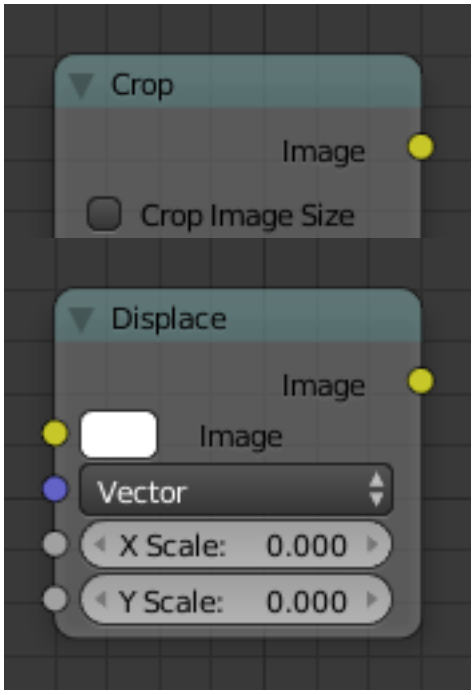


Fig. 2.2288: Crop Node.
Fig. 2.2289: Displace Node.

Displace Node

This node displaces the pixel position based on an input vector.

This node could be used to model phenomena, like hot air distortion, refractions of uneven glass or for surreal video effects.

Inputs

Image Standard image input.

Vector Input of the displacement map. If the a color output is implicitly converted in the vector input, the first channel (red) value determines displacement along X axis. The second channel (green) the displacement along Y axis. If the input is a grayscale image, where both the channel values are equal, the input image will be displaced equally in both X and Y directions.

Scale X, Y Separate scaling of the vector input in X- and Y-direction. Acting as multipliers by increasing or decreasing the strength of the displacement along their respective axes.

Properties

This node has no properties.

Outputs

Image Standard image output.

Flip Node

This node flips an image at defined axis .

You can use this node to just flip or use it as a part of mirror setting. Mix half of the image to be mirrored with its flipped version to produce mirrored image.

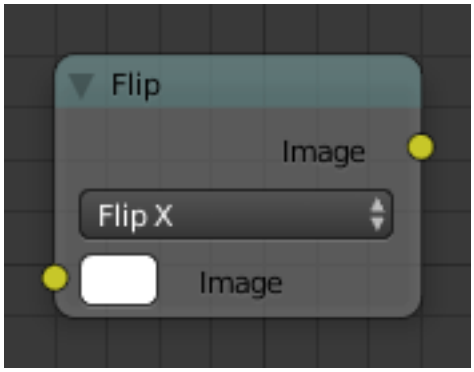


Fig. 2.2290: Flip Node.

Inputs

Image Standard image input.

Properties

Axis This can be either X or Y. Also, flipping can be done on both X and Y axis' simultaneously.

Flip X, Flip Y, Flip X & Y

Outputs

Image Standard image output.

Lens Distortion Node

Use this node to simulate distortions that real camera lenses produce.

Inputs

Image Standard image input.

Distort This creates a bulging or pinching effect from the center of the image.

Dispersion This simulates chromatic aberration, where different wavelengths of light refract slightly differently, creating a rainbow colored fringe.

Properties

Projector Enable or disable slider projection mode. When on, distortion is only applied horizontally. Disables *Jitter* and *Fit*.

Jitter Adds jitter to the distortion. Faster, but noisier.

Fit Scales image so black areas are not visible. Only works for positive distortion.

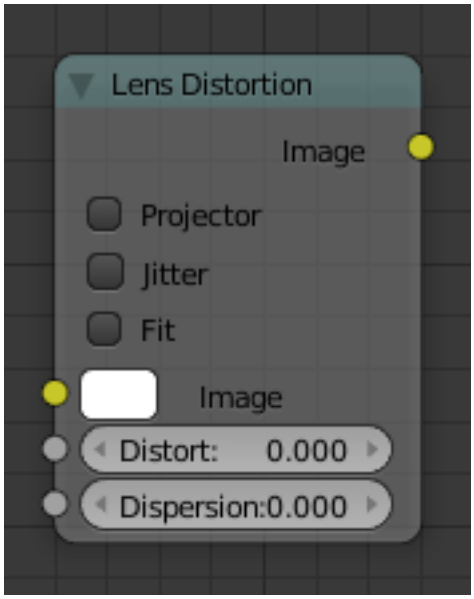


Fig. 2.2291: Lens Distortion Node.

Outputs

Image Standard image output.

Map UV Node

With this node objects can be "re-textured" after they have been rendered.

To apply a texture to individual enumerated objects the *ID Mask Node* could be used.

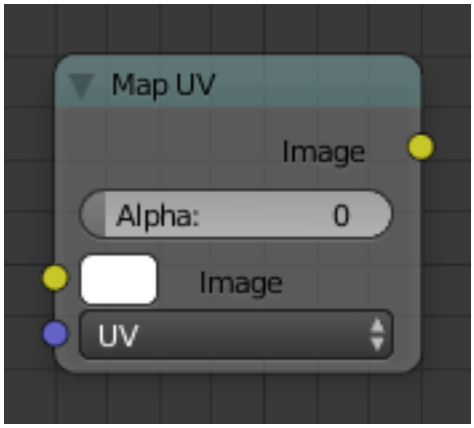


Fig. 2.2292: Map UV Node.

Inputs

Image The new 2D Texture.

UV The input for UV render pass. See *Cycles render passes* or *Blender internal render passes*.

Hint: To store the UV pass a multilayer OpenEXR format could be used.

Properties

Alpha Alpha threshold is used to fade out pixels on boundaries.

Outputs

Image The resulting image is the input image texture distorted to match the UV coordinates. That image can then be overlay mixed with the original image to paint the texture on top of the original. Adjust alpha and the mix factor to control how much the new texture overlays the old.

Hint: When painting the new texture, it helps to have the UV maps for the original objects in the scene, it is recommended to keep those UV texture outlines around even, when shooting is done.

Examples

In the example below, we have overlaid a grid pattern on top of the two heads after they have been rendered. During rendering, we enabled the UV layer in the Properties editor *Render Layer* → *Passes*. Using a mix node ("Overlay" in figure), we mix that new UV Texture over the original face. We can use this grid texture to help in any motion tracking that we need to do.

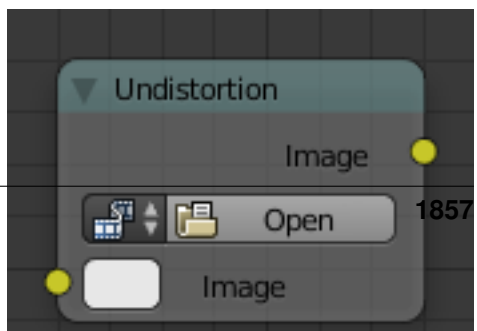
In the next example, we overlay a logo on top of a cubie-type thing, and we ensure that we Enable the Alpha pre-multiply button on the Mix node. The logo is used as additional UV Texture on top of the existing texture. Other examples include the possibility that there was used an unauthorized product box during the initial animation, and it is needed to substitute in a different product sponsor after rendering.

Hint: Due to limits of this node, it is not recommended rush pre-production rendering under the guise of "fixing it later".

Movie Distortion Node

In real life, all camera lenses produce some or the other sort of lens distortion. But, whatever we render has got no distortion. So, this node helps in removing distortion from movies or adding distortion to render to make our render blend in with the movie clip.

Usually, it is used while motion tracking.



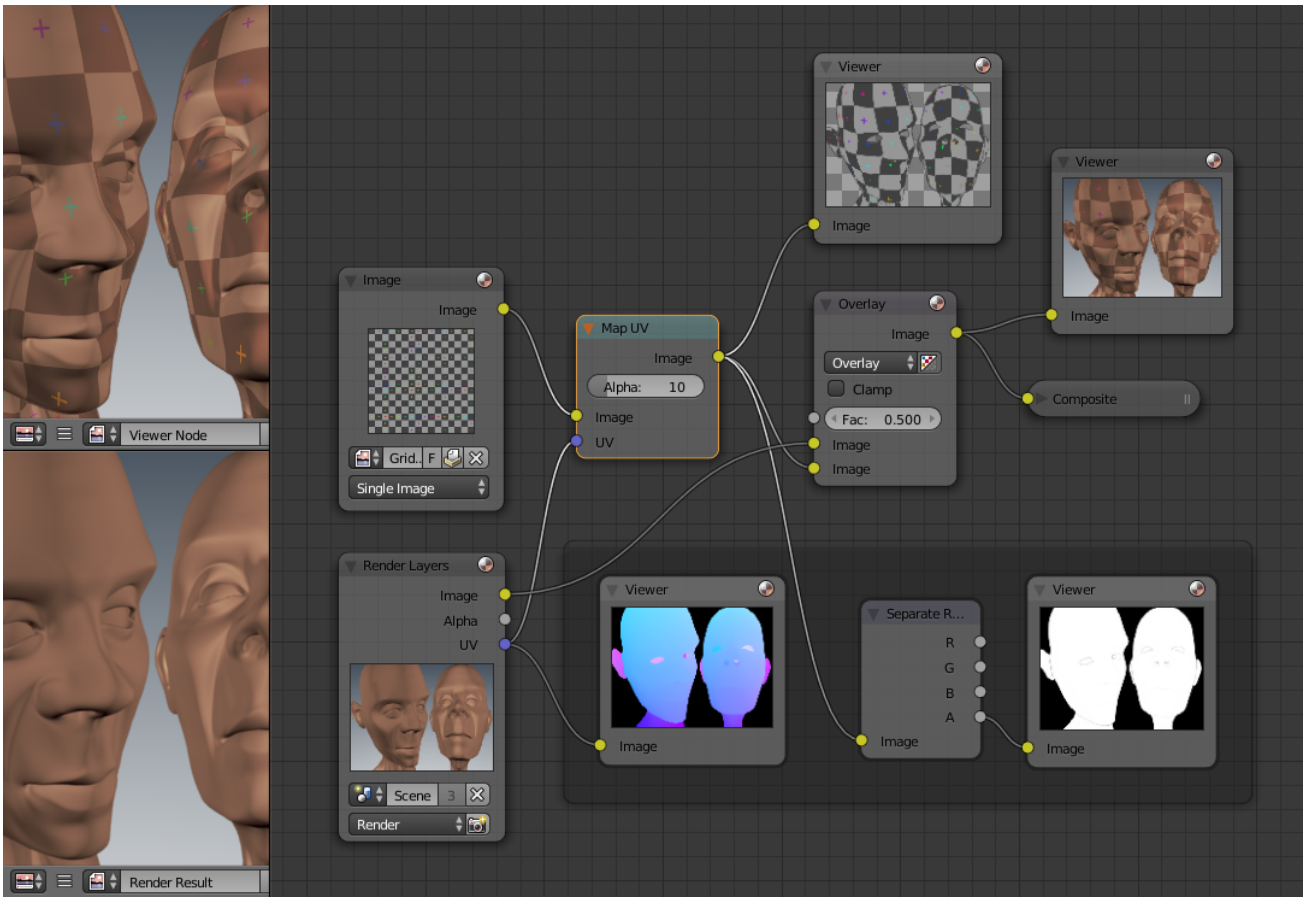


Fig. 2.2293: Adding a Grid UV Textures for Motion Tracking.

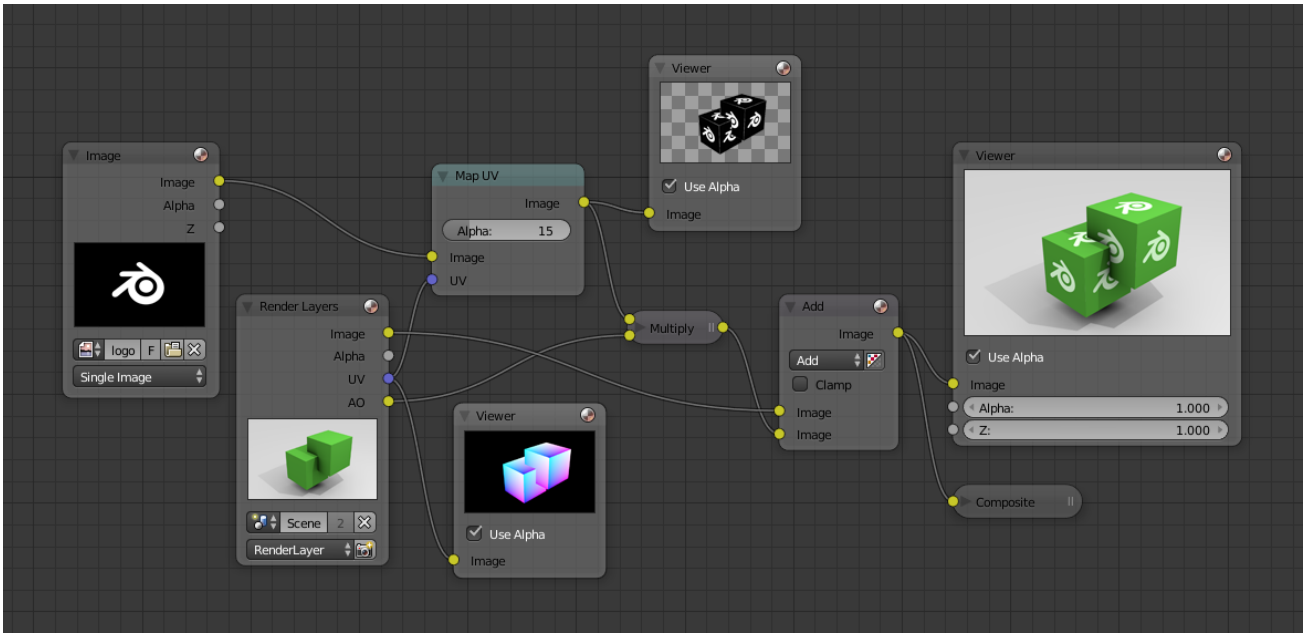


Fig. 2.2294: Adding UV Textures in Post-Production

Calculating Distortion

Before using this node, one has to calculate the lens distortion of the clip. This can be done by adjusting K1, K2 and K3 values in *Movie Clip Editor* → *Properties* → *Lens*. For more information on how to edit those values, check this out.

Inputs

Image Standard image input.

Properties

Movie Clip Used to select the movie clip whose distortion is to be used. This can be useful if more than one movie clips are present, each having a different distortion setting. For controls see *Data-Block Menu*.

Distortion Method

Undistort Used to undistort the image received, and is usually used for the raw distorted movie clip.

Distort Used to distort the image received, and is usually used for rendered images.

Outputs

Image The image after distorting/undistorting.

Distortion vs Undistortion

Although, both, distortion of render and undistortion of movie clip are possible, and produce similar results, there is a difference between these two methods.

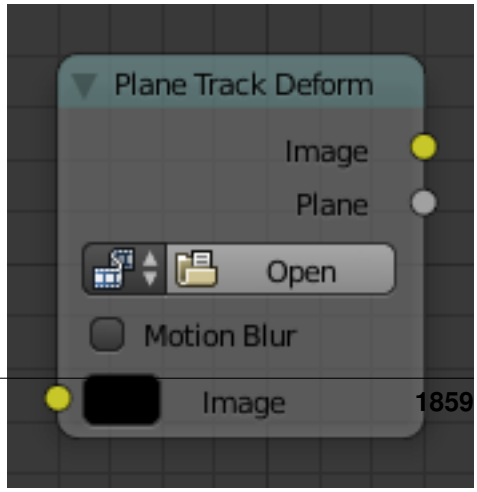
There are two kinds of lens distortion possible and, in simple terms, they can be said as:

1. When the movie clip is bulging out.
2. When the movie clip is bulging in.

For the first case, it is recommended to distort the render and leave the movie clip as it is, because, undistorting the movie clip will require extra pixel information, which is not available to Blender. Similarly, in the second case, it is recommended to undistort the movie clip and leave the render as it is, becuase, distorting the render will require those extra unavailable pixels. Doing the wrong method in the wrong case can create weird results around the edges, such as in the image shown.

Plane Track Deform Node

The Plane Track Deform Node is used to incorporate the special "plane track" in your composite by checking areas which are planes, and replacing their footage with some other image.



Plane Track

Before using this node, plane track for the footage should be made in the *Movie Clip Editor*.

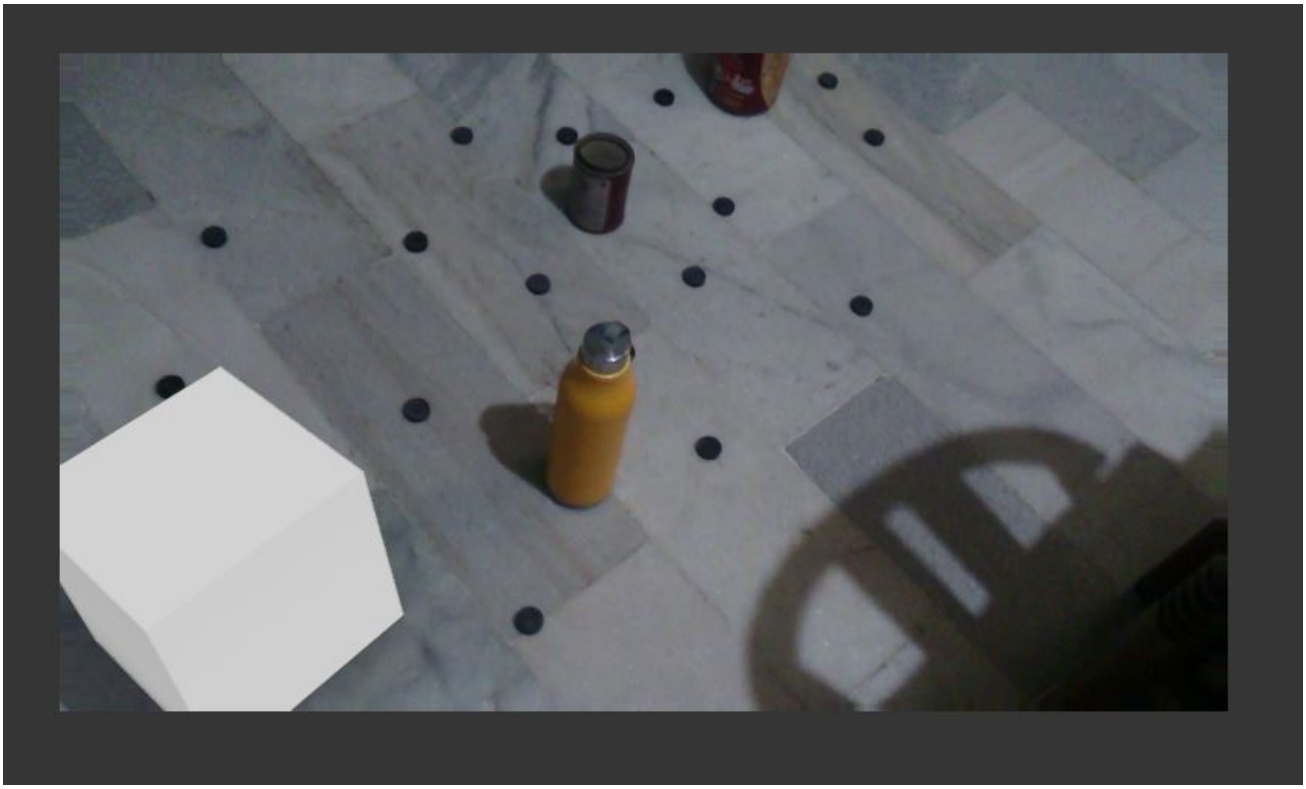


Fig. 2.2296: Problems (notice the edges?)

Inputs

Image Image to put in place of the plane track, and thus, override that area in the movie clip.

Properties

Movie Clip Used to select the movie clip whose plane track to use. For controls see *Data-Block Menu*.

Object Used to select the object to which the plane track is linked.

Track Used to select the plane track to use.

Motion Blur Specify whether to use blur caused by motion of the plane track or not.

Samples Set the number of samples to take for each frame. The higher the samples, the smoother the blur effect, but the longer the render, as each virtual intermediate frame has to be rendered.

Note: Samples are taken only from the *next* frame, not the previous one. Therefore the blurred object will appear to be slightly ahead of how it would look without motion blur.

Shutter Time (in frames) the shutter is open. If you are rendering at 24 fps, and the Shutter is set to 0.5, the time in between frames is 41.67 ms, so the shutter is open for half that, 20.83 ms.

Outputs

Image The output by wrapping the image to that plane track.

Plane Produces a bw mask of the plane track.

Examples

Using Image output

This can simply be achieved by using the alpha over node.

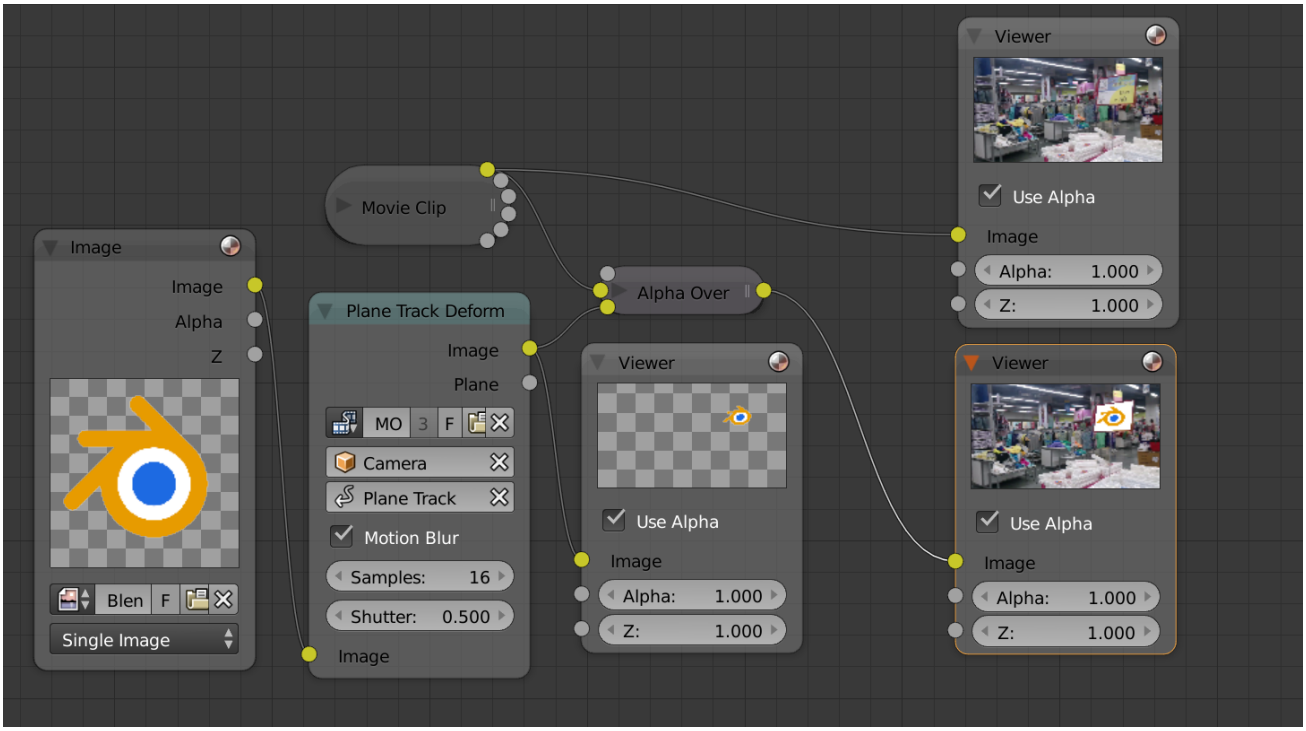


Fig. 2.2298: Image output.

Using Plane output

This can be achieved by mixing the movie clip and the image using the plane output as the factor.

Using Image output vs using original image

Using Image output scales, translates and skews the input image according to the track while using the original image and mixing it with the movie clip using Plane output as factor will display the part of the image that lies inside that mask. This image shows the difference:

Rotate Node

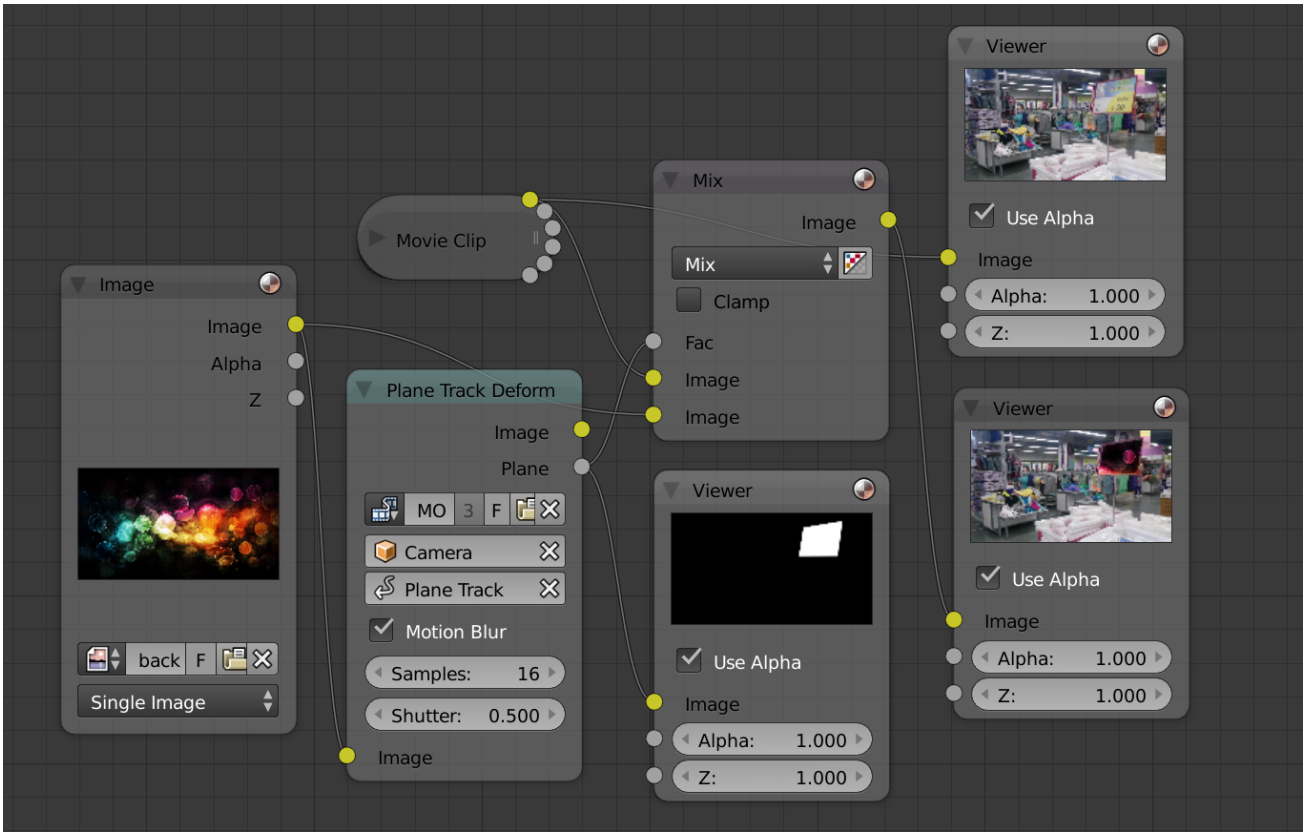


Fig. 2.2299: Plane output.

This node rotates an image.

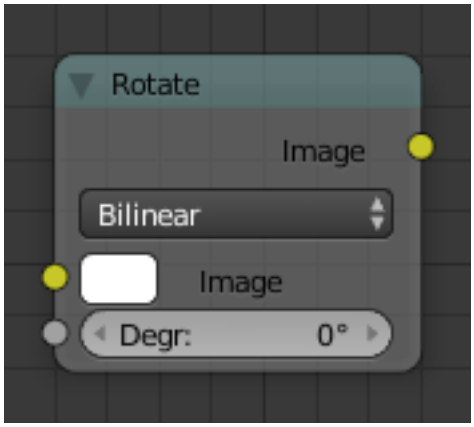


Fig. 2.2301: Rotate Node.

Inputs

Image Standard image input.

Degr Rotation angle in degree. Positive values rotate clockwise and negative ones counterclockwise.

Properties

Filter Interpolation Methods.

Nearest No interpolation, uses nearest neighboring pixel.

Bilinear Simple interpolation between adjacent pixels.

Bicubic Highest quality interpolation.

Outputs

Image Standard image output.

Scale Node

This node scales the size of an image.

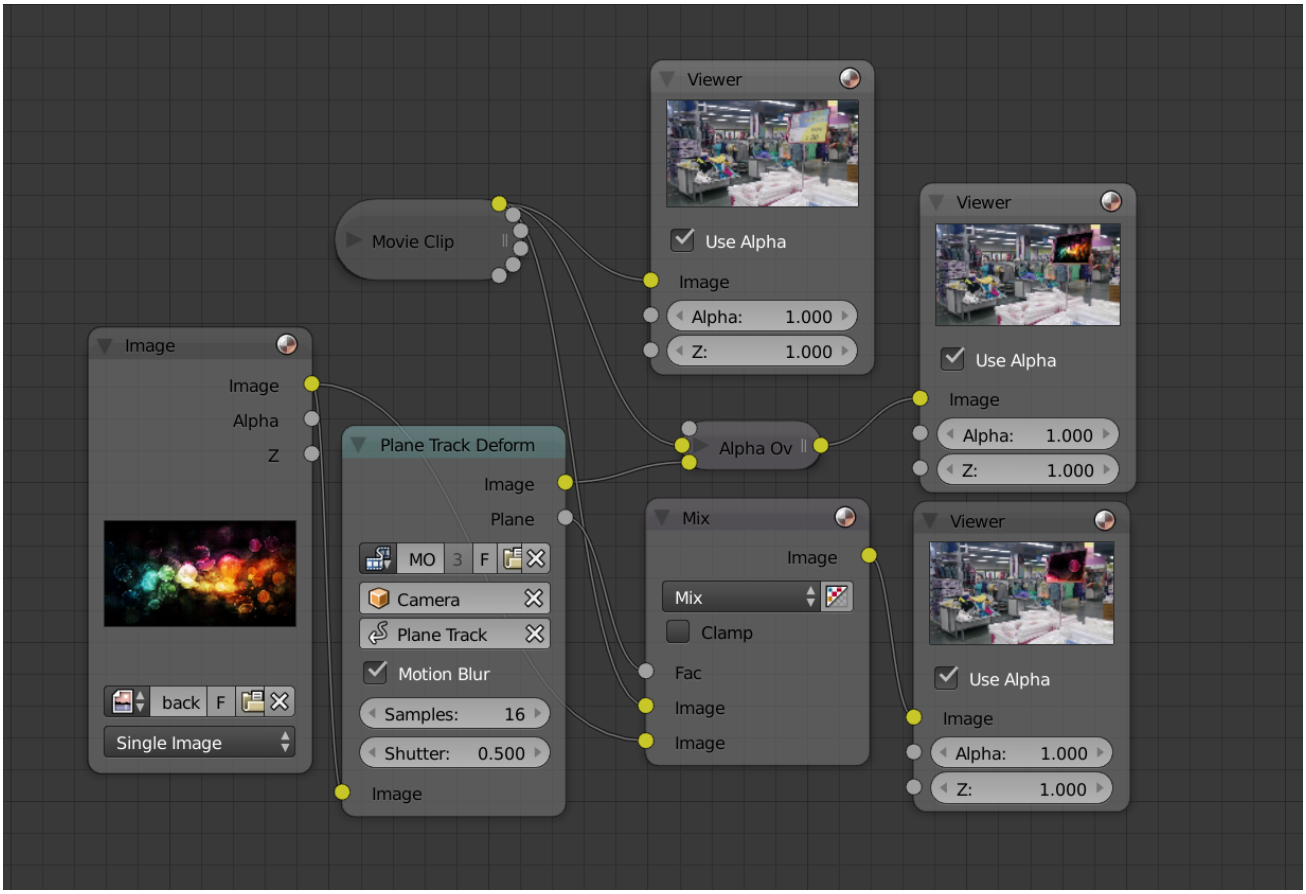


Fig. 2.2300: Comparison between image output and original image (see viewer nodes carefully).

Inputs

Image Standard image input.

X, Y Available if Space is relative or absolute. Scale in the axis directions.

Properties

Space Coordinate Space to scale relative to.

Relative Percentage values relative to the dimensions of the image input.

Absolute Size of an image by using absolute pixel values.

Scene Size TODO.

Render Size Image dimensions set in the Render panel.

Stretch, Fit, Crop TODO.

X, Y TODO.

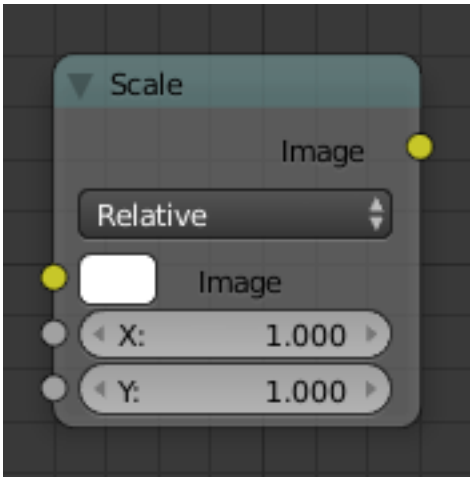


Fig. 2.2302: Scale Node.

Outputs

Image Standard image output.

Examples

For instance X: 0.5 and Y: 0.5 would produce an image which width and height would be half of what they used to be.

Use this node to match image sizes. Most nodes produce an image that is the same size as the image input into their top image socket. To uniformly combine two images of different size, the second image has to be scaled up to match the resolution of the first.

Stabilize 2D Node

Stabilizes the footage according to the settings set in *Movie Clip Editor* → *Properties* → *2D Stabilization* For more information, *check this out*.

Inputs

Image Standard image input.

Properties

Movie Clip The movie clip whose stabilization to use.

Filter Various methods for the stabilization. Usually, the same as used in *Movie Clip Editor* → *Properties* → *2D Stabilization* → *Filter*. For technical details on their difference, see this. But for most purposes, default of Bilinear should suffice.

Invert Invert the stabilization. If the stabilization calculated is to move the movie clip up by 5 units, this will move the movie clip down by 5 units.

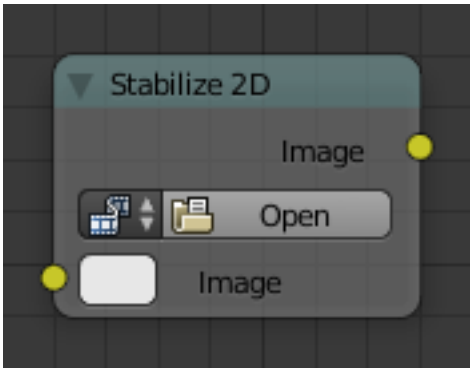


Fig. 2.2303: Stabilize 2D Node.

Outputs

Image Standard image input.

Transform Node

This node combines the functionality of three other nodes: *Scale*, *translate*, and *rotate* nodes.

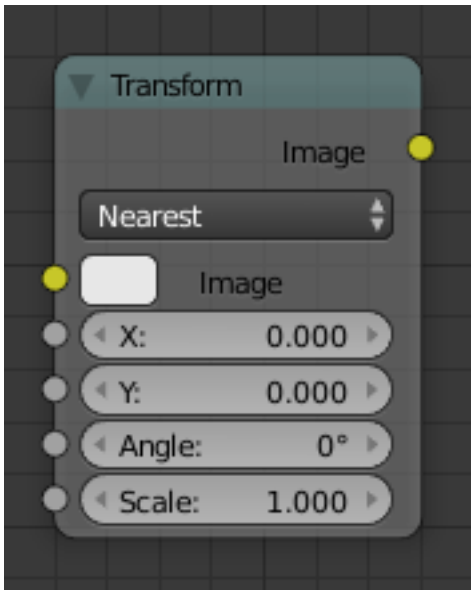


Fig. 2.2304: Transform Node.

Inputs

Image Standard image input.

X, Y Used to move the input image horizontally and vertically.

Angle Used to rotate an image around its center. Positive values rotate counter-clockwise and negative ones clockwise.

Scale Used to resize the image. The scaling is relative, meaning a value of 0.5 gives half the size and a value of 2.0 gives twice the size of the original image.

Properties

Filter Interpolation Methods.

Nearest No interpolation, uses nearest neighboring pixel.

Bilinear Simple interpolation between adjacent pixels.

Bicubic Highest quality interpolation.

Outputs

Image Standard image output.

Translate Node

The translate node translates (moves) an image.

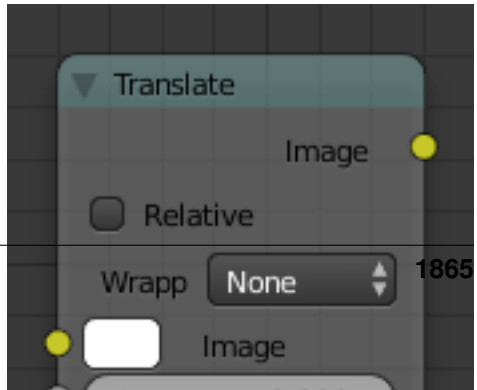
Could also be used to add a 2D Camera shake.

Inputs

Image Standard image input.

X, Y Used to move the input image horizontally and vertically.

Properties



Relative Percentage translation values relative to the input image size.

Wrapping Repeat image.

None, X Axis, Y Axis, Both Axis

Outputs

Image Standard image output.

Node Groups

Both material and composite nodes can be grouped. Grouping nodes can simplify the node network layout in the node editor, making your material or composite node setup easier to work with. Grouping nodes also creates what are called NodeGroups (inside a blend-file) or NodeTrees (when appending).

Conceptually, "grouping" allows you to specify a *set* of nodes that you can treat as though it were "just one node". You can then re-use it one or more times in this or some other blend-file(s).

As an example: If you have created a material using nodes that you would like to use in another blend-file, you could simply append the material from one blend-file to another. However, what if you would like to create a new material, and use a branch from an existing material node network? You could re-create the branch. Or you could append the material to the new blend-file, then cut and paste the branch that you want into the new material. Both of these options work, but are not very efficient when working across different blend-files. A better method of re-use, for either material node branches or composite node networks, would be to create groups of nodes.

Once a group has been defined, it becomes an opaque object; a reusable software component. You can (if you choose) ignore exactly how it is "defined", and simply use it as many times as you like.

Grouping Nodes

To create a node group, in the node editor, select the nodes you want to include, then press `Ctrl-G`, *Group* → *Make Group*, `Shift-A`. A node group will have a green title bar. All of the selected nodes will now be contained within the group node. Default naming for the node group is "NodeGroup", "NodeGroup.001" etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful. When appending node groups from one blend file to another, Blender does not make a distinction between material node groups or composite node groups, so it is recommended some naming convention, that will allow you to easily distinguish between the two types.

Note: What **not** to include in your groups (all types of Node editors)

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not** include:

Input nodes If you include a source node in your group, you will end up having the source node appearing *twice*: once inside the group, and once outside the group in the new material node-network.

Output node If you include an output node in the group, there will not be an output socket available *from* the group!

Editing Node Groups

With a group node selected, `Tab` expands the node to a frame, and the individual nodes within it are shown. You can move them around, play with their individual controls, re-thread them internally, etc. just like you can if they were a normal part of the editor view. You will not be able, though, to thread them to a node outside the group; you have to use the external sockets on the side of the group node. To add or remove nodes from the group, you need to ungroup them.

Ungrouping Nodes

The `Alt-G` command removes the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread internal nodes to other nodes in your workspace.

Appending Node Groups

Once you have appended a NodeTree to your blend-file, you can make use of it in the node editor by pressing `Shift-A`, *Add* → *Group*, then select the appended group. The "control panel" of the Group is the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

Layout Nodes

These are nodes which help you control the layout and connectivity of nodes within the Compositor.

Frame Node

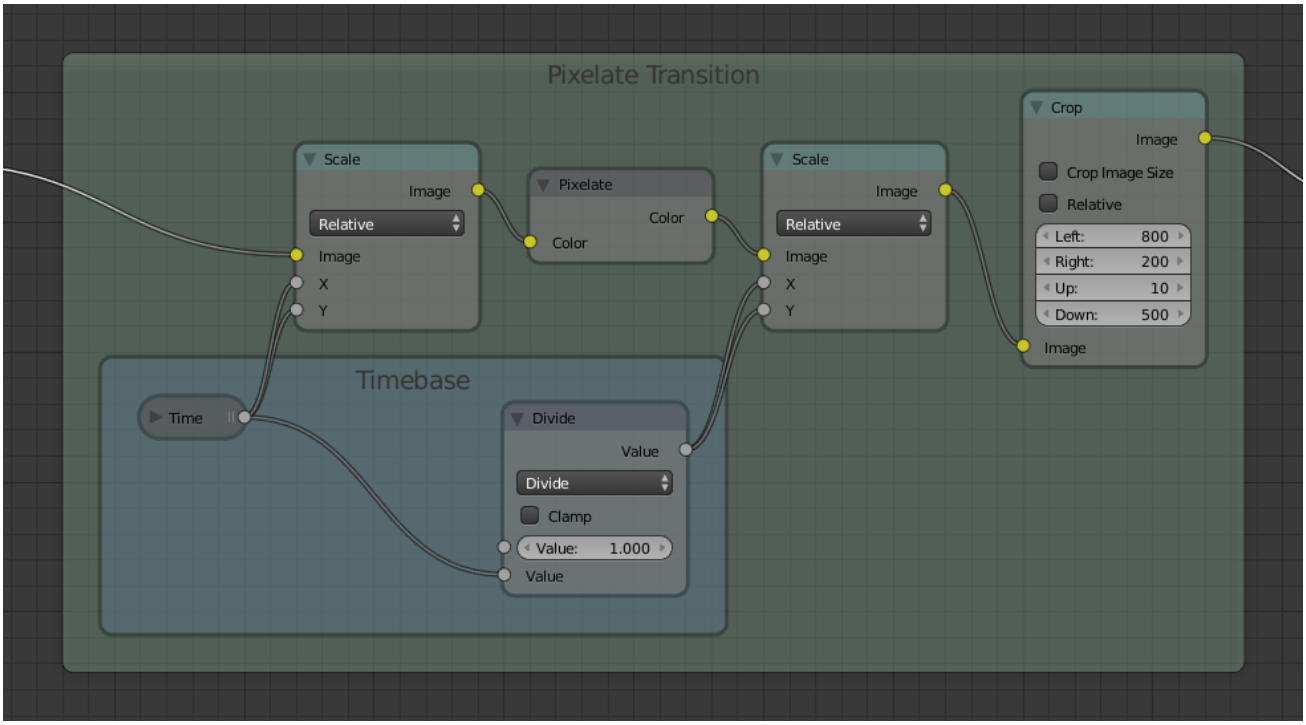
The Frame node is a useful tool for organizing nodes by collecting related nodes together in a common area. Frames are useful when a node setup becomes large and confusing yet the re-usability of a Node Group is not required.

Properties

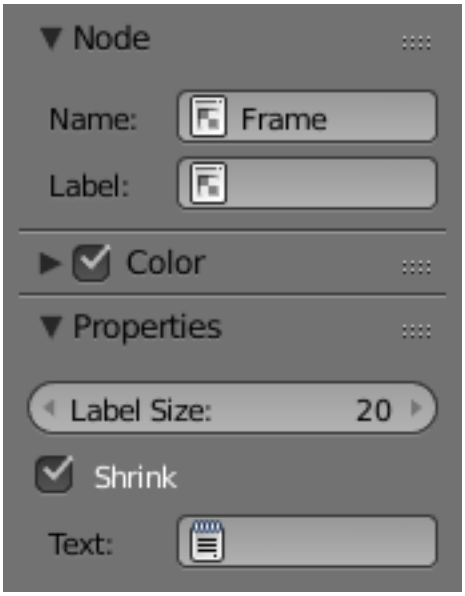
Label size Font size of the label. For example, for subordinate frames to have smaller titles.

Shrink Once a node is placed in the Frame, the Frame shrinks around it so as to remove wasted space. At this point it is no longer possible to grab the edge of the Frame to resize it, instead resizing occurs automatically when nodes within the Frame are rearranged. This behavior can be changed by disabling this option.

Text When you need to display more comprehensive text, frame nodes can display the contents of a text-block. This is read-only, so you will need to use the *Text Editor* to modify the contents.



Adding and Removing Nodes



Once a Frame node is placed in the editor, nodes can be added by simply dropping them onto the frame or by selecting the node(s) then the frame and using `Ctrl-P`.

To remove them select the node(s) and use the `Alt-P` shortcut. This uses the same default keyboard bindings as Parenting and can be thought of as a similar concept.

Reroute Node

A node used primarily for organization. Reroute looks and behaves much like a socket on other nodes in that it supports one input connection while allowing multiple output connections.

To quickly add a Reroute node into an existing connection, hold `Shift` and `LMB` while sweeping across the link to add a *Reroute node*.

Properties

Input Input value used for unconnected sockets.

Switch Node

Switch between two images using a checkbox.

Inputs

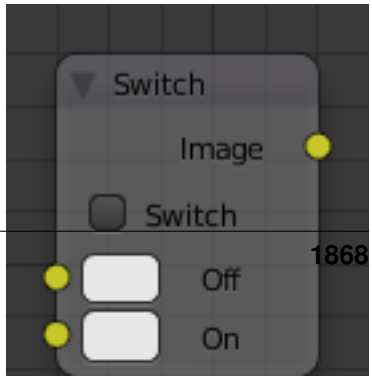


Image First image input.

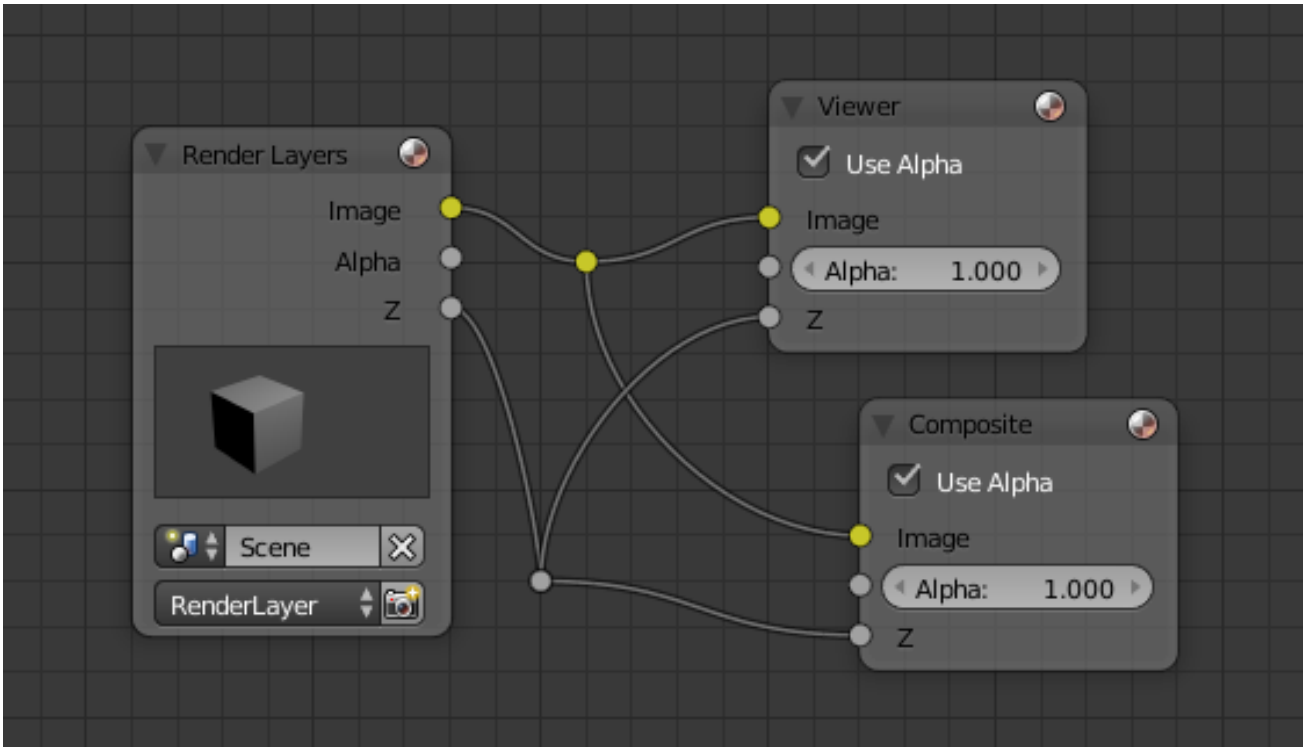


Image Second image input.

Properties

Switch

- When it is unchecked, the first input labeled "Off" is passed to the output.
- When checked, the second input labeled "On" is passed to the output.

Outputs

Image Standard image output.

Tip: Switch state may be animated by adding a *keyframe* This makes the Switch node useful for bypassing nodes which are not wanted during part of a sequence.

2.11 Game Engine

2.11.1 Introduction

The Blender Game Engine (BGE) is Blender's tool for real time projects, from architectural visualizations and simulations to games.

A word of warning, before you start any big or serious project with the Blender Game Engine, you should note that it is currently not very supported and that there are plans for its retargeting and refactoring that, in the very least, will break compatibility. For further information, you should get in touch with the developers via mailing list or IRC and read the development roadmap.

```
data-conversion --> input | animation, physics and logic | rendering
```

different renderer, logic is done via logic bricks or Python press `P` to play in preview mode in the embedded player.

Use Cases and Sample Games

Blender has its own built in Game Engine that allows you to create interactive 3D applications or simulations. The major difference between Game Engine and the conventional Blender system is in the rendering process. In the normal Blender engine, images and animations are built off-line – once rendered they cannot be modified. Conversely, the Blender Game Engine renders scenes continuously in real-time, and incorporates facilities for user interaction during the rendering process.

