Blender Documentation

Last modified September 22 2003 S68

Florian Findeiss

Alex Heizer

Reevan McKay

Jason Oppel

Ton Roosendaal

Stefano Selleri

Bart Veldhuizen

Carsten Wartmann

Blender Documentation: Last modified September 22 2003 S68

by Florian Findeiss, Alex Heizer, Reevan McKay, Jason Oppel, Ton Roosendaal, Stefano Selleri, Bart Veldhuizen, and Carsten Wartmann

This is a first working document for the Blender Documentation project. Feel free to add or modify your changes and send them clearly marked to the Blender documentation board (bf-docboard@blender.org).

Table of Contents

Chapter 1. Introduction

About this manual (x)

This manual is the result of a joint effort of Blender users around the world. Since we have only just started there is not a lot to find here, but the table of contents should give you an idea of what to expect.

This manual is published under two 'Open' licenses (see Appendix I). The most recent version can always be found on http://download.blender.org/documentation.

If you have a suggestion for us, if you would like to help, or if you already have a piece of text that you think could be added to the Manual, please pay a visit to the home of our mail list[2], and drop us a line.

We tried to keep a complete track of what is ready and what is not, but it was too big an effort. Energy better used in writing, so the following conventions are used:

- Titles ending with (-) : Still empty or badly outdated
- Titles ending with (x) : outdated, pending revision

What is Blender?

Blender is a suite of tools enabling the creation of and replay of linear and real-time, interactive 3D content. It offers full functionality for modeling, rendering, animation, post-production and game creation and playback with the singular benefits of cross-platform operability and a download file size of less than 2.5MB.

Aimed at media professionals and individual creative users, Blender can be used to create commercials and other broadcast quality linear content, while the incorporation of a real-time 3D engine allows for the creation of 3D interactive content for stand-alone playback or integration in a web browser.

Originally developed by the company 'Not a Number' (NaN), Blender now is continued as 'Free Software', with the sources available under GNU GPL.

Key Features:

- Fully integrated creation suite, offering a broad range of essential tools for the creation of 3D content, including modeling, animation, rendering, video post production and game creation
- Small executable size, for easy distribution
- High quality 3D architecture enabling fast and efficient creation work-flow
- Free support channels via www.blender3d.org
- 250k+ worldwide user community

You can download the latest version of Blender at download.blender.org.

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 multi-national 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 tool set for NeoGeo was too old and cumbersome to maintain and upgrade and needed to be rewritten from scratch. In 1995 this rewrite began and was destined to become the 3D software creation suite we all now know and love 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 creation suite for free. At the time this was a revolutionary concept as most commercial modelers cost several thousands of (US) 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 1999 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!

On the wings of a successful Siggraph in early 2000, NaN secured financing of 4.5 million EUR 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 v2.0 was released. This version of Blender added the integration of a game engine to the 3D suite. By the end of 2000, the number of users registered on the NaN website surpassed 250,000.

Unfortunately, NaN's ambitions and opportunities didn't match the company's capabilities and the market realities of the time. This overextension 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 current version of Blender, with a complex internal software architecture, unfinished features and a non-standard way of providing the GUI, enthusiastic support from the user community and customers who had purchased Blender Publisher in the past, Ton couldn't justify leaving Blender to disappear into oblivion. Since restarting a company with a sufficiently large team of developers wasn't feasible, in March 2002 Ton Roosendaal founded the non-profit organization *Blender Foundation*.

The Blender Foundation's primary goal was to find a way to continue developing and promoting Blender as a community-based Open Source[3] project. In July 2002, Ton managed to get the NaN investors to agree to a unique Blender Foundation plan to attempt to Blender as open source. The "Free Blender" campaign sought to raise 100,000 EUR 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 fund raising campaign was launched to "Free Blender." To everyone's surprise and delight the campaign reached the 100,000 EUR goal in only seven short weeks. On Sunday October 13, 2002, Blender was released to the world under the under the terms of the GNU General Public License (GPL). Blender development continues to this day driven by a team of far-flung, dedicated volunteers from around the world led by Blender's original creator, Ton Roosendaal.

Blender's history and road-map

- 1.00 Jan 1996 Blender in development at animation studio NeoGeo
- 1.23 Jan 1998 SGI version published on the web, IrisGL

- 1.30 April 1998 Linux and FreeBSD version, port to OpenGL and X
- 1.3x June 1998 NaN founded
- 1.4x Sept 1998 Sun and Linux Alpha version released
- 1.50 Nov 1998 First Manual published
- 1.60 April 1999 C-key (new features behind a lock, $95), 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 Aug 2000 Interactive 3D and real-time engine
- 2.10 Dec 2000 New engine, physics and Python
- 2.20 Aug 2001 Character animation system
- 2.21 Oct 2001 Blender Publisher launch
- 2.2x Dec 2001 Mac OSX version

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 true in most cases, 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"). Free software in this sense is software which you (the user) 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; in addition, since the source code is available universally, there are 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, copy, and distribute the program;
- you have the right to modify the program;
- you have the right to a copy of the source code.

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 the recipient is aware of his 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 under the GPL and make the source code of your changes available. (You may not use GPL'd code as part of a proprietary program.)
- 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 the GNU Project Web site[4]. For reference, a copy of the GNU General Public License is included in Appendix I.

Getting support - the Blender community

Being Blender free, even if closed source, from start helped a lot in its diffusion, and a wide, stable, active community of users gathered around it from very early.

The community showed its best in the crucial moment of freeing Blender itself and letting it go Open Source under GNU GPL later summer 2002.

The community itself is now subdivided into two, widely overlapping sets:

1. The Developer Community, Centered around Blender Foundation site http://www.blender.org/. Here is the home of the Functionality and Documentation Boards, the CVS repository of Blender sources and documentation sources and related Forum of Discussion. Coders hacking on Blender itself, Python scripters, Doc writers and anyone working for Blender development in general hangs hereby.
2. The User Community, centered around the independent site http://www.elysiun.com/. Here Blender artists, Blender gamemakers and any Blender fan gathers to show their productions, get feedback, ask help to get better insight in Blender functionalities.

But, let me repeat it, it's just a single Blender Community.

Another relevant source of informations lies in Blender Knowledge Base, a fully searchable database of questions and aswers located at http://www.vrotvrot.com/support

For immediate online Blender feedback there are three chatboxes permanently opened on irc.freenode.net. You can join these with your favorite IRC client (I'm not going to promote any).

Chatboxes are **#blenderchat**, **#blenderqa** and **#gameblender**. The first of these is accessible even without a IRC client but with a plain Java enabled Web Browser through elYsiun site (http://www.elysiun.com/).

Notes

1. http://download.blender.org/documentation
2. http://www.blender.org/mailman/listinfo/bf-docboard
3. http://www.opensource.org/
4. http://www.gnu.org
5. http://www.blender.org/
6. http://www.elysiun.com/
7. http://www.vrotvrot.com/support
8. http://www.elysiun.com/

Chapter 2. Installation (x)

Blender is available both as binary executables and as source code on the Foundation site (http://www.blender.org/). From the main page look for the 'Download' section.

Downloading and installing the binary distribution

The Binary distributions comes in 6 basic flavours:

- Windows
- Linux
- MacOSX
- FreeBSD
- Irix
- Solaris

The Linux flavour comes actually in 4 different sub-flavours, for Intel and PowerPC architectures, with statically linked libraries or for dynamic loading libraries.

The difference between the dynamic and the static flavour is important. The static build has the OpenGL libraries compiled in. This makes Blender running at your system without using hardware accelerated graphics. Use the static version to check if Blender runs fine when the dynamic version fails! OpenGL is used in Blender for all drawing, including menus and buttons. This dependency makes a proper and compliant OpenGL installation at your system a requirement. Not all 3D card manufacturors provide such compliancy, especially cheaper cards aimed at the gaming market.

Of course since renderings are made by Blender rendering engine in core memory and by the main CPU of your machine a graphic card with hardware acceleration makes no difference at rendering time.

Windows

Quick Install

Download the file `blender-#.##-windows.exe`, being `#.##` the version number, from the downloads section of the Blender Website. Start the installation by double-clicking the file. This presents you with some questions, for which the defaults should be ok. After setup is complete, you can start Blender right away, or use the entry in the Start menu.

In-depth Instructions

Download the file `blender-#.##-windows.exe` from the downloads section of the Blender Website. Choose to download it (if prompted), select a location and click "Save". Then navigate with explorer to the location you saved the file in and double-click it to start the installation.

The first dialog presents you the license. You are expected to accept it if you want the installation to go any further. After accepting the license,select the components you wish to install (there is just one, Blender) and the additional actions you want to take. There are three: Add a shortcut to the Stat menu, Add Blender's icon to desktop, associate .blend files with Blender. By default it is all checked. If you don't want some action to be taken simply uncheck it. When done, click on `Next`.

Select a place to install the files to (the default should do well), and click `Next` to install Blender. Press `Close` when installation is over.

Afterwards you will be asked whether you want to start Blender immediately. Blender is now installed and can be started by means of the Start menu (an entry named "Blender Foundation" has been created by the setup routine) or by double-clicking a Blender file (`*.blend`).

OSX

THIS MUST BE REWRITTEN FOR 2.28 BY SOMEONE WITH A MAC ACCORDINGLY TO THE WINDOWS STYLE

Quick Install

Download the file `blender-publisher-2.25-mac-osx-10.1.zip` from the downloads section of the Blender Website. Extract it by double-clicking the file, if it does not automatically extract as it downloads. Open the folder `blender-publisher-2.25-mac-osx-10.1` and double-click the `blenderpublisher` icon to start it. Drag the `blenderpublisher` icon to the Dock to make an alias there.

In-depth Instructions

Blender Publisher is available from the Blender Web site (http://www.blender.org/) in source form, and as a binary for Mac OSX. Unless you have problems running the binary, you will not need to download and compile the sources. From the downloads page, choose the "NaN Blender Publisher 2.25" link. Next, select the "Blender executables" link. You will not need a Publisher Key file for OS X.

Download the file `blender-publisher-2.25-mac-osx-10.1.zip` from the downloads section of the Blender Website. If you use Internet Explorer, the file will download and be automatically extracted with Stuffit(R) (http://www.stuffit.com/) to a folder on your Desktop named `blender-publisher-2.25-mac-osx-10.1`. If you use Netscape, you will be prompted to choose whether to download the file or have it automatically extracted with Stuffit(R). If you choose to have Stuffit(R) extract it, it will be extracted to a folder on your Desktop named `blender-publisher-2.25-mac-osx-10.1`. If you choose to download it, select a location and click "Save". Then navigate to the location you saved the file in and double-click it to have Stuffit(R) open it in that location. It will extract the files to a folder named `blender-publisher-2.25-mac-osx-10.1`.

Open the `blender-publisher-2.25-mac-osx-10.1` folder, and double-click the `blenderpublisher` icon to run Blender. You can also open your hard drive (the Macintosh HD icon on your Desktop), and open your Applications, then drag the `blenderpublisher` icon from the original folder to the Applications folder. If you wish to leave the original `blenderpublisher` file and make a copy in the Applications folder, hold the option key while dragging. If you wish to make an alias to the original `blenderpublisher` program, hold both the option and command keys while dragging the icon.

You can also place the binary, a copy of the binary or an alias to the binary on your Desktop instead of the Applications folder, or put an alias on your Dock simply by dragging the program icon down to the Dock.

Linux

Quick Install

Download the file `blender-#.##-linux-glibc#.#.#-ARCH.tar.gz` from the downloads section of the Blender Website. Here `#.##` is Blender version, `#.#.#` is glibc version and `ARCH` is the machine architecture, either `i386` or `powerpc`. You should get the one matching your system, remember the choiche between static and dynamic builds.

Unpack the archive to a location of your choice. This will create a directory named `blender-#.##-linux-glibc#.#.#-ARCH`, in which you will find the **blender** binary.

To start blender just open a shell and execute **./blender**, of course when running X.

In-depth Instructions

Download the file `blender-#.##-linux-glibc#.#.#-ARCH.tar.gz` from the downloads section of the Blender Website. Choose to download it (if prompted), select a location and click "Save". Then navigate to the location you wish blender to install to (e.g. `/usr/local/`) and unpack the archive (with **tar xzf /path/to/blender-#.##-linux-glibc#.#.#-ARCH.tar.gz**). If you like, you can rename the resulting directory from `blender-#.##-linux-glibc#.#.#-ARCH` to something shorter, e.g. just `blender`.

Blender is now installed and can be started on the command line by entering **cd /path/to/blenderpublisher** followed by pressing the enter key in a shell. If you are using KDE or Gnome you can start Blender using your filemanager of choice by navigating to the Blender executable and (double-) clicking on it.

If you are using the Sawfish window manager, you might want to add a line like `("Blender" (system "blender &"))` to your `.sawfish/rc` file.

To add program icons for Blender in KDE

1. Select the "Menu Editor" from the System submenu of the K menu.
2. Select the submenu labeled "Graphics" in the menu list.
3. Click the "New Item" button. A dialog box will appear that prompts you to create a name. Create and type in a suitable name and click "OK". "Blender" or "Blender #.##" would be logical choices, but this does not affect the functionality of the program.
4. You will be returned to the menu list, and the Graphics submenu will expand, with your new entry highlighted. In the right section, make sure the following fields are filled in: "Name", "Comment", "Command", "Type" and "Work Path".

   - The "Name" field should already be filled in, but you can change it here at any time.
   - Fill the "Comment" field. This is where you define the tag that appears when you roll over the icon.
   - Click the folder icon at the end of the "Command" field to browse to the blenderpublisher program icon. Select the program icon and click "OK" to return to the Menu Editor.
   - The "Type" should be "Application".
   - The "Work Path" should be the same as the "Command", with the program name left off. For example, if the "Command" field reads `/home/user/blender-publisher-#.##-linux-glibc#.#.#-ARCH/blender`, the "Work Path" would be `/home/user/blender-publisher-#.##-linux-glibc#.#.#-ARCH/`.

5. Click "Apply" and close out of the Menu Editor.

To add a link to Blender on the KPanel, right-click on a blank spot on the KPanel, then hover over "Add", then "Button", then "Graphics", and select "Blender" (or whatever you named the menu item in step 3). Alternately, you can navigate through the "Configure Panel" submenu from the K menu, to "Add", "Button", "Graphics", "Blender".

To add a Desktop icon for Blender, open Konquerer (found on the Panel by default, or in the "System" submenu of the K menu) and navigate to the blenderpublisher program icon where you first unzipped it. Click and hold the program icon, and drag it from Konquerer to a blank spot on your Desktop. You will be prompted to Copy Here, Move Here or Link Here, choose Link Here.

To add program icons for Blender in GNOME

1. Select "Edit menus" from the Panel submenu of the GNOME menu.
2. Select the "Graphics" submenu, and click the "New Item" button.
3. In the right pane, fill in the "Name:", "Comment:" and "Command:" fields. Fill the "Name:" field with the program name, for example "Blender". You can name this whatever you'd like, this is what appears in the menu, but does not affect the functionality of the program. Fill the "Comment:" field with a descriptive comment. This is what is shown on the tooltips popups. Fill the "Command:" field with the full path of the blenderpublisher program item, for example, `/home/user/blender-publisher-#.##-linux-glibc#.#.#-ARCH/blender`
4. Click the "No Icon" button to choose an icon. There may or may not be an icon for Blender in your default location. You can make one, or look for the icon that goes with KDE. This should be `/opt/kde/share/icons/hicolor/48x48/apps/blender.png`. If your installation directory is different, you can search for it using this command in a Terminal or Console: **find / -name "blender.png" -print**
5. Click the "Save" button and close the Menu Editor.

To add a Panel icon, right-click a blank area of the Panel, then select "Programs", then "Graphics", then "Blender". Alternatively, you could click the GNOME menu, then select "Panel", then "Add to panel", then "Launcher from menu", then "Graphics", and "Blender".

To add a Desktop icon for Blender, open Nautilus (double-click the Home icon in the upper-left corner of your Desktop, or click the GNOME menu, then "Programs", then "Applications", and "Nautilus"). Navigate to the folder which contains the blenderpublisher program icon. Right-click the icon, and drag it to the Desktop. A menu will appear asking to Copy Here, Move Here, Link Here or Cancel. Select Link Here.

FreeBSD

Quick Install

Download the file `blender-#.##-freebsd-#.#-i386.tar.gz` from the downloads section of the Blender Website. Here `#.##` is Blender version, `#.#` is FreeBSD version and `i386` is the machine architecture.

TBW

In-depth Instructions

TBW

Irix

Quick Install

Download the file `blender-#.##-irix-#.#-mips.tar.gz` from the downloads section of the Blender Website. Here `#.##` is Blender version, `#.#` is Irix version and `mips` is the machine architecture.

TBW

In-depth Instructions

TBW

Solaris

Quick Install

Download the file `blender-#.##-freebsd-#.#-sparc.tar.gz` from the downloads section of the Blender Website. Here `#.##` is Blender version, `#.#` is Solaris version and `sparc` is the machine architecture.

TBW

In-depth Instructions

TBW

(others)

(to be written)

Building Blender from the sources (x)

Blender is available as source code on the Foundation site (http://www.blender.org/). From the main page look for the 'Download' section and then for 'Source Code'.

There is just one source, which is then customized depending on the architecture. THe source code is released compressed, either via gzip or via bzip2.

The source is hence available as `blender-#.##.tar.gz` or as `blender-#.##.tar.bz2`, being `#.##` Blender's version.

Blender is developed exploiting the Concurrent Version Server (CVS) system. Sources cab hence be downloaded via CVS access too.

The CVS server also provides daily checkouts, available from the source download page, as well as as the standard cvs checkout mechanism: **cvs -z3 -d:pserver:anonymous@cvs.blender.org:/cvsroot/bf-blender co blender**

Once you got the sources, the adventure of building begins.

Introduction (-)

(to be written)

Windows (-)

(to be written)

Linux (-)

(to be written)

(others) (-)

(to be written)

Notes

1. http://www.blender.org/
2. http://www.blender.org/
3. http://www.stuffit.com/
4. http://www.blender.org/

Chapter 3. Understanding the interface

By Martin Kleppmann

If you are new to Blender, it is important to get a good grip of the user interface before starting on modelling. This is because the concepts are quite non-standard - it differs from other 3D software packages, and Windows users especially will need to get used to the different handling of controls. But this interface concept is in fact one of Blender's great strengths: once you have found out how it works, it enables you to work exceedingly quickly and productively.

Blender's interface concept

The interface is the two way means of interaction between the user and the code. The user communicates with the code via the keyboard and the mouse, the code gives feedback via the screen and its windowing system.

Keyboard and mouse

Blender's interface makes use of three mouse buttons and a wide range of hotkeys (for a quick, compact list see Appendix A, for a complete in-depth discussion refer to Part IX in *Blender Documentation*). If your mouse has only two buttons, you can activate an emulation of the middle mouse button (the Section called *User preferences* describes how to do this). A mouse wheel can be used, but it is not necessary, as there are also appropriate keyboard shortcuts.

This manual uses the following conventions to describe user input:

- The mouse buttons are called **LMB** (left mouse button), **MMB** (middle mouse button) and **RMB** (right mouse button).
- If your mouse has a wheel, **MMB** refers to clicking the wheel as if it were a button, while **MW** means rolling the wheel.
- Hotkey letters are named by appending *KEY* to the letter, e.g. **GKEY** refers to the letter *g* on the keyboard. Keys may be combined with the modifiers **SHIFT**, **CTRL** and/or **ALT**. For modified keys the **KEY** suffix is generally dropped e.g. **CTRL-W** or **SHIFT-ALT-A**.
- **NUM0** to **NUM9**, **NUM+** etc. refer to the keys on the separate numeric keypad. NumLock should generally be switched on.
- Other keys are referred to by their names, e.g. **ESC**, **TAB**, **F1** to **F12**.
- Other special keys of note are the arrow keys, **UPARROW**, **DOWNARROW** etc.

Because Blender makes such extensive use of both mouse and keyboard, a "golden rule" evolved amongst users: have one hand on the mouse and the other on the keyboard! If you normally use a keyboard that is significantly different from the English layout, you may want to think about changing to the English or American layout for the time you work with Blender. The most frequently used keys are grouped so that they are reachable by the left hand in standard position (index finger on **FKEY**) on the English keyboard layout. This assumes that you use the mouse with your right hand.

The window system

Now it's time to start Blender and begin playing around.

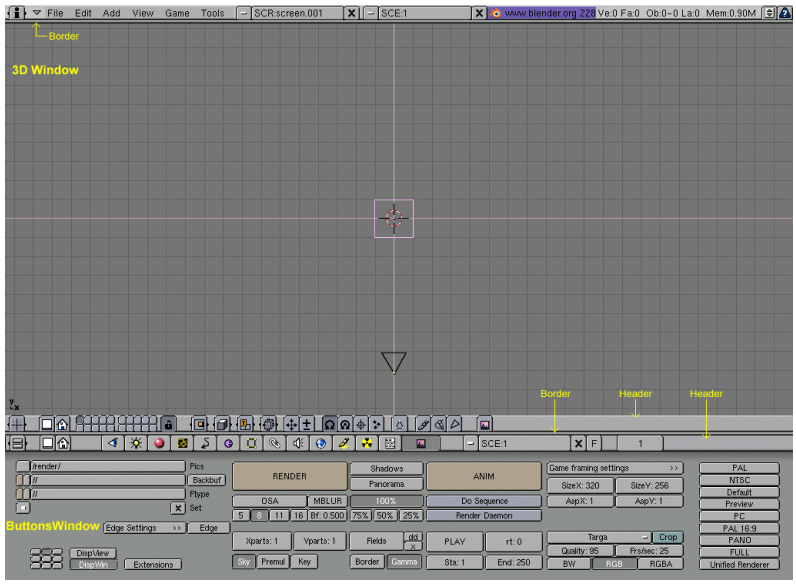


Figure 3-1. The default Blender scene.

Figure 3-1 shows the screen you should get after starting Blender (except for the yellow text and arrows). At default it is separated into three windows: the main menu at the top, the large 3D Window and the Buttons Window at the bottom. Most windows have a header (the strip with a lighter grey background containing icon buttons - for this reason we will also refer to the header as the window *ToolBar*); if present, the header may be located at the top (as with the Buttons window) or the bottom (as with the 3D Window) of a window's area.

If you move the mouse over a window, note that its header changes to a lighter shade of grey. This means that it is "focused", i.e. all hotkeys you press will now affect the contents of this window.

The window system is easily customizable to your needs and wishes. To create a new window, you can split an existing one in half. Do this by focusing the window you want to split (move the mouse into it), clicking the border with **MMB** or **RMB**, and selecting "Split Area" (Figure 3-2). You can now set the new border's position by clicking with **LMB**, and cancel by pressing **ESC**. The new window will start as a clone copy of the window you split, but can then be set to do new things, like displaying the scene from a different perspective.

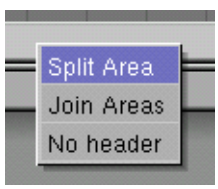


Figure 3-2. The Split menu for creating new windows.

Create a new vertical border by choosing "Split Area" from a horizontal border, and vice versa. You can resize each window by dragging a border with **LMB**. To reduce the number of windows, click a border between two windows with **MMB** or **RMB**

and choose "Join Areas". The resulting window receives the properties of the previously focused window.

You can set a header's position by clicking **RMB** on the header and choosing "Top" or "Bottom". It is also possible to hide the header by choosing "No Header", but this is only advisable if you know all the relevant hotkeys. A hidden header can be shown again by clicking the window's border with **MMB** or **RMB** and selecting "Add Header".

Window types

Each window frame may contain different types and sets of information, depending what you are working on: 3D models, animation, surface materials, Python scripts etc. The type for each window can be selected by clicking its header's leftmost button with **LMB** (Figure 3-3).

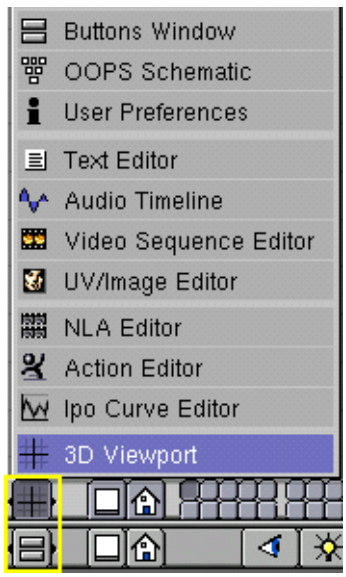


Figure 3-3. The window type selection menu.

The functions and usage of the respective window types will be explained at the relevant places in this manual. For now we only need the three window types that are already provided in Blender's default scene:

3D viewport

Provides a graphical view into the scene you are working on. You can view from any angle with a variety of options; see the Section called *Navigating in 3D space* for details. Having several 3D viewports on the same screen can be useful to watch your changes from different perspectives at the same time.

Buttons window

Contains most tools for editing objects, surfaces, textures, lights and much more. You will constantly need this window if you don't know absolutely all hotkeys off by heart, but having it twice would be useless.

User preferences (Main menu)

This window is usually hidden, so that only the menu part is visible - see the Section called *User preferences* for details. Compared to other software packages, this menu need hardly be used, though.

A feature of windows that sometimes comes in handy for precise editing is maximizing to full screen: if you press the appropriate button in the header (the second from the left in Figure 3-3) or the hotkey **CTRL-DOWNARROW**, the focused window will extend to fill the whole screen. To return to normal size, press the button again or **CTRL-UPARROW**.

Button types

Blender's buttons are more exciting than those in most other user interfaces. This is largely due to the fact that they are vectorial and drawn in OpenGL, hence elegant and zoomable.

There are several kind of buttons:

Operation Button

These are buttons that perform an operation when they are clicked (with **LMB**, as all buttons). They can be identified by their brownish colour (Figure 3-4).

Figure 3-4. An operation button

Operation Button

Toggle buttons come in various sizes and colours (Figure 3-5). The colours green, violet and grey do notchange functionality, they just help the eye to group them and recognize contents of the interface quicker. Clicking this type of button does not perform any operation, but only toggles a state as "on" or "off".

Some buttons also have a third state that is identified by the text turning yellow (the Ref button in Figure 3-5). Usually the third state means "negative", and the normal "on" state means "positive".

Figure 3-5. Toggle buttons

Radio Buttons

Radio buttons are particular groups of mutually exclusive Toggle buttons. No more than a single Radio Button of a given group can be "on" at a time.

Num Buttons

Number buttons (Figure 3-7) can be identified in that their caption contains a colon followed bya number. Some number buttons also contain a slider. Number buttons have several ways of handling: To increase the value, click **LMB** on the right half of the button, to decrease it, on the left half. To change the value in a wider range, hold down **LMB** and drag the mouse to the left or right. If you hold **CTRL** while doing this, the value is changed in discrete steps; if you hold **SHIFT**, the control is finer.

Figure 3-6. Number buttons

You can enter a value via the keyboard by holding **SHIFT** and clicking **LMB**. Press **SHIFT-BACKSPACE** to clear the value, **SHIFT-LEFTARROW** to move the cursor to the beginning and **SHIFT-RIGHTARROW** to move the cursor to the end. By pressing **ESC** you can restore the original value.

Menu Buttons

Menu buttons are used to choose from dinamically created list. Their principaluse is to link datablocks to each other. (Datablocks are structures like Meshes, Objects, Materials, Textures etc.; by linking a Material to an Object, you assign it.) An example for such a block of buttons is shown in Figure 3-7. The first button (with the dash) opens a menu that lets you select the datablock to link to by holding down **LMB** and releasing it over the requested item. The second button displays the type and name of the linked datablock and lets you edit the name it after clicking **LMB**. The "X" button clears the link; the "car" button generates an automatic name for the datablock and the "F" button specifies whether the datablock should be saved in the file even if it is unused.

> **Unlinked objects:** Unlinked data is *not* lost untill you quit Blender. This is a powerfull Undo feature. if you delete an object the material assigned to it becomes unlinked,but is still there! You just have to re-link it to another object or press the "F" button.

Figure 3-7. Datablock link buttons

Screens

Blender's flexibility with windows lets you create customized working environments for different tasks e.g. modelling, animating and scripting. It is often useful to quickly switch between different environments within the same file. This is possible by creating several screens: All changes to windows as described in the Section called *The window system* and the Section called *Window types* are saved within one screen, so

if you change your windows in one screen, other screens won't be affected. But the scene you are working on stays the same in all screens.

Three different default screens are provided with Blender; they are available via the SCR link buttons in the menu/preferences window shown in Figure 3-8. To change to the alphabetically next screen, press **CTRL-RIGHTARROW**; to change to the alphabetically previous screen, press **CTRL-LEFTARROW**.

Figure 3-8. Screen selector

Scenes

It is also possible to have several scenes within the same Blender file; they may use one another's objects or be completely separate from another. Scenes can be selected and created with the SCE link buttons in the menu/preferences window (Figure 3-9).

Figure 3-9. Scene selector

When you create a new scene, you can choose between four options about its contents:

- `Empty` creates an empty scene (surprise).
- `Link Objects` creates the new scene with the same contents as the currently selected scene. Changes in one scene will also modify the other.
- `Link ObData` creates the new scene based on the currently selected scene, with links to the same meshes, materials etc. This means you can change objects' positions and related properties, but modifications on the meshes, materials etc. will also affect other scenes unless you manually make single-user copies.
- `Full Copy` does what it says and creates a fully independent scene with copies of the currently selected scene's contents.

Navigating in 3D space

Blender lets you work in three-dimensional space, but the screen is only two-dimensional. To be able to work in three dimensions, you need to be able to change your viewing point and the viewing direction of the scene. This is possible in all of the 3D Viewports.

Most non-3D windows use an equivalent handling, as appropriate. It is even possible to translate and zoom a Buttons Window.

The viewing direction (rotating)

Blender provides three default viewing directions: from the side, the front and the top. As Blender uses a right-hand 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. You can select the viewing direction for a 3D Viewport with the view button (Figure 3-10) or by pressing the hotkeys **NUM3** for "side", **NUM1** for "front" and **NUM7** for "top".

> **Hotkeys:** Remember that most hotkeys affect the focused window, so check that the mouse cursor is in the area you want to work in first!

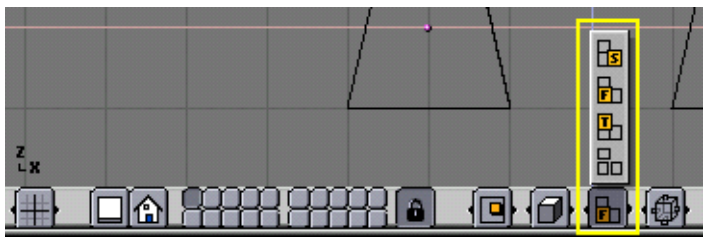


Figure 3-10. A 3D Viewport's view button.

Apart from these three default directions, the view can be rotated to any angle you wish. Drag **MMB** on the Viewport's area: if you start in the middle of the window and move up and down or left and right, the view is rotated around the middle of the window. If you start the edge and don't move towards the middle, you can rotate around your viewing axis (turn head-over). Just play around with this function until you get the feeling for it.

To change the viewing angle in discrete steps, use **NUM8** and **NUM2**, corresponding to vertical **MMB** dragging, or **NUM4** and **NUM6**, corresponding to horizontal **MMB** dragging.

Translating and zooming the view

To translate the view, hold down **SHIFT** and drag **MMB** in the 3D Viewport. You can also click the translate button in the Viewport's header (Figure 3-11 left) with **LMB**, drag it into the window area and translate the view until you let go **LMB**. For discrete steps, use the hotkeys **CTRL-NUM8**, **CTRL-NUM2**, **CTRL-NUM4** and **CTRL-NUM6** as for rotating.

Figure 3-11. View translation and zoom buttons.

You can zoom in and out by holding down **CTRL** and dragging **MMB** or using the zoom button (Figure 3-11 right) analogously. The hotkeys are **NUM+** and **NUM-**.

Wheel Mouse: If you have a wheel mouse, then all the actions you can do with **NUM+** and **NUM-** can also be done by rotating the wheel (**MW**). The direction of rotation selects the action.

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. **NUM.** zooms the view to the currently selected objects.

Perspective and orthonormal projection

Each 3D Viewport supports two different types of projection. These are demonstrated in Figure 3-12: orthonormal (left) and perspective (right).

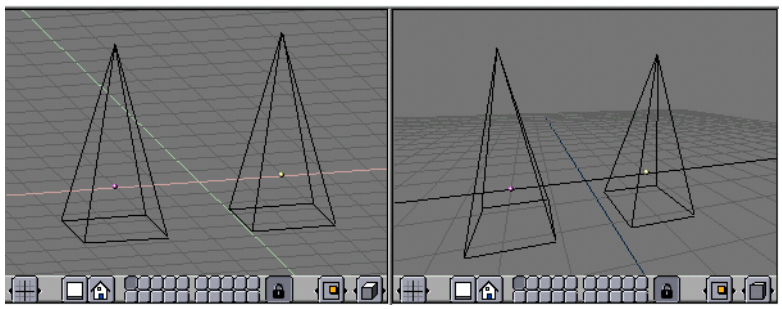


Figure 3-12. Demonstration of orthonormal (left) and perspective (right) projection.

Perspective viewing is what our eye is used to, because distant objects appear smaller. Orthonormal projection often seems a bit odd at first, because objects stay the same size independent of their distance: it is like viewing the scene from an infinitely distant point. Nevertheless, orthonormal 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.

To change the projection for a 3D Viewport, choose the lower or the middle item from the view mode button in the Viewport's header (Figure 3-13). The hotkey **NUM5** toggles between the two modes.

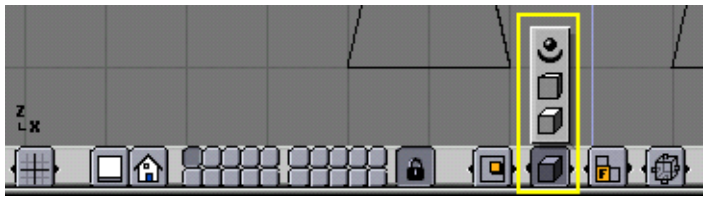


Figure 3-13. A 3D Viewport's projection mode button.

> **Camera projection:** Note that changing the projection for a 3D Viewport does not affect the way the scene will be rendered. Rendering is in perspective by default. If for any reason you need an Orthonormal rendering, select the camera and press "Ortho" in the EditButtons (**F9**).

The upper item of the view mode button sets the 3D Viewport to camera mode. (Hotkey: **NUM0**) The scene is then displayed as it will be rendered later (see Figure 3-14): the rendered image will contain everything within the outer dotted line. Zooming in and out is possible in this view, but to change the viewpoint, you have to move or rotate the camera.

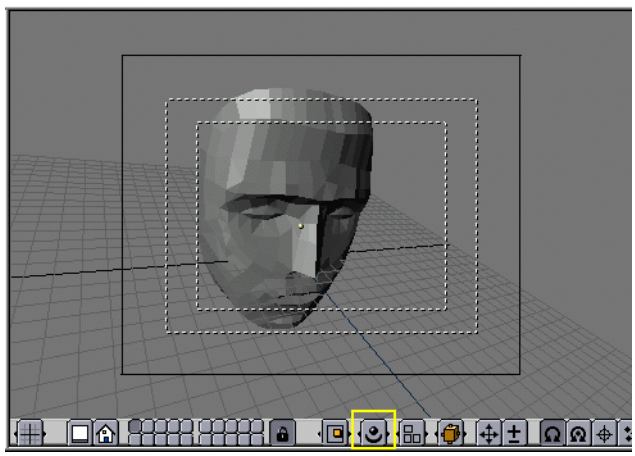


Figure 3-14. Demonstration of camera view.

Draw mode

Depending on the speed of your computer, the complexity of your scene and the type of work you are currently doing, you can switch between several drawing modes:

- Textured: Attempts to draw everything as completely as possible. Still no alternative to rendering. Note that if you have no lighting in your scene, everything will remain black.
- Shaded: Draws solid surfaces including the lighting calculation. As with textured drawing, you won't see anything without lights.

- Solid: Surfaces are drawn as solids, but the display also works without lights.
- Wireframe: Objects only consist of lines that make their shapes recognizable. This is the default drawing mode.
- Bounding box: Objects aren't drawn at all, instead only the rectangular boxes that correspond to each object's size and shape.

The drawing mode can be selected with the appropriate button in the header (Figure 3-15) or with hotkeys: **ZKEY** toggles between wireframe and solid display, **SHIFT-Z** toggles between wireframe and shaded display.

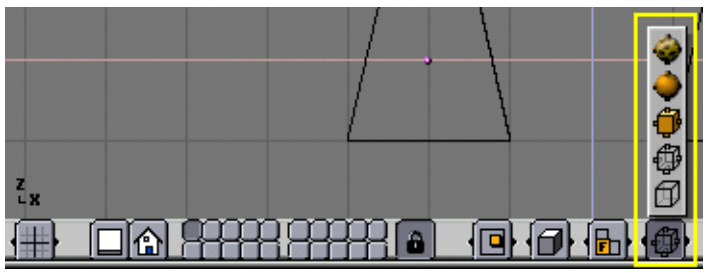


Figure 3-15. A 3D Viewport's draw mode button.

Local view

When in local view, only the selected objects are displayed. This can make editing easier in complex scenes. To enter local view, first select the objects you want (see the Section called *Selecting objects* in Chapter 5) and then select the upper item from the local view button in the header (Figure 3-16). The hotkey is **NUM/**.

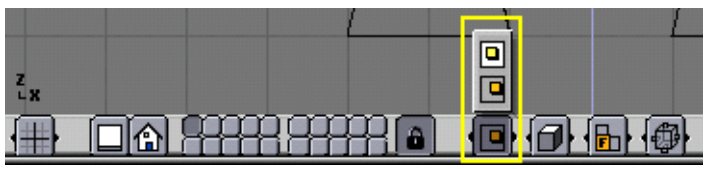


Figure 3-16. A 3D Viewport's local view button.

The layer system

3D scenes often get exponentially more confusing with growing complexity. To get this in control, objects can be grouped into "layers", so that only selected layers are displayed at a time. 3D layers are different from the layers you may know from 2D graphics applications: they have no influence on the drawing order and are (except for some special functions) solely for the modeller's better overview.

Blender provides 20 layers; you can choose which are to be displayed with the small unlabelled buttons in the header (Figure 3-17). To select only one layer, click the appropriate button with **LMB**; to select more than one, hold **Shift** while clicking.

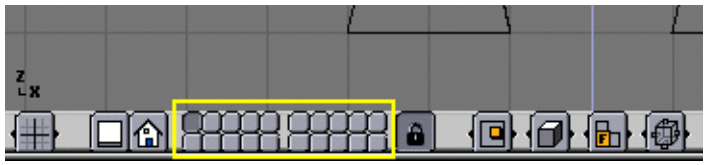


Figure 3-17. A 3D Viewport's layer buttons.

To select layers via the keyboard, press **1KEY** to **0KEY** (on the main area of the keyboard) for layers 1 to 10 (the top row of buttons), and **ALT-1** to **ALT-0** for layers 11 to 20 (the bottom row). The **SHIFT** key for multiple selection works like with clicking the buttons.

By default, the lock button directly right of the layer buttons is pressed; this means that changes to the viewed layers affect all 3D Viewports. If you want to select certain layers only in one window, deselect locking first.

To move selected objects to a different layer, press **MKEY**, select the layer you want from the pop-up dialog and press the Ok button.

The vital functions

Loading files

Blender uses the `.blend` file format to save nearly everything: Objects, scenes, textures, and even all your user interface window settings.

To load a Blender file from disk, press **F1**. The focused window then temporarily transforms into the file load dialog shown in Figure 3-18. The bar on the left can be dragged with **LMB** for scrolling. To load a file, select it with **LMB** and press **ENTER**, or simply click it with **MMB**.

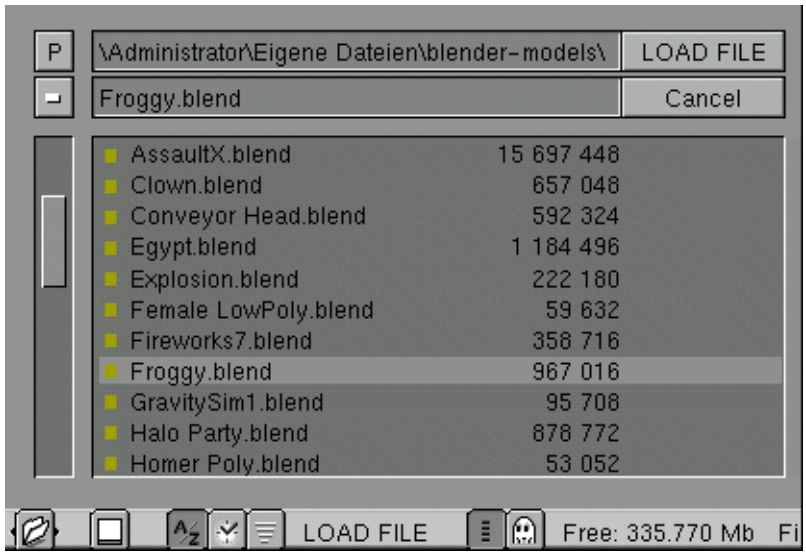


Figure 3-18. File load dialog.

The upper text box displays the current directory path, and the lower contains the selected filename. The P button (**PKEY**) moves you up to the parent directory; the button with the dash maintains a list of recently used paths. On Windows operating systems, the latter also contains a list of all drives (C:, D: etc).

Please note that Blender expects that you know what you are doing! When you load a file, you are not asked about unsaved changes to the scene you were previously working on: completing the file load dialog is regarded as being enough confirmation that you didn't do this by accident. Make sure for yourself that you save your files.

Saving files

Saving files works analogously to loading: When you press **F2**, the focused window temporarily changes into a file save dialog, as shown in Figure 3-19. Click the lower edit box to enter a filename. If it doesn't end with ".blend", the extension is automatically appended. Then press **ENTER** to write the file. If a file with the same name already exists, you will have to confirm the overwrite prompt.

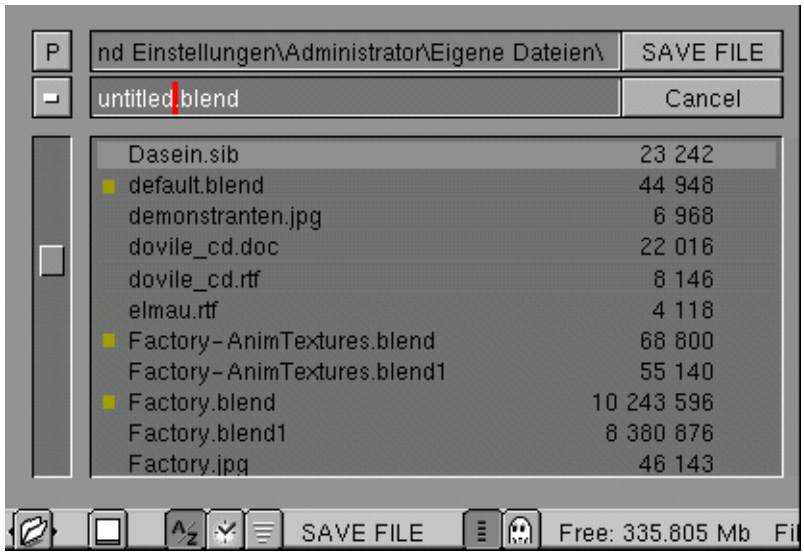


Figure 3-19. File save dialog.

The save dialog contains a small feature to help you to create multiple versions of your work: Pressing **NUM+** or **NUM-** increments or decrements a number contained in the filename. If you want to simply save over the currently loaded file and skip the save dialog, you can press **CTRL-W** instead of **F2** and just need to confirm the prompt.

Rendering

This section should only give you a quick overview of the vitals to be able to render your scene. A detailed description of all options can be found in Chapter 14.

The render settings can be found in the DisplayButtons (Figure 3-20) that can be reached by clicking the  , or simply by pressing **F10**.

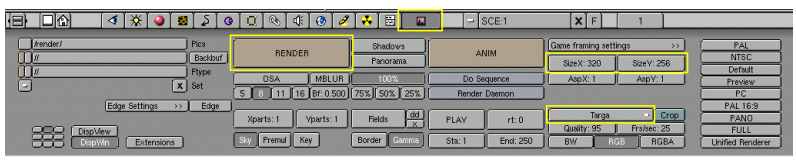


Figure 3-20. Rendering options in the DisplayButtons.

All we are interested in for now is the size (number of pixels horizontally and vertically) and file format for the image to be created. The size may be set using the `SizeX` and `SizeY` buttons, and clicking the selection box below (in Figure 3-20, "Targa" is chosen) opens a menu with all available output formats for images and animations. For still images, we may choose Jpeg, for instance.

Now that the settings are made, the scene may be rendered by hitting the `RENDER` button or by pressing **F12**. Depending on the complexity of the scene, this usually takes between a few seconds and several minutes, and the progress is displayed in a

separate window. If the scene contains an animation, only the current frame is rendered. (To render the whole animation, see the Section called *Rendering Animations* in Chapter 14.)

If you don't see anything in the rendered view, make sure your scene is constructed properly: Does it have lighting? Is the camera positioned correctly, and does it point in the right direction? Are all the layers you want to render visible?

Please note that a rendered image is not automatically saved to disk. If you are satisfied with the rendering, you may save it by pressing **F3** and using the save dialog as described in the Section called *Saving files*. It is saved in the format you previously selected in the DisplayButtons.

User preferences

Blender has a few options that are not saved with each file, but apply for all files of a user instead. These preferences primarily concern user interface handling details, and system properties like mouse, fonts and languages.

As the user preferences are rarely needed, they are neatly hidden behind the main menu. To make them visible, pull down the window border of the menu (usually the topmost border in the screen). The settings are grouped into six categories which can be selected with the violet buttons shown in Figure 3-21.

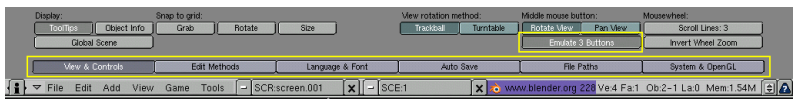


Figure 3-21. User preferences window.

Most buttons are self-explaining or display a helpful tool-tip if you hold the mouse still over them, so they won't be described in detail here. We will just give an overview of the preference categories:

View & Controls

Settings concerning how the user interface should react to user input, e.g. which method of rotation should be used in 3D views. Here you can also activate 3-button mouse emulation if you have a two-button mouse. **MMB** can then be input as **ALT-LMB**.

Edit Methods

Lets you specify details of the workings of certain editing commands like `duplicate`.

Language & Fonts

Select an alternative TrueType font for display in the interface, and choose from available interface languages.

Auto Save

Auto saves can be created to have an emergency backup in case something goes wrong. These files are named Filename.blend1, Filename.blend2, etc.

File Paths

Choose the default paths for various file load dialogs.

System & OpenGL

You should consult this section if you experience problems with graphics or sound output, or if you don't have a numerical keypad and want to emulate it (for laptops).

Setting the default scene

You don't like Blender's default window setup, or want specific render settings for each project you start? No problem. You can use any scene file as a default when Blender starts up. Make the scene you are currently working on default by pressing **CTRL-U**. It will then be copied into a file called .B.blend in your home directory.

You can clear the working project and revert to the default scene anytime by pressing **CTRL-X**. Please remember to save your changes to the previous scene first!

Chapter 4. Your first animation in 30 minutes

This chapter will guide you step by step through the animation of a small "Gingerbread Man" character.

In the following allactions will be described as step-by-step as possible, anyway it will be assumed that you have read the whole Chapter 3 and have understood the conventions used throughout this manual.

Warming up

Fire up Blender by double clicking its icon or from the command line. Blender will open showing you, from top view, the default set-up: a camera and a plane. The plane is pink, this means it is selected (Figure 4-1). Delete it with **XKEY** and confirm by clicking the `Erase Selected` entry in the dialog which will appear.

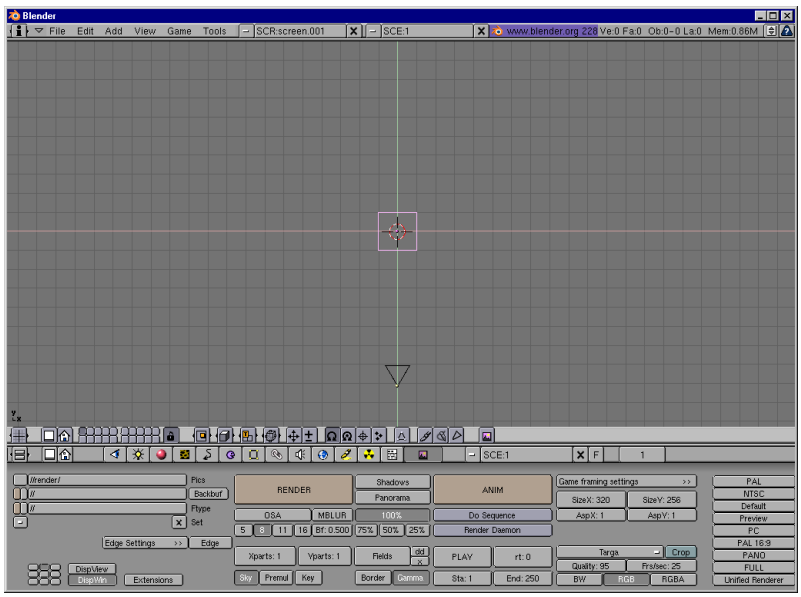


Figure 4-1. Blender window as soon as you start it.

Now select the camera with **RMB** and press **MKEY**. A small toolbox, like the one in Figure 4-2, will appear beneath your mouse, with the first button checked. Check the rightmost button on the top row and then the `OK` button. This will move your camera to layer 10. Blender provides you with 20 layers to help you organize your work. You can see which layers are currently visible from the group of twenty buttons in the 3D window toolbar (Figure 4-3). You can change the visible layer with **LMB** and toggle visibility with **SHIFT-LMB**

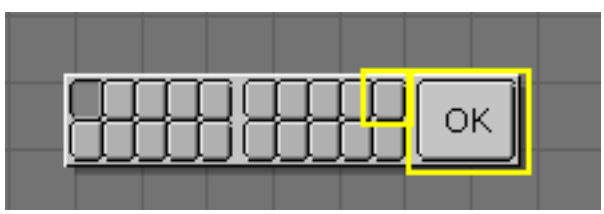


Figure 4-2. Layer control toolbox.

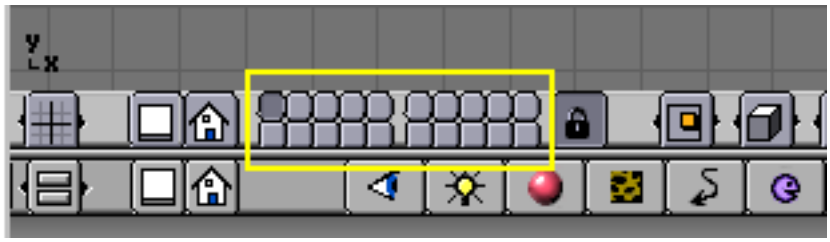


Figure 4-3. Layer visibility controls.

Building the body

Turn to the front view with **NUM1** and add a cube by pressing **SPACE** and selecting menu `ADD`, submenu `Mesh`, sub-submenu `Cube`. In the following exercises we will write **SPACE**>>ADD>>Mesh>>Cube as shorthand for these kind of actions. A cube will appear (Figure 4-4). A newly added mesh is in a particular mode called *EditMode* in which you can move the single vertices that comprise the mesh. By default all vertices are selected (Yellow), all edges selected (Dark Yellow) and all faces selected (Pink).