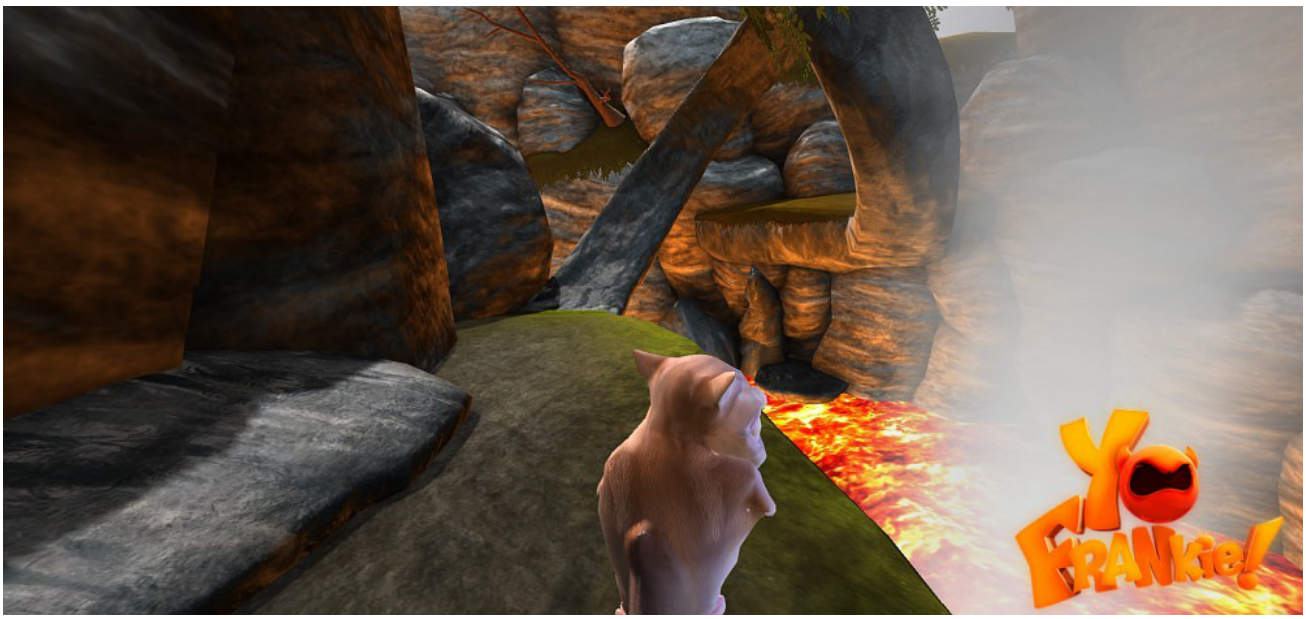


Fig. 2.2307: Screenshot from "Yo Frankie", produced with Blender Game Engine.

The Blender Game Engine oversees a game loop, which processes logic, sound, physics and rendering simulations in sequential order. The engine is written in C++.

By default, the user has access to a powerful, high level, Event Driven *Logic Editor* which is comprised of a series of specialized components called "Logic Bricks". The *Logic Editor* provides deep interaction with the simulation, and its functionality can be extended through Python scripting. It is designed to abstract the complex engine features into a simple user interface, which does not require experience with Programming. An overview of the *Logic Editor* can be found in the *Game Logic Screen Layout*

The Game Engine is closely integrated with the existing code base of Blender, which permits quick transitions between the traditional modeling feature set and game-specific functionality provided by the program. In this sense, the Game Engine can be efficiently used in all areas of game design, from prototyping to final release.

The Game Engine can simulate content within Blender, however, it also includes the ability to export a binary run-time to Linux, macOS, and MS-Windows.

There are a number of powerful libraries the Game Engine takes advantage of:

- Audaspace: A sound library for control of audio. Uses OpenAL or SDL.
- Bullet: A physics engine featuring 3D collision detection, soft body dynamics, and rigid body dynamics.
- Detour: A path-finding and spatial reasoning toolkit.
- Recast: A state of the art navigation mesh construction tool set for games.

When creating a game or simulation in the BGE, there are four essential steps:

- Create visual elements that can be rendered. This could be 3D models or images.
- Enable interaction within the scene using logic bricks to script custom behavior and determine how it is invoked (using the appropriate "sensors" such as keyboards or joysticks).
- Create one (or more) camera to give a frustum from which to render the scene, and modify the parameters to support the environment in which the game will be displayed, such as Stereo rendering.
- Launch the game, using the internal player or exporting a runtime to the appropriate platform.

2.11.2 Game Logic Screen Layout

The design, construction, debugging and running of a game utilizes a wide range of Blender functions. To help with the process, Blender incorporates a suggested screen layout for setting up BGE games. This includes many already-familiar panels but also a new *Logic Editor* panel (4) concerned solely with the BGE.

The diagram below shows this default Game Logic screen layout, together with the appropriate options for game setup/debug/running (these should be set up in the order shown).

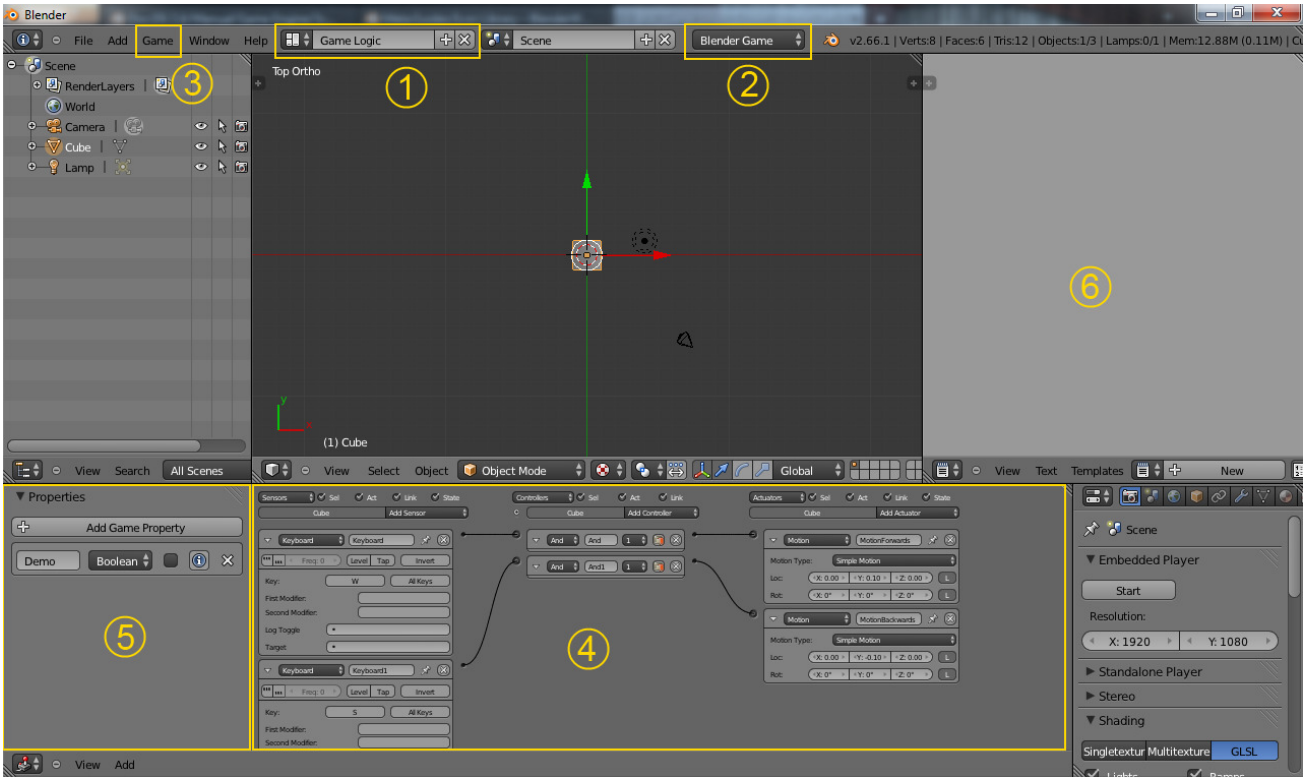


Fig. 2.2308: Game Logic Screen Layout.

1) Game Logic

Selected from the list of screen layouts for various applications. This includes many already-familiar panels Information, 3D View, Properties but also a new Logic Editor panel concerned solely with the BGE.

2) Blender Game

Selected from the render engine menu. This specifies that all output will be output by the real-time Blender Game Engine renderer. It also opens various other menu options such as the Game options (see below) and a range of Properties for the BGE renderer properties (see below)

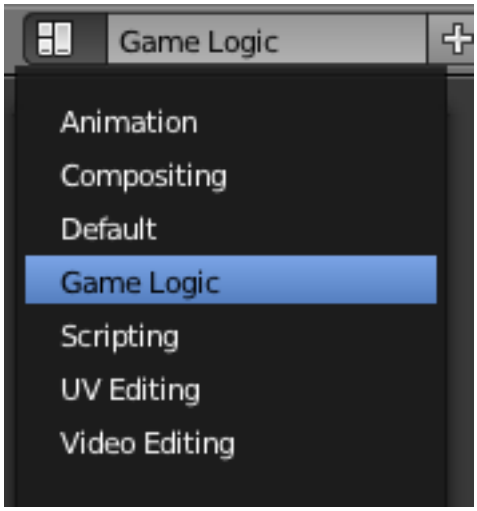


Fig. 2.2309: Game Logic Menu.

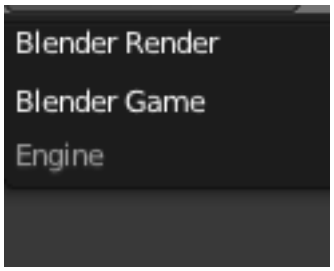


Fig. 2.2310: Render Engine Menu.

Fig. 2.2311: Game Options.

3) Game

This menu gives various options for conditions for running the Game Engine. Note that this menu is only available when the render engine is set to Blender Game.

Start Game Run game in Game Engine (`P` or `Shift-P` when the mouse cursor is over the 3D View editor).

Show Debug Properties Show properties marked for debugging while game runs.

Show framerate and profile Show framerate and profiling information while game runs.

Show Physics visualization Show a vizualisation of physics bounds and interactions.

Deprecation warnings Print warnings when using deprecated features in the Python API.

Record animation Record animation to F-Curves.

Auto Start Automatically start game at load time.

4) Logic Editor panel

The *Logic Editor* is where the *logic, properties and states* are set up to control the behavior of the objects in the game. (The Logic Editor panel can also be displayed by selecting Logic Editor in the Display Editor menu, by pressing `Shift-F2`, or by pressing `Ctrl-Right`).

5) Properties

Tip: Two Meanings for the Same Word

Note that the name "Property" has two different uses in Blender terminology – firstly in the wider use of the Property Display Panel as described here, and secondly as the term used for specific Game Engine logic variables which are also called "properties".

The Property panel of the screen is selected as usual from the main Information menu. However, note that several sections of the Property panel are changed when the render engine (2) is changed from Blender Render to Blender Game.

See following sections for details of the content of *Physics* Properties panels.

2.11.3 Game Materials

Game Settings

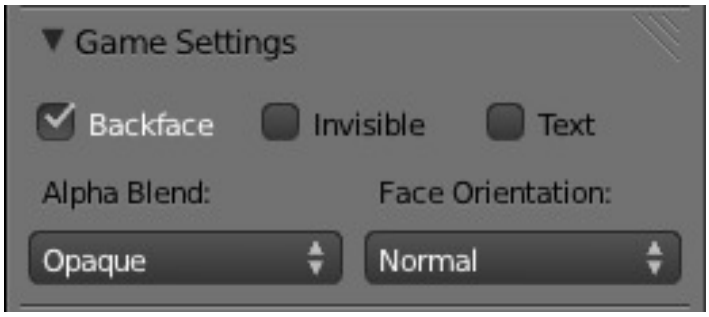


Fig. 2.2312: Game Settings Panel.

This panel contains properties that control how the object surfaces that use the material are rendered in real time by the Blender Game Engine.

Backface Cull Hide the back faces of objects rendered with this material. If "Off", both sides of the surface are visible (at the expense of lower rendering speed). Note that this setting is applied per-material and not per-face; e.g. if the material is applied to a cube, only the back and front faces of the cube are visible, and not both sides of each face.

Invisible Hide all faces of objects rendered with this material.

Text Use material as *Text object* in the Game Engine.

Alpha Blend Controls how the alpha channel is used to create a transparent texture in the rendered image.

Alpha Sort Orders the sequence in which transparent objects are drawn on top of each other, so that ones in front receive more light than ones behind.

Alpha Blend Uses the alpha values present in the bitmap image sourced in the Image slot.

Alpha Clip Uses the alpha channel as a simple mask.

Add Render face transparent and add color of face.

Opaque (default) All alpha values are ignored; the scene is completely non-transparent.

Face Orientation

Provides options regarding the orientation (i.e. rotation transformation) of faces to which the material is applied.

Shadow Faces are used for shadow.

Billboard Billboard with Z-axis constraint.

Halo Screen aligned billboard.

Normal (default) No transformation.

Material Physics

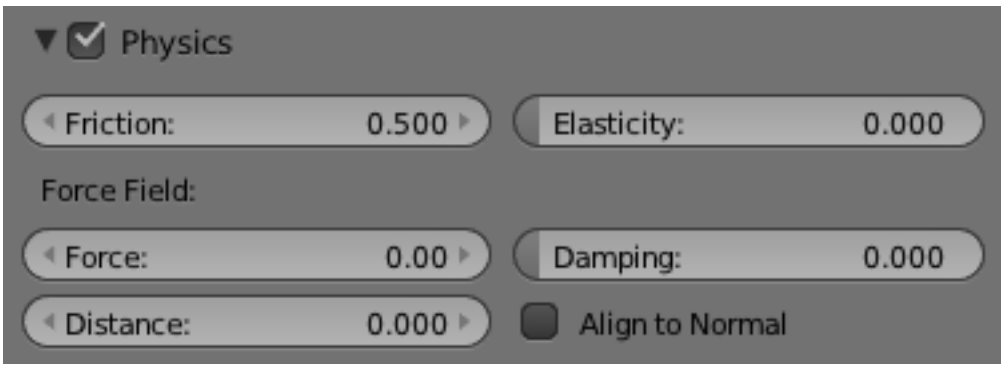


Fig. 2.2313: Panel Physics in Material context.

This panel contains physical properties that control how the object surfaces that use the material are rendered in real time by the Blender Game Engine.

Physics settings are visible when using the Game Engine for rendering, and handled by the *Game Engine's physics engine*

Friction Coulomb friction coefficient when inside the physics distance area.

Elasticity The elasticity of collisions determines how much of the kinetic energy is retained after the collision. A value of 1 will result in a collision where objects bounce off each other, and the kinetic energy after the collision is the same as before. A value of 0 will result in a collision where the objects stick together after the collision, as all energy will have been converted to heat (or other energy forms that Blender also does not model).

In macroscopic nature (so bigger than atomic particles) an elasticity of 1 is never seen, as at least some energy is converted to heat, sound, etc. An elastic (elasticity=high) collision occurs when two metal balls collide. An inelastic (elasticity=low) collision is seen when two half-inflated beach balls collide.

Force Field Controls force field settings.

Force Upward spring force when inside the physics distance area.

Distance Distance of physics area.

Damping Damping of the spring force when inside the physics distance area.

Align to Normal Align dynamic game objects along the surface normal when inside the physics distance area.

2.11.4 Logic

Introduction

Game Logic is the default scripting layer in the Game Engine. Each *Game Object* in the game may store a collection of logical components (Logic Bricks) which control its behavior within the scene. Logic bricks can be combined to perform user-defined actions that determine the progression of the simulation.

Logic Bricks

The main part of game logic can be set up through a graphical interface the *Logic Editor*, and therefore does not require detailed programming knowledge. Logic is set up as blocks (or "bricks") which represent preprogrammed functions; these can be tweaked and combined to create the game/application. There are three types of logic brick: *Sensors*, *Controllers* and *Actuators*. Sensors are primitive event listeners, which are triggered by specific events, such as a collision, a key press or mouse movement. Controllers carry out logic operations on sensor output, and trigger connected actuators when their operating conditions are met. Actuators interact with the simulation directly, and are the only components in the game which are able to do so (other than the Python controller, and other simulation components such as Physics).

Properties

Properties are like variables in other programming languages. They are used to save and access data values either for the whole game (eg. scores), or for particular objects/players (e.g. names). However, in the Blender Game Engine, a property is associated with an object. Properties can be of different types, and are set up in a special area of the *Logic Editor*.

States

Another useful feature is object *States*. At any time while the simulation is running, the object will process any logic which belongs to the current state of the object. States can be used to define groups of behavior – e.g. an actor object may be "sleeping", "awake" or "dead", and its logic behavior may be different in each of these three states. The states of an object are set up, displayed and edited in the Controller logic bricks for the object.

Sensors

Introduction

Sensors are the logic bricks that cause the logic to do anything. Sensors give an output when something happens, e.g. a trigger event such as a collision between two objects, a key pressed on the keyboard, or a timer for a timed event going off. When a sensor is triggered, a positive pulse is sent to all controllers that are linked to it.

The logic blocks for all types of sensor may be constructed and changed using the *Logic Editor* details of this process are given in the *Sensor Editing* page.

The following types of sensor are currently available:

Actuator Detects when a particular actuator receives an activation pulse.

Always Gives a continuous output signal at regular intervals.

Collision Detects collisions between objects or materials.

Delay Delays output by a specified number of logic ticks.

Joystick Detects movement of specified joystick controls.

Keyboard Detects keyboard input.

Message Detects either text messages or property values

Mouse Detects mouse events.

Near Detects objects that move to within a specific distance of themselves.

Property Detects changes in the properties of its owner object.

Radar Detects objects that move to within a specific distance of themselves, within an angle from an axis.

Random Generates random pulses.

Ray Shoots a ray in the direction of an axis and detects hits.

Sensor Editing

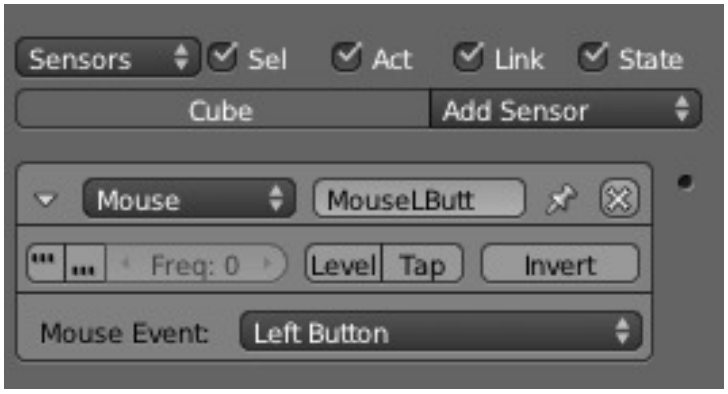


Fig. 2.2314: Sensor Column with Typical Sensor.

Blender sensors can be set up and edited in the left-hand column of the Logic Panel. This page describes the general column controls, and also those parameters which are common to all individual sensor types.

The image shows a typical sensor column with a single example sensor. At the top of this column, the column heading includes menus and buttons to control which of all the sensors in the current Game Logic are displayed.

Column Heading

The column headings contain controls to set which sensors, and the level of detail given, in the sensor column. This is very useful for hiding unnecessary sensors so that the necessary ones are visible and easier to reach. Both these can be controlled individually.

Sensors

Show Objects Expands all objects.

Hide Objects Collapses all objects to just a bar with their name.

Show Sensors Expands all sensors.

Hide Sensors Collapses all sensors to bars with their names.

It is also possible to filter which sensors are viewed using the four heading buttons:

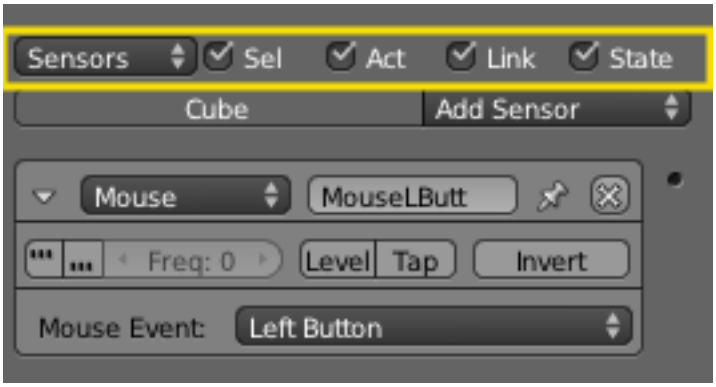


Fig. 2.2315: Sensor Column Heading.

Sel Shows all sensors for selected objects.

Act Shows only sensors belonging to the active object.

Link Shows sensors which have a link to a controller.

State Only sensors connected to a controller with active states are shown.

Object Heading

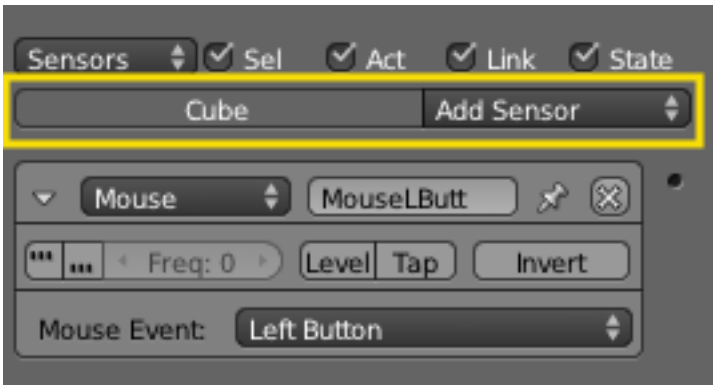


Fig. 2.2316: Sensor Object Heading.

In the column list, sensors are grouped by object. By default, sensors for every selected object appear in the list, but this may be modified by the column heading filters.

At the head of each displayed object sensor list, two entries appear:

Name The name of the object.

Add Sensor When clicked, a menu appears with the available sensor types. Selecting an entry adds a new sensor to the object. See *Sensors* for a list of available sensor types.

Sensor Common Options

All sensors have a set of common buttons, fields and menus. They are organized as follows:

Triangle button Collapses the sensor information to a single line (toggle).

Sensor type menu Specifies the type of the sensor.

Sensor name The name of the sensor. This can be selected by the user. It is used to access sensors with Python; it needs to be unique among the selected objects.

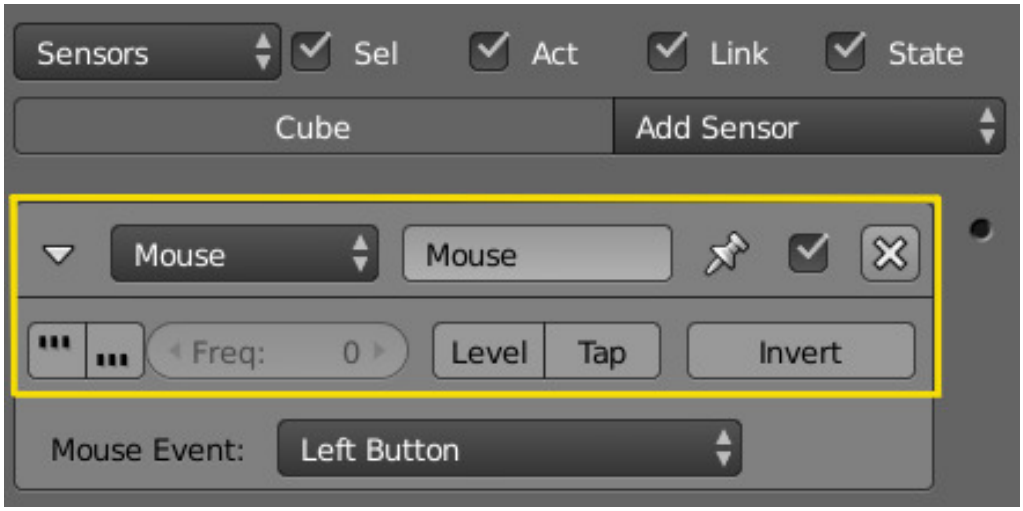


Fig. 2.2317: Common Sensor Options.

Pin button Display the sensor even when it is not linked to a visible states controller.

Checkbox button Sets active state of the sensor

X button Deletes the sensor.

Note: Note about triggers

If a controller does not get trigger by any connected sensor (regardless of the sensors' state) it will not be activated at all.

A sensor triggers the connected controllers on state change. When the sensor changes its state from negative to positive or positive to negative, the sensor triggers the connected controllers. A sensor triggers a connected controller as well when the sensor changes from deactivation to activation.

The following parameters specifies how the sensor triggers connected controllers:

True level triggering If this is set, the connected controllers will be triggered as long as the sensor's state is positive. The sensor will trigger with the delay (see parameter: frequency) of the sensor.

False level triggering If this is set, the connected controllers will be triggered as long as the sensor's state is negative. The sensor will trigger with the delay (see parameter: frequency) of the sensor.

Freq Despite it is name "Frequency", this parameter sets the delay between repeated triggers, measured in frames (also known as logic ticks). The default value is 0 and it means no delay. It is only used at least one of the level triggering parameters are enabled.

Raising the value of *freq* is a good way for saving performance costs by avoiding to execute controllers or activate actuators more often than necessary.

Examples: (Assuming the default frame rate with a frequency of 60 Hz (60 frames per second)).

freq	meaning	frames with trigger	frames without trigger	period in frames	frequency in frames/sec
0	The sensor triggers the next frame.	1	0	1	60
1	The sensor triggers at one frame and waits another one until it triggers again. It results in half speed.	1	1	2	30
29	The sensor triggers one frame and waits 29 frames until it triggers again.	1	29	30	2
59	The sensor triggers one frame and waits 59 frames until it triggers again.	1	59	30	1

***Level* Button** Triggers connected controllers when state (of the build-in state machine) changes. (For more information see *States*).

The following parameters specifies how the sensor's status gets evaluated:

***Tap* Button** Changes the sensor's state to to negative one frame after changing to positive even if the sensor evaluation remains positive. As this is a state change it triggers the connected controllers as well. Only one of *Tap* or *Level* can be activated. If the *TRUE level triggering* is set, the sensor state will consecutive change from True to False until the sensor evaluates False. The *FALSE level triggering* will be ignored when the *Tap* parameter is set.

***Invert* Button** This inverts the sensor output. If this is set, the sensor's state will be inverted. This means the sensor's state changes to positive when evaluating False and changes to False when evaluating True. If the *Tap* parameter is set, the sensor triggers the controller based on the inverted sensor state.

Sensor Types

Actuator Sensor

Fig. 2.2318: Actuator sensor.

The Actuator sensor detects when a particular actuator receives an activation pulse.

The *Actuator* sensor sends a TRUE pulse when the specified actuator is activated.

The sensor also sends a FALSE pulse when the specified actuator is deactivated.

See *Sensor Common Options* for common options.

Special Options:

Actuator Name of actuator (NB This must be owned by the same object).

Always Sensor

Fig. 2.2319: Always sensor.

The *Always* sensor is used for things that need to be done every logic tick, or at every x logic tick (with non-null f), or at start-up (with *Tap*).

See *Sensor Common Options* for common options.

This sensor does not have any special options.

Collision Sensor

Fig. 2.2320: Collision sensor.

A *Collision* sensor works like a *Touch* sensor but can also filter by property or material. Only objects with the property/material with that name will generate a positive pulse upon collision. Leave blank for collision with any object.

See *Sensor Common Options* for common options.

Special Options:

Pulse button Makes it sensible to other collisions even if it is still in touch with the object that triggered the last positive pulse.

M/P button Toggles between material and property filtering.

Note: Note about soft bodies

The *Collision* sensor cannot detect collisions with soft bodies. This is a limitation in Bullet, the physics library used by the Game Engine.

Delay Sensor

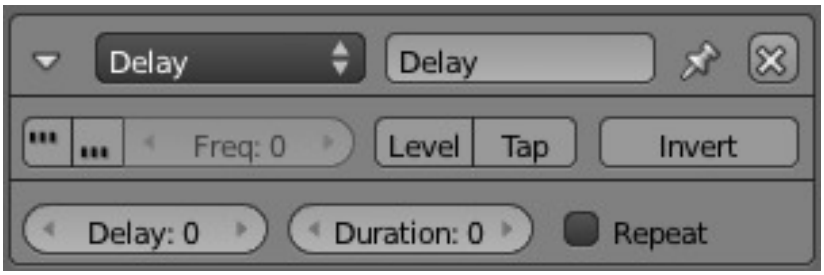


Fig. 2.2321: Delay sensor.

The *Delay* sensor is designed for delaying reactions a number of logic ticks. This is useful if an other action has to be done first or to time events.

See *Sensor Common Options* for common options. Special Options:

Delay The number of logic ticks the sensor waits before sending a positive pulse.

Duration The number of logic ticks the sensor waits before sending the negative pulse.

Repeat Button Makes the sensor restart after the delay and duration time is up.

Joystick Sensor

The *Joystick* sensor triggers whenever the joystick moves. It also detects events on a range of ancillary controls on the joystick device (hat, buttons, etc.). More than one joystick may be used (see "Index"). The exact layout of the joystick

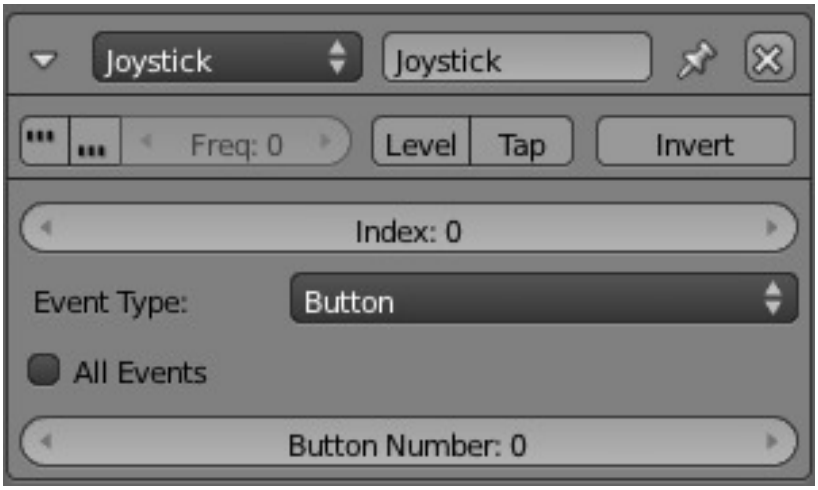


Fig. 2.2322: Joystick sensor.

controls will depend on the make and model of joystick used.

See *Sensor Common Options* for common options.

Special Options:

Index Specifies which joystick to use.

All Events Sensor triggers for all events on this joystick's current type.

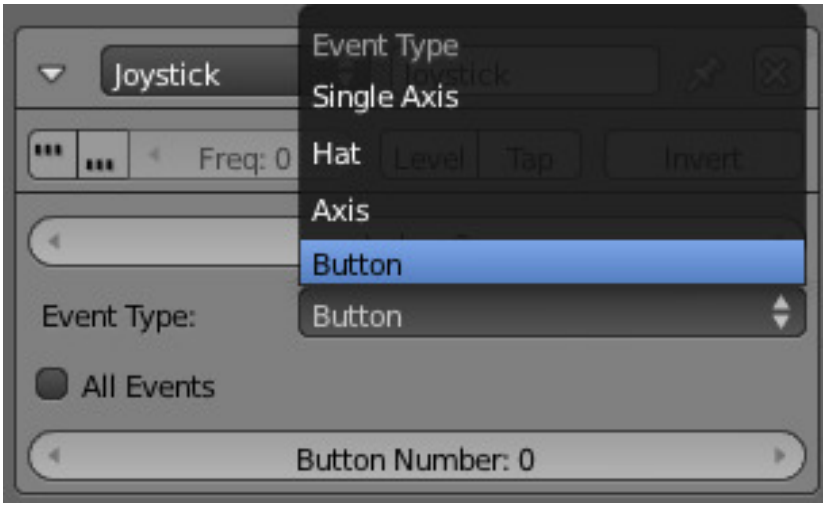


Fig. 2.2323: Joystick Events.

Event Type A menu to select which joystick event to use.

Single Axis

Single Axis Detect movement in a single joystick Axis.

Axis Number

- 1 = Horizontal axis (left/right)
- 2 = Vertical axis (forward/back)
- 3 = Paddle axis up/down

Fig. 2.2324: Joystick Single Axis.

- 4 = Joystick axis twist left/right

Axis Threshold Threshold at which joystick fires (Range 0 - 32768)

Hat

Fig. 2.2325: Joystick Hat.

Hat Detect movement of a specific hat control on the joystick.

Hat number Specifies which hat to use (max. 2).

Hat Direction Specifies the direction to use: up, down, left, right, up/right, up/left, down/right, down/left.

Axis

Axis

Axis Number Specifies the axis (1 or 2).

Axis Threshold Threshold at which joystick fires (Range 0 - 32768).

Axis Direction Specifies the direction to use:

- (Axis Number = 1) Joystick Left, Right, Up, Down

Fig. 2.2326: Joystick Axis.

- (Axis Number = 2) Paddle upper (Left); paddle Lower (Right);
- Joystick twist left (Up) Joystick twist right (Down)

Button

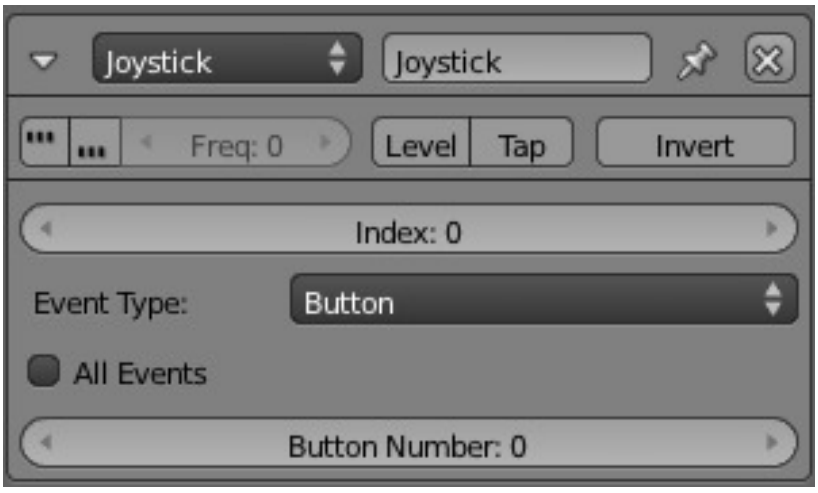


Fig. 2.2327: Joystick Button.

Button Specify the *button number* to use.

Keyboard Sensor

The *Keyboard* sensor is for detecting keyboard input. It can also save keyboard input to a *String property*.

See *Sensor Common Options* for common options.

Special Options:

Key This field detects presses on a named key. Press the button with no label and a key to assign that key to the sensor. This is the active key, which will trigger the TRUE pulse. Click the button and then click outside of the button to deassign the key.

Fig. 2.2328: Keyboard sensor.

A FALSE pulse is given when the key is released.

All keys button Sends a TRUE pulse when any key is pressed. This is useful for custom key maps with a *Python controller*.

First Modifier, Second Modifier Specifies additional key(s), all of which must be held down while the active key is pressed in order for the sensor to give a TRUE pulse. These are selected in the same way as Key. This is useful if you wish to use key combinations, for example `Ctrl-R` or `Shift-Alt-Esc` to do a specific action.

Log Toggle Assigns a *Bool* property which determines if the keystroke will or will not be logged in the target *String*. This property needs to be TRUE if you wish to log your keystrokes.

Target The name of property to which the keystrokes are saved. This property must be of type *String*. Together with a *Property* sensor this can be used for example to enter passwords.

Message Sensor

Fig. 2.2329: Message Sensor.

The *Message* sensor can be used to detect either text messages or property values. The sensor sends a positive pulse once an appropriate message is sent from anywhere in the engine. It can be set up to only send a pulse upon a message with a specific subject.

See *Sensor Common Options* for common options.

Special Options:

Subject Specifies the message that must be received to trigger the sensor (this can be left blank).

Note: See *Message Actuator* for how to send messages.

Mouse Sensor

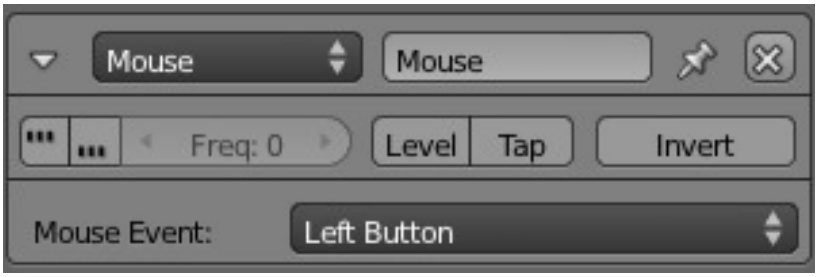


Fig. 2.2330: Mouse sensor.

The *Mouse* sensor is for detecting mouse events.

See *Sensor Common Options* for common options.

Special Options: The controller consist only of a list of types of mouse events. These are:

Mouse over any Gives a TRUE pulse if the mouse moves over any game object.

Mouse over Gives a TRUE pulse if the mouse moves over the owner object.

Movement Any movement with the mouse causes a stream of TRUE pulses.

Wheel Down Causes a stream of TRUE pulses as the scroll wheel of the mouse moves down.

Wheel Up Causes a stream of TRUE pulses as the scroll wheel of the mouse moves up.

Right button Gives a TRUE pulse.

Middle button Gives a TRUE pulse.

Left button Gives a TRUE pulse.

A FALSE pulse is given when any of the above conditions ends.

There is no logic brick for specific mouse movement and reactions (such as first person camera), these have to be coded in Python.

Near Sensor

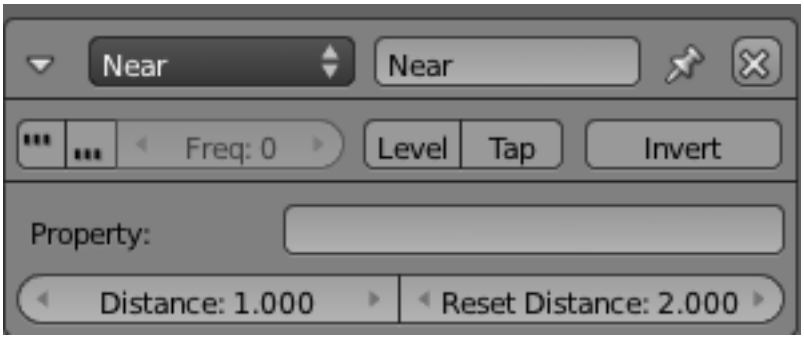


Fig. 2.2331: Near sensor.

A *Near* sensor detects objects that move to within a specific distance of themselves. It can filter objects with properties, like the *Collision* sensor.

Options

See *Sensor Common Options* for common options.

Property This field can be used to limit the sensor to look for only those objects with this property.

Distance The number of Blender units it will detect objects within.

Reset The distance the object needs to be to reset the sensor (send a FALSE pulse).

Note:

1. The Near sensor can detect objects "through" other objects (walls etc).
2. Objects must have "Actor" enabled to be detected.

Note: Note about soft bodies

The *Near* sensor cannot detect soft bodies. This is a limitation in Bullet, the physics library used by the Game Engine.

Property Sensor

Fig. 2.2332: Property sensor.

The *Property* sensor detects changes in the properties of its owner object.

See *Sensor Common Options* for common options.

Special Options:

Fig. 2.2333: Property Evaluation.

Evaluation Type Specifies how the property will be evaluated against the value(s).

Greater Than Sends a TRUE pulse when the property value is greater than the *Value* in the sensor.

Less Than Sends a TRUE pulse when the property value is less than the *Value* in the sensor.

Changed Sends a TRUE pulse as soon as the property value changes.

Interval Sends a TRUE pulse when the *Value* of the property is between the *Min* and *Max* values of the sensor.

Not Equal Sends a TRUE pulse when the property value differs from the *Value* in the sensor.

Equal Sends a TRUE pulse when the property value matches the *Value* in the sensor.

Note: The names of other properties can also be entered to compare properties.

Radar Sensor

Fig. 2.2334: Radar sensor.

The *Radar* sensor works much like a *Near* sensor, but only within an angle from an axis, forming an invisible cone with the top in the objects' center and base at a distance on an axis.

See *Sensor Common Options* for common options.

Special Options:

Property This field can be used to limit the sensor to look for only those objects with this property.

Note:

1. The Radar sensor can detect objects "through" other objects (walls etc).
2. Objects must have "Actor" enabled to be detected.

Axis This menu determines the direction of the radar cone. The ± signs is whether it is on the axis direction (+), or the opposite (-).

Angle Determines the angle of the cone. (Range: 0.00 to 179.9 degrees).

Distance Determines the length of the cone. (Blender units).

This sensor is useful for giving bots sight only in front of them, for example.

Note: Note about soft bodies

The *Radar* sensor cannot detect soft bodies. This is a limitation in Bullet, the physics library used by the Game Engine.

Random Sensor

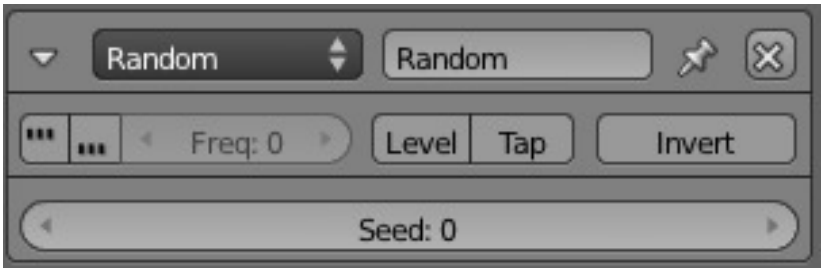


Fig. 2.2335: Random sensor.

The *Random* sensor generates random pulses.

See *Sensor Common Options* for common options.

Special Options:

Seed This field to enter the initial seed for the random number algorithm. (Range 0-1000).

Note: 0 is not random, but is useful for testing and debugging purposes.

Note: If you run several times with the same Seed, the sequence of intervals you get will be the same in each run, although the intervals will be randomly distributed.

Ray Sensor

Fig. 2.2336: Ray sensor.

The *Ray* sensor shoots a ray in the direction of an axis and sends a positive pulse once it hits something. It can be filtered to only detect objects with a given material or property.

See *Sensor Common Options* for common options.

Special Options: It shares a lot of buttons and fields with *Radar* sensor.

Property This field can be used to limit the sensor to look for only those objects with this property.

Note:

1. Unless the Property field is set, the Ray sensor can detect objects “through” other objects (walls etc).
2. Objects must have “Actor” enabled to be detected.

Axis This menu determines the direction of the ray. The ± signs is whether it is on the axis direction (+), or the opposite (-).

Range Determines the length of the ray. (Blender units).

X-Ray Mode button Makes it x-ray, so that it sees through objects that do not have the property or material specified in the filter field.

Controllers

Introduction

The controllers are the bricks that collect data sent by the sensors, and also specify the state for which they operate. After performing the specified logic operations, they send out pulse signals to drive the actuators to which they are connected.

When a sensor is activated, it sends out a positive pulse, and when it is deactivated, it sends out a negative pulse. The controllers' job is to check and combine these pulses to trigger the proper response.

The logic blocks for all types of controller may be constructed and changed using the *Logic Editor*; details of this process are given in the *Controller Editing* page.

Controller Types

There are eight types of controller logic brick to carry out the logic process on the input signal(s): these are described in the separate pages shown below:

- *AND*
- *OR*
- *XOR*
- *NAND*
- *NOR*
- *XNOR*
- *Expression*
- *Python*

This table gives a quick overview of the logic operations performed by the logical controller types. The first column, input, represents the number of positive pulses sent from the connected sensors. The following columns represent each controller's response to those pulses. True means the conditions of the controller are fulfilled, and the actuators it is connected to will be activated; false means the controller's conditions are not met and nothing will happen. Please consult the individual controller pages for a more detailed description of each controller.

Note: It is assumed that more than one sensor is connected to the controller. For only one sensor, consult the "All" line.

Positive sensors	Controllers					
	AND	*OR*	*XOR*	*NAND*	*NOR*	*XNOR*
None	False	False	False	True	True	True
One	False	True	True	True	False	False
Multiple, not all	False	True	False	True	False	True
All	True	True	False	False	False	True

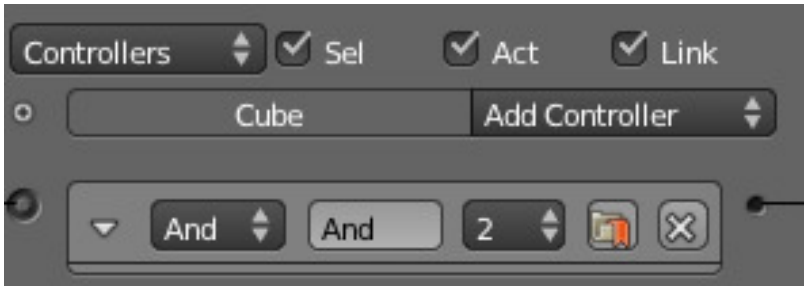


Fig. 2.2337: Controller Column with Typical Sensor.

Controller Editing

Blender controllers can be set up and edited in the central column of the Logic Panel. This page describes the general column controls, those parameters which are common to all individual controller types, and how different states for the objects in the logic system can be set up and edited.

The image shows a typical controller column with a single controller. At the top of this column, and for sensors and actuators, the column heading includes menus and buttons to control which of all the controllers in the current Game Logic are displayed.

Column Heading

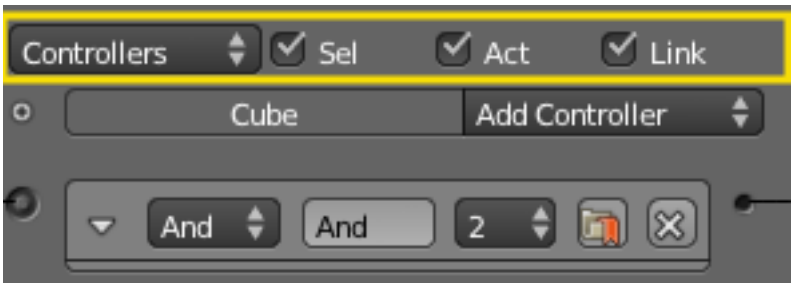


Fig. 2.2338: Controller Column Headings.

The column headings contain controls to set which controllers appear, and the level of detail given, in the controller column. This is very useful for hiding unnecessary controllers so that the necessary ones are visible and easier to reach. Both these can be controlled individually.

Controllers

Show Objects Expands all objects.

Hide Objects Collapses all objects to just a bar with their name.

Show Controllers Expands all Controllers.

Hide Controllers Collapses all Controllers to bars with their names.

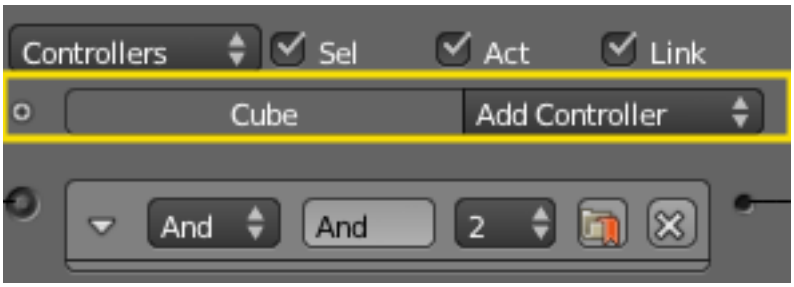
It is also possible to filter which controllers are viewed using the three heading buttons:

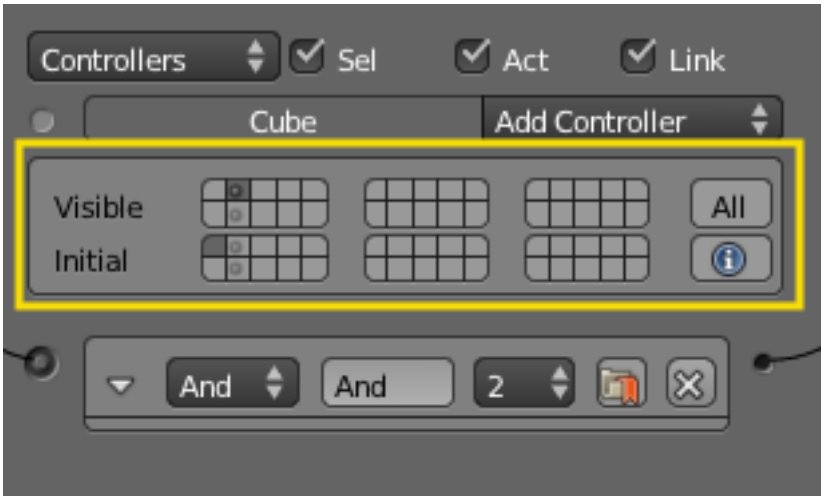
Sel Shows all controllers for selected objects.

Act Shows only controllers belonging to the active object.

Link Shows controllers which have a link to actuators/sensors.

Object Heading





In the column list, controllers are grouped by object. By default, controllers for every selected object appear in the list, but this may be modified by the column heading filters.

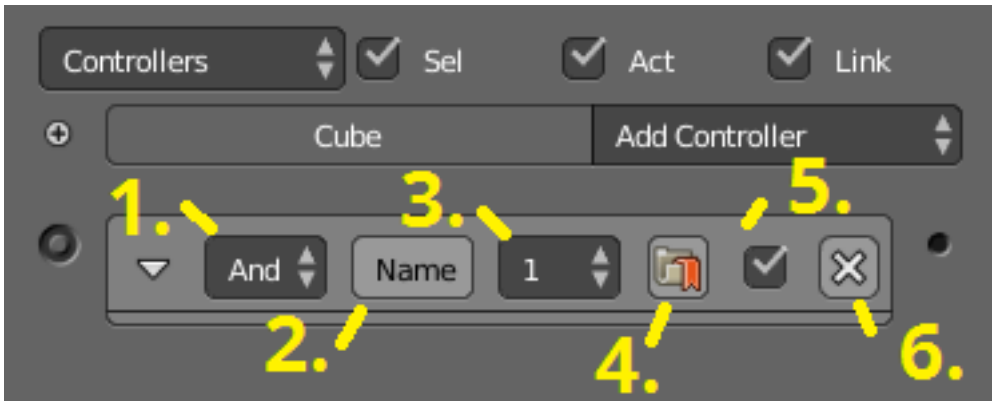
At the head of each displayed object controller list, three entries appear: *Used States Button* Shows which states are in use for the object. Detailed description of the marked panel is given in *States*.

Name The name of the object.

Add Controller When clicked, a menu appears with the available controller types. Selecting an entry adds a new controller to the object. See *Controllers* for a list of available controller types.

Standard Controller Parts

The controller heading is standard to every controller.



Controller Type menu Specifies the type of the controller.

Controller Name The name of the controller. This can be selected by the user. It is used to access controllers with Python; it needs to be unique among the selected objects.

State Index Sets the designated state for which this controller will operate.

Preference Button If on, this controller will operate before all other non-preference controllers (useful for start-up scripts).

Active Checkbox When unchecked the controller is deactivated, no pluses will be sent to the connect actuators.

X Button Deletes the sensor.

Controller Types

AND Controller

This controller gives a positive (TRUE) output when All its inputs are TRUE, and The object is in the designated State. For all other conditions the controller gives a negative (FALSE) output.

Options:

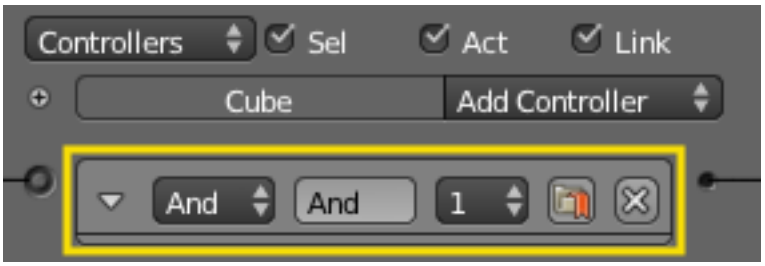


Fig. 2.2339: AND Controller.

See *standard controller parts* for descriptions of the remaining options.

OR Controller

This controller gives a positive (TRUE) output when Any one or more of its inputs are TRUE, and The object is in the designated State. For all other conditions the controller gives a negative (FALSE) output.

Options:

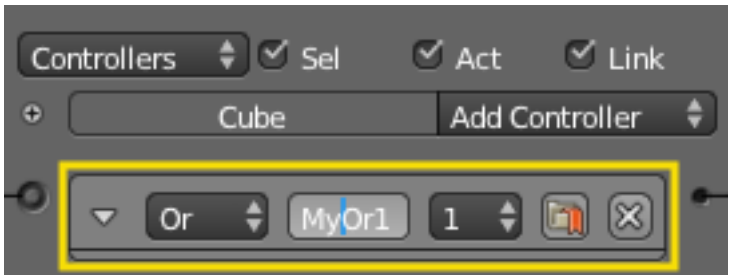


Fig. 2.2340: OR Controller.

See *standard controller parts* for descriptions of the remaining options.

NAND Controller

This controller *activates* all connected actuators if...

- the game object is in the designated state.
- at least one connected sensor triggers the controller.
- at least one connected sensor evaluated False.

This controller *deactivates* all connected actuators if...

- the game object is in the designated state.
- at least one connected sensor triggers the controller.
- ALL connected sensor evaluated True.

Options:

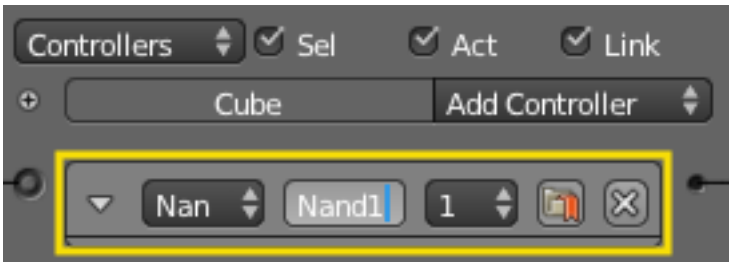


Fig. 2.2341: NAND Controller.

See *standard controller parts* for descriptions of the remaining options.

NOR Controller

This controller gives a positive (TRUE) output when None of its inputs are TRUE, and The object is in the designated State. For all other conditions the controller gives a negative (FALSE) output.

Options:

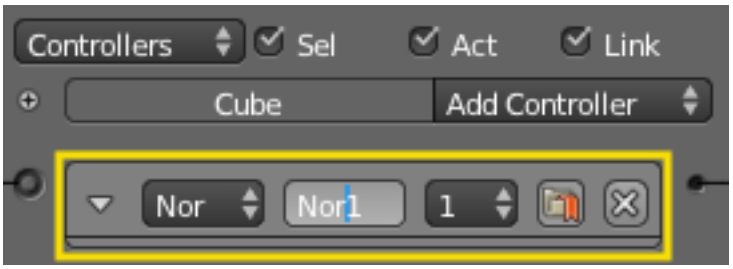


Fig. 2.2342: NOR Controller.

See *standard controller parts* for descriptions of the remaining options.

XOR Controller

This controller gives a positive (TRUE) output when One (and only one) of its inputs are TRUE, and The object is in the designated State. For all other conditions the controller gives a negative (FALSE) output.

Options:

See *standard controller parts* for descriptions of the remaining options.

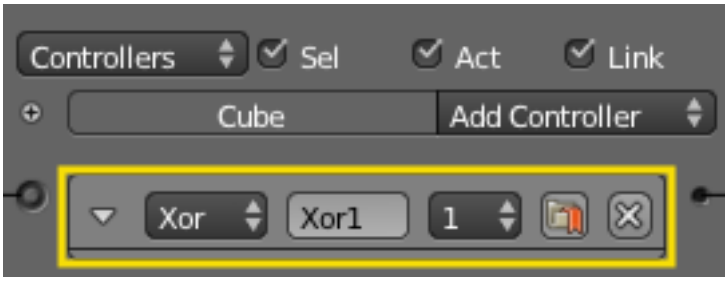


Fig. 2.2343: XOR Controller.

XNOR Controller

This controller gives a positive (TRUE) output when One (and only one) of its inputs are FALSE, and The object is in the designated State. For all other conditions the controller gives a negative (FALSE) output.

Options:

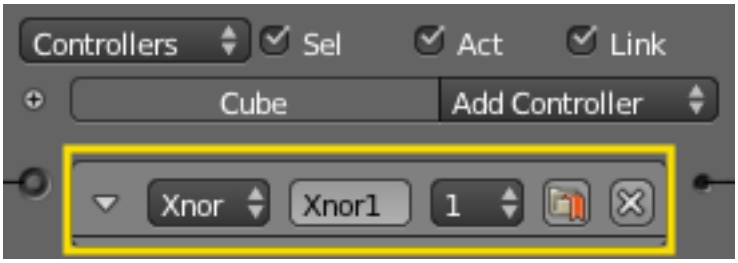


Fig. 2.2344: XNOR Controller.

See *standard controller parts* for descriptions of the remaining options.

Expression Controller

This controller evaluates a user written expression, and gives a positive (TRUE) output when The result of the expression is TRUE, and The object is in the designated State. For all other conditions the controller gives a negative (FALSE) output.

Fig. 2.2345: Expression Controller.

Expression

The expression, which is written in the box, can consist of variables, constants and operators. These must follow the rules laid out below.

Variables

You can use:

- *sensors names*,
- *properties* : assign a game property to an object and use it in a controller expression.

These cannot contain blank spaces.

Operations

Mathematical operations

Operators: `*`, `/`, `+`, `-`

Returns: a number

Examples: `3 + 2`, `35 / 5`

Logical operations

- Comparison operators: `<`, `>`, `>=`, `<=`, `==`, `!=`
- Booleans operators: `AND`, `OR`, `NOT`

Returns: `True` or `False`.

Examples: `3 > 2 (True)`, `1 AND 0 (False)`

Conditional statement (if)

Use:

```
if( expression, pulse_if_expression_is_true, pulse_if_expression_is_false )
```

If the controller evaluates `expression` to True:

- if `pulse_if_expression_is_true` is `True`, the controller sends a positive pulse to the connected actuators.
- if `pulse_if_expression_is_true` is `False`, the controller sends a negative pulse to the connected actuators.

If the controller evaluates `expression` to False:

- if `pulse_if_expression_is_false` is `True`, the controller sends a positive pulse to the connected actuators.
- if `pulse_if_expression_is_false` is `False`, the controller sends a negative pulse to the connected actuators.

Examples

Given the object has a property `coins` equal to 30:

```
coins > 20
```

returns True (the controller sends a positive pulse to the connected actuators).

Given the object has:

- a sensor called `Key_Inserted` equal to True,
- a property named `Fuel` equal to False,

```
Key_Inserted AND Fuel
```

returns False (the controller sends a negative pulse to the connected actuators).

This is the same as doing:

```
if (Key_Inserted AND Fuel, True, False)
```

Instead, you could do:

```
if (Key_Inserted AND Fuel, False, True)
```

to return a positive pulse when `Key_Inserted AND Fuel` returns False.

You can also do:

```
if ((Key_Inserted AND Fuel) OR (coins > 20), True, False)
```

This expression returns True, hence in this case the controller sends a positive pulse to the connected actuators.

Parts of the Expression Controller

Fig. 2.2346: The Expression to calculate.

See *standard controller parts* for descriptions of the remaining options.

Python Controller

The Python controller runs a Python script when a sensor triggers the controller. This Python script can interact with the scene or logic bricks through *Blender's API*.

A Python script can either run as an entire file or a single module. A file must be added in the text editor, and is identified simply by its name, not its path. Names are case sensitive. Modules are identified by the file name *without* the extension followed by a `.` and then the name of the module. For example:

A file `myscript.py` contains:

```
def myModule ():
   print("Go Open Source!");
```

The function can be accessed as `myscript.myModule`, which will run `print("Go Open Source!");` every time the controller is triggered.

The entire file can be run by setting the type to *Script* and setting the name to myscript.py.

Fig. 2.2347: Python Controller.

Parts of the Python Controller

Type Specifies whether it is a module or entire file.

Name The name of the file to be loaded.

D (Use Debug) Continuously reloads the file.

See *standard controller parts* for descriptions of the remaining options.

See also:

For more information on the Python API see:

- The API docs
- *This chapter for more Game Engine related API*.

Actuators

Introduction

Actuators perform actions, such as move, create objects, play a sound. The actuators initiate their functions when they get a positive pulse from one (or more) of their controllers.

The logic blocks for all types of actuator may be constructed and changed using the *Logic Editor*; details of this process are given in the *Actuator Editing* page.

The following types of actuator are currently available:

Action Handles armature actions. This is only visible if an armature is selected.

Camera Has options to follow objects smoothly, primarily for camera objects, but any object can use this.

Constraint Constraints are used to limit object's locations, distance, or rotation. These are useful for controlling the physics of the object in game.

Edit Object Edits the object's mesh, adds objects, or destroys them. It can also change the mesh of an object (and soon also recreate the collision mesh).

Filter 2D Filters for special effects like sepia colors or blur.

Game Handles the entire game and can do things as restart, quit, load, and save.

Message Sends messages, which can be received by other objects to activate them.

Motion Sets object into motion and/or rotation. There are different options, from "teleporting" to physically push rotate objects.

Parent Can set a parent to the object, or unparent it.

Property Manipulates the object's properties, like assigning, adding, or copying.

Random Creates random values which can be stored in properties.

Scene Manage the scenes in your blend-file. These can be used as levels or for UI and background.

Sound Used to play sounds in the game.

State Changes states of the object.

Steering Provides pathfinding options for the object.

Visibility Changes visibility of the object.

Actuator Editing

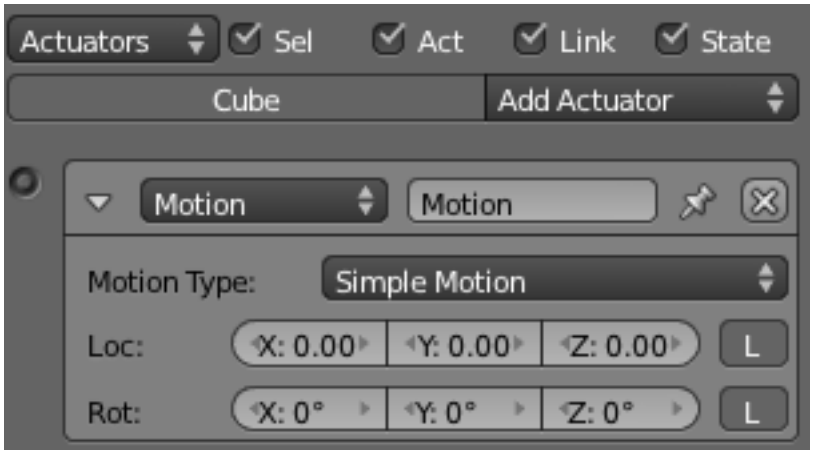


Fig. 2.2348: Actuator Column with Typical Actuator.

Blender actuators can be set up and edited in the right-hand column of the Logic Panel. This page describes the general column controls, and also those parameters which are common to all individual actuator types.

The image shows a typical actuator column with a single example actuator. At the top of this column, the column heading includes menus and buttons to control which of all the actuators in the current Game Logic are displayed.

Column Heading

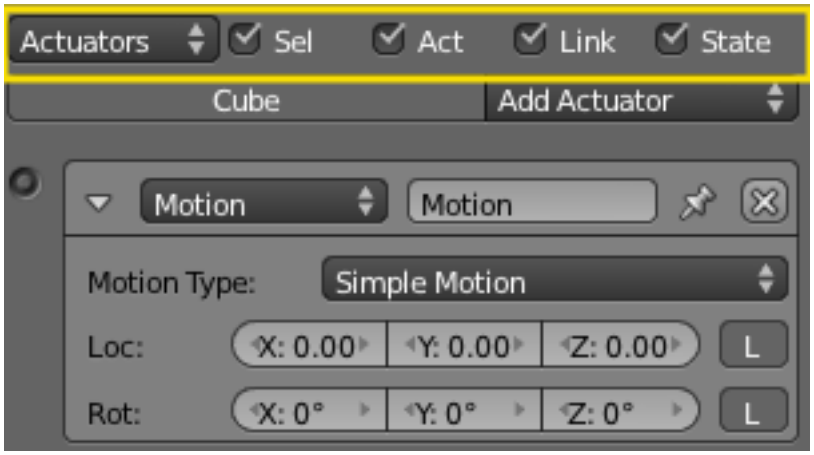


Fig. 2.2349: Actuator Column Heading.

The column headings contain controls to set which actuators, and the level of detail given, in the actuator column. This is very useful for hiding unecessary actuators so that the necessary ones are visible and easier to reach. Both these can be controlled individually.

Actuators

Show Objects Expands all objects.

Hide Objects Collapses all objects to just a bar with their name.

Show Actuators Expands all actuators.

Hide Actuators Collapses all actuators to bars with their names.

It is also possible to filter which actuators are viewed using the four heading buttons:

Sel Shows all actuators for selected objects.

Act Shows only actuators belonging to the active object.

Link Shows actuators which have a link to a controller.

State Only actuators connected to a controller with active states are shown.

Object Heading

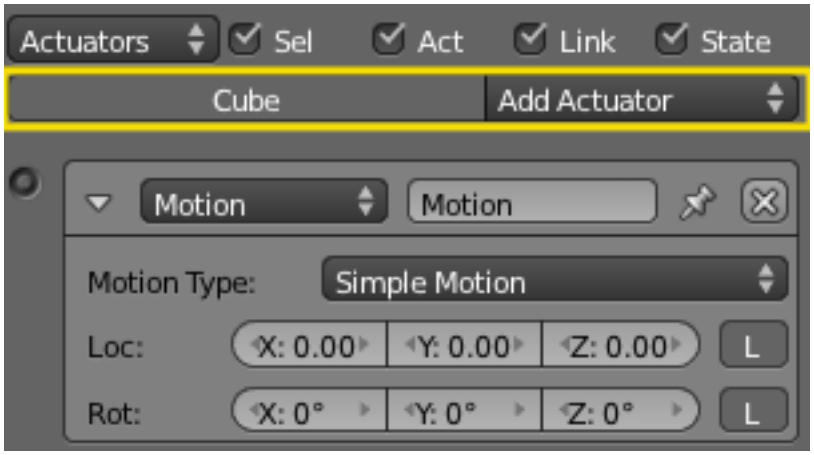


Fig. 2.2350: Actuator Object Heading.

In the column list, actuators are grouped by object. By default, actuators for every selected object appear in the list, but this may be modified by the column heading filters.

At the head of each displayed object sensor list, two entries appear:

Name The name of the object.

Add When clicked, a menu appears with the available actuator types. Selecting an entry adds a new actuator to the object. See *Actuators* for list of available actuator types.

Actuator Common Options

All actuators have a set of common buttons, fields and menus. They are organized as follows:

Triangle button Collapses the sensor information to a single line (toggle).

Actuator type menu Specifies the type of the sensor.

Actuator name The name of the actuator. This can be selected by the user. It is used to access actuators with Python; it needs to be unique among the selected objects.

X *Button* Deletes the actuator.

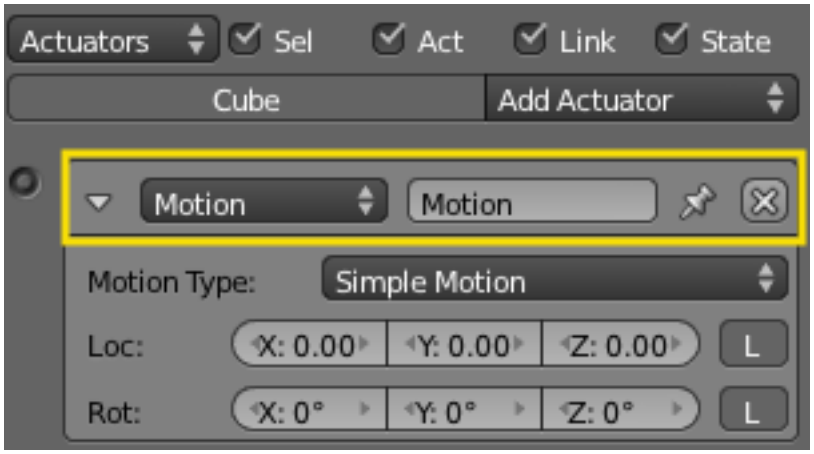


Fig. 2.2351: Common Actuator Options.

Actuators Types

Action Actuator

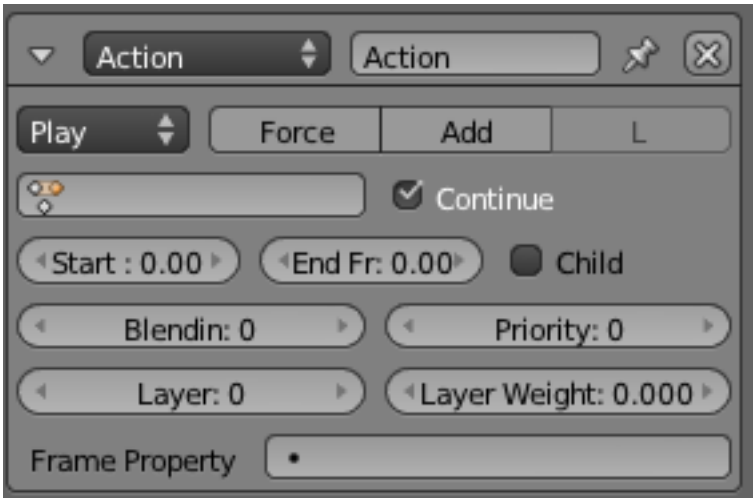


Fig. 2.2352: Action Actuator.

Actuates armature actions, and sets the playback method. The Action actuator is only visible when an armature is selected, because actions are stored in the armature.

See *Actuator Common Options* for common options.

Special Options:

Action Playback Type

Play Play F-Curve once from start to end when a TRUE pulse is received.

Ping Pong Play F-Curve once from start to end when a TRUE pulse is received. When the end is reached play F-Curve once from end to start when a TRUE pulse is received.

Flipper Play F-Curve once from start to end when a TRUE pulse is received. (Plays backwards when a FALSE pulse is received).

Loop End Play F-Curve continuously from end to start when a TRUE pulse is received.

Loop Start Play F-Curve continuously from start to end when a TRUE pulse is received.

Property Uses a property to define what frame is displayed.

Action Select the action to use

Continue Restore last frame when switching on/off, otherwise play from the start each time.

Start Frame Set the start frame of the action.

End Frame Set the end frame of the action.

Child Button Update action on all children objects as well.

Blendin Number of frames of motion blending.

Priority Execution priority – lower numbers will override actions with higher numbers. With 2 or more actions at once, the overriding channels must be lower in the stack.

Frame Property Assign the action's current frame number to this property.

Property Use this property to define the Action position. Only for Property playback type.

Layer The animation layer to play the action on.

Layer Weight How much of the previous layer to blend into this one.

Camera Actuator

Makes the camera follow or track an object.

See *Actuator Common Options* for common options.

Special Options:

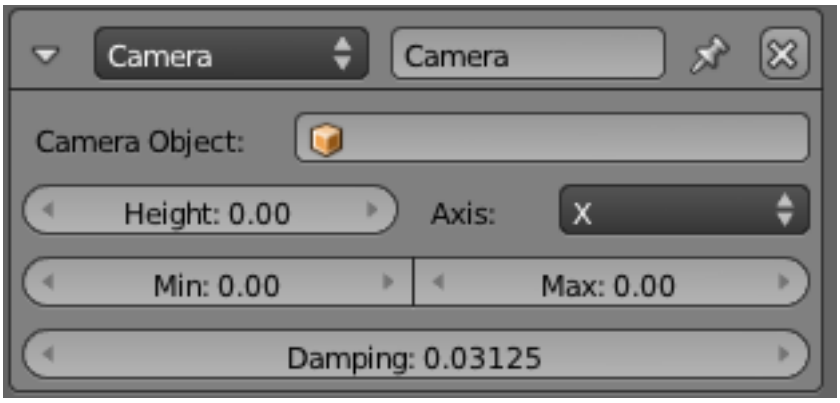


Fig. 2.2353: Camera Actuator.

Camera Object Name of the Game Object that the camera follows/tracks.

Height Height the camera tries to stay above the Game Object's object center

Axis Axis in which the Camera follows (X or Y)

Min Minimum distance for the camera to follow the Game Object

Max Maximum distance for the camera to follow the Game Object

Damping Strength of the constraint that drives the camera behind the target. Range: 0 to 10. The higher the parameter, the quicker the camera will adjust to be inside the constrained range (of min, max and height).

Constraints Actuator

Adds a constraint to the location, orientation.

See *Actuator Common Options* for common options.

Special Options:

Constraint Mode

Menu specifying type of constraint required.

- Force Field Constraint
- Orientation Constraint
- Distance Constraint
- Location Constraint

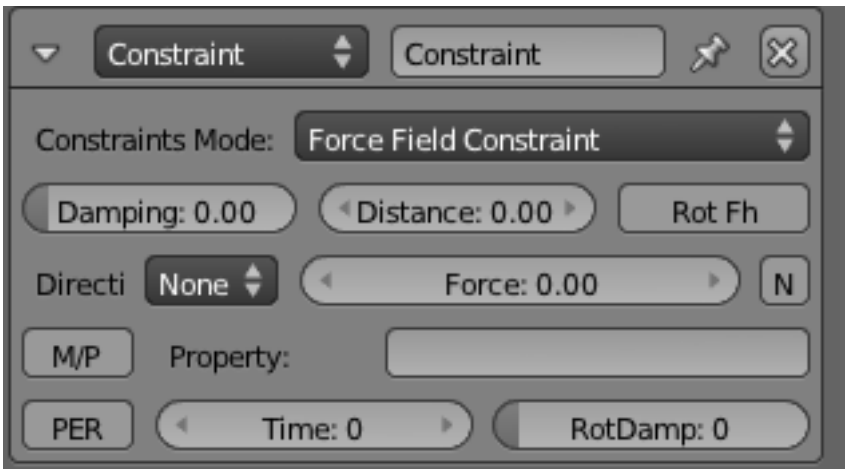


Fig. 2.2354: *Constraint actuator → Force Field.*

Force Field Constraint Create a force field buffer zone along one axis of the object.

Damping Damping factor of the Fh spring force.

Distance Height of Fh area.

Rot Fh Make game object axis parallel to the normal of trigger object.

Direction Axis in which to create force field (can be + or -, or None).

Force Force value to be used.

N When on, use a horizontal spring force on slopes.

M/P Trigger on another Object will be either Material (M) or Property (P).

Property Property/Material that triggers the Force Field constraint (blank for **all** Properties/Materials).

Per Persistence button When on, force field constraint always looks at Property/Material; when off, turns itself off if it cannot find the Property/Material.

Time Number of frames for which constraint remains active.

RotDamp Damping factor for rotation.

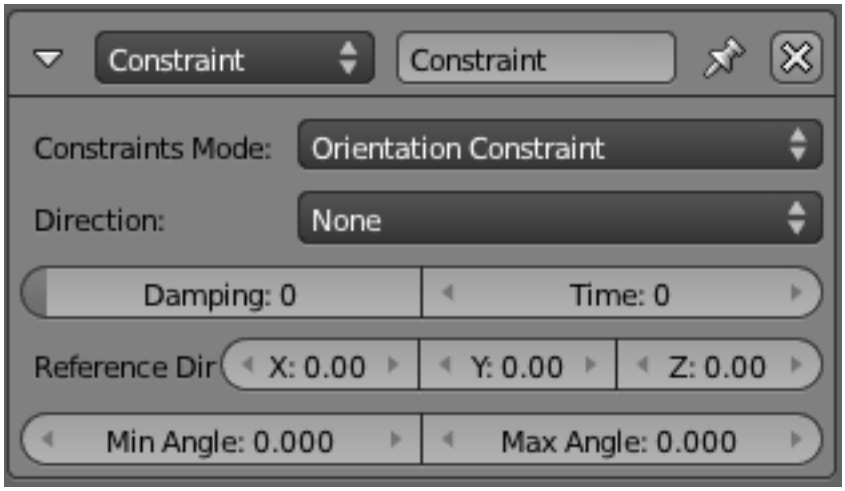


Fig. 2.2355: *Constraint Actuator → Orientation*.

Orientation Constraint

Constrain the specified axis in the Game to a specified direction in the World axis.

Direction Game axis to be modified (X, Y, Z or none).

Damping Delay (frames) of the constraint response.

Time Time (frames) for the constraint to remain active.

Reference Direction Reference direction (global coordinates) for the specified game axis.

Min Angle Minimum angle for the axis modification.

Max Angle Maximum angle for the axis modification.

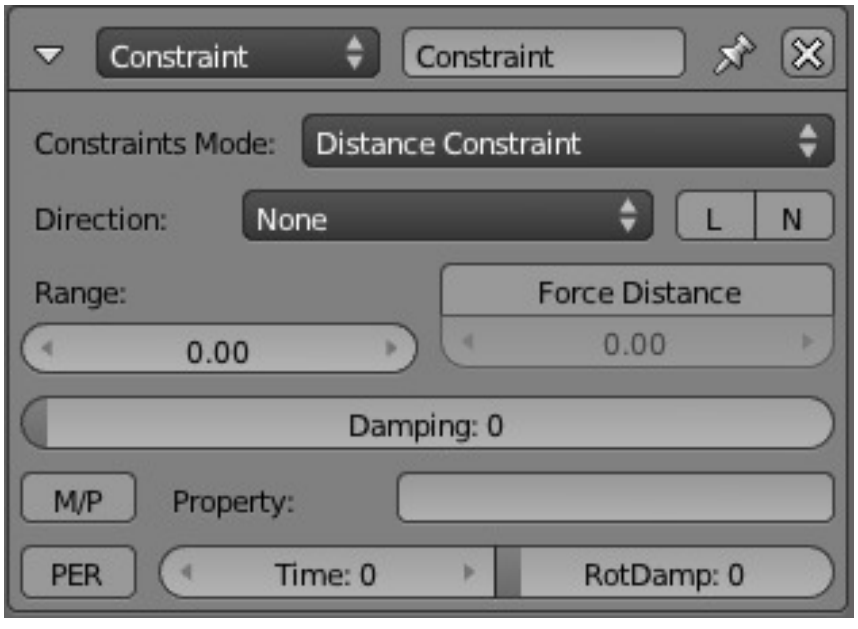


Fig. 2.2356: *Constraint actuator → Distance*.

Distance Constraint

Maintain the distance the Game Object has to be from a surface.

Direction Axis Direction (X, Y, Z, -X, -Y, -Z, or None).

L If on, use local axis (otherwise use World axis).

N If on, orient the Game Object axis with the mesh normal.

Range Maximum length of ray used to check for Material/Property on another game object.

Force Distance Distance to be maintained between object and the Material/Property that triggers the Distance Constraint (-2000 to +2000 Blender Units).

Damping Delay (frames) of the constraint response.

M/P Trigger on another Object will be either Material (M) or Property (P).

Property Property/Material that triggers the Force Field constraint (blank for **all** Properties/Materials).

Per Persistence button: When on, force field constraint always looks at Property/Material; when off, turns itself off if it cannot find the Property/Material.

Time Number of frames for which constraint remains active.

Rotation Damping Damping factor for rotation.

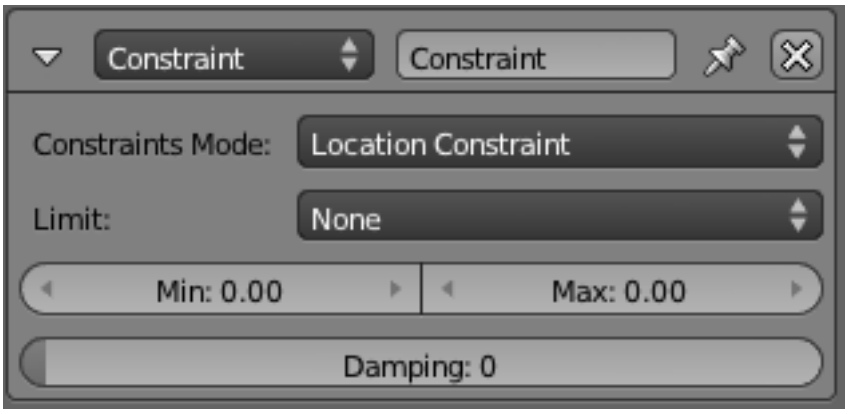


Fig. 2.2357: *Constraint actuator → Location*.

Location Constraint

Limit the position of the Game Object within one World Axis direction. To limit movement within an area or volume, use two or three constraints.

Limit Axis in which to apply limits (LocX, LocY, LocZ or none).

Min Minimum limit in specified axis (Blender Units).

Max Maximum limit in specified axis (Blender Units).

Damping Delay (frames) of the constraint.

Edit Object Actuator

The Edit Object actuator allows the user to edit settings of objects in game.

See *Actuator Common Options* for common options.

Special Options:

Edit Object Menu of options for Edit Object actuator.

- Dynamics
- Track To

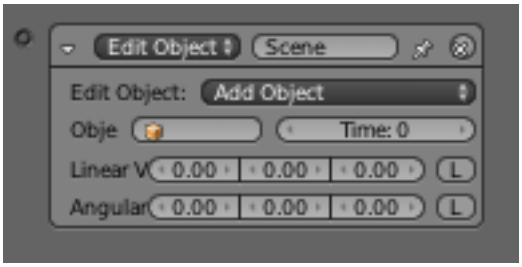


Fig. 2.2358: Edit Object actuator.

- Replace Mesh
- End Object
- Add Object

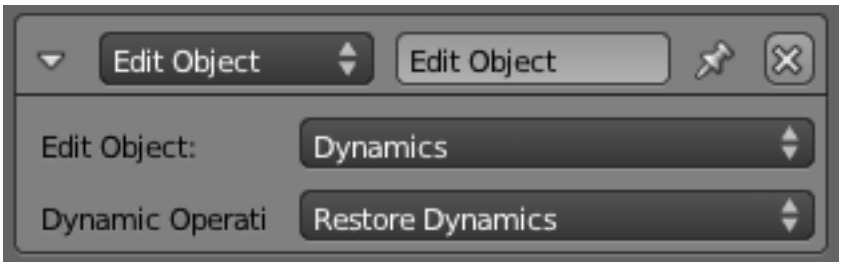


Fig. 2.2359: *Edit Object actuator → Dynamics*.

Dynamics Provides a menu of *Dynamic Operations* to set up dynamics options for object.

Set Mass Enables the user to set the mass of the current object for Physics (Range 0 - 10,000).

Disable Rigid Body Disables the Rigid Body state of the object – disables collision.

Enable Rigid Body Disables the Rigid Body state of the object – enables collision.

Suspend Dynamics Suspends the object dynamics (object velocity).

Restore Dynamics Resumes the object dynamics (object velocity).

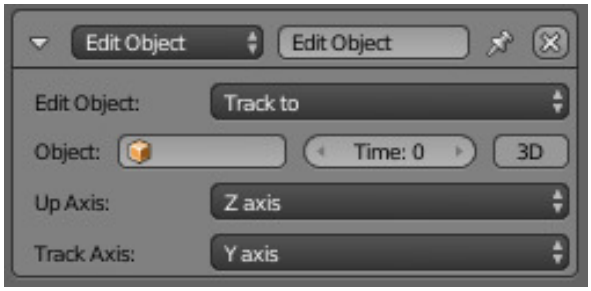


Fig. 2.2360: *Edit Object actuator → Track to*.

Track To Makes the object "look at" another object, in 2D or 3D. The Y-axis is considered the front of the object.

Object Object to follow.

Time No. of frames it will take to turn towards the target object (Range 0-2000).

3D Button (toggle). Enable 2D (X, Y) or 3D (X, Y, Z) tracking.

Replace Mesh Replace mesh with another. Both the mesh and/or its physics can be replaced, together or independently.

Mesh name of mesh to replace the current mesh.

Fig. 2.2361: *Edit Object actuator → Replace Mesh.*

Gfx Button replace visible mesh.

Phys Button replace physics mesh (not compound shapes)

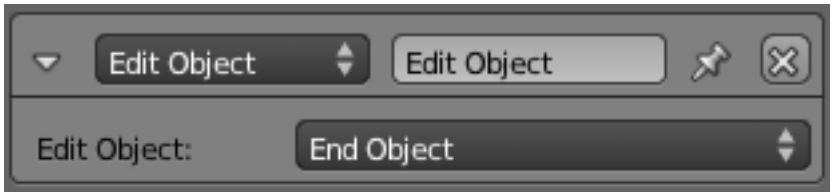


Fig. 2.2362: *Edit Object actuator → End Object.*

End Object

Destroy the current object (Note, debug properties will display error Zombie Object in console)

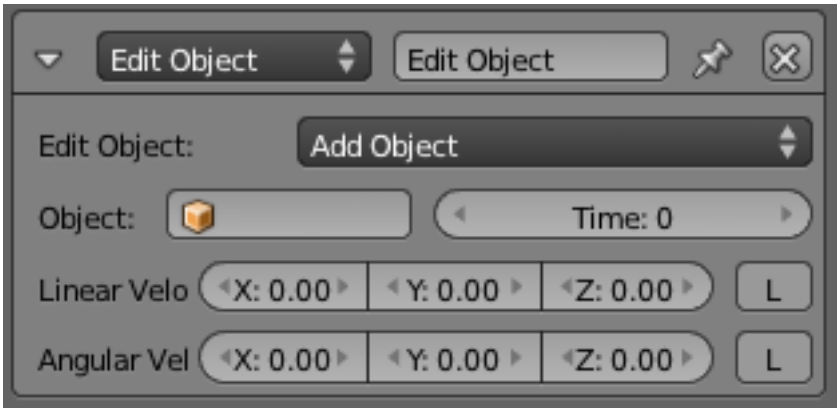


Fig. 2.2363: *Edit Object actuator → Add Object.*

Add Object

Adds an object at the center of the current object.

The object that is added needs to be on another, hidden, layer.

Object The name of the object that is going to be added.:

Time The time (in frames) the object stays alive before it disappears. Zero makes it stay forever.

Linear Velocity Linear Velocity, works like in the motion actuator but on the created object instead of the object itself. Useful for shooting objects, create them with an initial speed.

Angular Velocity Angular velocity, works like in the motion actuator but on the created object instead of the object itself.

Filter 2D Actuator

2D Filter s are image filtering actuators, that apply on final render of objects.

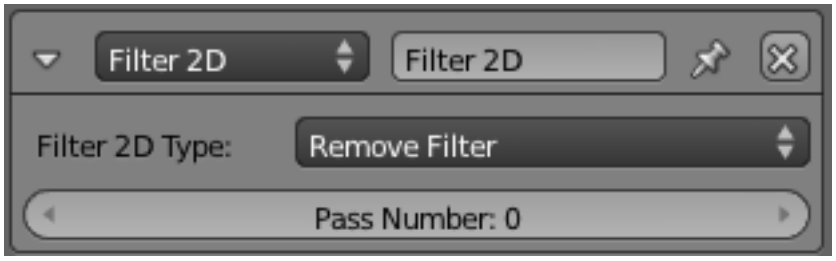


Fig. 2.2364: Edit Object actuator.

Filter 2D Type

Select the type of 2D Filter required.

- Custom Filter
- Invert
- Sepia
- Gray Scale
- Prewitt
- Sobel
- Laplacian
- Erosion
- Dilation
- Sharpen
- Blur
- Motion Blur
- Remove Filter
- Disable Filter
- Enable Filter

Only one parameter is required for all filters:

Pass Number The pass number for which this filter is to be used.

Details of the filters are given in the descriptive text below.

Motion Blur

Motion Blur is a *2D Filter* that needs previous rendering information to produce motion effect on objects. Below you can see *Motion Blur* filter in Blender window, along with its logic bricks:

You can enable Motion Blur filter using a *Python* controller:

Fig. 2.2365: 2D Filters: Motion Blur.

```
from bge import render
render.enableMotionBlur(0.85)
```

And disable it:

```
from bge import render
render.disableMotionBlur()
```

Note: Your graphic hardware and OpenGL driver must support accumulation buffer (`glAccum` function).

Built-In 2D Filters

All 2D filters you can see in *2D Filter* actuator have the same architecture, all built-in filters use fragment shader to produce final render view, so your hardware must support shaders.

Fig. 2.2366: 2D Filters: Motion Blur.

Blur, Sharpen, Dilation, Erosion, Laplacian, Sobel, Prewitt, Gray Scale, Sepia and Invert Are built-in filters. These filters can be set to be available in some passes.

To use a filter you should:

- Create appropriate sensor(s) and controller(s).
- Create a *2D Filter* actuator.
- Select your filter, for example *Blur*.

Fig. 2.2367: 2D Filters: Sepia.

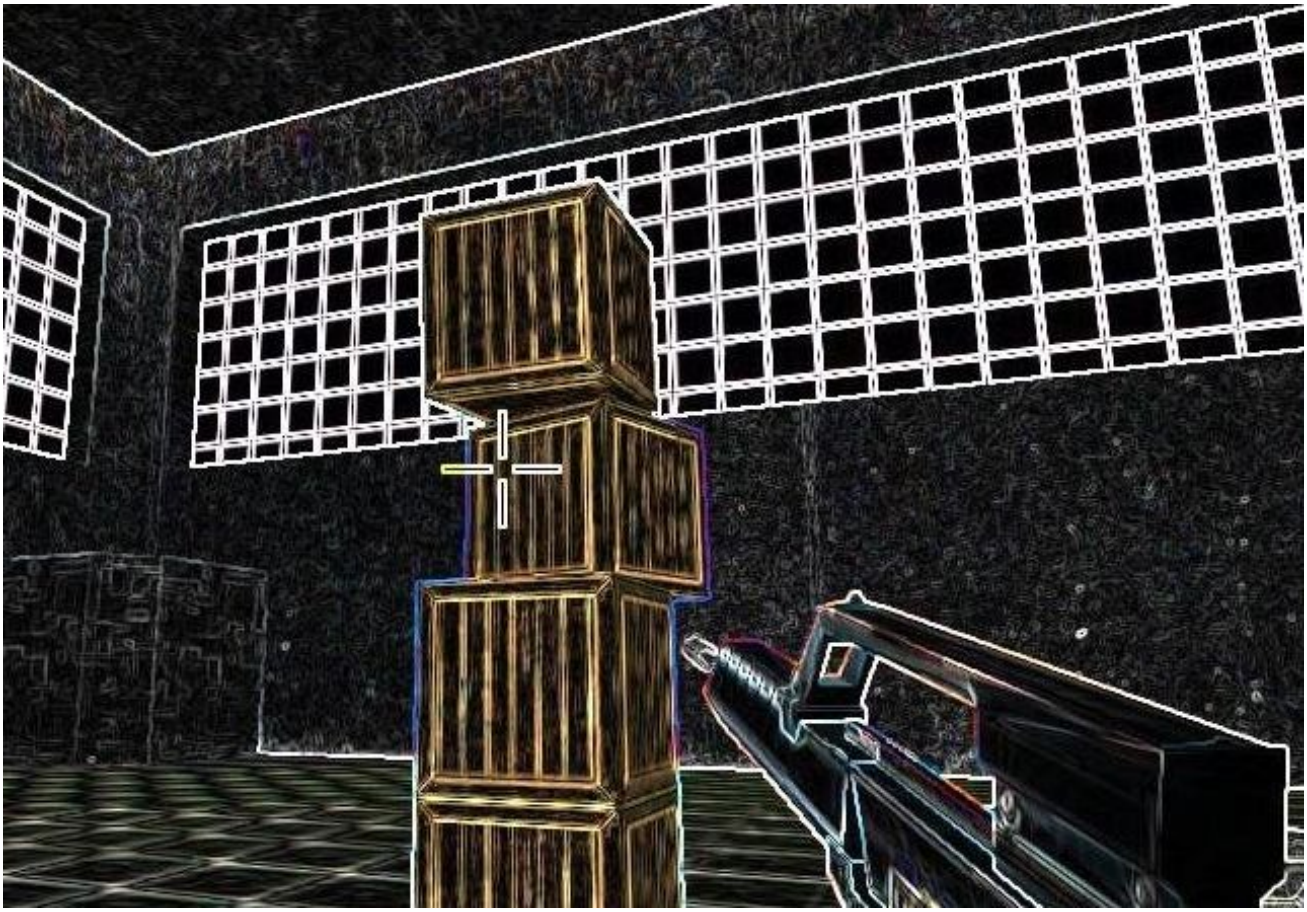


Fig. 2.2368: 2D Filters: Sobel.

- Set the pass number that the filter will be applied.

To remove a filter on a specific pass:

- Create appropriate sensor(s) and controller(s).
- Create a *2D Filter* actuator.
- Select *Remove Filter*.
- Set the pass number you want to remove the filter from it.

To disable a filter on a specific pass:

- Create appropriate sensor(s) and controller(s).
- Create a *2D Filter* actuator.
- Select *Disable Filter*.
- Set the pass number you want to disable the filter on it.

To enable a filter on a specific pass:

- Create appropriate sensor(s) and controller(s)
- Create a *2D Filter* actuator.
- Select *Enable Filter*.
- Set the pass number you want to enable the filter on it.

Custom Filters

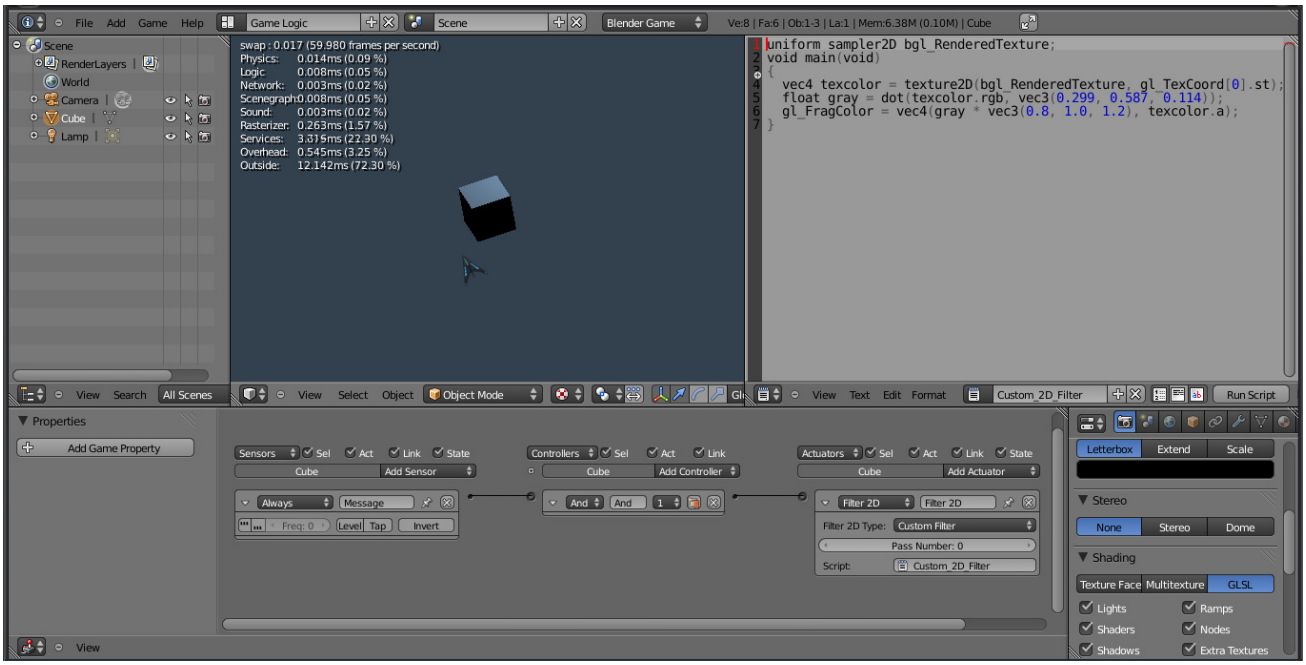


Fig. 2.2369: 2D Filters: Custom Filter.

Custom filters give you the ability to define your own 2D filter using GLSL. Its usage is the same as built-in filters, but you must select *Custom Filter* in *2D Filter* actuator, then write shader program into the Text Editor, and then place shader script name on actuator.

Blue Sepia Example:

```
uniform sampler2D bgl_RenderedTexture;
void main(void)
{
  vec4 texcolor = texture2D(bgl_RenderedTexture, gl_TexCoord[0].st);
  float gray = dot(texcolor.rgb, vec3(0.299, 0.587, 0.114));
  gl_FragColor = vec4(gray * vec3(0.8, 1.0, 1.2), texcolor.a);
}
```

Game Actuator

The Game actuator allows the user to perform Game-specific functions, such as Restart Game, Quit Game and Load Game.

See *Actuator Common Options* for common options.

Special Options:

Fig. 2.2370: Game actuator.

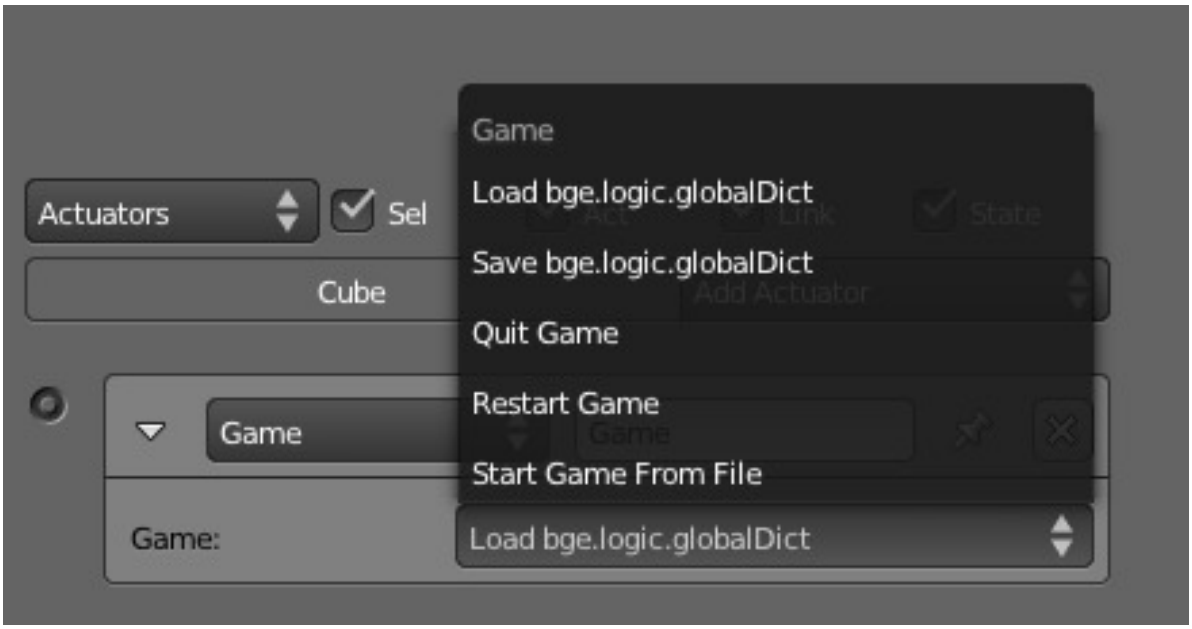


Fig. 2.2371: Game.

Game

Load `bge.logic.globalDict` Load `bge.logic.globalDict` from `.bgeconf`.

Save `bge.logic.globalDict` Save `bge.logic.globalDict` to `.bgeconf`.

Quit Game Once the actuator is activated, the blenderplayer exits the runtime.

Restart Game Once the actuator is activated, the blenderplayer restarts the game (reloads from file).

Start Game From File Once the actuator is activated, the blenderplayer starts the blend-file from the path specified.

File Path to the blend-file to load.

Note: If you use the keyboard sensor as a hook for `Esc`, in the event that the quit game actuator fails, such as an error in a Python file, the game will be unable to close. Data may be recovered from `quit.blend` *File → Recover Last Session*

Message Actuator

The Message actuator allows the user to send data across a scene, and between scenes themselves.

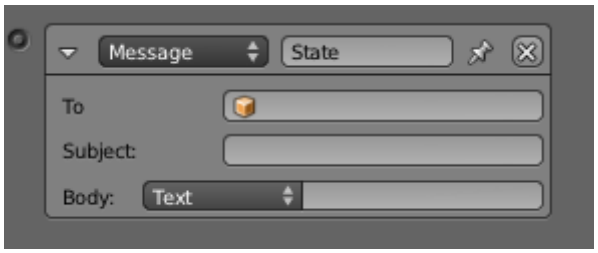


Fig. 2.2372: Message actuator.

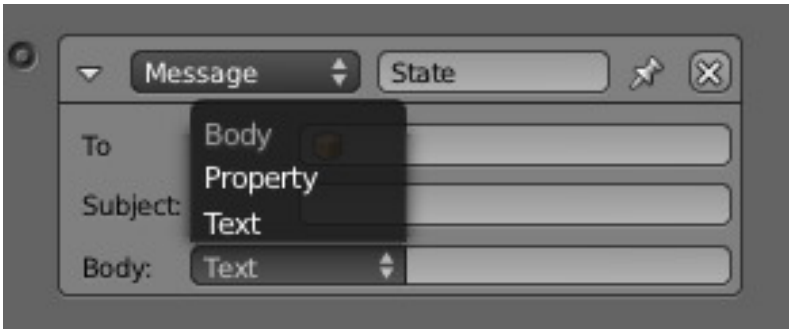


Fig. 2.2373: Message actuator Options.

See *Actuator Common Options* for common options.

Special Options:

To Object to broadcast to. Leave blank if broadcast to all (or sending to another scene).

Subject Subject of message. Useful if sending certain types of message, such as "end-game", to a message sensor listening for "end game" `and` Quit Game actuator.

Body Body of message sent (only read by Python).

Text User specified text in body.

Property User specified property.

Note: You can use the Message Actuator to send data, such as scores to other objects, or even across scenes! (alternatively use `bge.logic.globalDict`).

Mouse Actuator

Todo.

Motion Actuator

The Motion actuator sets an object into motion. There are two modes of operation, Simple or Servo, in which the object can either teleport & rotate, or dynamically move.

See also:

Actuator Common Options for common options.

Special Options:

Motion Type

Which determines the type of motion:

Simple Motion Applies a change in location and/or rotation directly.

Servo Control Sets a target speed, and also how quickly it reaches that speed.

The *Simple Motion* actuator gives control over position and velocity, but does this as an instant displacement; the object never passes any of the coordinates between the start and end positions. This can interfere with the physical simulation of other objects, and can cause an object to go through another object. The *Servo Control* actuator does not suffer from this, since it produces physically correct velocities, and leaves updating the position to the physics simulation.