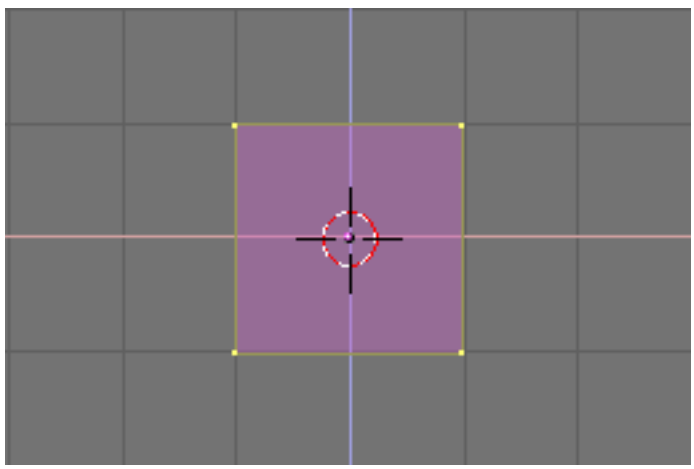


Figure 4-4. Our cube in EditMode, all vertices selected.

We will call the Gingerbread man "Gus". Our first task is to build Gus' body. This will be done by working on our cube in EditMode with the set of tools Blender provides. To have a look at these tools push the button which has a square with yellow vertices in the Button Window (Figure 4-5). There is a keyboard shortcut for this, **F9**.

Figure 4-5. The Edit Buttons window button.

Now locate the `Subdivide` button and press it once (Figure 4-6). This will split each side of the cube in two, creating new vertices and faces (Figure 4-7).

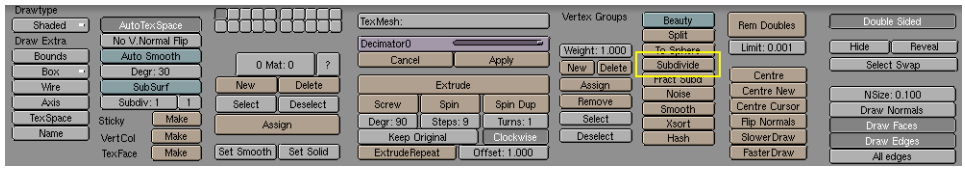


Figure 4-6. The Edit Buttons window for a Mesh.

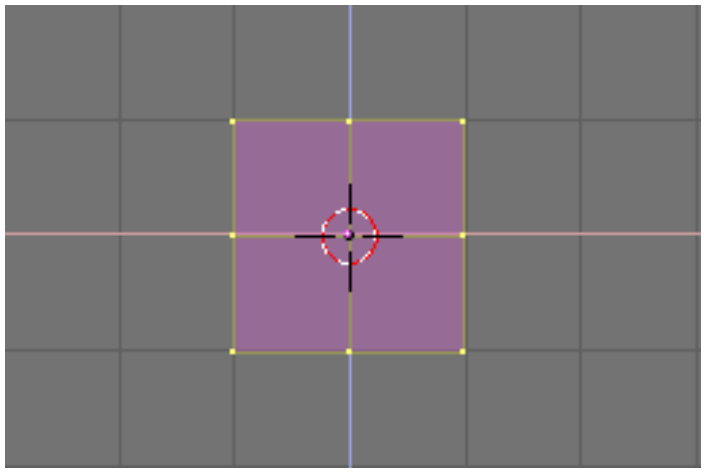


Figure 4-7. The cube, subdivided once.

With your cursor hovering on the 3D window press **AKEY**. This will de-select all. Vertices will turn Pink, edges Black and faces Blue. Now press **BKEY**. The cursor will change to a couple of orthogonal gray lines. Go above and to the left with respect to the cube, press **LMB** and, while keeping it pressed, drag the mouse down and to the right so that the gray box which appears encompasses all the leftmost vertices. Now release the **LMB**. This sequence, which let you select a group of vertices in a box, is summarized in Figure 4-8.

> **Box Select:** In many occasions there may be vertices hidden behind other vertices. This is the case here, our subdivided cube has 26 vertices, yet you can only see nine because the others are hidden.
>
> A normal **RMB** click selects only one of these stacked vertices, whereas a Box select selects all. Hence, in this case, even if you see only three vertices turning yellow you have actually selected nine vertices.

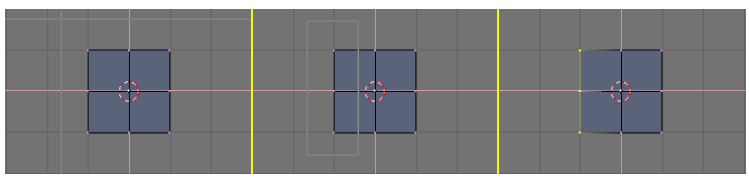


Figure 4-8. The sequence of Box selecting a group of vertices.

Now press **XKEY** and, from the menu that pops up, select `Vertices` to erase the selected vertices (Figure 4-9).

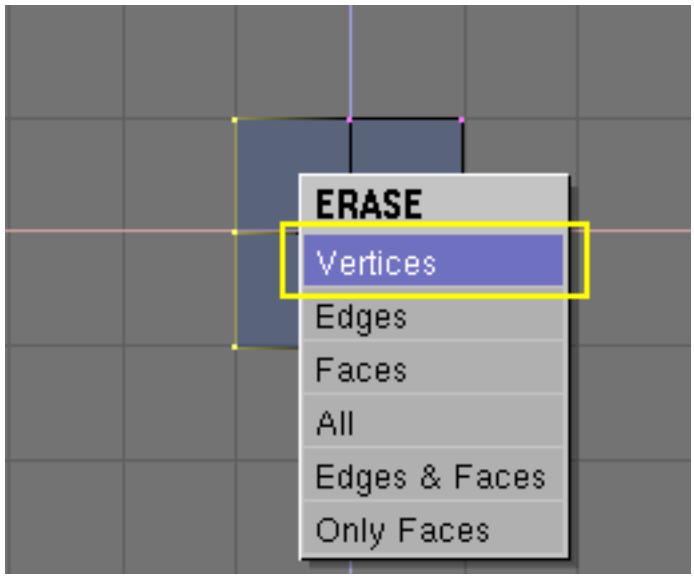


Figure 4-9. The pop-up menu of the Delete (XKEY) action.

> **Undo:** Beware, for performance issues Blender does *not* have an Undo function, at least not the one everyone thinks of as an Undo function.
>
> In any case, this is not a great issues as there are many mechanisms to recover from mistakes.
>
> One of these is the Mesh Undo function. This works *only* in EditMode and returns the mesh to the state it had *when EditMode was entered*.
>
> You can switch in and out of EditMode by pressing **TAB**. You might wish to switch out of EditMode every time you do one of our modelling steps correctly, and then switch back in. And press **UKEY** to turn back to the last correct mesh if necessary.
>
> Also, pressing **ESC** in the middle of an action cancels the action, reverting to the previous state.
>
> We will cover other Undo methods later.

Now, with the sequence you just learned, Box Select the two topmost vertices (Figure 4-10, left). Press **EKEY** and click on the `Extrude` menu entry which appears to extrude them. This will create new vertices and faces. The newly created vertices are free to move and will follow the mouse. Move them to the right.

To constrain the movement horizontally or vertically you can click **MMB** while moving. The movement will then be constrained horizontally if you were moving more or less horizontally, or vertically otherwise. You can swich back to unconstrained movement by clicking **MMB** again.

Move them 1 square and a half to the right, then click **LMB** to fix their position. Extrude again, via **EKEY** and move the new vertices another half square to the right. Figure 4-10 show this sequence.

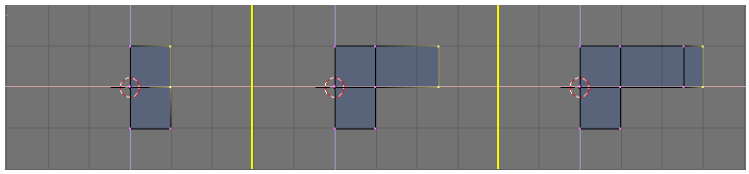


Figure 4-10. Extruding the arm in two steps.

Now Gus has his left arm (he's facing us). We will build the left leg the same way by extruding the lower vertices.

Try to get to something like that shown in Figure 4-11. Please note that for the leg we used the Extrude tool three times. We don't care of elbows... but we will need a knee later on!

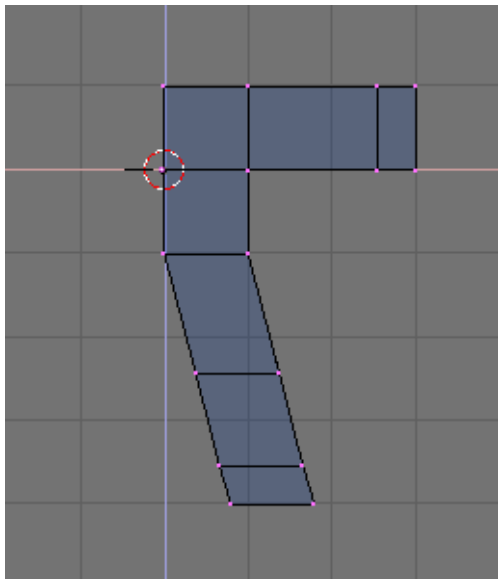


Figure 4-11. Half body.

> **Coincident vertices:** If you extrude and, in the moving process you change your mind and press **Esc** to recover, the extruded vertices will still be there, in their original location!
>
> You can move them again, by pressing **GKEY** or do whatever you want (scale rotate etc.), but you probably don't want to extrude them again.
>
> To fully undo the extrusion look for the `Remove Doubles` button, highlighted in Figure 4-12. This will eliminate coincident vertices.

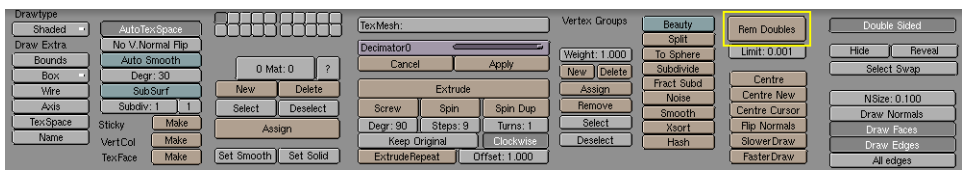


Figure 4-12. The Edit Buttons window.

Now it is time to create the other half of Gus, select all vertices (**AKEY**) and press the 3DWindow toolbar button which resembes a cross-hair (Figure 4-13). Now, leave the mouse still. Press **SHIFT-D** to duplicate all selected vertices, edges and faces, **SKEY** to switch to "Scale" mode, then **XKEY** followed by either **ENTER** or a **LMB** click to flip the duplicate. The result is shown in Figure 4-14.

Figure 4-13. Setting the reference center to the cursor.

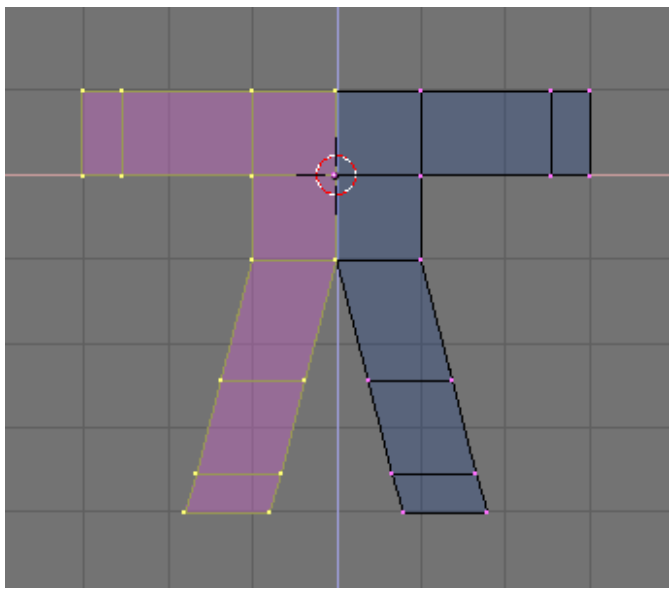


Figure 4-14. Flipped copy of the half body to obtain a full body.

De-select all and re-select all by pressing **AKEY** twice and eliminate the coincident vertices by pressing the `Remove doubles` button (Figure 4-12). A box will appear, notifying you that 8 vertices have been removed.

> **Reference center:** In Blender, scaling, rotating and other mesh modifications occur either with respect to the cursor position, or the object center or the baricentrum of the selected items, depending on which of the four buttons in the top center group in Figure 4-13 is active. The cross-hair one selects the cursor as reference.

> **Moving the cursor:** Here we have assumed that the cursor never moved from its original position. As a matter of fact, you move the cursor by clicking **LMB** in the 3D window.
>
> If you need to place the cursor at a specific grid point, as it is in the present case, you can place it next to the desired position and press **Shift-S**. This brings up the Snap Menu. The entry `Curs->Grid` places the cursor exactly on a grid point. The `Curs->Sel` places it exactly on the selected object. The other entries move objects rather than the cursor.

Use what you have just learned about moving the cursor to place it exactly one grid square above Gus' body (Figure 4-15, left). Add a new cube here (**SPACE**>>ADD>>Mesh>>Cube). Press **GKEY** to switch to Grab Mode for the newly created vertices and move them down, constraining the movement with **MMB**, for about one third of a grid unit (Figure 4-15, right).

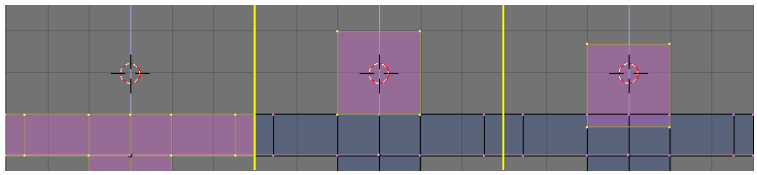


Figure 4-15. The sequence of adding the head.

This is a rough figure at best, to make it smoother locate the `SubSurf` Toggle Button Figure 4-16 and switch it on. Be sure to set both the two NumButtons below to 2. Switch out of EditMode (**TAB**) and switch from the current default Wireframe Mode to Solid mode with **ZKEY** to have a look at Gus. It should look like Figure 4-17 left. The SubSurf technique dynamically computes a high poly smooth mesh of a low poly coarse mesh.

Figure 4-16. The Edit Buttons window.

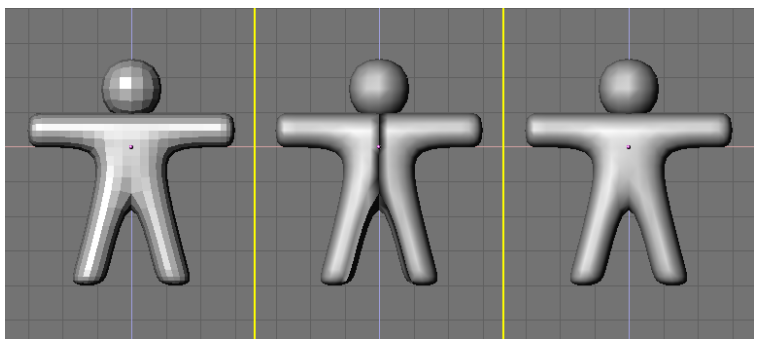


Figure 4-17. Setting Gus to smooth.

To make it look smooth, press the `SetSmooth` button in Figure 4-16. Gus will now be smooth but with funny black lines in the middle (Figure 4-17, middle). This because the SubSurf is computed using information about the coarse mesh normal directions, and these might not be coherent if extrusions and flippings have been made. To reset the normals, switch back to EditMode (**TAB**), select all vertices (**AKEY**) and press **CTRL-N**. Click with **LMB** on the `Recalc normals outside` box which appears. Now Gus will be nice and smooth, as shown in Figure 4-17, right.

Press **MMB** and drag the mouse around to view Gus from all angles. He is too thick! Switch to side view **NUM3**. Switch to EditMode if you are not there already, and back to Wireframe Mode (**ZKEY**), and select all vertices with **AKEY** (Figure 4-18, left).

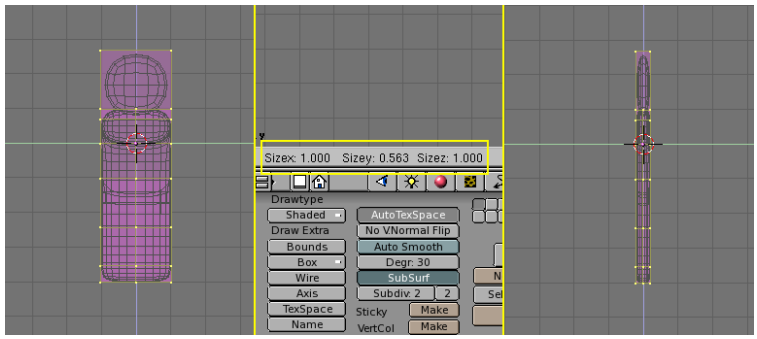


Figure 4-18. Slimming down Gus by constrained scaling.

Now, to make Gus thin, press **SKEY** and start to move the mouse horizontally. Click **MMB** to constrain scaling to just one axis. If you now move the mouse toward Gus he should become thinner but remain the same height. You should note on the 3DWindow toolbar three numbers giving the scaling factor. After you have clicked **MMB** only one will vary. Press and keep pressed **Ctrl**. The scale factor will now vary in discrete steps of value 0.1. Scale Gus down so that the factor is 0.2 and fix the dimension by clicking **LMB**.

Switch back to Front view and to Solid mode (**ZKEY**) rotate your view via **MMB**. Gus is much better!

Let's see what he looks like

We can't wait any longer and we want our first render! But first we still need to do some work.

Shift-LMB on the top right small button of the layer visibility buttons in the 3DWindow toolbar (Figure 4-19) to make both Layer 1 (Gus' layer) and Layer 10 (the layer with the camera) visible.

Figure 4-19. Making both layer 1 and 10 visible.

Remember that the *last* layer selected is the active layer, so all subsequent additions will automatically be on layer 10.

Select the camera (**RMB**) and move it to a location like (x=7, y=-10, z=7). You can do this by pressing **GKEY** and dragging the camera around keeping **CTRL** pressed to move it in steps of 1 grid unit.

> **Entering precise locations and rotations:** If you prefer to type in numerical values for an Object's location you can do so by pressing **NKEY** and modifying the NumButtons in the dialog that appears (Figure 4-20). Remember to press OK to confirm your input.

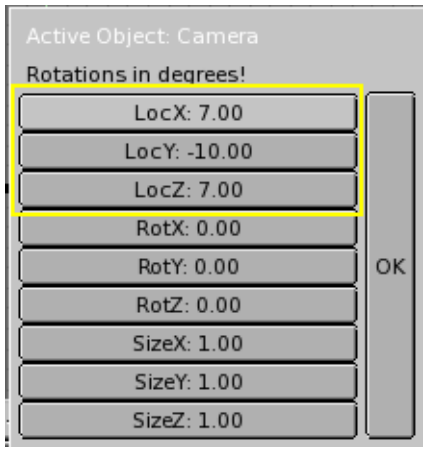


Figure 4-20. The window for numerical imput of object position/rotation etc.

To make the camera point at Gus, with your camera still selected also select Gus via **SHIFT-RMB**. Now the camera will be Magenta and Gus Light Pink. Press **CTRL-T** and select the `Make Track` entry in the pop up. This will force the camera to track Gus and always point at him. You can move the camera wherever you want later on and be sure Gus is in the center of the camera view!

> **Tracking:** If the tracking object already has a rotation of its own, as is often the case, the result of the **CTRL-T** sequence might not be what was expected.
>
> In this case select the tracking object, in our example the camera, and press **ALT-R** to remove any object rotation. Once you do this the camera will really track Gus!

Figure 4-21 shows Top, Front, Side and Camera view of Gus. To obtain a Camera view press **NUM0**.

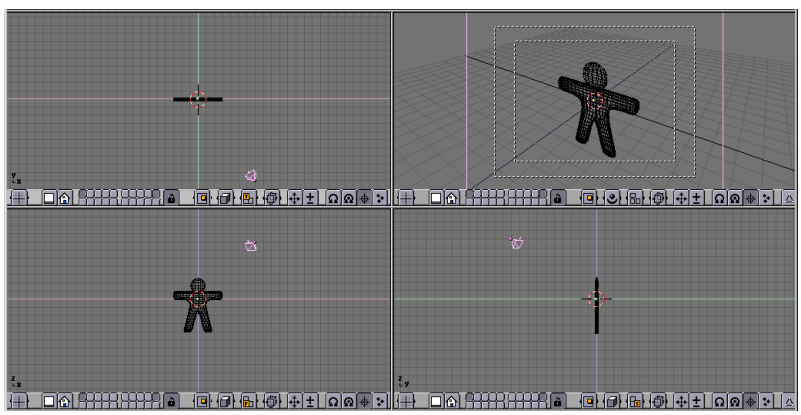


Figure 4-21. Camera position with respect to Gus.

Now we need a ground on which Gus can stand. In top view (**NUM7**), and *out* of EditMode add a plane (**SPACE**>>ADD>>Mesh>>Plane). It is important to be out of EditMode, otherwise the newly added object would be part of the object currently in EditMode, as it was for Gus' head when we added it. If the cursor is where Figure 4-21 shows, such a plane will be in the middle of Gus' head. Switch to ObjectMode and Front view (**NUM1**) and move (**GKEY**) the plane down to Gus feet, using **CTRL** to keep it aligned with Gus.

Switch the reference center from cursor (where we set it at the beginning) to object pressing the highlighted button of Figure 4-22. Go to Camera view (**NUM0**) and, with the plane still selected, press **SKEY** to start scaling.

Figure 4-22. Set the reference center to Object center.

Scale the plane up, so that it is so big that its edges are outside of the camera viewing area. This is indicated by the outer white dashed rectangle in Camera view.

In Top view (**NUM7**) add a Lamp light (**SPACE**>>ADD>>Lamp) in front of Gus, but on the other side with respect to the camera, for example in (x=-9, y=-10, z=7) (Figure 4-23).

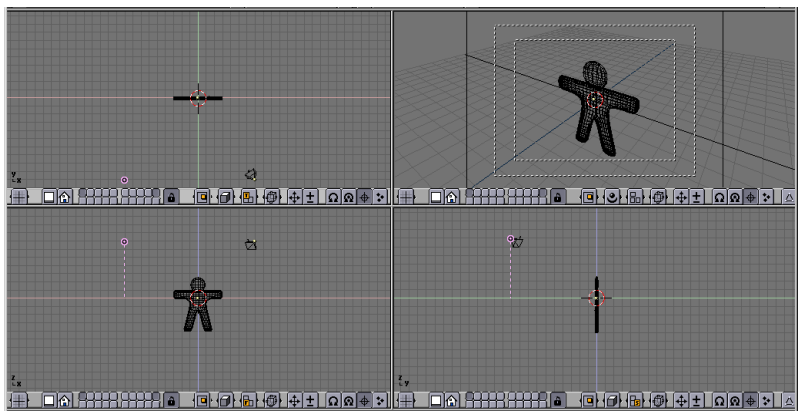


Figure 4-23. Inserting a Lamp.

Switch to Lamp Buttons via the button with a lamp in the Button Window toolbar (Figure 4-24) or **F4**.

Figure 4-24. The Lamp Buttons window button.

In the Buttons Window press the `Spot` toggle button to make the lamp a Spotlight (Figure 4-25) of Pale Yellow (R=1, G=1, B=0.9) colour. Adjust `ClipSta:` Num Button to 5, `Samples:` to 4 and `Soft:` to 8.

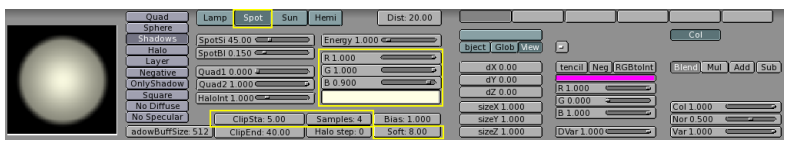


Figure 4-25. Spot light settings.

Make this Spotlight track Gus exactly as you did for the camera (Select Spot, **SHIFT** select Gus, press **Ctrl-T**. If you added the spot in Top View you should need not to clear its rotation via **Alt-R**.

In the same location as the Spot, and again in Top View, add a second Lamp (**SPACE**>>ADD>>Lamp). Make this one a `Hemi` type with Energy of 0.6 (Figure 4-26).

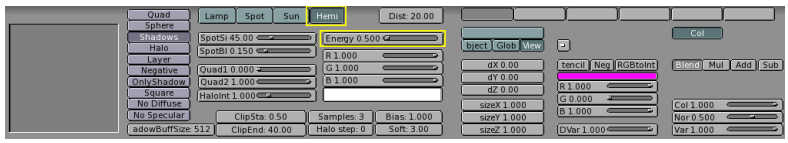


Figure 4-26. The Hemi lamp settings

> **Two lamps?:** Having two or more lamps helps a lot to give soft, realistic lighting. In reality light never comes from a single point. You will learn more about this in the Lighting Chapter.

We're almost ready to render. First go to the Render Buttons by pressing the image-like icon in the Button Window toolbar (Figure 4-27).

Figure 4-27. The Rendering Buttons window buttons.

In the Render Buttons set the image size to 640x480 with the Num Buttons top right, set the `Shadows` Toggle Button top center to On, and the `OSA` Toggle Button center-left to On as well (Figure 4-28). These latter controls will enable shadows and oversampling (OSA) to prevent jagged edges.

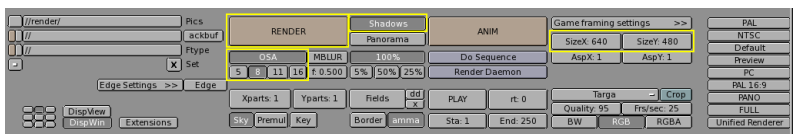


Figure 4-28. The Rendering Buttons window

You can now hit the `RENDER` button or hit **F12**. The result is shown in Figure 4-29... and is actually quite poor. We still need materials! And lots of details, such as eyes, etc.

Figure 4-29. Your first rendering. Congratulations!

Saving: If you have not done it yet, this is a good point to save your work, via the `File>>Save` menu shown in Figure 4-30, or **CTRL-WKEY**

Blender will always warn you if you try to overwrite an existing file.

Blender does automatic saves into your system's temporary directory. By default this happens every 4 minutes and the file name is a number. This is another way to undo your last changes!

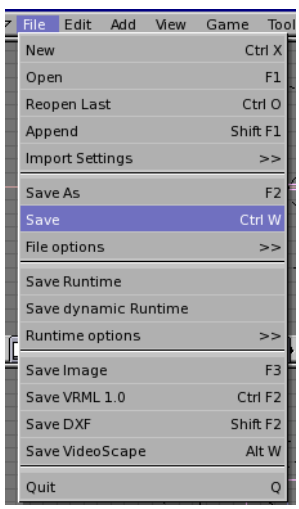


Figure 4-30. The Save menu.

Materials and Textures

Select Gus, it is time to give him some nice cookie like material. In the Button Window toolbar press the red dot button (Figure 4-31) or use the **F5** key.

Figure 4-31. The Material Buttons window Button.

The Button window will be almost empty because Gus has no materials yet. To add one, click on the white square button in the Button Window toolbar and select `ADD NEW` (Figure 4-32).

Figure 4-32. The Material Menu button.

The Buttons window will be populated by buttons and a string holding the Material name, "Material" by default, will appear next to the white square button. Change this to something meaningful, like GingerBread.

Modify the default values as per Figure 4-33 to obtain a first rough material.

Figure 4-33. The Material Buttons window and a first gingerbread material.

Press the small button with a white square on the right of the Material Buttons, in the Textures area (Figure 4-34) and select `Add new`. We're adding a texture in the first channel. Give it some name like "GingerTex"

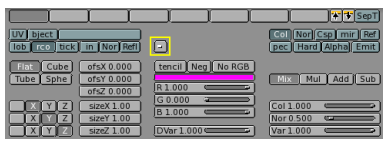


Figure 4-34. The Textures menu button in the Material Buttons

Select the Texture Buttons by clicking the button in Figure 4-35 or by pressing **F6**

Figure 4-35. The Texture Buttons window Button.

From the top row of ToggleButtons which appear select `Stucci` and set all parameters as in Figure 4-36.

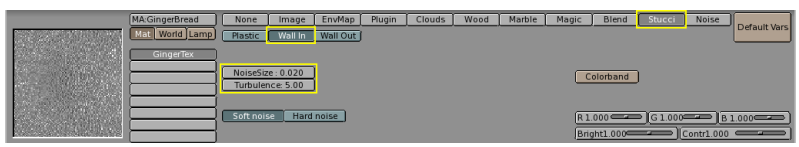


Figure 4-36. The Texture Buttons window with a stucci texture.

Go back to the Material Buttons (**F5**) and set the Texture buttons as in Figure 4-37. The only settings to change should actually be the un-setting of the `Col` Toggle Button and the setting of the `Nor` Toggle Button and rising the `Nor` slider to 0.75. This will make our Stucci texture act as a "bumpmap" and make Gus look more biscuit-like.

Figure 4-37. Settings for the Stucci texture in the Material Buttons window.

You can also add a second texture, name it 'Grain' and make it affect only the `Ref` property with a 0.4 `Var` (Figure 4-38). The texture itself being a plain `Noise` texture.

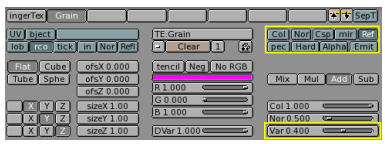


Figure 4-38. Settings for an additional Noise texture in channel 2.

Also give the ground an appropriate material. For example, the dark blue one shown in Figure 4-39.

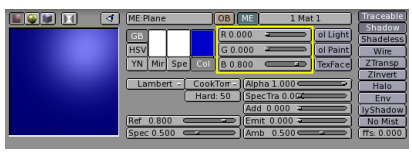


Figure 4-39. A very simple material for the ground.

To give some finishing touches we should add eyes and some other details.

First make Layer 1 the only visible by clicking with **LMB** on the layer 1 button (Figure 4-40). This will hide the lamps, camera and ground.

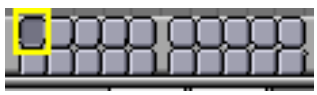


Figure 4-40. Layer visibility buttons on toolbar.

Place the cursor at the center of Gus' head, remember you are in 3D so you must check at least two views to be sure! Add a sphere (**SPACE**>>ADD>>Mesh>>UVsphere). You will be asked for the number of `Segments:` (meridians) and `Rings:` (parallels) into which to divide the sphere. The default of 32 is more than we need here, so use a value of 16 for both. The Sphere is in the first image top left of the sequence in Figure 4-41.

Scale it down (**SKEY**) to a factor 0.1 in all dimensions, then switch to side view (**NUM3**) and scale it only in the horizontal direction to a further 0.5 (Second two images in Figure 4-41).

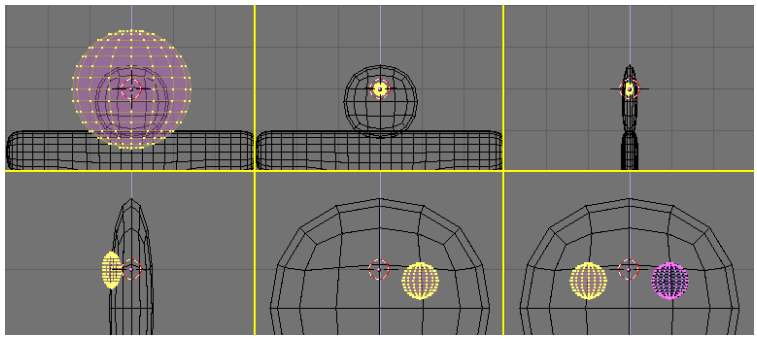


Figure 4-41. Sequence for creation of the eyes.

Zoom a little if you need via **NUM+** or **MW** or **CTRL-MMB** and drag, and move the sphere (**GKEY**) to the left so that it is half in, half out of the head (First image in the second row of Figure 4-41).

Go back to front view (**NUM1**) and move the sphere sideways to the right. Place it at a point where Gus should have an eye.

Flip a duplicate with respect to the cursor by following the sequence you learned when flipping Gus' body (Select the crosshair toolbar button, **SHIFT-D**, **SKEY**, **XKEY**, **LMB**). Now Gus has two eyes.

Exit out of EditMode, and place the cursor as close as you can at the center of Gus' face. Add a new sphere and scale/move it exactly as before, but make it smaller and place it lower than and right of the cursor, centered on the SubSurfed mesh vertex Figure 4-42).

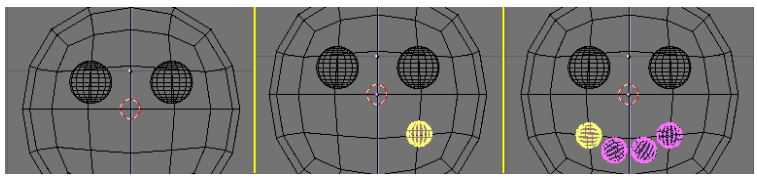


Figure 4-42. Creating a mouth with Spinning tools.

Now, in the Edit Buttons (**F9**) locate at the center the group of buttons in Figure 4-43. Set `Degr:` to 90, `Steps:` to 3 and verify that the `Clockwise:` TogButton is on. Then press `SpinDup`. This will create 3 duplicates of the selected vertices on an arc of 90° centered at the cursor. The result is Gus' mouth, like the last image of the sequence in Figure 4-42.

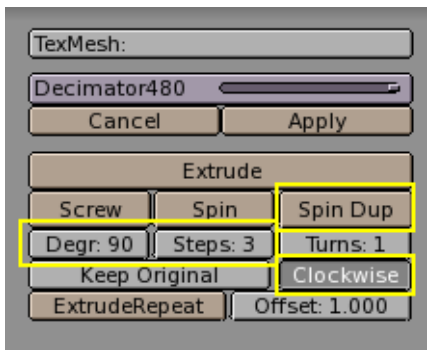


Figure 4-43. The Spin Tools buttons in the Edit Buttons window.

Now you have learned this trick, add three more of these ellipsoids to form Gus' buttons. Once you have made one, you can simply exit from EditMode, press **Shift-D** to create a duplicate and move it into place, like in Figure 4-44.

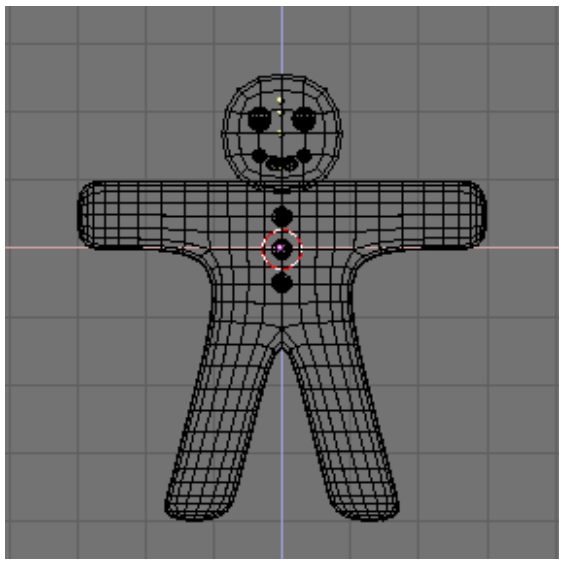


Figure 4-44. The complete Gus!

Give the eyes a Chocolate-like material as per the one at the top in Figure 4-45. Give the mouth a White Sugar like material like the second one in Figure 4-45, and give the buttons a Red, White and Green sugar like material. From top to bottom, these are shown in Figure 4-45 too.

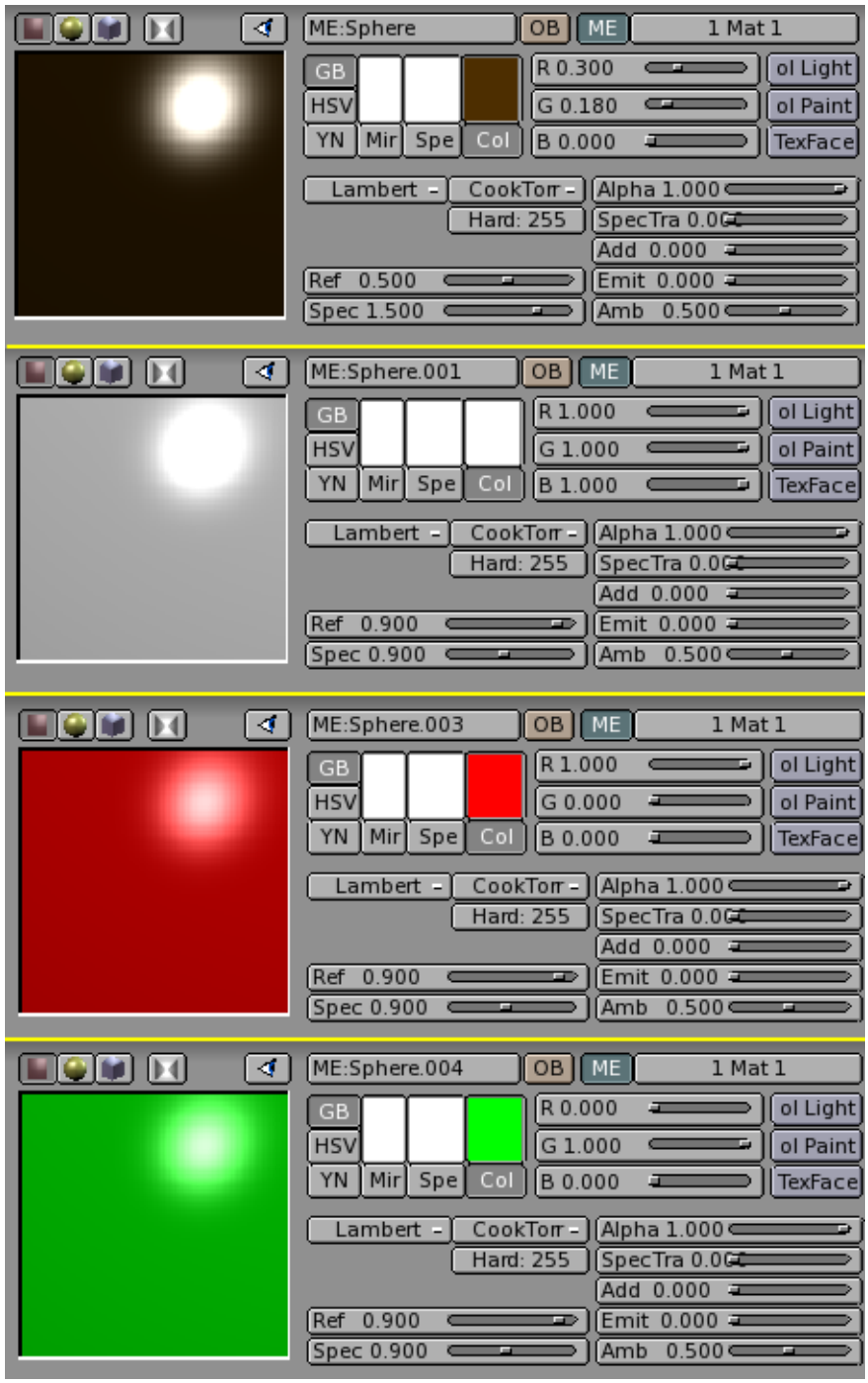


Figure 4-45. Some other candy materials.

Objects sharing a material: To give an Object the same material as another object, select that material in the list which appears when you press the button with the white square in the ButtonWindow toolbar.

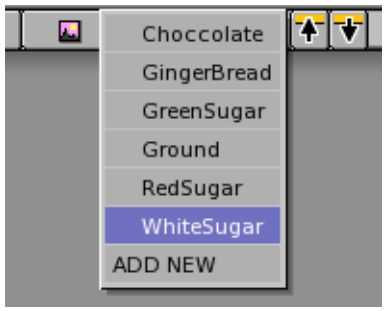


Figure 4-46. Selecting an existing material from the Material Menu in the Toolbar.

When you have finished assigning materials, set layer 10 visible again (you should know how by now), so that lights and the camera also appear, and do a new rendering (**F12**). The result should look more or less like Figure 4-47.

Figure 4-47. The complete Gus still rendering.

You might want to save your image now. Press **F3**. You will be presented with a file window, type in the name of your image and save.

> **Image types and extension:** You must decide the image format (JPEG, PNG etc.) *before* pressing **F3** by setting it in the Rendering Buttons (Figure 4-27) and using the PopUp menu (Figure 4-48).
>
> Beware that Blender does *not* add the extension to the file name by default, it is up to you to type one in if you want.

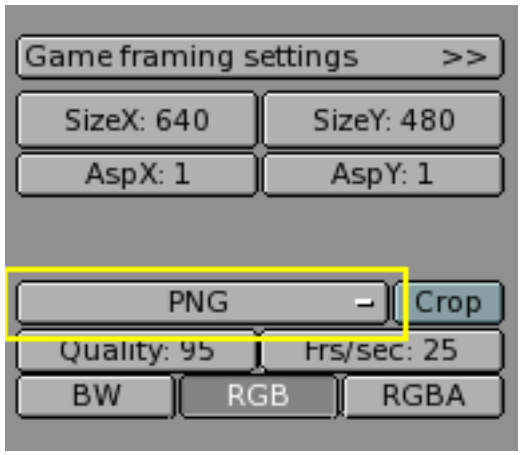


Figure 4-48. File type selection menu in the Rendering Buttons window.

Rigging

If we were going for a still picture, this would be enough, but we want Gus to move! The next step is to give him a skeleton, or Armature, which will move him. This is the fine art of rigging.

Our Gus will have a very simple rigging, four limbs, two arms and two legs, but no joints (elbows or knees), not even feet or hands.

Set the cursor where the shoulder will be, press **SPACE**>>ADD>>Armature. A romboidal object will appear, stretching from cursor to mouse pointer. This is a bone of the armature system. Place its other end in Gus' hand (Figure 4-49) with **LMB**. The bone will be fixed and a new bone will be created from the end point of the previous, ready to build a bone chain. We don't need other bones right now, so press **ESC** to exit.

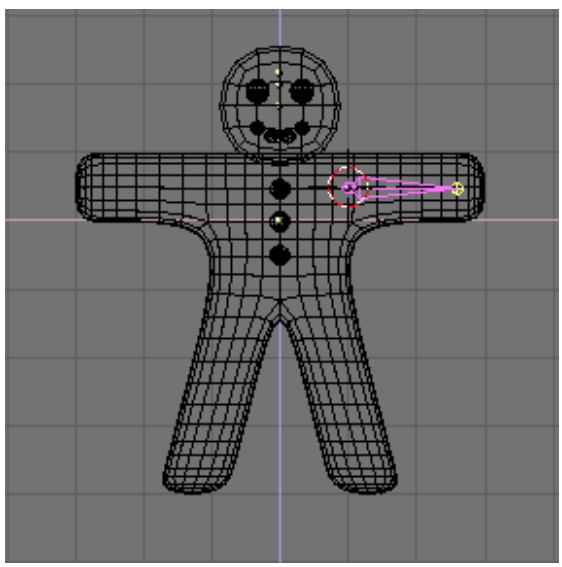


Figure 4-49. Adding the first bone, an elbowless arm.

Stay in EditMode, move the cursor where the hip joint will be and add a new bone (**SPACE**>>ADD>>Armature) down to the knee. Press **LMB** a new bone automatically is there. Stretch this down to the foot (Figure 4-50).

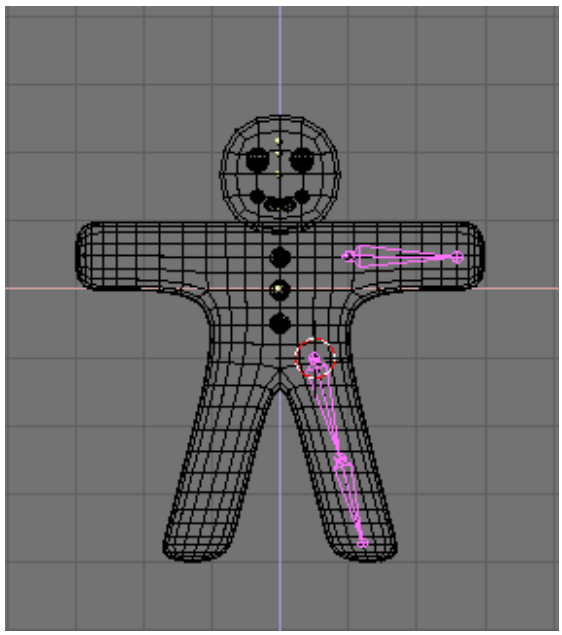


Figure 4-50. Adding the second bone and third bones, a leg bone chain.

> **Bone position:** The bones we are adding will deform Gus body mesh. To have neat result it is very important that you try to place the bone joints as in the figures.

Now place the cursor in the center and select all bones with **AKEY**, duplicate them with **Shift-D** and Flip them as you did for the meshes with **XKEY** (Figure 4-51).

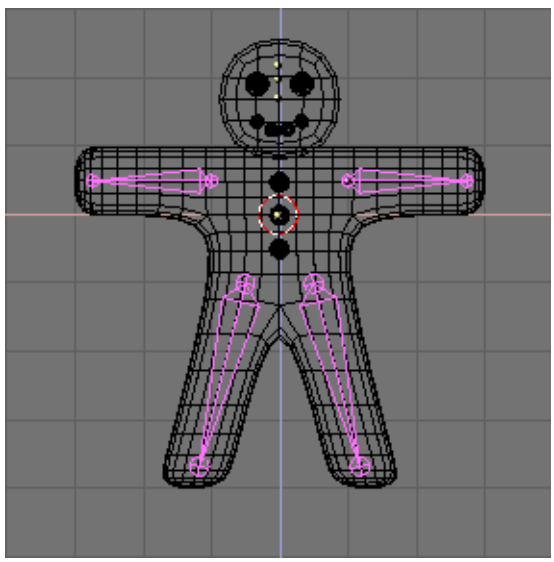


Figure 4-51. Complete armature after duplicating and flipping.

If you take a look at the Edit Buttons window (Figure 4-9) it will be very different now, and, once you've selected all the bones (**AKEY**), will exhibit the Armature buttons (Figure 4-52).

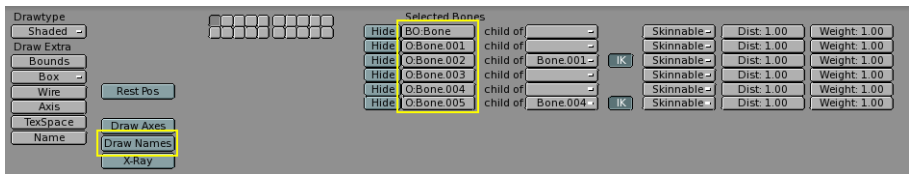


Figure 4-52. The Edit Buttons window for an armature.

First press the `Draw Names` button to see the names of the bones, then, **SHIFT-LMB** on the names in the Edit Button window (Figure 4-52) to change them to something appropriate Like Arm.R, Arm.L, UpLeg.R, LoLeg.R UpLeg.L, LoLeg.L. Exit from EditMode (**TAB**).

> **Naming Bones:** It is very important to name bones with trailing '.L' or '.R' to distinguish between left and right because this way the Action editor will be able to automatically 'flip' your poses.

Skinning

Now we must make it so that a deformation in the armature causes a matching deformation in the body. This is accomplished in the Skinning process, where vertices are assigned to bones and so are subject to their movements.

Select Gus' body first, then **SHIFT** select the armature so that the body is magenta and the armature light pink. Press **CTRL-P** to parent the body to the armature. A Pop up dialog will appear (Figure 4-53). Select the `Use Armature` Entry.

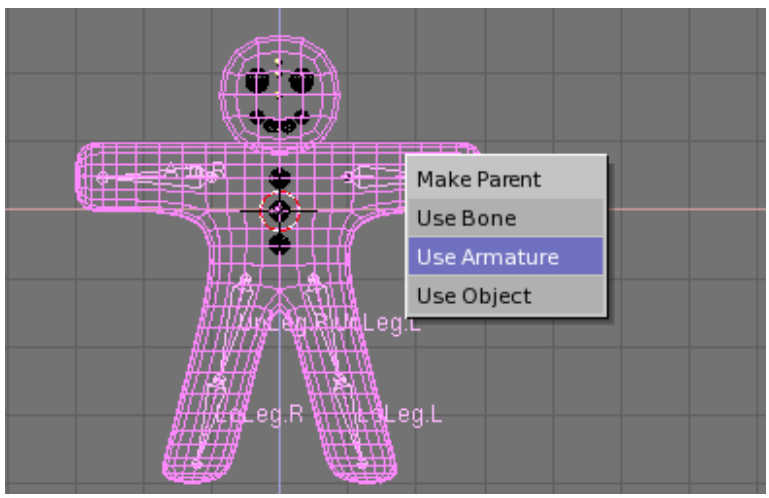


Figure 4-53. The pop-up menu which appears when parenting an Object to an Armature.

A new menu appears, asking you if you want Blender to do nothing else, create empty vertex groups, or create and populate vertex groups (Figure 4-54).

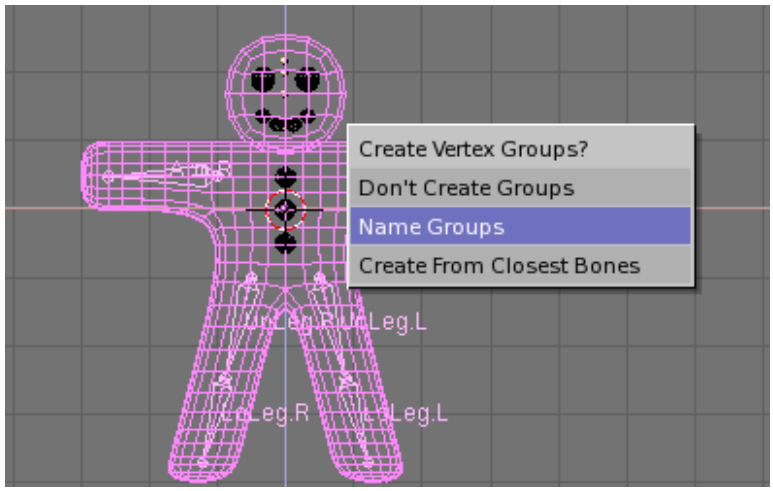


Figure 4-54. Automatic Skinning options.

For our example we will try the automatic skinning option and select `Create From Closest Bones`

Now select only Gus' body and go to EditMode (**TAB**). You will notice in the Edit Buttons (**F9**) window the presence of a vertex group menu and buttons (Figure 4-55).

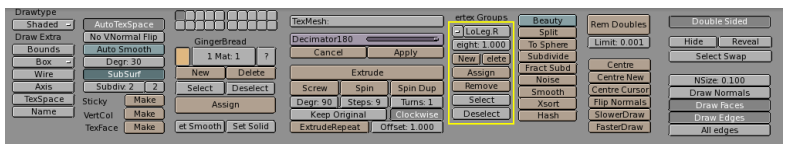


Figure 4-55. The vertex groups buttons in the Edit Buttons window of a mesh.

By pressing the button with the small white square a menu with all available vertex group pops up (six in our case, but trulycomplex character, whith hands andfeets completely rigged can have tens of them! Figure 4-56). The buttons `Select` and `Deselect` shows you which vertices belongs to which group.

Figure 4-56. The menu with the vertex groups automatically created in the skinning process.

Select the Right arm (Arm.R) group and press `Select`. You should see something like Figure 4-57.

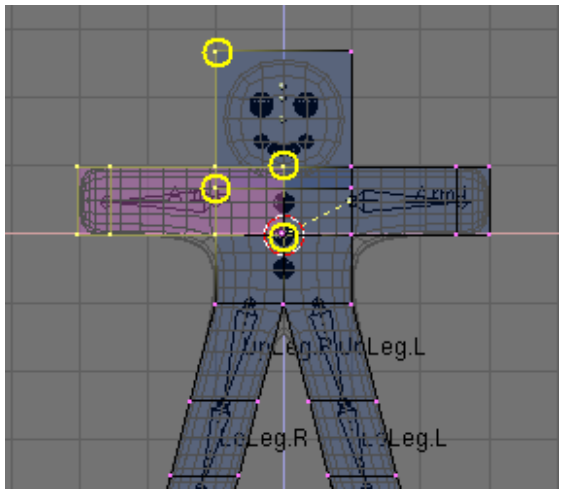


Figure 4-57. Gus in EditMode with all the vertices of group Arm.R selected.

The vertices marked with the yellow circles in Figure 4-57 do belong to the deformation group because the autoskinning process found that they were very close to the bone, but should not, since some are in the head and some in the chest and we don't want them to be deformed. To remove them from the group De-select all the others by using Box selection (**BKEY**) but resorting to the **MMB** to define the box. This way all vertices within the box which are selected become de-selected.

Once only the 'undesired' vertices remain selected, press the `Remove` button (Figure 4-55) to eliminate them from group Arm.R.

De-select all (**AKEY**) and check another group. Check them all and be sure they look like those in Figure 4-58.

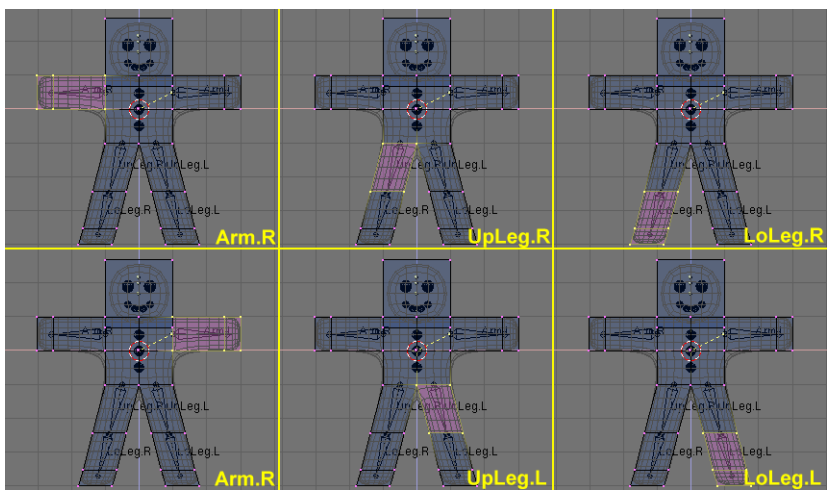


Figure 4-58. The six vertex groups.

> **Vertex groups:** Be very careful when assigning/removing vertices from vertex groups. If later on you see unexpected deformations, you might have forgotten some vertices, or taken too many away.
>
> You can of course modify your vertex groups at any time.

Other details: Please note that what we are doing will only affect Gus' body, not his eyes, teeth or buttons, which are separate objects.

This is not an issue in our simple animation, but must be taken into account for more complex projects, for example by parenting them to the Body or by joining them to it, to make a single mesh.

All these options will be described in detail in the pertinent chapters.

Posing

Once you have a rigged and skinned character like Gus you can start playing with it as if it were a doll, moving its bones and looking at the results.

First, select the armature alone, then press the small button that looks like a yellow sleeping guy in the 3D Window toolbar (Figure 4-59). This button is there only if an armature is selected.

Figure 4-59. The toggle button to switch to pose mode in the 3D Window toolbar.

The button will turn "awake" (Figure 4-60) and the armature turn blue. You are in Pose Mode.

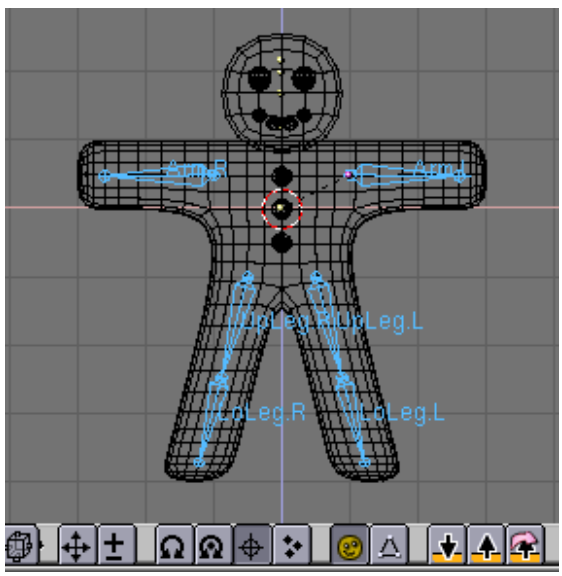


Figure 4-60. You are in pose mode now!

If you now select a bone it will turn Cyan, not Pink, and if you move it (**GKEY**), or rotate it (**RKEY**), the body will deform!

Original position: Blender remembers the original position of the bones, you can set your armature back to it by pressing the `RestPos` button in the Armature Edit Buttons (Figure 4-52).

Forward and Inverse Kinematics: Handling bones in pose mode you will notice that they act as rigid, unextensible bodies with spherical joints at the end. You can actually grab only the first bone of a chain, all the other follows, but you can rotate any of them, and all teh subsequent bones of the chain follows.

This procedure, called *Forward Kinematics* is easy to handle but makes precise location of the last bone of the chain difficult.

It is possible to use another method, called *inverse kinematics* where is the location of a special bone, usually at the end of the chain, to determine the position of all the others, hence making precise positioning of the hands and feet much easier.

Now we want Gus to walk. We will do so by defining four different poses relative to four different stages of a stride. Blender will take care of making a fluid animation by itself.

First verify that you are at frame 1 of the timeline. The frame number is in a NumBut in the far right of the Buttons Window Toolbar (Figure 4-61). If it is not at 1, set it to 1.

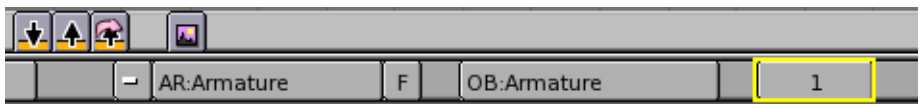


Figure 4-61. The current frame Num Button in the Buttons window Toolbar.

Now, by using only rotations on one bone at a time (**RKEY**), let's raise UpLeg.L and bend LoLeg.L backwards. Raise Arm.R a little and lower Arm.L a little, as in Figure 4-62.

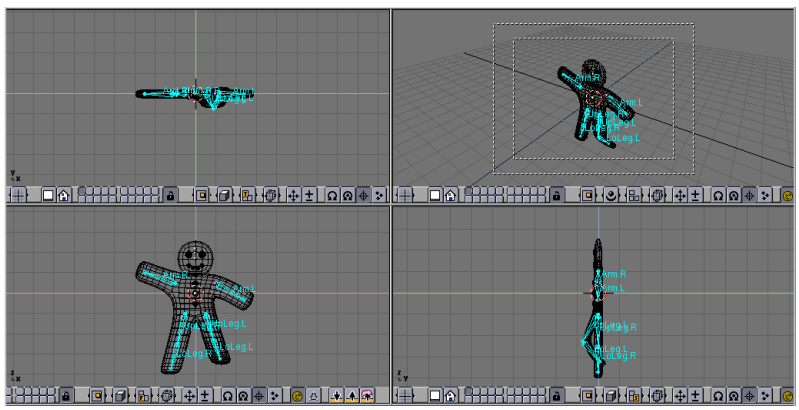


Figure 4-62. Our first pose.

Select all bones with **AKEY**. With the mouse pointer on the 3D Window press **IKEY** . A menu pops up Figure 4-63. Select `LocRot` This will get the position and orientation of all bones and store it in a pose at frame 1.

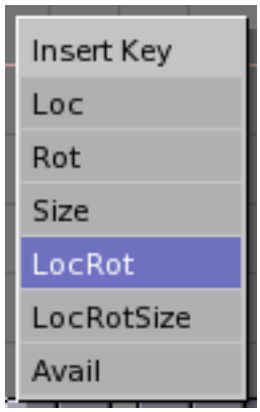


Figure 4-63. Storing the pose to the frame.

This pose represents Gus in the middle of the stride, while he is moving the left leg forward, which is above the ground.

Now move to Frame 11 either by entering the number in the NumButton or by pressing **UPARROW**

Move Gus to a different position, akin to Figure 4-64, with the left leg forward and the right leg backward, both slightly blended. Please note that Gus is walking in place!

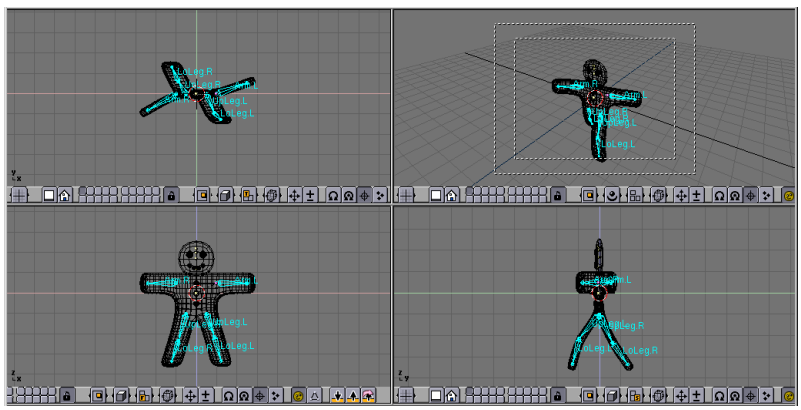


Figure 4-64. Our second pose.

Select again all bones and press **IKEY** to store this pose at frame 11.

We now need a third pose at frame 21, with the right leg up as we are in the middle of the other half of the stride.

This pose is actually the mirror-pose of the one we defined at frame 1 so, go back to frame 1 and locate the button with an arrow pointing down in the 3DWindow toolbar (Figure 4-65); press it.

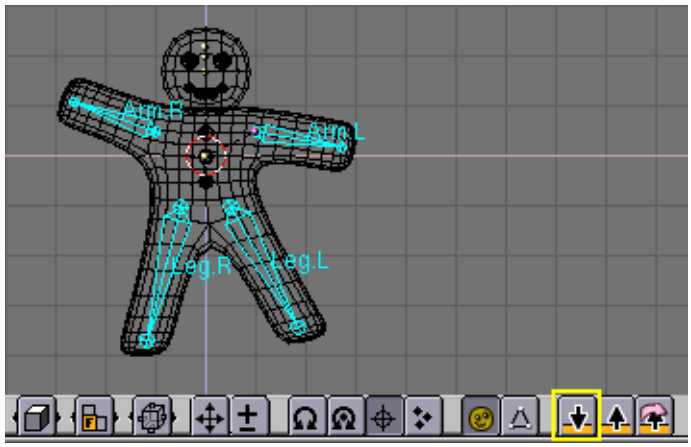


Figure 4-65. Copying the pose to the buffer

You have copied the current pose to the buffer. Go to frame 21 and paste the pose with the up arrow button with a pink arrow around it (Figure 4-66). This button will paste the cut pose exchanging the positions of bones with suffix .L with those of bones with suffix .R effectively flipping it!

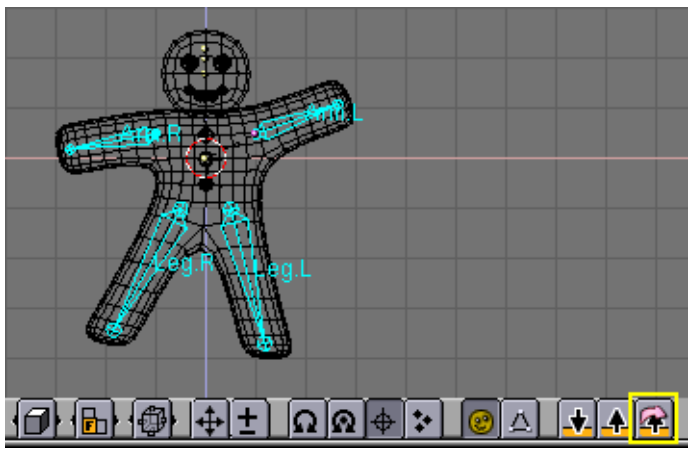


Figure 4-66. Pasting the copy as a new, flipped, pose.

Pay attention! The pose is there but it has not been stored yet! You still have to press **IKEY** with all bones selected.

Now apply the same procedure to copy the pose at frame 11 to frame 31, again flipping it.

To complete the cycle the pose at frame 1 needs to be copied, *without* flipping to frame 41. You do so by copying it as usual, and by using the button with the arrow pointing up *without* the pink arrow to paste. End the sequence storing the pose with **IKEY**.

> **Checking the animation:** You can have a quick preview of your animation by setting the current frame to 1 and pressing **Alt-A** in the 3D window.

He walks!

The single in-place step is the core of a walk, and there are techniques to make a character walk along a complex path after you have merely defined a single step, as we have done here, but for the purpose of our Quick Start this is enough.

Turn to the Rendering Buttons (**F12**) and set the Animation start and end to 1 and 40 respectively (Figure 4-67). This is because frame 41 is identical to frame 1, hence we only need to render frames from 1 to 40 to have the full cycle.

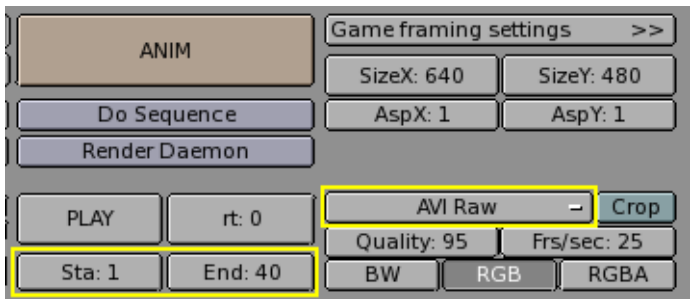


Figure 4-67. Setting the Rendering Buttons for an animation.

Now select AVI Raw as a file type (Figure 4-67). This is generally not the best choice as will be explained later on, but it is quick and it will work on any machine, so it suits our needs. You can also select AVI Jpeg, which will produce a more compact file, but using a lossy Jpeg compression.

Finally press `ANIM`. Remember that *all* the layers that you want must be shown! In our case 1 and 10.

> **Stopping a Rendering:** If you realize that you have made a mistake, like forgetting to set layer 10 to on, you can stop the rendering process with the **ESC** key.

The scene is pretty simple, and Blender will probably render each of the 40 images in few seconds. Watch them as they appear.

> **Stills:** Of course you can always render each of your animation frames as a still by selecting the frame and pressing the `RENDER` button instead.

Once the rendering is over you will have a file named `0001_0040.avi` in a `render` subdirectory of your current directory, that is the one containing your `.blend` file.

You can play it back directly within Blender by pressing the `Play` button beneath the `ANIM` button (Figure 4-67).

The animation will automatically cycle. To stop it press **ESC**.

This is just a very basic walk cycle. There is much more in Blender, just read on to discover!

Chapter 5. ObjectMode

By Martin Kleppmann

The geometry of a Blender scene is constructed from one or more Objects: Lamps, Curves, Surfaces, Cameras, Meshes, including, but not limited to, the basic objects described in the Section called *Basic objects* in Chapter 6. Each object can be moved, rotated and scaled; these operations are performed in *ObjectMode*. For more detailed changes to the geometry, you can work on the mesh of an Object in *EditMode* (see the Section called *EditMode* in Chapter 6).

After adding a basic object via **SPACE**>>Add menu, if the Object is a Mesh, a Curve or a Surface Blender changes into EditMode by default. You can change to Object-Mode by pressing **TAB**. The object's wireframe, if any, should now appear pink: That means this object is currently selected and active.

Selecting objects

Select an object by clicking it with the **RMB**. Multiple objects can be selected by holding down **SHIFT** and clicking with the **RMB**. Generally, the last object to be selected becomes the *active* object: It appears in a lighter pink, whereas the non-active selected objects appear purple. The definition of the active object is important for various issues, including parenting.

If you click the active object while **SHIFT** is pressed, it is deselected. Pressing **AKEY** selects all objects in the scene (if none are selected previously) or deselects all (if one or more is selected previously).

BKEY activates *Border select*: This allows you to draw a rectangle by holding down **LMB** and then selects all objects that lie within or touch this rectangle. Note that Border select adds to the previous selection, so if you want to be sure to select only the contents of the rectangle, deselect all with **AKEY** first. Holding down **SHIFT** while you draw the border inverts the operation: all objects within the rectangle are deselected.

Moving (translating) objects

Pressing **GKEY** activates *Grab mode* for all selected objects. These are now displayed as white wireframes and can be moved by using the mouse (without pressing any mouse button). To confirm the new position, click **LMB** or press **ENTER**; to cancel Grab mode, click **RMB** or press **ESC**. The distance of your movement is displayed in the header of your 3D Window.

You can lock movement to an axis of the global coordinate system. To do this, enter Grab mode, move the object roughly along the desired axis, and press **MMB**. Deactivate locking by pressing **MMB** again.

If you keep **CTRL** pressed while moving the object you will activate the *snap mode* and the object will move by an integer number of Blender units (grid squares). Snap mode ends when you release **CTRL** so be sure to confirm the position before releasing it.

If you are strifing for very fine and precise positioning you can try to keep **SHIFT** pressed. This way a large mouse movement reflects in a small object movement, hence allowing fine tuning.

An alternative way to enter Grab mode is to draw a straight line while holding down **LMB**. The location of selected objects can be reset to the default value by pressing **ALT-G**.

Rotating objects

To rotate objects, activate Rotate mode by pressing **RKEY**. As in Grab mode, you can now change the rotation by moving the mouse, confirm with **LMB** or **ENTER** and cancel with **RMB** or **ESC**.

Rotation in 3D space occurs around an axis, and there are various ways to define this axis. Blender defines an axis via its direction and a point that it passes through. By default, the direction of the axis is orthogonal to your screen. If you are viewing the scene precisely from the front, side or top, the rotation axis will be parallel to one of the global coordinate system axes. If you are viewing from an angle, the rotation axis is angled too, which can easily lead to a very odd rotation of your object. In this case, you may want to keep the rotation axis parallel to the coordinate system axes. Toggle this behaviour by pressing **MMB** during Rotate mode and watch the angle display in the window header.

Alternatively, once you are in rotate mode, you can press **XKEY YKEY** or **ZKEY** to constrain rotation along that axis.

The point that the rotation axis should pass through can be selected with four buttons in the header of the 3D window (Figure 5-1).

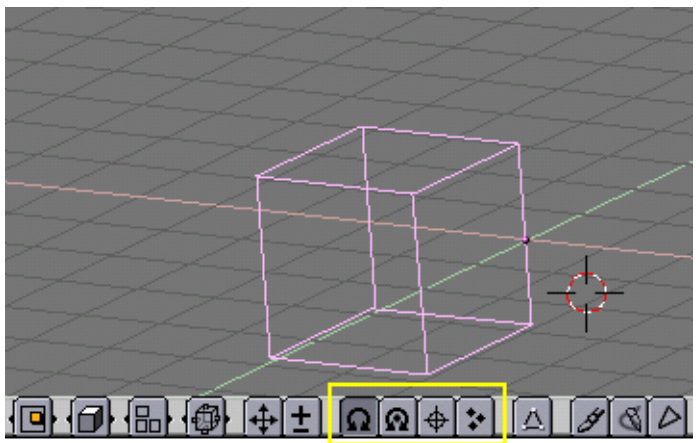


Figure 5-1. The rotation point selection buttons

- If the first button is pushed, the axis passes through the center of the selection's bounding box. (If only one object is selected, the point used is the center point of the object, which must not necessarily be in the geometric center: in Figure 5-1 it is on the middle of the rightmost edge, marked by a purple dot. For more on this point, see the Section called *EditMode* in Chapter 6.)
- If the second button is pushed, the axis passes through the median point of the selection. This difference is actually relevant only in EditMode and the 'Median' point is the barycentrum of all vertices.
- If the third button is pushed, the axis passes through the 3D cursor. The cursor can be placed anywhere you wish before rotating. Using this tool you can easily perform certain translations in the same working step as the rotation.
- If the fourth button is pushed, each selected object receives its own rotation axis; they are all parallel and pass through the center point of each object, respectively. If you select only one object, you will obviously get the same effect as with the first button.

All these details are very theoretical and not necessary if you are getting started; just play around with Blender's tools and you'll get a feeling for it.

Keeping **CTRL** pressed switch snap mode in this case too. In snap mode rotations are constrained to 5° steps. Keeping **SHIFT** pressed allows fine tuning here too.

An alternative way to enter Rotate mode is to draw a circular line while holding down **LMB**. The rotation of selected objects can be reset to the default value by pressing **ALT-R**.

Scaling/mirroring objects

To change the size of objects, press **SKEY**. As in grab mode and rotate mode, scale the objects by moving the mouse, confirm with **LMB** or **ENTER** and cancel with **RMB** or **ESC**.

Scaling in 3D space requires a center point. This point is defined with the same buttons as the axis' supporting point for rotation (Figure 5-1). If you increase the size of the object, all points are moved away from the selected center point; if you decrease it, all points move towards this point.

By default, the selected objects are scaled uniformly in all directions. To change the proportions (make the object longer, broader, etc.), the scaling process can be locked to one of the global coordinate axes, in the same way as when moving objects: Enter scale mode, move the mouse a bit in the direction of the axis you want to scale, and press **MMB**. To change back to uniform scaling, press **MMB** again. You will see the scaling factors in the header of the 3D window.

A different application of the scale tool is mirroring objects, which is effectively nothing but a scaling with a negative factor in one direction. To mirror in the direction of the *screen* X or Y axes, press **XKEY** or **YKEY**, respectively, during scale mode. If you want a precise mirroring, make sure you don't move the mouse before confirming the scaling with **LMB** or **ENTER**.

Here again **CTRL** switches to snap mode, with discrete scaling factor at 0.1 steps, while **SHIFT** allows fine tuning.

An alternative way to enter scale mode is to draw a V-shaped line while holding down **LMB**. The scaling of selected objects can be reset to the default value by pressing **ALT-S**.

The number dialog

At some point, you may want to display the effect of your object editing in numbers. Or, if you know the location, rotation and scaling values for an object, you may want to enter them directly instead of having to create them with the mouse. To do this, select the object you want to edit and press **NKEY**. The number dialog (Figure 5-2) is displayed; **SHIFT-LMB**-click a number to enter a value, press OK to confirm the changes or move the mouse outside the window to cancel.

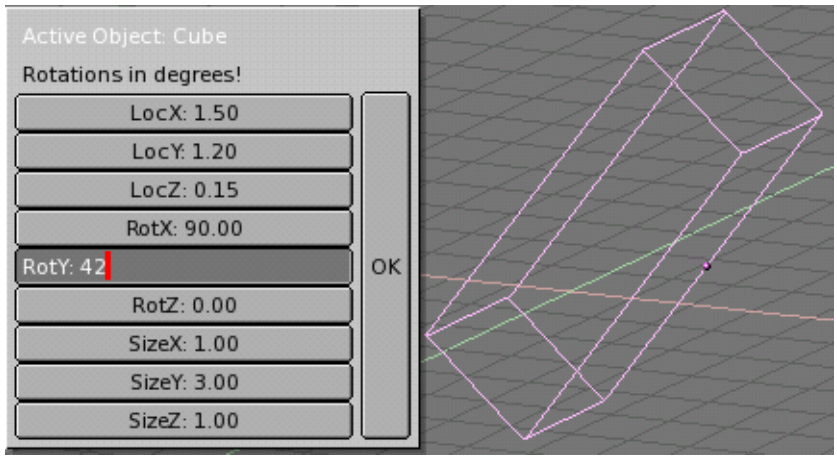


Figure 5-2. The number dialog

Duplicate

Press **SHIFT-DKEY** to create an identical copy of the selected objects. The copy is created at the same position, but is automatically in Grab mode.

This is a new object in all senses, except that it shares any Material, Texture and IPO with the original. This means that these attributes are linked to both copies and changing the material of one object also changes the material of the other.

You can make separate materials for each, as described in the Materials Chapter if you need.

You can, on the other hand, make a *Linked Duplicate* rather than a real duplicate by pressing **ALT-D**.

This will create a new Object having *all* of its data linked to the original object.This implies that if you modify one of the linked Objects in EditMode, all linked copies will be modified too.

Parenting (Grouping)

To create a group of object you need to make one of them *parent* of the others.

This is simply done by selecting at least two objects, pressing **CTRL-P** and confirm on the dialog `Make Parent?` which appears. The *active* object will be made parent of all the others. The center of all children is now linked to the center of the parent bya dashed line.

Now, Grabbing, Rotating and scaling the parent makes the children being grabbed, rotated and scaled likewise.Parenting is a very important tool and has many advanced applications which will be discussed in subsequent chapters.

By pressing **SHIFT-G** with an active object you are presented with the Group Selection menu (Figure 5-3). This contains:

- `Children` - selects all the active object childrens, and the children's childrens up to the last generation.
- `Immediate Children` - selects all the active object childrens but not these latter's childrens.
- `Parent` - selects the parent of the active object.

- `Objects on shared layers` - this actually has nothing to do with parents. It selects all objects on the same layer(s) of the active object.

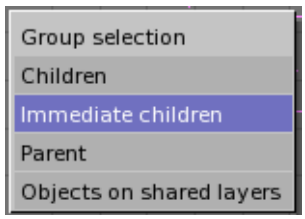


Figure 5-3. Group Select

You remove a parent relation via **ALT-P**. You can (Figure 5-4):

- `Clear parent` - frees the children, which returns to its *original* location, rotation and size.
- `Clear parent...and keep transform` - frees the children, which *keeps* the location, rotation and size given to him by the parent.
- `Clear parent inverse` - Makes the children be placed with respect to the parent as it were placed in the Global reference. This effectively clears the parent's transformation from the children.

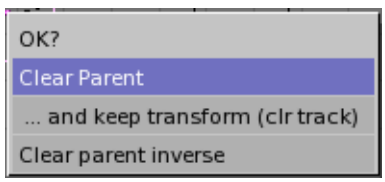


Figure 5-4. Freeing Childrens

Tracking

Is it possible to make an object rotate so that it faces another object and keep this facing even if any of them is moved.

Just select at least two objects and press **CTRL-T** and click on the `Make Track?` dialog which appears.

By default the un-active Object(s) now track the active object so that their local y axis points to the tracked object. This may not happen if the tracking object has already a rotation of its own. You can make a correct tracking by cancelling this rotation (**ALT-R**) of the tracking Object.

The orientation of the tracking Object is also chosen so that the z axis is upward. You can change this by selecting the tracking Object, switch the Button Window to Animation Buttons (F9) and selecting the Track axis from the first row of 6 Radio Buttons and the Upward-pointing axis from the second (Figure 5-5).

Figure 5-5. Setting track axis.

To clear a track constrain select the tracking object and press **ALT-T**. As for the Parent constrain clearing you must choose if you want to loose the rotation imposed by the tracking or you want to keep it.

Other Actions

Erase

Press **XKEY** or **DEL** to erase the selected objects. Using **XKEY** is more practical for most people, because it can easily be reached with the left hand on the keyboard.

Join

Press **CTRL-J** to join all selected objects to one single Object.

The Objects must be of the same type. The center point of the resulting object is obtained from the previously *active* object.

Select Links

Press **SHIFT-L** to have the possibility to select all Objects sharing a link with the active one. You can select Objects sharing an IPO, Data, Material or Texture link (Figure 5-6).

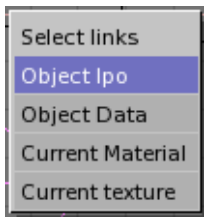


Figure 5-6. Setting track axis.

Boolean operations

The boolean operations are particular actions which can be taken only on Objects of Mesh type.

They will work for all objects but is really intended for use with solid closed objects with a well defined interior and exterior region. In the case of open objects the interior is defined in a rather mathematical way by extending the boundary faces of the object off into infinity. So results may be unexpected for these objects. A boolean operation never affects the original operands, the result is always a new blender object.

For this same reason it is very important that the normals in each object are defined consistently, and outward. Please have a look at Chapter 6 for further info on normals.

Boolean operations are invoked by selecting *exactly* two Meshes and pressing **WKEY**. There are three types of boolean operations to choose from in the popup menu, Intersect, Union and Difference

The boolean operations also take Materials and UV-Textures into account, producing objects with material indices or multi UV-mapped objects.

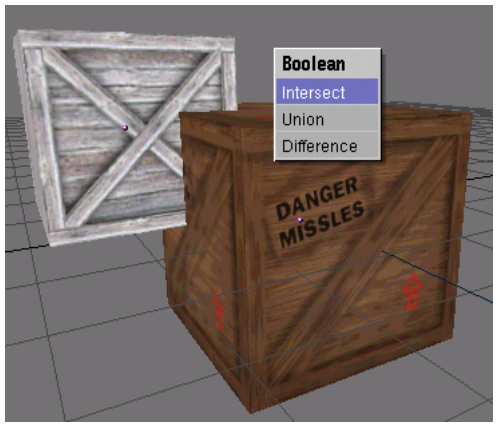


Figure 5-7. Options for boolean operations

Lets consider the object of Figure 5-7.

- The `Intersect` operation creates a new object whose surface encloses the volume common to *both* original objects.
- The `Union` operation creates a new object whose surface encloses the volume of *both* original objects.
- The `Difference` operation is the only one in which the order of selection is important. The active object (light purple in wire-frame view) is subtracted by the selected object. That is, the resulting object surface encloses a volume which is the volume belonging to the selected *and inactive* object but *not* to the selected *and active* one.

Figure 5-8 shows the results of the three operations.

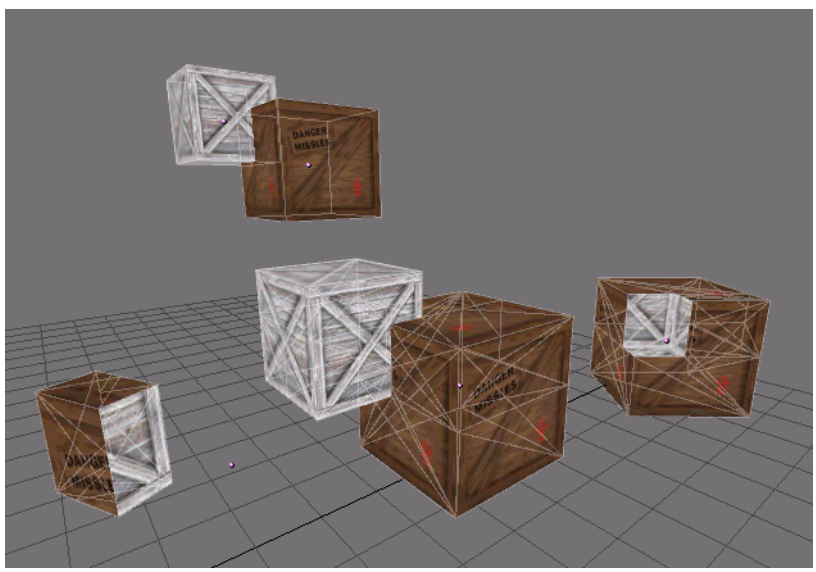


Figure 5-8. Resulting objects, original top, intersect, union, difference

The number of polygons generated can be very large compared to the original meshes. This is especially true for complex concave objects. Furthermore output

polygons can be of generally poor quality, meaning they can be very long and thin and sometimes very small, you can try the Mesh Decimator (EditButtons **F9**) to fix this

Vertices in the resulting mesh falling on the boundary of the 2 original objects often do not match up and boundary vertices are duplicated. This is good in some respects because it means you can select parts of the original meshes by selecting one vertex in the result and hitting the select linked button (**LKEY**) in Blender. Handy if you want to assign materials etc to the result.

Sometime the boolean operation can fail, a message is popped up saying ("An internal error occurred -- sorry"). Try to move or rotate the objects just a very small amount.

Chapter 6. Mesh Modelling

The principal Object of a 3D scene is usually a *Mesh*. In this chapter we will first enumerate the basic mesh objects, or *primitives*, then a long series of sections describing in detail the actions which can be taken on Mesh Objects.

Basic objects

To create a basic object press **SPACE** and select "ADD>>Mesh". You can also access the 'add'-menu by pressing **SHIFT-A**. Then you can select the basic object you'd like to create. Every basic object or *primitive* you can create within Blender is described below. Figure 6-1 also shows the different basic objects that can be created.

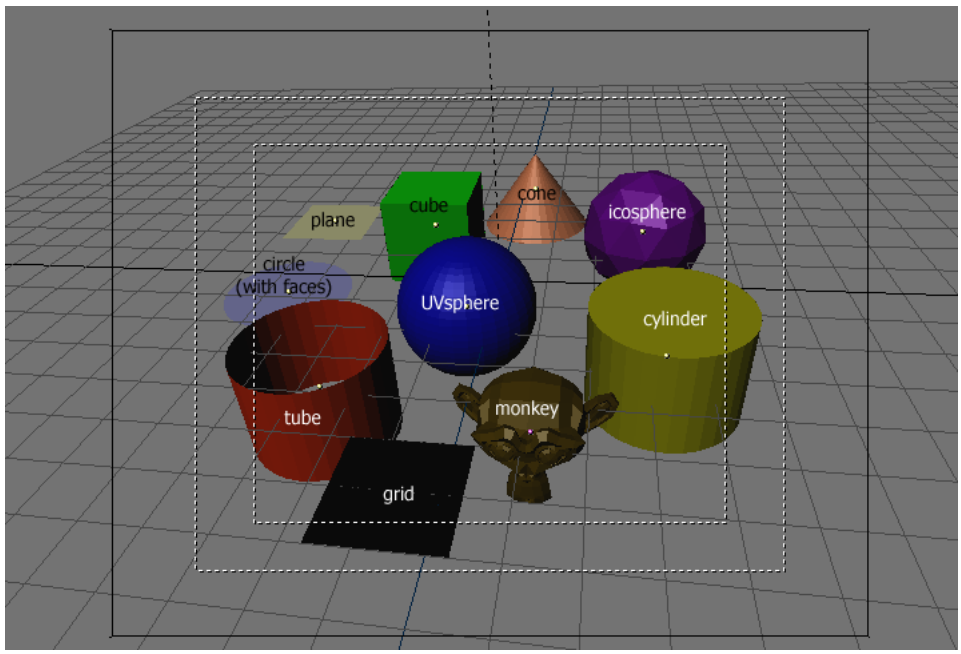


Figure 6-1. Basic Objects

Plane

A standard plane is made out of 4 vertices, 4 edges and one face. It is like a piece of paper lying on a table. A plane is not a real 3-dimensional object, because it is flat and has no 'thickness'. Example objects that can be created out planes are ground surfaces or flat objects like tabletops or mirrors.

Cube

A standard cube is made out of 8 vertices, 12 edges and 6 faces and is a real 3-dimensional object. Example objects that can be created out of cubes are dice, boxes or crates.

Circle

A standard circle is made out of *n* vertices. The number of vertices can be specified in the popup window which appears when the circle is created. The more vertices it consists of, the smoother the circle's contour becomes. Example objects that can be created out of circles are discs, plates or any kind of flat round object.

UVSphere

A standard UVsphere is made out of *n* segments and *m* rings. The level of detail can be specified in the popup window which appears when the UVsphere is created. Increasing the number of segments and rings makes the surface of the UVsphere smoother. Segments are akin to Earth meridians, rings to earth parallels. Just for your information, if you ask for a 6 segment 6 ring UVsphere you'll get something which, in top view, is a hexagon (6 segments) and has 5 rings plus two points at the poles, hence one ring less than expected, or two more, depending if you count the poles as rings of radius 0. Example objects that can be created out of UVspheres are balls, heads or pearls for a necklace.

Icosphere

An Icosphere is made up of triangles. The number of subdivisions can be specified in the window that pops up when the Icosphere is created. Increasing the number of subdivisions makes the surface of the Icosphere smoother. 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, resulting in a more 'spherical' appearance. This object is normally used to achieve a more isotropical and economical layout of vertices, in comparison to a UVsphere.

Cylinder

A standard cylinder is made out of *n* vertices. The number of vertices in the circular cross-section can be specified in the popup window that appears when the object is created. The higher the number of vertices, the smoother the circular cross-section becomes. Example objects that can be created out of cylinders are handles or rods.

Tube

A standard tube is made out of *n* vertices. The number of vertices in the hollow circular cross-section can be specified in the popup window that appears when the object is created. The higher the number of vertices, the smoother the hollow circular cross-section becomes. Example objects that can be created out of tubes are pipes or drinking glasses. The basic difference between a cylinder and a tube is that the former has closed ends.

Cone

A standard cone is made out of *n* vertices. The number of vertices in the circular base can be specified in the popup window that appears when the object is created. The higher the number of vertices, the smoother the circular base becomes. Example objects that can be created out of cones are spikes or pointed hats.

Grid

A standard grid is made out of *n* by *m* vertices. The resolution of the x-axis and y-axis can be specified in the popup window which appears when the object is created. The higher the resolution, the more vertices are created. Example objects that can be created out of grids are landscapes (with the proportional editing tool) or other organic surfaces.

Monkey

This is a gift from NaN to the community and is seen as a programmer's joke or 'Easter Egg'. It creates a monkey's head after you have pressed the 'Oooh Oooh Oooh' button.

EditMode

When working with geometric objects in Blender, you can work in two modes: ObjectMode and EditMode. Basically, as seen in the previous section, operations in Ob-

jectMode affect whole objects, and operations in EditMode affect only the geometry of an object, but not its global properties such as the location or rotation.

In Blender you switch between these two modes with the **TAB** key. EditMode only works on one object at a time: the active object. An object outside EditMode is drawn in purple in the 3D Windows (in wireframe mode) when selected, black otherwise. The active object is drawn black in EditMode, but each vertex is highlighted in purple (Figure 6-2). Selected vertices are drawn yellow (Figure 6-3).

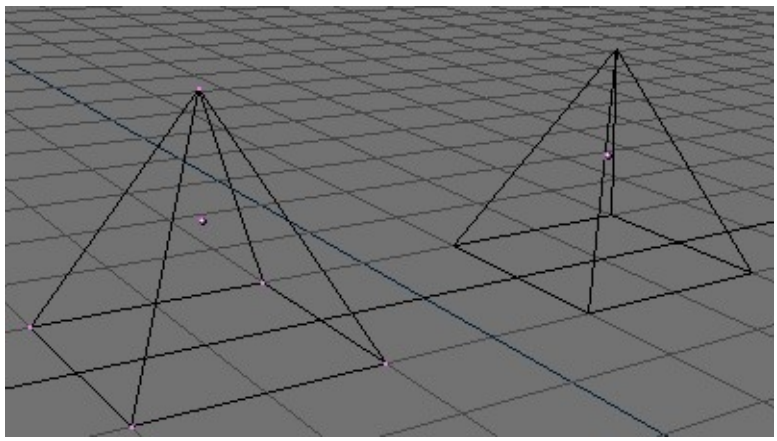


Figure 6-2. Two pyramids, one in EditMode (left) and one in ObjectMode (right).

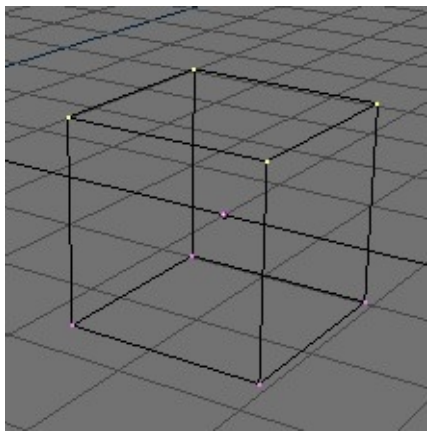


Figure 6-3. Cube with selected vertices in yellow.

Structures: Vertices, Edges and Faces

In basic meshes, everything is built from three basic structures: Vertices, Edges and Faces. (We're not talking about Curves, NURBS and so forth here.) But there is no need to be disappointed: this simplicity still provides us with a wealth of possibilities that will be the foundation for all our models.

Vertices

A vertex is primarily a single point or position in 3D space. It is usually invisible in rendering and in ObjectMode. (Don't mistake the center point of an object for a vertex. It looks similar, but bigger and you can't select it.)

Create a new vertex in EditMode by holding down **CTRL** and clicking with the **LMB**. Of course, as a computer screen is two-dimensional, Blender can't determine all three vertex coordinates from one mouse click - so the new vertex is placed at the depth of the 3D cursor 'into' the screen. If another vertex was selected previously, they are automatically connected with an Edge.

Edges

An edge always connects two vertices with 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 - their use is to construct Faces. Create an Edge by selecting two vertices and pressing **FKEY**.

Faces

A Face is the most high level structure in a mesh, building the actual surface of the object. This is what you actually see when you render the mesh. A Face is defined as the area between either three or four vertices, with an Edge on every side. Triangles always work well, because they are always flat and nicely easy to calculate. Take care when using four-sided faces, because internally they are simply divided into two triangles each. This only works well if the Face is pretty much flat (all points lie within one imaginary plane) and convex (the angle at no corner is greater than or equal to 180 degrees). This is the case with the faces of a cube, for example, which is why you can't see any diagonals its wireframe model, that would divide each square face into two triangles. It would be possible to build a cube with triangular faces, it would just look more confusing in EditMode.

An area between three or four vertices, outlined by Edges, doesn't have to be a face. If no face was created, this area will simply be transparent or non-existant in the rendered image. To create a face, select three or four suitable vertices and press **FKEY**.

Basic

Most simple operations from ObjectMode (selecting, moving, rotating, scaling) work identically on vertices as they do on objects. Thus, you can learn how to handle basic EditMode operations very quickly. The truncated pyramid in Figure 6-4, for example, was created with the following steps:

1. Add a cube to an empty scene. Enter EditMode.
2. Make sure all vertices are deselected (purple). Use border select (**BKEY**) to select the upper four verices.
3. Check that the scaling center is set to *anything but* the 3D cursor (see Figure 5-1), switch to scale mode (**SKEY**), reduce the size and confirm with **LMB**.
4. Exit EditMode by pressing **TAB**.

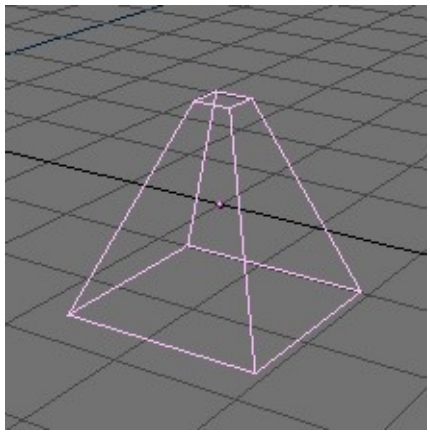


Figure 6-4. Chopped-off pyramid

One additional feature of EditMode is the CircleSelect mode. It is invoked by pressing **BKEY** twice instead of only once, as you would for BorderSelect. A light gray circle is drawn around the cursor and any **LMB** click selects all vertices within. **NUM+** and **NUM-** or the **MW**, if any, enlarge or shrink the circle.

All operations in EditMode are in the end performed on the Vertices; the connected Edges and Faces automatically adapt, as they depend on the Vertices' positions. To select an Edge, you must select the two endpoints or either place the mouse on the edge and press **CTRL-ALT-MMB**. To select a Face, each corner must be selected.

Edit Mode operations are many, and most are summarized in the Edit Buttons window, which can be accessed via the () button of the Button Window Toolbar or via **F9** (Figure 6-5). In this Window it is important to note, at the moment, the group of buttons in the lower left corner:

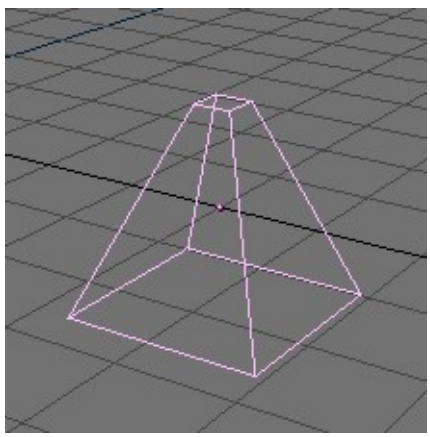


Figure 6-5. Chopped-off pyramid

- `NSize:` - Determines the length, in Blender Units, of the normals to the faces, if these are drawn.
- `Draw Normals` - Toggle Normals drawing. If ON Face Normals are drawn as cyan segments.
- `Draw Faces` - If ON faces are drawn semi-transparent blue, or semi-transparent purple if selected. If OFF faces are invisible.

- `Draw Edges` - Edges are always drawnblack, but if this button is ON then selected edges are drawn in yellow. Edges joining a selected node and an un-selected one have a yellow-black gradient.
- `All Edges` - In Object Mode not all Edges are shown, but only those strictly necessary to show the Object shape. You can force BLender to draw all edges with this button.

With **WKEY** you can call up the "Specials" menu in EditMode (Figure 6-6). With this menu you can quickly access functions which are frequently required for polygon-modelling. You will find the same functionality in the EditButtons **F9**.

> **Tip:** You can access the entries in a PopupMenu by using the corresponding numberkey. For example, the keypresses **WKEY**, **1KEY**, will subdivide the selected vertices without you having to touch the mouse.

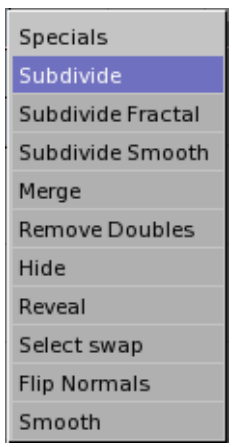


Figure 6-6. Specials Menu

- `Subdivide` - Each selected edge is split in two, new vertices are created at middle points and faces are splitted too, if necessary.
- `Subdivide Fractal` - As above, but new vertices are randomly displaced within a user-defined range.
- `Subdivide Smooth` - As above, but new vertices are displaced towards the baricentrum of the connected vertices.
- `Merge` - Merges selected vertices in a single one, at the baricentrum position or at the cursor position.
- `Remove Doubles` - Merges all of the selected vertices whose relative distance is under a given threshold, by default 0.001.
- `Hide` - Hides selected vertices.
- `Reveal` - Shows hidden vertices.
- `Select Swap` - All selected vertices become unselected and vice-versa.
- `Flip Normals` - Change the Normals directions in the selected faces.
- `Smooth` - Smooth out a mesh by moving each vertex towards the baricentrum of the linked vertices.

It is worth noting that many of these actions have a button of their own in the Mesh Edit Buttons Window (Figure 6-5). Here the `Remove doubles` threshold can be adjusted too.

Smoothing

Most objects in Blender are represented by polygons. In Blender, truly curved objects are often approximated by polygon meshes. When rendering images you may notice that these objects appear as a series of small flat facets. (Figure 6-7). Sometimes this is a desirable effect, but usually we want our objects to look nice and smooth. This section guides you through the steps of smoothing an object and applying the AutoSmooth filter to quickly and easily combine smooth and faceted polygons in the same object.

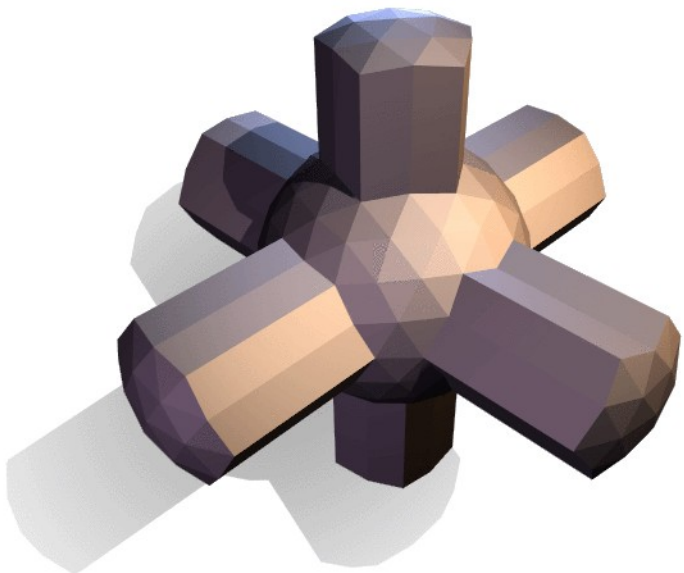


Figure 6-7. Simple un-smoothed test object

There are two ways of activating the face smoothing features of Blender. The easiest way is to set an entire object as smooth or faceted. This can be accomplished by selecting a mesh object, switching to the EditButtons window (**F9**) and clicking the Set Smooth button shown in Figure 6-8 You will notice that the button does not stay pressed, but Blender has assigned the "smoothing" attribute to each face in the mesh. Rendering an image with **F12** should produce the image shown in Figure 6-9. Notice that the outline of the object is still strongly faceted. Activating the smoothing features doesn't actually modify the object's geometry. Instead it changes the way the shading is calculated across the surfaces, giving the illusion of a smooth surface.

Clicking the Set Solid button reverts the shading to that shown in Figure 6-7.

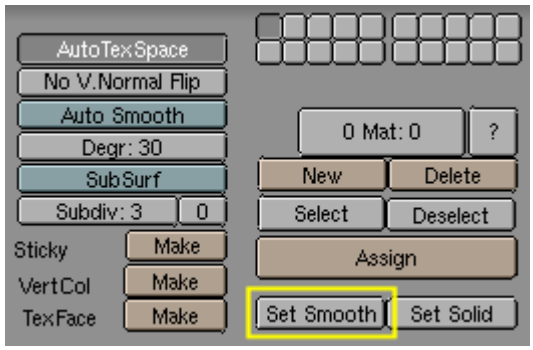


Figure 6-8. Set Smooth and Set Solid buttons of EditButtons window

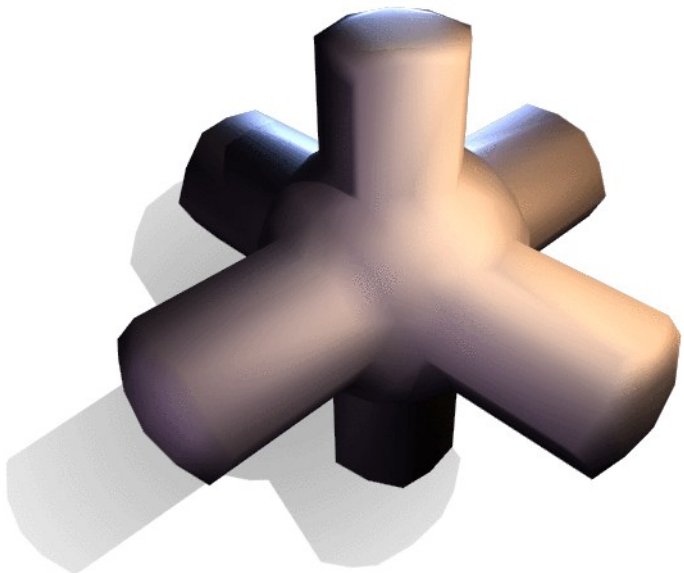


Figure 6-9. Same object as above, but completely smoothed by 'Set Smooth'

An alternate method of selecting which faces to smooth can be done by entering editmode for the object with **TAB**, selecting faces and clicking the Set Smooth button (Figure 6-10). When the mesh is in editmode, only faces that are selected will receive the "smoothing" attribute. You can set solid faces (removing the "smoothing" attribute) in the same way: by selecting faces and clicking the Set Solid button.

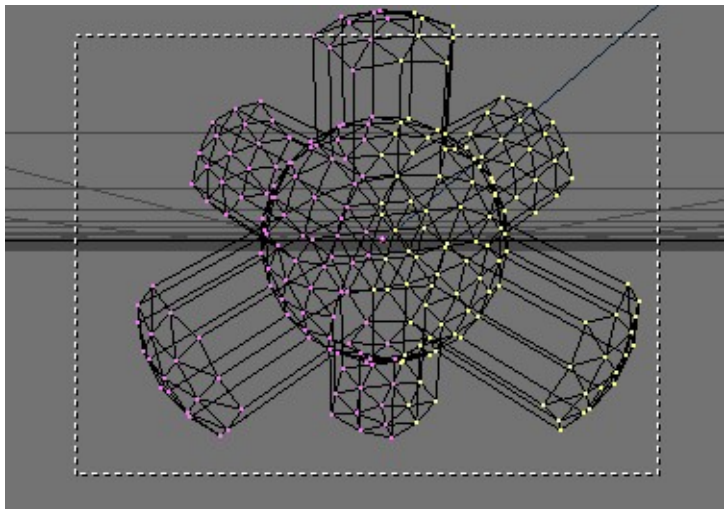


Figure 6-10. Object in editmode with some faces selected.