Simple Motion

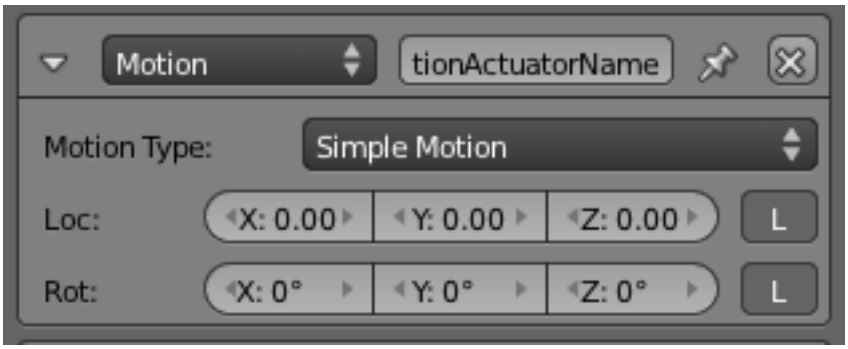


Fig. 2.2374: Motion actuator for Simple Motion.

Loc The object jumps the number of Blender units entered, each time a pulse is received.

Rot The object rotates by the specified amount, each time a pulse is received.

L Coordinates specified are Global (gray) or Local (White).

Servo Control

The Servo Control actuator influences the velocity of a game object by applying forces, resulting in correct behavior when colliding with other objects controlled by the physics simulation. The amount of force necessary is determined by a PID controller, a type of controller that is often used in control systems. Only the positional velocity is influenced by this actuator; it does not control rotation at all, and it controls position only indirectly.

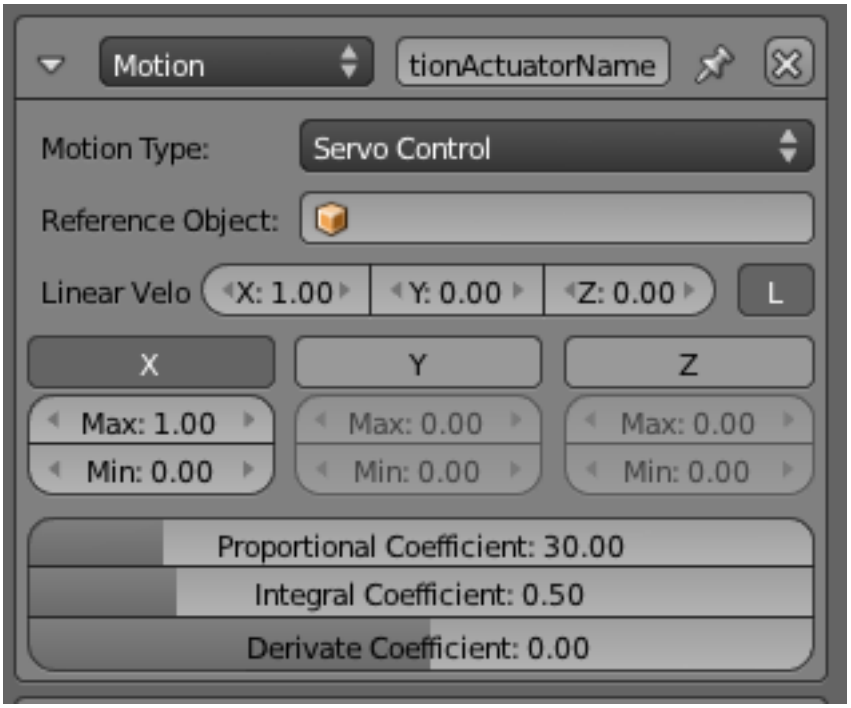


Fig. 2.2375: Motion actuator set to *Servo Control*.

Controlling the position is not necessary in that respect; that is left to a player moving the object via direction-type controls (such as the WSAD keys in a first person shooter). In such a scenario, each direction-key sensor should be attached to a different Servo Control actuator setting a different target velocity.

Tip: To use the Servo Control actuator, it is necessary to set the object's Physics Type to "Dynamic" or "Rigid Body", and to mark the object as "Actor" in the same panel. This actuator does not work with the Character physics type.

Reference Object Specifies the object which the actuator uses as a reference for the velocity. When set, it will use a velocity relative to that object instead of absolute (i.e. world-relative) velocity. Use this for a player object standing on a moving platform.

Linear Velocity The target linear velocity for the object.

L Determines whether the Linear Velocity specified are in Local (button depressed) or Global (button released) coordinates.

X, Y, Z force limits Sets minimum and maximum limits for the force applied to the object. If disabled (i.e. X, Y or Z buttons are depressed) the force applied is unlimited.

The following three coefficients determine the response to the *velocity error*, which is the difference between the target velocity and the object's actual velocity.

Proportional Coefficient This controls the reaction proportional to the velocity error. Small values cause smooth (but possibly too slow) changes in velocity. Higher values cause rapid changes, but may cause overshooting.

Integral Coefficient This controls the reaction to the sum of errors so far. Using only the Proportional component results in a systematic velocity error if there is friction: some velocity delta is necessary to produce the force that compensates the friction. Using the Integral component suppresses this effect (the target velocity is achieved on average) but can create oscillations; the control will speed to compensate the initial velocity error. To avoid the oscillation, the Proportional component must be used with the Integral component (the Proportional component damps the control) This is why the GUI sets the Proportional Coefficient systematically when you change the Integral Coefficient.

Derivative Coefficient Set the Derivative Coefficient. This dampens the acceleration when the target velocity is almost reached.

Parent Actuator

Enables you to change the parent relationships of the current object.

See *Actuator Common Options* for common options.

Special Options:

Scene

Menu for parenting operation required.

Fig. 2.2376: Parent Actuator.

Set Parent Make this object to be current object's parent.

Parent Object Name of parent object.

Compound' Add this object shape to the parent shape (only if the parent shape is already compound).

Ghost' Make this object ghost while parented.

Remove Parent Remove all parents of current object.

Parent Object Name of parent object.

Property Actuator

Using the Property actuator you can change the value of a given property once the actuator itself is activated.

See also:

Actuator Common Options for common options.

Special Options:

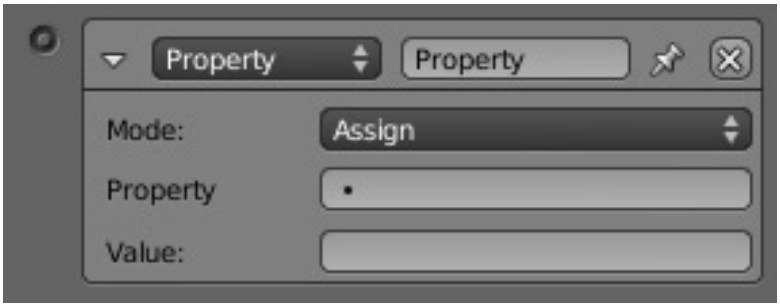


Fig. 2.2377: Property actuator.

Mode

Assign the *Property* target property will become equal to the set *Value* once the actuator is activated

Add adds *Value* to the value of the property *Property* once the actuator is activated (enter a negative value to decrease). For *Bool*, a value other than 0 (also negative) is counted as True.

Copy copies a property from another object to a property of the actuator owner once the actuator is activated.

Toggle switches 0 to 1 and any other number than 0 to 0 once the actuator is activated. Useful for on/off switches.

Property The target property that this actuator will change.

Value The value to be used to change the property.

Example

You have a character, it has a property called "hp" (hit points) to determine when he has taken enough damage to die. hp is an int with the start value of 100.

You set up two *Collision* sensors, one for enemy bullets, and one for picking up more health. The first one is connected (through an *AND* controller) to an *Add Property* actuator with the property hp and the value -10. Every time the player is hit by an enemy bullet he loses 10 hp. The other sensor is connected (through an *AND* controller) to an other *Add Property* actuator, this one with the value 50. So every time the player collides with a health item the hp increases by 50. Next you set up a *Property* sensor for an interval, greater than 100. This is connected (through an *AND* controller) to an *Assign Property* actuator which is set to 100. So if the players hp increases over 100 it is set to 100.

Random Actuator

Sets a random value into a property of the object.

See also:

Actuator Common Options for common options.

Special Options:

Fig. 2.2378: Camera Actuator.

Seed

Starting seed for random generator (range 1 - 1000).

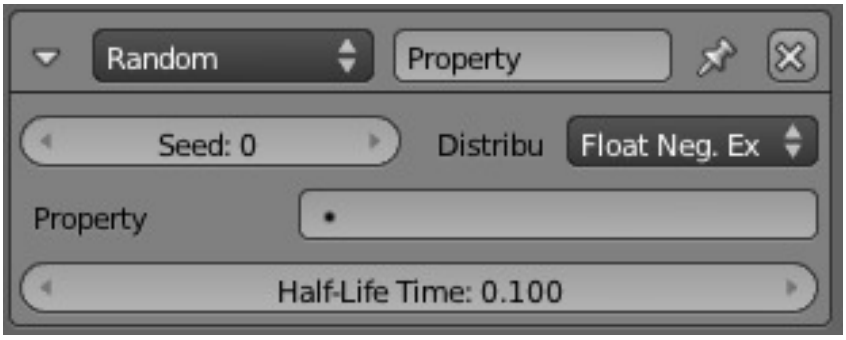


Fig. 2.2379: Float Neg. Exp.

Distribution

Menu of distributions from which to select the random value. The default entry of Boolean Constant gives either True or False, which is useful for test purposes.

Float Neg. Exp. Values drop off exponentially with the specified half-life time.

Property Float property to receive value.

Half-Life Time Half-life time (Range 0.00 - 10000.00).

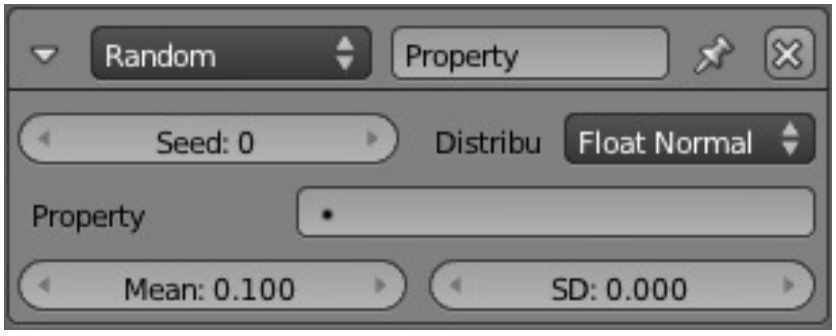


Fig. 2.2380: Float Normal.

Float normal Random numbers from a normal distribution.

Property Float property to receive value.

Mean Mean of normal distribution (Range -10000.00 to +10000.00).

SD Standard deviation of normal distribution (Range 0.00 to +10000.00).

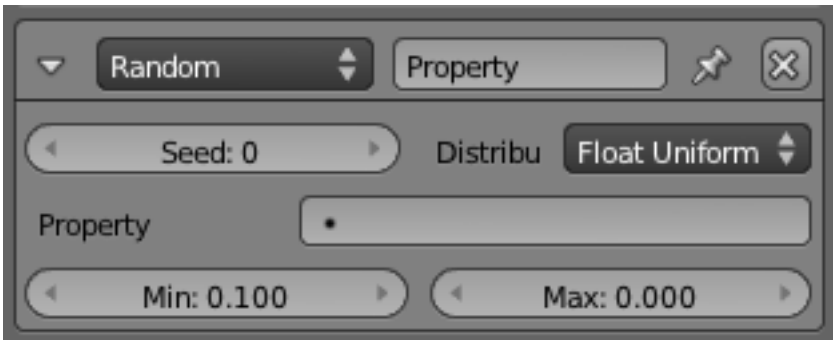


Fig. 2.2381: Float Uniform.

Float uniform Random values selected uniformly between maximum and minimum.

Property Float property to receive value.

Min Minimum value (Range -10000.00 to +10000.00).

Max Maximum value (Range -10000.00 to +10000.00).

Fig. 2.2382: Float Constant.

Float constant Returns a constant value.

Property Float property to receive value.

Value Value (Range 0.00 to +1.00).

Fig. 2.2383: Random Integer Poisson.

Int Poisson Random numbers from a Poisson distribution.

Property Integer property to receive value.

Mean Mean of Poisson distribution (Range 0.01 to +100.00).

Fig. 2.2384: Random Integer Uniform.

Int uniform Random values selected uniformly between maximum and minimum.

Property Integer property to receive value.

Min Minimum value (Range -1000 to +1000).

Max Maximum value (Range -1000 to +1000).

Fig. 2.2385: Random Integer Constant.

Int constant Returns a constant value.

Property Integer property to receive value.

Value Value (Range 0.00 to +1.00).

Fig. 2.2386: Random Bool Bernoulli.

Bool Bernoulli Returns a random distribution with specified ratio of TRUE pulses.

Property Boolean property to receive value.

Chance Proportion of TRUE responses required.

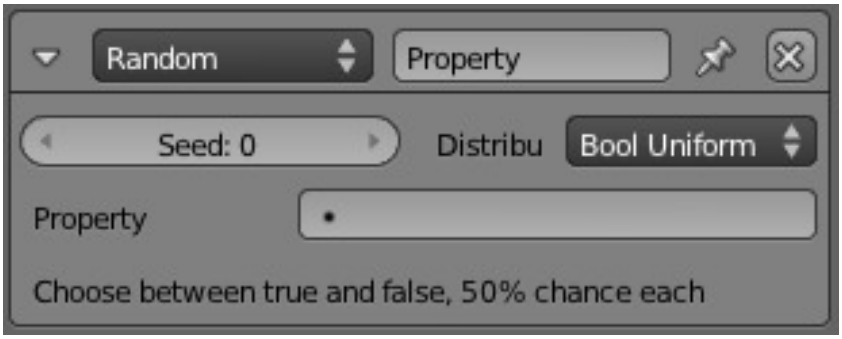


Fig. 2.2387: Random Bool Uniform.

Bool uniform A 50/50 chance of obtaining True/False.

Property Boolean property to receive value.

Bool constant Returns a constant value.

Fig. 2.2388: Random Bool Constant.

Property Boolean property to receive value.

Value Value (True or False).

Scene Actuator

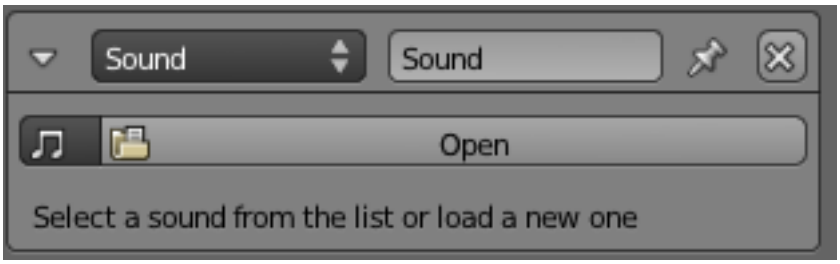


Fig. 2.2389: Scene actuator.

The *Scene* actuator manages the scenes in your blend-file, these can be used as levels or for UI and background.

See *Actuator Common Options* for common options.

Special Options: The actuator has eight modes:

Fig. 2.2390: Scene actuator options.

Restart Restarts the current scene, everything in the scene is reset.

Set Scene Changes scene to selected one.

Set Camera Changes which camera is used.

Add Overlay Scene This adds an other scene, and draws it on top of the current scene. It is good for interfacing: keeping the health bar, ammo meter, speed meter in an overlay scene makes them always visible.

Add Background Scene This is the opposite of an overlay scene, it is drawn behind the current scene.

Remove Scene Removes a scene.

Suspend Scene Pauses a scene.

Resume Scene Resumes a paused scene.

Note: A scene that it is paused cannot resume itself. You need an active scene to resume other scene that it is paused.

Steering Actuator

The steering actuator moves an object towards a target object, with options to seek, flee, or follow a path. This actuator will not actually try to avoid obstacles by deviating the objects course.

See *Actuator Common Options* for common options.

Options

Fig. 2.2391: Steering Actuator Panel.

Behavior Seek, Flee or Path following

Target Object The game object to seek.

Navigation Mesh Object: The name of the navigation mesh object used by the Steering Actuator when in Path following behavior. The game object will use the Navigation Mesh to create a path to follow the Target Object.

Tip: You can create your own mesh to use for navigation and make it a Navigation Mesh in:

Properties editor → Physics → Physics panel → choosing Physics Type: Navigation Mesh

Or you can let Blender create a Navigation Mesh, then select a mesh. (Floor or ground or etc.)

Properties editor → Scene → Navigation mesh object panel → Build navigation mesh

---

Distance The maximum distance for the game object approach the Target Object.

Velocity The velocity used to seek the Target Object.

Acceleration The maximum acceleration to use when seeking the Target Object.

Turn Speed The maximum turning speed to use when seeking the Target Object.

Facing Set a game object axis that always faces the Target Object.

Axis The game object axis that always faces the Target Object. Options are: Positive (X, Y, Z) and Negative (-X, -Y, -Z).

Axis N Use the Normal of the Navigation Mesh to align the up vector of the game object.

Self Terminated

Disabled Stops moving toward the Target Object once it reaches the maximum distance to approach the Target Object. Will follow the Target Object if it moves further away than the maximum distance.

Enabled Stops moving toward the Target Object once it reaches the maximum distance to approach the Target Object. Will not follow even if the Target Object moves further away than the maximum distance.

Visualize This checkbox let the user specify whether to show or not the debug informations of the actuator. It is also necessary to enable Debug Properties in the Display menu of the *Render* tab.

Sound Actuator

Select a sound file from the list or make a new one.

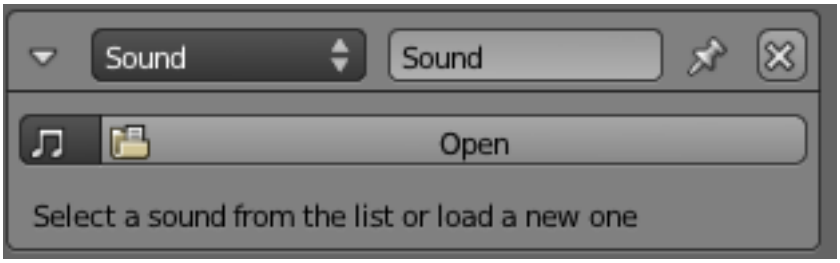


Fig. 2.2392: Sound Actuator.

See *Actuator Common Options* for common options.

Special Options:

Music File title Select music file from the list presented.

State Actuator

The State actuator allows the user to create complex logic, while retaining a clear user interface. It does this by having different states, and performing operations upon them.

See also:

Actuator Common Options for common options.

Special Options:

Fig. 2.2393: State actuator.

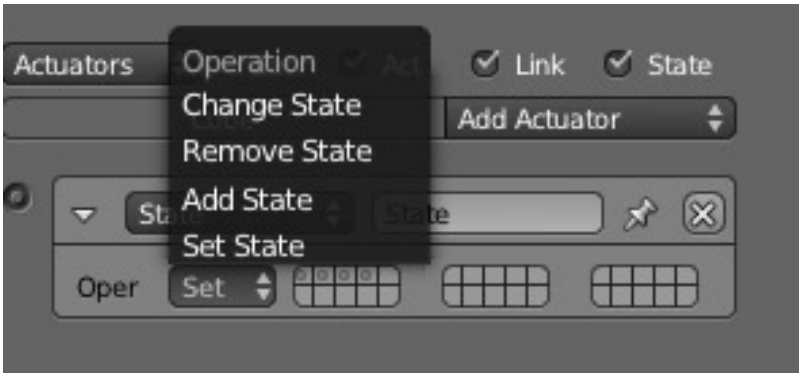


Fig. 2.2394: State actuator options.

Operation

Menu to select the state operation required.

Change State Change from the current state to the state specified.

Remove State Removes the specified states from the active states (deactivates them).

Add State Adds the specified states to the active states (activates them).

Set State Moves from the current state to the state specified, deactivating other added states.

Usage Notes

With the state actuator, you can create tiers of logic, without the need for hundreds of properties. Use it well, and you benefit greatly, but often problems may be circumvented by Python.

Visibility Actuator

The Visibility actuator allows the user to change the visibility of objects during runtime.

See *Actuator Common Options* for common options.

Special Options:

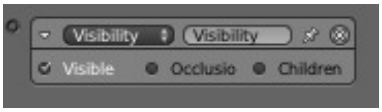


Fig. 2.2395: Visibility actuator.

Visible Toggle checkbox to toggle visibility

Occlusion Toggle checkbox to toggle occlusion. Must be initialized from the *Physics* tab.

Children Toggle checkbox to toggle recursive setting – will set visibility / occlusion state to all child objects, children of children (recursively)

Usage Notes

Using the visibility actuator will save on Rasterizer usage, however, not Physics, and so is limited in terms of Level of Detail (LOD). For LOD look at replace mesh, but be aware that the logic required can negate the effect of the LOD.

Properties

Properties are the game logic equivalent to variables. They are stored with the object, and can be used to represent things about them such as ammo, health, name, and so on.

Property Types

There are five types of properties:

Timer Starts at the property value and counts upwards as long as the object exists. It can for example be used if you want to know how long time it takes the player to complete a level.

Float Uses decimal numbers as values, can range from -10000.000 to 10000.000. It is useful for precision values.

Integer Uses integers (whole numbers) as values, between -10000 and 10000. Useful for counting things such as ammunition, where decimals are unnecessary.

String Takes text as value. Can store 128 characters.

Boolean Boolean variable, has two values: true or false. This is useful for things that have only two modes, like a light switch.

Using Properties

When a game is running, values of properties are set, manipulated, and evaluated using the *Property Sensor* and the *Property Actuator*.

Logic Properties are created and edited using the panel on the left of the Logic Editor Panel. The top menu provides a list of the available property types.

Add Game Property button This button adds a new property to the list, default is a *Float* property named `prop`, followed by a number if there already is one with this name.

Name field Where you give your property its name, this is how you are going to access it through Python or expressions. The way to do so in Python is by dictionary style lookup (`GameObject["propname"]`). The name is case sensitive.

Type menu This menu determines which type of property it is. The available options are in *Property Types*.

Value field Sets the initial value of the property.

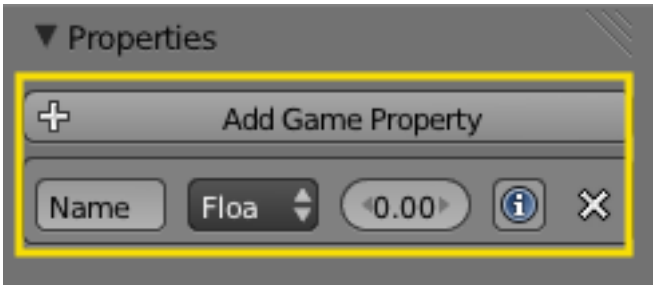


Fig. 2.2396: Properties Panel of the Logic Editor.

Information (*i* button) Display property value in debug information. If debugging is turned on, the value of the property is given in the top left-hand corner of the screen while the game is running. To turn debugging on, tick the *Show Debug Properties* checkbox in the *Game* menu. All properties with debugging activated will then be presented with their object name, property name and value during gameplay. This is useful if you suspect something with your properties is causing problems.

States

In the BGE, an object can have different "states". At any time while the game is playing, the current state of the object defines its behavior. For instance, a character in your game may have states representing awake, sleeping or dead. At any moment their behavior in response to a loud bang will be dependent on their current state; they may crouch down (awake); wake up (asleep) or do nothing (dead).

How States Operate

States are set up and used through controllers: note that only controllers, not actuators and sensors, are directly controlled by the state system. Each object has a number of states (up to 30; default = 1), and can only be in one state at any particular time. A controller must always specify the state for which it will operate – it will only give an output pulse if a) its logic conditions are met, and b) the object is currently in the specified State. States are set up and edited in the object's Controller settings (for details see below).

Tip: State settings are automatic in simple games. By default, the number of states for each object is 1, and all controllers are set to use State 1. So, if a game does not need multiple states, everything will work without explicitly setting states – you do not need to bother about states at all.

One of the actuators, the State actuator, can set or unset the object's State bits, and so allow the object's reaction to a sensor signal to depend on its current state. So, in the above example, the actor will have a number of controllers connected to the "loud bang" sensor, for each of the "awake", "asleep" or "dead" states. These will operate different actuators depending on the current state of the actor, and some of these actuators may switch the actor's state under appropriate conditions.

Editing States

States are set up and edited using the Controller (center) column of the Game Logic Panel. To see the State panel, click on the State Panel Button shown. The panel shows two areas for each of the 30 available states; these show Visible states, and Initial states (see below). Setting up the State system for a game is performed by choosing the appropriate state for each controller in the object's logic.

The display of an object's state logic, and other housekeeping, is carried out using the State Panel for the object, which is switched on and off using the button shown. The panel is divided into two halves, Visible and Initial.

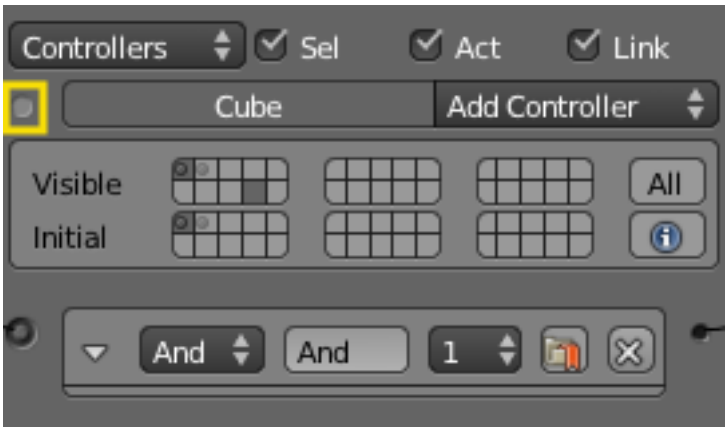


Fig. 2.2397: State Panel Button.

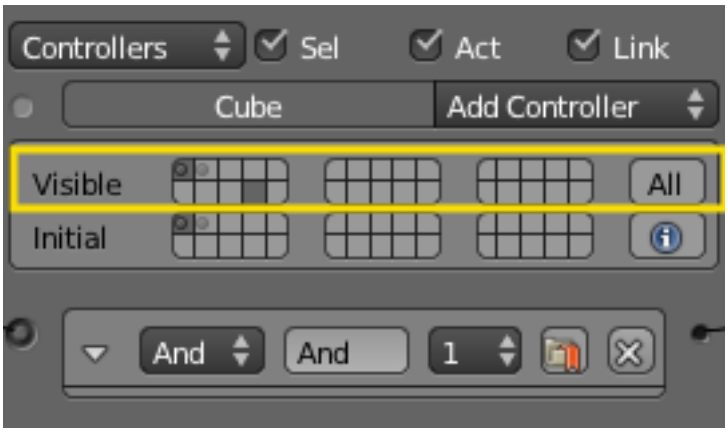


Fig. 2.2398: State Panel Visible.

Visible States

In the Visible area, each of the 30 available states is represented by a light-gray square. This panel shows what logic is visible for the logic brick displayed for the object. At the right is the All button; if clicked, then all the object's logic bricks are displayed (this is a toggle), and all State Panel squares are light-gray. Otherwise, individual states can be clicked to make their logic visible. (Note that you can click more than one square). Clicking the square again unselects the state.

States for the object that are in use (i.e. the object has controllers which operate in that state) have dots in them, and squares are dark-gray if these controllers are shown in the Game Logic display. The display of their connected sensors and actuators can also be controlled if the State buttons at the head of their columns are ticked.

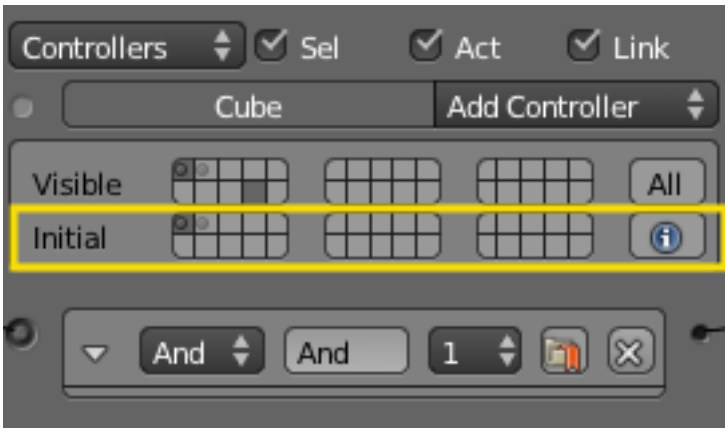


Fig. 2.2399: State Panel Initial.

Initial State

In the Initial area, each of the 30 available states is again represented by a light-gray square. One of these states may be clicked as the state in which the object starts when the game is run.

At the right is the button; if clicked, and the *Game* → *Show Debug Properties* is clicked, the current state of the object is shown in the top left-hand corner of the display while the game is running.

2.11.5 Camera

Introduction

The Game Engine camera is in many ways similar to the Camera in the normal Blender Render system, and is created, parameterized and manipulated in similar ways. However, because of its use as a real-time device, the Game Engine camera has a number of additional features – it may be used as not only as a static camera, but also as a moving device with its default characteristics (ie. with its own programmed moves), or it may track another object in the game. Furthermore, any game object may be used as a camera; the view is taken from the object's origin point. Lastly, it may be given special capabilities such as Stereo vision, Dome visualization etc. which have special relevance to game technology.

When you start the Game Engine, the initial camera view is taken from the latest 3D View. This may be either a selected camera object or the default camera (see below). Thus to start the game with a particular camera, you must select the camera and press `Numpad0` before starting the Game Engine.

Tip: To avoid camera distortion

Always zoom the view in until the camera object fills the entire viewport.

Default Camera

The default camera view is taken from the latest 3D View view, at a distance equivalent to the viewer. This means that if the normal 3D View is active the scene does not change when the Game Engine is started.

Camera Object

The Camera object in the Game Engine follows much the same structure as the conventional Blender camera – see *Camera* for details of how to set up, manipulate and select a camera. The following sections show some of the special facilities available in BGE cameras.

Parent Camera to Object

The camera will follow the object. First select the camera and then select the object. Next `Ctrl-P` → *Make Parent*.

Note that if your object has any rotations then the camera will also have those rotations. To avoid this use *Parent to Vertex*.

Parent to Vertex

The easiest way to accomplish this is to select your object and `Tab` to *Edit Mode*. Now select the vertex and `Tab` back to *Object Mode*.

Next, without any objects selected, select the camera and, holding `Shift`, select the object. `Tab` into *Edit Mode*, and `Ctrl-P` and choose *Make vertex parent*.

Now the camera will follow the object and it will maintain its rotation, while the object rotates.

Camera Editing

The camera (or cameras) used in a Blender game have a wide-ranging effect on the way in which the game is rendered and displayed. Mostly this is controlled using the Properties panel of the camera(s) used in the game.

Tip: Render Engine

Make sure that the render engine is set to Blender Game when attempting to set these controls, otherwise this description will not tally with what you see!

In the Camera Properties area, there are six panels available, as shown. Each can be expanded or contracted using the usual triangle button. The features in each panel will be described in detail below.

Embedded Player

The *Start* button starts the Game Engine. Shortcut `P`.

Standalone Player

This panel provides information for the Standalone Game Player which allows games to be run without Blender. See *Standalone Player* for further details.

Fullscreen Off – opens standalone game as a new window. On – opens standalone game in full screen.

Resolution

X Sets the X size of the viewport for full-screen display.

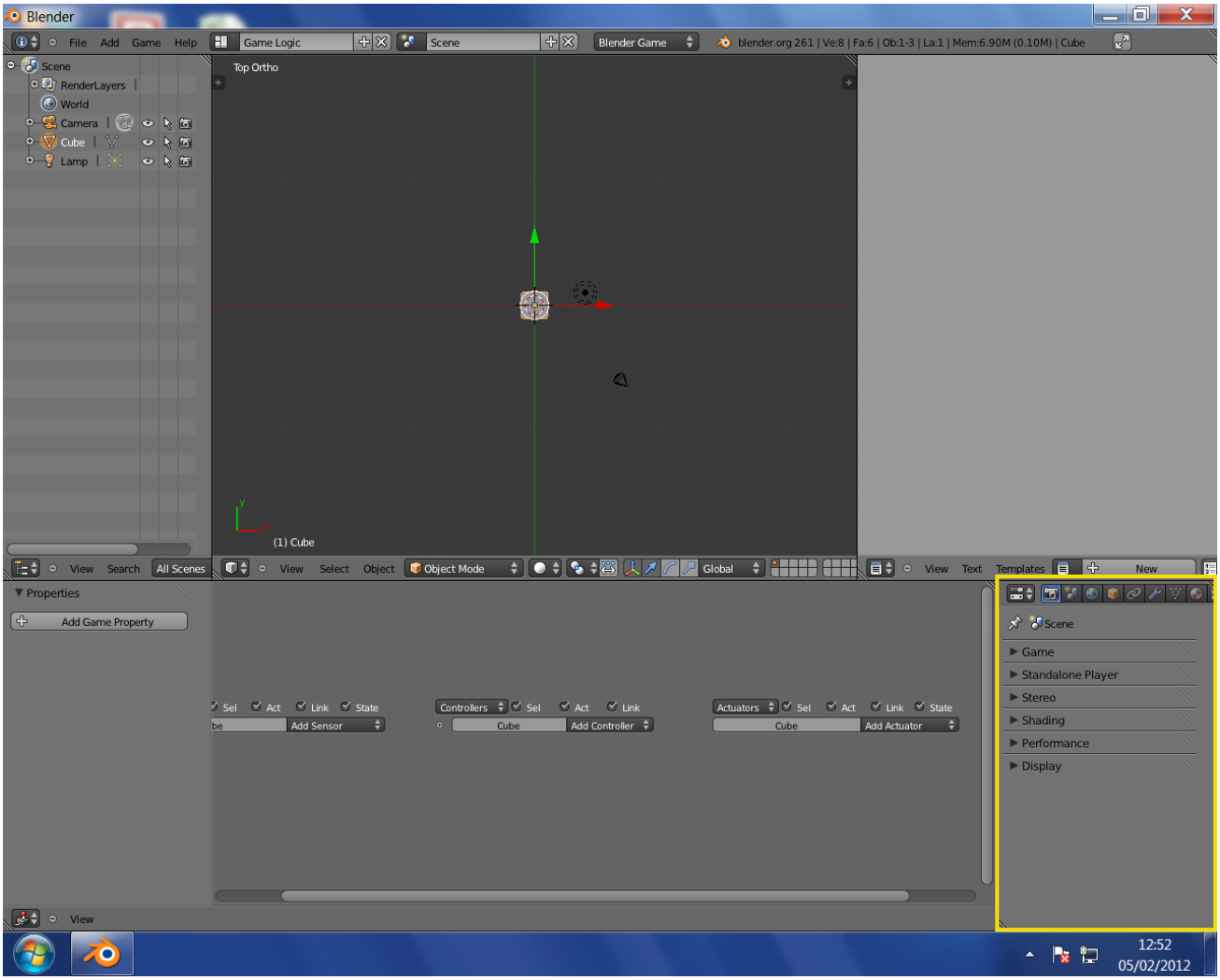


Fig. 2.2400: Camera Properties.

Fig. 2.2401: Game Panel.

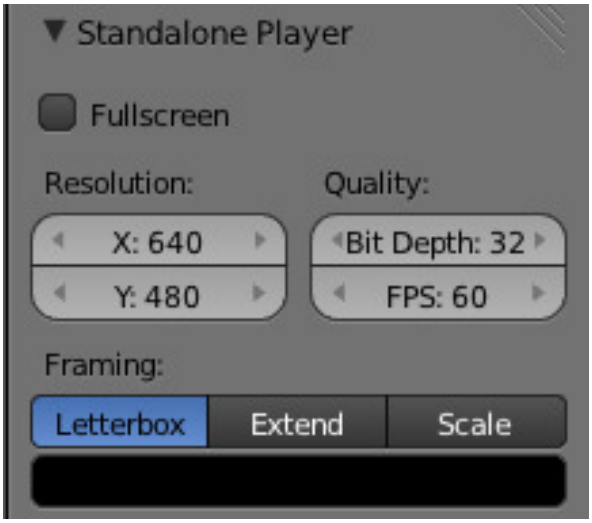


Fig. 2.2402: Standalone Panel.

Y Sets the Y size of the viewport for full-screen display.

Quality

Bit Depth Number of bits used to represent color of each pixel in full-screen display.

FPS Number of frames per second of full-screen display.

Framing Shows how the display is to be fitted in to the viewport.

Letterbox Show the entire viewport in the display window, and fill the remainder with the "bar" color.

Extend Show the whole display in the viewport, and fill the remainder with bars.

Scale Scale the display in X and Y to exactly fill the entire viewport.

Bar Color Select a color to use as the color of bars around the viewport.

Stereo

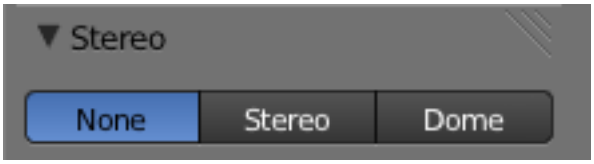


Fig. 2.2403: Stereo Panel.

Select a stereo mode that will be used to capture stereo images of the game (and also, by implication, that stereo displays will use to render images in the standalone player).

None Render single images with no stereo.

Stereo Render dual images for stereo viewing using appropriate equipment. See *Stereo Camera* for full details of available options.

Dome Provides facilities for an immersive dome environment in which to view the game. See *Dome Camera* for full details of available options.

Fig. 2.2404: Shading Panel.

Shading

Specifies the shading mode to be used in rendering the game. The shading facilities available in Blender for use in *Materials* and *Textures* are essentially the same in the Blender Game Engine. However, the constraints of real-time display mean that only some of the facilities are available.

Single Texture Use single texture facilities.

Multitexture Use Multitexture shading.

GLSL Use GLSL shading. GLSL should be used whenever possible for real-time image rendering.

Performance

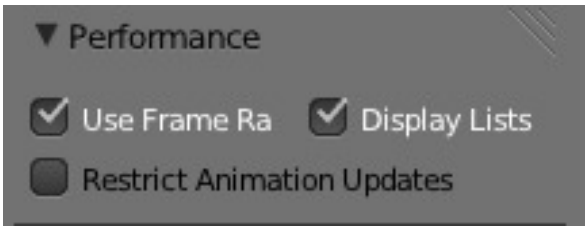


Fig. 2.2405: Performance Panel.

Use Frame Rate Respect the frame rate rather than rendering as many frames as possible.

Display Lists Use display lists to speed up rendering by keeping geometry on the GPU.

Restrict Animation Updates Restrict number of animation updates to the animation FPS (this is better for performance but can cause issues with smooth playback).

Display

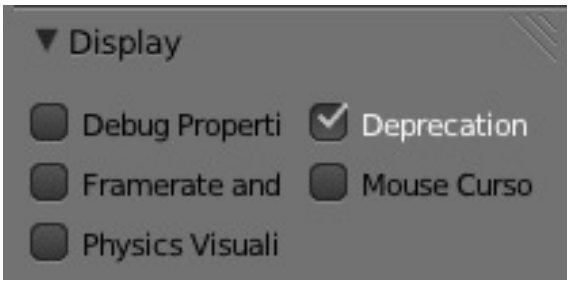


Fig. 2.2406: Display Panel.

Gives various display options when running the Game Engine. Under the...

Debug Properties Show properties marked for debugging while game runs. Note that debug properties to be shown must be requested at source (eg. i-button in state tables). Only available when game is run within Blender – not in standalone player version.

Framerate and Profile Show framerate and profiling information while game runs. Only available when game is run within Blender – not in standalone player version.

Physics Visualization Show physics bounds and interactions while game runs (available in both Blender and standalone versions).

Deprecation Warnings Print warnings when using deprecated features in the Python API. Only available when game is run within Blender – not in standalone player version.

Mouse Cursor Show mouse cursor while game runs (available in both Blender and standalone versions).

Stereo Camera

Stereo Cameras allow you to generate images that appear three dimensional when wearing special glasses. This is achieved by rendering two separate images from cameras that are a small distance apart from each other, simulating how our own eyes see. When viewing a stereo image, one eye is limited to seeing one of the images, and the other eye sees the second image. Our brain is able to merge these together, making it appear that we are looking at a 3D object rather than a flat image. See Stereoscopy for more information on different stereoscopic viewing methods.

Stereo Settings

Stereo Mode

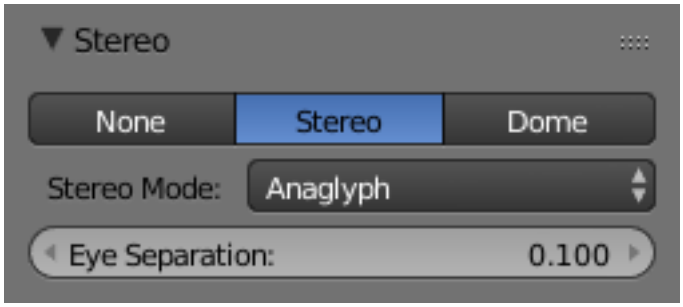


Fig. 2.2407: Set the type of stereo camera to use. Possible modes are detailed below.

Eye Separation This value is extremely important. It determines how far apart the two image-capturing cameras are, and thus how "deep" the scene appears. Too small a value and the image appears flat; too high a value can result in headaches and eye strain. The ideal value mimics the separation of the viewer's two eyes.

Stereo Modes

Specifies the way in which the left-eye image and the right-eye image pixels are put together during rendering. This must be selected according to the type of apparatus available to display the appropriate images to the viewer's eyes.

Anaglyph One frame is displayed with both images color encoded with red-blue filters. This mode only requires glasses with color filters, there are no special requirements for the display screen and GPU.

Quad Buffer Uses double buffering with a buffer for each eye, totaling four buffers (Left Front, Left Back, Right Front and Right Back), allowing to swap the buffers for both eyes in sync. See Quad Buffering for more information.

Side by Side Lines are displayed one after the other, so providing the two images in two frames side by side.

Above-Below Frames are displayed one after the other, so providing the two images in two frames, one above the other.

Interlaced One frame is displayed with the two images on alternate lines of the display.

Vinterlaced One frame is displayed with both images displayed on alternate columns of the display. This works with some 'autostereo displays'.

3D Tv Top-Bottom One frame displays the left image above and the right image below. The images are squashed vertically to fit. This mode is designed for passive 3D TV.

Dome Camera

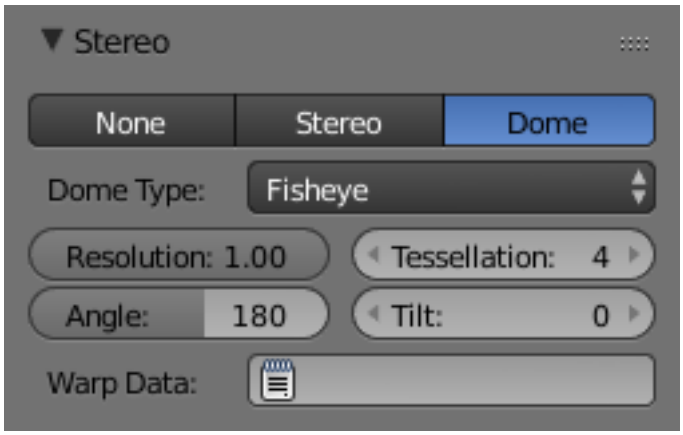
This feature allows artists to visualize their interactive projects within an immersive dome environment. In order to make it an extensible tool, we are supporting Fulldome, Truncated domes (front and rear), Planetariums and domes with spherical mirrors.

Tip: The Dome camera uses a multipass texture algorithm as developed by Paul Bourke and was implemented by Dalai Felinto with sponsorship from SAT – Society for Arts and Technology within the SAT Metalab immersion research program, that involves rendering the scene four times and placing the subsequent images onto a mesh designed especially such that the result, when viewed with an orthographic camera, is a fisheye projection.

Note: Remember to use Blender in 'fullscreen mode' to get the maximum out of your projector.

To accomplish that launch Blender with the command-line argument `-W`. Also to get away of the top menu on Blender try to join all areas (buttons, 3D View, text,...) in a single one. Otherwise if you only maximize it `Ctrl-Up`) you cannot get the whole screen free to run your game (the top bar menu takes about 20 pixels).

Dome Camera Settings



Dome Type This menu allows you to select which type of dome camera to use. They are outlined below, along with their respective settings.

- *Fisheye Mode*
- *Front-Truncated Dome Mode*
- *Rear-Truncated Dome Mode*
- *Cube Map Mode*
- *Spherical Panoramic Mode*

Available camera settings change depending on the selected Dome Type:

Resolution Sets the resolution of the Buffer. Decreasing this value increases speed, but decreases quality.

Tessellation 4 is the default. This is the tessellation level of the mesh. (Not available in Cube Map mode).

Angle Sets the field of view of the dome in degrees, from 90 to 250. (Available in Fisheye and Truncated modes).

Tilt Set the camera rotation in the horizontal axis. Available in Fisheye and Truncated modes).

Warp Data Mesh Use a custom warp mesh data file.

Fisheye Mode

An Orthogonal Fisheye view from 90º to 250º degrees.

- From 90º to 180º we are using four renders.
- From 181º to 250º we are using five renders.

Fig. 2.2408: Fisheye Mode.

Front-Truncated Dome Mode

Designed for truncated domes, this mode aligns the fisheye image with the top of the window, while touching the sides.

- The Field of view goes from 90º to 250º degrees.
- From 90º to 180º we are using four renders.
- From 181º to 250º we are using five renders.

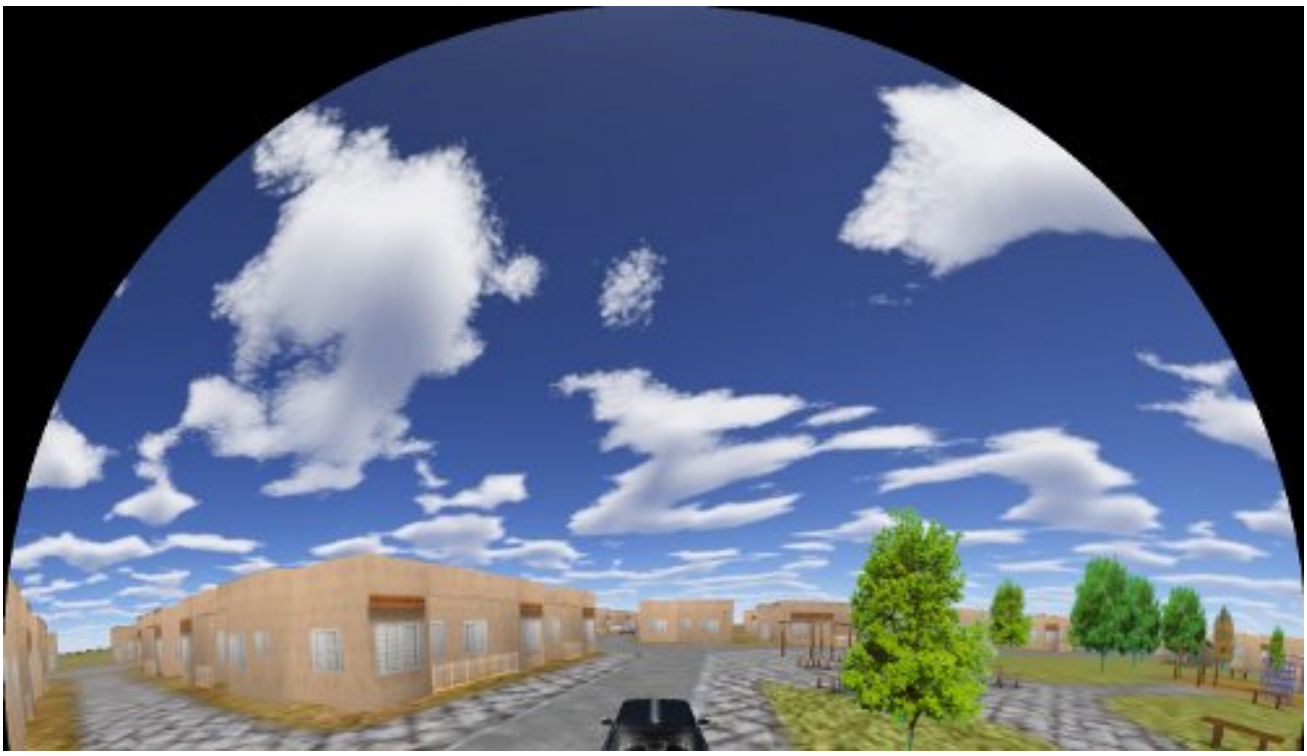


Fig. 2.2409: Front Truncated Dome Mode.

Rear-Truncated Dome Mode

Designed for truncated domes, this mode aligns the fisheye image with the bottom of the window, while touching the sides.

- The Field of view goes from 90º to 250º degrees.
- From 90º to 180º we are using four renders.
- From 181º to 250º we are using five renders.

Cube Map Mode

Cube Map mode can be used for pre-generate animated images for CubeMaps.

- We are using six renders for that. The order of the images follows Blender internal EnvMap file format: - first line: right, back, left - second line: bottom, top, front

Spherical Panoramic Mode

A full spherical panoramic mode.

- We are using six cameras here.
- The bottom and top start to get precision with *Definition* set to 5 or more.

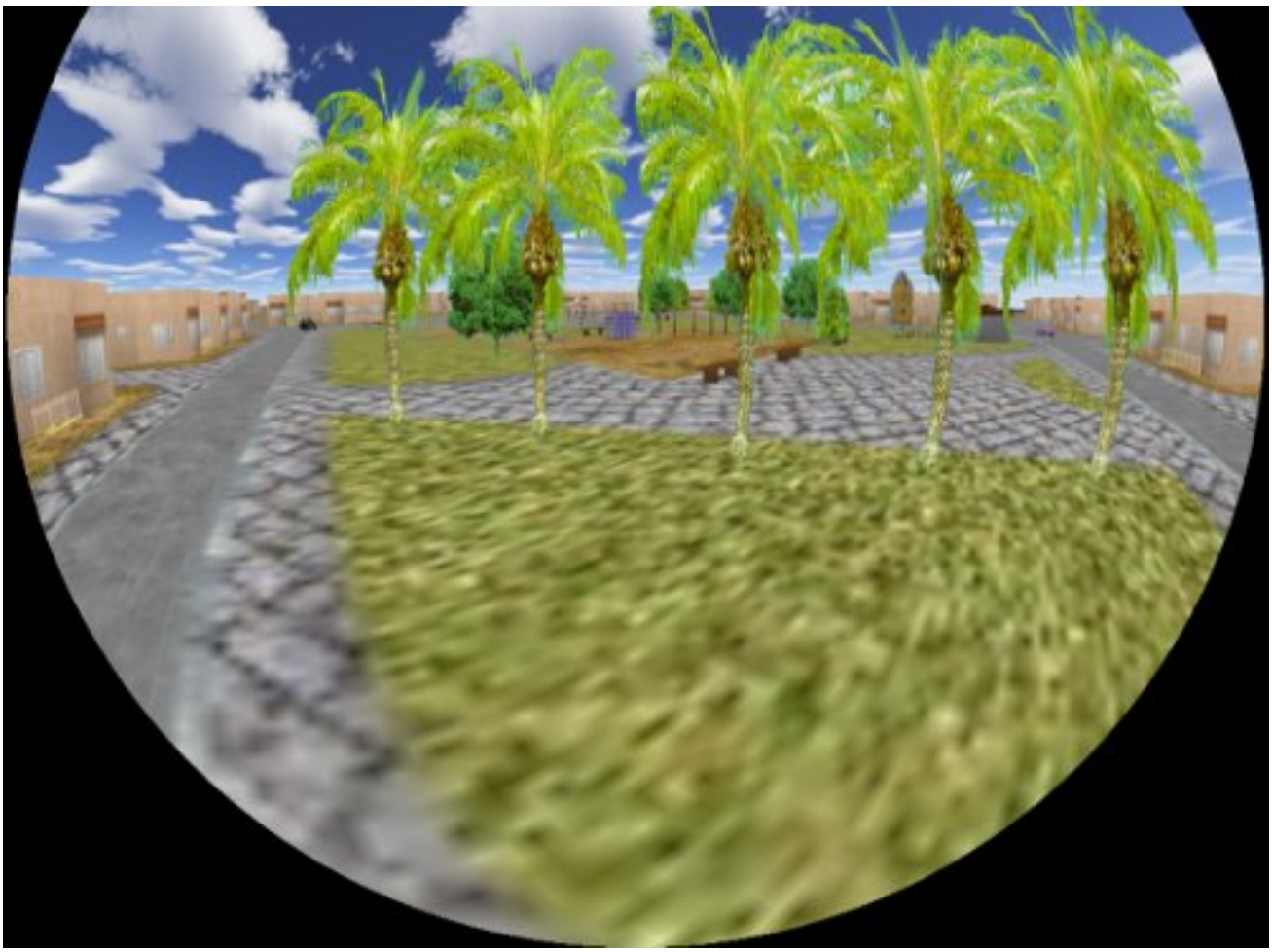


Fig. 2.2410: Rear Truncated Dome Mode.

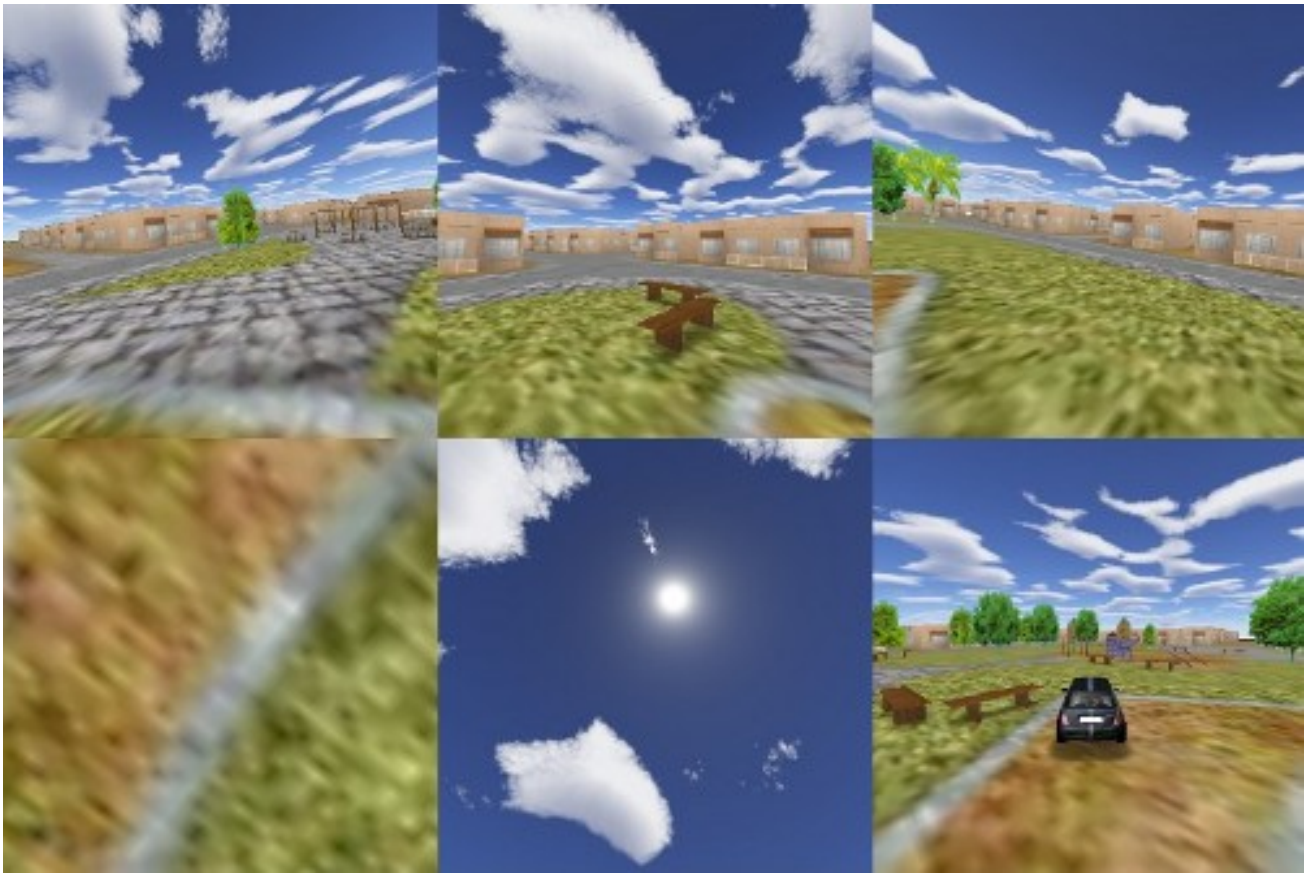


Fig. 2.2411: Environment Map Mode.

Fig. 2.2412: Full Spherical Panoramic Mode.

Warp Data Mesh

Many projection environments require images that are not simple perspective projections that are the norm for flat screen displays. Examples include geometry correction for cylindrical displays and some new methods of projecting into planetarium domes or upright domes intended for VR.

For more information on the mesh format see Paul Bourke's article.

In order to produce that images, we are using a specific file format.

File template:

```
mode
width height
n0_x n0_y n0_u n0_v n0_i
n1_x n1_y n1_u n1_v n1_i
n2_x n1_y n2_u n2_v n2_i
n3_x n3_y n3_u n3_v n3_i
(...)
```

First line is the image type the mesh is support to be applied to: `2 = rectangular`, `1 = radial` Next line has the mesh dimensions in pixelsRest of the lines are the nodes of the mesh.

Each line is compound of *x*, *y*, *u*, *v*, *i* (x, y) are the normalized screen coordinates (u, v) texture coordinates *i* a multiplicative intensity factor.

x varies from -screen aspect to screen aspect varies from -1 to 1 *u* and *v* vary from 0 to 1. *i* ranges from 0 to 1, if negative do not draw that mesh node.

1. You need to create the file and add it to the Text Editor in order to select it as your Warp Mesh data file.
2. Open the Text Editor *Editor Types* → *Text Editor*.
3. Open your mesh data file (ie. myDome.data) in the text editor (*Text* → *Open* or `Alt-O`).
4. Go to Game Framing Settings *Editor Types* → *Properties editor* → *Scene*.
5. Enable Dome Mode.
6. Type filename in Warp Data field (i.e. myDome.data).

To create your own Warp Meshes an interactive tool called meshmapper is available as part of Paul Bourke's Warpplayer software package (requires full version).

Examples

Spherical Mirror Dome 4×3, Truncated Dome 4×3, Sample Fullscreen File 4×3, Sample Fullbuffer File 4×3.

Important: The viewport is calculated using the ratio of canvas width by canvas height. Therefore different screen sizes will require different warp mesh files. Also in order to get the correct ratio of your projector you need to use Blender in Fullscreen mode.

2.11.6 Physics

Introduction

Blender includes advanced physics simulation in the form of the Bullet Physics Engine (Bullet Physics). Most of your work will involve setting the right properties on the objects in your scene, then you can sit back and let the engine take over. The physics simulation can be used for Games, but also for Animation.

The Blender Game Engine (BGE) is based on Rigid-Body Physics, which differs significantly from the complementary set of tools available in the form of Soft Body Physics Simulations. Though the BGE does have a Soft Body type, it is not nearly as nuanced as the non-BGE Soft Body. The inverse is even more true: it is difficult to get the non-BGE physics to resemble anything like a stiff shape. Rigid Body Physics does not have, as an effect or a cause, any mesh deformations. For a discussion on how to partially overcome this, see: *Mesh Deformations*.

Global Options

The global Physics Engine settings can be found in the *World Properties*, which include the Gravity constant and some important engine performance tweaks.

Object Physics

Physics Type

No Collision Is not affected by the simulation nor affects other objects.

Static Participates in the simulation, affecting other objects, but is not affected by it.

Dynamic Object that can move besides colliding and being collided with.

Rigid Body Has rigid body dynamics.

Soft Body Soft body dynamics.

Character Controller Character controller.

Vehicle Controller Vehicle controller.

Occluder Prevents calculation of rendered objects (not their physics, though!).

Sensor Detects presence without restituting collisions.

Navigation Mesh To make pathfinding paths. Useful for Artificial Intelligence.

Material Physics

Physics can be associated with a material on the material properties tab. These are settings that one would normally associate with a material, such has its friction and they are meant to be used in conjunction with the object physics settings, not replace it.

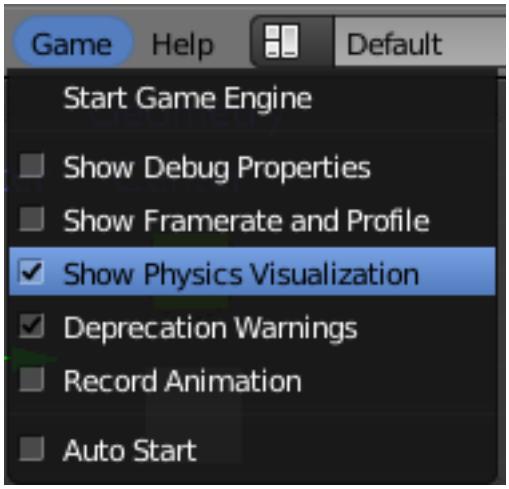
Constraints

It is imperative to understand that the Blender Constraints generally do not work inside the BGE. This means interesting effects such as *Copy Rotation* are unavailable directly.

Your options include:

- *Parenting* - But not Vertex Parenting.
- *Rigid Body Joint* – This is the one Constraint that you can set up through the UI that works in the BGE. It has several options, and can be very powerful – see ITS page for a detailed description and demo blend-file. Do not forget that you can loop through objects using `bpy` instead of clicking thousands of times to set up chains of these Constraints.
- Rigid Body Joints on the Fly – You can add/remove them after the BGE starts by using `bge.constraints.createConstraint()`. This can be good either to simply automate their setup, or to truly make them dynamic. A simple demo can be viewed in: BGE-Physics-DynamicallyCreateConstraint.blend
- Python Controllers – As always, in the BGE, you can get the most power when you drop into Python and start toying with the settings directly. For instance, the *Copy Rotation* mentioned above is not hard – All you have to do is something to the effect of `own.worldOrientation = bge.logic.getCurrentScene().objects['TheTargetObject'].worldOrientation`

Visualizing Physics



Go to *Game* → *Show Physics Visualization* to show lines representing various attributes of the Bullet representation of your objects. Note that these might be easier to see when you turn on Wireframe Mode `Z` before you press `P`. Also note that you can see how the Bullet triangulation is working (it busts all your Quads to Tris at run-time, but the BGE meshes are still quads at run-time).

- *RGB/XYZ Widget* - Representing the object's Local Orientation and Origin.
- *Green* - "sleeping meshes" that are not moving, saving calculations until an external event "wakes" it.
- *White* - White lines represent active bounding meshes at are undergoing physics calculations, until such calculations are so small that the object is put to rest. This is how you can see the effects of the *Collision Bounds*. - *Thick*, or *Many White Lines* - A compound collision mesh/meshes.
- *Violet* - Bounding meshes for Soft bodies.
- *Red* - The Bounding Box, the outer boundary of object. It is always aligned with global X Y and Z, and is used to optimize calculations. Also represents meshes that have been forced into "no sleep" status.
- *Yellow* - Normals.
- *Black* - When in wireframe, this is your mesh's visual appearance.

If you want finer-grained control over the display options, you can add this as a Python Controller and uncomment whichever pieces you want to see:

```
import bge
debugs = (
    bge.constraints.DBG_DRAWAABB,
    )
for d in debugs:
   bge.constraints.setDebugMode(d)
```

For all debug modes, API docs for `bge.constraints`.

Show Framerate and Profile

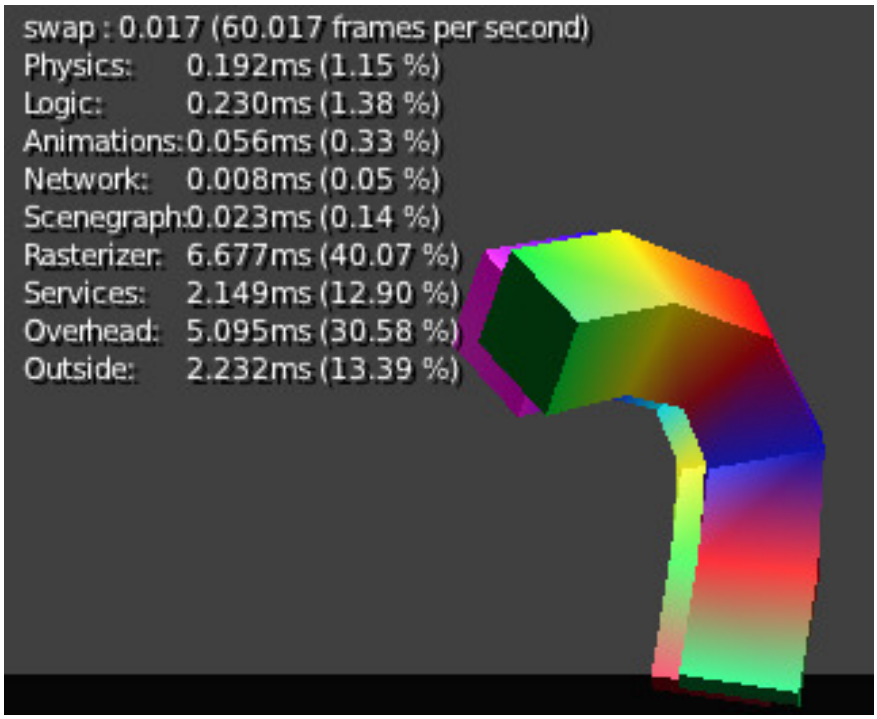


Fig. 2.2413: A shot of Manual-BGE-Physics-DancingSticks.blend with *Game* → *Show Framerate and Profile* enabled.

If you enable *Game → Show Framerate and Profile*, it will put some statistics in the upper-left area of the game window.

See also:

These can be very informative, but also a bit cryptic. Moguri has elaborated on their meanings, for us: Moguris blog.

Mesh Deformations

As mentioned above, Rigid Body physics do not affect mesh deformations, nor do they account for them in the physics model. This leaves you with a few options:

Soft Bodies

You can try using a *Soft Body*, but these are fairly hard to configure well.

Actions

To use an *Action Actuator* to do the deformation, you have to make a choice. If you use Shapekeys in the Action, you will be fine as far as the overall collisions (but see below for the note on `reinstancePhysicsMesh()`). The mesh itself is both a display and a physics mesh, so there is not much to configure.

To use an Armature as the deformer will require a bit of extra thought and effort. Basically the Armature will only deform a mesh if the Armature is the parent of that mesh. But at that point, your mesh will lose its physics responsivenes, and only hang in the air (it is copying the location/rotation of the Armature). To somewhat fix this you can then parent the Armature to a collision mesh (perhaps a simple box or otherwise very-low-poly mesh). This "Deformation Mesh" will be the physics representative, being type: Dynamic or Rigid Body, but it will be set to Invisible. Then "Display Mesh" will be the opposite set to *No Collision*, but visible. This still leaves the problem mentioned in the previous paragraph.

When you deform a display mesh, it does not update the corresponding physics mesh. You can view this evidently when you enable physics visualization (*Visualizing Physics*) – the collision bounds will remain exactly as when they began. To fix this, you must call `own.reinstancePhysicsMesh()` in some form. Currently this only works on *Triangle Mesh* bounds, not *Convex Hull*. We have prepared a demonstration file in Manual-BGE-Physics-DancingSticks.blend. Note that we had to increase the *World → Physics → Physics Steps → Substeps* to make the collisions work well. The more basic case is the case the Shapekeyed Action, which you can see in the back area of the scene. Since it is the only object involved, you can call `reinstancePhysicsMesh()` unadorned, and it will do the right thing.

The more complicated case is the *Collision Mesh → Armature → Display Mesh* cluster, which you can see in the front of the scene. What it does in the blend-file is call `reinstancePhysicsMesh(viz)`, that is, passing in a reference to the visual mesh. If we tried to establish this relationship without the use of Python, we would find that Blender's dependency check system would reject it as a cyclic setup. This is an example of where Blender's checking is too coarsely-grained, as this circle is perfectly valid: the grandparent object (the Collision Mesh) controls the location/rotation, while the middle object (the Armature) receives the animated Action, where the child (the Display Mesh) receives the deformation, and passes that on up to the top, harmlessly. Something to note is that the Collision Mesh is merely a plane – that is all it requires for this, since it will be getting the mesh data from `viz`.

Ragdolls

A third option is to create your items out of many sub-objects, connected together with Rigid Body Joints or similar. This can be quite a bit more work, but the results can be much more like a realistic response to collisions. For an Add-on that can help you out in the process, check out the Blender Ragdoll Implementation Kit.

Digging Deeper

Sometimes you will want to look at:

- The main Bullet Physics page

- The Bullet Wiki
- The Bullet API Docs
- The Bullet Forums

Recording to Keyframes

Beyond gaming, sometimes you wish to render a complex scene that involves collisions, multiple forces, friction between multiple bodies, and air drag or even a simple setup that is just easier to achieve using the realtime physics.

Blender provides a way to ''bake" or ''record" a physics simulation into keyframes allowing it then to be played as an action either for animation or games. Keep in mind that the result of this method is a recording, no longer a simulation. This means that the result is completely deterministic (the same everytime it is run) and unable to interact with new objects that are added to the physics simulation after it was recorded. This may, or not, be desired according to the situation.

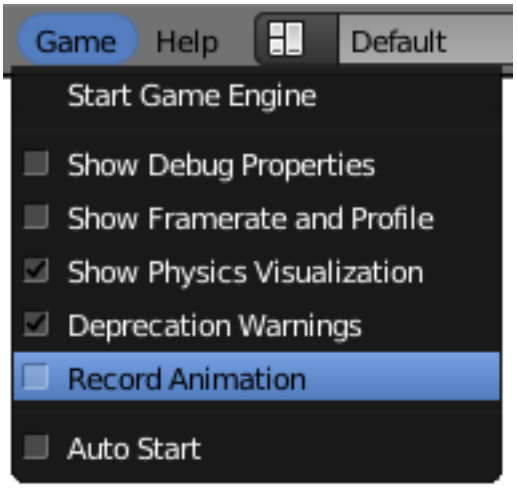


Fig. 2.2414: Menu to record Keyframes to the Dopesheet.

All you have to do to achieve this effect is go to the Info Editor (the bar at the top of the window) *Game* → *Record Animation*, and it will lock away your keyframes for use in *Blender Render* mode. You can go back to the 3D View and press `Alt-A` to play it back, or `Ctrl-F12` to render it out as an animation.

Note that you can also use Game Logic Bricks and scripting. Everything will be recorded.

Keyframe Clean-up

Record Animation keys redundant data (data that was did not change relative to the last frame). Pressing `O` while in the *DopeSheet* will remove all superfluous keyframes. Unwanted channels can also be removed.

Exporting

.bullet / Bullet compatible engines

You can snapshot the physics world at any time with the following code:

```
import bge
bge.constraints.exportBulletFile("test.bullet")
```

This will allow importing into other Bullet-based projects. See the Bullet Wiki on Serialization for more.

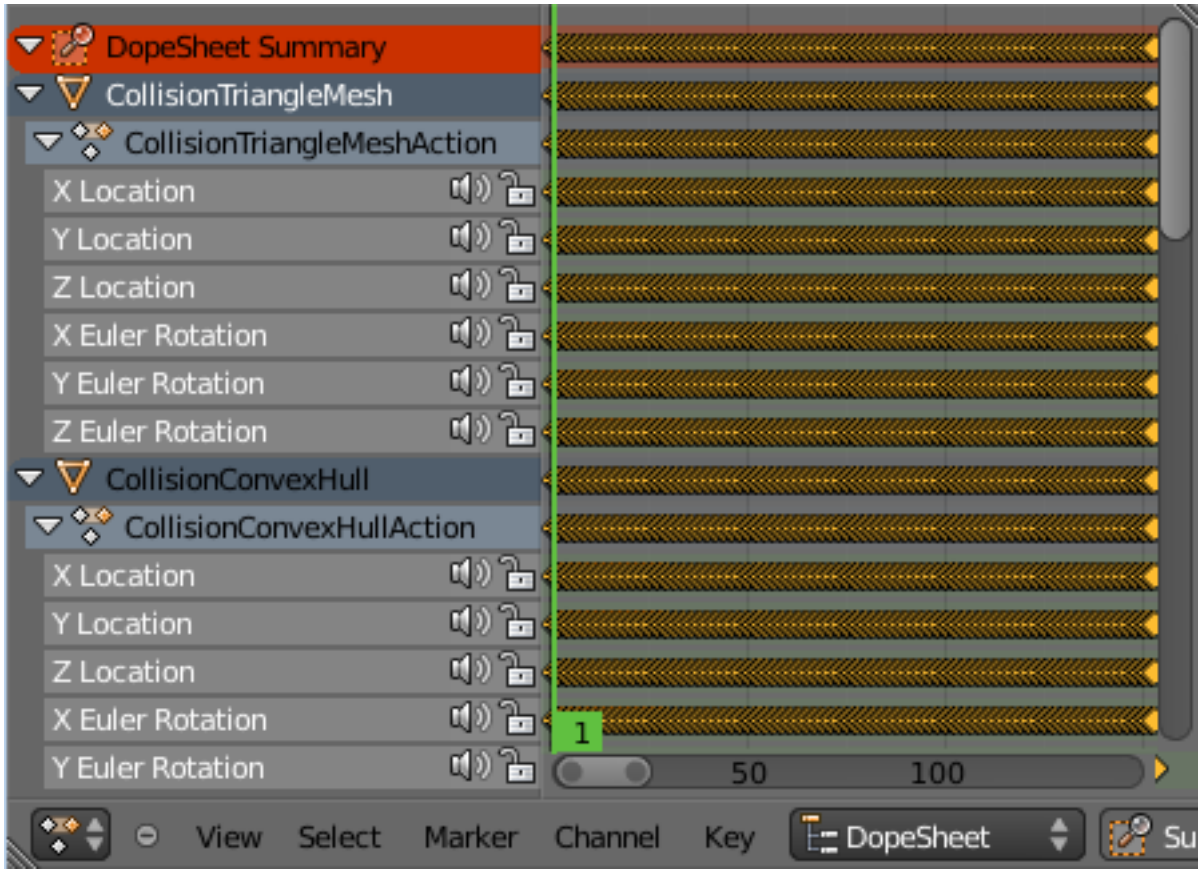


Fig. 2.2415: Resulting recorded animation.

World Physics

World settings enable you to set some basic effects which affect all scenes throughout your game, so giving it a feeling of unity and continuity. These include ambient light, depth effects (mist) and global physics settings. These effects are a limited subset of the more extensive range of effects available with the Blender internal or cycles renderer.

Tip: While world settings offer a simple way of adding effects to a scene, *compositing nodes* are often preferred, though more complex to master, for the additional control and options they offer. For example, filtering the Z value (distance from camera) or normals (direction of surfaces) through compositing nodes can further increase the depth and spacial clarity of a scene.

World

These two color settings allow you to set some general lighting effects for your game.

Horizon Color The RGB color at the horizon; i.e. the color and intensity of any areas in the scene which are not filled explicitly.

Ambient Color Ambient light mimics an overall background illumination obtained from diffusing surfaces (see *Ambient Light*, *Exposure* and *Ambient Occlusion*). Its general color and intensity are set by these controls.

Mist

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender makes objects farther away more transparent (decreasing their Alpha value) so that they mix more of the background color with the object color. With Mist

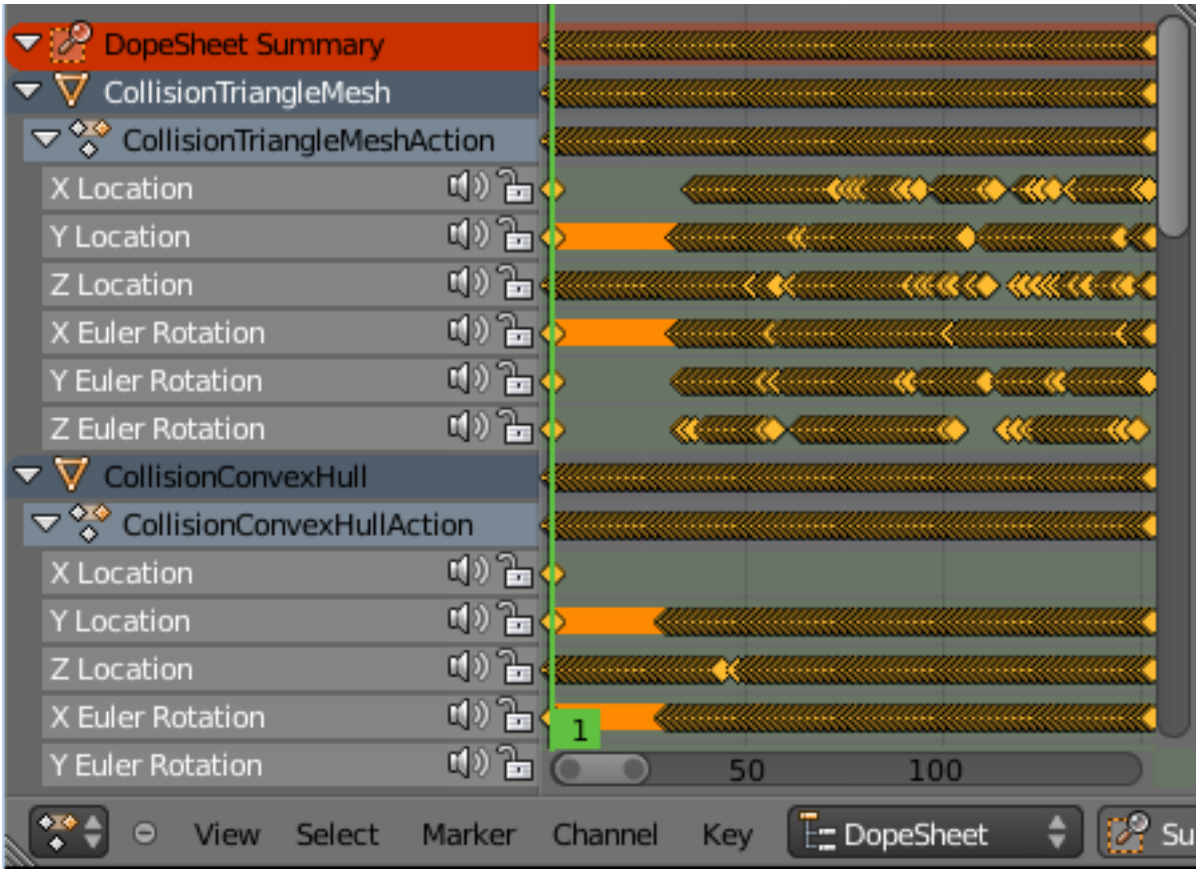


Fig. 2.2416: Cleaned up recording.

enabled, the further the object is away from the camera the less it is alpha value will be. For full details, see *Mist*.

Mist Toggles mist on and off.

Falloff Sets the shape of the falloff of the mist.

Start The starting distance of the mist effect. No misting will take place for objects closer than this distance.

Depth The depth at which the opacity of objects falls to zero.

Minimum intensity Overall minimum intensity of the mist.

Game Physics

The *Game Physics* located in the *World* panel determine the type of physical rules that govern the Game Engine scene, and the gravity value to be used. Based on the physics engine selected, in physics simulations in the Game Engine, Blender will automatically move *Actors* in the downward (-Z) direction. After you arrange the actors and they move as you wish, you can then bake this computed motion into keyframes (see *Digging Deeper* for more info).

Physics Engine Set the type of physics engine to use.

Bullet The default physics engine, in active development. It handles movement and collision detection. The things that collide transfer momentum to the collided object.

None No physics in use. Things are not affected by gravity and can fly about in a virtual space. Objects in motion stay in that motion.

Gravity The gravitational acceleration, $m.s^{-2}$ (in units of meters per squared second), of this world. Each object that is an actor has a mass and size slider. In conjunction with the frame rate (see *Render* section), Blender uses this info to calculate how fast the object should accelerate downward.

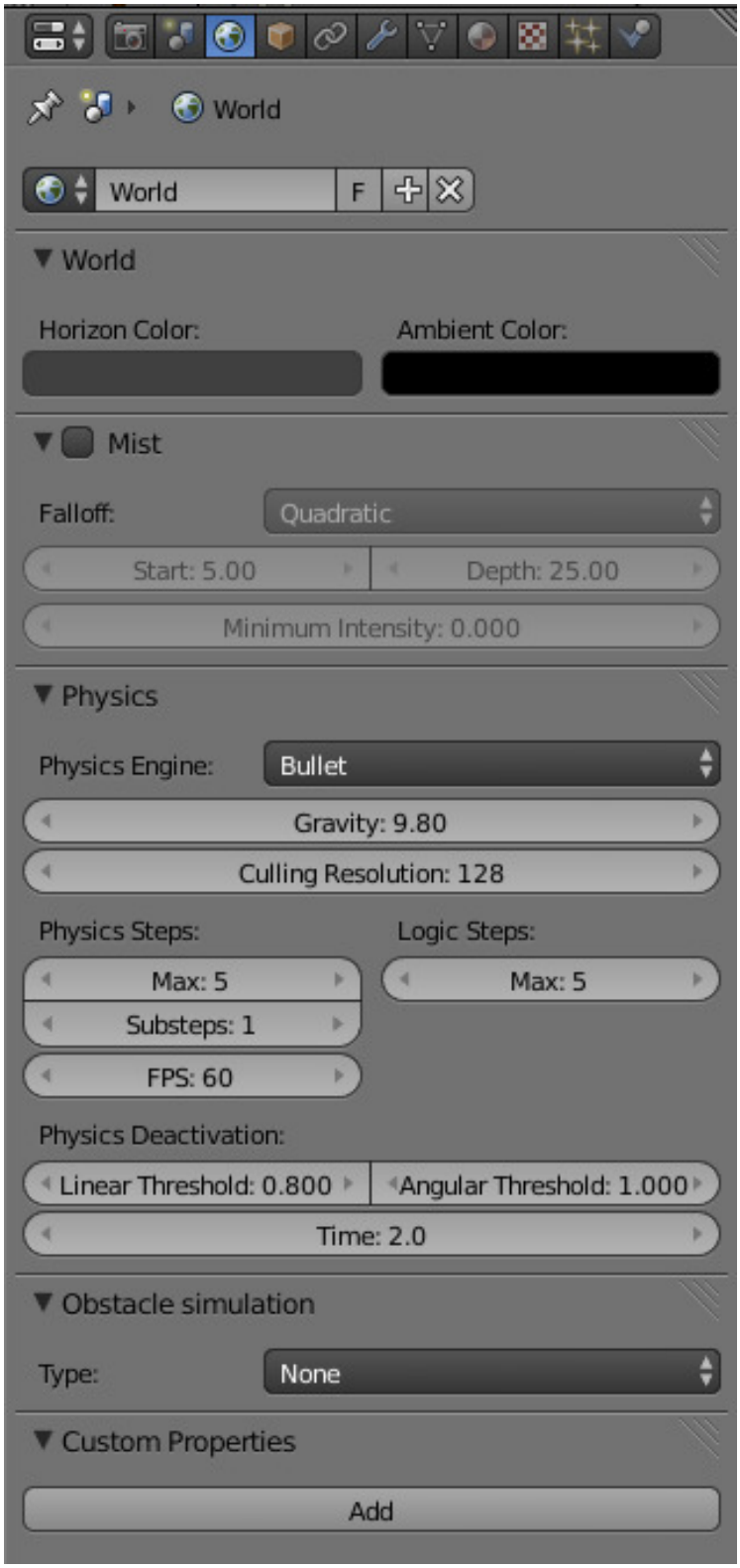


Fig. 2.2417: BGE World Panel (fully expanded).

Culling Resolution The size of the occlusion culling buffer in pixel, use higher value for better precision (slower). The optimized Bullet DBVT for view frustum and occlusion culling is activated internally by default.

Physics Steps

Max Sets the maximum number of physics steps per game frame if graphics slow down the game. higher value allows physics to keep up with realtime.

Substeps Sets the number of simulation substeps per physics timestep. Higher value give better physics precision.

FPS Set the nominal number of game frames per second. Physics fixed timestep = 1/fps, independently of actual frame rate.

Logic Steps Sets the maximum number of logic frame per game frame if graphics slows down the game, higher value allows better synchronization with physics.

Physics Deactivation These settings control the threshold at which physics is deactivated. These settings help reducing the processing spent on Physics simulation during the game.

Linear Threshold The speed limit under which a rigid bodies will go to sleep (stop moving) if it stays below the limits for a time equal or longer than the deactivation time (sleeping is disabled when deactivation time is set to 0).

Angular Threshold Same as linear threshold, but for rotation limit (in rad/s)

Time The amount of time in which the object must have motion below the thresholds for physics to be disabled (0.0 disables physics deactivation).

Obstacle Simulation

Simulation used for obstacle avoidance in the Game Engine, based on the RVO (Reciprocal Velocity Obstacles) principle. The aim is to prevent one or more actors colliding with obstacles.

See Path finding and steering behaviors for more details.

Type

None Obstacle simulation is disabled, actors are not able to avoid obstacles

RVO (cells) Obstacle simulation is based on the RVO method with cell sampling.

RVO (rays) Obstacle simulation is based on the RVO method with ray sampling.

Level height Max difference in heights of obstacles to enable their interaction. Used to define minimum margin between obstacles by height, when they are treated as those which are situated one above the other i.e. they does not influence to each other.

Visualization Enable debug visualization for obstacle simulation.

Converting

Sometimes, you may want to animate a wall being broken down by an object, or a bunch of objects collapsing, falling, or bouncing with accurate physics. You could manually insert keyframes and do trial and error adjusting with F-Curves to simulate physics and acceleration, or, you can do it much easier and automatically by taking advantage of Blender Game Engine Physics. Blender now has a feature which allows you to record animation in a Blender game and turn it into Blender Animation Keyframes.

Animation can be recorded by going *Game* → *Record Animation*. The animation can them to recorded with `Alt-A`

If you just want a static pile of stuff, you can move to the last frame, and delete all the keyframes quickly by turning them into NLAs and deleting.

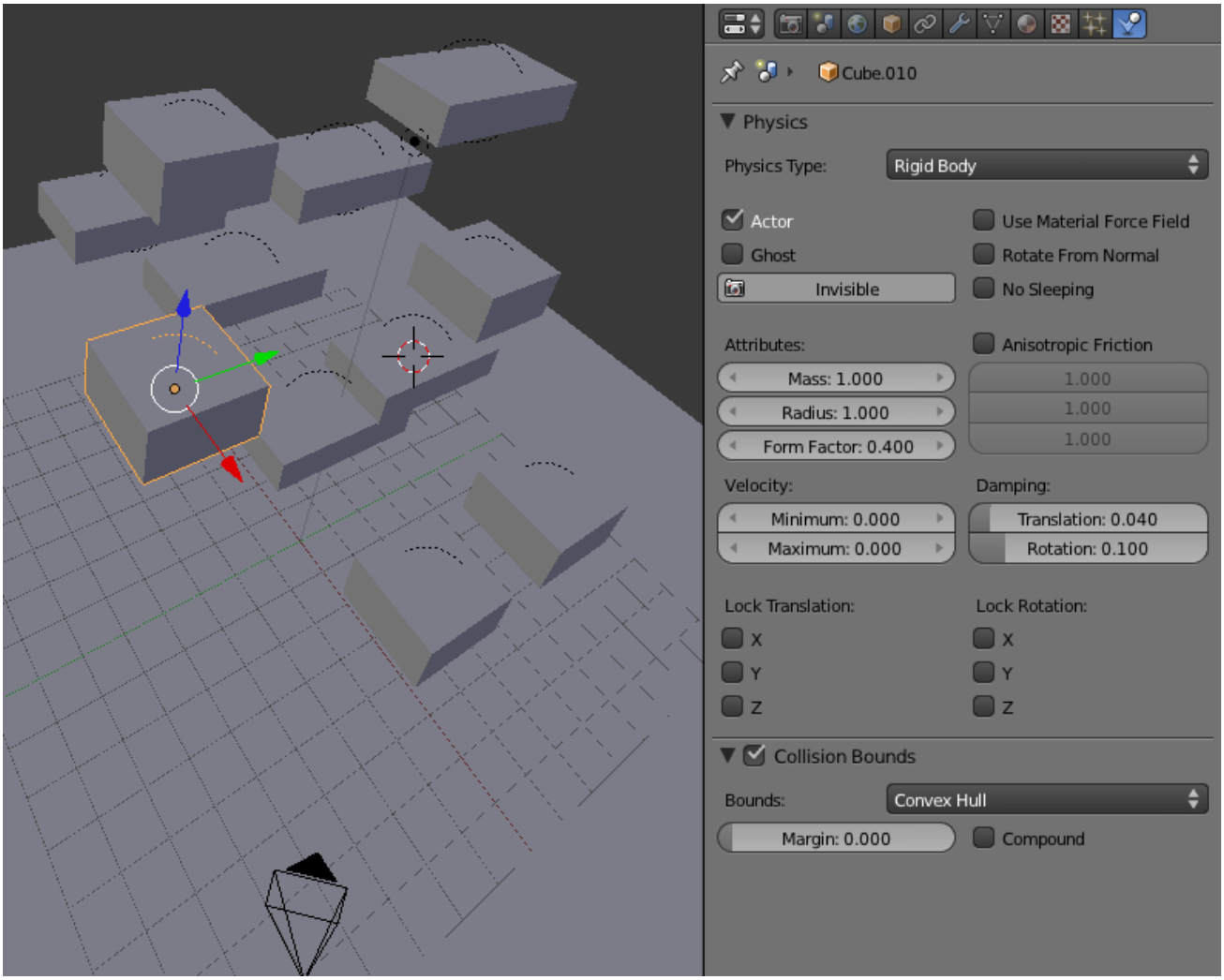


Fig. 2.2418: Some Blocks about to fall.

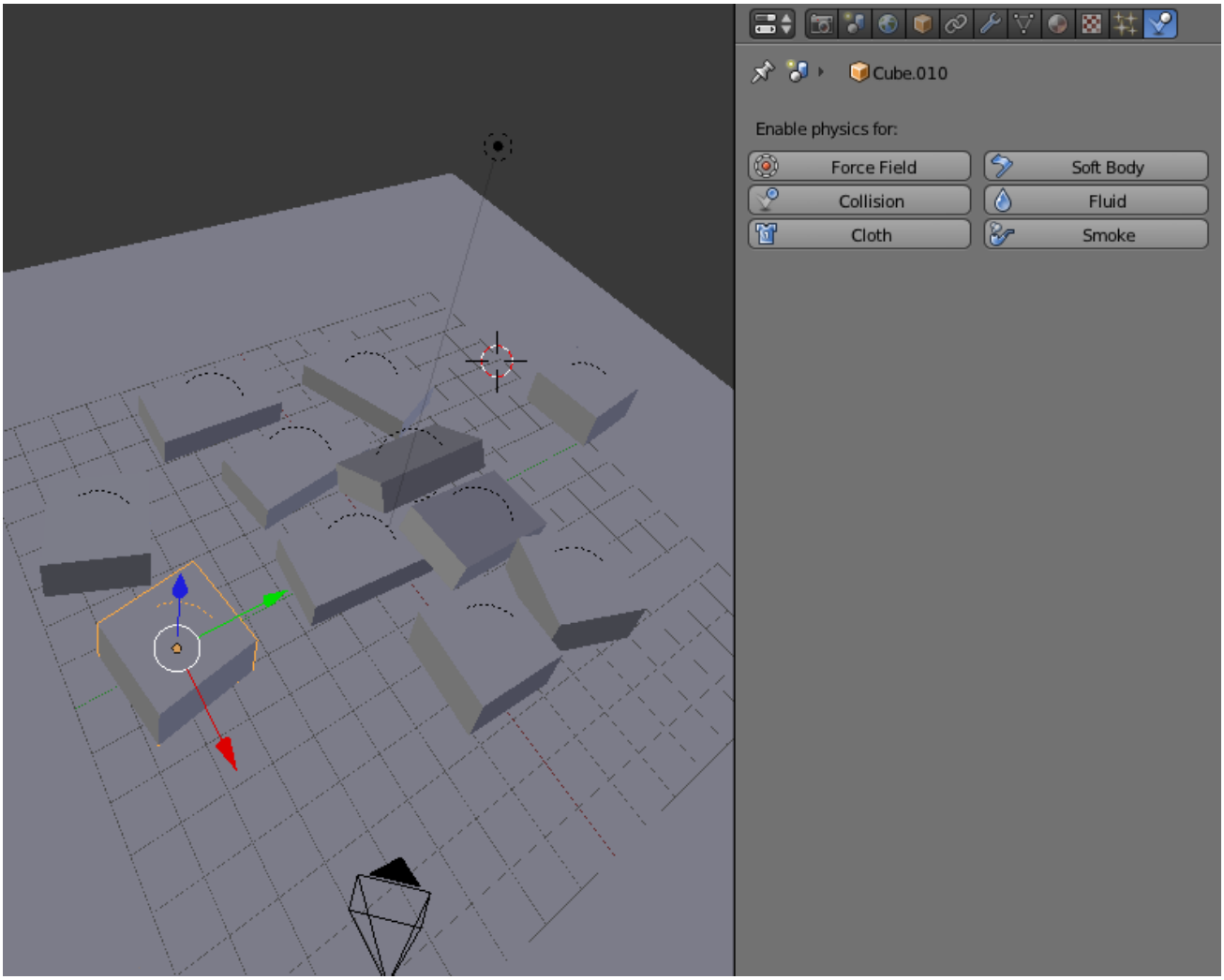


Fig. 2.2419: A Pile.

Physics Types

Static Physics

Static objects in the *Blender Game Engine* do not automatically react to physics, including gravity and collisions. Even if hit by the force of a speeding 18-wheeler truck, it will remain unresponsive in terms of location, rotation, or deformation.

It will, however, give collision reactions. Objects will bounce off of Static Objects, and rotational inertia will transfer to objects capable of rotating (that is, Rigid Body Objects will spin in response, though Dynamic Objects will not).

Note that none of this prevents you from transforming the Static Objects with *Logic Bricks* or Python code. The visual objects will correctly move and their physics representation will update in the engine as well.

Another important note is that the default *Collision Bounds* is a Triangle Mesh, meaning it is higher in computational requirements but also in detail. This in turn means the "Radius" option has no effect by default.

For more documentation, see the *Top BGE Physics page*.

Options

Note: bpy Access

Note that most of these properties are accessible through the non- BGE scripting API via `bpy.data.objects["ObjectName"].game`, which is of type `bpy.types.GameObjectSetting`. This is useful so you can, for example, set a range of objects to have gradated values via a for-loop.

Actor Enables detection by Near and Radar Sensors.

- Default: On.
- Python property: `obj.game.use_actor`

Ghost Disables collisions completely, similar to No Collision.

- Default: Off.
- Python property: `obj.game.use_ghost`

Invisible Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the Outliner).

- Default: Off.
- Python property: `obj.use_render`

Radius If you have the "Collision Bounds: Sphere" set explicitly (or implicitly through having the Collision Bounds subpanel unchecked), this will multiply with the Object's (unapplied) Scale. Note that none of the other bounds types are affected. Also note that in the 3D View the display will show this for all types, even though it is only actually used with Sphere. Python property: `obj.game.radius`

Basic	Radius= 1.5	Unapplied Scale	Applied Scale	Collision Bounds
Rolls, radius of 1 BU	Rolls, radius of 1.5 BU (after "popping" upward)	Rolls, radius of 1.5 BU	Rolls, radius of 1 BU (!)	Default (which is Sphere)
Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1 BU	Box
""	""	""	""	Convex Hull
Slides, extent of 1 BU (but with more friction than above)	Slides, extent of 1 BU (but with more friction than above)	Acts insane	Slides extent of 1.5 BU	Triangle Mesh

Anisotropic Friction Isotropic friction is identical at all angles. Anisotropic is directionally-dependant. Here you can vary the coefficients for the three axes individually, or disable friction entirely. Python properties: `obj.game.use_anisotropic_friction` (boolean) and `obj.game.friction_coefficients` (a 3-element array).

Collision Bounds

Note: The Static type differs from the others in that it defaults to a Triangle Mesh bounds, instead of a simple sphere.

The first thing you must understand is the idea of the 3D Bounding Box. If you run through all the vertices of a mesh and record the lowest and highest x values, you have found the *x min/max* the complete boundary for all x values within the mesh. Do this again for y and z, then make a rectangular prism out of these values, and you have a *Bounding Box*. This box could be oriented relative globally to the world or locally to the object's rotation.

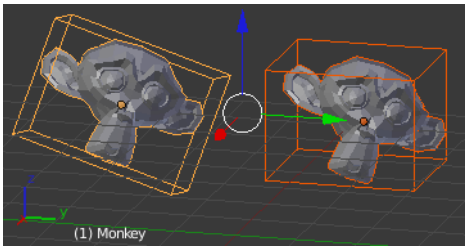


Fig. 2.2420: Demonstration of a Local Bounding Box (left) and a Global Bounding Box (right).

The *x extent*, then, is half of the distance between the x min/max.

Throughout all of this you must be cognizant of the Object Origin. For the Game engine, the default `Ctrl-Alt-Shift-C`, `3` or *Set Origin* → *Origin to Geometry* is unlikely to get the desired placement of the Collision Bounds that you want. Instead, you should generally set the origin by looking at the Tool Shelf after you do the *Set Origin*, and changing the *Center* from *Median Center* to *Bounds Center*. Blender will remember this change for future `Ctrl-Alt-Shift-C` executions.

All Collision Bounds are centered on this origin. All boxes are oriented locally, so object rotation matters.

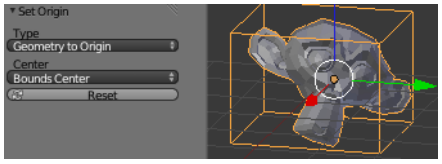


Fig. 2.2421: Setting the origin to Bounds Center instead of Median Center.

A final introductory comment: When you set the Collision Bounds on an object, Blender will attempt to display a visualization of the bounds in the form of a dotted outline. Currently, there is a bug: *The 3D View* does not display this bounds preview where it actually will be during the game. To see it, go to *Game* → *Show Physics Visualization* and look for the white (or green, if sleeping) geometry.

Now we can explain the various options for the *Collision Bounds* settings:

Default For Dynamic and Static objects, it is a Triangle Mesh (see below). For everything else, it is a Sphere (see below).

Capsule Which is a cylinder with hemispherical caps, like a pill. Radius of the hemispheres is the greater of the X or Y extent. Height is the Z bounds

Box The X, Y, Z bounding box, as defined above.

Sphere Radius is defined by the object's scale (visible in the N properties panel) times the physics radius (can be found in *Physics* → *Attributes* → *Radius*. Note: This is the only bounds that respects the Radius option.

Cylinder Radius is the greater of the x or y extent. Height is the z bounds.

Cone Base radius is the greater of the x or y extent. Height is the z bounds.

Convex Hull Forms a shrink-wrapped, simplified geometry around the object.

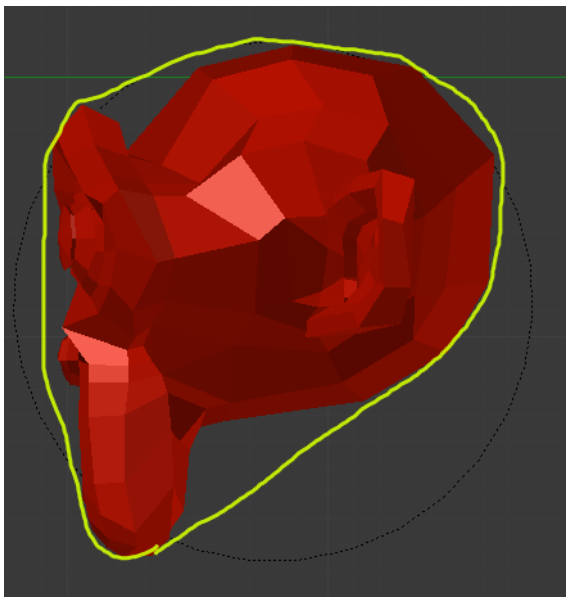


Fig. 2.2422: A convex hull sketch.

Triangle mesh Most expensive, but most precise. Collision will happen with all of triangulated polygons, instead of using a virtual mesh to approximate that collision.

By Hand This is not an option in the Physics tab's Collision Bounds settings, but a different approach, entirely. You create a second mesh, which is invisible, to be the physics representation. This becomes the parent for your display object. Then, your display object is set to ghost so it does not fight with the parent object. This method allows you to strike a balance between the accuracy of *Triangle Mesh* with the efficiency of some of the others. See the demo of this in the dune buggy to the right.