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 **YKEY**), there is an easier way to combine smooth and solid faces.

Pressing the AutoSmooth button in the EditButtons (Figure 6-11) causes Blender to decide which faces should be smoothed and which ones shouldn't, based on the angle between faces (Figure 6-12). Angles on the model that are sharper than the angle specified in the "Degr" NumBut will not be smoothed. You can change this value to adjust the amount of smoothing that occurs in your model. Higher values will produce more smoothed faces, while the lowest setting will look identical to a mesh that has been set completely solid.

Only faces that have been been set as smooth will be affected by the AutoSmooth feature. A mesh, or any faces that have been set as solid will not change their shading when AutoSmooth is activated. This allows you extra control over which faces will be smoothed and which ones won't by overriding the decisions made by the AutoSmooth algorithm.

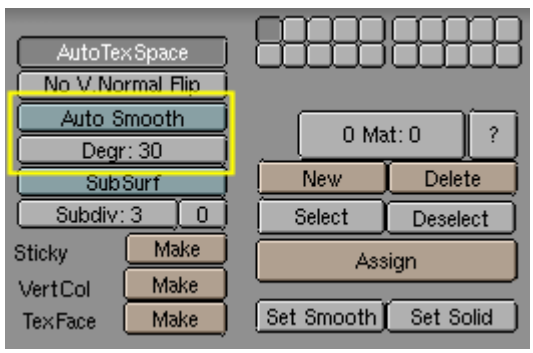


Figure 6-11. AutoSmooth button group in the EditButtons window.

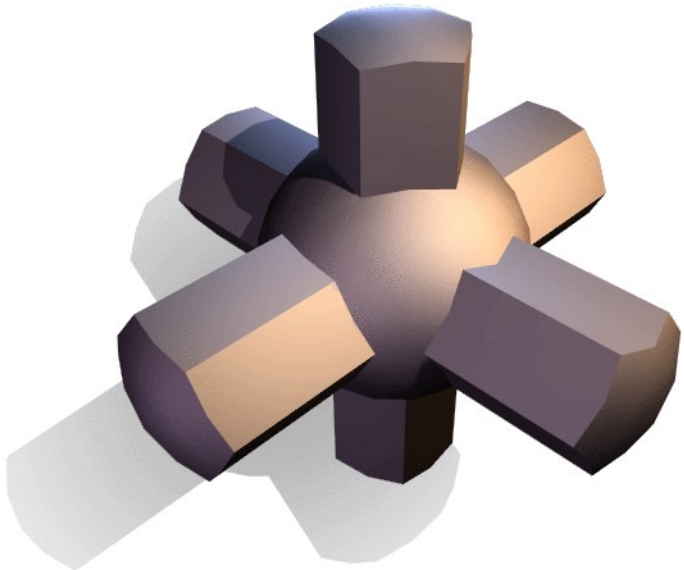


Figure 6-12. Same test object with AutoSmooth enabled

Proportional Editing Tool

When working with dense meshes, it can become difficult to make subtle adjustments to the vertices without causing nasty lumps and creases in the model's surface. The proportional editing tool works like a magnet to smoothly deform the surface of the model.

In a top-down view, add a plane mesh to the scene with **SPACE**>>MESH>>PLANE. Subdivide it a few times with **WKEY**>>SUBDIVIDE (or by clicking on the `Subdivide` button in the EditButtons) to get a relatively dense mesh (Figure 6-13). Otherwise you can directly add a grid via **SPACE**>>MESH>>GRID specifying the number of vertices in each direction. When you are done, deselect all vertices with **AKEY**.

> **Vertex limit:** There is a limit on how many vertices a single mesh can have. This limit is equal to 65000.

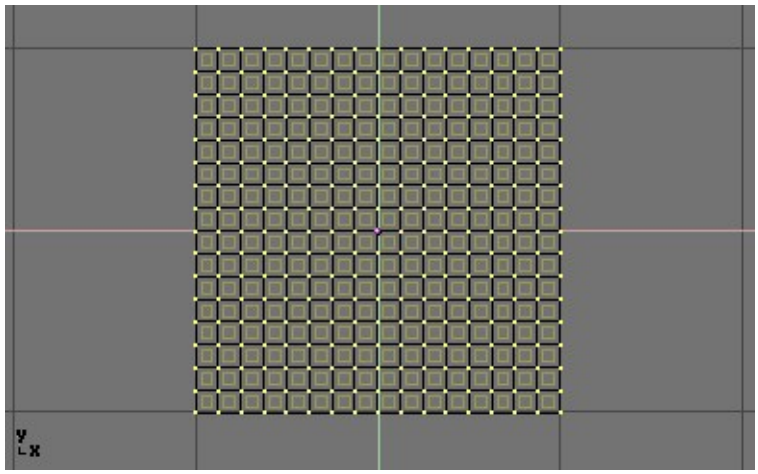


Figure 6-13. A planar dense mesh.

Select a single vertex in the mesh by clicking it with the right mousebutton (Figure 6-14).

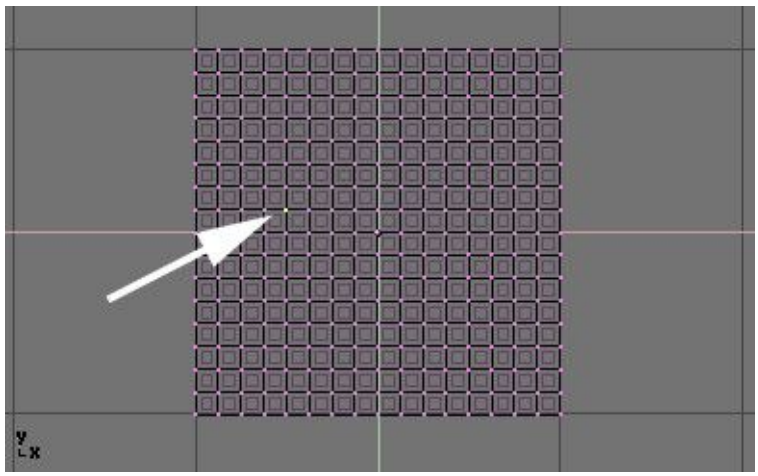


Figure 6-14. A planar dense mesh with just one selected vertex.

Still in EditMode, activate the proportional editing tool by pressing **OKEY** or by clicking on the grid icon in the header bar of the 3dWindow (Figure 6-15 top).

> **Header Bar panning:** If the icon isn't visible in the header bar because your window is too narrow, you can pan the header bar left-right by **MMB** on it and drag left or right.

You should see the icon change to a distorted grid with two curve-shape buttons next to it (Figure 6-15 bottom).

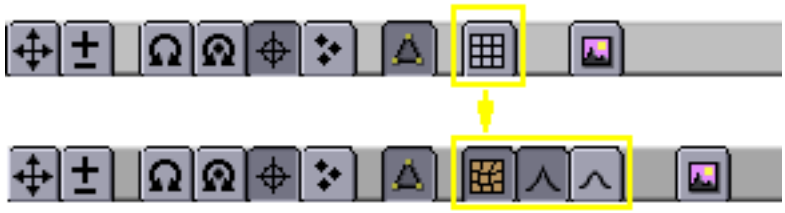


Figure 6-15. Proportional Editing icon and schemes

Switch to a front view (**NUM 1**) and activate the move tool with **GKEY**. As you drag the point upwards, notice how other nearby vertices are dragged along with it in a curve similar to the one selected in the header bar (Figure 6-16).

You can change which curve profile is used by either clicking on the corresponding icon in the header bar, or by pressing **SHIFT-O**. Note that you cannot do this while you are in the middle of a proportional editing operation; you will have to press **ESC** to cancel the editing operation before you can change the curve.

When you are satisfied with the placement of the vertex, you can press **LMB** to fix its position or, if you are not satisfied, nullify the operation and revert your mesh to the way it looked before you started dragging the point with **ESC** key.

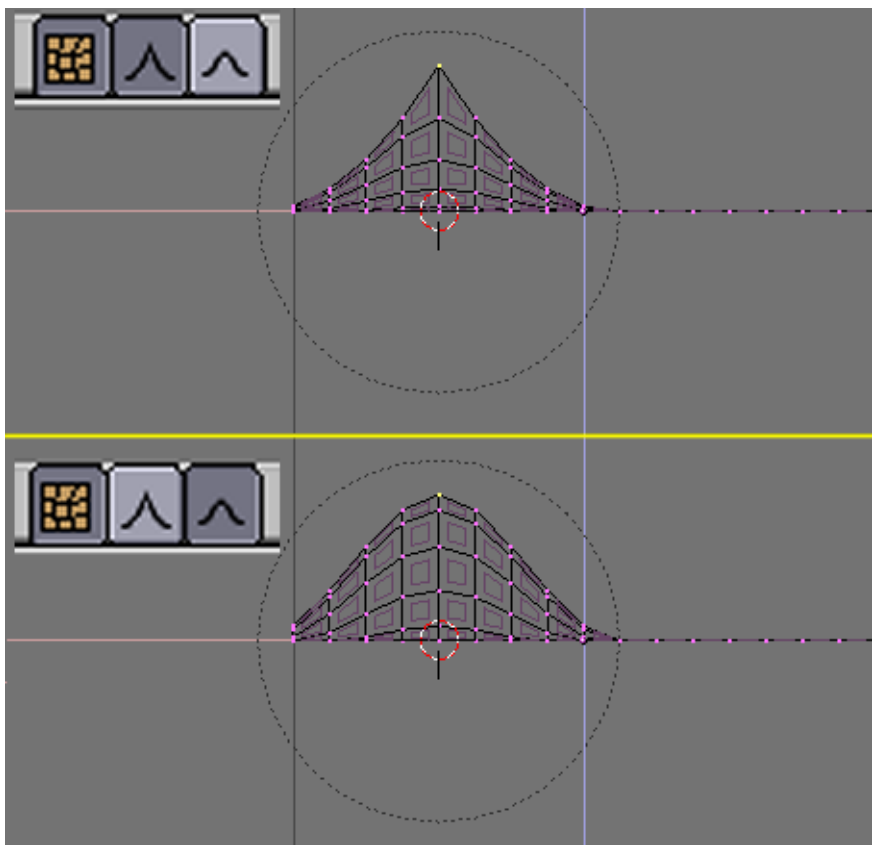


Figure 6-16. Different 'Magnets' for proportional Editing.

You can increase or decrease the radius of influence (shown by the dotted circle in Figure 6-16) while you are editing by pressing **NUM+** and **NUM-** respectively. As you change the radius, you will see the points surrounding your selection adjust

their positions accordingly. Alternatively, you can use **MW**it to enlarge and shrink the circle.

You can get great effects using the proportional editing tool with scaling (**SKEY**) and rotation (**RKEY**) tools, as Figure 6-17 shows.

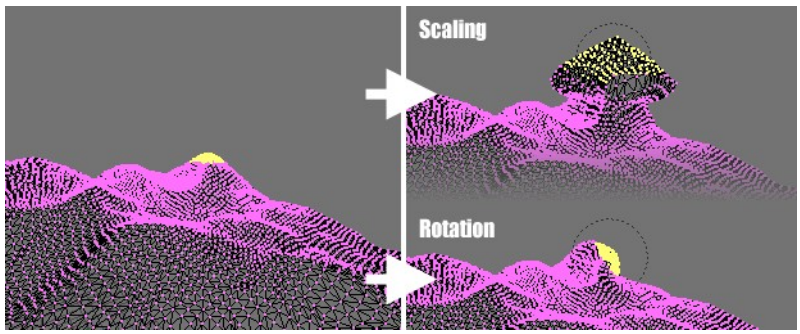


Figure 6-17. A landscape obtained via Proportional Editing

Combine these techniques with vertex painting to create fantastic landscapes.

Figure 6-18 shows the results of proportional editing after the application of textures and lighting.

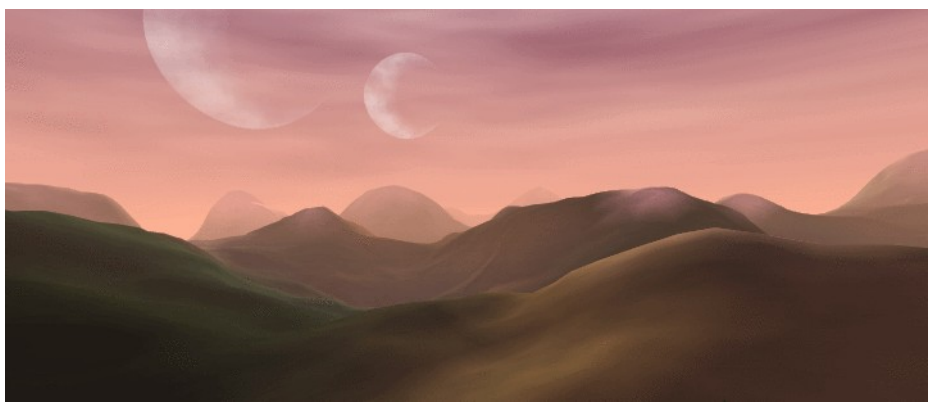


Figure 6-18. Final rendered landscape

Extrude

A tool of paramount importance for working with Meshes is the "Extrude" command (**EKEY**). This command allows you to create cubes from rectangles and cylinders from circles. Extrude allows you create things such as tree limbs very easily. Although the process is quite intuitive, the principle behind Extrude is outlined below:

- First, the algorithm determines the outside edge-loop of the Extrude, *i.e.* which among the selected edges will be changed into faces. By default, 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, e.g. rectangles result in cubes 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.
- Edges not belonging to selected faces, hence forming an 'open' edge-loop are simply duplicated and a new face is created between the new edge and the original one.
- Single selected vertices, not belonging to selected edges are duplicated and a new edge created between the two.

Grab mode is automatically started when the Extrude algorithm terminates, so newly created faces/edges/vertices can be moved around with the mouse.

Extrude is one of the most frequently used modelling tools in Blender. It's simple, straightforward and easy to use, yet very powerful. The following short lesson describes how to build a sword using Extrude.

The Blade

Start Blender and delete the default plane. In top view add a mesh circle with 8 vertices. Move the vertices so they match the configuration shown in Figure 6-19.

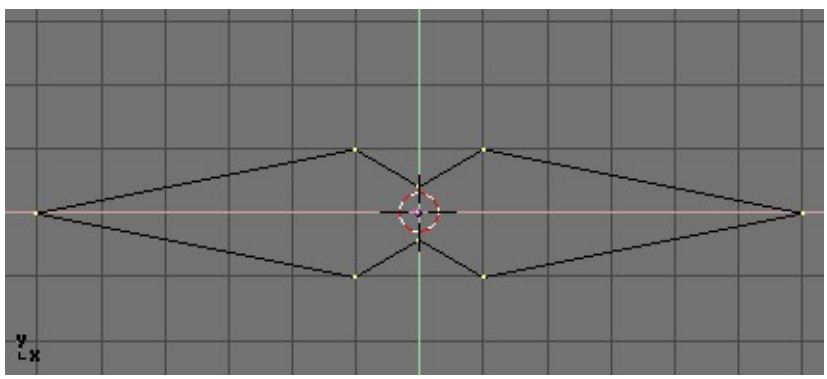


Figure 6-19. Deformed circle, to become the blade cross section.

Select all the vertices and scale them down with the **SKEY** so the shape fits in 2 grid units. Switch to front view with **NUM 1**.

The shape we've created is the base of the blade. Using Extrude we'll create the blade in a few simple steps. With all vertices selected press **EKEY**, or click the button labelled `extrude` in the EditButtons (**F9** - Figure 6-20). A box will pop up asking `Ok? Extrude` (Figure 6-21).

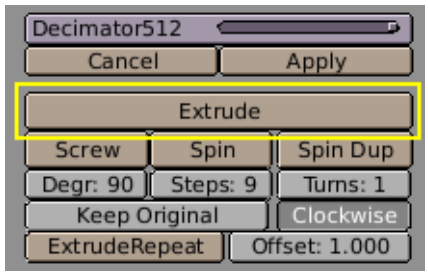


Figure 6-20. Extrude button in EditButtons window

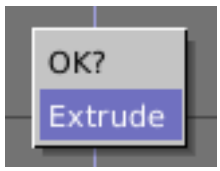


Figure 6-21. Extrude confirmation box.

Click it or press **ENTER** to confirm, move the mouse outside or press **ESC** to exit. If you move the mouse now you'll see the following has happened: Blender has duplicated the vertices, connected them to the original ones with edges and faces, and has entered grab mode. Move the new vertices up 30 units, constraining the movement with **CTRL**. Click the left mousebutton to confirm their new position and scale them down a little bit with the **SKEY** (Figure 6-22).

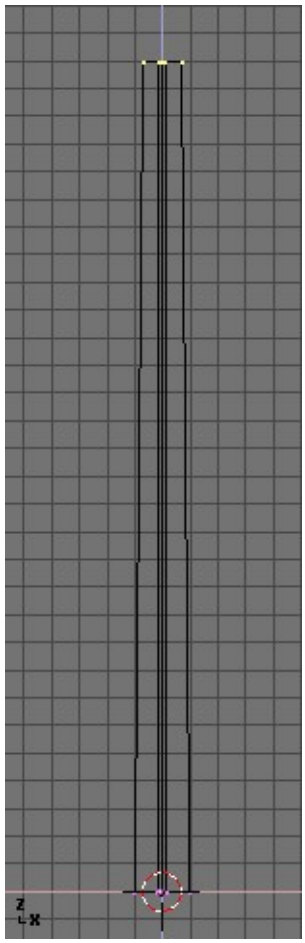


Figure 6-22. The Blade

Press **EKEY** again to extrude the tip of the blade. Move the vertices five units up. To make the blade end in one vertex, scale the top vertices down to 0.000 (hold **CTRL** for this) and press **WKEY**>`Remove Doubles` (Figure 6-23) or click the `Rem Doubles` button in the EditButtons (**F9**). Blender will inform you it removed seven of the eight vertices and only one vertex remains: the blade is done! (Figure 6-24)

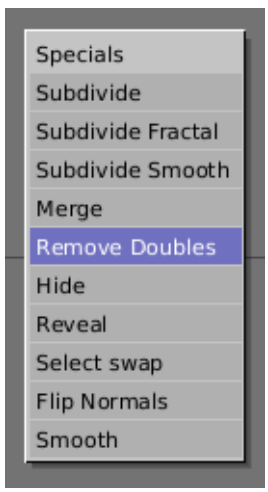


Figure 6-23. Mesh Edit Menu

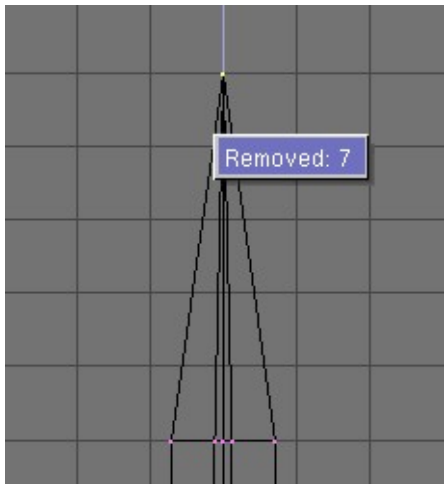


Figure 6-24. The completed blade

The Handle

Leave edit mode and move the blade to the side. Add a UVsphere with 16 segments and rings and deselect all the vertices with the **AKEY**. Borderselect the top three rings of vertices with **BKEY** and delete them with **XKEY**>`Vertices` (Figure 6-25).

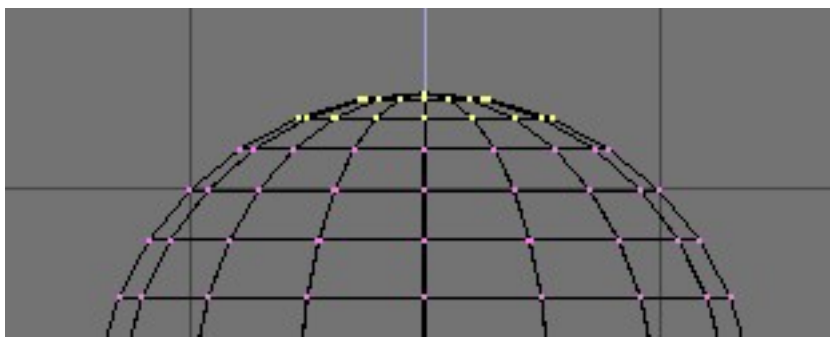


Figure 6-25. UV sphere for the handle: vertices to be removed

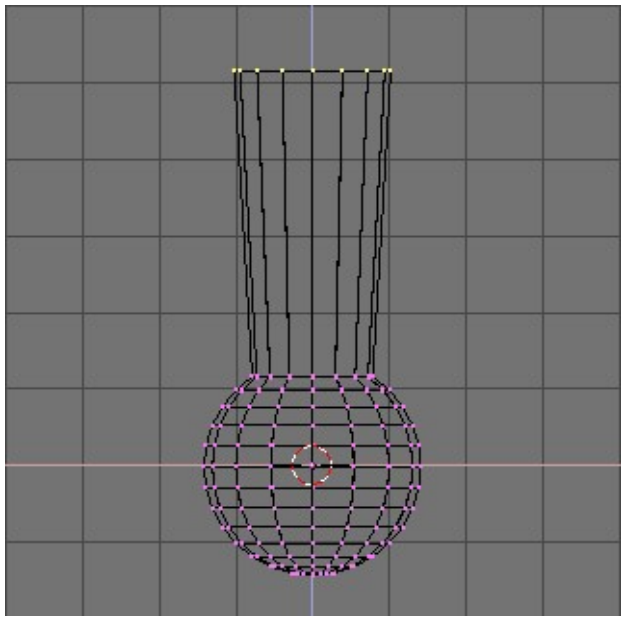


Figure 6-26. First extrusion for the handle

Select the top ring of vertices and extrude them. Move the ring up four units and scale them up a bit (Figure 6-26), extrude and move 4 units again twice and scale the last ring down a bit (Figure 6-27). Leave EditMode and scale the entire handle down so it's in proportion with the blade. Place it just under the blade.

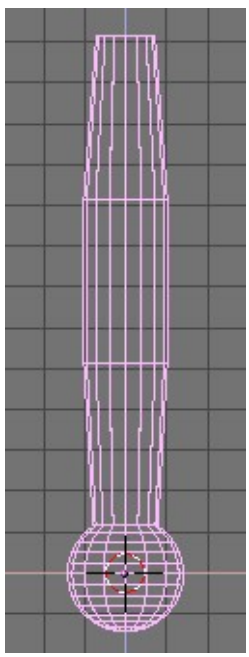


Figure 6-27. Complete handle

The Hilt

By now you should be used to the 'extrude>move>scale' sequence, so try to model a nice hilt with extrude. Start out with a cube and extrude different sides a few times,

scaling them where needed. You should be able to get something like that shown in Figure 6-28.

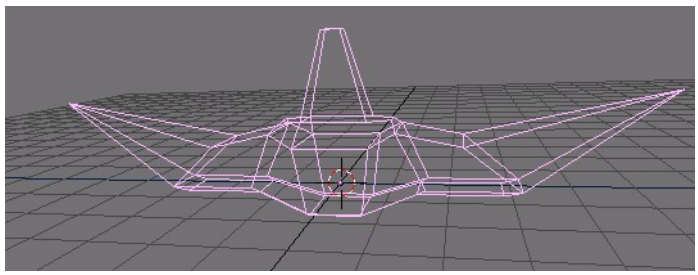


Figure 6-28. Complete Hilt

After texturing, the sword looks like Figure 6-29

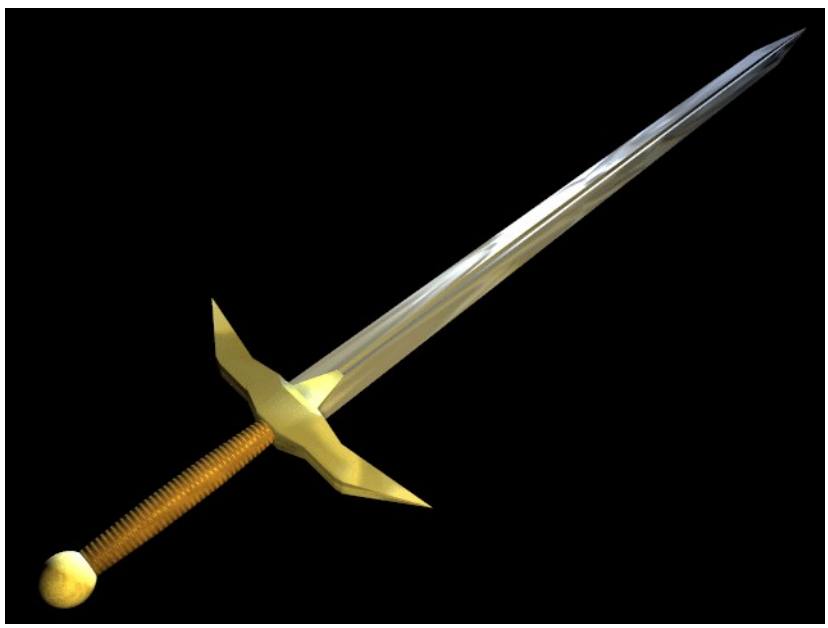


Figure 6-29. Finished sword, with textures and materials

As you can see, extrude is a very powerful tool allowing you to model relatively complex objects very quickly (the entire sword was created in less than a half hour!). Getting the hang of extrude>move>scale will make your life as a Blender modeller a lot easier.

Spin and SpinDup

The spin and spin dup are two other very powerful modelling tools.

Spin

The Spin tool in Blender is for creating the sort of objects that you can produce on a lathe. This tool is therefore often called a "lathe"-tool or a "sweep"-tool in the literature.

First you must create a mesh representing the profile of your object. If you are modelling a hollow object, it is a good idea to give a thickness to the outline. Figure 6-30 shows the profile for a wine glass we will model to demonstrate this tool.

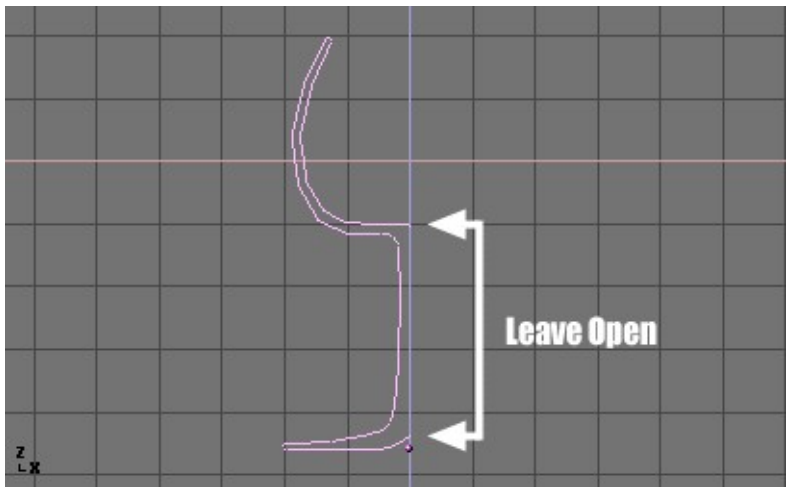


Figure 6-30. Glass profile

In EditMode, with all the vertices selected, access the EditButtons window (**F9**). The `Degr` button indicates the number of degrees to spin the object (in this case we want a full 360° sweep). The `Steps` button specifies how many profiles there will be in the sweep (Figure 6-31).

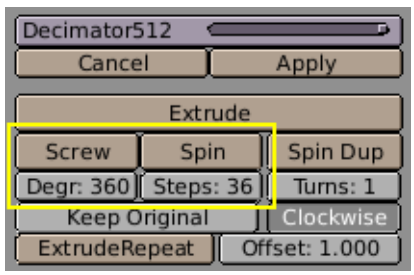


Figure 6-31. Spin Buttons

Like Spin Duplicate (covered by the next section), the effects of Spin depend on the placement of the cursor and which window (view) is active. We will be rotating the object around the cursor in the top view. Switch to the top view with **NUM 7**. The cursor should be placed along the center of the profile. This is easily accomplished by selecting one of the vertices along the center, and snapping the cursor to that location with **SHIFT+S**>>CURS->SEL.

Figure 6-32 shows the wine glass profile from top view, with the cursor correctly positioned.

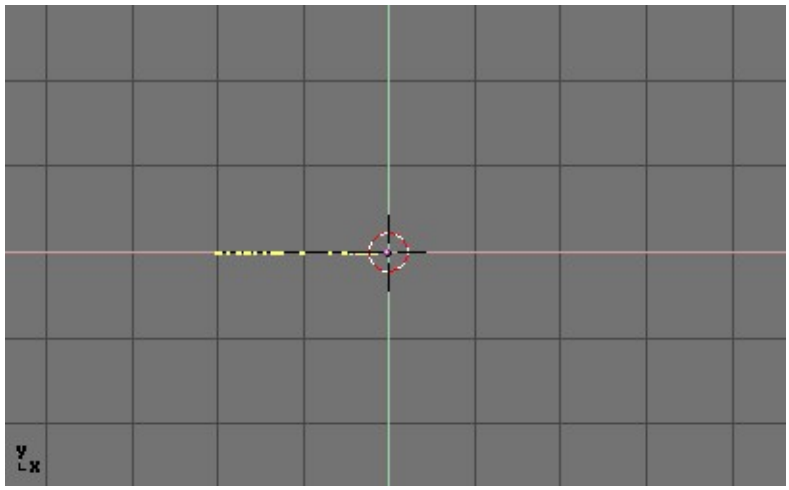


Figure 6-32. Glass profile, top view in edit mode, just before spinning.

Before continuing, make a note of the number of vertices in the profile. This information can be found in the Info bar at the top of the Blender interface (Figure 6-33).

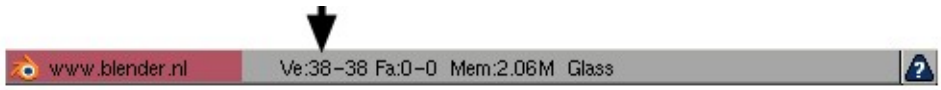


Figure 6-33. Mesh data - Vertex and face numbers.

Click the "Spin" button. If you have more than one window open, the cursor will change to an arrow with a question mark and you will have to click in the window containing the top view before continuing. If you have only window open, the spin will happen immediately.

Figure 6-34 shows the result of a successful spin.

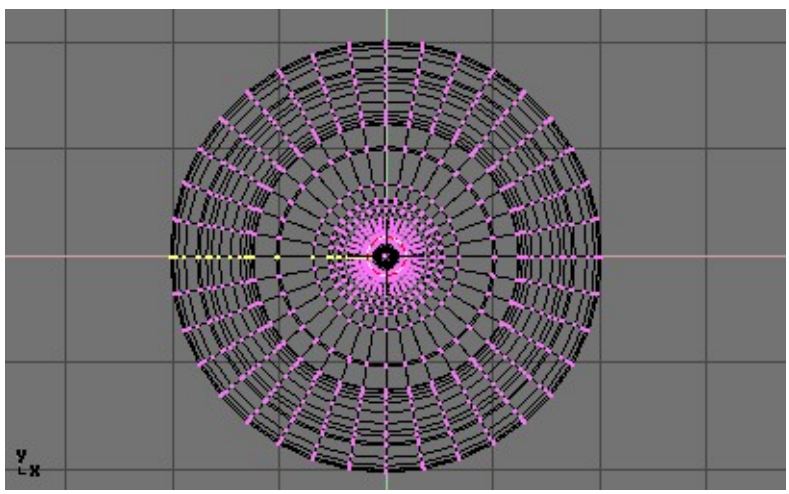


Figure 6-34. Spinned profile

The spin operation leaves duplicate vertices along the profile. You can select all vertices at the seam with Box select (**BKEY**) (Figure 6-35) and do a Remove Doubles operation.

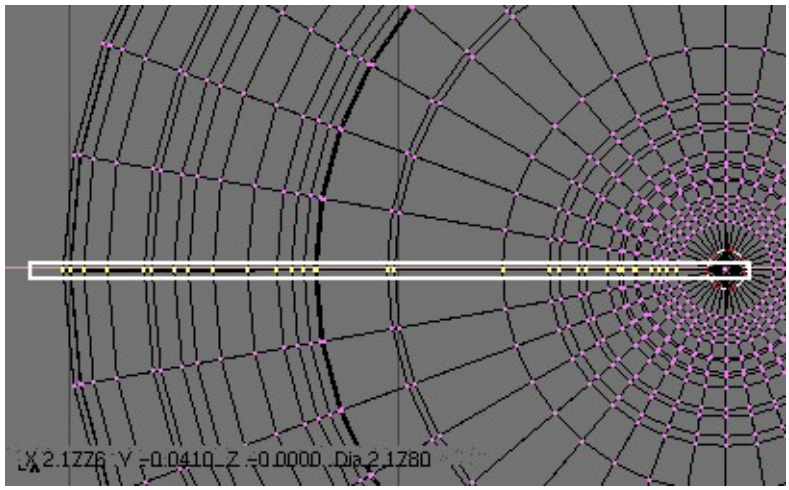


Figure 6-35. Seam vertex selection

Notice the selected vertex count before and after the Remove Doubles operation (Figure 6-36). If all goes well, the final vertex count (38 in this example) should match the number of the original profile noted in Figure 6-33. If not, some vertices were missed and you will have to go and weld them manually. Or, worse, too many vertices have been merged.

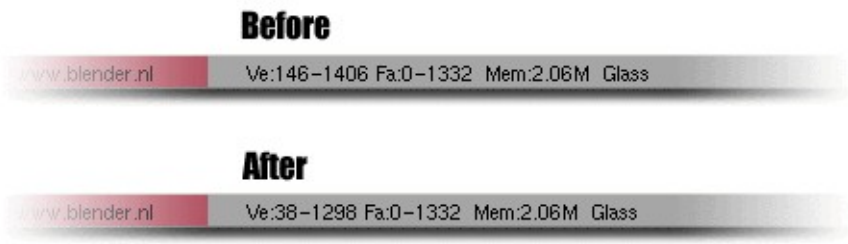


Figure 6-36. Vertex count after removing doubles.

Merging two vertices in one: To merge (weld) two vertices together, select both of them by holding **SHIFT** and **RMB** on them. Press **SKEY** to start scaling and hold down **CONTROL** 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 EditButtons window.

Alternatively, you can press **WKEY** and select "Merge" from the appearing Menu (Figure 6-37). You can then choose in a new menu if the merged node will have to be at the center of the selected nodes or in the location of the cursor. The first choice is better in our case.

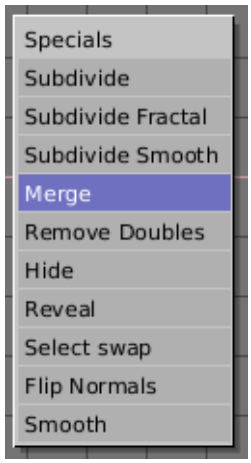


Figure 6-37. Merge menu

All that remains now is to recalculate the normals by selecting all vertices and pressing **CONTROL+N**>>`Recalc Normals Outside`. At this point you can leave Edit-Mode and apply materials or smoothing, set up some lights, a camera and make a rendering. Figure 6-38 shows our wine glass in a finished state.

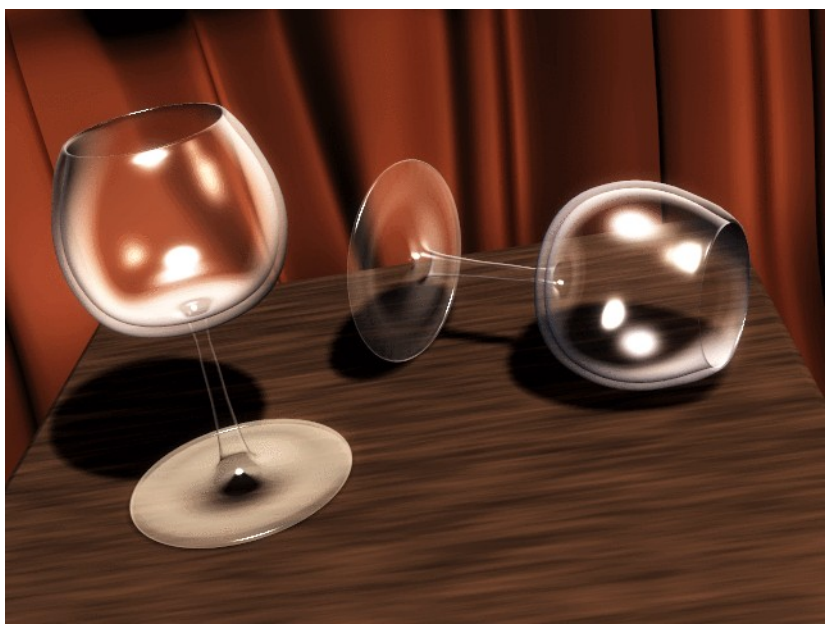


Figure 6-38. Final render of the glasses.

SpinDup

The Spin Dup tool is a great way to quickly make a series of copies of an object laid out in a circular pattern. Let's assume you have modelled a clock, and you now want to add hour marks.

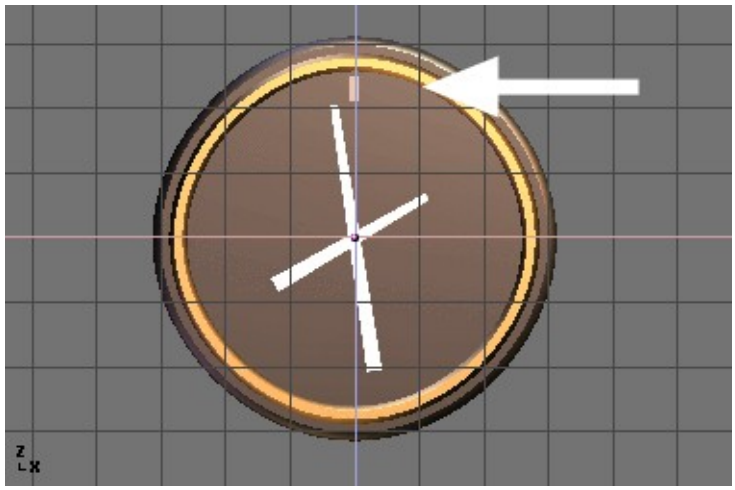


Figure 6-39. Hour mark indicated by the arrow

Model just one mark, in the 12 o'clock position (Figure 6-39). Select the mark and switch to the EditButtons window with **F9**. Set the number of degrees in the `Degr` NumBut to 360. We want to make 12 copies of our object, so set the "Steps" to 12 (Figure 6-40).

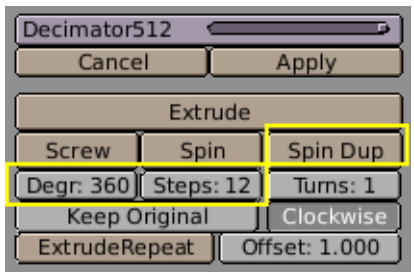


Figure 6-40. Spin Dup buttons

- Switch the view to the one in which you wish to rotate the object by using the keypad. Note that the result of the `Spin Dup` command depends on the view you are using when you press the button.
- Position the cursor at the center of rotation. The objects will be rotated around this point.
- Select the object you wish to duplicate and enter EditMode with **TAB**.
- In EditMode, select the vertices you want to duplicate (note that you can select all vertices with **AKEY** or all of the vertices linked to the point under the cursor with **LKEY**) See Figure 6-41.

> **Cursor Placement:** If you want to place the cursor at the precise location of an existing object or vertex, select the object or vertex, and press **SHIFT+S**>>CURS>>SEL.

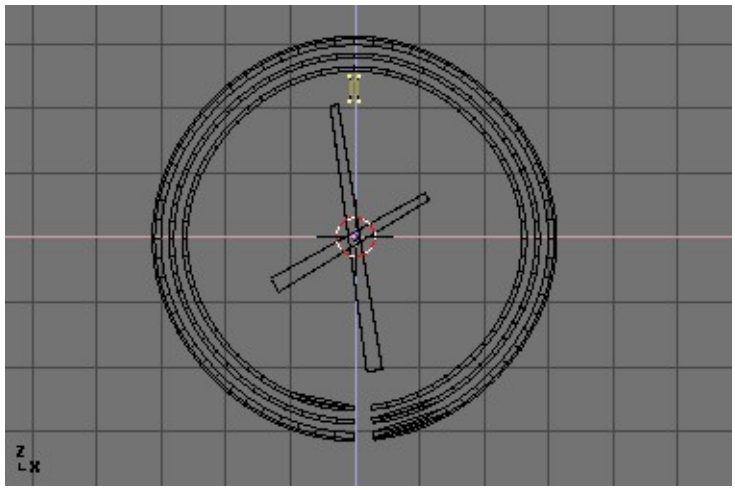


Figure 6-41. Mesh selected and ready to be SpinDuped

- Press the Spin Dup button. If you have more than one 3DWindow open, you will notice the mouse cursor change to an arrow with a question mark. Click in the window in which you want to do your rotation. In this case, we want to use the front window (Figure 6-42).

If the view you want is not visible, you can dismiss the arrow/question mark with **ESC** until you can switch a window to the appropriate view with the keypad

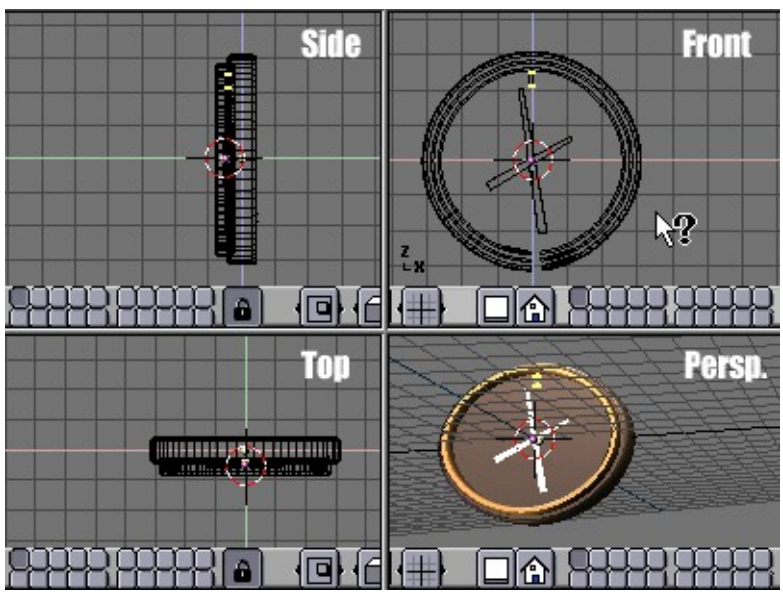


Figure 6-42. View selection for Spin Dup.

When spin-duplicating an object 360 degrees, a duplicate object is placed at the same location of the first object, producing duplicate geometry. You will notice that after clicking the `Spin Dup` button, the original geometry remains selected. To delete it, simply press **XKEY**>>VERTICES. The source object is deleted, but the duplicated version beneath it remains (Figure 6-43).

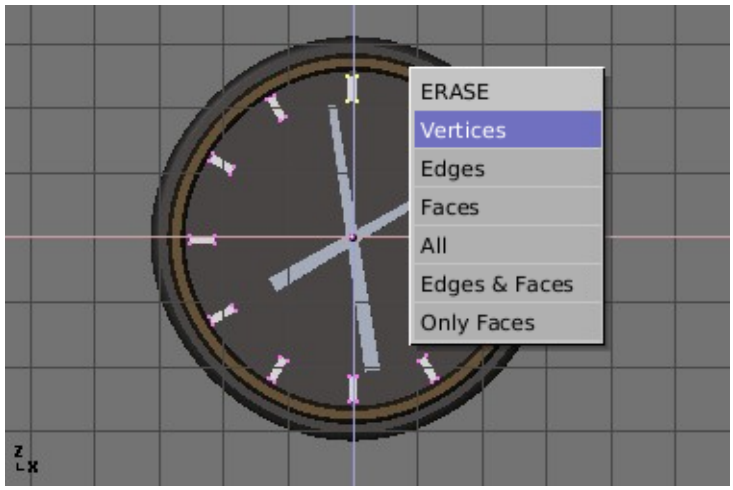


Figure 6-43. Removal of duplicated object

> **Avoiding duplicates:** If you like a little math you needn't bother with duplicates because you can avoid them at the start. Just make 11 duplicates, not 12, and not around the whole 360°, but just through 330° (that is 360*11/12). This way no duplicate is placed over the original object.
>
> In general, to make *n* duplicates over 360 degrees without overlapping, just spin one less object over 360*(*n*-1)/*n* degrees.

Figure 6-44 shows the final rendering of the clock.

Figure 6-44. Final Clock Render.

Screw

The "Screw" tool starts a repetitive "Spin" combined with a translation generating a screw-like, or spiral-shaped, object. You can use this to create screws, springs or shell-shaped structures.

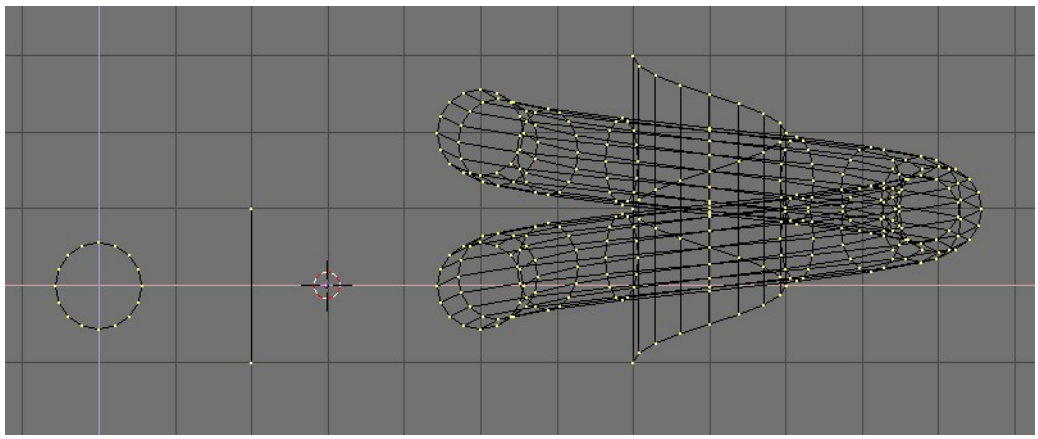


Figure 6-45. How to make a spring: before (left) and after (right) the Screw tool.

The method for using the "Screw" function is strict:

- Set the 3DWindow to *front view* (**NUM1**).
- Place the 3DCursor at the position through which the rotation axis must pass. Such an axis will be vertical.
- Ensure that an *open poly line* is available. This can be a single edge as in the figure or half a circle, or ensure that there are two 'free' ends; two edges with only one vertex linked to another edge. The "Screw" function localises these two points and uses them to calculate the translation vector that is added to the "Spin" per full rotation (Figure 6-45). If these two vertices are at the same location, this creates a normal "Spin". Otherwise, interesting things happen!
- Select all vertices to participate in the "Screw".
- Give the Num Buttons `Steps:` and `Turns:` the desired values. `Steps:` is related to how many times the profile is repeated within each 360° rotation, while `Turns:` is the number of complete 360° rotations to be performed.
- Press `Screw`!

If there are multiple 3DWindows, the mouse cursor changes to a question mark. Click on the 3DWindow in which the "Screw" is to be executed.

If the two "free" ends are aligned vertically the result is the one seen above. If they are not, then the translation vector stays vertical, equal to the vertical component of the vector joining the two 'free' vertices, while the horizontal component generates an enlargement (or reduction) of the screw as shown in Figure 6-46.

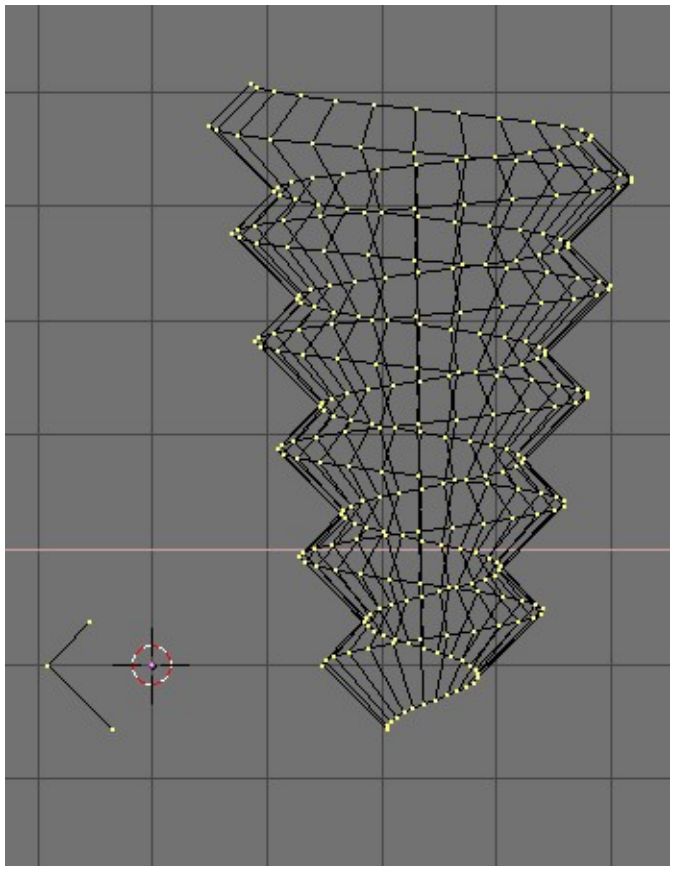


Figure 6-46. Enlarging screw (right) obtained with the profile on the left.

Noise

The noise function allows you to displace vertices in meshes based on the grey-values of a texture. That way you can generate great landscapes or carve text into meshes.

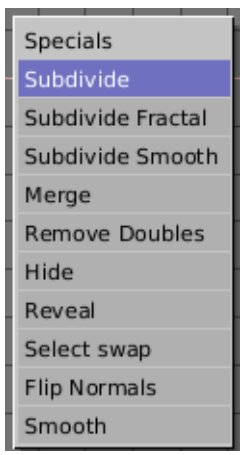


Figure 6-47. Subdivide tool

Add a plane and subdivide it at least five times with the special menu **WKEY**->Subdivide (Figure 6-47). Now add a material and assign a Clouds-texture

to it. Adjust the `NoiseSize:` to 0.500. Choose white as the color for the material and black as the texture color, that will give us a good contrast for the noise operation.

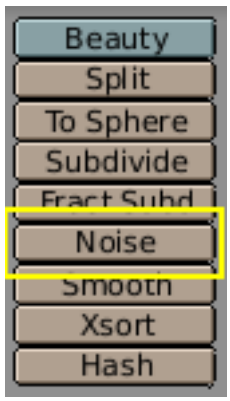


Figure 6-48. Noise button in EditButtons

Ensure that you are in EditMode and all vertices are selected and switch to the EditButtons **F9**. Press the `Noise` button (Figure 6-48) several times until the landscape looks nice. Figure 6-49 shows the original - textured - plane as well as what happens as you press `Noise`. You should remove the texture from the landscape now because it will disturb the look. You can now add some lights, maybe some water, set smooth and SubSurf the terrain, etc. (Figure 6-50).