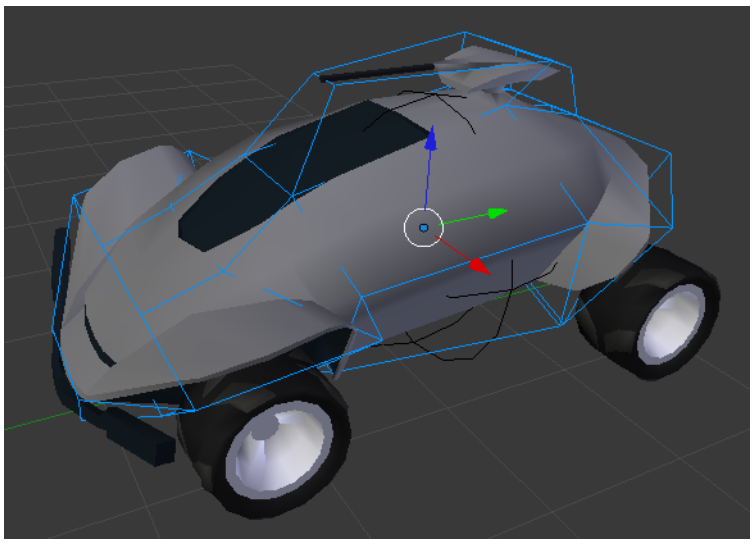


Fig. 2.2423: Another way to create Collision Bounds – By hand.

Options

There are only two options in the Collision Bounds subpanel.

Margin "Add extra margin around object for collision detection, small amount required for stability." If you find your objects are getting stuck in places they should not, try increasing this to, say, 0.06.

Sometimes 0.06 is the default (such as on the Default Cube), but sometimes it is not. You have to keep an eye on the setting, or else learn the symptoms so you can respond when it gives you trouble. If you are lazy/paranoid/unsure/diligent/bored, you can always run this on the Python Console to bump all 0.0 margins to 0.06: for `obj` in `bpy.data.objects`: `obj.game.collision_margin = obj.game.collision_margin` or 0.06

Compound "Add children to form compound collision object." Basically, if you have a child object and do not have this enabled, the child's collisions will not have an effect on that object "family" (though it will still push other objects around). If you do have it checked, the parent's physics will respond to the child's collision (thus updating the whole family). Python property: `obj.game.use_collision_compound`

Create Obstacle

Todo

No Collision Physics

"No Collision" objects in the *Game Engine* are completely unaffected by *Physics*, and do cause physics reactions. They are useful as pure display objects, such as the child of a *Custom Collision Hull* (*Collision Bounds*).

For more documentation, see the *Top BGE Physics page*.

Options

The only option available on No Collision types is:

Invisible Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the Outliner). Python property: `obj.use_render`

Dynamic Physics

Dynamic objects in the *Game Engine* give/receive collisions, but when they do so they themselves do not rotate in response. So, a Dynamic ball will hit a ramp and slide down, while a Rigid Body ball would begin rotating.

If you do not need the rotational response the Dynamic type can save the extra computation.

Note that these objects can still be rotated with *Logic Bricks* or Python code. Their physics meshes will update when you do these rotations – so collisions will be based on the new orientations.

For more documentation, see the *Top BGE Physics page*.

Options

Note: bpy Access

Note that most of these properties are accessible through the non- BGE scripting API via `bpy.data.objects["ObjectName"].game`, which is of type `bpy.types.GameObjectSetting`. This is useful so you can, for example, set a range of objects to have gradated values via a for-loop.

Actor Enables detection by Near and Radar Sensors. Python property: `obj.game.use_actor`

Ghost Disables collisions completely, similar to *No collision*.

Invisible Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the *Outliner*. Python property: `obj.use_render`

Use Material Force Field Materials can have physics settings on them as well: Friction, Elasticity, Force Field (positive or negative force), and also Dampening to other materials. When you turn on this checkbox, you are enabling the Material to exhibit this spring force. Python property: `obj.game.use_material_physics_fh`

Rotate From Normal Todo Python property: `obj.game.use_rotate_from_normal`

No Sleeping Prevents simulation meshes from sleeping. When an object has a linear velocity or angular velocity, it is in motion. It will detect collisions, receive gravity, etc. Once these thresholds are close to zero, it will cease these calculations – until another object interacts with it wake it up. Python property: `obj.game.use_sleep`

Mass Affects the reaction due to collision between objects – more massive objects have more inertia. Will also affect material force fields. Will also change behaviors if you are using the suspension and steering portions of Bullet physics. Python property: `obj.game.mass`

Radius If you have the "Collision Bounds: Sphere" set explicitly (or implicitly through having the Collision Bounds subpanel unchecked), this will multiply with the Object's (unapplied) Scale. Note that none of the other bounds types are affected. Also note that in the 3D View the display will show this for all types, even though it is only actually used with Sphere. Python property: `obj.game.radius`

Basic	Radius= 1.5	Unapplied Scale	Applied Scale	Collision Bounds
Rolls, radius of 1 BU	Rolls, radius of 1.5 BU (after "popping" upward)	Rolls, radius of 1.5 BU	Rolls, radius of 1 BU (!)	Default (which is Sphere)
Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1 BU	Box
""	""	""	""	Convex Hull
Slides, extent of 1 BU (but with more friction than above)	Slides, extent of 1 BU (but with more friction than above)	Acts insane	Slides extent of 1.5 BU	Triangle Mesh

Form Factor For affecting the Inertia Tensor. The higher the value, the greater the rotational inertia, and thus the more resistant to torque. You might think this is strange, considering Dynamic types do not have torque in response to collisions – but you can still see this value's effects when you manually apply Torque. Python property: `obj.game.form_factor`

Anisotropic Friction Isotropic friction is identical at all angles. Anisotropic is directionally-dependant. Here you can vary the coefficients for the three axes individually, or disable friction entirely. Python properties: `obj.game.use_anisotropic_friction` (boolean) and `obj.game.friction_coefficients` (a 3-element array).

Velocity Limit the speed of an object 0 - 1000.

Minimum The object is allowed to be at complete rest, but as soon as it accelerates it will immediately jump to the minimum speed. Python property: `obj.game.velocity_min`

Maximum Top speed of the object. Python property: `obj.game.velocity_max`

Damping- Increase the "sluggishness" of the object.

Translation Resist movement 0 - 1. At "1" the object is completely immobile. Python property: `obj.game.damping`

Rotation Resist rotation, but not the kind of rotation that comes from a collision. For example, if a Motion Controller applies Torque to an object, this damping will be a factor. Python property: `obj.game.rotation_damping`

Lock Translation Seize the object in the world along one or more axes. Note that this is global coordinates, not local or otherwise.

- X Python property: `obj.game.lock_location_x`
- Y Python property: `obj.game.lock_location_y`
- Z Python property: `obj.game.lock_location_z`

Lock Rotation Same, but for rotation (also with respect to the global coordinates).

- X Python property: `obj.game.lock_rotation_x`
- Y Python property: `obj.game.lock_rotation_y`
- Z Python property: `obj.game.lock_rotation_z`

Collision Bounds

The first thing you must understand is the idea of the 3D Bounding Box. If you run through all the vertices of a mesh and record the lowest and highest x values, you have found the *x min/max* the complete boundary for all x values within the mesh. Do this again for y and z, then make a rectangular prism out of these values, and you have a *Bounding Box*. This box could be oriented relative globally to the world or locally to the object's rotation.

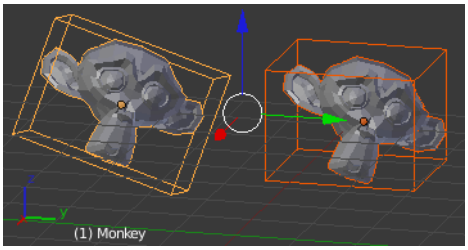


Fig. 2.2424: Demonstration of a Local Bounding Box (left) and a Global Bounding Box (right).

The *x extent*, then, is half of the distance between the x min/max.

Throughout all of this you must be cognizant of the Object Origin. For the Game engine, the default `Ctrl-Alt-Shift-C`, `3` or *Set Origin* → *Origin to Geometry* is unlikely to get the desired placement of the Collision Bounds that you want. Instead, you should generally set the origin by looking at the Tool Shelf after you do the *Set Origin*, and changing the *Center* from *Median Center* to *Bounds Center*. Blender will remember this change for future `Ctrl-Alt-Shift-C` executions.

All Collision Bounds are centered on this origin. All boxes are oriented locally, so object rotation matters.

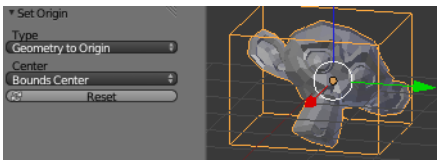


Fig. 2.2425: Setting the origin to Bounds Center instead of Median Center.

A final introductory comment: When you set the Collision Bounds on an object, Blender will attempt to display a visualization of the bounds in the form of a dotted outline. Currently, there is a bug: *The 3D View* does not display this bounds preview where it actually will be during the game. To see it, go to *Game* → *Show Physics Visualization* and look for the white (or green, if sleeping) geometry.

Now we can explain the various options for the *Collision Bounds* settings:

Default For Dynamic and Static objects, it is a Triangle Mesh (see below). For everything else, it is a Sphere (see below).

Capsule Which is a cylinder with hemispherical caps, like a pill. Radius of the hemispheres is the greater of the X or Y extent. Height is the Z bounds.

Box The X, Y, Z bounding box, as defined above.

Sphere Radius is defined by the object's scale (visible in the N properties panel) times the physics radius (can be found in *Physics* → *Attributes* → *Radius*. Note: This is the only bounds that respects the Radius option.

Cylinder Radius is the greater of the x or y extent. Height is the z bounds.

Cone Base radius is the greater of the x or y extent. Height is the z bounds.

Convex Hull Forms a shrink-wrapped, simplified geometry around the object.

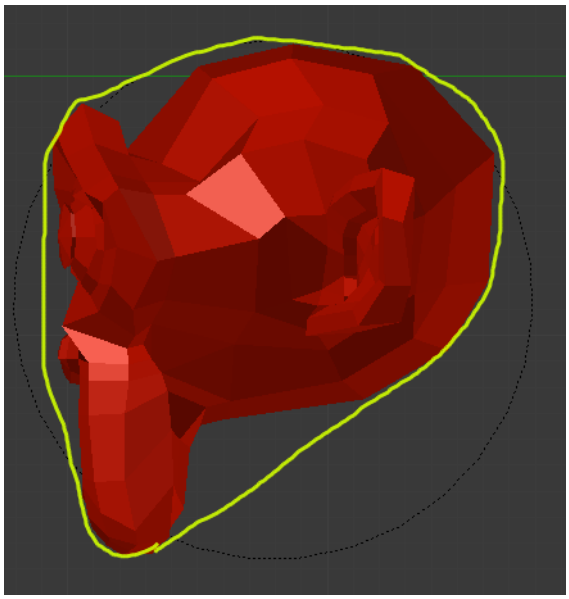


Fig. 2.2426: A convex hull sketch.

Triangle mesh Most expensive, but most precise. Collision will happen with all of triangulated polygons, instead of using a virtual mesh to approximate that collision.

By Hand This is not an option in the Physics tab's Collision Bounds settings, but a different approach, entirely. You create a second mesh, which is invisible, to be the physics representation. This becomes the parent for your display object. Then, your display object is set to ghost so it does not fight with the parent object. This method allows you to strike a balance between the accuracy of *Triangle Mesh* with the efficiency of some of the others. See the demo of this in the dune buggy to the right.

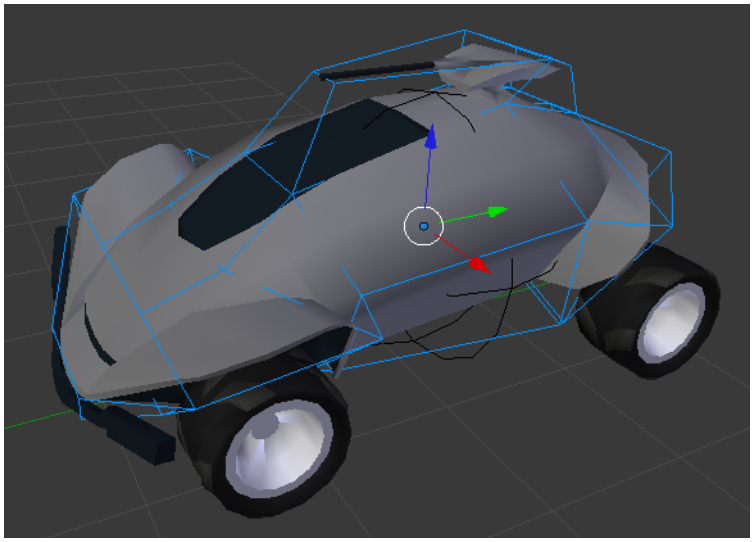


Fig. 2.2427: Another way to create Collision Bounds – By hand.

Options

There are only two options in the Collision Bounds subpanel.

Margin "Add extra margin around object for collision detection, small amount required for stability." If you find your objects are getting stuck in places they should not, try increasing this to, say, 0.06.

Sometimes 0.06 is the default (such as on the Default Cube), but sometimes it is not. You have to keep an eye on the setting, or else learn the symptoms so you can respond when it gives you trouble. If you are lazy/paranoid/unsure/diligent/bored, you can always run this on the Python Console to bump all 0.0 margins to 0.06: for `obj` in `bpy.data.objects`: `obj.game.collision_margin = obj.game.collision_margin` or 0.06

Compound "Add children to form compound collision object." Basically, if you have a child object and do not have this enabled, the child's collisions will not have an effect on that object "family" (though it will still push other objects around). If you do have it checked, the parent's physics will respond to the child's collision (thus updating the whole family). Python property: `obj.game.use_collision_compound`

Create Obstacle

Todo

Rigid Body Physics

Probably the most common type of object in the *Game Engine*. It will give/receive collisions and react with a change in its velocity and its rotation. A Rigid Body ball would begin rotating and roll down (where a *Dynamic* ball would only hit and slide down the ramp).

The idea behind Rigid Body dynamics is that the mesh does not deform. If you need deformation you will need to either go to *Soft Body* or else fake it with animated Actions.

For more documentation, see the *Top BGE Physics page*.

Options

Note: bpy Access

Note that most of these properties are accessible through the non- BGE scripting API via `bpy.data.objects["ObjectName"].game`, which is of type `bpy.types.GameObjectSetting`. This is useful so you can, for example, set a range of objects to have gradated values via a for-loop.

Actor Enables detection by Near and Radar Sensors. Python property: `obj.game.use_actor`

Ghost Disables collisions completely, similar to *No collision*.

Invisible Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the *Outliner*. Python property: `obj.use_render`

Use Material Force Field Materials can have physics settings on them as well: Friction, Elasticity, Force Field (positive or negative force), and also Dampening to other materials. When you turn on this checkbox, you are enabling the Material to exhibit this spring force. Python property: `obj.game.use_material_physics_fh`

Rotate From Normal Todo Python property: `obj.game.use_rotate_from_normal`

No Sleeping Prevents simulation meshes from sleeping. When an object has a linear velocity or angular velocity, it is in motion. It will detect collisions, receive gravity, etc. Once these thresholds are close to zero, it will cease these calculations – until another object interacts with it wake it up. Python property: `obj.game.use_sleep`

Mass Affects the reaction due to collision between objects – more massive objects have more inertia. Will also affect material force fields. Will also change behaviors if you are using the suspension and steering portions of Bullet physics. Python property: `obj.game.mass`

Radius If you have the "Collision Bounds: Sphere" set explicitly (or implicitly through having the Collision Bounds subpanel unchecked), this will multiply with the Object's (unapplied) Scale. Note that none of the other bounds types are affected. Also note that in the 3D View the display will show this for all types, even though it is only actually used with Sphere. Python property: `obj.game.radius`

Basic	Radius= 1.5	Unapplied Scale	Applied Scale	Collision Bounds
Rolls, radius of 1 BU	Rolls, radius of 1.5 BU (after "popping" upward)	Rolls, radius of 1.5 BU	Rolls, radius of 1 BU (!)	Default (which is Sphere)
Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1 BU	Box
""	""	""	""	Convex Hull
Slides, extent of 1 BU (but with more friction than above)	Slides, extent of 1 BU (but with more friction than above)	Acts insane	Slides extent of 1.5 BU	Triangle Mesh

Form Factor For affecting the Inertia Tensor. The higher the value, the greater the rotational inertia, and thus the more resistant to torque. You might think this is strange, considering Dynamic types do not have torque in response to collisions – but you can still see this value's effects when you manually apply Torque. Python property: `obj.game.form_factor`

Anisotropic Friction Isotropic friction is identical at all angles. Anisotropic is directionally-dependant. Here you can vary the coefficients for the three axes individually, or disable friction entirely. Python properties: `obj.game.use_anisotropic_friction` (boolean) and `obj.game.friction_coefficients` (a 3-element array).

Velocity- Limit the speed of an object 0 - 1000.

Minimum The object is allowed to be at complete rest, but as soon as it accelerates it will immediately jump to the minimum speed. Python property: `obj.game.velocity_min`

Maximum Top speed of the object. Python property: `obj.game.velocity_max`

Damping- Increase the "sluggishness" of the object.

Translation Resist movement (0 - 1). At "1" the object is completely immobile. Python property: `obj.game.damping`

Rotation Resist rotation, but not the kind of rotation that comes from a collision. For example, if a Motion Controller applies Torque to an object, this damping will be a factor. Python property: `obj.game.rotation_damping`

Lock Translation Seize the object in the world along one or more axes. Note that this is global coordinates, not local or otherwise.

- X Python property: `obj.game.lock_location_x`
- Y Python property: `obj.game.lock_location_y`
- Z Python property: `obj.game.lock_location_z`

Lock Rotation Same, but for rotation (also with respect to the global coordinates).

- X Python property: `obj.game.lock_rotation_x`
- Y Python property: `obj.game.lock_rotation_y`
- Z Python property: `obj.game.lock_rotation_z`

Collision Bounds

The first thing you must understand is the idea of the 3D Bounding Box. If you run through all the vertices of a mesh and record the lowest and highest x values, you have found the *x min/max* the complete boundary for all x values within the mesh. Do this again for y and z, then make a rectangular prism out of these values, and you have a *Bounding Box*. This box could be oriented relative globally to the world or locally to the object's rotation.

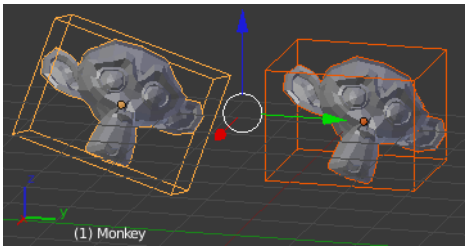


Fig. 2.2428: Demonstration of a Local Bounding Box (left) and a Global Bounding Box (right).

The *x extent*, then, is half of the distance between the x min/max.

Throughout all of this you must be cognizant of the Object Origin. For the Game engine, the default `Ctrl-Alt-Shift-C`, `3` or *Set Origin* → *Origin to Geometry* is unlikely to get the desired placement of the Collision Bounds that you want. Instead, you should generally set the origin by looking at the Tool Shelf after you do the *Set Origin*, and changing the *Center* from *Median Center* to *Bounds Center*. Blender will remember this change for future `Ctrl-Alt-Shift-C` executions.

All Collision Bounds are centered on this origin. All boxes are oriented locally, so object rotation matters.

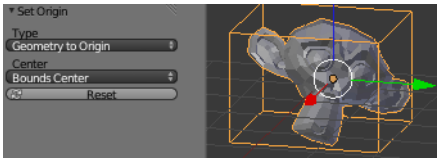


Fig. 2.2429: Setting the origin to Bounds Center instead of Median Center.

A final introductory comment: When you set the Collision Bounds on an object, Blender will attempt to display a visualization of the bounds in the form of a dotted outline. Currently, there is a bug: *The 3D View* does not display this bounds preview where it actually will be during the game. To see it, go to *Game* → *Show Physics Visualization* and look for the white (or green, if sleeping) geometry.

Now we can explain the various options for the *Collision Bounds* settings:

Default For Dynamic and Static objects, it is a Triangle Mesh (see below). For everything else, it is a Sphere (see below).

Capsule – A cylinder with hemispherical caps, like a pill. Radius of the hemispheres is the greater of the X or Y extent. Height is the Z bounds

Box The X, Y, Z bounding box, as defined above.

Sphere Radius is defined by the object's scale (visible in the N properties panel) times the physics radius (can be found in *Physics* → *Attributes* → *Radius*. Note: This is the only bounds that respects the Radius option.

Cylinder Radius is the greater of the x or y extent. Height is the z bounds.

Cone Base radius is the greater of the x or y extent. Height is the z bounds.

Convex Hull Forms a shrink-wrapped, simplified geometry around the object.

Triangle mesh Most expensive, but most precise. Collision will happen with all of triangulated polygons, instead of using a virtual mesh to approximate that collision.

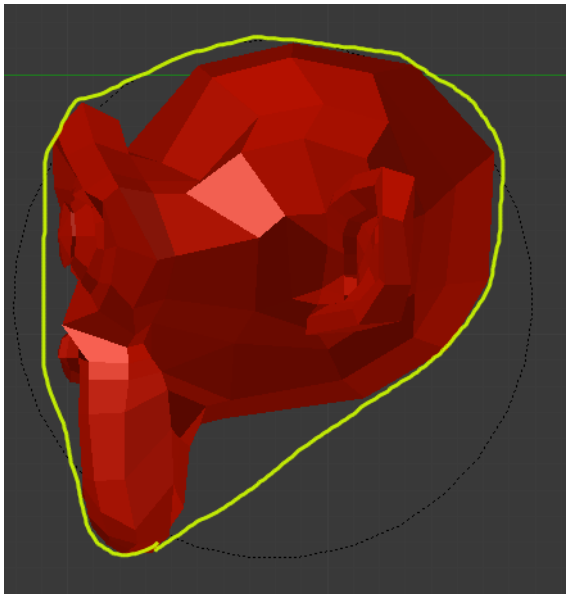


Fig. 2.2430: A convex hull sketch.

By Hand This is not an option in the Physics tab's Collision Bounds settings, but a different approach, entirely. You create a second mesh, which is invisible, to be the physics representation. This becomes the parent for your display object. Then, your display object is set to ghost so it does not fight with the parent object. This method allows you to strike a balance between the accuracy of *Triangle Mesh* with the efficiency of some of the others. See the demo of this in the dune buggy to the right.

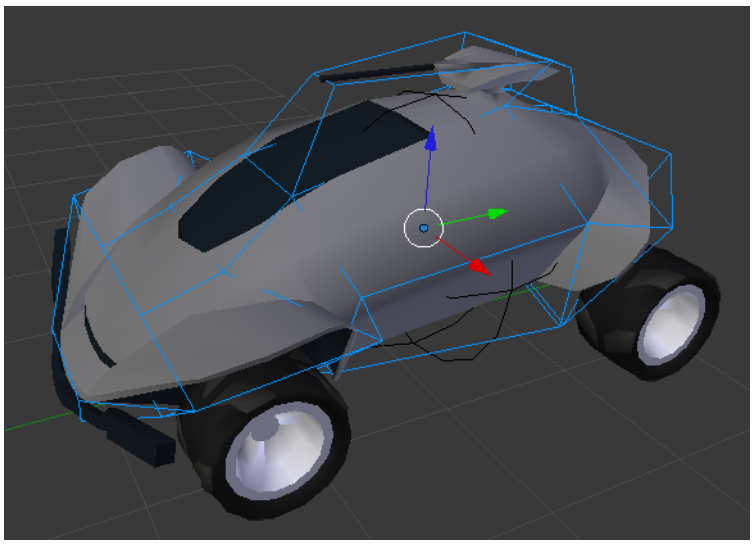


Fig. 2.2431: Another way to create Collision Bounds – By hand.

Options

There are only two options in the Collision Bounds subpanel.

Margin "Add extra margin around object for collision detection, small amount required for stability." If you find your objects are getting stuck in places they should not, try increasing this to, say, 0.06.

Sometimes 0.06 is the default (such as on the Default Cube), but sometimes it is not. You have to keep an eye on the setting, or else learn the symptoms so you can respond when it gives you trouble. If you are

lazy/paranoid/unsure/diligent/bored, you can always run this on the Python Console to bump all 0.0 margins to 0.06: for `obj` in `bpy.data.objects`: `obj.game.collision_margin = obj.game.collision_margin` or 0.06

Compound "Add children to form compound collision object." Basically, if you have a child object and do not have this enabled, the child's collisions will not have an effect on that object "family" (though it will still push other objects around). If you do have it checked, the parent's physics will respond to the child's collision (thus updating the whole family). Python property: `obj.game.use_collision_compound`

Create Obstacle

Todo

Soft Body Physics

The most advanced type of object in the *Game Engine*. Also, it is the most finicky. If you are used to the fun experimentation that comes from playing around with the non-BGE Soft Body sims (such as Cloth), you will probably find a frustrating lack of options and exciting results. Do not despair, we are here to help you get some reasonable settings.

Your setup will involve making sure you have sufficient geometry in the Soft Body's mesh to support the deformation, as well as tweaking the options.

Options

Actor Enables detection by Near and Radar Sensors.

- Default: On.
- Python property: `obj.game.use_actor`

Ghost Disables collisions completely, similar to No Collision.

- Default: Off.
- Python property: `obj.game.use_ghost`

Invisible Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the Outliner).

- Default: Off.
- Python property: `obj.use_render`

Mass Affects the reaction due to collision between objects – more massive objects have more inertia. Will also affect material force fields. Will also change behaviors if you are using the suspension and steering portions of Bullet physics.

- Range: 0.01 - 10,000.
- Default: 1.
- Python property: `obj.game.mass`

Shape Match Upon starting the Game Engine this will record the starting shape of the mesh as the "lowest energy" state. This means that the edges will have tension whenever they are flexed to some other form. This is set to on by default, and in this configuration turns the object into more of a thin sheet of metal rather than a cloth.

- Default: On.
- Python property: `obj.game.soft_body.use_shape_match`

Threshold Linearly scales the pose match.

- A threshold of 1.0 makes it behave like *Shape Match* on with a *Linear Stiffness* of 1.0.

- A threshold of 0.0 makes it behave like *Shape Match* off with a *Linear Stiffness* of 0.0.
- Range: 0-1.
- Default: 0.5.
- Python property: `obj.game.soft_body.shape_threshold`

Welding TODO.

Position Iteration Increase the accuracy at a linearly-increasing expense of time. The effect is visible especially with Soft Bodies that fall on sharp corners, though this can slow down even very simple scenes.

- Range: 0-10.
- Default: 2.
- Python property: `obj.game.soft_body.location_iterations`

Linear Stiffness Linear stiffness of the soft body links. This is most evident when you have *Shape Match* off, but it is also evident with it on.

- Range: 0-1.
- Default: 0.5.
- Python property: `obj.game.soft_body.linear_stiffness`

Friction Dynamic friction coefficient.

Margin Small value makes the algorithm unstable.

Bending Constraint Enable Bending Constraints.

Cluster Collision Affects Collision sensors as well as physics.

Rigid to Soft Body Enable cluster collisions between Rigid and Soft Bodies.

- Default: Off.
- Python property: `obj.game.soft_body.use_cluster_rigid_to_softbody`

Soft to Soft Body Enable cluster collisions among Soft Bodies.

- Default: Off.
- Python property: `obj.game.soft_body.use_cluster_soft_to_softbody`

Iterations Number of cluster iterations.

- Range: 1-128.
- Default: 64.
- Python property: `obj.game.soft_body.cluster_iterations`

Hints

- A very important configurable in the case of Soft Body interactions is *World Properties* → *Physics* → *Physics Steps* → *Substeps*.
- Surprisingly, the more vertices you have in your hit object, the less likely the Soft Body is to react with it. If you try letting it hit a Plane, it might stop, but a subdivided Grid might fail.

Note: Soft bodies do not work with the Collision, Touch, Near, and Radar logic brick sensors.

Warning: A common practice within the non-BGE Cloth simulator is to employ *Force Fields* to animate the cloth. These do not work in the BGE, so you will have to figure out a way to use Python (or perhaps plain Logic Bricks) to apply forces to the Soft Body objects.

Goal Weights

TODO. See Python API.

Vehicle Controller Physics

Introduction

The Vehicle Controller is a special *type of physics object* that the Physics Engine (bullet) recognizes.

It is composed of a *rigid body* representing the chassis and a set of wheels that are set to *no collision*. Emphasizing the distinction between a GameEngine, Logical or Render object and its representation for the Physics Engine is important.

To simulate a vehicle as a true rigid body, on top of also rigid body wheels, with a real suspension system made with joints, would be far too complicated and unstable. Cars and other vehicles are complicated mechanical devices and most often we do not want to simulate that, only that it 'acts as expected'. The Vehicle Controller exists to provide a dedicated way of simulating a vehicle behavior without having to simulate all the physics that would actually happen in the real world. It abstracts the complexity away by providing a simple interface with tweakable parameters such as suspension force, damping and compression.

How it works

Bullet's approach to a vehicle controller is called a "Raycast Vehicle". Collision detection for the wheels is approximated by ray casts and the tire friction is an anisotropic friction model.

A raycast vehicle works by casting a ray for each wheel. Using the ray's intersection point, we can calculate the suspension length and hence the suspension force that is then applied to the chassis, keeping it from hitting the ground. In effect, the vehicle chassis 'floats' along on the rays.

The friction force is calculated for each wheel where the ray contacts the ground. This is applied as a sideways and forwards force.

You can check Kester Maddock's approach to vehicle simulation here. It includes some common problems, workarounds and tips and tricks.

How to use

Currently the Vehicle Controller can only be used as a constraint via Python. There are plans to add it to the interface.

Setup

You should have a body acting as the chassis, set it as a 'Rigid Body'.

The wheels should be separate objects set to 'No Collision'. The vehicle controller will calculate the collisions for you as rays so, if you set it to something else, it will calculate it twice in different ways and produce weird results.

Collisions

A cylinder is typically a good collision shape for the wheels. For the chassis, the shape should be rough, like a box. If the vehicle is very complicated, you should split it into simpler objects and parent those (with their collision shapes) to the vehicle controller so that they will follow it. If your vehicle even has moving bits (weapons, wrecking balls, trolleys etc) they should also be simulated separately and connected to the vehicle as a joint.

Python

Assembling the Vehicle

The overall steps are:

- Create a constraint for the vehicle and save its ID for future reference
- Attach the wheels
- Set wheel parameters: influence, stiffness, damping, compression and friction
- Init variables

You can see an example in the file below.

Controlling the Vehicle

This is done in two parts and it should be modeled according to the desired behavior. You should think of your gameplay and research appropriate functions for the input. For instance, can the vehicle reverse? jump? drift? does it turn slowly? How much time does it take to brake or get to full speed? The first part is *response to keys*. Whenever the player presses a key, you should set a value accordingly, such as increase acceleration. Example:

```
if key[0] == events.UPARROWKEY:
    logic.car["force"] = -15.0
elif key[0] == events.RIGHTARROWKEY:
    logic.car["steer"] -= 0.05
```

The second part is to *compute the movement* according to your functions:

```
## apply engine force ##
for i in range(0, totalWheels):
    vehicle.applyEngineForce(logic.car["force"],i)
...
## slowly ease off gas and center steering ##
logic.car["steer"] *= 0.6
logic.car["force"] *= 0.9
```

Both should be run each frame.

Example

demo_file.zip (last update 9 September 2014)

Occlude Object Physics

If an Occlude type object is between the camera and another object, that other object will not be rasterized (calculated for rendering). It is culled because it is occluded.

There is a demo blend-file to exemplify some concepts: BGE-Physics-Objects-Occluder.blend

- A messed-up, subdivided Cube named “Cube”.
- Another one behind a “Physics Type: Occlude” plane, named “Cube.BG”.
- Another one outside the view Frustum, named “Cube.OffCamera”.

Now observe what happens to the profiling stats for each of the following (in order):

1. Hit `P` as the scene is. It hums along at a fairly slow rate. On my system the Rasterizer step takes 130ms. The framerate will finally jump up once the “Cube” object has completely moved out of the view frustum. It is as if the Occluder does not do anything while the Cube is behind it.
2. Delete the “Cube.OffCamera” object above, and notice that there is no improvement in speed. This is the view frustum culling working for you – it does not matter if that object exists or not.
3. Hit `Z` to view wireframe. Notice that in the 3D View you can see “Cube.BG”, but once you press `P`, it is not there.
4. Make the “Occluder” object take up the whole camera’s view with `S-X-5`. You will see a huge leap in framerate, since almost nothing is being Rasterized. On my system the Rasterizer step drops to 5ms.
5. Try a run with *World properties* → *Physics* → *Occlusion Culling* disabled. It will be slow again.
6. Reenable *World properties* → *Physics* → *Occlusion Culling* and run it one more time to prove to yourself that your speed is back.
7. Change the Occluder to “Physics Type: Static”. Notice that it is back to the original slowness.
8. Change it back to “Physics Type: Occlude”.
9. Now make the “Occluder” invisible. The framerate is back down to its original, slow rate.

Details

As far as Physics is concerned, this type is equivalent to Rigid Object “No collision”. The reason why the Occluder mode is mutually exclusive with other physics mode is to emphasize the fact that occluders should be specifically designed for that purpose and not every mesh should be an occluder. However, you can enable the Occlusion capability on physics objects using Python and Logic bricks. See (Link- TODO)

When an occluder object enters the view frustum, the BGE builds a Z-Depth buffer from the faces of that object. Whether the faces are one-side or two-side is important: only the front faces and two-side faces are used to build the Z-Depth buffer. If multiple occluders are in the view frustum, the BGE combines them and keeps the most foreground faces.

The resolution of the Z-Depth buffer is controllable in the World settings with the “Occlusion Culling Resolution” button:

By default the resolution is 128 pixels for the largest dimension of the viewport while the resolution of the other dimension is set proportionally. Although 128 is a very low resolution, it is sufficient for the purpose of culling. The resolution can be increased to maximum 1024 but at great CPU expense.

The BGE traverses the DBVT (Dynamic Bounding Volume Tree) and for each node checks if it is entirely hidden by the occluders and if so, culls the node (and all the objects it contains).

To further optimize the feature, the BGE builds and uses the Z-Depth buffer only when at least one occluder is in the view frustum. Until then, there is no performance decrease compared to regular view frustum culling.

Recommendations

Occlusion culling is most useful when the occluders are large objects (buildings, mountains...) that hide many complex objects in an unpredictable way. However, do not be too concerned about performance: even if you use it inappropriately, the performance decrease will be limited due to the structure of the algorithm.

There are situations where occlusion culling will not bring any benefit:

- If the occluders are small and do not hide many objects.
  - In that case, occlusion culling is just dragging your CPU down).
- If the occluders are large but hides simple objects.

  - In that case you are better off sending the objects to the GPU).
- If the occluders are large and hides many complex objects but in a very predictable way.
  - Example: a house full of complex objects. Although occlusion culling will perform well in this case, you will get better performance by implementing a specific logic that hides/unhides the objects; for instance making the objects visible only when the camera enters the house).
- Occluders can be visible graphic objects but beware that too many faces will make the Z Depth buffer creation slow.
  - For example, a terrain is not a good candidate for occlusion: too many faces and too many overlap. Occluder can be invisible objects placed inside more complex objects (ex: "in the walls" of a building with complex architecture). Occluders can have "holes" through which you will see objects.

Sensor Physics

The object detects static and dynamic objects but not other collisions sensors objects. The Sensor is similar to the physics objects that underlie the Near and Radar sensors. Like the Near and Radar object it is:

- Static and ghost.
- Invisible by default.
- Always active to ensure correct collision detection.
- Capable of detecting both static and dynamic objects.
- Ignoring collision with their parent.
- Capable of broadphase filtering based on:
  - Actor option: the collisioning object must have the Actor flag set to be detected
  - property/material: as specified in the collision sensors attached to it.

Broadphase filtering is important for performance reason: the collision points will be computed only for the objects that pass the broadphase filter.

- Automatically removed from the simulation when no collision sensor is active on it.

Unlike the Near and Radar object it can:

- Take any shape, including triangle mesh.
- Be made visible for debugging (just use the Visible actuator).
- Have multiple collision sensors using it.

Other than that, the sensor objects are ordinary objects. You can move them freely or parent them. When parented to a dynamic object, they can provide advanced collision control to this object.

The type of collision capability depends on the shape:

- Box, sphere, cylinder, cone, convex hull provide volume detection.
- Triangle mesh provides surface detection but you can give some volume to the surface by increasing the margin in the Advanced Settings panel. The margin applies on both sides of the surface.

Performance tip

- Sensor objects perform better than Near and Radar: they do less synchronizations because of the Scenegraph optimizations and they can have multiple collision sensors on them (with different property filtering for example).
- Always prefer simple shape (box, sphere) to complex shape whenever possible.
- Always use broadphase filtering (avoid collision sensor with empty propery/material)
- Use collision sensor only when you need them. When no collision sensor is active on the sensor object, it is removed from the simulation and consume no CPU.

Known limitations

- When running Blender in debug mode, you will see one warning line of the console:

```
warning btCollisionDispatcher::needsCollision: static-static collision!"
In release mode this message is not printed.
```

- Collision margin has no effect on sphere, cone and cylinder shape.

Settings

Invisible See *Here*

Collision Bounds

See *Here*.

Character Physics

TODO.

Navigation Mesh Physics

TODO.

2.11.7 Performance

Introduction

When developing games, game engineers, software and hardware developers uses some tools to fine tune their games to specific platforms and operating systems, defining a basic usage scenario whereas the users would have the best possible experience with the game.

Most of these tools, are software tools available for the specific Game Engines whereas the games were being developed and will run.

Blender Game Engine also comes with some visual tools to fine tune the games being developed, so the game developers could test the best usage scenario and minimum software and hardware requirements to run the game.

In Blender, those tools are available at the *System* and *Display* panel of *Render* tab in the *Properties editor*. There are options for specific performance adjusts and measurements, ways to control the frame rate or the way the contents are rendered in Blender window (game viewport) while the game runs, as well as controls for maintaining geometry allocated in graphic cards memory.

Blender Game Engine rendering system controls: *System* – Controls for Scene rendering while the game is running.

Blender Game Engine Performance measurements: *Display* – Controls for showing specific data about performance while the game is running.

System

The *System* panel at the Render tab of the Properties editor, lets the game developer specify options about the system performance regarding to frame discards and restrictions about frame renderings, the key to stop the Blender Game Engine, and whether to maintain geometry in the internal memory of the Graphic card.

Options

Fig. 2.2432: System panel in the Render tab.

Use Frame Rate When checked, this will inform Blender whether to run freely without frame rate restrictions or not. The frame rate is specified at the *Display* panel in the *Render* tab of the Properties editor. For more information about frame rates, see the *Display* page.

Display Lists When checked, this will tell Blender to maintain the lists of the meshes geometry allocated at the GPU memory. This can help to speed up viewport rendering during the game if you have enough GPU memory to allocate geometry and textures.

Restrict Animation Updates When checked, this will force the Game Engine to discard frames (even at the middle of redrawing, sometimes causing *tearing* artifacts) if the rate of frame rendered by the GPU is greater than the specified at the *Display* Tab.

Exit Key Clicking at this button will ask the user to type a key to specify a key to stop the Game Engine from running.

Display

The *Display* panel in the *Render* tab of the *Properties* editor, lets the game developer specify the maximum frame rate of the animations shown during the game execution, whether to see informations like framerate and profile, debug properties, physics geometry visualization, warnings, if the mouse cursor is shown during the game execution, and options to specify the framing style of the game to fit the window with the specified resolution.

Options

Animation Frame Rate This number button/slider specify the maximum frame rate at which the game will run. Minimum is 1, maximum is 120.

Debug Properties When checked, if a property was previously checked to be debugged during the game, the values of this property will be shown with the `Framerate and Profile` contents.

Framerate and Profile When checked, this will show values for each of the calculations Blender is doing while the game is running, plus the properties marked to be debugged.

Physics visualization Shows a visualization of physics bounds and interactions (like hulls and collision shapes), and their interaction.

Deprecation Warnings Every time when the game developer uses a deprecated functionality (which in some cases are outdated or crippled OpenGL Graphic cards functions), the system will emit warnings about the deprecated function.

Mouse Cursor Whether to show or not the mouse cursor when the game is running.

Framing There are three types of framing available:

Letterbox Show the entire viewport of the game in display window, using horizontal and/or vertical bars when needed.

Extend Show the entire viewport of the game in display window, viewing more horizontally or vertically.

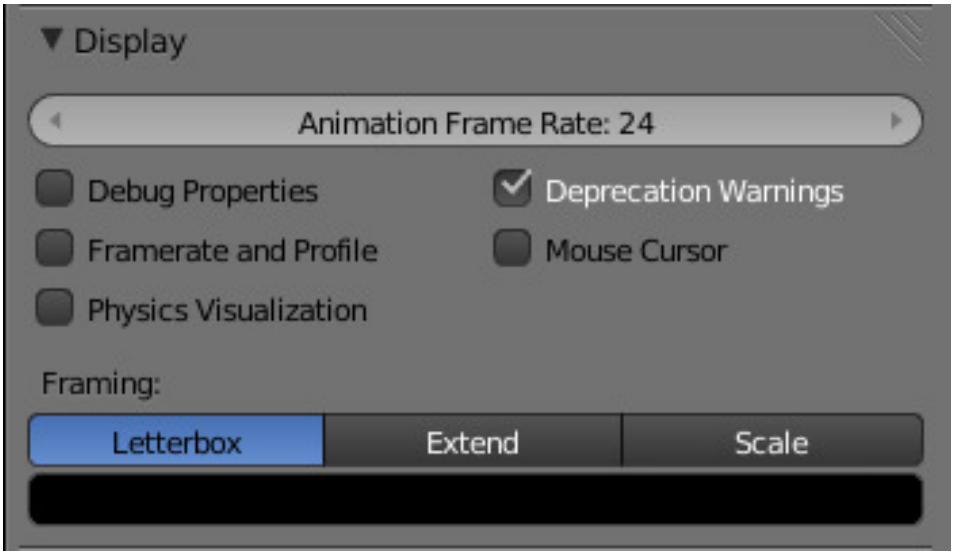


Fig. 2.2433: Display panel at the Render tab.

Scale Stretch or Squeeze the viewport to fill the display window.

Color Bar This will let the game developer choose the bar colors when using the *Letterbox* Framing mode.

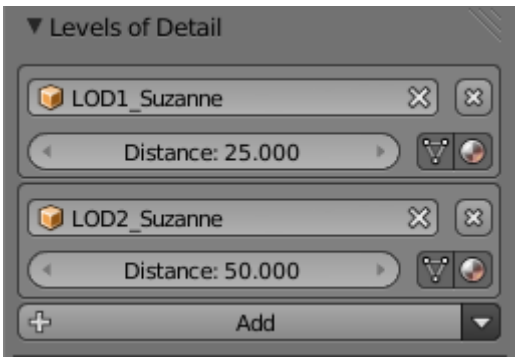
Level of Detail

When creating visual assets it is often desirable to have a high amount of detail in the asset for up close viewing. However, this high amount of detail is wasted if the object is viewed from a distance, and brings down the scene's performance. To solve this, the asset can be swapped out at certain viewing distances. This is commonly referred to as a level of detail system. Each visual step of the asset is known as a level of detail. Levels of detail are most appropriate to use when you have a large scene where certain objects can be viewed both up close and from a distance.

Settings

Note: Modifiers on Level of Detail Objects

Any level of detail objects that have a modifier do not display correctly in the Game Engine. You will need to apply any modifiers for level of detail objects to appear correctly. A fix for this is being looked into.



Level of detail settings can be found in the Object settings when the renderer is set to Blender Game. In the Levels of Detail panel is a button to add a new level of detail to the current object. The settings for each level of detail is displayed in its own box. The exception to this is the base level of detail. This is automatically setup as the current object with a distance setting of 0. To remove a level of detail, click on the X button in the top right corner of the box of the level to be removed.

Object The object to use for this level of detail.

Distance The distance at which this level of detail becomes visible.

Use Mesh When this option is enabled, the mesh from the level of detail object is used until a lower level of detail overrides it.

Use Material When this option is enabled, the material from the level of detail object is used until a lower level of detail overrides it.

Tools

Some tools for making levels of detail easier to manage and create can be found from the select menu next to the add button in the Levels of Detail panel.

Set By Name

Searches the scene for specifically named objects and attempts to set them up as levels of detail on the currently selected object. The selected object must be the base level of detail (e.g. LOD0). This can be useful to quickly setup levels of detail on imported assets. In order to make use of this tool, your naming must be consistent, and each level must be prefixed or suffixed with "lodx" where x is the level that object is intended for. The case on "lod" must be consistent across all objects. Below are some example names that the tool will recognize.

- LOD0_Box, LOD1_Box, LOD2_Box
- Box.lod0, Box.lod1, Box.lod2
- LoD0box, LoD1box, LoD2box

Generate

This tool generates and sets up levels of details based on the selected object. Generation is done using the decimate modifier. Generation does not apply the modifier to allow further changing the settings. Generated objects are automatically named based on the level they are generated for. Below are some settings for the operator.

Count The number of levels desired after generation. This operator creates Count-1 new objects.

Target Size The ratio setting for the decimate modifier on the last level of detail. The ratio settings for the other levels are determined by linear interpolation.

Package into Group With this setting enabled the operator performs some extra tasks to make the asset ready for easy linking into a new file. The base object and all of its levels of detail are placed into a group based on the base object's name. Levels other than the base are hidden for both the viewport and rendering. This simplifies the appearance of the system and does not affect the appearance of the base object. Finally, all levels are parented to the base object to remove clutter from the outliner.

Clear All

Clears the level of detail settings from the current object.

2.11.8 Python API

Introduction

This site is currently under development.

To see the full Python API please click on the following link: Python API.

More informations:

- *Bullet physics*
- *Video Texture*

Bullet physics Python API

Bullet Physics provides collision detection and rigid body dynamics for the Blender Game Engine. It takes some settings from Blender that previously were designed for the former collision detection system (called Sumo).

However, new features do not have a user interface yet, so Python can be used to fill the gap for now.

Features:

- Vehicle simulation.
- Rigid body constraints: hinge and point to point (ball socket).
- Access to internal physics settings, like deactivation time, debugging features.

Easiest is to look at the Bullet physics demos, how to use them. More information can be found here.

Python script example:

```
import PhysicsConstraints
print dir(PhysicsConstraints)
```

Note: Note about parameter settings

Since this API is not well documented, it can be unclear what kind of values to use for setting parameters. In general, damping settings should be in the range of 0 to 1 and stiffness settings should not be much higher than about 10.

The VideoTexture module: bge.texture

The `bge.texture` module allows you to manipulate textures during the game. Several sources for texture are possible: video files, image files, video capture, memory buffer, camera render or a mix of that. The video and image files can be loaded from the Internet using a URL instead of a file name. In addition, you can apply filters on the images before sending them to the GPU, allowing video effect: blue screen, color band, gray, normal map. `bge.texture` uses FFmpeg to load images and videos. All the formats and codecs that FFmpeg supports are supported by `bge.texture`, including but not limited to:

- AVI
- Ogg
- Xvid
- Theora
- dv1394 camera
- video4linux capture card (this includes many webcams)
- videoForWindows capture card (this includes many webcams)
- JPG

How it works

The principle is simple: first you identify an existing texture by object and name, then you create a new texture with dynamic content and swap the two textures in the GPU. The GE is not aware of the substitution and continues to display the object as always, except that you are now in control of the texture. At the end, the new texture is deleted and the old texture restored.

The present page is a guide to the `bge.texture` module with simple examples.

Game preparation

Before you can use the thing `bge.texture` module, you must have objects with textures applied appropriately.

Imagine you want to have a television showing live broadcast programs in the game. You will create a television object and UV-apply a different texture at the place of the screen, for example `tv.png`. What this texture looks like is not important; probably you want to make it dark gray to simulate power-off state. When the television must be turned on, you create a dynamic texture from a video capture card and use it instead of `tv.png`: the TV screen will come to life.

You have two ways to define textures that `bge.texture` can grab:

- Simple UV texture.
- Blender material with image texture channel.

Because `bge.texture` works at texture level, it is compatible with all GE fancy texturing features: GLSL, multi-texture, custom shaders, etc.

First example

Let us assume that we have a game object with one or more faces assigned to a material/image on which we want to display a video.

The first step is to create a `Texture` object. We will do it in a script that runs once. It can be at the start of the game, the video is only played when you refresh the texture; we will come to that later. The script is normally attached to the object on which we want to display the video so that we can easily retrieve the object reference:

```
import bge.texture

contr = GameLogic.getCurrentController()
obj = contr.owner

if not hasattr(GameLogic, 'video'):
```

The check on `video` attribute is just a trick to make sure we create the texture only once.

Find material

```
matID = bge.texture.materialID(obj, 'IMvideo.png')
```

`bge.texture.materialID()` is a handy function to retrieve the object material that is using `video.png` as texture. This method will work with Blender material and UV texture. In case of UV texture, it grabs the internal material corresponding to the faces that are assigned to this texture. In case of Blender material, it grabs the material that has an image texture channel matching the name as first channel.

The `IM` prefix indicates that we are searching for a texture name but we can also search for a material by giving the `MA` prefix. For example, if we want to find the material called `VideoMat` on this object, the code becomes:

```
matID = bge.texture.materialID(obj, 'MAVideoMat')
```

Create texture

`bge.texture.Texture` is the class that creates the `Texture` object that loads the dynamic texture on the GPU. The constructor takes one mandatory and three optional arguments:

gameObj The game object.

materialID Material index as returned by `bge.texture.materialID()`, 0 = first material by default.

textureID Texture index in case of multi-texture channel, 0 = first channel by default. In case of UV texture, this parameter should always be 0.

textureObj Reference to another `Texture` object of which we want to reuse the texture. If we use this argument, we should not create any source on this texture and there is no need to refresh it either: the other `Texture` object will provide the texture for both materials/textures.

```
GameLogic.video = bge.texture.Texture(obj, matID)
```

Make texture persistent

Note that we have assigned the object to a `GameLogic`, `video` attribute that we create for the occasion. The reason is that the `Texture` object must be persistent across the game scripts. A local variable would be deleted at the end of the script and the GPU texture deleted at the same time. `GameLogic` module object is a handy place to store persistent objects.

Create a source

Now we have a `Texture` object but it cannot do anything because it does not have any source. We must create a source object from one of the possible sources available in `bge.texture`:

VideoFFmpeg Moving pictures. Video file, video capture, video streaming.

ImageFFmpeg Still pictures. Image file, image on web.

ImageBuff Image from application memory. For computer generated images, drawing applications.

ImageViewport Part or whole of the viewport (=rendering of the active camera displayed on screen).

ImageRender Render of a non active camera.

ImageMix A mix of two or more of the above sources.

In this example we use a simple video file as source. The `VideoFFmpeg` constructor takes a file name as argument. To avoid any confusion with the location of the file, we will use `GameLogic.expandPath ()` to build an absolute file name, assuming the video file is in the same directory as the blend-file:

```
movie = GameLogic.expandPath('//trailer_400p.ogg')
GameLogic.video.source = bge.texture.VideoFFmpeg(movie)
```

We create the video source object and assign it to the `Texture` object `source` attribute to set the source and make it persistent: as the `Texture` object is persistent, the source object will also be persistent.

Note that we can change the `Texture` source at any time. Suppose we want to switch between two movies during the game. We can do the following:

```
GameLogic.mySources[0] = bge.texture.VideoFFmpeg('movie1.avi')
GameLogic.mySources[1] = bge.texture.VideoFFmpeg('movie2.avi')
```

And then assign (and reassign) the source during the game:

```
GameLogic.video.source = GameLogic.mySources[movieSel]
```

Setup the source

The `VideoFFmpeg` source has several attributes to control the movie playback:

range [start,stop] (floats). Set the start and stop time of the video playback, expressed in seconds from beginning. By default the entire video.

repeat (integer). Number of video replay, -1 for infinite.

framerate (float). Relative frame rate, <1.0 for slow, >1.0 for fast.

scale (bool). Set to True to activate fast nearest neighbor scaling algorithm. Texture width and height must be a power of 2. If the video picture size is not a power of 2, rescaling is required. By default `bge.texture` uses the precise but slow `gluScaleImage()` function. Best is to rescale the video offline so that no scaling is necessary at runtime!

flip (bool). Set to True if the image must be vertically flipped. FFmpeg always delivers the image upside down, so this attribute is set to True by default.

filter Set additional filter on the video before sending to GPU. Assign to one of `bge.texture` filter object. By default the image is send unchanged to the GPU. If an alpha channel is present in the video, it is automatically loaded and sent to the GPU as well.

We will simply set the `scale` attribute to True because the `gluScaleImage ()` is really too slow for real time video. In case the video dimensions are already a power of 2, it has no effect.

```
GameLogic.video.source.scale = True
```

Play the video

We are now ready to play the video:

```
GameLogic.video.source.play()
```

Video playback is not a background process: it happens only when we refresh the texture. So we must have another script that runs on every frame and calls the `refresh ()` method of the `Texture` object:

```
if hasattr(GameLogic, 'video'):
GameLogic.video.refresh(True)
```

If the video source is stopped, `refresh()` has no effect. The argument of `refresh ()` is a flag that indicates if the texture should be recalculated on next refresh. For video playback, you definitively want to set it to True.

Checking video status

Video source classes (such as VideoFFMpeg) have an attribute `status`. If video is playing, its value is 2, if it's stopped, it's 3. So in our example:

```
if GameLogic.video.source.status == 3:
#video has stopped
```

Advanced work flow

True argument in `Texture.refresh()` method simply invalidates the image buffer after sending it to the GPU so that on next frame, a new image will be loaded from the source. It has the side effect of making the image unavailable to Python. You can also do it manually by calling the `refresh ()` method of the source directly.

Here are some possible advanced work flow:

- Use the image buffer in Python (does not effect the Texture):

```
GameLogic.video.refresh(False)
image = GameLogic.video.source.image
# image is a binary string buffer of row major RGBA pixels
# ... use image
# invalidates it for next frame
GameLogic.video.source.refresh()
```

- Load image from source for Python processing without download to GPU:
- Note that we do not even call refresh on the Texture.
- We could also just create a source object without a Texture object:

```
image = GameLogic.video.source.image
# ... use image
GameLogic.video.source.refresh()
```

- If you have more than one material on the mesh and you want to modify a texture of one particular material, get its ID:

```
matID = bge.texture.materialID(gameobj, "MAmat.001")
```

GLSL material can have more than one texture channel, identify the texture by the texture slot where it is defined, here two:

```
tex=bge.texture.Texture(gameobj, matID, 2)
```

Advanced demos

Here is a demo that demonstrates the use of two videos alternatively on the same texture. Note that it requires an additional video file which is the elephant dream teaser. You can replace with another other file that you want to run the demo.

Here is a demo that demonstrates the use of the `ImageMix` source. `ImageMix` is a source that needs sources, which can be any other `Texture` source, like `VideoFFmpeg`, `ImageFFmpeg` or `ImageRender`. You set them with `setSource()` and their relative weight with `setWeight()`. Pay attention that the weight is a short number between 0 and 255, and that the sum of all weights should be 255. `ImageMix` makes a mix of all the sources according to their weights. The sources must all have the same image size (after reduction to the nearest power of two dimension). If they do not, you get a Python error on the console.

2.11.9 Standalone Player

The standalone player allows a Blender game to be run without having to load the Blender system. This allows games to be distributed to other users, without their requiring a detailed knowledge of Blender (and also without the possibility of unauthorized modification). Note that the Game Engine Save as Runtime is an add-on facility which must be pre-loaded before use.

The following procedure will give a standalone version of a working game.

1. *File* → *User Preferences* → *Add-ons* → *Game Engine* → *Save As Game Engine Runtime* enable the checkbox. (You can also *Save User Settings*, in which case the add-on will always be present whenever Blender is re-loaded).
2. *File* → *Export* → *Save As Game Engine Runtime* (give appropriate directory/filename) confirm with *Save as Game Engine Runtime*.

The game can then be executed by running the appropriate `.exe` file. Note that all appropriate libraries are automatically loaded by the add-on.

If you are interested in licensing your game, read Licensing for a discussion of the issues involved.

Tip: Exporting...

If the game is to be exported to other computers, make a new empty directory for the game runtime and all its ancillary libraries etc. Then make sure the **whole** directory is transferred to the target computer

2.11.10 Licensing of Blender Games

Blender and the Blender Game Engine (BGE) are licensed as GNU GPL, which means that your games (if they include Blender software) have to comply with that license as well. This only applies to the software, or the bundle if it has software in it, not to the artwork you make with Blender. All your Blender creations are your sole property.

GNU GPL – also called "Free Software" – is a license that aims at keeping the licensed software free, forever. GNU GPL does not allow you to add new restrictions or limitations on the software you received under that license. That works fine if you want your clients or your audience to have the same rights as you have (with Blender).

In summary, the software and source-code are bound to the GNU GPL, but the blend-files (models, textures, sounds) are not.

Standalone Games

In case you save out your game as a single "Standalone" the blend-file gets included in the binary (the BGE player). That requires the blend-file to be compatible with the GNU GPL license.

In this case, you could decide to load and run another blend-file game (using the Game Actuator logic brick). That file then is not part of the binary, so you can apply any license you wish on it.

More Information

More information you can find in the blender.org FAQ.

2.12 Advanced

This chapter covers advanced use (topics which may not be required for typical usage).

2.12.1 Scripting & Extending Blender

Introduction

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. Python combines remarkable power with very clear syntax.

Python scripts are a powerful and versatile way to extend Blender functionality. Most areas of Blender can be scripted, including Animation, Rendering, Import and Export, Object Creation and the scripting of repetitive tasks.

To interact with Blender, scripts can make use of the tightly integrated API (Application Programming Interface).

General information

Links that are useful while writing scripts:

- Python.org - General information about Python.
- Blender Python API - Official API documentation. Use this for referencing while writing scripts.
- API Introduction - A short introduction to get you started with the API. Contains examples.
- CookBook - A section of handy code snippets (yet to be written)

Links that deal with distributing your scripts:

- Sharing scripts - Information on how to share your scripts and get them included in the official Blender distribution.
- Creating Add-ons - Add-ons are used to encapsulate and distribute scripts.
- Add-ons project - Project to maintain a central repository of extensions to Blender.

Getting Started

Manual links

The following links take you from the basics to the more advanced concepts of Python scripting for Blender.

- *Text Editor*
- *Python Console*
- Info Editors *Report Console*

External links

Here are external links containing a lot of good information to start learning how to write scripts for Blender:

- Introductory tutorial by Satish Goda - Takes you from the beginning and teaches how to do basic API manipulations.
- Ira Krakow's video tutorials - First video in a series of video tutorials.
- Quickstart guide - A quick start guide for people who already have some familiarity with Python and Blender.
- Examples thread - A forum thread containing many short working script examples.
- Introduction to Python - An one-hour video tutorial introducing Python and the Blender API.

Extending Blender

Add-ons

Add-ons are scripts you can enable to gain extra functionality within Blender, they can be enabled from the user preferences.

Outside of the Blender executable, there are literally hundreds of add-ons written by many people:

- Officially supported add-ons are bundled with Blender.
- Other **Testing** add-ons are included in development builds but not official releases. Many of them work reliably and are very useful but are not ensured to be stable for release.

For an overview of all add-ons available see the Scripts Catalog.

Scripts

Apart from add-ons, there are also scripts you can use to extend Blender's functionality:

- Modules: Utility libraries for import into other scripts.
- Presets: Settings for Blender's tools and key configurations.
- Startup: These files are imported when starting Blender. They define most of Blender's UI, as well as some additional core operators.
- Custom scripts: In contrast to add-ons they are typically intended for one-time execution via the *Text Editor*

Saving your own scripts

File location

All scripts are loaded from the `scripts` folder of the *local, system and user paths*.

You can setup an additional search path for scripts in *File Paths* *User Preferences* → *File Paths*.

Installation

Add-ons are conveniently installed through Blender in the *User Preferences*. Click the *Install from File...* button and select the `.py` or `.zip` file.

To manually install scripts or add-ons place them in the `add-ons`, `modules`, `presets` or `startup` directory according to their type. See the description above.

You can also run scripts by loading them in the *Text Editor*.

Scripting & Security

The ability to include Python scripts within blend-files is valuable for advanced tasks such as rigging, automation and using the Game Engine. However, it poses a security risk since Python does not restrict what a script can do.

Therefore, you should only run scripts from sources you know and trust.

Automatic execution is disabled by default, however, some blend-files need this to function properly.

When a blend-file tries to execute a script and is not allowed, a message will appear in the header with the option to **Reload Trusted** or **Ignore** the message.

Fig. 2.2434: Info Header.

Scripts in Blend Files

Auto Execution

Here are the different ways blend-files may automatically run scripts.

Registered Text-Blocks A text block can have its *Register* option enabled which means it will load on start.

Animation Drivers Python expressions can be used to *Drive* values and are often used in more advanced rigs and animations.

Game Engine Auto-Start Scripts are often used for game logic, blend-files can have *Auto Start* enabled with runs the game on load.

Manual Execution

There are other ways scripts in a blend-file may execute that require user interaction (therefore will run even when auto-execution is off), but you should be aware that this is the case since it is not necessarily obvious.

- Running a script in the text editor.
- Rendering with FreeStyle, because FreeStyle uses scripts to control line styles.

- Running the Game Engine.

Controlling Script Execution

Blender provides a number of ways to control whether scripts from a blend-file are allowed to automatically execute.

First of all, the File Browser has the option **Trusted Source** which you can use on a case-by-case basis to control auto-execution.

However, you may forget to set this, or open a file without going through the File Browser – so you can change the default (described next).

Setting Defaults

In the *File* tab of the User Preferences, there is the toggle **Auto Run Python Scripts**.

This means the **Trusted Source** option in the File Browser will be enabled by default, and scripts can run when blend-files are loaded without using the File Browser.

Once enabled you have the option to exclude certain directories, a typical configuration would be to trust all paths except for the download directory.

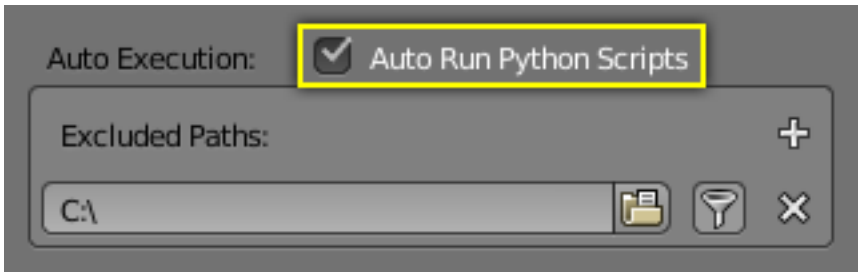


Fig. 2.2435: Auto Run Python Scripts

Command Line

You may want to perform batch rendering or some other task from the command line – running Blender without an interface.

In this case, the User Preferences are still used but you may want to override them:

- Enable with `-y` or `--enable-autoexec`
- Disable with `-Y` or `--disable-autoexec`

Example

Rendering an animation in background mode, allowing drivers and other scripts to run:

```
blender --background --enable-autoexec my_movie.blend --render-anim
```

Note: These command line arguments can be used to start a regular Blender instance and will still override the User Preferences.

2.12.2 Command Line

Introduction

The *Console Window* is an operating system text window that displays messages about Blender operations, status, and internal errors.

Use Cases:

- If Blender exits unexpectedly, the messages may indicate the cause or error.
- To see the output of Python scripts `print()` command.
- To launch Blender with different *Arguments*.
- When troubleshooting, to see the output of `--debug` messages.

Platform Dependant Instructions

Linux

The Blender *Console Window* in Linux will typically only be visible on the desktop if Blender is manually started from a terminal, as Blender outputs to the *Console Window* it is started from.

Fig. 2.2436: Starting Blender from a Linux console window.

Depending on your desktop environment setup, a Blender icon may appear on your desktop or an entry for Blender added to your menu after you install Blender. When you start Blender using a desktop icon or menu entry rather than a Terminal window, the Blender *Console Window* text will most likely be hidden on the Terminal that your XWindow server was started from.

This screenshot shows Blender started from a Linux Terminal and the resulting console text being printed to it.

macOS

macOS uses "files" with the `.app` extension called *applications*. These files are actually folders that appear as files in Finder. In order to run Blender you will have to specify that path to the Blender executable inside this folder, to get all output printed to the terminal. You can start a terminal from *Applications* → *Utilities*. The path to the executable in the `.app` folder is `./blender.app/Contents/MacOS/blender`.

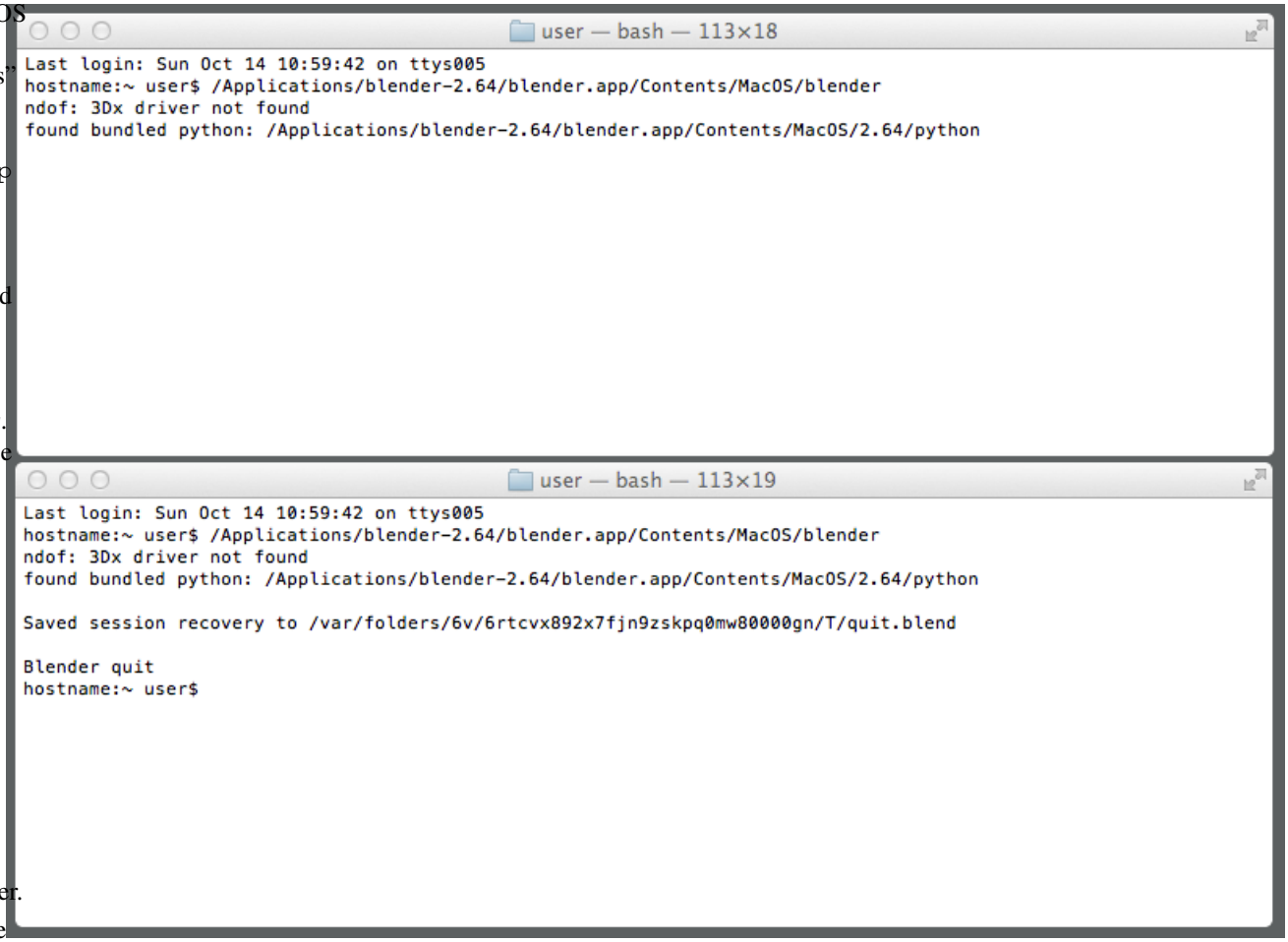


Fig. 2.2437: Starting Blender from a macOS console window.

If you have Blender installed in the Applications folder, the following command can be used:

```
/Applications/blender-2.78/blender.app/Contents/MacOS/blender
```

MS-Windows

When Blender is started on a MS-Windows operating system, the *Console Window* is first created as a separate window on the desktop. The main Blender window will also appear and the *Console Window* will then be toggled off. To display the console again, go to *Window* → *Toggle System Console*.

The screenshot shows the Blender *Console Window* on MS-Windows directly after starting Blender and then a short while later after opening a file along with the relevant messages.

Tip: Closing the Blender Console Window

Closing the *Console Window* will also close Blender, losing any unsaved work.

To turn off the console without closing Blender, just run *Toggle System Console* again from the menu (as mentioned above).

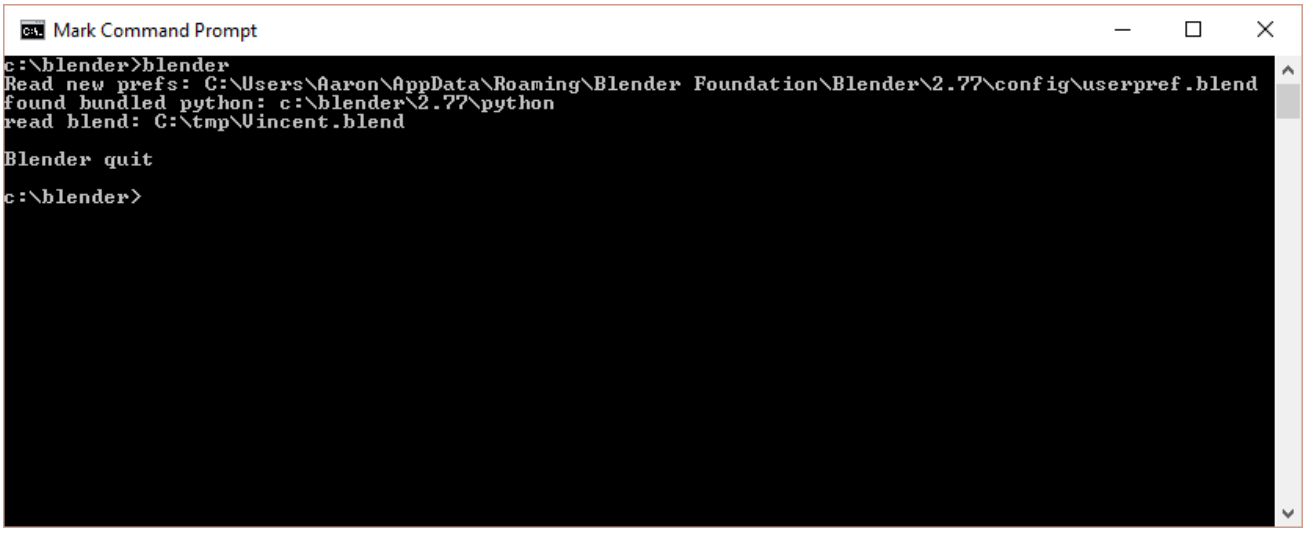


Fig. 2.2438: Blender's Console Window on MS-Windows.

Console Window Status and Error Messages

The *Blender Console Window* can display many different types of Status and Error Messages. Some messages simply inform the user what Blender is doing, but have no real impact on Blender's ability to function. Other messages can indicate serious errors that will most likely prevent Blender carrying out a particular task and may even make Blender non-responsive or shut down completely. The *Blender Console Window* messages can also originate internally from within the Blender code or from external sources such as *Python scripts*.

Common Messages

- `found bundled python: (FOLDER)`

  This message indicates that Blender was able to find the *Python* library for the Python interpreter embedded within Blender. If this folder is missing or unable to be found, it is likely that an error will occur, and this message will not appear.

- `malloc returns nil()`

  When Blender carries out operations that require extra memory (RAM), it calls a function called malloc (short for memory allocate) which tries to allocate a requested amount of memory for Blender. If this cannot be satisfied, malloc will return nil/null/0 to indicate that it failed to carry out the request. If this happens Blender will not be able to carry out the operation requested by the user. This will most likely result in Blender operating very slowly or shutting down. If you want to avoid running out of memory you can install more memory in your system, reduce the amount of detail in your Blender models, or shut down other programs and services which may be taking up memory that Blender could use.

Command Line Arguments

Blender 2.78 Usage: blender [args ...] [file] [args ...]

Render Options

-b, --background Run in background (often used for UI-less rendering)

-a, --render-anim Render frames from start to end (inclusive)

-S, --scene <name> Set the active scene `<name>` for rendering

-f, --render-frame <frame> Render frame <frame> and save it.

- +<frame> start frame relative, -<frame> end frame relative.
- A comma separated list of frames can also be used (no spaces).
- A range of frames can be expressed using `..` seperator between the first and last frames (inclusive).

-s, --frame-start <frame> Set start to frame <frame>, supports +/- for relative frames too.

-e, --frame-end <frame> Set end to frame <frame>, supports +/- for relative frames too.

-j, --frame-jump <frames> Set number of frames to step forward after each rendered frame

-o, --render-output <path> Set the render path and file name. Use `//` at the start of the path to render relative to the blend-file.

The # characters are replaced by the frame number, and used to define zero padding.

- `ani_##_test.png` becomes `ani_01_test.png`
- `test-######.png` becomes `test-000001.png`

When the filename does not contain #, The suffix #### is added to the filename.

The frame number will be added at the end of the filename, eg:

```
blender -b foobar.blend -o //render_ -F PNG -x 1 -a
```

`//render_` becomes `//render_####`, writing frames as `//render_0001.png`

-E, --engine <engine> Specify the render engine use -E help to list available engines

-t, --threads <threads> Use amount of <threads> for rendering and other operations [1-64], 0 for systems processor count.

Format Options

-F, --render-format <format>

Set the render format, Valid options are... TGA RAWTGA JPEG IRIS IRIZ AVIRAW AVIJPEG PNG BMP

(formats that can be compiled into blender, not available on all systems) HDR TIFF EXR MULTILAYER MPEG FRAMESERVER QUICKTIME CINEON DPX DDS JP2

-x, --use-extension <bool> Set option to add the file extension to the end of the file

Animation Playback Options

-a <options> <file(s)>

Playback <file(s)>, only operates this way when not running in background. -p <sx> <sy> Open with lower left corner at <sx>, <sy> -m Read from disk (Do not buffer) -f <fps> <fps-base> Specify FPS to start with -j <frame> Set frame step to <frame> -s <frame> Play from <frame> -e <frame> Play until <frame>

Window Options

-w, --window-border Force opening without borders

-W, --window-borderless Force opening without borders

-p, --window-geometry <sx> <sy> <w> <h> Open with lower left corner at <sx>, <sy> and width and height as <w>, <h>

-con, --start-console Start with the console window open (ignored if -b is set), (Windows only)

--no-native-pixels Do not use native pixel size, for high resolution displays (MacBook `Retina`)

Game Engine Specific Options

-g Game Engine specific options

-g fixedtime	Run on 50 hertz without dropping frames
-g vertexarrays	Use Vertex Arrays for rendering (usually faster)
-g nomipmap	No Texture Mipmapping
-g linearmipmap	Linear Texture Mipmapping instead of Nearest (default)

Python Options

-y, --enable-autoexec Enable automatic Python script execution (default)

-Y, --disable-autoexec Disable automatic Python script execution (pydrivers & startup scripts)

-P, --python <filename> Run the given Python script file

--python-text <name> Run the given Python script text block

--python-expr <expression> Run the given expression as a Python script

--python-console Run blender with an interactive console

--python-exit-code Set the exit-code in [0..255] to exit if a Python exception is raised (only for scripts executed from the command line), zero disables.

--addons Comma separated list of add-ons (no spaces)

Debug Options

-d, --debug Turn debugging on

- Enables memory error detection
- Disables mouse grab (to interact with a debugger in some cases)
- Keeps Python's `sys.stdin` rather than setting it to None

--debug-value <value> Set debug value of `<value>` on startup

--debug-events Enable debug messages for the event system

--debug-ffmpeg Enable debug messages from FFmpeg library

--debug-handlers Enable debug messages for event handling

--debug-libmv Enable debug messages from libmv library

--debug-cycles Enable debug messages from Cycles

--debug-memory Enable fully guarded memory allocation and debugging

--debug-jobs Enable time profiling for background jobs.

--debug-python Enable debug messages for Python

--debug-depsgraph Enable debug messages from dependency graph

--debug-depsgraph-no-threads Switch dependency graph to a single threaded evaluation

--debug-gpumem Enable GPU memory stats in status bar

--debug-wm Enable debug messages for the window manager, also prints every operator call

--debug-all Enable all debug messages

--debug-fpe Enable floating point exceptions

--disable-crash-handler Disable the crash handler

Misc Options

--factory-startup Skip reading the startup.blend in the users home directory

--env-system-datafiles Set the `BLENDER_SYSTEM_DATAFILES` environment variable

--env-system-scripts Set the `BLENDER_SYSTEM_SCRIPTS` environment variable

--env-system-python Set the `BLENDER_SYSTEM_PYTHON` environment variable

-nojoystick Disable joystick support

-noglsl Disable GLSL shading

-noaudio Force sound system to None

-setaudio Force sound system to a specific device NULL SDL OPENAL JACK

-h, --help Print this help text and exit

-R Register blend-file extension, then exit (Windows only)

-r Silently register blend-file extension, then exit (Windows only)

-v, --version Print Blender version and exit

-- Ends option processing, following arguments passed unchanged. Access via Python's `sys.argv`

Experimental Features

--enable-new-depsgraph Use new dependency graph

--enable-new-basic-shader-glsl Use new GLSL basic shader

Other Options

/? Print this help text and exit (windows only)

--debug-freestyle Enable debug messages for FreeStyle

--debug-gpu Enable gpu debug context and information for OpenGL 4.3+.

--disable-abort-handler Disable the abort handler

--verbose <verbose> Set logging verbosity level.

Argument Parsing

Arguments must be separated by white space, eg:

```
blender -ba test.blend
```

...will ignore the `a`

```
blender -b test.blend -f8
```

...will ignore `8` because there is no space between the `-f` and the frame value

Argument Order

Arguments are executed in the order they are given. eg:

```
blender --background test.blend --render-frame 1 --render-output '/tmp'
```

...will not render to `/tmp` because `--render-frame 1` renders before the output path is set

```
blender --background --render-output /tmp test.blend --render-frame 1
```

...will not render to `/tmp` because loading the blend-file overwrites the render output that was set

```
blender --background test.blend --render-output /tmp --render-frame 1
```

...works as expected.

Environment Variables

BLENDER_USER_CONFIG Directory for user configuration files.

BLENDER_USER_SCRIPTS Directory for user scripts.

BLENDER_SYSTEM_SCRIPTS Directory for system wide scripts.

BLENDER_USER_DATAFILES Directory for user data files (icons, translations, ..).

BLENDER_SYSTEM_DATAFILES Directory for system wide data files.

BLENDER_SYSTEM_PYTHON Directory for system python libraries.

TEMP Store temporary files here.

TMP or $TMPDIR Store temporary files here.

SDL_AUDIODRIVER LibSDL audio driver - alsa, esd, dma.

PYTHONHOME Path to the python directory, eg. /usr/lib/python.

2.12.3 Working Limits

Space

While object positions, vertex locations are not clamped, larger values become increasingly imprecise.

To get an idea of the precision you can work with using different scales.

Heres a table of scales and their associated accuracy.

10 1/1,048,576th

100 1/131,072th

1,000 1/16,384th

10,000 1/1,024th

100,000 1/128th

1,000,000 1/16th

Hint: For a rough rule of thumb, values within -5,000/+5,000 are typically reliable (range of 10,000).

Internally *single precision* floating point calculations are used.

Time

The maximum number of frames for each scene is currently 500,000, and allows for continuous shots for durations of:

24 fps 5 hours, 47 seconds.

25 fps 5 hours, 33 seconds.

30 fps 4 hours, 37 seconds.

60 fps 2 hours, 18 seconds.

Note: In practice, a finished work is typically composted of output from many scenes. So this limit does not prevent you from creating longer works.

Text Fields

Fixed strings are used internally, and while it's not useful to list all limits, here are some common limits.

directory 767

file-name 255

file-path 1023

identifier 63

Used for data-block names, modifiers, vertex-groups, UV-layers...

Note: Multi-byte encoding means some unicode characters use more than a single ASCII character.

2.13 Pipeline

This section of the manual focuses on the integration of Blender into a production pipeline. This is a vast topic that covers many areas of the software, but here we will focus on file/asset management and data I/O.

Note: The tools and workflows documented here require familiarity with working with a command line interface and are mostly aimed at TDs and technical users.

2.13.1 BAM Asset Manager

Refactoring linked .blend files is a common practice in a production environment. While some basic operations can be accomplished within Blender, sometimes it is more practical to perform them on the command line or via a script. During the production of Cosmos Laundromat (Gooseberry Open Movie Project) the *BAM Asset Manager* (BAM) was developed. The original scope of BAM included client-server asset management tools going beyond Blender, but it was later refocused on core utilities to perform two operations:

- blendfile packing
- automatic dependencies remapping

The following section of the manual focuses on how to use BAM.

Installing BAM

BAM is a standalone Python package, that can be run on any system without any particular configuration. The only requirement is Python 3 (and pip, the Python package manager, to easily install BAM).

Windows, Linux and macOS provide different ways to install Python 3 and pip. Check out the online docs to learn more about a specific platform.

Once Python 3 and pip are available, BAM can be installed via command line by typing:

```
pip3 install blender-bam
```

After a successful installation, the *bam* command will be availabe. By typing it and pressing the Enter key, all the available subcommands will be displayed.

bam pack

This command is used for packing a `.blend` file and *all* its dependencies into a `.zip` file for redistribution.

```
usage: bam pack [-h] [-o FILE] [-m MODE] [-e PATTERNS] [-a] [-q] [-c LEVEL]
            paths [paths ...]
```

You can simply pack a blend file like this to create a zip-file of the same name.

```
bam pack /path/to/scene.blend
```

You may also want to give an explicit output directory. The example shows how to pack a blend with maximum compression for online downloads

```
bam pack /path/to/scene.blend --output my_scene.zip --compress=best
```

The command provides several options to adapt to different workflows (final distribution, partial extraction, rendering).

-o, --output <FILE> Output file or a directory when multiple inputs are passed

-m, --mode <MODE> Output file or a directory when multiple inputs are passed. Possible choices: `ZIP`, `FILE`

-e, --exclude <PATTERN(S)> Optionally exclude files from the pack.

--exclude="*.png" Using Unix shell-style wildcards *(case insensitive)*.

--exclude="*.txt;*.avi;*.wav" Multiple patterns can be passed using the `;` separator.

-a, --all-deps Follow all dependencies (unused indirect dependencies too)

-q, --quiet Suppress status output

-c, --compress <LEVEL> Compression level for resulting archive Possible choices: `default`, `fast`, `best`, `store`

--repo <DIR PATH> Specify a "root" path from where to pack the selected file. This allows for the creation of a sparse copy of the production tree, without any remapping.

--warn-external Report external libraries errors (missing paths)

Examples

Consider the following directory layout, and in particular the file *01_01_A.lighting.blend* with its linked libraries.

```
~/agent327/
- lib/
   - chars/
   |  - agent.blend  ------------>|
   |  - boris.blend  ------------>|
   |  - barber.blend              |
```

```
   - scenes/                        |
      - 01-opening                  |
      - 01_01_A.lighting.blend  <--|  < BAM pack this file
      - 01_01_A.anim.blend  ------>|
```

Once we run `bam pack /scenes/01-opening/01_01_A.lighting.blend` we obtain a *01_01_A.lighting.zip* inside of which we find the following structure.

```
~/01_01_A.lighting
   - 01_01_A.lighting.blend
   - __/
      - 01_01_A.anim.blend
      - __/
         - lib/
            - chars/
               - agent.blend
               - boris.blend
```

Note how all paths have been remapped relative to the placement of *01_01_A.lighting.blend* in the root of the output. If we run `bam pack /scenes/01-opening/01_01_A.lighting.blend --repo ~/agent327`, the output will be different.

```
~/01_01_A.lighting
   - lib/
   |  - chars/
   |     - agent.blend
   |     - boris.blend
   - scenes
      - 01-opening/
         - 01_01_A.lighting.blend  < The BAM packed file
         - 01_01_A.anim.blend
```

In this case no path is remapped, and we simply strip out any file that is not referenced as a direct or indirect dependency of *01_01_A.lighting.blend*. This is effectively a sparse copy of the original production tree.

bam remap

Remap blend file paths

```
usage: bam remap [-h] {start,finish,reset} ...
```

This command is a 3 step process:

- first run `bam remap start .` which stores the current state of your project (recursively).
- then re-arrange the files on the filesystem (rename, relocate).
- finally run `bam remap finish` to apply the changes, updating the `.blend` files internal paths.

```
cd /my/project

bam remap start .
mv photos textures
mv barbershop_v14_library.blend barberhop_libraray.blend
bam remap finish
```

Note: Remapping creates a file called `bam_remap.data` in the current directory. You can relocate the entire project to a new location but on executing `finish`, this file must be accessible from the current directory.

Note: This command depends on files unique contents, take care not to modify the files once remap is started.

Subcommands

remap start

Start remapping the blend files

```
usage: bam remap start [-h] [-j] [paths [paths ...]]
```

-j, --json Generate JSON output

remap finish

Finish remapping the blend files

```
usage: bam remap finish [-h] [-r] [-d] [-j] [paths [paths ...]]
```

-r, --force-relative Make all remapped paths relative (even if they were originally absolute)

-d, --dry-run Just print output as if the paths are being run

-j, --json Generate JSON output

remap reset

Cancel path remapping

```
usage: bam remap reset [-h] [-j]
```

-j, --json Generate JSON output

2.14 Troubleshooting

2.14.1 Startup

Blender

There are some common causes for problems when using Blender. If you cannot find a solution to your problem here, try asking the community for help.

If Blender crashes on startup there are a few things to check for:

- See if your computer meets the minimum requirements.
- Confirm that your graphics card is supported and that the drivers support at least OpenGL 2.1 .
- Make sure you are using the correct Blender version (32 or 64 bit) for your architecture.

Known causes listed below.

Python

If you get an error on startup like:

```
Fatal Python error: Py_Initialize: unable to load the file system codec
```

you may have set your systems `PYTHONPATH` environment variable.

In this case, Blender's bundled Python will attempt to use the `PYTHONPATH`. If the Python version is different from the version used by Blender, this will crash Blender on startup.

To solve the problem, either clear the `PYTHONPATH` before starting Blender (can also be done with a launcher script), or set it to a compatible Python version.

2.14.2 3D View

Drawing

Depth Buffer Glitches

Sometimes when setting a large *clipping range* will allow you to see both near and far objects, but reduces the depth precision resulting in artifacts.

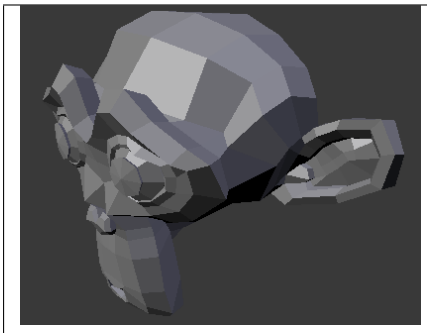


Fig. 2.2439: Model with no clipping artifacts.

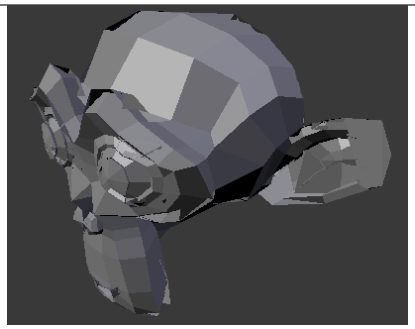


Fig. 2.2440: Model with clipping artifacts.

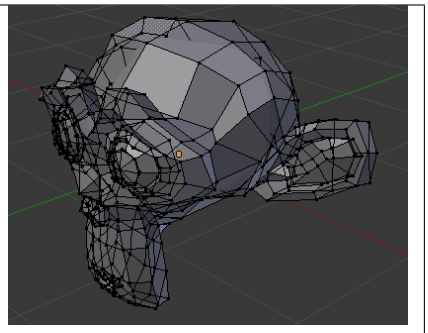


Fig. 2.2441: Mesh with artifacts in Edit Mode.

To avoid this:

- Increase the near clipping when working on large scenes.
- Decrease the far clipping when objects are not viewed at a distance.

When perspective is disabled only the far Clip-End is used, very high values can still give artifacts.

This is **not** specific to Blender, all OpenGL/ DirectX graphics applications have these same limitations.

Objects Invisible in Camera View

If you have a large scene, viewing it through Camera View may not display all of the Objects in the scene. One possibility may be that the *clipping distance* of the camera is too low. The camera will only show objects that fall within the clipping range.

Performance

Slow Drawing

There are a couple of reasons why you may be experiencing a slow viewport.

Old Hardware Sometimes your hardware, mainly your graphics card, may be too slow to keep up with your model.

Upgrade Graphics Driver In some cases, slow selection is resolved by using updated drivers.

Slow Selection

Blender uses OpenGL drawing for selection, some graphics card drivers are slow at performing this operation.

This becomes especially problematic on dense geometry.

Possible Solutions:

OpenGL Occlusion Queries (User Preference) See *User Preferences* → *System* → *Selection*

This option defaults *Automatic*, try setting this to *OpenGL Occlusion Queries*, since there is a significant performance difference under some configurations.

Upgrade Graphics Driver In some cases, slow selection is resolved by using updated drivers. *It is generally good to use recent drivers when using 3D software.*

Select Centers (Workaround) In *Object Mode*, holding `Ctrl` while selecting uses the object center point. While this can be useful on its own, its has the side-effect of not relying on OpenGL selection.

Change Draw Modes (Workaround) Using *Wireframe* or even *Bounding Box* draw modes can be used to more quickly select different objects.

Note: Obviously, the workarounds listed here are not long term solutions, but it is handy to know if you are stuck using a system with poor OpenGL support.

Ultimately, if none of these options work out it may be worth upgrading your hardware.

Navigation

Lost in Space

When navigating your scene, you may accidentally navigate away from your scene and find yourself with a blank viewport. There are two ways to fixes this:

- Select an object in the *Outliner*, then zoom to that object with *View* → *Show Active* or `NumpadPeriod`.
- Use `Home` to fit all objects into the 3D View.

Invisible Limit Zooming In

Sometimes when navigating you may be trying to zoom in but it seems that you have hit a limit to how far you can zoom. This is because Blender uses a central point to orbit around.

In practice this is good for modeling an object which you rotate about a lot to see from all sides (think of a potter using a wheel). However, this makes it awkward to explore a scene or model an object from the 'inside', for example.

Solutions

- Use *View Dolly*
- Use *Walk/Fly modes*.
- Use *Auto Depth* and *Zoom to Mouse Position*. These tool will make sure the distance is always the value under the mouse cursor,
- Use *Border Zoom* as it also resets the center-point when zooming.

- Center the view around the mouse cursor `Alt-F`. This will take the position under the cursor and make it your viewpoint center.
- Center the view around the 3D cursor `Alt-Home`.
- Use a NDOF, also known as a 3D mouse. See *configuring peripherals* for more information.

Tools

Invalid Selection

There are times when selection fails under some configurations, often this is noticeable in mesh *Edit Mode*, selecting vertices/edges/faces where random elements are selected.

Internally Blender uses *OpenGL* for selection, so the graphics card driver relies on giving correct results.

Possible Solutions:

Disable Anti-Aliasing *FSAA, Multi-Sampling* This is by far the most common cause of selection issues.

There are known problems with some graphics cards when using FSAA/multi-sampling.

You can disable this option by:

- Turning FSAA/multi-sampling off in your graphics card driver options.
- Turning *Multi-Sampling* off in the *system preferences*.

Change Anti-Aliasing Sample Settings Depending on your OpenGL configuration, some specific sample settings may work while others fail.

Unfortunately finding working configuration involves trial & error testing.

Upgrade Graphics Driver As with any OpenGL related issues, using recent drivers can resolve problems.

However, it should be noted that this is a fairly common problem and remains unresolved with many drivers.

2.14.3 Graphics Hardware

Blender makes use of OpenGL, which is typically hardware accelerated.

This means issues with the graphics card hardware and drivers can impact on Blender's behavior. This page lists some known issues using Blender on different graphics hardware and how to troubleshoot them.

Performance

When the entire interface is very slow and unresponsive (*even* with the default startup scene), this is likely a problem with the OpenGL configuration.

Unfortunately, in this situation, you may have to do some of your own tests to find the cause, below are some common causes and possible solutions.

Upgrade your OpenGL Driver If you are experiencing any strange graphics problems with Blender, it is always good to double check if you are using the latest drivers.

Disable Anti-Aliasing *FSAA, Multi-Sampling* See *Invalid Selection, Disable Anti-Aliasing*.

Change the *Window Draw Method* This is set in the *system preferences*. It is selected automatically, however, when experiencing problems it's worth checking if changing this resolves interface drawing problems.

2.14.4 Crashes

The most common causes of Blender crashes:

- Running out of memory.
- Issues with graphics hardware or drivers.
- Bugs in Blender.

Firstly, you may be able to recover your work with *File → Recover Last Session*.

To prevent the problem from happening again, you can check that the graphics drivers are up to date, upgrade your machine's hardware (the RAM or graphics card), and disable some options that are more memory intensive:

- Reduce undo steps *User Preferences → Editing → Undo Steps*.
- Disable *Region Overlap* and *Triple buffering* at *User Preferences → System → Window Draw Method*.
- Using multisample, anti-aliasing also increases the memory usage and make display slower.
- On Linux, the Window Manager (KDE, Gnome, Unity) may be using hardware accelerated effects (e.g. window shadows and transparency) that are using up the memory that Blender needs. Try disabling the desktop effects or switch to a lightweight Window Manager.

Crash Log

When Blender crashes it writes out a text file which contains informations that may help identify the cause of the crash.

On a crash, a file is written based on the name of the currently loaded blend file, so `test.blend` will create a file called `test.crash.txt`. The crash log for unsaved files will be written into the *Temporary Directory* directory.

This file contains a log of tools used up until the crash as well as some other debug information.

When reporting bugs on crashes it can be helpful to attach this file to your reports, especially when others are unable to reproduce the crash.

2.14.5 Python Errors

PYTHONPATH

Blender will fail to load if the `PYTHONPATH` is set incorrectly.

This can be useful for Python developers who want to use their own Python installation however, it will prevent Blender from opening at all when set to an incompatible version of Python.

To see if this is the cause of an error temporary unset the environment variable and reload Blender.

See Python's documentation for details.

Pre-Compiled Libraries

While not common practice, Python add-ons can be distributed with their own pre-compiled libraries. Unlike regular Python scripts, these are not portable between different platforms.

It is possible the library is incompatible with your Blender installation (attempting to load a library built for a different version of Python, or loading a 32-bit library on a 64-bit system).

If the add-on contains `.pyd` or `.so` files, check that the distribution is compatible with your operating system.

Platform Specific

MS-Windows

Mixed Python Libraries (DLL's)

If Python is raising errors or you have an add-on that just fails when enabled with an error, eg: `... is not a valid Win32 application.`.

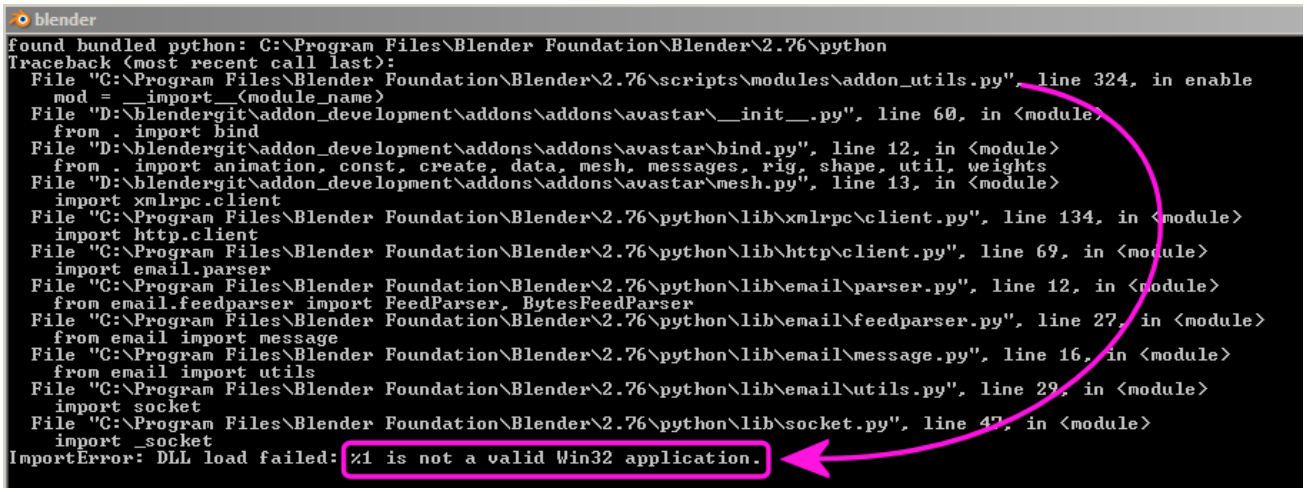


Fig. 2.2442: A Python traceback.

This may be caused by some inconsistency in the Python libraries. While Blender comes with its own bundled Python interpreter, duplicate, incompatible libraries can cause problems.

To find out which Python Library caused the Problem check the error message.

This is normally reported somewhere around the bottom line of the traceback. With the error above you see the problem is caused while trying to import `_socket`. This corresponds to either a file named `_socket.py` or `_socket.pyd`.

To help troubleshoot this problem, the following script can be pasted into the text edit and run to check for duplicate libraries in your search path. (output will show in *Command Line Window*).

```
import os
import sys

# Change this based on the library you wish to test
test_lib = "_socket.pyd"

def GetSystemDirectory():
    from ctypes import windll, create_string_buffer, sizeof
    GetSystemDirectory = windll.kernel32.GetSystemDirectoryA
    buffer = create_string_buffer(260)
    GetSystemDirectory(buffer, sizeof(buffer))
    return os.fsdecode(buffer.value)

def library_search_paths():
    return (
        # Windows search paths
        os.path.dirname(sys.argv[0]),
        os.getcwd(),
        GetSystemDirectory(),
        os.environ["WINDIR"],  # GetWindowsDirectory
        *os.environ["PATH"].split(";"),
```

```
        # regular Python search paths
        *sys.path,
        )

def check_library_duplicate(libname):
    paths = [p for p in library_search_paths()
             if os.path.exists(os.path.join(p, libname))]

    print("Library %r found in %d locations:" % (libname, len(paths)))
    for p in paths:
        print("- %r" % p)

check_library_duplicate(test_lib)
```

2.14.6 Recovering Data

Blender provides a number of ways for the user to recover from mistakes, and reduce the chance of losing his work in the event of operation errors, computer failures, or power outages. There are two ways for you to recover from mistakes or problems:

At the *User Level* (Relating to *Actions*)

- For your actions, there are options like *Undo*, *Redo* and an *Undo History*, used to roll back from mistakes under normal operation, or return back to a specific action.
- Blender also has new features like *Repeat* and *Repeat History*, and the new *Redo Last* which you can use in conjunction with the options listed.

At the *System Level* (Relating to *Files*)

- There are options to save your files like *Auto Save* that saves your file automatically over time, and *Save on Quit*, which saves your blend-file automatically when you exit Blender.

Note: In addition to these functions being enabled by default, the *Save on Quit* functionality cannot be disabled.

Options for Files (System Level)

Save and Auto Save

Computer crashes, power outages, or simply forgetting to save can result in the loss or corruption of your work. To reduce the chance of losing files when those events occur, Blender can use an *Autosave* function. The *File* tab of the *User Preferences* allows you to configure the two ways that Blender provides for you to regress to a previous version of your work.

See *Auto Save* for details.

Recovering Auto Saves

Recover Last Session *File* → *Recover Last Session* will open the `quit.blend` that is saved into the *Temporary Directory* when you exit Blender. Note that files in your *Temporary Directory* may be deleted when you reboot (depending on your system configuration).

Tip: When recovering files, you will navigate to your temporary folder. It is important, when browsing, to enable the detailed list view. Otherwise, you will not be able to figure out the dates of the auto-saved blend-files. (See Fig. *File Browser with detailed view selected.*).

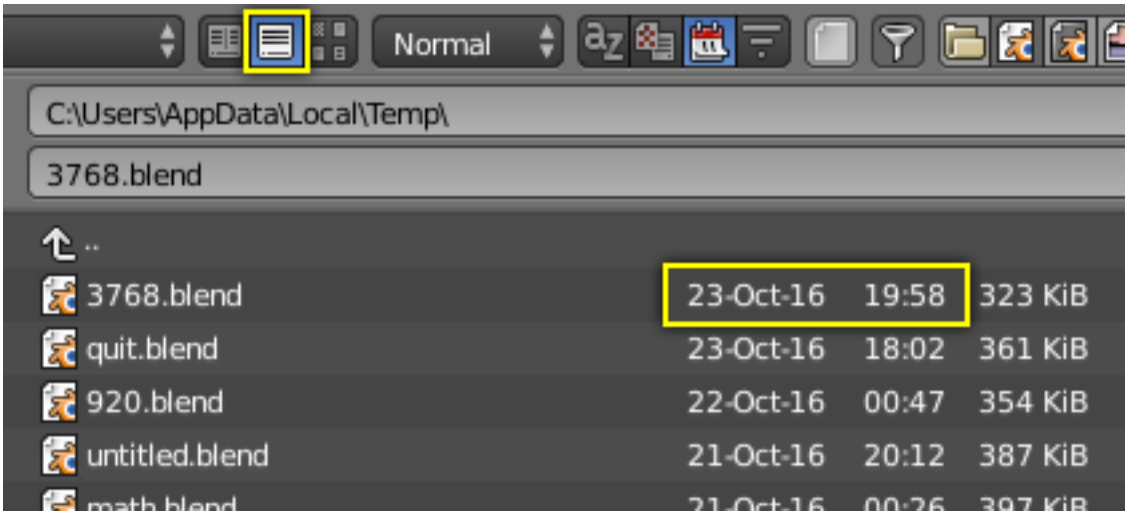


Fig. 2.2443: File Browser with detailed view selected.