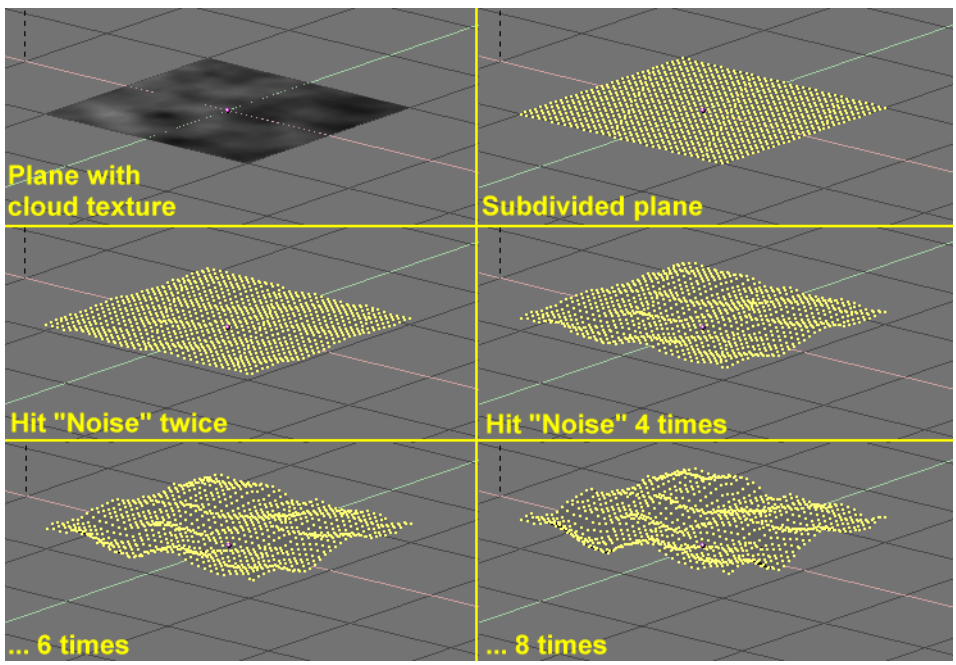


Figure 6-49. Noise application process

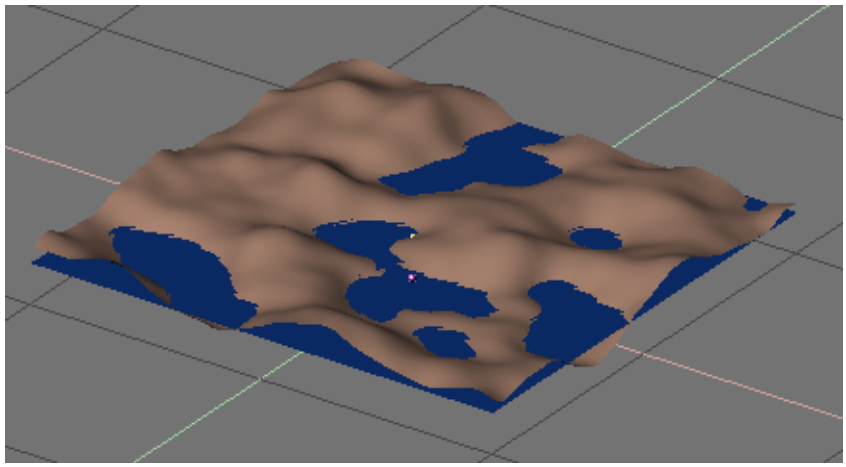


Figure 6-50. Noise generated landscape

Beware that the noise displacement always occurs along the mesh's *z* co-ordinate, that is along the direction of the *z* axis of the Object local reference.

Warp Tool

The warp tool is a little-known tool in Blender, partly because it is not found in the EditButtons window, and partly because it is only useful in very specific cases. It is not something the average Blender-user needs every day.

A piece of text wrapped into a ring shape is useful in flying logos, but it would be difficult to model without the warp tool. We will warp the phrase "Amazingly Warped Text" around a sphere. First add the sphere.

Then add the text in front view, set "Ext1" to 0.1 - making the text 3D, and set "Ext2" to 0.01, adding a nice bevel to the edge. Make the "BevResol" 1 or 2 to have a smooth bevel and lower the resolution so that the vertex count will not not be too high when you subdivide the object later on (Figure 6-51). Convert the object to curves, then to a mesh, (**ALT+C** twice) because the warp tool does not work on text or on curves. Subdivide the mesh twice, so that the geometry will change shape cleanly, without artifacts.

Figure 6-51. Text settings

Switch to top view and move the mesh away from the 3D cursor. This distance defines the radius of the warp. See Figure 6-52.

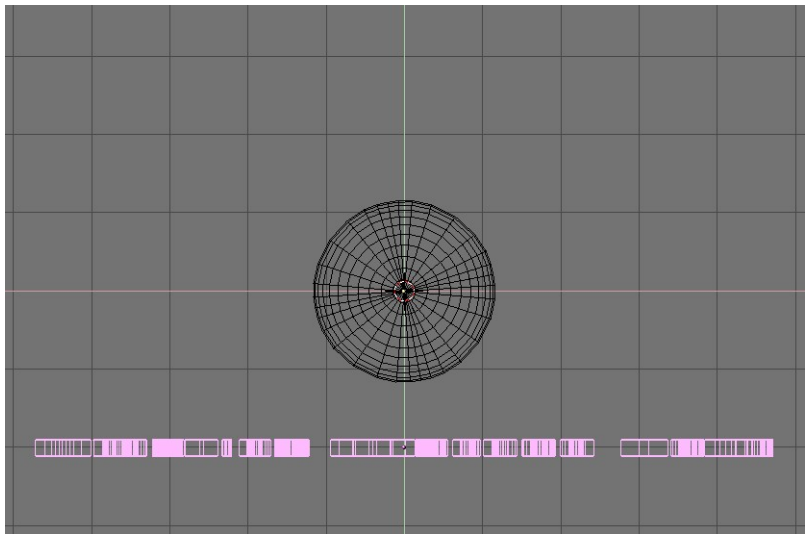


Figure 6-52. Top view of text and sphere

Place the mesh in edit mode and press **AKEY** to select all vertices. Press **SHIFT+W** to activate the warp tool. Move the mouse up or down to interactively define the amount of warp. (Figure 6-53). Holding down **CTRL** makes warp change in steps of five degrees.

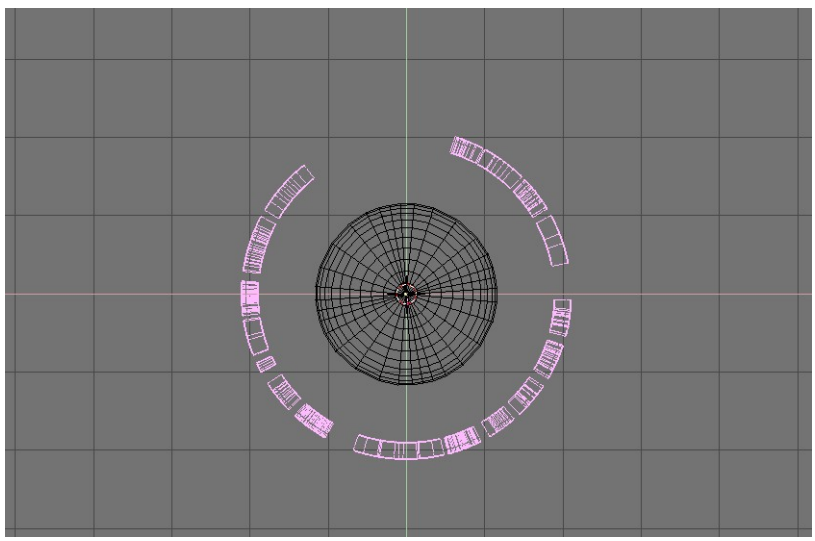


Figure 6-53. Warped text

Now you can switch to camera view, add materials, lights and render (Figure 6-54).

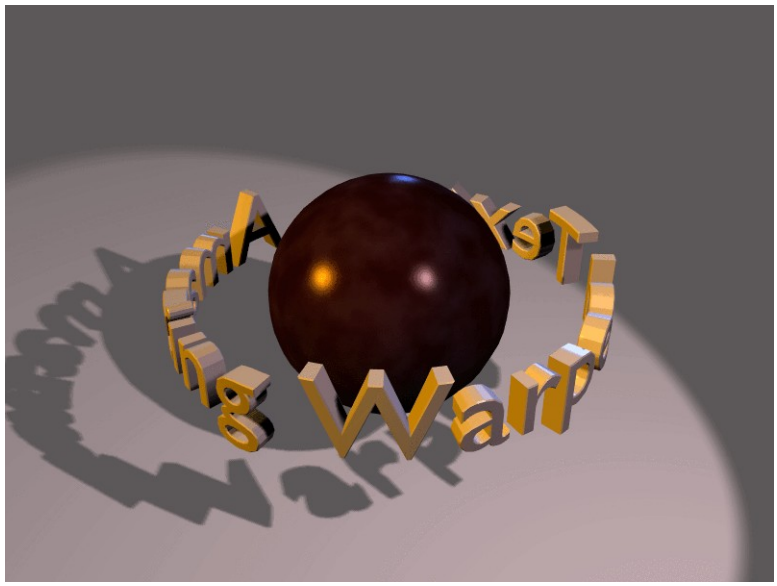


Figure 6-54. Final rendering

Catmull-Clark Subdivision Surfaces

Catmull-Clark Subdivision Surfaces or, in short *SubSurf,* is a mathematical algorithm to compute a "smooth" subdivision of a mesh. With any regular Mesh as a starting point, Blender can calculate a smooth subdivision on the fly; while modelling or while rendering. This allows high resolution Mesh modelling without the need to save and maintain huge amounts of data. This also allows for a smooth 'organic' look to the models.

Actually a SubSurfed Mesh and a NURBS surface have many points in common inasmuch as both rely on a "coarse" low-poly "mesh" to define a smooth "high definition" surface, but there are also notable differences:

- NURBS have finer control on the surface, since you can set "weights" independently on each control point of the control mesh. On a SubSurfed mesh you cannot act on weights.
- SubSurfs have a more flexible modelling approach. Since a SubSurf is a mathematical operation occurring on a mesh, you can use all the modelling techniques described in this Chapter on the mesh, which are more numerous, and far more flexible, than those available for NURBS control polygons.

SubSurf is a Mesh option, the button to activate this is in the EditButtons **F9** (Figure 6-55). The NumButtons immediately below it define, on the left, the resolution (or level) of subdivision for 3D visualization purposes, the one on the right the resolution for rendering purposes.

Since SubSurf computations are performed both real-time, while you model, and at render time, and they are CPU intensive, it is usually good practice to keep the SubSurf level low (but non-zero) while modelling and higher for rendering.

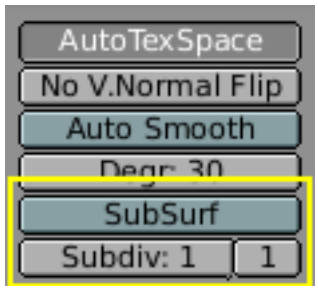


Figure 6-55. SubSurf buttons

Figure 6-56 and Figure 6-57 shows, respectively, a rendering of a SubSurf mesh and the original, un-SubSurfed mesh.

Figure 6-56. Render of a SubSurfed vase.

Figure 6-57. The un-SubSurfed mesh of that same vase.

Figure 6-58 shows a 0,1,2,3 level of SubSurf on a single square face or on a single triangular face. Such a subdivision is performed, on a generic mesh, for *each* square or rectangular face.

It is evident how each single quadrilateral face produces *4^n* faces in the SubSurfed mesh, *n* being the SubSurf level, or resolution, while each triangular face produces *3*4^(n-1)* new faces (Figure 6-58). This dramatic increase of face (and vertex) num-

ber reflects in a slow-down of all editing, and rendering, actions and calls for lower SubSurf level in the Editing process than in the rendering one.

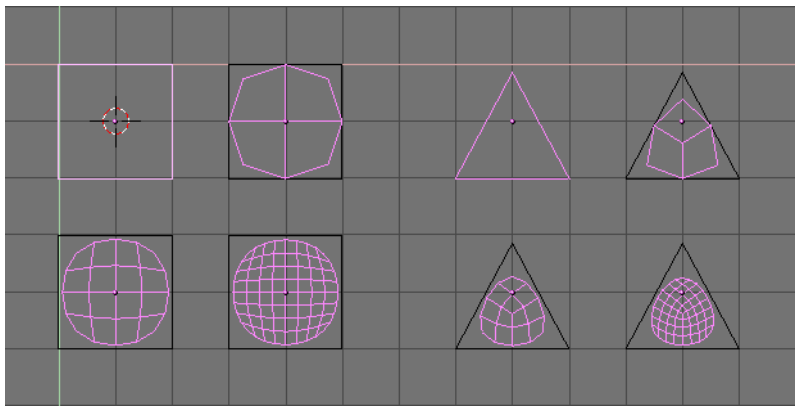


Figure 6-58. SubSurf of simple square and Rectangular faces.

Blender's subdivision system is based on the Catmull-Clarke algorithm. This produces nice smooth SubSurf meshes but any 'SubSurfed' face, that is, any small face created by the algorithm from a single face of the original mesh, shares the normal orientation of that original face.

This is not an issue for the shape itself, as Figure 6-59 shows, but it is an issue in the rendering phase, and in solid mode, where abrupt normal changes can produce ugly black lines (Figure 6-60).

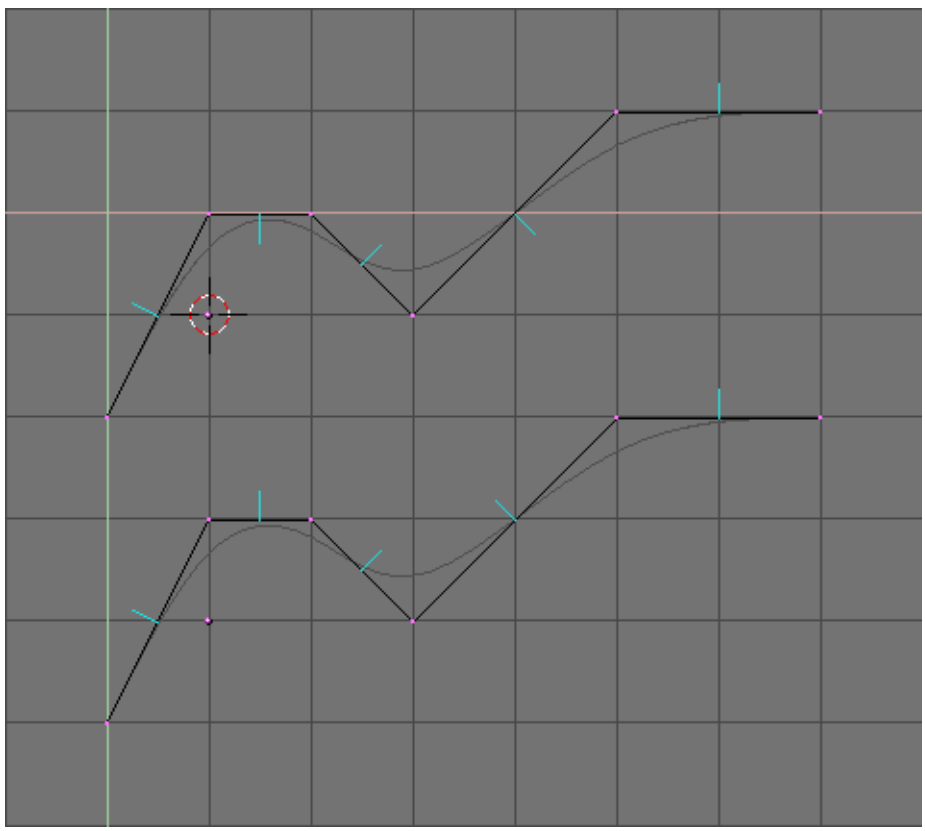


Figure 6-59. Side view of subsurfed meshes. With random normals (top) and with

coherent normals (bottom)

Use the **CTRL+N** command in EditMode and with all vertices selected to make Blender recalculate the normals.

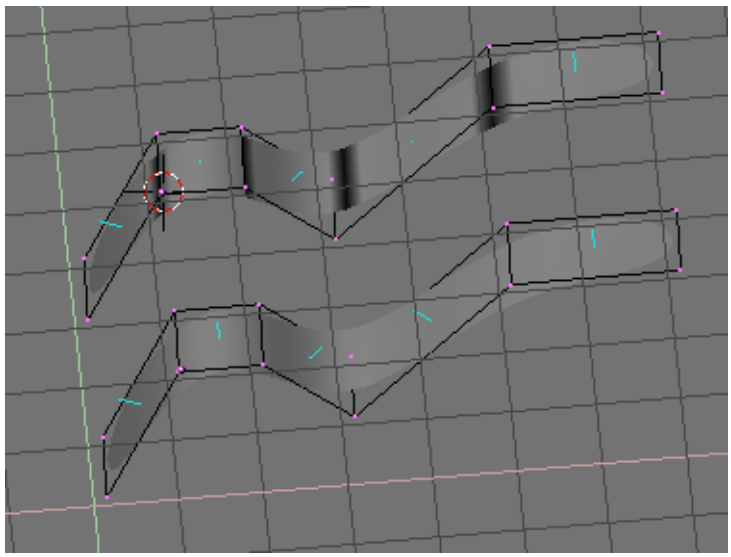


Figure 6-60. Solid view of SubSurfed meshes with inconsistent normals (top) and consistent normals (bottom).

In the images the face normals are drawn cyan. You can enable drawing normals in the EditButtons (**F9**) menu.

It is worth noting that Blender cannot recalculate normals correcty if the mesh is not "Manifold". A "Non-Manifold" mesh is a mesh for which an 'out' cannot unequivocally be computed. Basically, from the Blender point of view, it is a mesh where there are edges belonging to *more* than two faces.

Figure 6-61 shows a very simple example of a "Non-Manifold" mesh. In general a "Non-Manifold" mesh occurs when you have internal faces and the like.

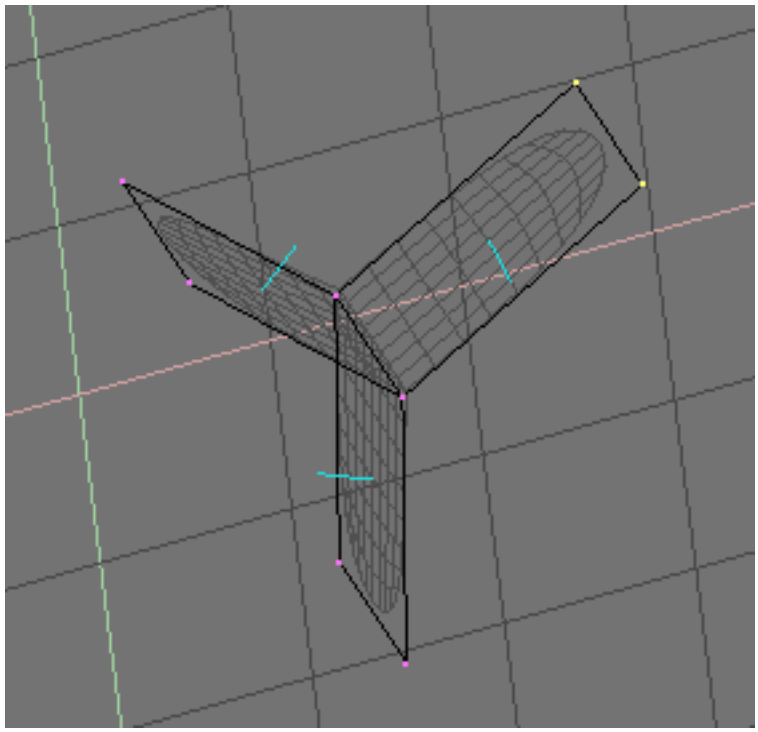


Figure 6-61. A "Non-Manifold" mesh

A "Non-Manifold" mesh is not a problem for conventional meshes, but can give rise to ugly artifacts in SubSurfed meshes, and does not allow decimation, so it is better to avoid them as much as possible.

You can tell that a mesh is "Non Manifold" from two hints:

- The Recalculation of normals leaves black lines somewhere
- The "Decimator" tool in the EditButtons refuses to work stating that the mesh is "No Manifold"

The SubSurf tool allows for very good "organic" models, but keep in mind that a regular Mesh with square faces gives the best result.

Figure 6-62 and Figure 6-63 shows an example of what can be done with Blender SubSurfs.

Figure 6-62. A Gargoyle base mesh (left) and pertinent level 2 SubSurfed Mesh (right).

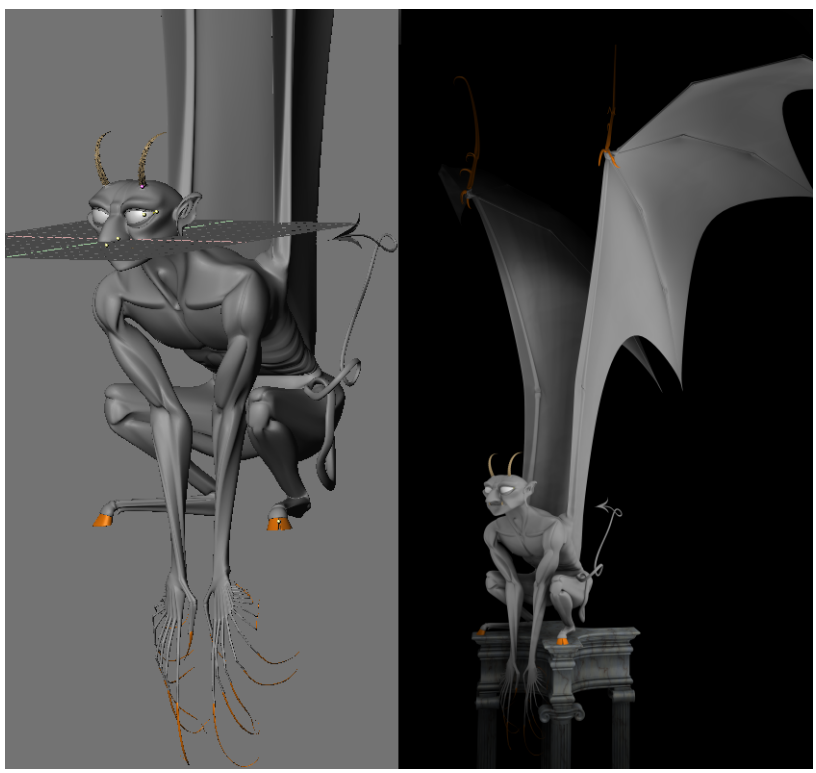


Figure 6-63. Solid view (left) and final rendering (right) of the Gargoyle.

MetaBall

MetaBalls consist of spherical or tubular elements that can operate on each other's shape (Figure 6-64). You can only create round and fluid, 'mercurial' or 'clay-like' forms that exist *procedurally*. Use MetaBalls for special effects or as a basis for modelling.

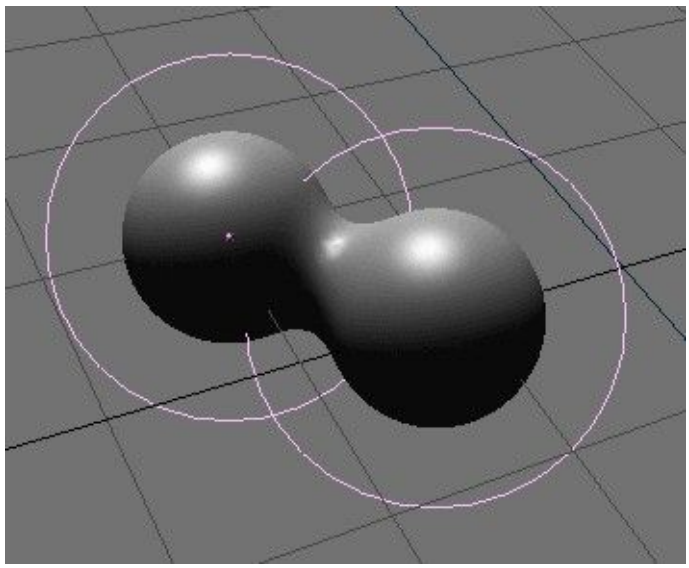


Figure 6-64. Two Metaballs

In fact, MetaBalls are nothing more than mathematical formulas that perform logical operations on one another (AND, OR), and that can be added and subtracted. This method is also called CSG, Constructive Solid Geometry. Because of its mathematical nature, CSG can be displayed well, and relatively quickly, with Ray Tracing. This is much too slow for interactive displays. Thus *polygonize* routines were developed. The complete CSG area is divided into a 3D grid, and for each *edge* in the grid a calculation is made, and if (and more importantly where) the formula has a turning point, a 'vertex' for the *polygonize* is created there.

The available quantity of CSG *primitives* and tools in Blender is limited. This will be developed further in future versions of Blender. The basis is already there; and it is outstandingly implemented. Unfortunately Blender has little need for modelling systems that are optimized for Ray Tracing. However, it is still fun to play with...

A MetaBall is displayed with the *transformations* of an Object and an exterior determined by the Material. Only one Material can be used here. In addition, MetaBall saves a separate texture area; this normalises the coordinates of the vertices. Normally the texture area is identical to the *boundbox* of all vertices. The user can force a texture area with the **TKEY** command (outside EditMode).

MetaBalls are extremely compact in memory and in the file. The requisite faces are only generated upon rendering. This can take a great deal of calculation time and memory.

To create a Metaball object press **SPACE** and select 'ADD->Metaball'. You can also access the 'add'-menu by pressing **SHIFT-AKEY**.

In EditMode you can move and scale the balls or rounded tubes as you wish. This is the best way to construct static - not animated - forms. MetaBalls can also influence each other *outside* EditMode. Now you have much more freedom in which to work; the balls can now rotate or move; they get every *transformation* of the Parents' Objects. This method requires more calculation time and is thus somewhat slower.

The following rules describe the relation between MetaBall Objects:

- All MetaBall Objects with the same 'family' name (the name without the number) influence each other. For example "Ball", "Ball.001", "Ball.002", "Ball.135". Note here that we are *not* talking about the name of the MetaBall ObData block.
- The Object with the family name *without* a number determines the basis, the resolution *and* the transformation of the polygonize. It also has the Material and texture area.

To be able to display animated MetaBalls 'stably', it is important to determine what Object forms the basis. If the basis moves and the rest remains still, you will see the polygonized faces move 'through' the balls.

The "Threshold" in EditButtons is an important setting for MetaBalls. You can make the entire system more fluid - less detailed - or harder using this option. The resolution of poygonize is also specified in EditButtons. This is the big memory consumer; however, it is released immediately *after* polygonize. It works efficiently and faster if you work with multiple, more compact 'families' of balls. Because it is slow, the polygonize is not immediately recalculated for each change. It is always recalculated after a Grab, Rotate or Size command.

Resources

- *Introduction to mesh editing: Modelling a Die.* - http://www.vrotvrot.com/xoom/tutorials/Die/dice.html
- *Introduction to mesh editing: A High-Tech Corridor.* - http://www.vrotvrot.com/xoom/tutorials/Corridor/Corridor.html

Notes

1. http://www.vrotvrot.com/xoom/tutorials/Die/dice.html
2. http://www.vrotvrot.com/xoom/tutorials/Corridor/Corridor.html

Chapter 7. Curves and Surfaces

Curves and surfaces are objects like meshes, but differ in that they are expressed in mathematical functions, rather than a mere point-to-point basis.

Blender implements Bézier and Non Uniform Rational B-Splines (NURBS) curves and surfaces. Both these, though following different mathematical laws, are defined in terms of a set of "control vertices" defining a "control polygon". The way the curve and the surface are interpolated (Bézier) or attracted (NURBS) by these might seem similar, at a first glance, to Catmull-Clark subdivision surfaces.

Curves and surfaces have advantages and disadvantages compared to meshes. They are defined by less data, so they produce nice results with less memory usage at modelling time, whereas the demands increase at rendering time.

Some modelling techniques are only possible with curves, like extruding a profile along a path, but very fine control, as that available on a per-vertex basis on a mesh, is not possible.

Briefly, there are occasions in which Curves and surfaces are more advantageous than meshes, and occasions where meshes are more useful. Only experience, and the reading of these pages, can give an answer.

Curves

This section will describe both Bézier and NURBS curves, and show a working example of the former.

Béziers

Bézier curves are the most commonly used type for designing letters or logos. They are very important to understand since they are also widely used in animation, both as path for objects to move on and as IPO curves to change the properties of objects as a function of time.

A control point (vertex) of a Bézier curve consists of a point and two handles. The point, in the middle, is used to move the entire control point; selecting it will also select the other two handles, and allow you to move the complete vertex. Selecting one or two of the other handles will allow you to change the shape of the curve by dragging them.

Actually a Bézier curve is tangent to the line segment which goes from the point to the handle. The 'steepness' of the curve is controlled by the handle's length.

There are four types of handles (Figure 7-1):

- Free Handle (black). This can be used in any way you wish. Hotkey: **HKEY** (toggles between Free and Aligned);
- Aligned Handle (purple). These handles always lie in a straight line. Hotkey: **HKEY** (toggles between Free and Aligned);
- Vector Handle (green). Both parts of a handle always point to the previous handle or the next handle. Hotkey: **VKEY**;
- Auto Handle (yellow). This handle has a completely automatic length and direction. Hotkey: **SHIFT+H**.

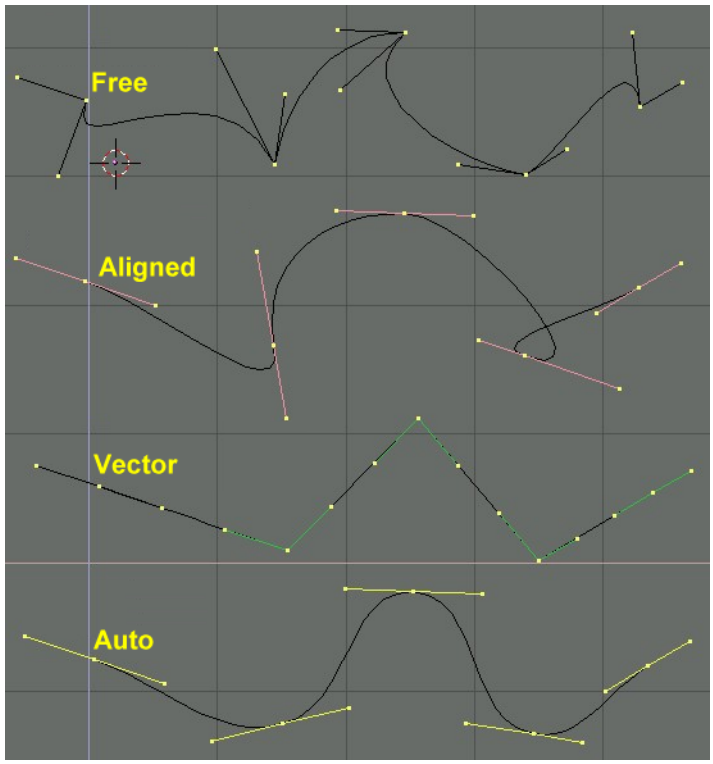


Figure 7-1. Types of Handles for Bézier curves

Handles can be *rotated* by selecting the end of one of the vertices. Again, use the grabber with **RMB**-hold-move.

As soon as the handles are rotated, the type is modified automatically:

- Auto Handle becomes Aligned;
- Vector Handle becomes Free.

Although the Bézier curve is a continuous mathematical object it must nevertheless be represented in discerete form from a rendering point of view.

This is done by setting a *resolution* property, which defines the number of points which are computed between every pair of control points.

A separate *resolution* can be set for each Bézier curve (the number of points generated between two points in the curve - Figure 7-2).

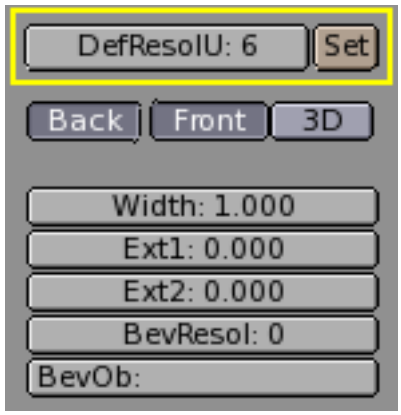


Figure 7-2. Setting Bézier resolution.

NURBS

NURBS curves are defined as rational polynomials, and are more general, strictly speaking, than conventional B-Splines and Bézier curves. They have a large set of variables, which allow you to create mathematically pure forms (Figure 7-3). However, working with them requires a little more intuition:

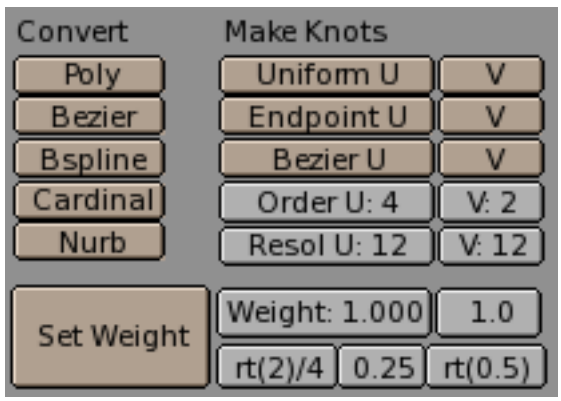


Figure 7-3. Nurbs Control Buttons.

- *Knots.* Nurbs curves have a *knot* vector, a row of numbers that specifies the parametric definition of the curve. Two pre-sets are important for this. "Uniform" produces a uniform division for closed curves, but for open ones you will get "free" ends, difficult to locate precisely. "Endpoint" sets the knots in such a way that the first and last vertices are always part of the curve, hence much easier to place;
- *Order.* The *order* is the 'depth' of the curve calculation. Order '1' is a point, order '2' is linear, order '3' is quadratic, etc. Always use order '5' for Curve paths; this behaves fluidly under all circumstances, without irritating discontinuities in the movement. Mathematically speaking this is the order of both the Numerator and the Denominator of the rational polynomial defining the NURBS;
- *Weight.* Nurbs curves have a 'weight' per vertex - the extent to which a vertex participates in the "pulling" of the curve.

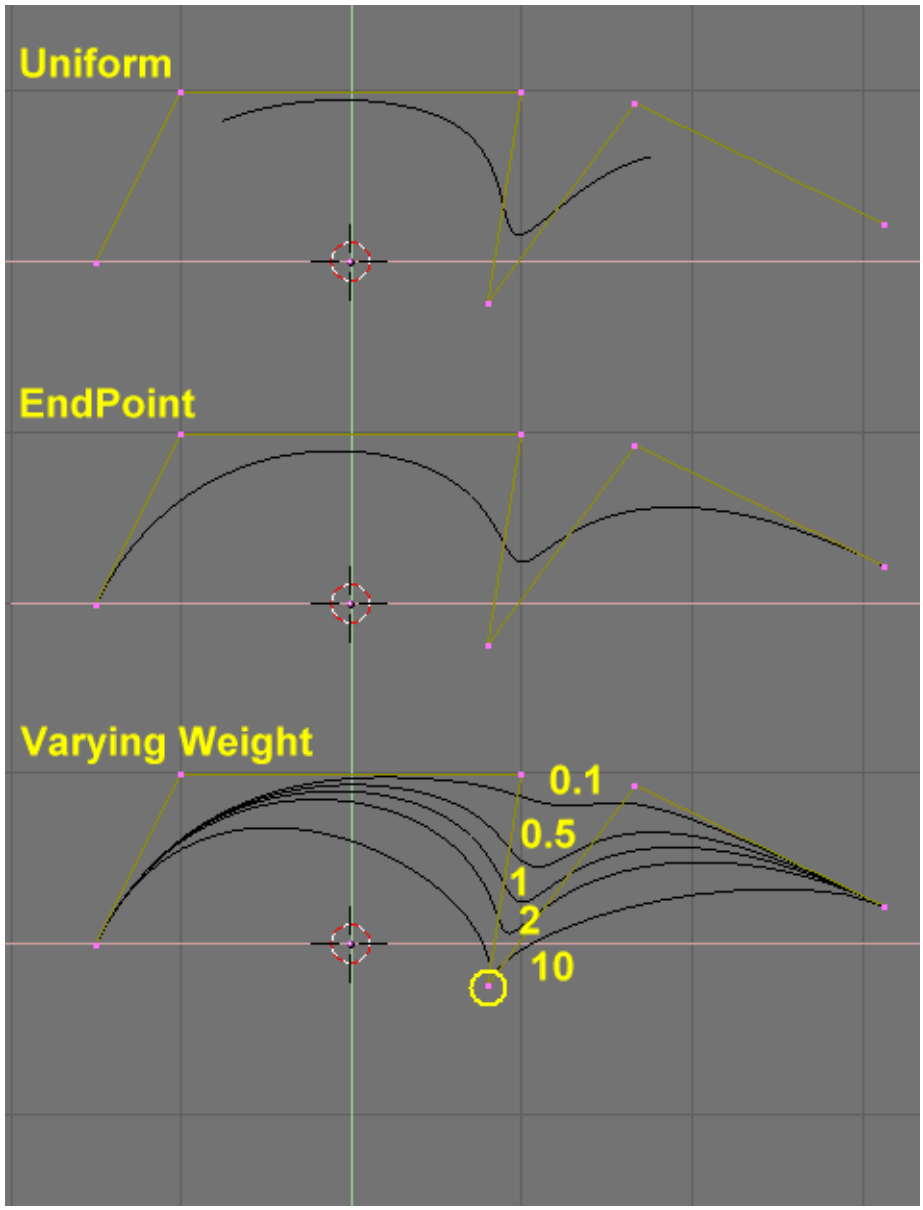


Figure 7-4. Setting Nurbs Control polygon and weights.

Figure 7-4 shows the Knot vector settings as well as the effect of varying a single knot weight. Just as with Béziers, the resolution can be set on a per curve basis.

Working example

Blender's curve tools provide a quick and easy way to build great looking extruded text and logos. We will use these tools to turn a rough sketch of a logo into a finished 3D object.

Figure 7-5 shows the design of the logo we will be building.

Figure 7-5. The sketched logo

First, we will import our original sketch so that we can use it as a template. Blender supports TGA, PNG and JPG format images. To load the image, move the cursor over a 3D window and press **SHIFT+F7** to get to the view settings for that window. Activate the BackGroundPic button and use the `LOAD` button to locate the image you want to use as a template (Figure 7-6).

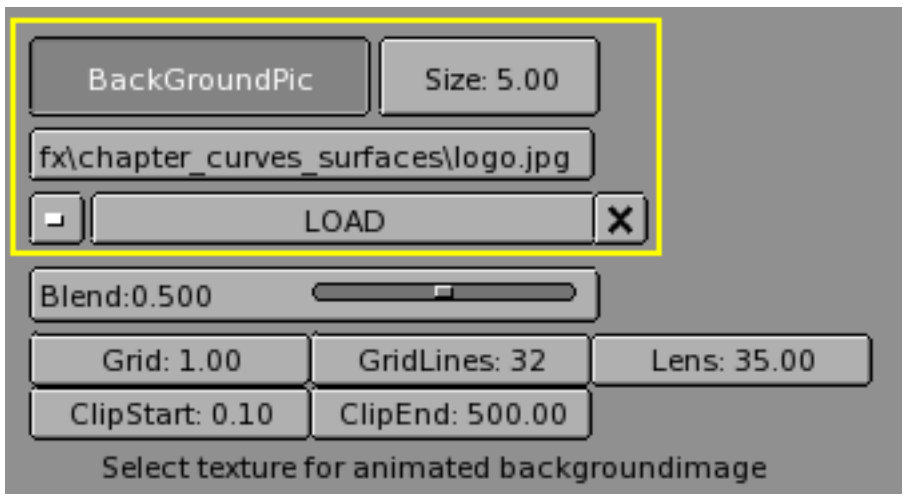


Figure 7-6. 3D window settings.

Return to the 3D view by pressing **SHIFT+F5** (Figure 7-7). You can hide the background image when you are finished using it by returning to the **SHIFT+F7** window and deselecting the `BackGroundPic` button.

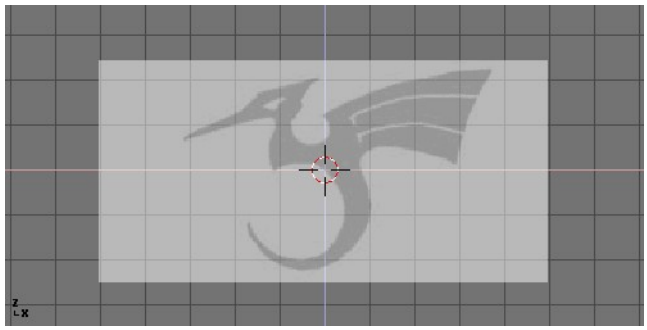


Figure 7-7. Logo sketch loaded as background

Add a new curve by pressing **SHIFT+A**>>CURVE>>BEZIER CURVE. A curved segment will appear and Blender will enter EditMode. We will move and add points to make a closed shape that describes the logo you are trying to trace.

You can add points to the curve by selecting one of the two endpoints, then holding **CTRL** and clicking **LMB**. Note that the new point will be connected to the previously selected point. Once a point has been added, it can be moved by selecting the control vertex and pressing **GKEY**. You can change the angle of the curve by grabbing and moving the handles associated with each vertex (Figure 7-8).

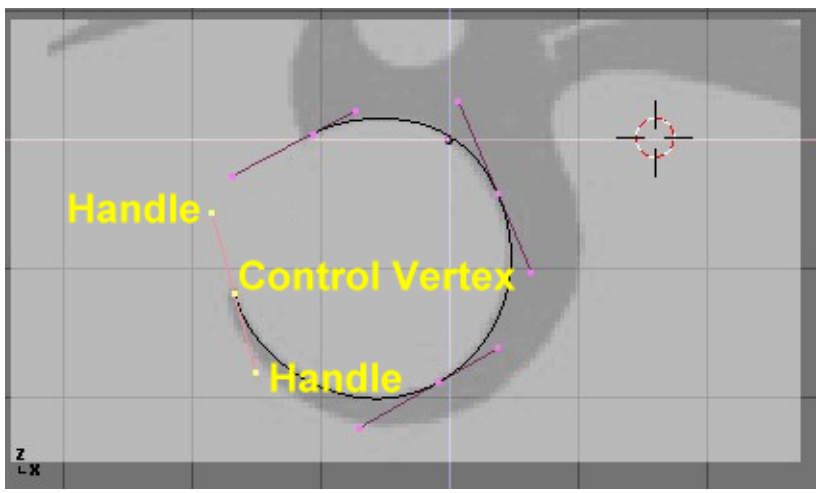


Figure 7-8. Bézier handles

You can add a new point between two existing points by selecting the two points and pressing **WKEY**>>SUBDIVIDE (Figure 7-9).

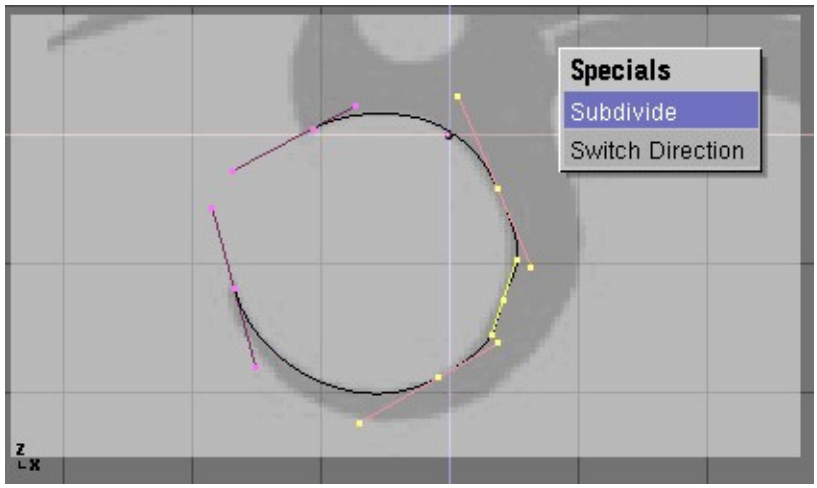


Figure 7-9. Adding a Contrl Point.

Points can be removed by selecting them and pressing **XKEY**>>SELECTED. You cut a curve into two curves by selecting two adjacent control vertices and pressing **XKEY**>>SEGMENT.

To make sharp corners, you can select a control vertex and press **VKEY**. You will notice the colour of the handles change from purple to green (Figure 7-10). At this point, you can adjust the handles to adjust the way the curve enters and leaves the control vertex (Figure 7-11).

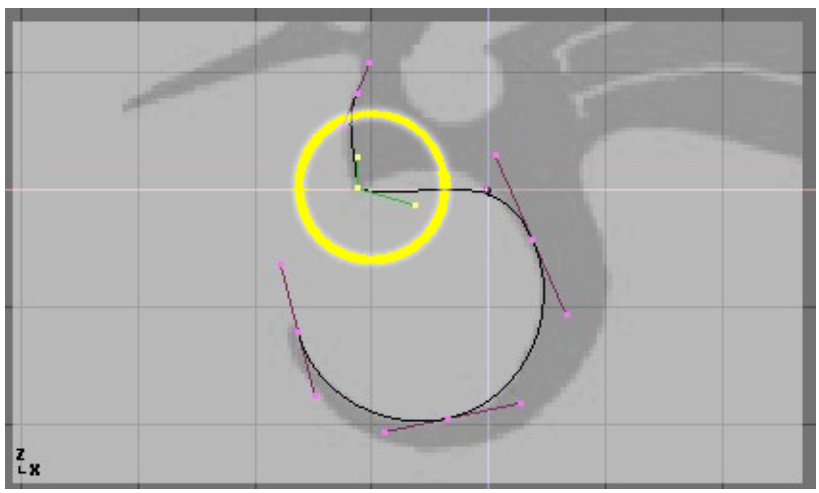


Figure 7-10. Vector (green) handles.

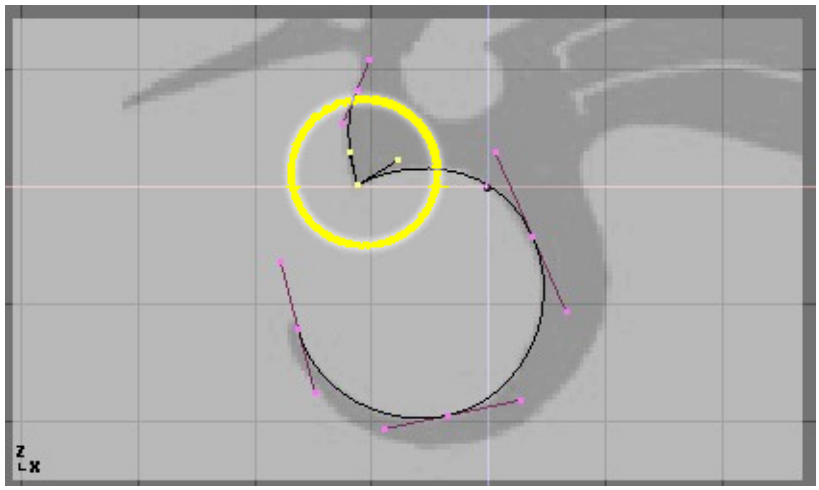


Figure 7-11. Free (black) handles.

To close the curve and make it into a single continuous loop, select at least one of the control vertices on the curve and press **CKEY**. This will connect the last point in the curve with the first one (Figure 7-12). You may need to manipulate some more handles to get the shape you want.

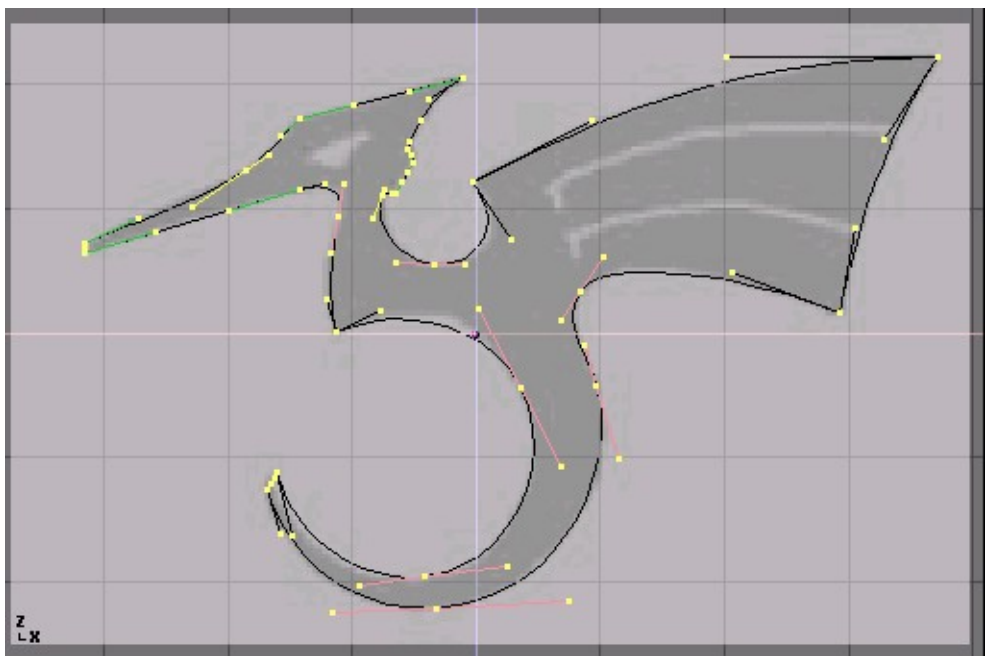


Figure 7-12. Figure Figure 8

Leaving editmode with **TAB** and entering shaded mode with **ZKEY** should reveal that the curve renders as a solid shape (Figure 7-13). We want to cut some holes into this shape to represent the eyes and wing details of the dragon. When working with curves, Blender automatically detects holes in the surface and handles them accordingly. Actually a closed curve is considered as the boundary of a surface. If a closed curve is completely included within another one, the former is subtracted by the latter, effectively defining a hole. Return to wireframe mode with **ZKEY** and enter editmode again with **TAB**.

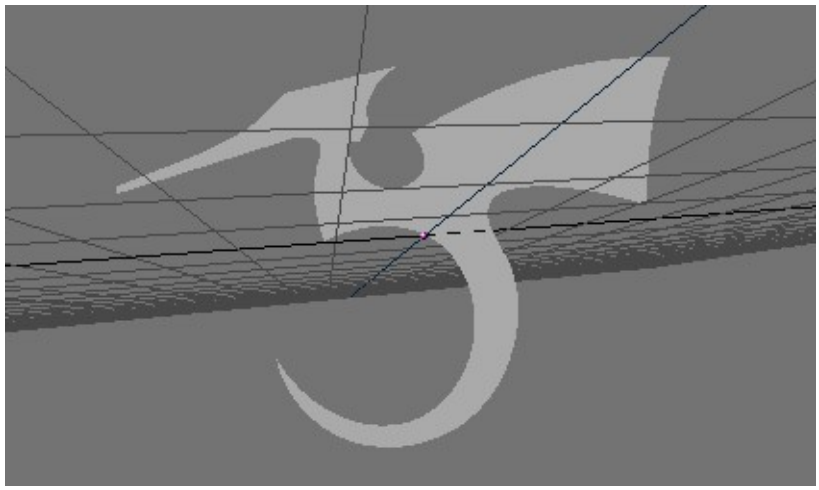


Figure 7-13. Shaded logo.

While still in editmode, add a circle curve with **SHIFT+A**>>CURVE>>BEZIER CIRCLE (Figure 7-14). Scale the circle down to an appropriate size with **SKEY** and move it with **GKEY**.

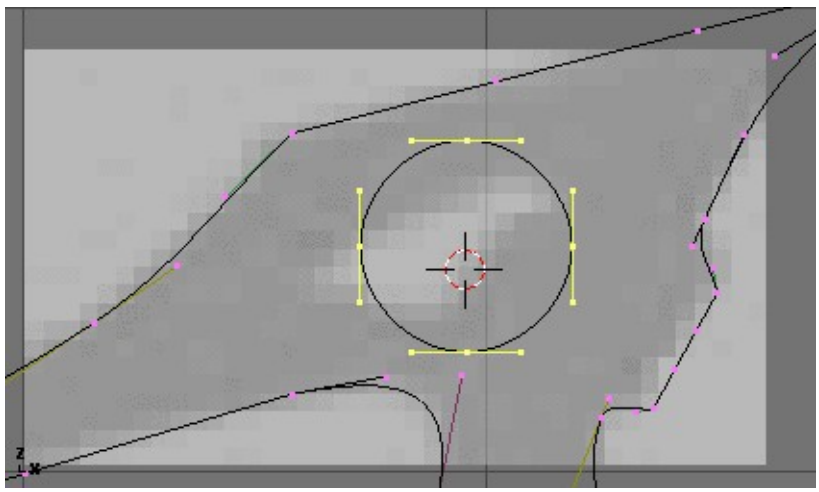


Figure 7-14. Adding a circle.