Recover Auto Save *File* → *Recover Auto Save...* allows you to open the Auto Saved file. After loading the Auto Saved version, you may save it over the current file in your working directory as a normal blend-file.

Important: When recovering an Auto Saved file, you will lose any changes made since the last *Auto Save* was performed. Only **one** *Auto Saved* file exists for each project (i.e. Blender does not keep older versions. Hence, you will not be able to go back more than a few minutes with this tool).

2.14.7 Compatibility

Some applications which integrate themselves into your system can cause problem's with Blender.

Here is a list of known compatibility issues.

2.15 Glossary

This page lists definitions for terms used in Blender and this manual.

Active One of the three *selection states*. Only one object or item can be active at any given time.

Action Safe Area of the screen visible on most devices. Place content inside it to ensure it does not get cut off.

Actuator A *logic brick* that acts like a muscle of a lifeform. It can move the object, or make a sound.

Aliasing Rendering artifacts in the form of jagged lines.

Alpha Channel Additional channel in an image for transparency.

Straight Alpha Method where RGBA channels are stored as (R, G, B, A) channels, with the RGB channels unaffected by the alpha channel. This is the alpha type used by paint programs such as Photoshop or Gimp, and used in common file formats like PNG, BMP or Targa. So, image textures or output for the web are usually straight alpha.

Premultiplied Alpha Method where RGBA channels are stored as (R × A, G × A, B × A, A), with the alpha multiplied into the RGB channel.

This is the natural output of render engines, with the RGB channels representing the amount of light that comes toward the viewer, and alpha representing how much of the light from the background is blocked. The OpenEXR file format uses this alpha type. So, intermediate files for rendering and compositing are often stored as premultiplied alpha.

Conversion (Straight/Premultiplied) Alpha Conversion between the two alpha types is not a simple operation and can involve data loss, as both alpha types can represent data that the other cannot though it is often subtle.

Straight alpha can be considered to be an RGB color image with a separate alpha mask. In areas where this mask is fully transparent, there can still be colors in the RGB channels. On conversion to premultiplied alpha this mask is *applied'* and the colors in such areas become black and are lost.

Premultiplied alpha, on the other hand, can represent renders that are both emitting light and letting through light from the background. For example, a transparent fire render might be emitting light, but also letting through all light from objects behind it. On converting to straight alpha, this effect is lost.

Ambient Light The light that comes from the surrounding environment as a whole.

Ambient Occlusion A ratio of how much *ambient light* a surface point would be likely to receive. If a surface point is under a foot or table, it will end up much darker than the top of someone's head or the tabletop.

Animation Simulation of motion.

Anti-aliasing See *oversampling*.

Armature An *Object* consisting of *bones*. Used to *rig* characters, props, etc.

Axis A reference line which defines coordinates along one cardinal direction in n-D space.

Axis Angle Rotation method where X, Y, and Z correspond to the axis definition, while W corresponds to the angle around that axis, in radians.

Baking The process of computing and storing the result of a potentially time-consuming calculation so as to avoid needing to calculate it again.

Bevel The operation to chamfer or bevel edges of an object.

BU

Blender Units Internal units used by Blender, equivalent to meters. Often abbreviated to "BU".

Bone The building block of an *Armature*. Made up of a *Head*, *Tail* and *Roll Angle* which define a set of local axes and a point of rotation at the *Head*.

Boolean A type of logic dealing with binary true/false states.

See also *boolean modifier*.

Bounce Refers to the reflection or transmission of a light ray upon interaction with a material. See also *Light Paths*.

Bounding Box The box that encloses the shape of an object. The box is aligned with the local space of the object.

Bump Mapping Technique for simulating slight variations in surface height using a grayscale "height-map" texture.

Bézier A computer graphics technique for generating and representing curves.

BVH

Bounding Volume Hierarchy A hierarchical structure of geometric objects.

See also Bounding Volume Hierarchy on Wikipedia.

Caustics Bright concentrations of light focused by specularly reflecting or refracting objects.

Child An *Object* that is affected by its *Parent*.

Clamp

Clamping Limits a variable to a range. The values over or under the range are set to the constant values of the ranges minimum or maximum.

Blend Modes

Color Blend Modes Methods for blending two colors together.

See also Blend Modes on Wikipedia.

Color Space A coordinate system in which a vector represent a color value.

RGB Red-Green-Blue traditional primary colors also broadcast directly to most computer monitors.

HSV Three values, often considered as more intuitive (human perception) than the RGB system.

Hue The Hue of the color.

Saturation The quantity of hue in the color (from desaturated – a shade of gray – to saturated – brighter colors).

Value The brightness of the color (dark to light).

HSL

Hue, Saturation See HSV.

Luminance TODO.

YUV Luminance-Chrominance standard used in broadcasting analog PAL (European) video.

YCbCr Luminance-ChannelBlue-ChannelRed Component video for digital broadcast use, whose standards have been updated for HDTV and commonly referred to as the HDMI format for component video.

+A The color space holds an additional *Alpha Channel*.

Concave face Face in which one vert is inside a triangle formed by other vertices of the face.

Constraint A way of controlling one *object* with data from another.

Controller A *logic brick* that acts like the brain of a lifeform. It makes decisions to activate muscles (*actuators*), using either simple logic or complex Python scripts.

Convex face Face where, if lines were drawn from each vertex to every other vertex, all lines would remain in the face. Opposite of a *concave face*.

Coplanar Refers to any set of elements that are all aligned to the same 2D plane in 3D space.

Crease Property of an *edge*. Used to define the sharpness of edges in *subdivision surface* meshes.

Curve A type of object defined in terms of a line interpolated between Control Vertices. Available types of curves include *Bézier* and *NURBS*.

Cyclic Often referring to an object being circular. This term is often associated with *Curve*.

DOF

Depth Of Field The distance in front of and behind the subject which appears to be in focus. For any given lens setting, there is only one distance at which a subject is precisely in focus, but focus falls off gradually on either side of that distance, so there is a region in which the blurring is tolerable. This region is greater behind the point of focus than it is in front, as the angle of the light rays change more rapidly; they approach being parallel with increasing distance.

Diffuse Light Even, directed light coming off a surface. For most things, diffuse light is the main lighting we see. Diffuse light comes from a specific direction or location and creates shading. Surfaces facing towards the light source will be brighter, while surfaces facing away from the light source will be darker.

Directional Light The light that has a specific direction, but no location. It seems to come from an infinitely far away source, like the sun. Surfaces facing the light are illuminated more than surfaces facing away, but their location does not matter. A Directional Light illuminates all objects in the scene, no matter where they are.

Displacement Mapping Method for distorting vertices based on an image or texture. Similar to *Bump Mapping*, but instead operates on the mesh's actual geometry. This relies on the mesh having enough geometry to represent details in the image.

Double Buffer Technique for drawing and displaying content on the screen. Blender uses two buffers (images) to draw the interface in. The content of one buffer is displayed while drawing occurs on the other buffer. When drawing is complete, the buffers are switched.

Edge Straight segment (line) that connects two *vertices*, and can be part of a *face*.

Edge Loop Chain of *edges* belonging to consecutive *quads*. An edge loop ends at a pole or a boundary. Otherwise, it is cyclic.

Edge Ring Path of all *edges* along a *face loop* that share two faces belonging to that loop.

Empty An *Object* without any *Vertices*, *Edges* or *Faces*.

Environment Map A method of calculating reflections. It involves rendering images at strategic positions and applying them as textures to the mirror. Now in most cases obsoleted by Raytracing, which though slower is easier to use and more accurate.

Euler

Euler Rotation Rotation method where rotations applied on each X, Y, Z axis component.

Face Mesh element that defines a piece of surface. It consists of three or more *edges*.

Face Loop Chain of consecutive *quads*. A face loop stops at a *triangle* or *Ngon* (which do not belong to the loop), or at a boundary. Otherwise, it is cyclic.

Face Normal The normalized vector perpendicular to the plane that a *face* lies in. Each face has its own normal.

F-Curve A curve that holds the animation values of a specific property.

Field of View The area in which objects are visible to the camera. Also see *Focal Length*

Focal Length The distance required by a lens to focus collimated light. Defines the magnification power of a lens. Also see *Field of View*.

FSAA

Full-Screen Anti-Aliasing A method of *Anti-aliasing* on the graphics card, so the entire image is displayed smooth. Also known as *Multi-Sampling*.

This can be enabled in the *user preferences*. On many graphics cards, this can also be enabled in the driver options.

Gamma An operation used to adjust the brightness of an image.

See also Gamma correction on Wikipedia.

Geometric Center The mean average of the positions of all vertices making up the object.

Gimbal A pivoted support that allows the rotation of an object about a single axis.

See also Gimbal on Wikipedia.

Gimbal Lock The limitation where axes of rotation can become aligned, losing the ability to rotate on an axis (typically associated with *euler rotation*).

- See also Gimbal lock on Wikipedia.
- See also Gimbal lock on Stackexchange.

Global Illumination A superset of radiosity and ray tracing. The goal is to compute all possible light interactions in a given scene, and thus, obtain a truly photo-realistic image. All combinations of diffuse and specular reflections and transmissions must be accounted for. Effects such as color bleeding and caustics must be included in a global illumination simulation.

Global Space See *World Space*.

Gouraud Shading Used to achieve smooth lighting on low-polygon surfaces without the heavy computational requirements of calculating lighting for each pixel. The technique was first presented by Henri Gouraud in 1971.

Head A subcomponent of a *Bone*. The point of rotation for that *Bone*. Has X, Y and Z coordinates measured in the *Local Space* of the *Armature Object*. Used in conjunction with the *Tail* to define the *local* Y axis of the *Bone* in *Pose Mode*. The larger of the two ends when drawn as an *Octahedron*.

HDRI

High Dynamic Range Image A set of techniques that allow a far greater dynamic range of exposures than normal digital imaging techniques. The intention is to accurately represent the wide range of intensity levels found in real scenes, ranging from direct sunlight to the deepest shadows.

See also HDRI on Wikipedia.

IOR

Index Of Refraction A property of transparent materials. When a light ray travels through the same volume it follows a straight path. However, if it passes from one transparent volume to another, it bends. The angle by which the ray is bent can be determined by the IOR of the materials of both volumes.

Interpolation Method of calculating new data between points of known value, like *keyframes*.

Inverse Kinematics The process of determining the movement of interconnected segments of a body or model. Using ordinary Kinematics on a hierarchically structured object you can, for example, move the shoulder of a puppet. The upper and lower arm and hand will automatically follow that movement. IK will allow you to move the hand and let the lower and upper arm go along with the movement. Without IK the hand would come off the model and would move independently in space.

Keyframe A frame in an animated sequence drawn or otherwise constructed directly by the user. In classical animation, when all frames were drawn by animators, the senior artist would draw these frames, leaving the "in between" frames to an apprentice. Now, the animator creates only the first and last frames of a simple sequence (keyframes); the computer fills in the gap.

Keyframing Inserting *Keyframes* to build an animated sequence.

Lattice A type of object consisting of a non-renderable three-dimensional grid of vertices.

See also *Lattice Modifier*.

Layer A device for organizing objects. See also *Layers*.

Local Space A 3D coordinate system that originates (for Objects) at the *Object Center*. or (for Bones) at the *Head* of the *Bone*.

Compare to *World Space*.

Logic brick A graphical representation of a functional unit in Blender's game logic. A Logic brick can be a *Sensor*, *Controller* or *Actuator*.

Manifold Manifold meshes, also called *water tight* meshes, define a *closed non-self-intersecting volume* (see also *non-manifold*). A manifold mesh is a mesh in which the structure of the connected faces in a closed volume will always point the normals (and there surfaces) to the outside or to the inside of the mesh without any overlaps. If you recalculate those normals, they will always point at a predictable direction (To the outside or to the inside of the volume). When working with non-closed volumes, a manifold mesh is a mesh in which the normals will always define two different and non-consecutive surfaces. A manifold mesh will always define an even number of non-overlapped surfaces.

Matte

Mask A grayscale image used to include or exclude parts of an image. A matte is applied as an *Alpha Channel*, or it is used as a mix factor when applying *Color Blend Modes*.

Mesh Type of object consisting of *vertices*, *edges* and *faces*.

Micropolygons A polygon roughly the size of a pixel or smaller.

Motion Blur The phenomenon that occurs when we perceive a rapidly moving object. The object appears to be blurred because of our persistence of vision. Simulating motion blur makes computer animation appear more realistic.

Multi-sampling See *FSAA*.

Ngon A *face* that contains more than four *vertices*.

Non-linear animation Animation technique that allows the animator to edit motions as a whole, not just the individual keys. Nonlinear animation allows you to combine, mix, and blend different motions to create entirely new animations.

Non-manifold Non-Manifold meshes essentially define geometry which cannot exist in the real world. This kind of geometry is not suitable for several types of operations, especially those where knowing the volume (inside/outside) of the object is important (refraction, fluids, booleans, or 3D printing, to name a few). A non-manifold mesh is a mesh in which the structure of a non-overlapped surface (based on its connected faces) will not determine the inside or the outside of a volume based on its normals, defining a single surface for both sides, but ended with flipped normals. When working with non-closed volumes, a non-manifold mesh will always determine at least one discontinuity in the normal directions, either by an inversion of a connected loop, or by an odd number of surfaces. A non-manifold mesh will always define an odd number of surfaces.

There are several types of non-manifold geometry:

- Some borders and holes (edges with only a single connected face), as faces have no thickness.
- Edges and vertices not belonging to any face (wire).
- Edges connected to three or more faces (interior faces).
- Vertices belonging to faces that are not adjoining (e.g. two cones sharing the vertex at the apex).

See also: *Select Non-Manifold* tool.

Normal The normalized vector perpendicular to a surface.

Normals can be assigned to vertices, faces and modulated across a surface using *normal mapping*.

Normal mapping Is similar to *Bump mapping*, but instead of the image being a grayscale heightmap, the colors define in which direction the normal should be shifted, the three color channels being mapped to the three directions X, Y and Z. This allows more detail and control over the effect.

NURBS A computer graphics technique for generating and representing curves and surfaces.

Object Container for a type (Mesh, Curve, Surface, Metaball, Text, Armature, Lattice, Empty, Camera, Lamp) and basic 3D transform data (*Object Center*).

Object Center

Object Origin A reference point used to position, rotate, and scale an *Object* and to define its *Local Space* coordinates.

Octahedron An eight-sided figure commonly used to depict the *Bones* of an *Armature*.

OpenGL The graphics system used by Blender (and many other graphics applications) for drawing 3D graphics, often taking advantage of hardware acceleration.

See also OpenGL on Wikipedia.

Oversampling Is the technique of minimizing *aliasing* when representing a high-resolution signal at a lower resolution.

Also called Anti-Aliasing.

Overscan The term used to describe the situation. when not all of a televised image is present on a viewing screen.

See also Overscan on Wikipedia.

Parent An *Object* that affects its *Child* objects.

Parenting Creating a *Parent*-*Child* relationship between two *objects*.

Particle system Technique that simulate certain kinds of fuzzy phenomena, which are otherwise very hard to reproduce with conventional rendering techniques. Common examples include fire, explosions, smoke, sparks, falling leaves, clouds, fog, snow, dust, meteor tails, stars and galaxies, or abstract visual effects like glowing trails, magic spells. Also used for fur, grass or hair.

Phong Local illumination model that can produce a certain degree of realism in three-dimensional objects by combining three elements: diffuse, specular and ambient for each considered point on a surface. It has several assumptions – all lights are points, only surface geometry is considered, only local modeling of diffuse and specular, specular color is the same as light color, ambient is a global constant.

Pivot Point The pivot point is the point in space around which all rotations, scalings and mirror transformations are centered.

See also the *Pivot Point* docs.

Pixel The smallest unit of information in a 2D raster image, representing a single color made up of red, green, and blue channels. If the image has an *alpha channel*, the pixel will contain a corresponding fourth channel.

Pole *Vertex* where three, five, or more edges meet. A vertex connected to one, two, or four edges is not a pole.

Pose Mode Used for *posing*, *keyframing*, *weight painting*, *constraining* and *parenting* the *bones* of an *armature*.

Posing Moving, Rotating and Scaling the *bones* of an *armature* to achieve an aesthetically pleasing pose for a character.

Premultiplied Alpha See *Alpha Channel*.

Primitive A basic object that can be used as a basis for modeling more complicated objects.

Procedural Texture Computer generated (generic) textures. Procedural textures can be configured via parameters.

Projection In computer graphics, there are two common camera projections used.

Perspective A *perspective* view is geometrically constructed by taking a scene in 3D and placing an observer at point *O*. The 2D perspective scene is built by placing a plane (e.g. a sheet of paper) where the 2D scene is to be drawn in front of point *O*, perpendicular to the viewing direction. For each point *P* in the 3D scene a *PO* line is drawn, passing by *O* and *P*. The intersection point *S* between this *PO* line and the plane is the perspective projection of that point. By projecting all points *P* of the scene you get a perspective view.

Orthographic In an *orthographic* projection, you have a viewing direction but not a viewing point *O*. The line is then drawn through point *P* so that it is parallel to the viewing direction. The intersection *S* between the line and the plane is the orthographic projection of the point *P*. By projecting all points *P* of the scene you get the orthographic view.

Quad

Quadrilateral

Quadrangle *Face* that contains exactly four *vertices*.

Quaternion

Quaternion Rotation Rotation method where rotations are defined by four values (X, Y, Z and W). X, Y, and Z also define an *axis*, and W an angle, but it is quite different from *Axis Angle*.

Radiosity A global lighting method. that calculates patterns of light and shadow for rendering graphics images from three-dimensional models. One of the many different tools which can simulate diffuse lighting in Blender.

See also Radiosity (computer graphics) on Wikipedia.

Raytracing Rendering technique that works by tracing the path taken by a ray of light through the scene, and calculating reflection, refraction, or absorption of the ray whenever it intersects an object in the world. More accurate than *scanline*, but much slower.

Refraction The change in direction of a wave due to a change in velocity. It happens when waves travel from a medium with a given *index of refraction* to a medium with another. At the boundary between the media, the wave changes direction; its wavelength increases or decreases but frequency remains constant.

Render The process of computationally generating a 2D image from 3D geometry.

Rig A system of relationships that determine how something moves. The act of building of such a system.

Roll

Roll Angle The orientation of the local X and Z axes of a *Bone*. Has no effect on the local Y axis as local Y is determined by the location of the *Head* and *Tail*.

Scanline Rendering technique. Much faster than *raytracing*, but allows fewer effects, such as reflections, refractions, motion blur and focal blur.

Sensor A *logic brick* that acts like a sense of a lifeform. It reacts to touch, vision, collision etc.

Shading Process of altering the color of an object/surface in the 3D scene, based on its angle to lights and its distance from lights to create a photorealistic effect.

Smoothing Defines how *faces* are shaded. Face can be either solid (faces are rendered flat) or smooth (faces are smoothed by interpolating the normal on every point of the face).

Specular light A light which is reflected precisely, like a mirror. Also used to refer to highlights on reflective objects.

Straight Alpha See *Alpha Channel*.

Subsurface scattering Mechanism of light transport in which light penetrates the surface of a translucent object, is scattered by interacting with the material, and exits the surface at a different point. All non-metallic materials are translucent to some degree. In particular, materials such as marble, skin, and milk are extremely difficult to simulate realistically without taking subsurface scattering into account.

Subdividing Technique for adding more geometry to a mesh. It creates new vertices on subdivided edges, new edges between subdivisions and new faces based on new edges. If new edges cross a new vertex is created at their crossing point.

Subsurf

Subdivision surface A method of creating smooth higher poly surfaces which can take a low polygon mesh as input.

Sometimes abbreviated to **Subsurf**.

See also Catmull-Clark subdivision surface on Wikipedia.

Tail A subcomponent of a *Bone*. Has X, Y and Z coordinates measured in the *Local Space* of the Armature Object. Used in conjunction with the *Head* to define the *local* Y axis of a *Bone* in *Pose Mode*. The smaller of the two ends when drawn as an *Octahedron*.

Tessellation The tiling of a plane using one or more geometric shapes usually resulting in *Micropolygons*.

Texture Specifies visual patterns on surfaces and simulates physical surface structure.

Texture Space The bounding box to use when using *Generated* mapping to add a *Texture* to an image.

Timecode A coded signal on videotape or film giving information about the frame number, time of recording, or exposure.

Title Safe Area of the screen visible on all devices. Place text and graphics inside this area to make sure they do not get cut off.

Topology The arrangement of *Vertices*, *Edges*, and *Faces* which define the shape of a mesh. See *vertex*, *edge*, and *face*.

Transforms The combined idea of location, rotation and scale.

Triangle *Face* with exactly three *vertices*.

UV map Defines a relation between the surface of a mesh and a 2D texture. In detail, each face of the mesh is mapped to a corresponding face on the texture. It is possible and often common practice to map several faces of the mesh to the same or overlapping areas of the texture.

Vertex

Vertices A point in 3D space containing a location. It may also have a defined color. Vertices are the terminating points of *edges*.

Vertex Group Collection of *vertices*. Vertex groups are useful for limiting operations to specific areas of a mesh.

Voxel A cubicle 3D equivalent to the square 2D pixel. The name is a combination of the terms "Volumetric" and "*Pixel*". Used to store smoke and fire data from physics simulations.

Walk Cycle In animation, a walk cycle is a character that has just the walking function animated. Later on in the animation process the character is placed in an environment and the rest of the functions are animated.

Weight Painting Assigning *vertices* to *Vertex Groups* with a weight of 0.0 - 1.0.

World Space A 3D coordinate system that originates at a point at the origin of the world. Compare to *Local Space*.

Z-buffer Raster-based storage of the distance measurement between the camera and the surface points. Surface points which are in front of the camera have a positive Z value and points behind have negative values. The Z-Depth map can be visualized as a grayscale image.

CHAPTER 3

Get Involved

This manual is maintained largely by volunteers.

Please consider to join the effort and *Contribute to this Manual*.

3.1 About this Manual

3.1.1 Introduction to the Blender Manual

In this manual aims to be a complete and concise source of information to help you to become familiar with the application.

Conventions

Keyboards

Hotkey letters are shown in this manual like they appear on a keyboard; for example,

G refers to the lowercase `g`.

Shift, **Ctrl**, **Alt** are specified as modifier keys.

Ctrl-W, **Shift-Alt-A**, **...** indicates that these keys should be pressed simultaneously

Numpad0 to Numpad9, **NumpadPlus** refer to the keys on the separate numeric keypad.

Other keys are referred to by their names, such as `Esc`, `Tab`, `F1` to `F12`. Of special note are the arrow keys, `Left`, `Right` and so on.

Mice

This manual referrers to mouse buttons as:

LMB Left Mouse Button

RMB Right Mouse Button

MMB Middle Mouse Button

Wheel Scrolling the wheel.

Contribute

The Blender Manual is a community driven effort to which anyone can contribute. Either if you found a typo or if you want to improve the general quality of the documentation, there are several options for helping out. You can:

1. Fix problems, improve the documentation and write new sections – see how to *contribute*.
2. Report problems in the documentation.
3. Get involved in discussions through the mailing list and `#blenderwiki` *IRC channel*.

3.1.2 License

Blender itself is released under the GNU General Public License. More info blender.org/about/license.

Please attribute the "Blender Documentation Team" and include a hyperlink (online) or URL (in print) to manual. For example:

```
The Blender 2.78 Manual
by the Blender Documentation Team
is licensed under a CC-BY-SA v4.0.
```

See Best practices for attribution for further explanation.

It means, that when contributing to the manual you do not hold exclusive copyright to your text. You are, of course, acknowledged and appreciated for your contribution. However, others can change and improve your text in order to keep the manual consistent and up to date.

If you have questions about the license, feel free to contact the Blender Foundation: foundation (at) blender (dot) org

Previous versions of the Blender Manual are made available under a Open Content License v2.26 – v2.77.

3.1.3 What's New

This page lists major changes and additions to the manual.

- New Pipeline Section (rBM3026, Dec. 7).
- *Track Position node* (rBM3014, Dec. 6).
- *Inpaint node* (rBM3011, Dec. 6).
- *Glare node* (rBM2998, Dec. 5).
- Korean Translations (rBM2980, Oct. 29).
- *Corner Pin node* (rBM2978, Oct. 28).
- *Despeckle node* primer (rBM2975, Oct. 27).
- *Stabilizer panel* update (rBM2840, Sep. 27).
- *Displacement controls & bump mapping* (rBM2776, rBM2773; Sep. 20).
- *Double edge mask node* (rBM2775, Aug. 25).
- *Keying node* (rBM2773, Aug. 25).
- *Keying Screen node* (rBM2772, Aug. 25).

3.1.4 Contribute

Whether you like to fix a tiny spelling mistake or rewrite an entire chapter, your help with the Blender manual is most welcome!

How It Works

You can modify the manual by editing local text files. These files are kept in sync with those online via a repository, based on this the server will update the online manual.

The manual is written in the reStructuredText (RST) markup language and can be edit using a plain text editor. For a local preview, you convert (build) the manual source files from RST into HTML web pages.

Getting Started

The following guides lead you through the process.

Install

This section documents how to install the software used to generate the manual. The installation is different for each operating system, instructions have been written for:

Installation on Linux

This guide covers the following topics:

1. *Installing Dependencies*
2. *Downloading the Repository*
3. *Setting up the Build Environment*

Installing Dependencies

Below are listed the installation commands for popular Linux distributions.

For the appropriate system, run the command in a terminal:

Debian/Ubuntu

```
sudo apt-get install python python-pip subversion
```

Redhat/Fedora

```
sudo yum install python python-pip
```

Arch Linux

```
sudo pacman -S python python-pip subversion
```

Downloading the Repository

Simply check out the blender-manual repository using:

```
   cd ~
   svn checkout https://svn.blender.org/svnroot/bf-manual/trunk/blender_docs

The repository will now be downloaded which may take a few minutes depending on your␣
↪internet connection.
```

Setting up the Build Environment

In a terminal, enter the `blender_docs` folder which was just added by the SVN checkout:

```
cd ~/blender_docs
```

Inside that folder is a file called `requirements.txt` which contains a list of all the dependencies we need. To install these dependencies, we can use the `pip` command:

```
sudo pip install -r requirements.txt
```

Note: Every now and then you may want to make sure your lib dependencies are up to date using:

```
sudo pip install -r requirements.txt  --upgrade
```

Continue with the next step: *Building*

Installation on macOS

This guide covers the following topics:

1. *Installing Dependencies*
2. *Downloading the Repository*
3. *Setting up the Build Environment*

Note: This guide relies heavily on command-line tools. It assumes you are the least familiar with the macOS Terminal application.

Installing Dependencies

Install those packages or make sure you have them in your system.

- Python
- PIP
- Subversion

Downloading the Repository

Simply check out the blender-manual repository using:

```
   cd ~
   svn checkout https://svn.blender.org/svnroot/bf-manual/trunk/blender_docs

The repository will now be downloaded which may take a few minutes depending on your␣
↪internet connection.
```

Setting up the Build Environment

In a terminal, enter the `blender_docs` folder which was just added by the SVN checkout:

```
cd ~/blender_docs
```

Inside that folder is a file called `requirements.txt` which contains a list of all the dependencies we need. To install these dependencies, we can use the `pip` command:

```
sudo pip install -r requirements.txt
```

Note: Every now and then you may want to make sure your dependencies are up to date using:

```
sudo pip install -r requirements.txt  --upgrade
```

Continue with the next step: *Building*

Installation on MS-Windows

This guide covers the following topics:

1. *Installing Python* (used to "convert" the source files to HTML)
2. *Installing SVN and Downloading the Repository*
3. *Setting up the Build Environment*

Installing Python

1. Download the Python installation package for MS-Windows. In this guide version 3.5.x is used.
2. Install Python with the installation wizard. In this guide the default settings are used.

Installing SVN and Downloading the Repository

In this guide, we will use TortoiseSVN though any Subversion client will do.

1. Download TortoiseSVN for MS-Windows.
2. Install TortoiseSVN with the installation wizard. When choosing which features will be installed, it is recommended that you enable *command line client tools* to give you access to SVN from the command line (there is no harm in doing this, and it may be helpful if you ever run into any trouble).
3. Once the installation has finished, create a new folder that will contain everything related to the Blender Manual. In this guide, we will use `C:\blender_docs`.
4. Open the new folder, right click and choose *SVN Checkout...* from the context menu.
5. In the *URL of repository* field, enter: `https://svn.blender.org/svnroot/bf-manual/trunk/blender_docs`.
6. In the *Checkout directory* field, enter: `C:\blender_docs`.
7. Click *OK* - the repository will now be downloaded which may take a few minutes depending on your internet connection.

Setting up the Build Environment

- Open a command prompt and change to the repository folder using:

```
cd C:\blender_docs
```

- Install all the dependencies using Python's `pip` command:

```
pip install -r requirements.txt
```

- If all goes well, you should see the following message when it is finished:

```
Successfully installed Jinja2 MarkupSafe Pygments Sphinx docutils sphinx-rtd-
↪theme Cleaning up...
```

During the setup, some warnings may be shown, but do not worry about them. However, if any errors occur, they may cause some problems.

Note: Every now and then you may want to make sure your dependencies are up to date using:

```
pip install -r requirements.txt  --upgrade
```

Continue with the next step: *Building*

Build

The building process is different for each operating system, instructions have been written for:

Building on Linux

Converting the rst files into pretty html pages.

Open a terminal to the folder `~/blender_docs` and simply run:

```
make
```

This is the command you will always use when building the docs. The building process may take several minutes the first time (or after any major changes), but the next time you build it should only take a few seconds.

Viewing the local manual

Once the docs have been built, all the html files can be found inside `~/blender_docs/build/html`. Try opening `build/html/contents.html` in your web browser and read the manual:

```
xdg-open build/html/contents.html
```

Now that you are able to build the manual, the next paragraph is about an optional quick build.

Building a Single Chapter

If you are working on a specific chapter of the manual, you can build it quickly using:

```
make <chapter name>
```

For example, to build only the documentation for the modifiers, use `make modifiers`. You can then view this quick build by opening `build/html/contents_quicky.html`.

This will build very quickly, but it will mean your next complete build of all the chapters will be slow.

Continue with the next step: *Editing*

Building on macOS

Converting the rst files into pretty html pages.

Open a terminal to the folder `~/blender_docs` and simply run:

```
make
```

This is the command you will always use when building the docs. The building process may take several minutes the first time (or after any major changes), but the next time you build it should only take a few seconds.

Viewing the local manual

Once the docs have been built, all the HTML files can be found inside `~/blender_docs/build/html`. Try opening `build/html/contents.html` in your web browser and read the manual:

```
open build/html/contents.html
```

Now that you are able to build the manual, the next paragraph is about an optional quick build.

Building a Single Chapter

If you are working on a specific chapter of the manual, you can build it quickly using:

```
make <chapter name>
```

For example, to build only the documentation for the modifiers, use `make modifiers`. You can then view this quick build by opening `build/html/contents_quicky.html`.

This will build very quickly, but it will mean your next complete build of all the chapters will be slow.

Continue with the next step: *Editing*

Building on MS-Windows

Converting the rst files into pretty html pages. There are two ways to run the build process.

File browser

Run `make.bat` in the `C:\blender_docs` folder.

Tip: Create a desktop shortcut to `make`.

Command prompt

1. Or open a command prompt and change to the repository with `cd C:\blender_docs`.
2. Build using the following command:

```
make
```

The building process may take several minutes the first time (or after any major changes), but the next time you build it should only take a few seconds.

Note: If you encounter an error ending with `TypeError: an integer is required (got type str)`, you may need to install an older version of *Babel* (the Python Internationalization Library).

To do this, simply run:

```
pip install sphinx "babel<2.0"
```

Viewing the local manual

Once the docs have been built, all the HTML files can be found inside `C:\blender_docs\build\html`. Try opening `\build\html\contents.html` in your web browser and read the manual.

Continue with the next step: *Editing*

Editing the manual

Update

Firstly, make sure that your local copy of the manual is up to date with the online repository using:

```
svn update
```

Writing

You can now edit the documentation files, which are the `.rst` files inside the `manual` folder using a text editor of your choice.

For instructions on using different editors, see the Editor Configuration section of the community wiki.

Be sure to check the *Writing Style Guide* for conventions and *Markup Style Guide* to learn how to write in the reStructured-Text markup language.

Happy writing!

Bigger Changes

If you are going to add or overhaul a section, be sure to check carefully that it does not already exist. In some places, the docs are so disorganized that sections may be duplicated or in a strange location. In the case that you find a duplicate or out of place section, create a task explaining the issue, and optionally include a revision (actual changes).

Before you make any edits that are not simple and plainly justified (for example, moving folders around), you should verify with a manual maintainer that your contribution is along the community's vision for the manual. This ensures the best use of your time and good will as it is otherwise possible that, for some reason, your changes will conflict and be rejected or

need time-consuming review. For example, another person may be already working on the section you wish to change, the section may be scheduled for deletion or to be updated according to a planned change to Blender.

Communicating early and frequently is the key to have a productive environment, to not waste people's effort and to attain a better Blender manual as a result.

Getting help/answers

If you are in doubt about functionality that you wish to document, you should pose your questions to the Blender developers responsible for that area or ask at the unofficial user support channel at `#blender` *IRC channel*.

Blender has its own system of module owners with developer and artist members who are responsible for the design and maintenance of the assigned Blender areas. See the module owners page for more information.

Preview

To view your changes, build the manual *as instructed*. Keep in mind that you can also build only the chapter you just edited to view it quickly. Open the generated `.html` files inside the `build/html` folder using your web browser, or refresh the page if you have it open already.

Upload

When you are happy with your changes, you can upload them, so that they will be in the online manual. At first, this is done by submitting patches so that someone can review your changes and give you feedback. After, you can commit your changes directly. This process is described in detail in the next section.

Patch & Commit

Submit Patches

The first few times you make changes to the manual, you will need to submit them as patches for the section owner to review. This is just to make sure that we maintain a quality user manual, and that you do not accidentally break anything vital before you get used to the system.

In order to submit a patch, follow this process:

1. Make any changes that you want
2. Create a patch file by running:

   ```
   svn diff > filename.diff
   ```

   This creates a simple text file that shows what text was added, removed or changed between your working files and the central repository.

   If you have created or deleted files, you will need to run `svn add /path/to/file` or `svn rm /path/to/file` before creating the diff. To see a list of affected files, run `svn status`.
3. Upload the diff file here. If you do not have an account already, you can register for one.
4. After submitting the diff, you will be asked to "Create a new Revision" before you can add a title and description of your changes.
5. If you know who the Section Owner (see Documentation Team) of that chapter is, assign them as the *Reviewer* and they will be notified of your patch. If you cannot find out who that is (or there is no one), instead, mail the bf-docboard mailing list, or tell someone in `#blenderwiki` on *IRC*.
6. They will review your patch and let you know about any changes you could make, or commit the patch if it is accepted.

Note: If your patch includes changes to or additional images, simply attach them when you are creating the revision.

Straight-forward patches are bound to be accepted very quickly. Once you get accustomed to making changes and no longer need feedback, we cut out the middle man and give you direct access to edit the manual.

Commit Directly

Instead of creating a patch file, committing will submit the change directly to our central repository.

All you need to do now is run:

```
svn commit -m "This is what I did"
```

If you leave out `-m "message"`, you will be prompted to type the message in a text editor.

Do not forget to always run `svn update` before committing.

Then you will be asked for your user name (from `developer.blender.org`) and password before the change is committed.

Your modified files are uploaded to the central repository for others to work with and continue collaborating. Commits are tracked in the repositories Diffusion. Soon after your changes become visible in the online manual.

Style Guides

Markup Style Guide

This page covers the conventions for writing and use of the reStructuredText (RST) markup syntax.

Conventions

- Three space indentation.
- Lines should be less than 120 characters long.
- Use italics for button/menu names.

Other loose conventions:

- Avoid Unicode characters.
- Avoid heavily wrapped text (i.e. sentences can have their own lines).

Headings

```
################
 Document Part
################

***************
Document Chapter
***************

Document Section
================

Document Subsection
-------------------
```

```
Document Subsubsection
^^^^^^^^^^^^^^^^^^^^^^

Document Paragraph
""""""""""""""""""
```

Note: *Parts* should only be used for contents or index pages.

Note: Each `.rst` file should only have one chapter heading (`*`) per file.

Text Styling

See the overview on ReStructured Text for more information on how to style the various elements of the documentation and on how to add lists, tables, pictures and code blocks. The Sphinx reference provides more insight additional constructs.

The following are useful markups for text styling:

```
*italic*
**bold**
``literal``
```

Interface Elements

- `:kbd:`LMB`` – keyboard and mouse shortcuts.
- `*Mirror*` – interface labels.
- `:menuselection:`3D View --> Add --> Mesh --> Monkey`` – menus.

Code Samples

There is support for syntax highlighting if the programming language is provided, and line numbers can be optionally shown with the `:linenos:` option:

```
.. code-block:: python
   :linenos:

   import bpy
   def some_function():
       ...
```

Images

Figures should be used to place images:

```
.. figure:: /images/editors_menu.png

   Image Caption.
```

Files

No Caps, No Gaps Lower case filenames underscore between words.

Sort Usefully Order naming with specific identifiers at the end.

Format Use `.png` for images that have solid colors such as screenshots of the Blender interface, and `.jpg` for images with a lot of color variances, such as sample renders and photographs.

Do not use animated `.gif` files, these are hard to maintain, can be distracting and are usually large in file size. If a video is needed, use YouTube or Vimeo (see *Videos* below).

Location Place the image in the `manual/images` folder. Use no other subfolders.

Naming Image files should be named: `chapter_subsection_id.png`, eg:

- `render_cycles_lighting_example_01.jpg`
- `interface_intro_splash.jpg`
- `interface_ui_panel.jpg`

Do not use special characters or spaces

Usage Guides

- Avoid specifying the resolution of the image or its alignment, so that the theme can handle the images consistently and provide the best layout across different screen sizes.
- When documenting a panel or section of the UI, it is better to use a single image that shows all of the relevant areas (rather than multiple images for each icon or button) placed at the top of the section you are writing, and then explain the features in the order that they appear in the image.

Note: It is important that the manual can be maintained long term, UI and tool options change so try to avoid having a lot of images (when they are not especially necessary). Otherwise, this becomes too much of a maintenance burden.

Videos

Videos from YouTube and Vimeo can be embedded using:

```
.. youtube:: ID

.. vimeo:: ID
```

The `ID` is found in the video's URL, e.g:

- The ID for `https://www.youtube.com/watch?v=Ge2Kwy5EGE0` is `Ge2Kwy5EGE0`
- The ID for `https://vimeo.com/15837189` is `15837189`

Usage Guides

- Avoid adding videos which rely on voice, as this is difficult to translate.
- Do not embed video tutorials as a means of explaining a feature, the writing itself should explain it adequately (though you may include a link to the video at the bottom of the page under the heading `Tutorials`).

Useful Constructs

- `|BLENDER_VERSION|` – Resolves to the current Blender version.
- `:abbr:`SSAO (Screen Space Ambient Occlusion)`` – Abbreviations display the full text as a tooltip for the reader.
- `:term:`Manifold`` – Links to an entry in the *Glossary*.

Cross References and Linkage

You can link to another document in the manual with:

```
:doc:`The Title </section/path/to/file>`
```

To link to a specific section in another document (or the same one), explicit labels are available:

```
.. _sample-label:

[section or image to reference]

Some text :ref:`Optional Title <sample-label>`
```

Linking to a title in the same file:

```
Titles are Targets
==================

Body text.

Implicit references, like `Titles are Targets`_
```

Linking to the outside world:

```
`Blender Website <https://www.blender.org>`__
```

Directory layout

Sections should be generally structured as follows:

- directory_name/
  - index.rst (contains links to internal files)
  - introduction.rst
  - section_1.rst
  - section_2.rst

For example:

- rendering/
  - index.rst
  - cycles/
    - index.rst
    - introduction.rst
    - materials/
      - index.rst

- `introduction.rst`
- `volumes.rst`

The idea is to enclose all the content of a section inside of a folder. Ideally every section should have an `index.rst` (containing the TOC for that section) and an `introduction.rst` (introducing) to the contents of the section.

Table of Contents

By default, a table of contents should show two levels of depth:

```
.. toctree::
   :maxdepth: 2

   introduction.rst
   perspective.rst
   depth_of_field.rst
```

Further Reading

To learn more about reStructuredText, see:

Sphinx RST Primer Good basic introduction.

Docutils reStructuredText reference Links to reference and user documentation.

Writing Style Guide

Primary Goals

The main goals for this manual are as follows.

User Focused While some areas of computer graphics are highly technical, this manual shall be kept understandable by non-technical users.

Complete So there is a canonical source of truth for each of Blender's key areas. This does not mean we have to document every small detail, but users should not have to rely on searching elsewhere to find the purpose of key features.

Concise Computer graphics is an incredibly interesting field, there are many rules, exceptions to the rules and interesting details. Expanding into details can add unnecessary content, so keep the text concise and relevant to the topic at hand.

Maintainable Keep in mind that Blender has frequent releases, so try to write content that will not have to be redone the moment some small change is made. This also helps a small documentation community to maintain the manual.

Content Guidelines

In order to maintain a consistent writing style within the manual, please keep this page in mind and only deviate from it when you have a good reason to do so.

Rules of thumb:

- Spell checking is *strongly* recommended.
- Use American English (e.g: modeling and not modelling, color and not colour).
- Take care about grammar, appropriate wording and use simple English.
- Keep sentences short and clear, resulting in text that is easy to read, objective and to the point.
- Including why or how an option might be useful is a good idea.

- If you are unsure about how a feature works, ask someone else or find out who developed it and ask them.

To be avoided:

- Avoid writing in first person perspective, about yourself or your own opinions.
- Avoid weasel words and being unnecessarily vague, e.g:

  "Reloading the file will probably fix the problem"
  "Most people do not use this option because ..."

- Avoid including specific details such as:

  "Blender has 23 different kinds of modifiers."
  "Enabling previews adds 65536 bytes to the size of each Blend file (unless it is compressed)."

  These details are not useful for users to memorize and they become quickly outdated.

- Avoid documenting bugs.

  Blender often has 100's of bugs fixed between releases, so it is not realistic to reference even a fraction of them from the manual, while keeping this list up to date.

  Issues which are known to the developers and are not going to be resolved before the next release can be documented as *Known Limitations*. In some cases, it may be best to include them in the *troubleshooting* section.

- Avoid Product Placements, i.e. unnecessarily promoting software or hardware brands. Keep content vendor-neutral where possible.
- Avoid technical explanations about the mathematical/algorithmic implementation of a feature if there is a simpler way to explain it.

  (e.g. explaining how mesh smoothing algorithms work is unnecessary, but the blending types of a mix node do need a mathematical explanation).

- Avoid repetition of large portions of text. Simply explain it once, and from then on refer to that explanation.

  For general terminology, consider defining a `:term:` in the *glossary*.

- Avoid enumerations of similar options, such as listing every preset or every frame-rate in a select menu.

  Their contents may be summarized or simply omitted. – Such lists are only showing what is already *obvious* in the interface and end up being a lot of text to read and maintain.

- Avoid documenting changes in Blender between releases, that is what the release notes are for. We only need to document the current state of Blender.
- Unless the unit a value is measured in is obscure and unpredictable, there is no need to mention it.
- Do not simply copy the tool-tips from Blender. – People will come to the manual to learn *more* than is provided by the UI.

  As a last resort you can add comment (which is not shown in the HTML page, but useful for other editors):

```
.. TODO, how does this tool work? ask Joe Blogg's.
```

Glossary

This section is specifically about the *Glossary* section, where we define common terms in Blender and computer-graphics.

Rules of thumb:

- Define the term before providing any further information.
- Avoid using constructs such as “It is” or “xyz is” before the definition.
- Avoid repeating the term immediately or using it in the definition.
- Avoid adding terms not found in Blender’s interface or the manual.
- Avoid overly long entries. If an explanation of a complex term is needed, supplement with external links.
- Avoid duplicating documentation; if explaining the term is the primary focus of another section of the manual (e.g. if the term is the name of a tool), either just link to that section, or avoid creating a glossary entry entirely.
- URL references are to be added at the end, formatted as follows, e.g:

```
See also `OpenGL <https://en.wikipedia.org/wiki/OpenGL>`__ on Wikipedia.
```

Examples

This entry:

```
Displacement Mapping
   Uses a grayscale heightmap, like Bump Mapping,
   but the image is used to physically move the vertices of the mesh at render time.
   This is of course only useful if the mesh has large amounts of vertices.
```

Would be written like this instead, putting a definition first:

```
Displacement Mapping
   A method for distorting vertices based on an image.
   Similar to Bump Mapping, but instead operates on the mesh's actual geometry.
   This relies on the mesh having enough geometry.
```

This entry:

```
Doppler Effect
   The Doppler effect is the change in pitch that occurs
   when a sound has a velocity relative to the listener.
```

Would be written more like this, avoiding the immediate repetition of the term:

```
Doppler Effect
   Perceived change in pitch that occurs
   when the source of a sound is moving relative to the listener.
```

This entry:

```
Curve
   It is a class of objects.
   In Blender there are Bézier curves and NURBS curves.
```

Would be written more like this, avoiding the “it is”:

```
Curve
   A type of object defined in terms of a line interpolated between Control Vertices.
   Available types of curves include Bézier and NURBS.
```

Translations

Contribute

On this page French (`fr`) is used for examples, however, it can be replaced with other languages codes.

To see which languages are currently available, you can browse the repository: https://developer.blender.org/diffusion/BMT/browse/trunk/blender_docs/locale

Installing

Dependencies

For translations, we use Sphinx's internationalization package. However, this is not included with Sphinx and needs to be installed:

```
pip install sphinx-intl
```

Your Language Repository

Note: First of all, it is assumed that you have the manual already building. If you have not done this already go back too the *Getting Started* section.

From the directory containing your checkout of the manual run the following command.

Note: Be sure to change the `/fr` suffixes to the language you are translating!

You can remove the suffix to check out all languages too, however, this will be a much larger download.

```
svn checkout https://svn.blender.org/svnroot/bf-manual-translations/trunk/blender_docs/
↪locale/fr locale/fr
```

This will create a `locale/fr` subdirectory.

You should have a directory layout like this:

```
blender_docs
   |- locale/
   |  |- fr/
   |  |  |- LC_MESSAGES/
   |- manual/
```

A PO Editor

To make edit the PO files you will need to install a PO editor. We recommended that you use Poedit however, any PO editor will do.

Note: For Linux users you will have to check with your distribution's software center for a version of Poedit.

Building with Translations

Note: This is optional, translations are automatically built online, eg: https://www.blender.org/manual/fr/

This is quite involved, so it is not be expected that translators should be doing their own builds locally.

To creates the `.mo` files (needed for building translation):

```
sphinx-intl build
```

Now you can build the manual with the translation applied:

```
make -e SPHINXOPTS="-D language='fr'"
```

If you are on MS-Windows and do not have `make`, run:

```
sphinx-build -b html -D language='fr' ./manual ./build/html
```

Now you will have a build of the manual with translations applied.

Editing Translation Files

Now you can edit the PO translation files, eg:

Original RST File `manual/getting_started/about_blender/introduction.rst`

Generated PO File `locale/fr/LC_MESSAGES/getting_started/about_blender/introduction.po`

The modified `.po` files can be edited and committed back to svn.

Maintenance

Keeping track of fuzzy strings

When the manual is updated, those translations which are outdated will be marked as fuzzy. To keep track with that, you can use a tool we created for that task.

You can do this by running:

```
make report_po_progress
```

This will only give a quick summery however, you can get more information by running:

```
python tools/report_translation_progress.py locale/fr/
```

You should get a list of all the files with information about the number of empty and fuzzy strings. For more options see:

```
python tools/report_translation_progress.py --help
```

See also:

- Instructions on this page are based on Sphinx Intl documentation
- The translation design task for discussion on the process.

Updating PO Files

As the original manual changes, the templates will need updating. Note, doing this is not required, as administrator usually update the files for all languages at once. This allows all languages to be on the same version of the manual. However, if you need to update the files yourself, it can be done as follows:

```
make update_po
```

The updated templates can then be committed to svn.

TODO: document how to handle files being added/removed/moved.

Style Guide

This page covers conventions concerning the translations.

Note:

- We expect our readers to use the English version of Blender, not a translated one.
- The translations are licensed under the same *License* as the original.

Should I translate... ?

Maybe

Hyperlinks Can be translated, but only as an addition, not as a replacement. See also *Adding Text*

Technical Terms Only translate these, when the localized expression is common! See also *Technical Terms*

Text you are not sure you understood Simply mark the text as fuzzy and/or add a comment. The next translator might understand it.

Never

Images You probably will not find the original scene if it is a screenshot of a file and it is too much load on the server (and too much work for you).

Menu and button names We expect our readers to use the English UI.

Text you do not understand Do not translate it! It will do more harm than good!

Technical Terms

In general, the technical terms used in CG are quite new or even downright neologisms invented for the needs, so they do not always have a translation in your language. Moreover, a large part of Blender users use its English interface.

As a result, unless a term has an evident translation, you should preferably use the English one, *putting it in italic*. You can then find a translation for it, which you will use from times to times (e.g. to avoid repetitions...). This is also valid in the other way: even when a term has a straightforward translation, do not hesitate to use its English version from times to times, to get the reader used with it...

If a term is definitively not translatable, simply use the English one, but make sure its manual entry is translated.

In the manual, the English term is written first (to maintain alphabetic order) with the translated entry following in parenthesis, when appropriate.

Adding Text

Generally, **you should always translate exactly what is in the text**, and avoid providing updates or extra information.

But sometimes that is necessary, for example when talking about the manual itself: To a foreign reader it is not clear, that he or she can contribute English text only, whereas this is obvious to an English reader.

In these (rare) cases you can and should provide extra information.

Keeping Pages Up To Date

When the manual is updated, those translations which are outdated will be marked as fuzzy. To keep track with that, you can use a tool we created for that task, see *How to install it*.

See also:

https://wiki.blender.org/index.php/Meta:Guides/Translation_Guide

Contacts

The Manual Teams project page.

- Mailing list is our main way of distributing documents, discussing ideas, and keeping track of progress.(Registration required).
- `#blenderwiki` channel in the *IRC chat* for informal discussions in real-time.
- Workboard for tasks.
- Patch tracker shared with the other Blender projects.