Shape the circle using the techniques we have learned (Figure 7-15). Remember that you can add vertices to the circle with **WKEY**>>SUBDIVIDE.

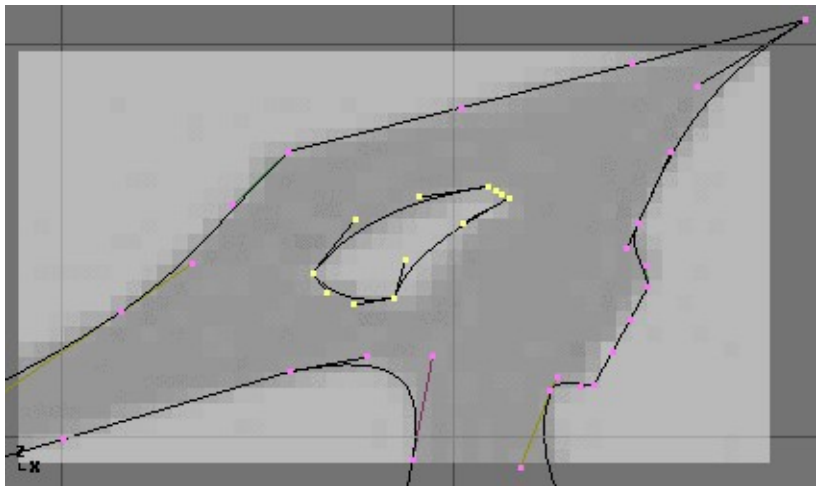


Figure 7-15. Defining the eye.

Create a wing cutout by adding a Bézier circle, converting all of the points to sharp corners, and then adjusting as necessary. You can duplicate this outline to save time when creating the second wing cutout. To do this, make sure no points are selected, then move the cursor over one of the vertices in the first wing cutout and select all linked points with **LKEY** (Figure 7-16). Duplicate the selection with **SHIFT+D** and move the new points into position.

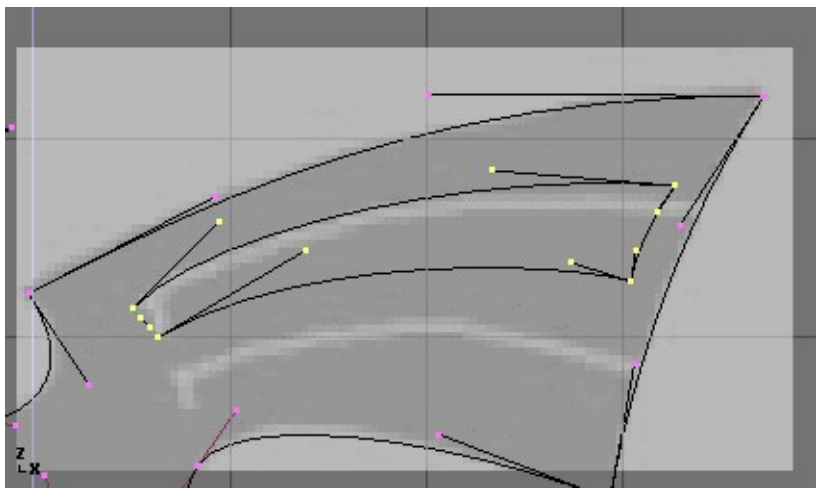


Figure 7-16. Defining the wings.

If you want to add more geometry that is not connected to the main body (placing an orb in the dragon's curved tail for example), you can do this by using the **SHIFT+A** menu to add more curves as shown in Figure 7-17

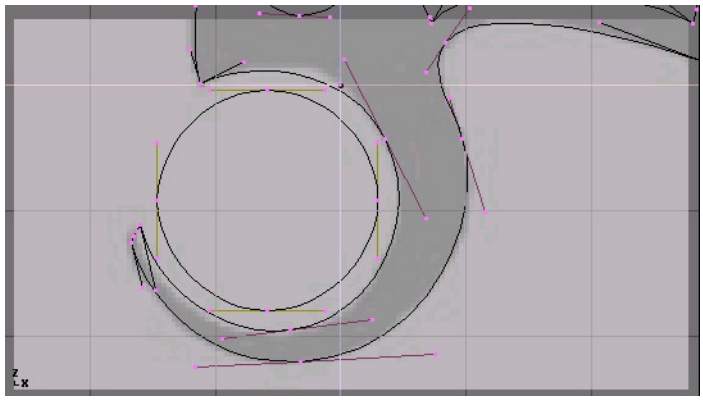


Figure 7-17. Orb placement within the tail.

Now that we have the curve, we need to set its thickness and beveling options. With the curve selected, go to the EditButtons (**F9**). The "Ext1" parameter sets the thickness of the extrusion while "Ext2" sets the size of the bevel. BevResol sets how sharp or curved the bevel will be.

Figure 7-18 shows the settings used to extrude this curve.

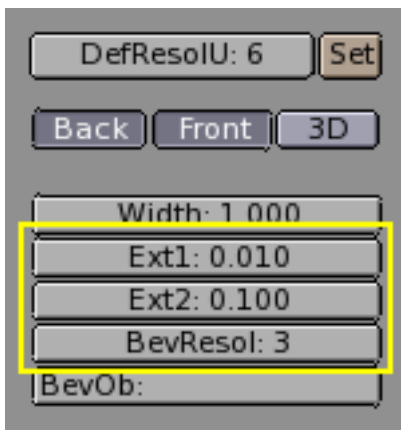


Figure 7-18. Bevel settings

> **From Curves to Meshes:** If want to perform more complex modeling operations, you can convert the curve to a mesh with **ALT+C**>>MESH. Note that this is a one-way operation: you cannot convert a mesh back into a curve.

When your logo model is complete, you can add materials and lights and make a nice rendering (Figure 7-19).

Figure 7-19. Final rendering.

Surfaces

Surfaces are actually an extension of NURBS curves. In Blender they are a separate ObData type. Whereas a curve only produces one-dimensional interpolation, Surfaces have a second extra dimension, the first called U, as for curves, the second V.

A two-dimensional grid of control points defines the form for these NURBS surfaces.

Use Surfaces to create and revise fluid curved surfaces. They can be cyclical in both directions, allowing you to easily create a 'donut' shape. Surfaces can also be drawn as 'solids' in EditMode (zbuffered, with OpenGL lighting). This makes working with surfaces quite easy.

Currently Blender has a basic tool set for Surfaces. It has limited functionality regarding the creation of holes and the melting of surfaces. Future versions will contain increased functionality in these areas.

You can take a surface 'primitive' from the ADD menu as a starting point Figure 7-20. NURBS curves are intrinsically NURBS Surfaces, simply having one dimension neglected. That's why you can choose 'Curve' from the 'Surface' Menu!

Beware anyway that a NURBS 'true' curve and a NURBS 'surface' curve are not interchangeable, as it will be clear for the extruding process described below and for the skinning section further on.

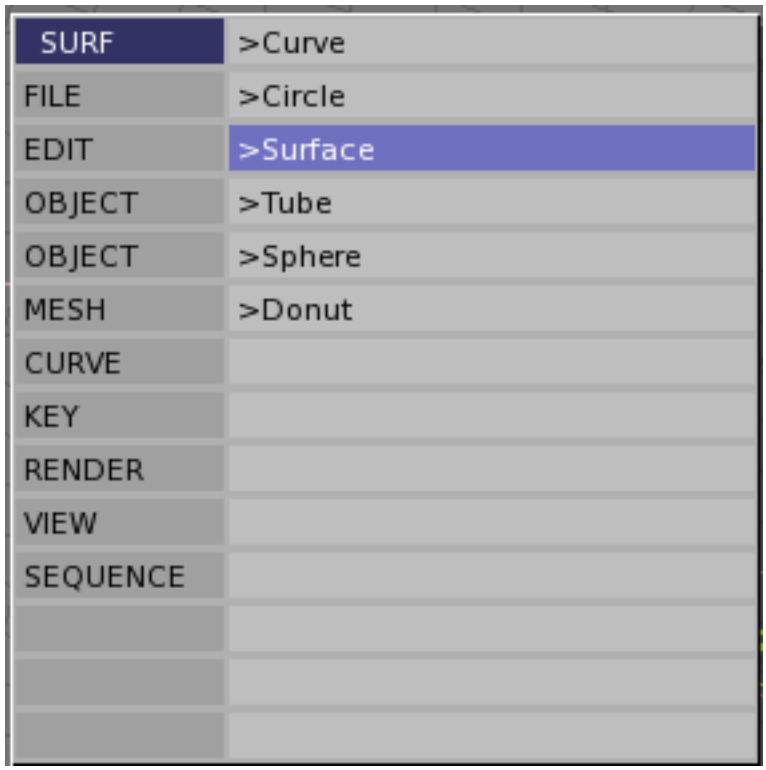


Figure 7-20. Add surface menu.

If you add a 'surface' curve you are then able to create a true surface simply by extruding the entire curve (**EKEY**). Each edge of a surface can then be extruded any way you wish to form the model. Use **CKEY** to make the U or V direction cyclic. It is important to set the 'knots' to "Uniform" or "Endpoint" with one of the pre-sets from the EditButtons.

A surface becomes *active* if one of its vertices is selected with the **RMB**. This causes the EditButtons to be re-drawn.

When working with surfaces, it is handy to always work on a complete column or row of vertices. Blender provides a selection tool for this: **SHIFT+R**, "Select Row". Starting from the last selected vertex, a complete row of vertices is *extend* selected in the 'U' or 'V' direction. Choose Select Row again with the same vertex and you toggle between the 'U' of 'V' selection.

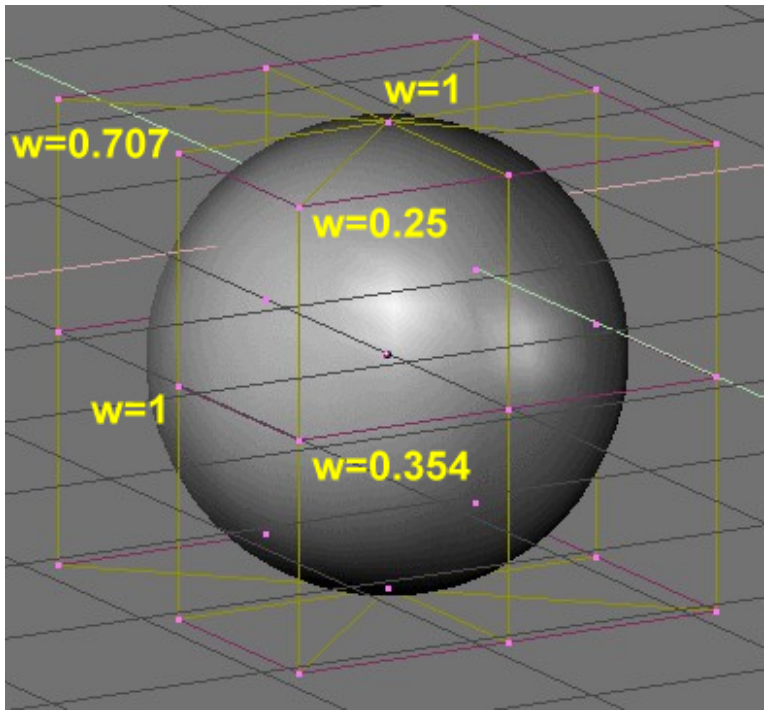


Figure 7-21. A sphere surface

NURBS are able to create pure shapes such as circles, cylinders and spheres. Beware that a Bézier circle is not a pure circle.

To create pure circles, globes or cylinders, you must set the weights of the vertices. This is not intuitive, and you are strongly advised to read more on NURBS first. Basically: to get a circular arc from a curve with 3 control points, the end points must have a unitary weight, and the central a weight equal to half the cosine of half the angle between the segments joining the points. Figure 7-21 shows this for a globe. Three standard numbers are included as pre-sets in the EditButtons (Figure 7-22). To read the weight of a vertex, once it is selected, press the **NKEY**.

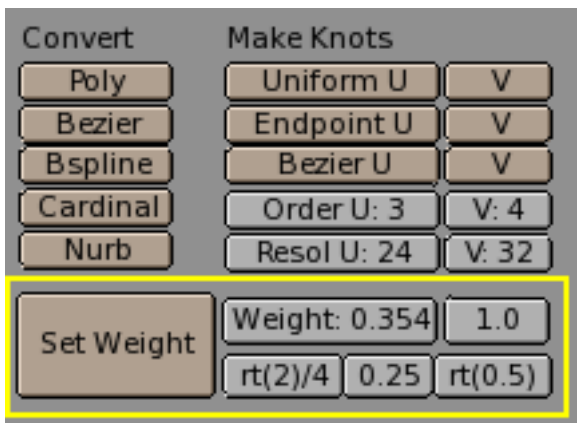


Figure 7-22. Pre-set weights

Text

Figure 7-23. Figure Text Examples

Text is a special curve type for Blender. Blender has its own built-in font but can use external fonts too, both PostScript Type 1 fonts and True Type fonts are supported by Blender (Figure 7-23).

Start with a fresh scene by pressing **CTRL-X** and add a TextObject with the Toolbox (ADD->Text). In EditMode you can edit the text with the keyboard, a text cursor shows your actual position in the text. When you leave the EditMode with **TAB**, Blender fills the text-curve, so that you have a flat filled object that is renderable at once.

Now go to the EditButtons **F9** (Figure 7-24).

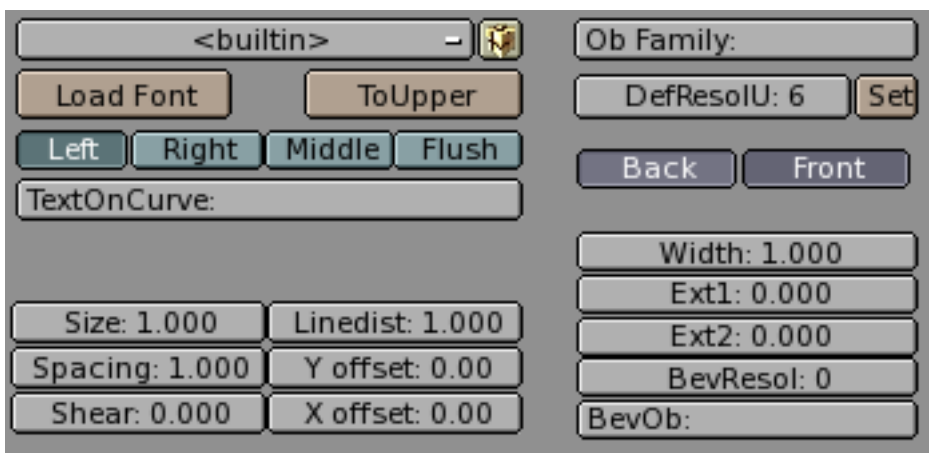


Figure 7-24. Text edit buttons

As you can see in the MenuButton, Blender uses by default its own `<builtin>` font when creating a new TextObject. Now click `Load Font` and, browsing what appears in the FileWindow, go to a directory containing PostScript Type 1 or True Type fonts

and load a new font (there are several free PostScript fonts that can be downloaded from the web, and Windows has many True Type fonts of its ownbut, in this latter case, be aware that some of them arecopyrighted!). Try out some other fonts. After loading a font, you can use the MenuButton to switch the font for a TextObject.

For now we have only a flat object. To add some depth, we can use the `Ext1:` and `Ext2:` buttons in just the same way as we have done with curves.

With the `TextOnCurve:` option you can make the text follow a 2D-curve. Use the alignment buttons above the `TextOnCurve:` textfield to align the text on the curve.

A powerful function is that a TextObject can be converted with **ALT-C** to a Curve, of the Bézier flavour, which allows you to edit the shape of every single character. This is especially handy for creating logos or for custom lettering.

The transformation from text to curve is irreversible and, of course, a further transformation from curve to mesh is possible too.

Special Characters

Normally, a Font Object begins with the word "Text". This can be deleted simply with **SHIFT+BACKSPACE**. In EditMode, this Object only reacts to text input. Nearly all of the hotkeys are disabled. The cursor can be moved with the arrowkeys. Use **SHIFT+ARROWLEFT** and **SHIFT+ARROWRIGHT** to move the cursor to the end of the lines or to the beginning or end of the text. Nearly all 'special' characters are available. A summary of these characters follows:

- **ALT+c**: copyright
- **ALT+f**: Dutch Florin
- **ALT+g**: degrees
- **ALT+l**: British Pound
- **ALT+r**: Registered trademark
- **ALT+s**: German S
- **ALT+x**: Multiply symbol
- **ALT+y**: Japanese Yen
- **ALT+DOTKEY**: a circle
- **ALT+1**: a small 1
- **ALT+2**: a small 2
- **ALT+3**: a small 3
- **ALT+%**: promillage
- **ALT+?**: Spanish question mark
- **ALT+!**: Spanish exclamation mark
- **ALT+>**: a double >>
- **ALT+<**: a double <<

Many special characters are, in fact, a combination of two other characters, e.g. the letters with accents. First pressing **ALT+BACKSPACE**, and then pressing the desired combination can call these up. Some examples are given below.

- **ALT+BACKSPACE**, **AKEY**, **TILDE**: ã
- **ALT+BACKSPACE**, **AKEY**, **COMMA**: à
- **ALT+BACKSPACE**, **AKEY**, **ACCENT**: á

- **ALT+BACKSPACE**, **AKEY**, **OKEY**: å
- **ALT+BACKSPACE**, **EKEY**, **QUOTE**: ë
- **ALT+BACKSPACE**, **OKEY**, **SLASH**: ø

Complete ASCII files can also be added to a Text Object. Save this file as **/tmp/.cutbuffer** and press **ALT+V**.

Otherwise you can write your text in a Blender Text Window, or load a text into such a window, or paste it there from the clipboard and press **ALT-M**. This creates a new Text Object from the content of the text buffer (Up to 1000 characters).

Extrude Along Path

The "Extrude along path" technique is a very powerful modelling tool. It consists of creating a surface by sweeping a given profile along a given path.

Both the profile and the path can be a Bézier or a NURBS curve.

Let's assume you have added a Bézier curve and a Bézier circle as separate objects to your scene (Figure 7-25).

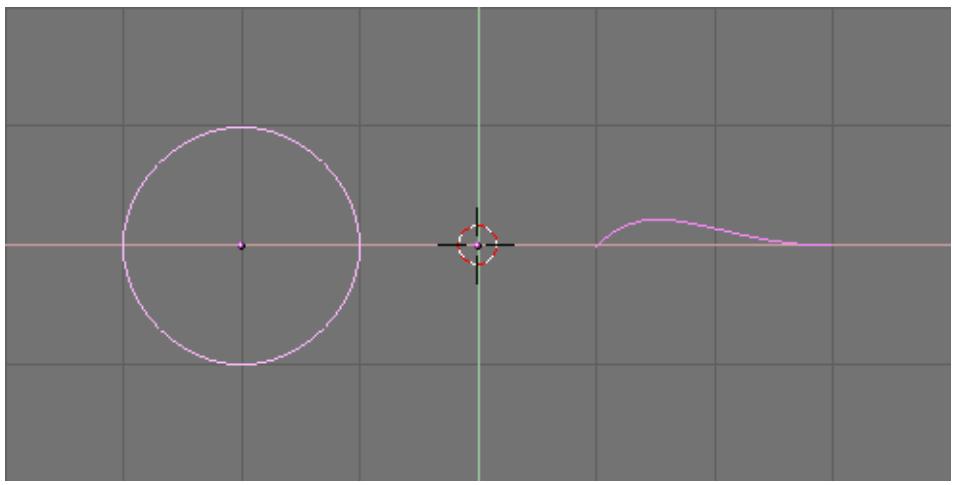


Figure 7-25. Profile (left) and path (right).

Play a bit with both to obtain a nice 'wing-like' profile and a fancy path (Figure 7-26). Please note that, by default, Béziers exist only on a plane, and are 2D objects. To make the path span in all three directions of space, as in the example, you must press the 3D button in the Curve EditButtons (**F9**) (Figure 7-27).

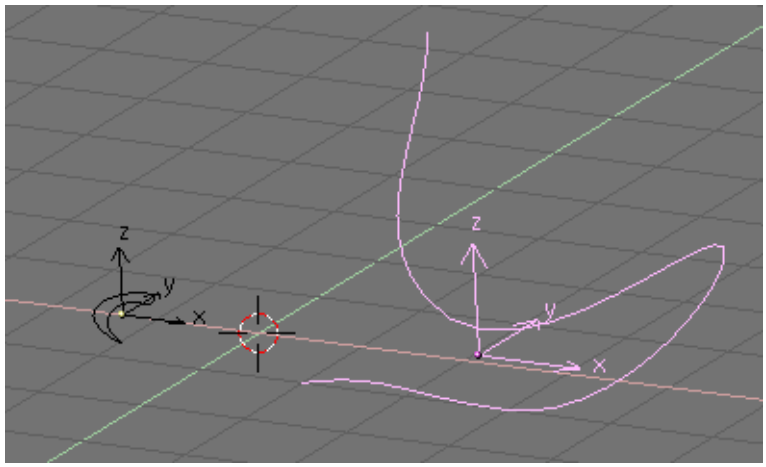


Figure 7-26. Modified profile (left) and path (right).

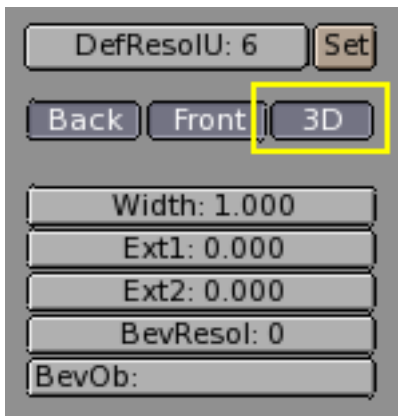


Figure 7-27. 3D Curve button.

Now look at the name of the profile object. By default it is "CurveCircle" and it is shown on the EditButton toolbar when it is selected. You can change it by **SHIFT-LMB** on the name, if you like (Figure 7-28).

Figure 7-28. Profile name.

Now select the Path. In its EditButton locate the one named `BevOb:` and write in there the name of the profile object. In our case "CurveCircle" (Figure 7-29).

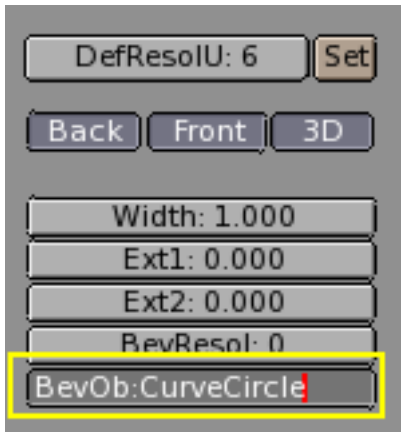


Figure 7-29. Specify the Profile on the Path.

The result is a surface defined by the Profile, sweeping along the path (Figure 7-30).

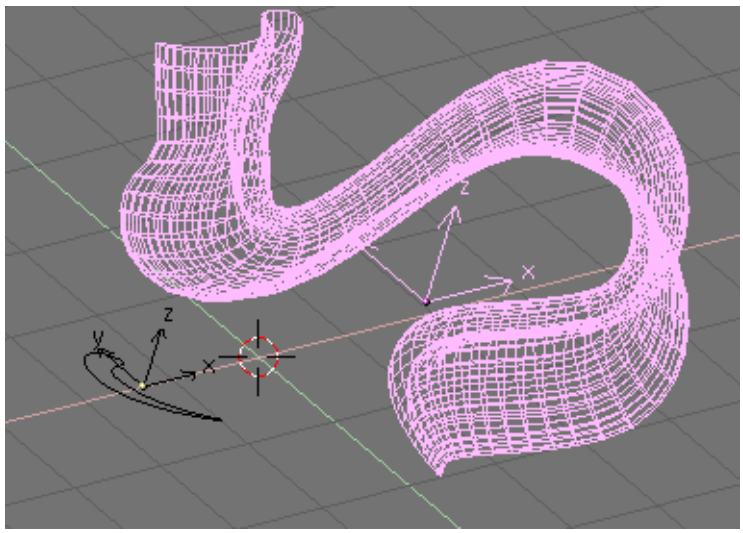


Figure 7-30. Extrusion result.

To understand the results, and hence obtain the desired effects it is important to understand the following points:

- The profile is oriented in such a manner that its z-axis is tangent (*i.e.* directed along) the Path and that its x-axis is on the plane of the Path; consequently the y-axis is orthogonal to the plane of the Path;
- If the Path is 3D the "plane of the path" is defined locally rather than globally and is visually rendered, in EditMode, by several short segments perpendicular to the Path (Figure 7-31);
- The y-axis of the profile always points upwards. This is often a source of unexpected results and problems, as it will be explained later on.

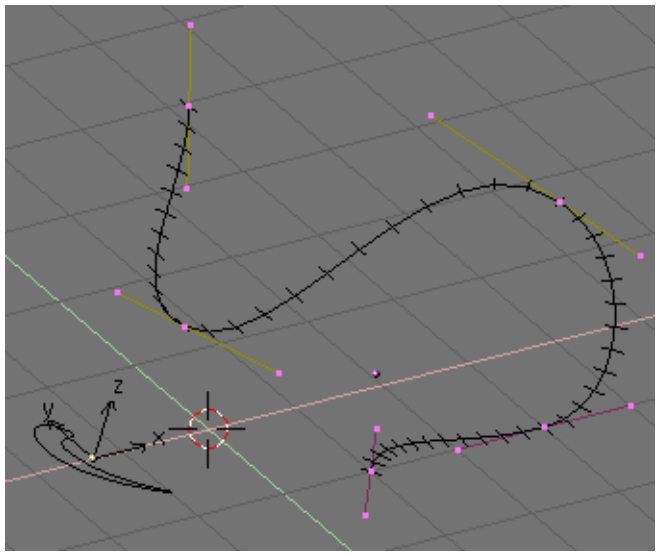


Figure 7-31. Path local plane.

> **Tilting:** You can modify the orientation of the local Path plane by selecting a control point and pressing **TKEY**. This done, moving the mouse changes the orientation of the short segments smoothly in the neighborhood of the control point. **LMB** fixes the position, **ESC** reverts to previous state.

With the y-axis constrained to being upwards, unexpected results can occur when the path is 3D and when the profile being extruded comes to a point where the path bends so that the y-axis of the profile should point downwards. If this occurs, there is an abrupt 180° rotation of the path so that the y-axis points upwards again.

Figure 7-32 clearly shows the problem. On the left there is a Path whose steepness is such that the normal to the local Path Plane is always upward. On the right a Path where, at the point circled in yellow, such a normal begins to point down. The result of the extrusion presents an abrupt turn there.

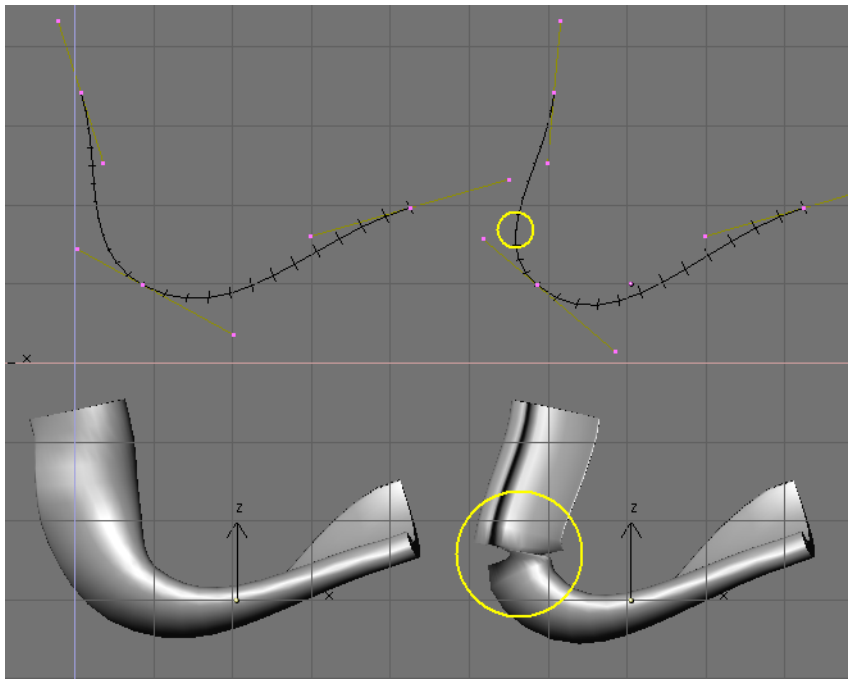


Figure 7-32. Extrusion problems due to y-axis constraint.

The only solutions to this problems are: To use multiple - matching - paths, or to carefully tilt the path to ensure a normal always pointing upwards.

> **Changing profile orientation:** If the orientation of the profile along the curve is not the expected one, and you want to rotate it for the whole Path length, there is a better way than tilting all Path control points.
>
> You can simply rotate the Profile in EditMode on its plane. This way the profile changes but its local reference doesn't.

Skinning

Skinning is the fine art of defining a surface by means of two or more profiles. In Blender you do so by preparing as many Curves of the the desired shape and then converting them to a single NURBS surface.

As an example we will create a sailing boat. The first thing to do, in side view (**NUM3**) is to add a Surface Curve. Beware, add a *Surface* curve and not a curve of Bézier or NURBS flavour, or the trick won't work (Figure 7-33).

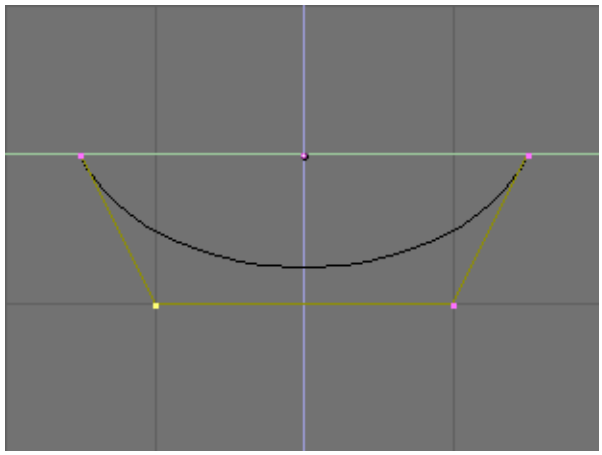


Figure 7-33. A Surface curve for skinning.

Give the curve the shape of the middle cross section of the ship, by adding vertices as needed with the Split button and, possibly, by setting the NURBS to 'Endpoint' both on 'U' and 'V' (Figure 7-34).

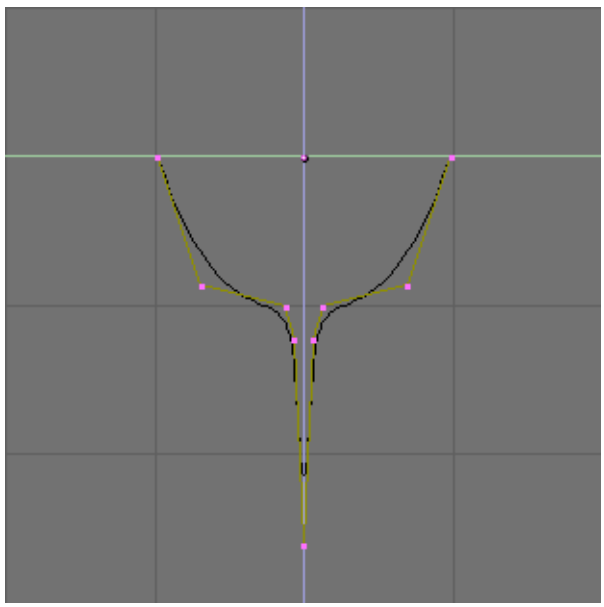


Figure 7-34. Profile of the ship.

Now duplicate (**SHIFT-D**) as many times as needed, to the left and to the right, the curve (Figure 7-35). Adjust the curves accordingly to the section of the ship at different points along its length. Having blueprints helps a lot. You can load a blueprint on the background as we did for the logo design in this same chapter to prepare all the cross section profiles (Figure 7-36).

Note that the surface which will be obtained will go smoothly from one profile to the next. To have abrupt changes it is then necessary to place profiles quite close one to the other, as it is the case for the selected profile in Figure 7-36.

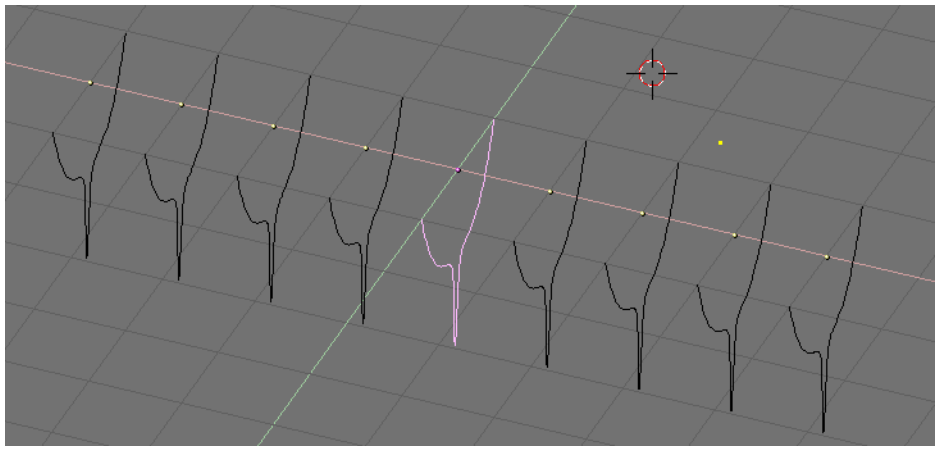


Figure 7-35. Multiple profiles along ship's axis.

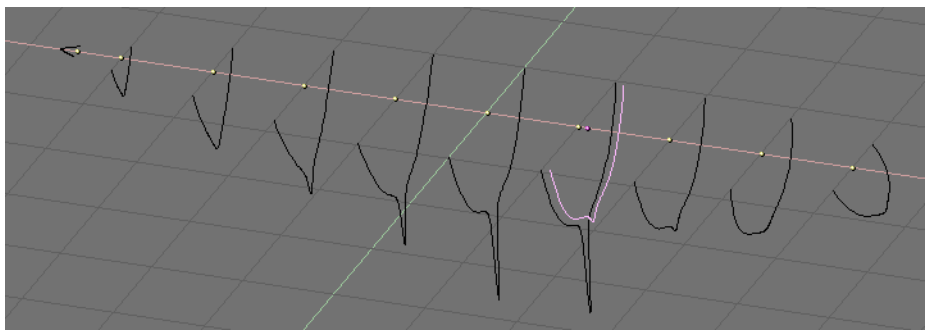


Figure 7-36. Multiple profiles of the correct shapes.

Now select all curves (with **AKEY** or **BKEY**) and join them together (by **CTRL-J** and by answering positively to the question 'Join selected NURBS?'). This will lead to the configuration of Figure 7-37

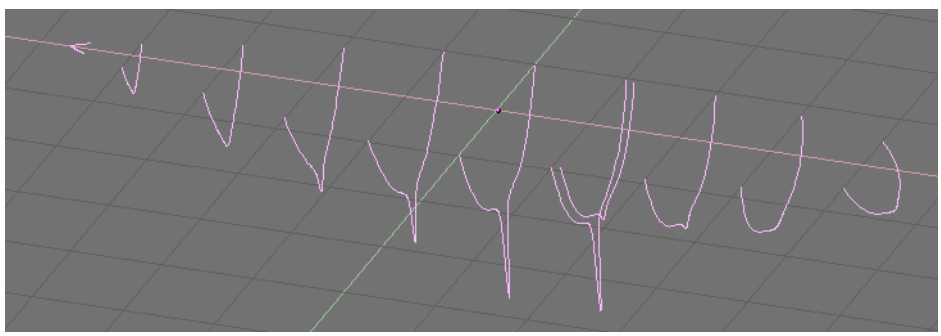


Figure 7-37. Joined profile.

Now switch to EditMode (**TAB**) and select all control points with **AKEY**; then press **FKEY**. The profiles will be 'skinned' and converted to a surface (Figure 7-38). Note that, as it is evident from the first and last profiles in this example, that the cross-sections need not to be coplanar.

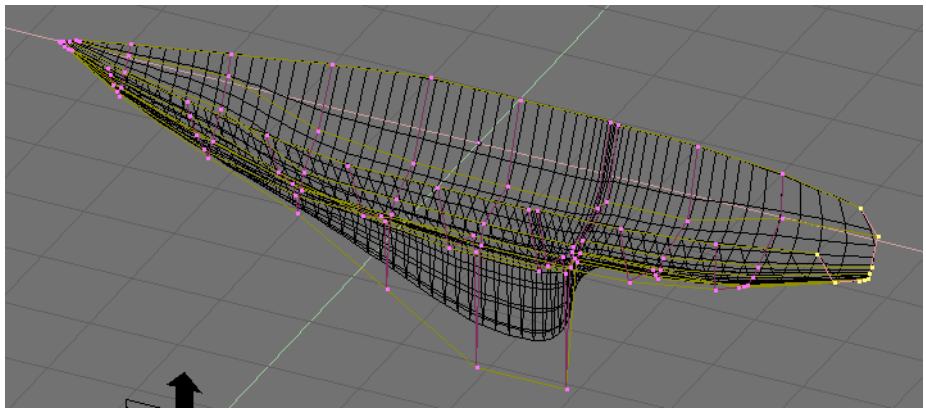


Figure 7-38. Skinned surface in edit mode.

You can then tweak the surface, if necessary, by moving the control points. Figure 7-39 shows a shaded view.

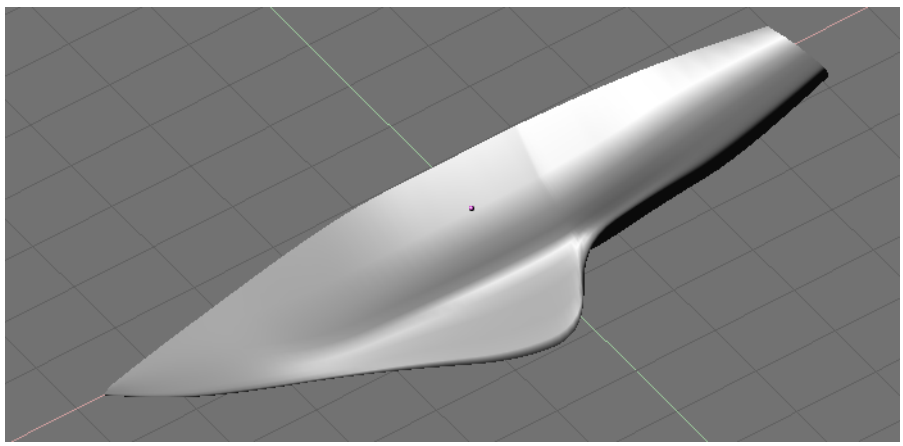


Figure 7-39. Final hull.

> **Profile setup:** The only limitation to this otherwise very powerful technique lies in the fact that all profiles must exhibit the same number of control points. This is why it is a good idea to model the most complex cross section first and then duplicate it, moving control points as needed, without adding or removing them, as is done in this example.

Resources

- *Curve rotoscoping: Creating a Logo* - http://www.vrotvrot.com/xoom/tutorials/logoTut/logoTut.html
- *Surface editing: Modeling a Dolphin* - http://www.vrotvrot.com/xoom/tutorials/Dolphin/UnderWater.ht
- *Skinning: The Cave of Torsan A* - http://www.vrotvrot.com/xoom/tutorials/Cave/Cave.html

Notes

1. http://www.vrotvrot.com/xoom/tutorials/logoTut/logoTut.html

2. http://www.vrotvrot.com/xoom/tutorials/Dolphin/UnderWater.html
3. http://www.vrotvrot.com/xoom/tutorials/Cave/Cave.html

Chapter 8. Materials and textures

Effective material design requires some understanding of 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 in getting the most from it.

The rendered image you obtain with Blender is a projection of the scene onto an imaginary surface called *viewing plane*. It is analogous to the film in a traditional camera, or the rods and cones in the human eye, but it receives simulated light, not real light. To render an image of a scene we must answer this question: what light from the scene is arriving at each point on the viewing plane?

This question is answered by following 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, then determining what light would strike that point. The surface properties and incident light angle tell us how much of that light would be reflected back along the incident viewing angle (Figure 8-1).

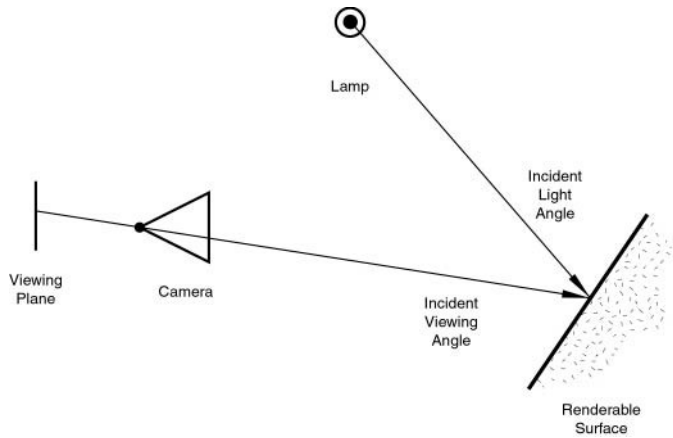


Figure 8-1. Rendering engine basic principle.

There are two basic types of phenomena which 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.

Diffusion

Light striking a surface and 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. This quality is why diffuse light is called *viewpoint independent*. Of course the amount of light effectively striking the surface does depend on the incident light angle. If most of the light striking a surface is being reflected diffusely, the surface will have a matte appearance (Figure 8-2).

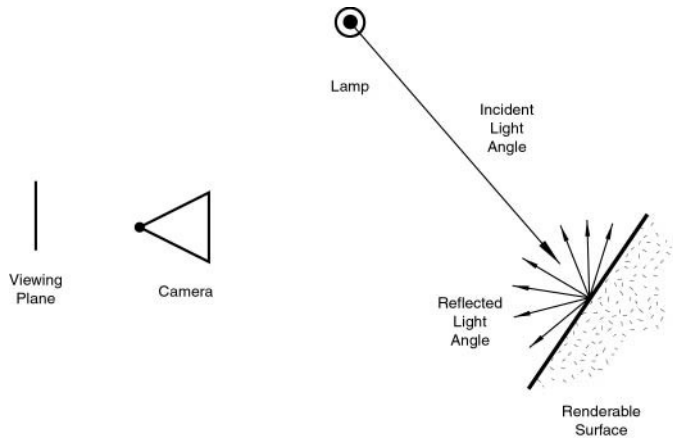


Figure 8-2. Light re-irradiated in the diffusion phenomenon.

Since version 2.28, Blender implements three different mathematical formulae to compute diffusion. And, even more notably, the diffusion and specular phenomena, which are usually bounded in a single type of material, have been separated so that it is possible to select diffusion and specular reflection implementation separately.

The three Diffusion implementations, or *shaders*, use two or more parameters each. The first two parameters are shared by all Diffuse Shaders and are the *Diffuse color*, or simply *color*, of the material, and the amount of incident light energy that is actually diffused. This latter quantity, given in a [0,1] range, is actually called `Refl` in the interface.

The implemented shaders are:

- *Lambert* - This was Blender's default diffuse shader up to version 2.27, so all old tutorials refer to this, and all pre-2.28 images were created with this. It only has the default parameters.
- *Oren-Nayar* - This is a new shader introduced in Blender 2.28. It has a somewhat more 'physical' approach to the diffusion phenomena inasmuch as, besides the two default parameters, it has a third one, determining the amount of microscopical roughness of the surface.
- *Toon* - This is a new shader introduced in Blender 2.28. It is a very 'un-physical' shader since it is not made to fake reality but to produce 'toonish' rendering, with clear light-shadow boundaries and uniform lit/shadowed regions. Notwithstanding its 'simplicity' it needs two more parameters, defining the size of the lit area and the sharpness of the shadow boundaries.

A subsequent section, devoted to the actual implementation of the material, will analyze all these and their relative settings.

Specular Reflection

Specular reflection, on the other hand, is *viewpoint dependent*. Light striking a specular surface, by Snell's Law, will be reflected at an angle which mirrors the incident light angle, so the viewing angle is very important. Specular reflection forms tight, bright highlights, making the surface appear glossy (Figure 8-3).

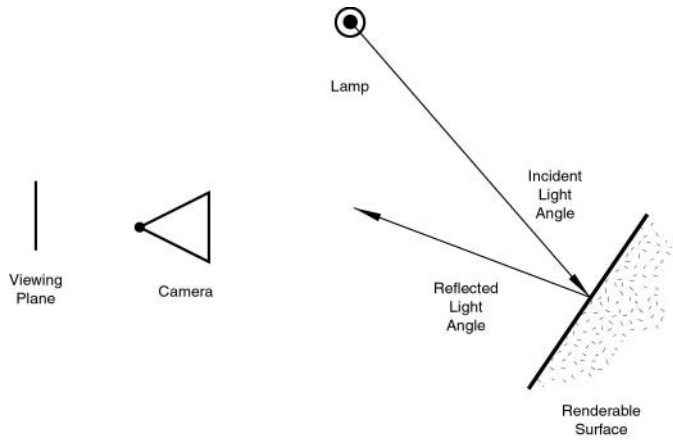


Figure 8-3. Specular Reflection.

In reality, Diffusion and Specular reflection are exactly the same process. Diffusion is seen on a surface which has so much small-scale roughness, with respect to wavelength, in the surface that light is reflected in many different directions from each tiny bit of the surface with tiny changes in surface angle.

Specular reflection appears on a surface with enough consistency in the angle of the surface that the light is reflected in a consistent direction, rather than being scattered. It's just a matter of the scale of the detail. If the surface roughness is much smaller than the wavelength of the incident light it appears flat and acts as a mirror.

It is not only a matter of wavelength but also a matter of the dimension of the object *on the rendered image* and, in particular, on the dimension of the rendered object with respect to the image pixel. An automobile's chrome fender normally looks shiny, but from a spacecraft it will appear as part of the diffuse detail of the planet. Likewise, sand has a matte appearance, but if you look at it through a microscope you will see smooth, shiny surfaces.

It is important to stress that the Specular reflection phenomenon treated here is not the reflection which occurs on a mirror, but rather light highlights on a glossy surface.

To obtain true mirror-like reflections you need a raytracer. Blender is not a raytracer as such, but it can produce convincing mirror-like surfaces via careful application of textures, as will be shown later on.

Similar to Diffusion, Specular reflection has a number of different implementations, or *specular shaders*. Again, each of these share two common parameters: the *Specular colour* and the energy of the specularity, in the [0-2] range - thus effectivelly allowing more energy to be shed as specular reflection as there is incident energy. It is important to note here that a material has therefore at least two different colours, a diffuse, and a specular one, the latter normally set to pure white, but which can be set to different values to obtain interesting effects.

The four specular shaders are:

- *CookTorr* - This was Blender's only Specular Shader up to version 2.27. Indeed, up to that version it was not possible to separately set diffuse and specular shaders and there was just one plain material implementation. Besides the two standard parameters it uses a third, *hardness*, which regulates how 'wide' the specular highlights are. The lower the hardness, the wider the highlights.
- *Phong* - A different mathematical algorithm to compute specular highlights, not very different from the previous, and governed by the same three parameters.
- *Blinn* - A more 'physical' specular shader, thought to match the Oren-Nayar diffuse one. It is more physical inasmuch as, besides the aforementioned three parameters, it adds a fourth, an *index of refraction (IOR)*. This is not actually used to compute

refraction of rays, a ray-tracer is needed for that, but to correctly compute specular reflection intensity and extension via Snell's Law. Hardness and Specular parameters give additional degrees of freedom.

- *Toon* - This specular shader matches the Toon diffuse one. It is designed to produce the sharp, uniform highlights of toons. It has no hardness but rather a Size and Smooth pair of parameters which dictates extension and sharpness of the specular highlights.

Thanks to this flexible implementation Blender allows us to easily control how much of the incident light striking a point on a surface is diffusely scattered, how much of it is reflected as specularity, and how much of it is not reflected at all. This determines in what directions (and in what amounts) the light is reflected from a given light source or, to look at it another way, from what sources (and in what amounts) the light is being reflected toward a given point on the viewing plane.

It is very important to remember that the material colour is just one element in the rendering process. What actually determines the colour seen in the rendered image also depends on the colour of the light illuminating the object. To put it simply, the colour is the product of the light colour and the material colour.

Materials in practice

In this section we will analyze how to set up the various material parameters in Blender, and what you should expect as a result.

Once an Object is selected by pressing the **F5** key or  the material buttons window appears (Figure 8-4). Of these buttons, the left block (Figure 8-5) is strictly relevant to material shaders, while the right block is relevant to material textures and will be analyzed in the pertinent section.

Figure 8-4. Material Buttons.

Figure 8-5. Material Buttons strictly pertinent to material shaders.

In this block the leftmost sub-block presents the material preview. By default it is a plane seen from the top, but it can be set to a sphere or to a cube with the buttons on top of the preview window (Figure 8-6).

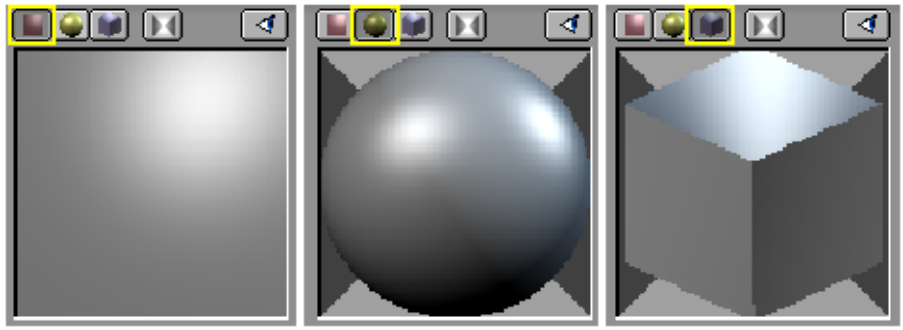


Figure 8-6. Material Preview, plane (left) sphere (middle) and cube (right).

Material Colours

The next group of buttons (Figure 8-7) determines the material colours.

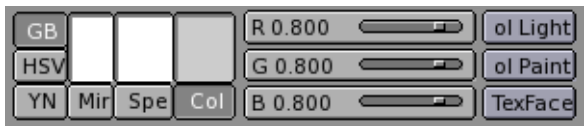


Figure 8-7. Material colours buttons.

Each material can exhibit up to three colours:

- *The basic material colour*, or the Diffuse colour, or, briefly the *Color* tout court (`Col` button in the interface) which is the colour used by the diffuse shader.
- *The Specular colour*, indicated by the `Spe` button in the interface, is the colour used by the specular shader.
- *The Mirror colour*, indicated by the `Mir` button in the interface, is the colour used by special textures to fake mirror reflections. More information on this will be given in the Environment Mapping section.

The aforementioned buttons select the pertinent colour, which is shown in preview immediately above the button. The three sliders on the right allow you to change the values for the colour both in a RGB scheme and in a HSV scheme. You can select these schemes via the `RGB` and `HSV` buttons on the far left.

The `DYN` button is used to set the Dynamic properties of the Object in the RealTime engine, which is outside the scope of this manual, while the three buttons on the far right are relative to advanced *Vertex Paint* and *UV Texture*.

The Shaders

Underneath the colour buttons there are the shader buttons (Figure 8-8). On the top, the two pop-up menus allow you to select one diffuse shader (on the right, Figure 8-9) and one specular shader (on the left, Figure 8-10).

Figure 8-8. Material colours buttons.

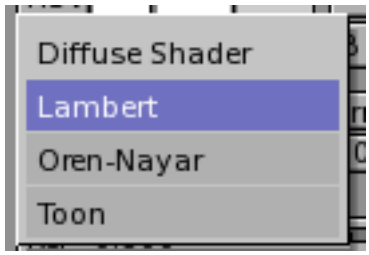


Figure 8-9. Material Diffuse shaders.

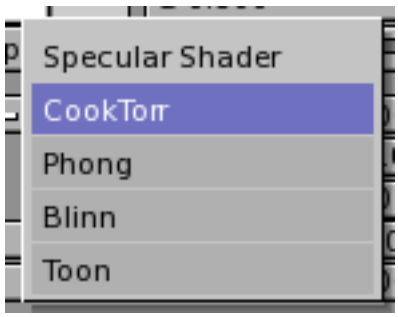


Figure 8-10. Material Specular shaders.

Below these there are two sliders, valid for all shaders, determining the intensity of the Diffusion and Specular phenomena. The `Ref` slider has a 0 to 1 range whereas the `Spec` has a 0 to 2 range. Strictly Physically speaking, if A is the light energy impinging on the object, then Ref times A is the energy diffused and $Spec$ times A is the energy specularly reflected and, to be physically correct it must be $Ref + Spec < 1$ or the object would radiate more energy that it receives.

But this is CG, so don't be too strict on physics.

Textures (-)

(to be written)

Texture plugins (-)

(to be written)

Environment Maps

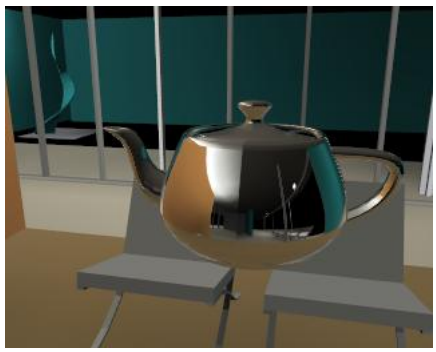


Figure 8-11. Environment Map Example

Introduction

This rendering technique uses texture mapping to mimic a mirroring surface. From a carefully choosen location, six images are rendered, each image representing the view from a side of a cube. These images can then be used as a 'look up table' for the reflections of the environment.

The usage of a cubical environment map allows the freedom to position the camera at any location in the environment, without the need to recalculate the map.

Figure 8-12. Environment Map 'Lookup table'

An environment map renders as if it were an Image texture in Blender. Environment map textures thus have a good filtering, use mipmapping and have all the antialiasing features of Image textures. In most cases an environment map is used to add the 'feeling' of reflection, it can be highly filtered (for metallic unsharp reflections) and be re-used with Materials of other Objects without annoying visual errors.

By default, the faces of all Objects that define the location of an environment map are not rendered in environment maps.

The EnvMap buttons

Figure 8-13. ...

Blender allows three types of environment maps:

- *Static (RowBut)* - The map is only calculated once during an animation or after loading a file.
- *Dynamic (RowBut)* - The map is calculated each time a rendering takes place. This means moving Objects are displayed correctly in mirroring surfaces.
- *Load (RowBut)* - When saved as an image file, environment maps can be loaded from disk. This option allows the fastest rendering with environment maps.

Other options are:

- *Free Data (But)* - This action releases all images associated with the environment map. This is how you force a recalculation when using a Static map.
- *Save EnvMap (But)* - You can save an environment map as an image file, in the format indicated in the DisplayButtons (F10).

Figure 8-14. Loading an environment map

These buttons are drawn when the environment map type is "Load". The environment map image then is a regular Image block in the Blender structure.

- *Load Image (But)* - The (largest) adjacent window becomes an ImageSelectWindow. Specify here what file to read in as environment map.
- *...(But)* - This small button does the same thing, but now gives a FileSelect.
- *ImageBrowse (MenuBut)* - You can select a previously loaded map from the list provided. EnvMap Images can be reused without taking up extra memory.
- *File Name (TextBut)* - Enter an image file name here, to load as an environment map.
- *Users (But)* - Indicates the number of users for the Image.
- *Reload (But)* - Force the Image file to be read again.

Figure 8-15. Settings

- *Ob: (TextBut)* - Fill in the name of an Object that defines the center and rotation of the environment map. This can be any Object in the current Scene.
- *Filter: (NumBut)* - With this value you can adjust the sharpness or blurriness of the reflection.
- *Clipsta, ClipEnd (NumBut)* - These values define the clipping boundaries when rendering the environment map images.
- *CubeRes (NumBut)* - The resolution in pixels of the environment map image.

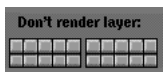


Figure 8-16. Selecting layers

- *Don't render layer* - Indicate with this option that faces that exist in a specific layer are NOT rendered in the environment map.

UV editor and FaceSelect

Introduction

The UV-Editor allows you to map textures directly on the faces of Meshes. Each face can have individual texture coordinates and an individual image assigned to it. You can also combine it with vertex colors to make the texture brighter/darker or give it a colour.

For each face there are two extra features added:

- *four UV coordinates* - These define the way an Image or a Texture is mapped on the face. It are 2D coordinates, that's why it is called UV do distinguish from XYZ coordinates. These coordinates can be used for rendering or for realtime OpenGL display.
- *a link to an Image* - Every face in Blender can have a link to a different Image. The UV coordinates define how this image is mapped to the face. This image then can be rendered or displayed in realtime.

A 3D window has to be in "Face Select" mode to be able to assign Images or change UV coordinates of the active Mesh Object.

Assigning Images to faces

First you add a Mesh Object to your Scene, next is to enter the FaceSelect Mode with **F-KEY** or by pressing the FaceSelect Button in the 3DWindow header.

Figure 8-17. Orange triangle button: FaceSelect Mode in the 3DWindow header

Your Mesh will now be drawn Z-buffered, if you enter the Textured draw mode (**ALT-Z**, also called "potato mode") you will see your Mesh drawn in purple, which indicates that there is currently no Image assigned to these faces. Now press **AKEY** and all the faces of the Mesh will be selected and drawn as dotted lines.

Then change one Window into the Image Window with **SHIFT-F10**. Here you can load or browse an Image with the "Load" button. Images have to be in the power of 64 pixels (64x64, 128x64 etc.) to be able to drawn in realtime (note: most 3D cards don't support images larger than 256x256 pixels). However, Blender can render all assigned Images regardless the size.

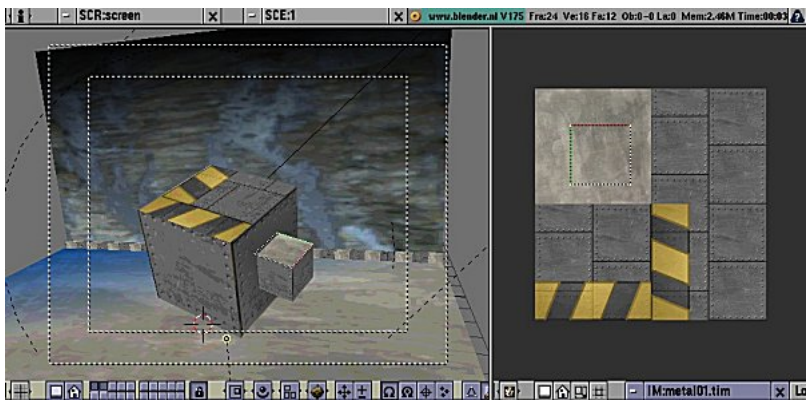


Figure 8-18. 3Dwindow and ImageWindow

Loading or browsing an Image in FaceSelect automatically assigns the Image to the selected faces. You can immediately see the this in the 3D window (when in Textured view mode).

Selecting faces

You can select faces with RightMouse or with BorderSelect (**BKEY**) in the 3D window. If you have problems with selecting the desired faces, you can also enter EditMode and select the vertices you want. After leaving EditMode the faces defined by the selected vertices are selected as well.

Only one face is active. Or with other words: the Image Window only displays the image of the active face. As usual within Blender, only the last selected face is active and this can only be done with a RightMouse click.

Only one face is active. Or with other words: the Image Window only displays the image of the active face. As usual within Blender, only the last selected face is active and this can only be done with a RightMouse click.

Editing UV coordinates

In the ImageWindow you will see a representation of your selected faces as yellow or purple vertices connected with dotted lines. You can use the same techniques here

as in the Mesh EditMode, to select, move, rotate, scale etc. With the "Lock" button pressed you will also see a realtime feedback in 3D what you are doing.

In the 3D window; you can press **UKEY** in FaceSelect mode to get a menu to calculate UV coordinates for the selected faces.

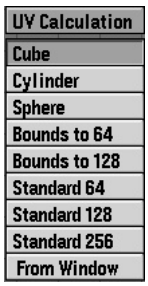


Figure 8-19. ..

- *Cube* - Cubical mapping, a number requester asks for a scaling property.
- *Cylinder, Sphere* - Cylindrical/spherical mapping, calculated from the center of the selected faces
- *Bounds to 64, 128* - UV coordinated are calculated using the projection as displayed in the 3D window. Then scaled to a boundbox of 64 or 128 pixels.
- *Standard 64, 128, 256* - Each face gets a set of default square UV coordinates.
- *From Window* - UV coordinated are calculated using the projection as displayed in the 3D window.

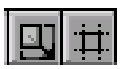


Figure 8-20. ..

New options In the ImageWindow: the first button keeps your UV polygons square while editing them, the second clips your UV polys to the size of the Image.

Some tips:

- Press **RKEY** in the 3D window to get a menu that allows rotating the UV coordinates.
- Sometimes it is necessary to move image files to a new location at your harddisk. Press **NKEY** in the ImageWindow to get a "Replace Image name" menu. You can fill in the old directory name, and the new one. Pressing "OK" changes the paths of all images used in Blender using the old directory. (Note: use as new directory the code "//" to indicate the directory where the Blender file is in).
- You can also use FaceSelect and VertexPaint (**VKEY**) simultaneously. Vertex painting then only works at the selected faces. This feature is especially useful to paint faces as if they don't share vertices. Note that the vertex colors are used to modulate the brightness or color of the applied image texture.

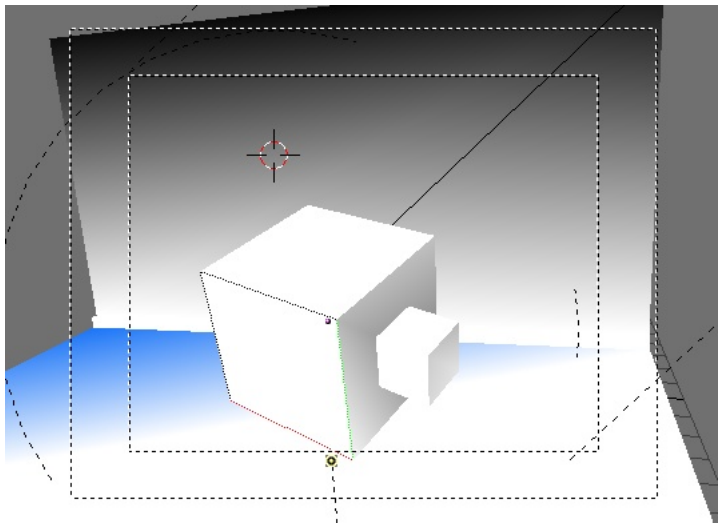


Figure 8-21. vertex colors modulate texture

Rendering and UV coordinates

Even without an Image assigned to faces, you can render textures utilizing the UV coordinates. For this, use the green "UV" button in the MaterialButtons (**F5**) menu.

If you want to render the assigned Image texture as well, you will have to press the "TexFace" button in the MaterialButtons. Combine this with the "VertexCol" option to use vertex colors as well.

Chapter 9. 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.

Lighting, sadly, is often overlooked by the inexperienced artist who commonly believes that, since real world scenes are often lit by a single light (a lamp, the sun, etc.) a single light would also do in computer graphics.

This is false because in the real world even if a single light source is present, the light shed by such a source bounces off objects and is re-irradiated all over the scene making shadows soft and shadowed regions not pitch black, but partially lit.

The physics of light bouncing is simulated by Ray Tracing renderers and can be simulated within Blender by resorting to the Radiosity (Chapter 15) engine.

Ray tracing and radiosity are slow processes. Blender can perform much faster rendering with its internal scanline renderer. A very good scanline 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 hypothesis.

In this chapter we will analyze the different type of lights in Blender and their behavior, we will analyze their strong and weak points, ending with describing a basic 'realistic' lighting scheme, known as the three point light method, as well as more advanced, realistic but, of course, CPU intensive, lighting schemes.

Lamp Types

Blender provides four Lamp types:

- Sun Light
- Hemi Light
- Lamp Light
- Spot Light

Any of these lamps can be added to the scene by pressing **SPACE** and by selecting the **Lamp** menu entry. This action adds a **Lamp Light** lamp type. To select a different lamp type, or to tune the parameters, you need to switch to the lamp buttons window Figure 9-1 (**F4** or ).

A row of toggle buttons, top left, allows you to choose the lamp type.

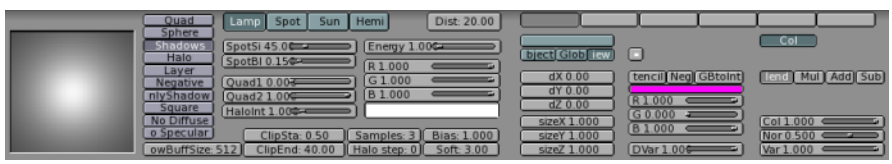


Figure 9-1. Lamp Buttons.

The lamp buttons can be divided into two categories: Those directly affecting light, which are clustered to the left, and those defining textures for the light, which are

on the right and are very similar to those relative to materials. In the following subsections we will focus on the first category (Figure 9-2), leaving a brief discussion on texture to the **Tweaking Light** section.

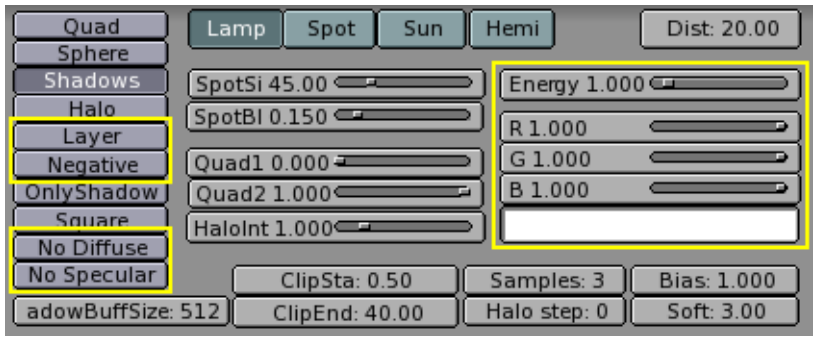


Figure 9-2. Lamp General Buttons.

The leftmost column of buttons is mainly devoted to Spot lights, but there are four buttons which have an effect on all four lamp types, and which deserve to be explained before going into the details of each type.

`Layer` - makes the light shed by the lamp affect only the objects which are on the same layer as the lamp itself.

`Negative` - makes the light cast 'negative' light, that is, the light shed by the lamp is subtracted, rather than added, to that shed by any other light in the scene.

`No Diffuse` - makes the light cast a light which does not affect the 'Diffuse' property of a material, hence giving only 'Specular' highlights.

`No Specular` - makes the light cast a light which does not affect the 'Specular' property of a material, hence giving only 'Diffuse' shading.

The central column is again devoted mainly to Spot lights and will not be treated here. Of the rightmost column of the first category of buttons, four are of general use:

`Energy` - the energy radiated by the lamp.

`R, G, B` sliders - the red, green and blue components of the light shed by the lamp.

Sun Light

The simplest light type is probably the Sun Light (Figure 9-3). A Sun Light is a light of constant intensity coming from a given direction. In the 3D view the sun light is represented by an encircled yellow dot, which of course turns to purple when selected, plus a dashed line.

This line indicates the direction of the sun's rays. It is by default normal to the view in which the sun lamp was added to the scene and can be rotated by selecting the sun and by pressing **RKEY**.

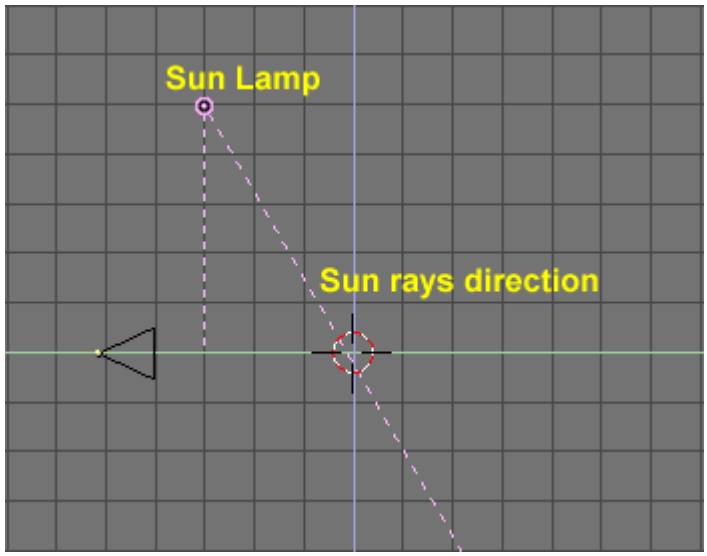


Figure 9-3. Sun Light.

The lamp buttons which are of use with the sun are plainly those described in the 'general' section. An example of sun light illumination is reported in Figure 9-4. As is evident, the light comes from a constant direction, has an uniform intensity and *does not cast shadows*.

This latter is a very important point to understand in Blender: no lamp, except for the "Spot" type, casts shadows. The reason for this lies in the light implementation in a scanline renderer and will be briefly discussed in the 'Spot' and 'Shadows' subsections.

Lastly, it is important to note that since the Sun light is defined by its energy, color and *direction*, the actual *location* of the Sun light itself is not important.

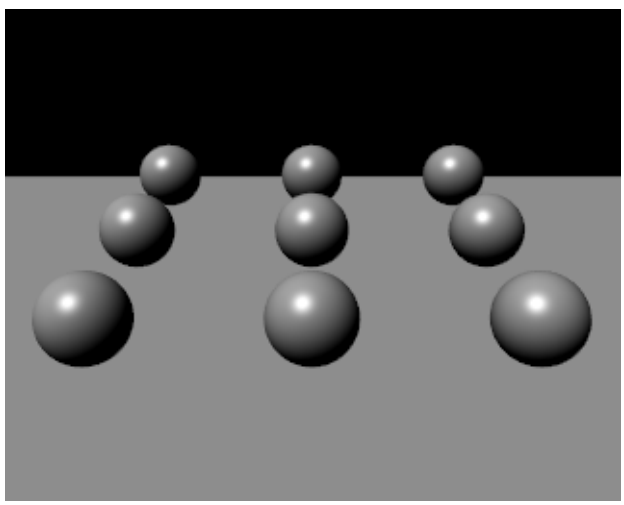


Figure 9-4. Sun Light example.

Figure 9-5 shows a second set-up, made by a series of planes 1 blender unit distant one from the other, lit with a Sun light. The uniformity of lighting is even more evident. This picture will be used as a reference to compare with other lamp types.

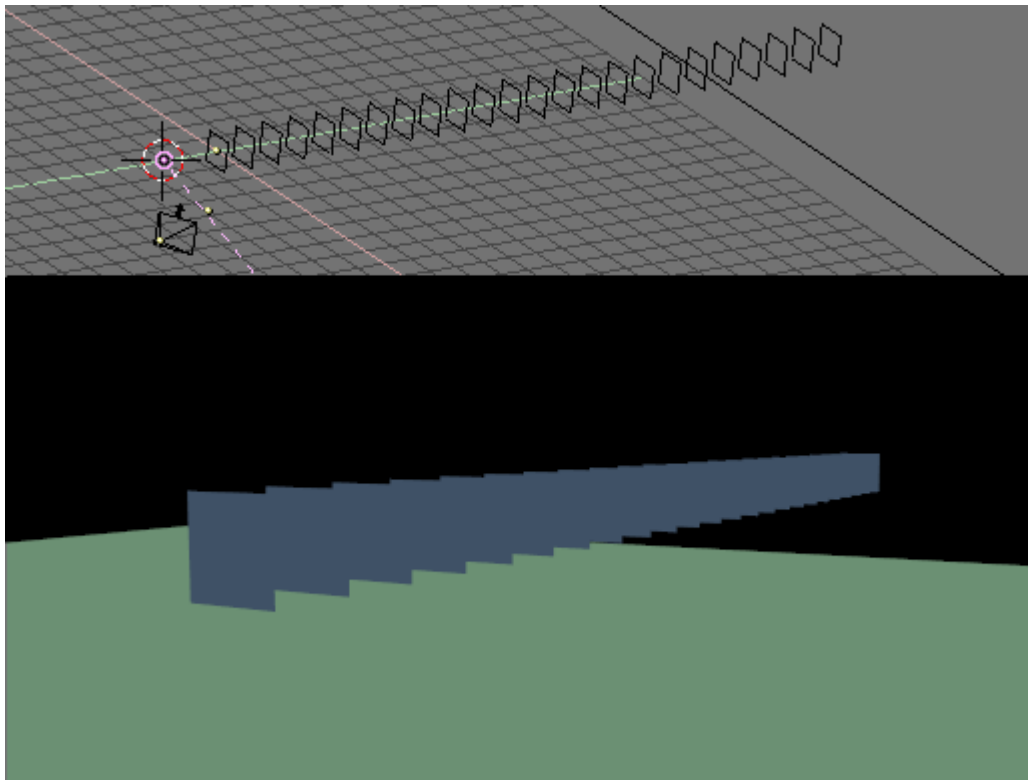


Figure 9-5. Sun Light example.

Sun Tips: A Sun light can be very handy for a uniform clear day-light open-space illumination. The fact that it casts no shadows can be circumvented by adding some 'shadow only' spot lights. See the Tweaking Light section!

Hemi Light

The Hemi light is a very peculiar kind of light 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 hemisphere surrounding the scene (Figure 9-6).

It is probably the least used Blender light, but it deserves to be treated before the two main Blender Lights because of its simplicity.

This light set-up basically resembles that of a Sun light. Its location is unimportant, while its orientation is important. Its dashed line represents the direction in which the maximum energy is radiated, that is the normal to the plane defining the cut of the hemisphere, pointing towards the dark side.

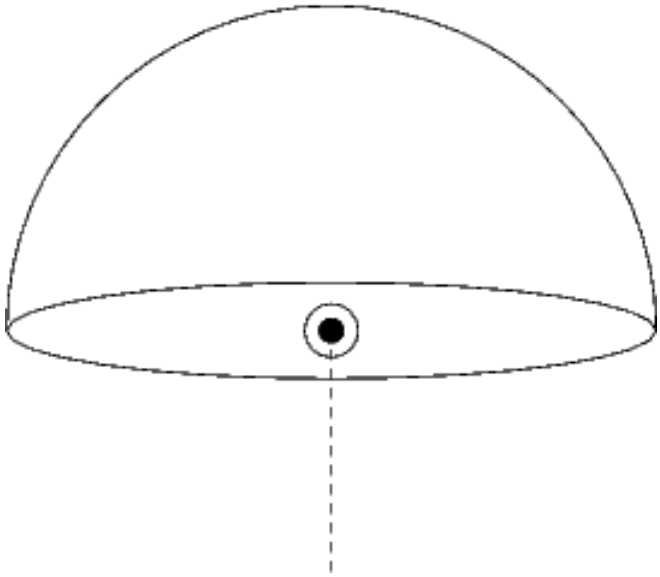


Figure 9-6. Hemi Light conceptual scheme.

The results of an Hemi Light for the 9 sphere set up are shown in Figure 9-7 the superior softness of the Hemi light in comparison to the sun light is evident.

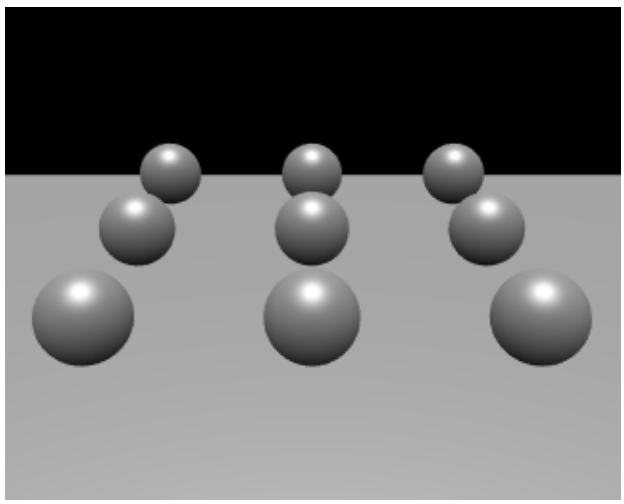


Figure 9-7. Hemi Light example.

Hemi Light Tip: To achieve quite realistic, were it not for the absence of shadows, outdoor lighting you can use both a Sun light, say of Energy 1.0 and warm yellow/orange tint, and a weaker bluish Hemi light faking the light coming from every point of a clear blue sky. Figure 9-8 shows an example with relative parameters. The figure also uses a World. See the pertinent chapter.

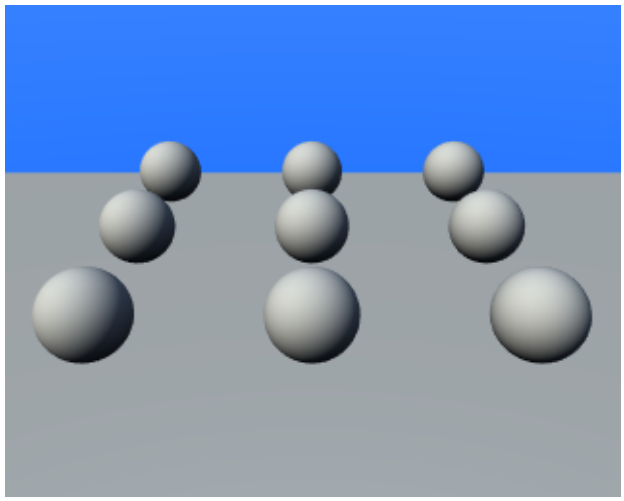


Figure 9-8. Outdoor Light example. Sun Light Energy=1 RGB=(1.,0.95,0.8) Sun direction in a polar reference is (135°,135°). Hemi Light Energy=0.5 RGB=(0.64,0.78,1.) pointing down.

Lamp Light

The Lamp light is an omni-directional point light, that is a dimensionless point radiating the same amount of light in all directions. In blender it is represented by a plain, circled, yellow dot.

Being a point light source the light rays direction on an object surface is given by the line joining the point light source and the point on the surface of the object itself. Furthermore, light intensity decays accordingly to a given ratio of the distance from the lamp.

Besides the above-mentioned buttons three more buttons and two sliders are of use in a Lamp light (Figure 9-9):

`Distance` - this gives, indicatively, the distance at which the light intensity is half the Energy. Objects closer than that receive more light, object further than that receive less light.

`Quad` - If this button is off, a linear - rather unphysical - decay ratio with distance is used. If it is on, a more complex decay is used, which can be tuned by the user from a fully linear, as for Blender default, to a fully - physically correct - quadratic decay ratio with the distance. This latter is a little more difficult to master and will be explained later on.

`Sphere` - If this button is pressed the light shed by the source is confined in the Sphere of radius Distance rather than going to infinity with its decay ratio.

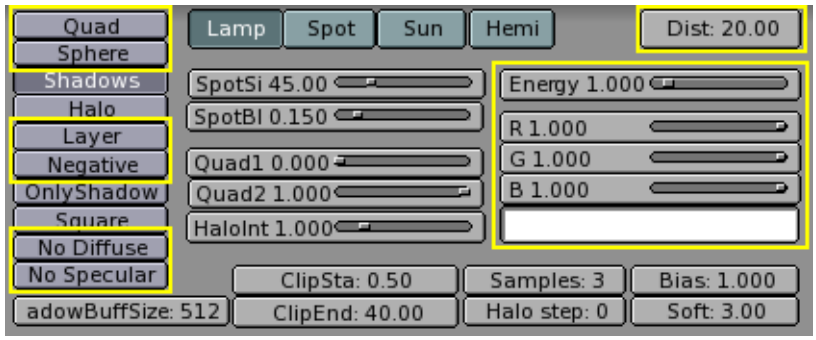


Figure 9-9. Lamp Light Buttons.

Following Figure 9-10 shows the same set-up as in the latter Sun light example, but with a Lamp light of different Distance values and with Quadratic decay on and off.

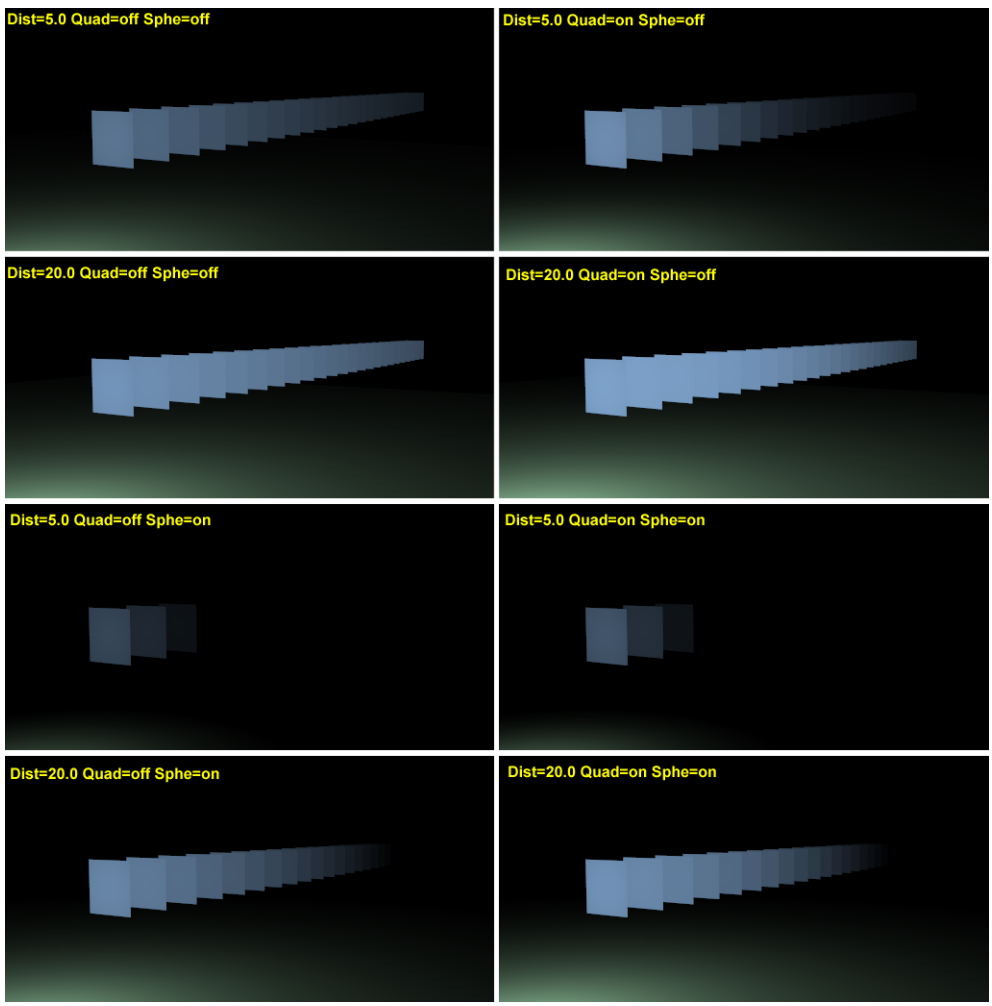


Figure 9-10. Lamp Light example. In Quad examples Quad1=0, Quad2=1.

The effect of the Distance parameter is very evident, while the effect of the Quad button is more subtle. In any case the absence of shadows is still a major issue. As a matter of fact only the first plane should be lit, because all the others should fall in the shadow of the first.

For the Math enthusiasts, and for those desiring deeper insight, the laws governing the decay are the following.

Let D be the value of the Distance Numeric Button, E the value of the Energy slider and r the distance from the Lamp to the point where rhe light intensity I is to be computed.

If Quad and Sphere buttons are off:

$$I = E\frac{D}{D+r}$$

It is evident what affirmed before: That the light intensity equals half the energy for $r = D$.

If Quad Button is on Q1 and Q2 are the values of the Quad1 and Quad2 sliders, respectively:

$$I = E\frac{D}{D+Q_1 r}\frac{D^2}{D^2+Q_2 r^2}$$

This is a little more complex and depends from the Quad1 (Q_1) and Quad2 (Q_2) values. Nevertheless it is apparent how the decay is fully linear for Q_1=1, Q_2=0 and fully quadratic for Q_1=0, Q_2=1, this latter being the default. Interestingly enough if Q_1= Q_2=0 then light intensity does not decay at all.

If the Sphere button is on the above computed light intensity I is further modified by multiplication by the term which has a linear progression for r from 0 to D and is identically 0 otherwise.

If the Quad button is off and the Sphere button is on:

$$I = E\frac{D}{D+r}\cdot\begin{cases}\frac{D-r}{D} & \text{if } r<D\\ 0 & \text{otherwise}\end{cases} =$$

$$=\begin{cases}E\frac{D-r}{D+r} & \text{if } r<D\\ 0 & \text{otherwise}\end{cases}$$

If both Quad and Sphe buttons are on:

$$I = E\frac{D}{D+Q_1 r}\frac{D^2}{D^2+Q_2 r^2}\cdot\begin{cases}\frac{D-r}{D} & \text{if } r<D\\ 0 & \text{otherwise}\end{cases} =$$

$$=\begin{cases}E\frac{D-r}{D+Q_1 r}\frac{D^2}{D^2+Q_2 r^2} & \text{if } r<D\\ 0 & \text{otherwise}\end{cases}$$

Figure 9-11 might be helpful in understanding these behaviors graphically.

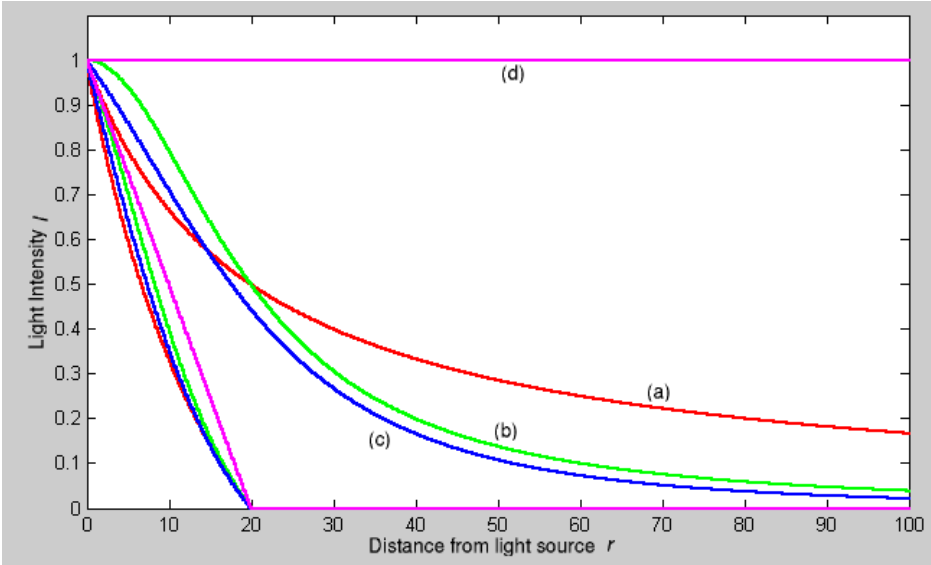


Figure 9-11. Light decays: a) Blender default linear; b) Blender default quadratic with Quad1=0, Quad2=1; c) Blender quadratic with Quad1=Quad2=0.5; d) Blender quadratic with Quad1=Quad2=0. Also shown in the graph the same curves, in the same colors, but with the Sphere button turned on.

Lamp Light Tip: Since the Lamp light does not cast shadows it shines happily 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.

Spot Light

The Spot light is the most complex of Blender lights and indeed among the most used thanks to the fact that it is the only one able to cast shadows.

A Spot light is a cone shaped beam generated from the light source location, which is the tip of the cone, in a given direction. Figure 9-12 should clarify this.

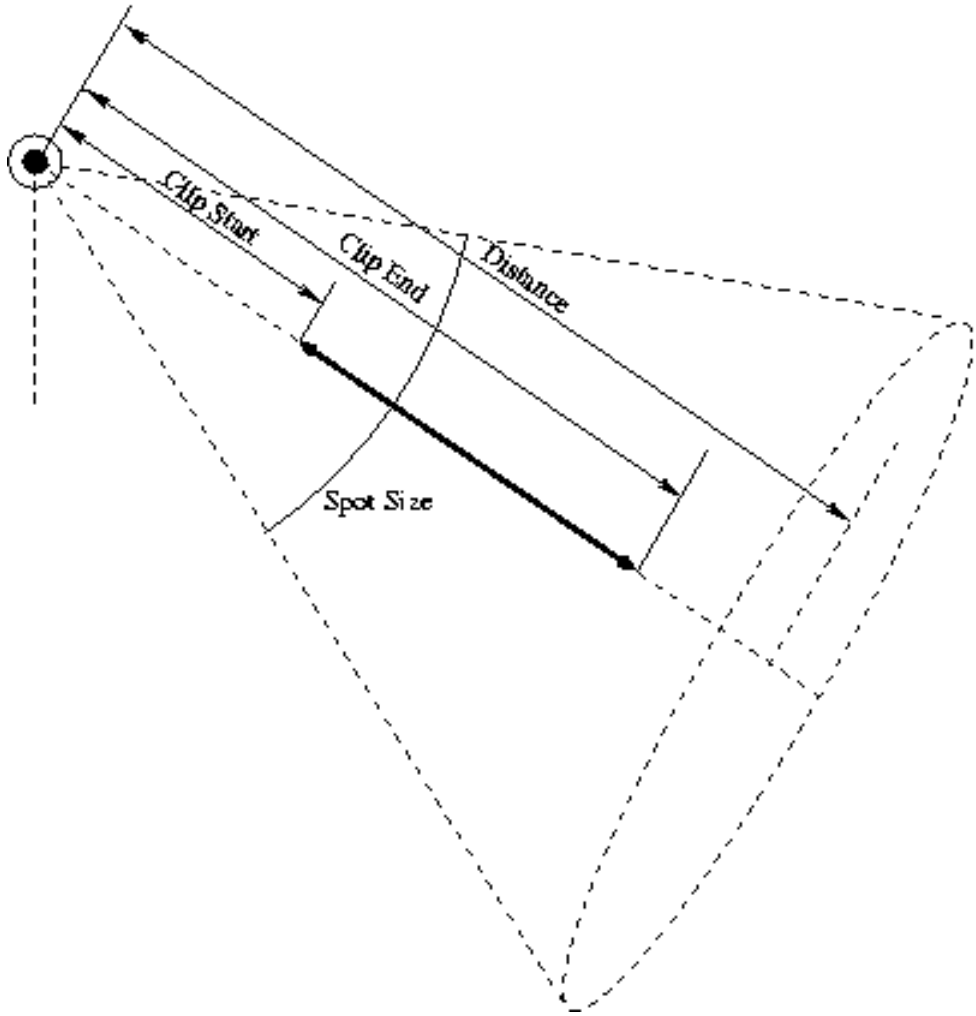


Figure 9-12. Spot Light Scheme.

The Spot light uses all of the buttons. This needs now a new, separate, througthful description.

Lamp Options

Figure 9-13. The Lamp Options buttons

Besides the `Negative` and `Layer` buttons whose utilization is known, and the `Quad` and `Sphere` buttons, whose effect is the same as for the Lamp light, the other buttons (Figure 9-13) meanings are:

`Shadows` - toggles shadow casting on and off for this spot. *Beware* Blender won't render shadows anyway, unless shadows are enabled at a *global* level in the rendering buttons window (**F12** or)

`Halo` - let the spot cast a halo as if the light rays were passing through a hazy medium. This option is explained later on in the 'Volumetric Light' section.

`Only Shadow` - let the spot cast only the shadow and no light. This option will be analyzed later on it the 'Tweaking Light' section.

`Square` - Spot lights usually by default cast a cone of light of circular cross-section. There are cases where a square cross section would be helpful, and indeed have a pyramid of light rather than a cone. This button toggles this option.

Spot Buttons

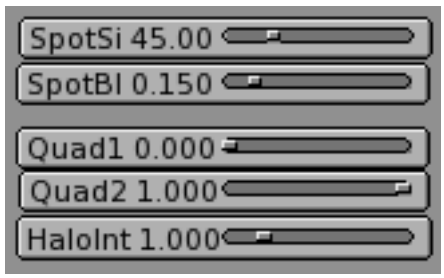


Figure 9-14. Spot Light Buttons.

The central column of buttons (Figure 9-14):

`SpotSi` - the angle at the tip of the cone, or the Spot aperture.

`SpotBl` - the blending between the light cone and the surrounding unlit area. The lower the sharper the edge, the higher the softer. Please note that this applies only to the spot edges, not to the softness of the edges of the shadows cast by the spot, these latter are governed by another set of buttons described in the 'Shadows' subsection.

`Quad1, Quad2` - has the same meaning as for the Lamp light.

`HaloInt` - If the Halo button is On this slider defines the intensity of the spot halo. Again, you are referred to the 'Volumetric Light' section.

The last button group of the Spot light governs shadows and it is such an ample topic that it deserves a subsection by its own. Before switching to Shadows, Figure 9-15 shows some results for a Spot light illuminating our first test case for different configurations.

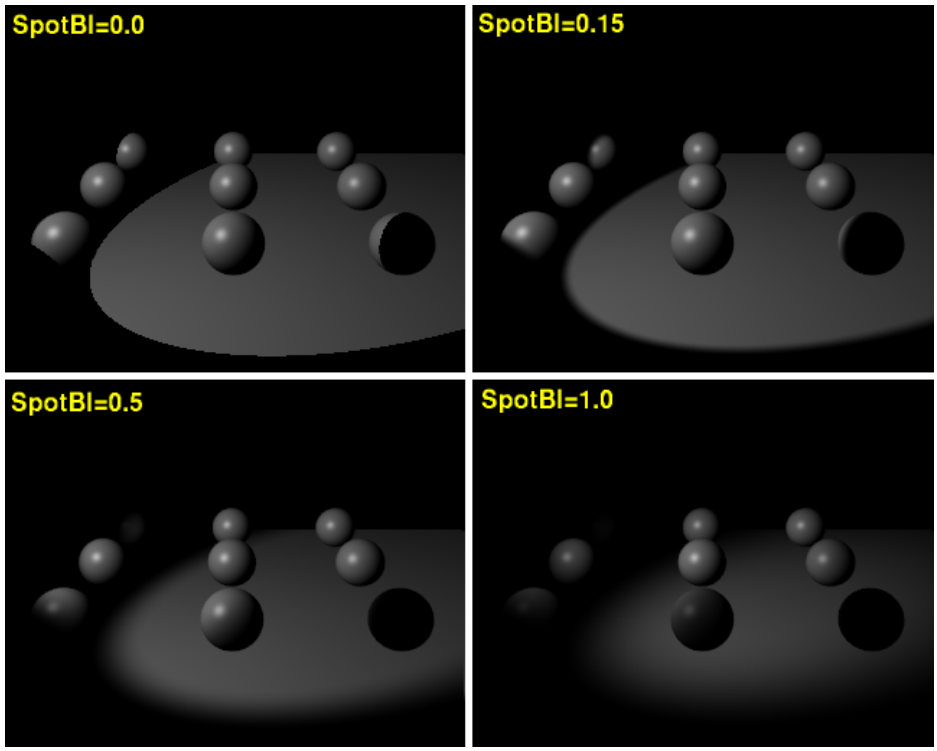


Figure 9-15. Spot Light Examples for SpotSi=45°

Shadows

The lighting schemes analyzed up to now produce on the objects only areas which are more or less lit, but no cast- or self-shadowing, and a scene without proper shadowing looses depth and realism.

On the other hand, proper shadow calculation requires a full - and slow - ray tracer. For a scan liner, as Blender is, shadows can be computed using a *shadow buffer* for shadow casting lights. This implies that an 'image', as seen from the Spot Light itself, is 'rendered' and that the distance for each point from the spotlight saved. Any point of the rendered image further than any of those points is then considered to be in shadow.

The shadow buffer stores this data. To keep the algorithm compact, efficient and fast this shadow buffer has a size which is fixed from the beginning and which in Blender can be from 512x512 to 5120x5120. The higher value is the most accurate.

The user can control the algorithm via the bottom two row of buttons in the Lamp window (Figure 9-16).

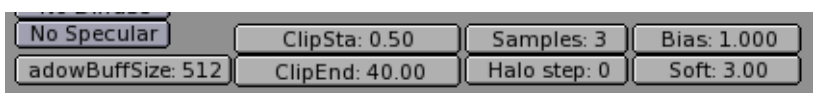


Figure 9-16. Spot Light shadow buttons.

`ShadowBuffSize` - Numeric Button, from 512 to 5120, defining the shadow buffer size.

`ClipSta, ClipEnd` - To further enhance efficiency the shadow computations are actually performed only in a predefined range of distances from the spot position. This range goes from ClipSta, nearer to the Spot light, to ClipEnd, further away (Figure 9-12). All objects nearer than ClipSta from the Spot light are never checked for shadows, and are always lit. Objects further than ClipEnd are never checked for light and are always in shadow. To have a realistic shadow ClipSta must be less than the smallest distance between any relevant object of the scene and the spot, and ClipEnd larger than the largest distance. For the best use of the allocated memory and better shadow quality, ClipSta must be as large as possible and ClipEnd as small as possible. This minimizes the volume where shadows will be computed.

`Samples` - To obtain soft shadows the shadow buffer, once computed, is rendered via its own anti-aliasing algorithm which works by averaging the shadow value over a square of a side of a given number of pixels. Samples is the number of pixels. Its default is 3, that is a 3x3 square. Higher values give better anti-aliasing, and a slower computation time.

`Bias` - Is the bias used in computing the shadows, again the higher the better, and the slower.

`Soft` - Controls the softness of the shadow boundary. The higher the value, the softer and more extended the shadow boundaries will be. Commonly it should be assigned a value which ranges from the same value of the Sample NumButton to double that value.

`Halo step` - The stepping of the halo sampling for volumetric shadows when volumetric light is on. This will be explained in the 'Volumetric light' section.

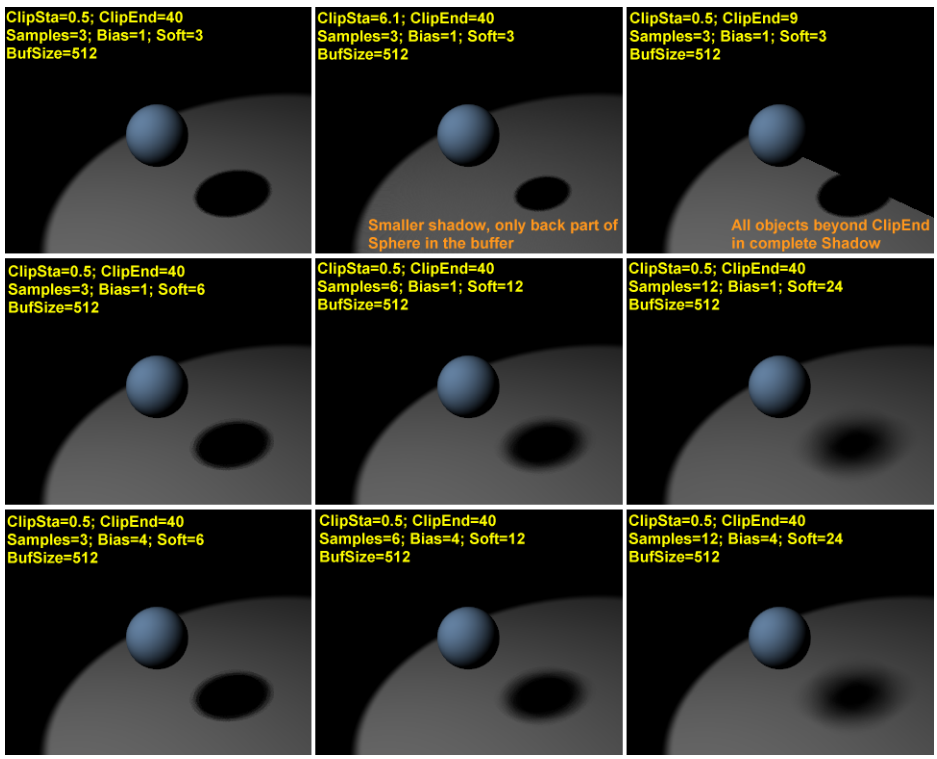


Figure 9-17. Spot Light shadow examples.

Volumetric Light

Volumetric light is the effect you see in a hazy air, when the light rays become visible

because of light scattering which occurs due to mist, fog, dust etc.

If used carefully it can add much realism to a scene... or kill it.

The volumetric light in Blender can only be generated for Spot Lights, once the 'Halo' button (Figure 9-18) is pressed.

Figure 9-18. Spot Light halo button.

If the test set up shown in Figure 9-19 is created, and the Halo button pressed, the rendered view will be like Figure 9-20.

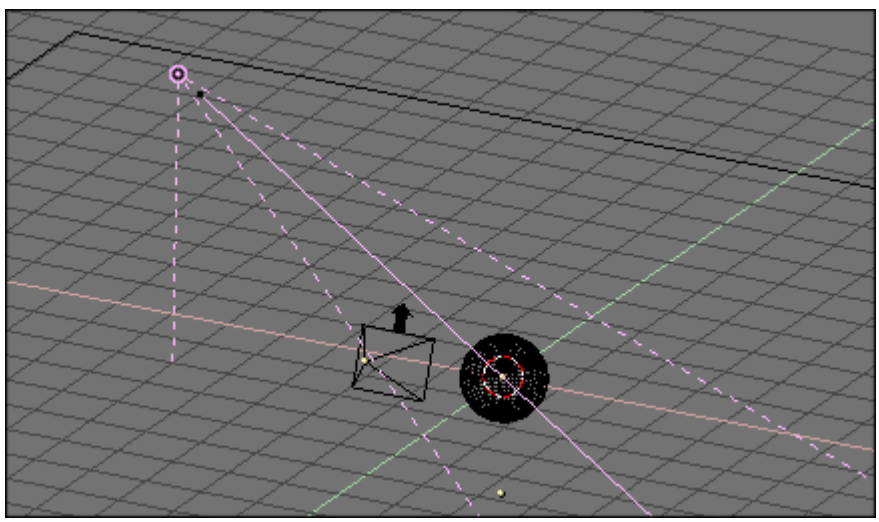


Figure 9-19. Spot Light setup.

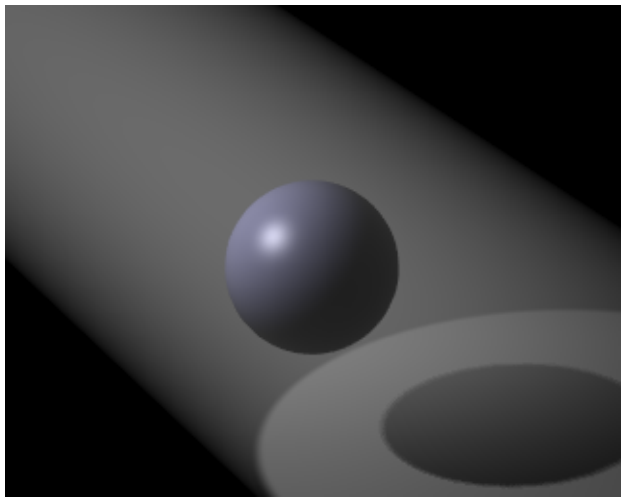


Figure 9-20. Halo rendering.

The volumetric light effect is rather strong. The intensity of the Halo can be regulated with the `HaloInt` slider (Figure 9-21). Lower values corresponding to weaker halos.

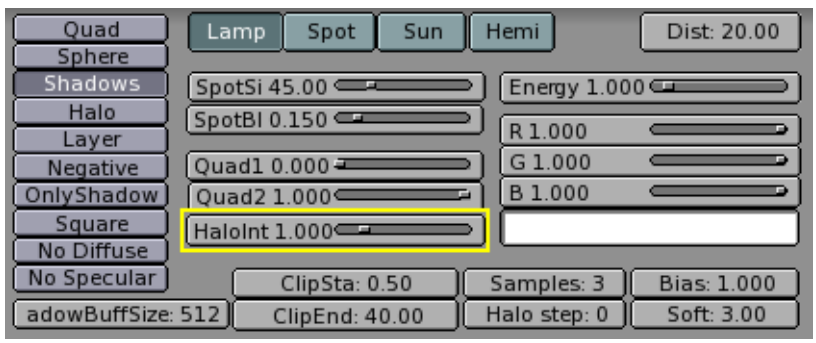


Figure 9-21. Halo Intensity Slider.

The result is interesting. We have volumetric light, but we lack volumetric shadow! The halo passes through the sphere, yet a shadow is cast. This is due to the fact that the Halo occurs in the whole Spot Light cone unless we tell Blender to do otherwise.

The cone needs to be sampled to get volumetric shadow, and the sampling occurs with a step defined by the `HaloStep` NumButton (Figure 9-22). The default value of 0 means no sampling at all, hence the lack of volumetric shadow. A value of 1 gives finer stepping, and hence better results, but with a slower rendering time (Figure 9-23), while a higher value gives worse results with faster rendering (Figure 9-24).

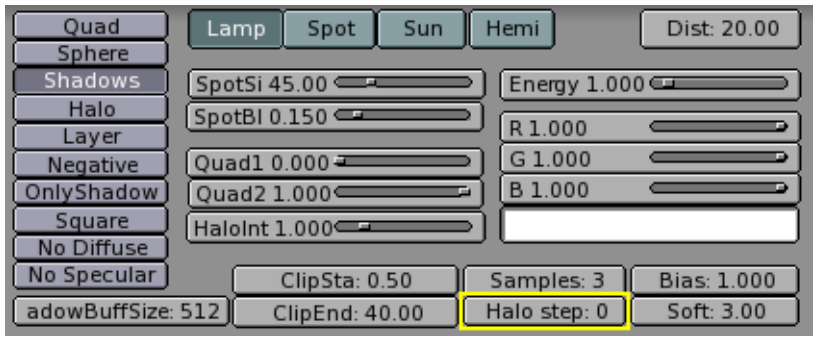


Figure 9-22. Halo Step NumButton.

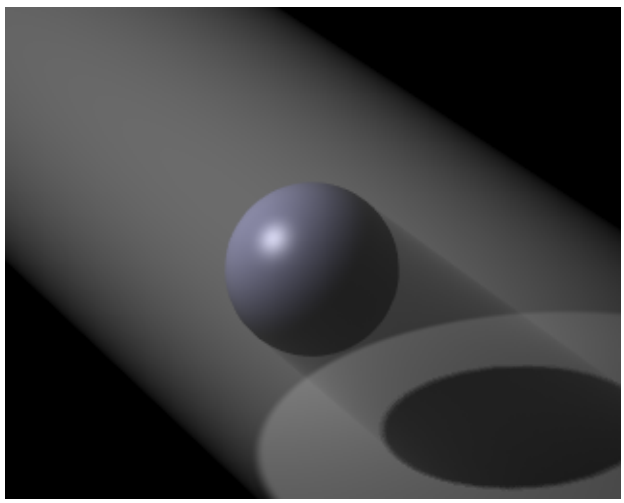


Figure 9-23. Halo with volumetric shadow, Halo Step = 1

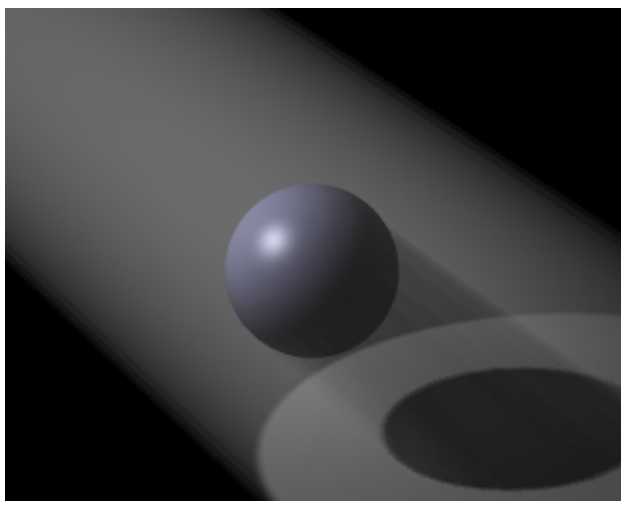


Figure 9-24. Halo with volumetric shadow, Halo Step = 12

HaloStep values: A value of 8 is usually a good compromise between speed and accuracy.

Tweaking Light

Ok, now you've got the basics. Now we can reallytalk of light. We will work on a single example, more complex than a plain 'sphere over a plane' setup, to see what we can achieve in realistic lighting with Blender.

We will resort to the setup in Figure 9-25. The humanoid figure is a gynoid - that is the female counterpart of an android - we will call her Liliana[1] in the following. She has a dull grey material (R=G=B=0.8, Ref=0.8 Spec=0.05, Hard=20 - Yes, a shiny chrome would have suited her better, but we are talking of lights, not of materials!) and stands on a blue plane (R=0.275,G=0.5,B=1.0, Ref=0.8, Spec=0.5, Hard=50). For now she's lit by a single spot (Energy=1.0, R=G=B=1.0, SpotSi=45.0, SpotBl=0.15 ClipSta=0.1,ClipEnd=100,Samples=3,Soft=3, Bias=1.0, BufSize=512).

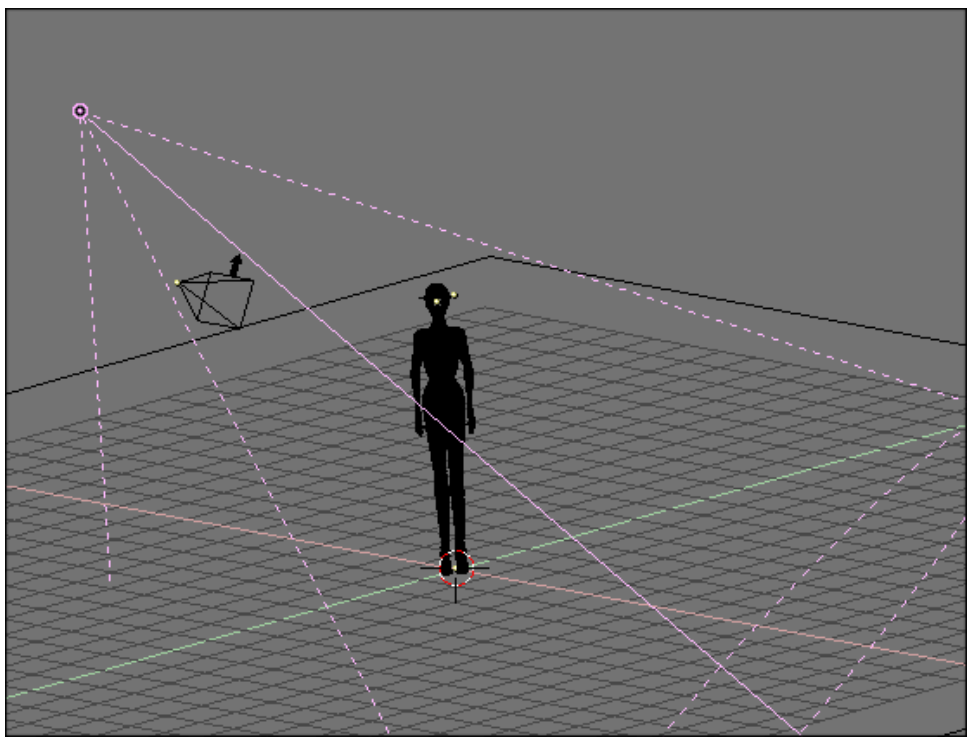


Figure 9-25. Light tweaking setup.

A rendering of Liliana in this setup, with OSA=8 and Shadows enabled gives the result in Figure 9-26. The result is ugly. You have very black, unrealistic shadows on Liliana, and the shadow cast by Liliana herself is unacceptable.

Figure 9-26. Simple Light Spot set up.

The first tweak is on ClipStart and ClipEnd, if they are adjusted so as to encompass the scene as tightly as possible (ClipSta=5, ClipEnd=20) the results get definitely better, at least for projected shadow. Liliana's own shadow is still too dark (Figure 9-27).

Figure 9-27. Single Spot Light set up with appropriate Clipping.

To set good values for the Clipping data here is a useful trick: 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 it by pressing **CTRL-NUM0**. What you would see, in shaded mode, is shown in Figure 9-28.

All stuff nearer to the Spot than ClipSta and further from the spot than ClipEnd is not shown at all. Hence you can fine tune these values by verifying that all shadow casting objects are visible.

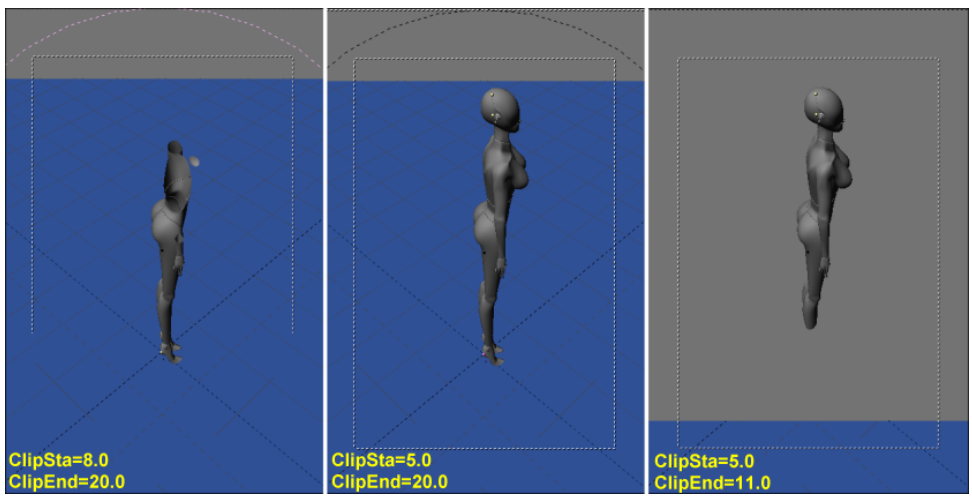


Figure 9-28. Spot Light Clipping tweak. Left: ClipSta too high; Center: Good; Right: ClipEnd too low.

What is still really lacking is the physical phenomenon of diffusion. A lit body sheds light from itself, hence shadows are not completely black because some light leaks in from neighboring lit regions.

This light diffusion is correctly accounted for in a Ray Tracer, and in Blender too, via the Radiosity Engine. But there are setups which can fake this phenomenon in an acceptable fashion.

We will analyze these, from simplest to more complex.

Three point light

The three point light setup is a classical, very simple, scheme to achieve a scene with softer lighting. Our Spot Light is the main, or Key, Light of the scene, the one casting shadow. We will add two more lights to fake diffusion.

The next light needed is called the 'Back Light'. It is placed behind Liliana (Figure 9-29). This illuminates the hidden side of our character, and allows us to separate the foreground of our picture from the background, adding an overall sense of depth. Usually the Back Light is as strong as the Key Light, if not stronger. Here we used an Energy 1 Lamp Light (Figure 9-30).

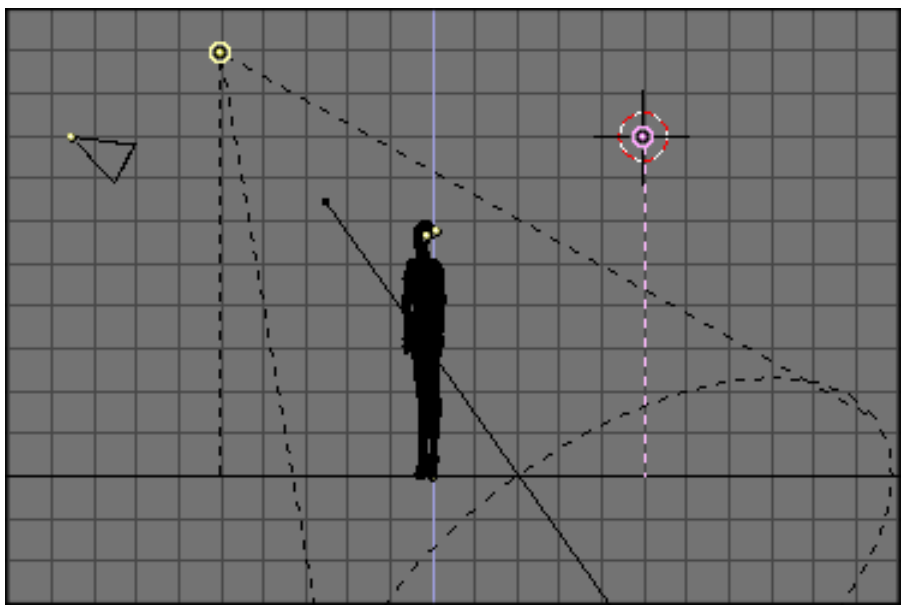


Figure 9-29. Back Light set up.

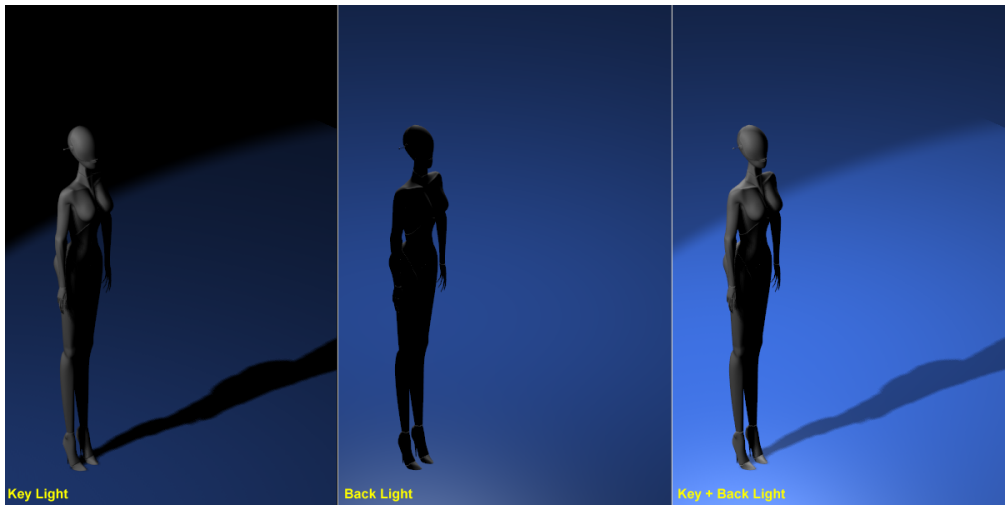


Figure 9-30. Key Light only (left) Back Light only (center) and both (right).

The result is already far better. Finally, the third light is the 'Fill' Light. The Fill light's aim is to light up the shadows on the front of Liliana. We will place the Fill Light exactly at the location of the camera, with an Energy lower than the lower of Key and Back Lights (Figure 9-31). For this example an Energy=0.75 has been chosen (Figure 9-32).

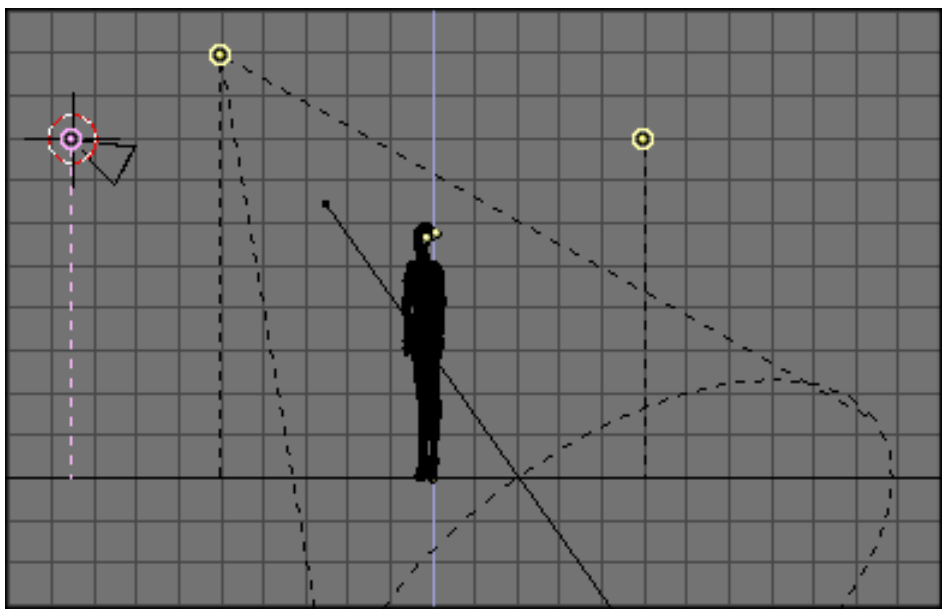


Figure 9-31. Fill Light set up.

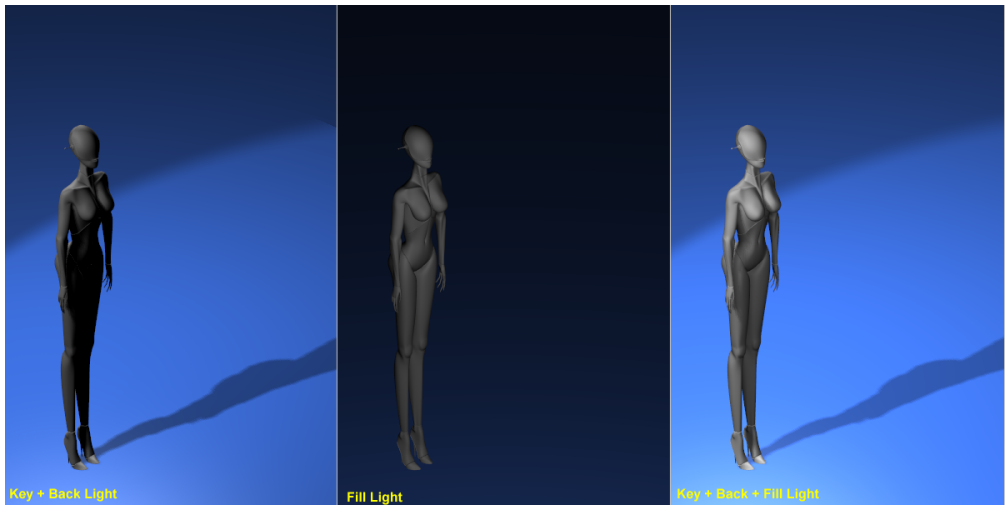


Figure 9-32. Key and Back Light only (left) Fill Light only (center) and all three (right).

The Fill light makes visible parts of the model which are completely in darkness with the Key and Back light only.

Color leakage: The three-point set up can be further enhanced with the addition of a fourth light, especially when a bright colored floor is present, like in this case.

If there is a bright colored floor our eye expects the floor to diffuse part of the light all around, and that some of this light impinges on the model.

To fake this effect we place a second spot exactly specular to the Key Light with respect to the floor. This means that - if the floor is horizontal and a z=0, as it is in our example, and the Key light is in point (x=-5, y=-5, z=10), then the floor diffuse light is to be placed in (x=-5,y=-5,z=-10), pointing upward (Figure 9-33).

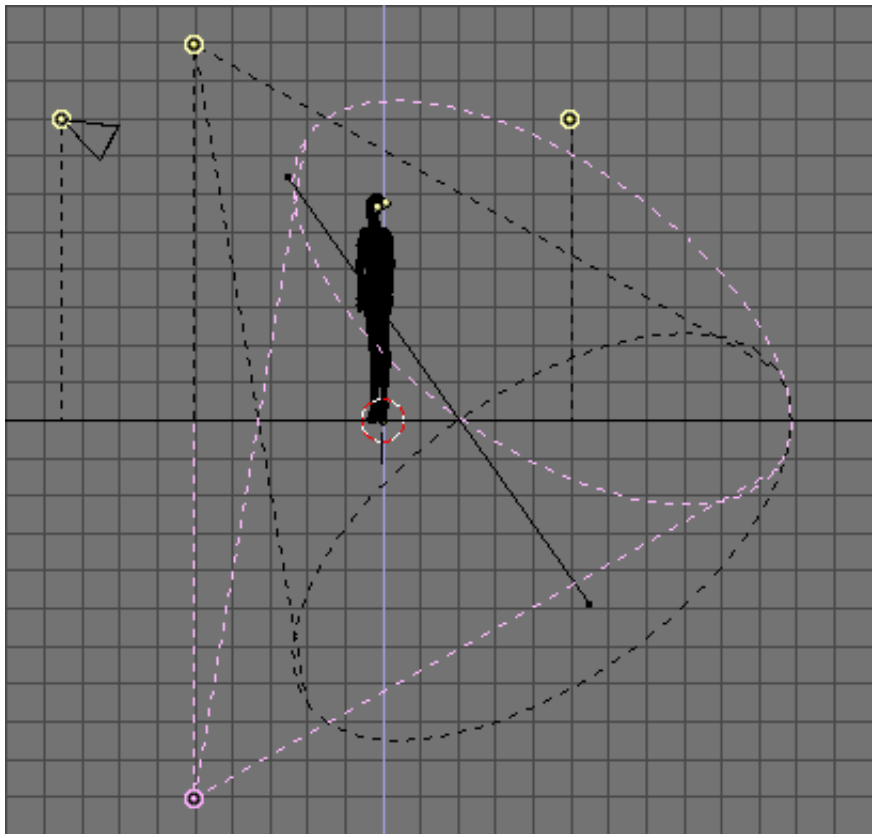


Figure 9-33. Floor Diffuse Light set up.

The energy for this light should be lower than that of the Key Light (here it is 0.8) and its color should match the color of the floor (here R=0.25, G=0.5, B=1.0). The result is shown in Figure 9-34

Figure 9-34. Four Light set up.

Please note that we used a Spot light and not a lamp, so it would be completely blocked by the floor (shadowed) unless we set this spot shadeless by pressing the appropriate button.

Indeed we could have used a Lamp but if the floor is shiny the light it sheds is more reflected than diffused. Reflected light, physically is itself a cone coming from the specular source.

Three point light - Outdoor

By using a Spot light as a key light the previous method is sadly bound to indoor settings or, at maximum, outdoor settings at night time. This is because the Key light is at a finite distance, its rays spread and the floor is not evenly illuminated.

If we were outdoor on a clear sunny day all the floor would be evenly lit, and shadows would be cast.

To have a uniform illumination all over the floor a Sun light is good. And if we add a Hemi light for faking the light coming from all points of the sky (as in Figure 9-8) we can achieve a nice outdoor light... but we have no shadows!

The setup of the Key light (the sun, R=1.0, G=0.95, B=0.9, Energy=1.0) and the Fill/Back Light (the Hemi, R=0.8, G=0.9,B=1.0, Energy=0.6) is shown in Figure 9-35 and the relevant rendering in Figure 9-36

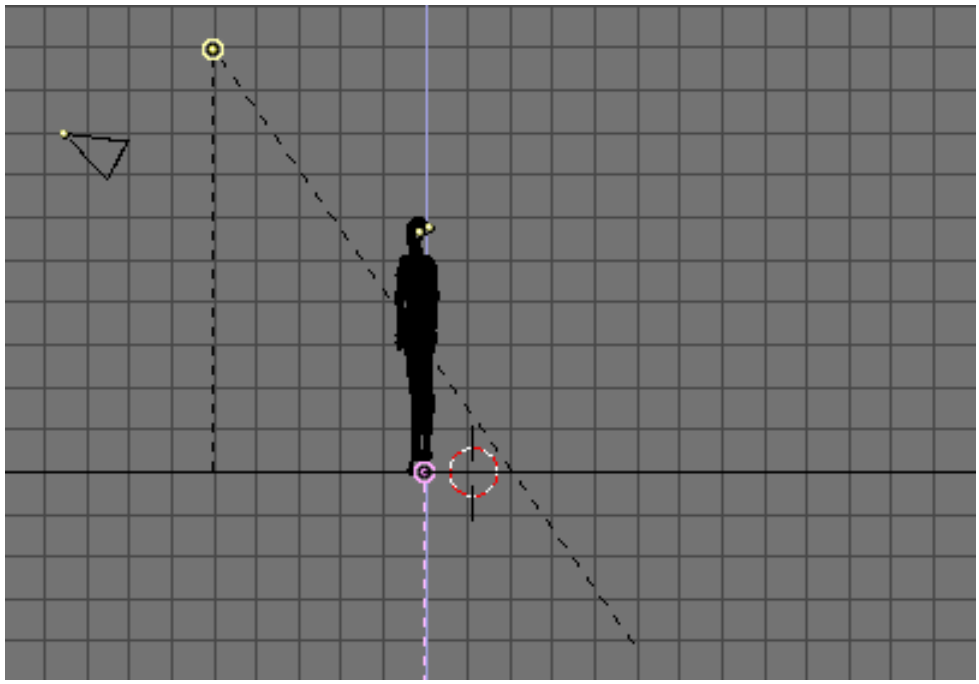


Figure 9-35. Sun and Hemi light for outdoor set up.

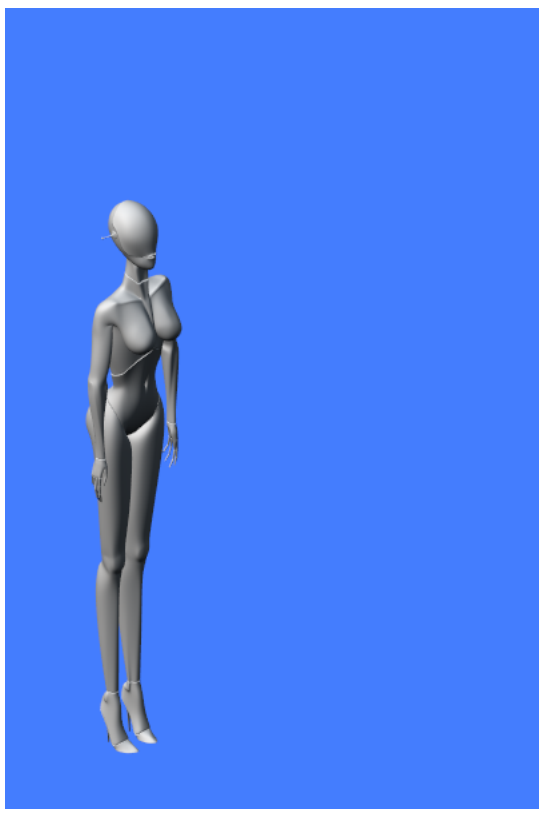


Figure 9-36. Sun and Hemi light for outdoor rendering.

The lack of shadow makes Liliana appear as if she were floating in space. To have

a shadow let's place a Spot coincident with the sun and with the same direction. Let's make this spot a Shadow Only Spot (with the appropriate button). If Energy is lowered to 0.9 and all other settings are kept at the values used in the previous example (BufSize=512, Samples=3 Soft=3 Bias=1 ClipSta=5, ClipEnd=20) the result is the one of Figure 9-37 (center).

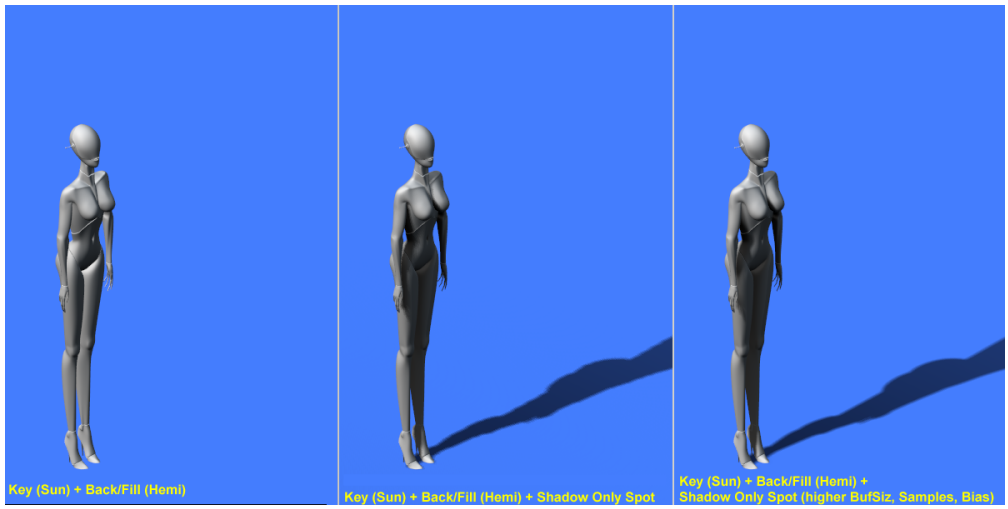


Figure 9-37. Outdoor rendering.

The shadow is a bit blocky because Liliana has many fine details and the BufSize is too small, and the Sample value is to low to correctly take them into account. If BufSize is raised to 2560, Samples to 6 and Bias to 3.0 the result is the one in Figure 9-37 (right). Much smoother.

Area Light

The concept of Light coming from a point is an approximation. No real world light source is dimensionless. All light is shed by surfaces, not by points.

This has a couple of interesting implications, mainly on shadows:

- Sharp shadows does not exist: shadows have blurry edges
- Shadow edge blurriness depends on the relative positions and sizes of the light, the shadow casting object and the object receiving the shadow.

The first of this issues is approximated with the 'Soft' setting of the Spot light, but the second is not. To have a clearer understanding of this point imagine a tall thin pole in the middle of a flat plain illuminated by the Sun.

The Sun is not a point, it has a dimension and, for us earthlings, it is half of a degree wide. If you look at the shadow you will notice that it is very sharp at the base of the pole and that it grows blurrier as you go toward the shadow of the tip. If the pole is tall and thin enough its shadow will vanish.

To better grasp this concept have a look at Figure 9-38. The sun shed light, the middle object obstruct sun rays completely only in the dark blue region. For a point in the light blue region the Sun is partially visible, hence each of those point is partially lit.

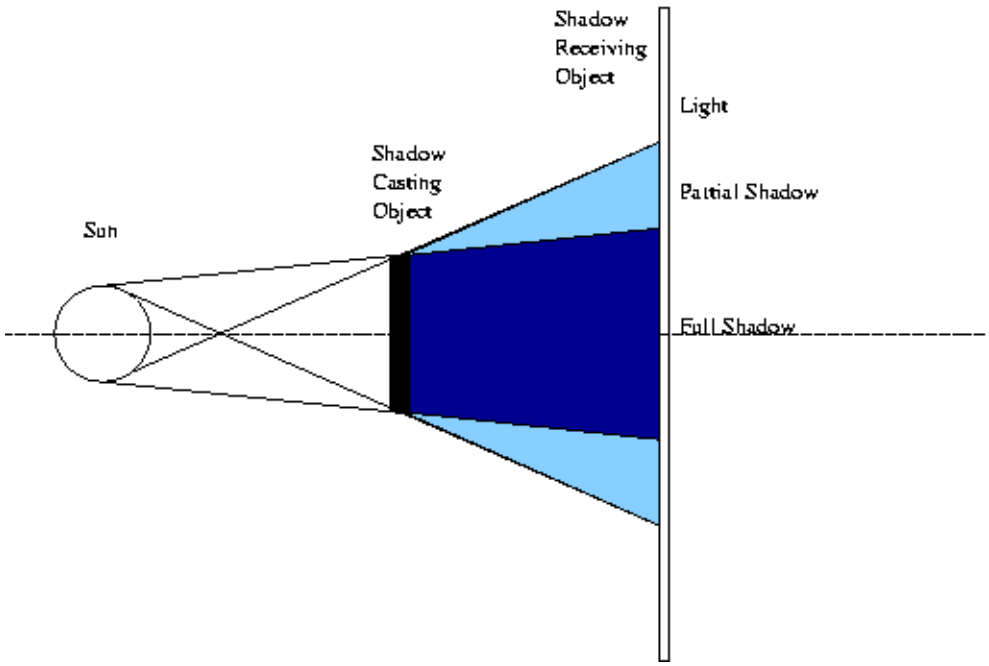


Figure 9-38. Area light and its shadow.

The light blue region is a partial shadow region where illumination drops smoothly from full light to full shadow. It is also evident, from Figure 9-38 than that this transition region is smaller next to the shadow casting object and grow larger far away from it. Furthermore, if the shadow casting object is smaller than the light casting object (and if the light casting object is the sun this is often the case) there is a distance beyond which only partial shadow remains Figure 9-39.

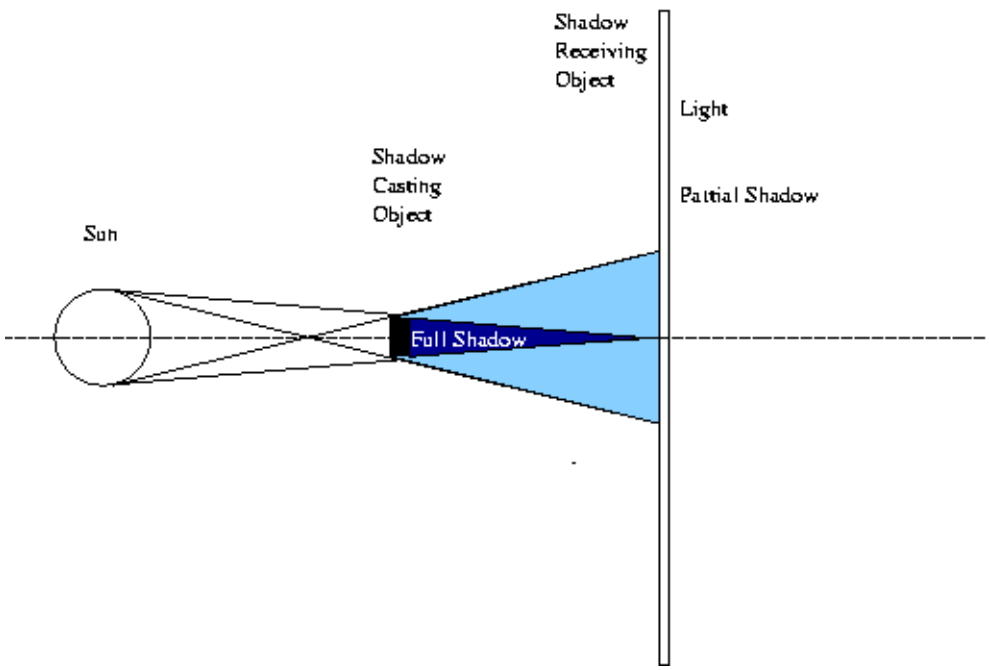


Figure 9-39. Area light and its shadow 2

In Blender, if we place a single Spot at a fixed distance from a first plane and look at

the shadow cast at a second plane as this second plane gets further away we notice that the shadow gets larger but not softer (Figure 9-40)

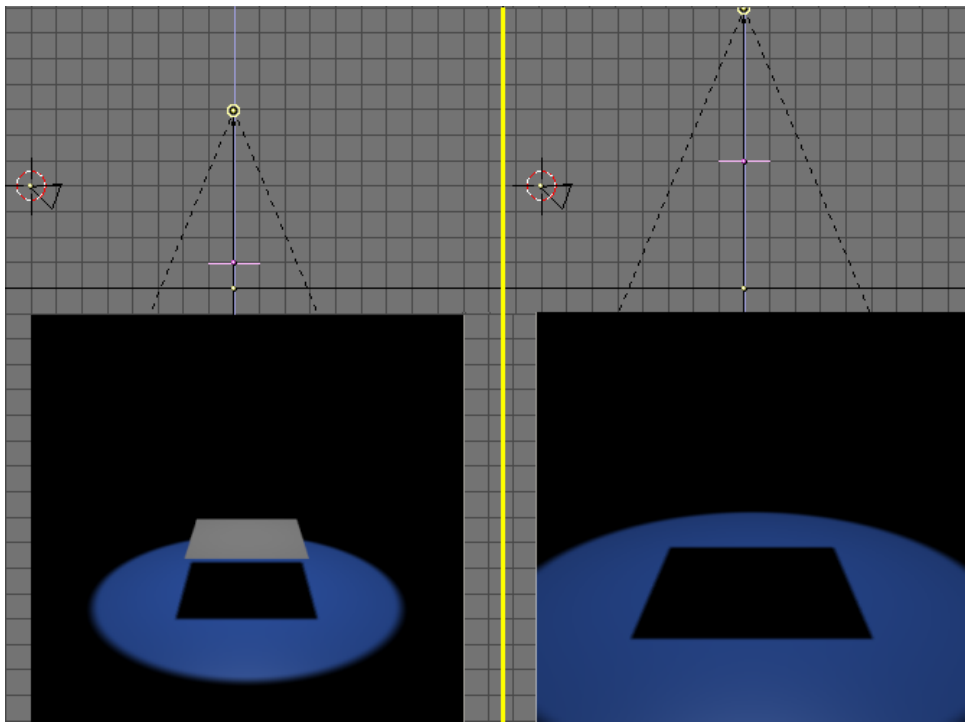


Figure 9-40. Spot light and its shadow

To fake an area light with Blender we can use several Spots, as if we were sampling the area casting light with a discrete number of point lights.

This can either be achieved by placing several Spots by hand, or by using Blender's DupliVert feature (the Section called *Dupliverts* in Chapter 17), which is more efficient.

Add a Mesh Grid 4x4. Where the spot is, be sure normals are pointing down, by letting Blender show the Normals and eventually flipping them, as explained in the Section called *Basic* in Chapter 6 (Figure 9-41). Parent the Spot to the Grid, select the Grid and in the Anim buttons (**F7**) press DupliVert and Rot. Rot is not strictly necessary but will help you in positioning the Area Light later on. You will have a set of Spots as in Figure 9-42.

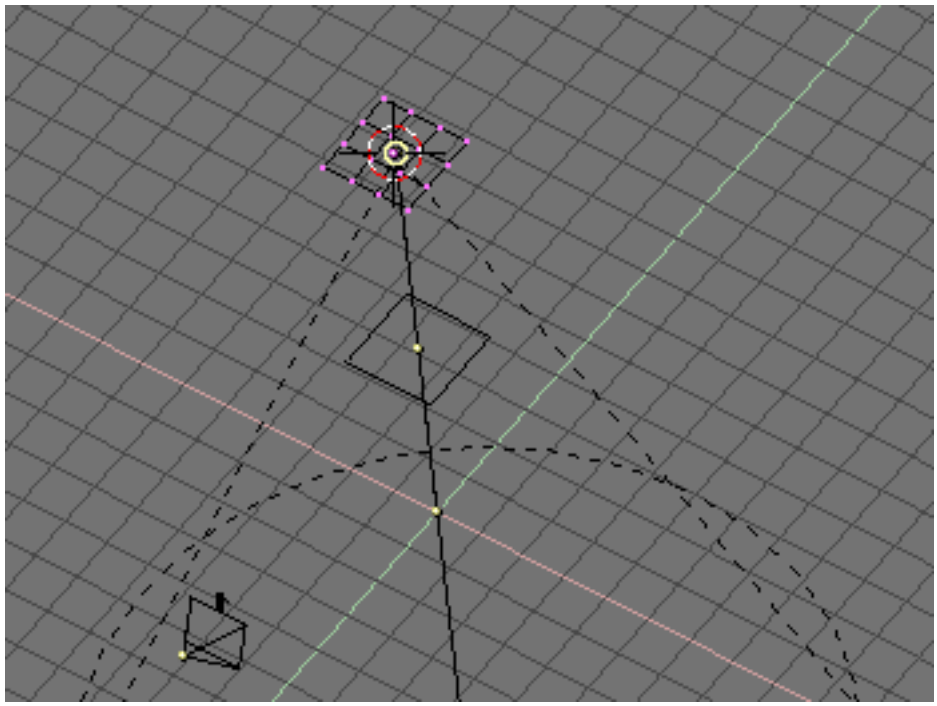


Figure 9-41. Grid setup

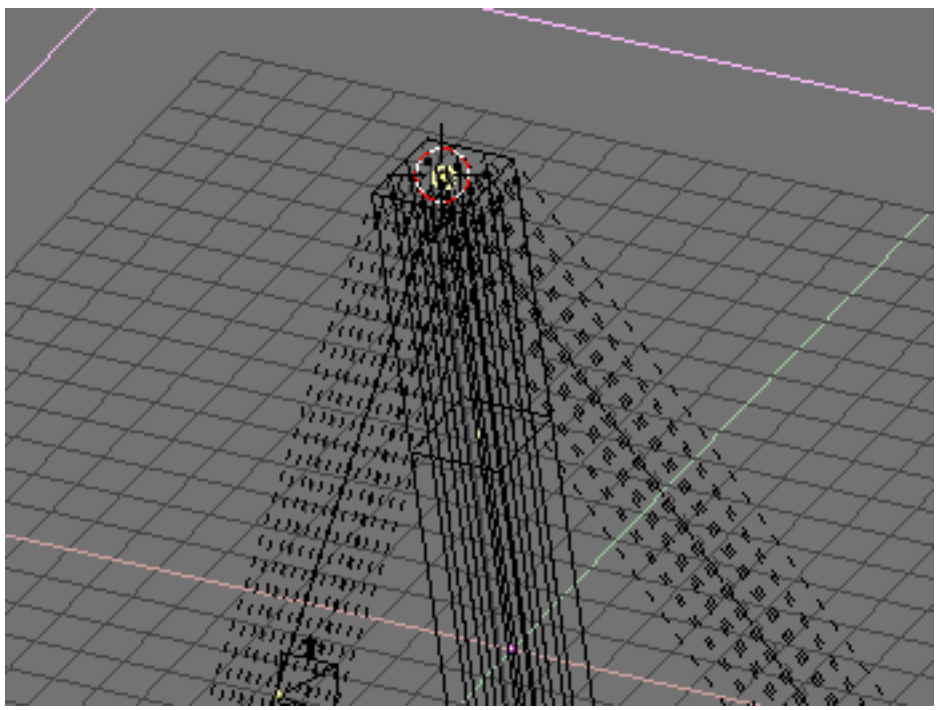


Figure 9-42. Spot light and its dupliverts

Then decrease the Energy of the Spot. If for a single Spot you used a certain energy, now you must subdivide that energy among all the duplicates. Here we have 16 spots, so each should be allotted 1/16 of Energy (that is Energy=0.0625).

The same two renderings of above, with this new hacked area light will yield to

the results in Figure 9-43. The result is far from the expected, because the Spot light sampling of the Area light is too coarse. On the other hand a finer sampling would yield to higher number of duplicated Spots and to unacceptable rendering times.

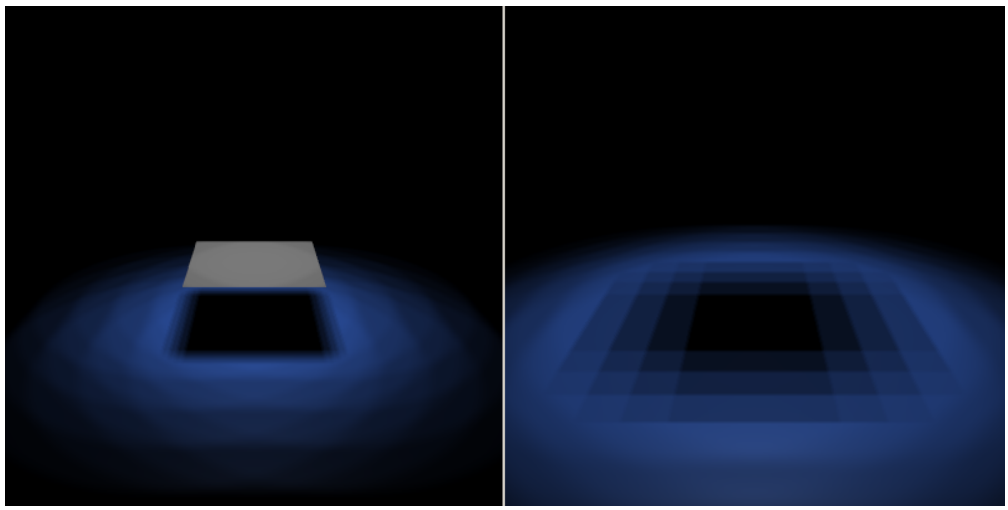


Figure 9-43. Fake area light with multiple spots.

A much better result can be attained by softening the spots, that is setting SpotBl=0.45, Sample=12, Soft=24 and Bias=1.5 (Figure 9-44)

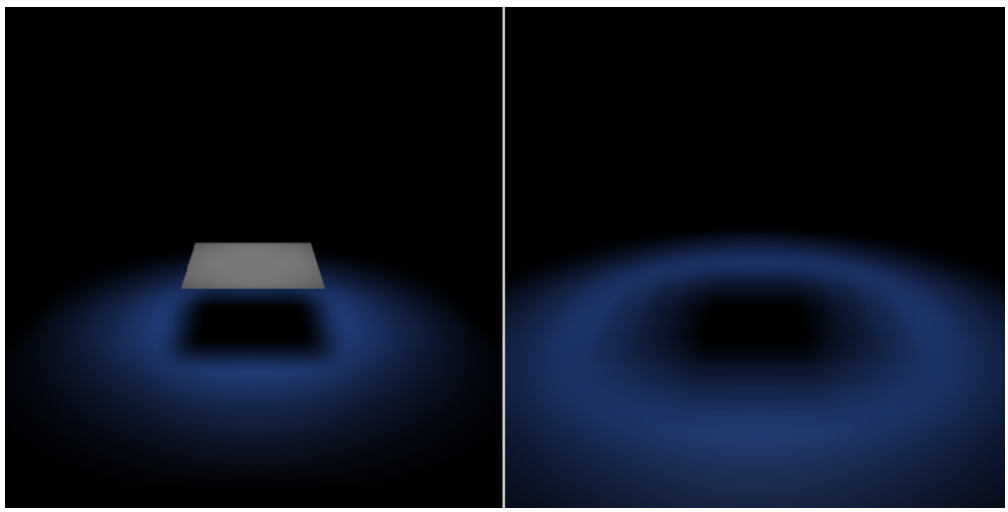


Figure 9-44. Fake area light with multiple soft spots.

Finally, Figure 9-45 shows what happens to Liliana once the Key Light is substituted with 65 duplicated Spots of Energy=0.0154 in a circular pattern. Please note how the shadow softly goes from sharp next to the feet to softer and softer as it gets further away from her.

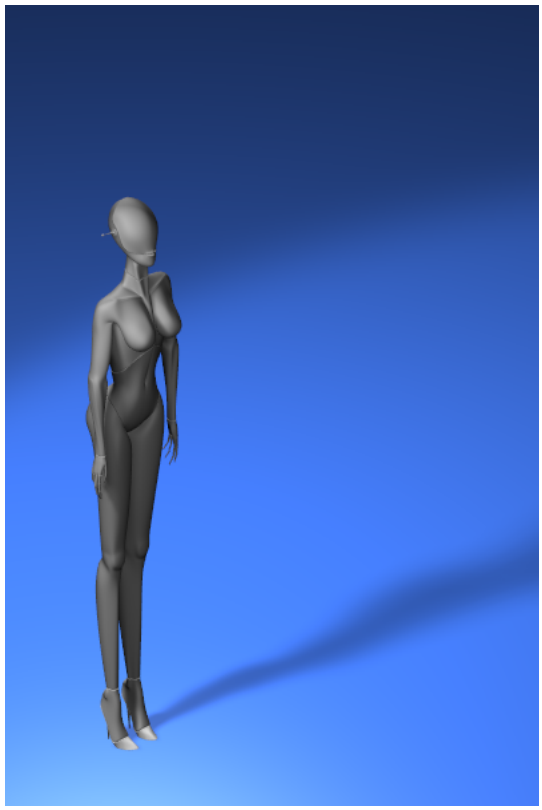


Figure 9-45. Liliana under Area Light.

Global Illumination (and Global Shadowing)

The above techniques work well when there is a single, or anyway finite number of lights, casting distinct shadows.

The only exceptions are the outdoor setting, where the Hemi Light fakes the light cast by the sky, and the Area Light, where multiple spots fakes a light source of finite extension.

The first of these two is very close to nice outdoor lighting, were it not for the fact that the Hemi Light casts no shadows and hence you don't have realistic results.

To obtain a really nice outdoor setting, especially for cloudy daylight, you need light coming from all directions of the sky, yet casting shadows!

This can be obtained by applying a technique very similar to the one used for the Area Light setup, but using half a sphere as a parent mesh. This is usually referred to as "Global Illumination".

You can either use a UVsphere or an IcoSphere, the latter has vertices evenly distributed whereas the former has a great concentration of vertices at poles. Using an IcoSphere hence yields a more 'uniform' illumination, all the points of the sky radiating an equal intensity; a UVsphere casts much more light from the pole(s). Personally I recommend the IcoSphere.

Let's prepare a setup, comprising a plane and some solids, as in Figure 9-46. We will use simple shapes to better appreciate the results.

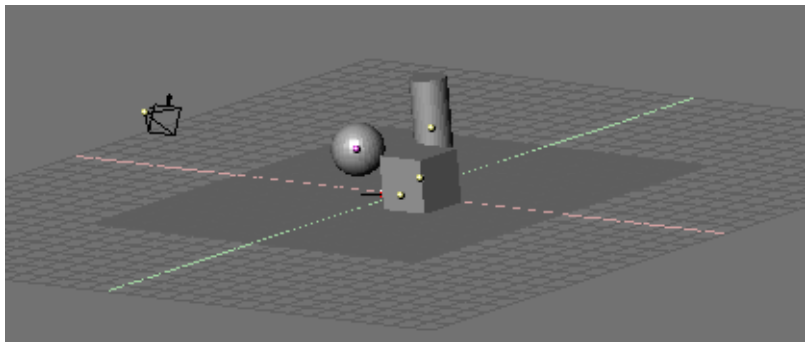


Figure 9-46. Global Illumination scene.

Switch to top view and add an IcoSphere, a subdivision level 2 IcoSphere is usually enough, a level 3 one yields even smoother results. Scale the IcoSphere so that it completely, and loosely, contains the whole scene. Switch to front view and, in EditMode, delete the lower half of the IcoSphere (Figure 9-47). This will be our "Sky Dome" where the spots will be parented and dupliverted.

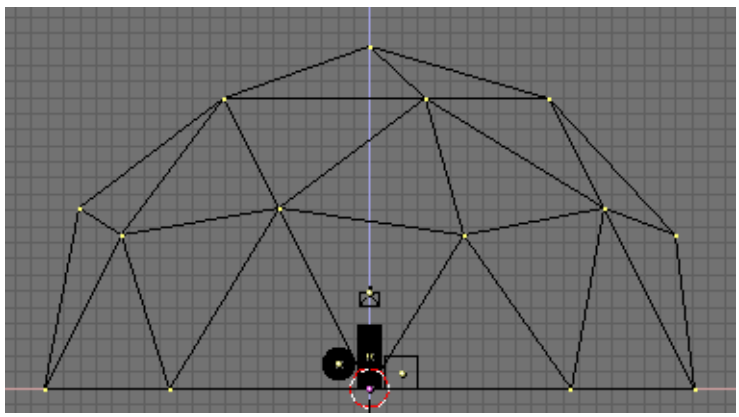


Figure 9-47. Sky dome.

Again in TopView add a Spot Light, parent it to the half IcoSphere and press the DupliVert and Rot buttons exactly as in the previous example. The result, in FrontView, is the one in Figure 9-48.

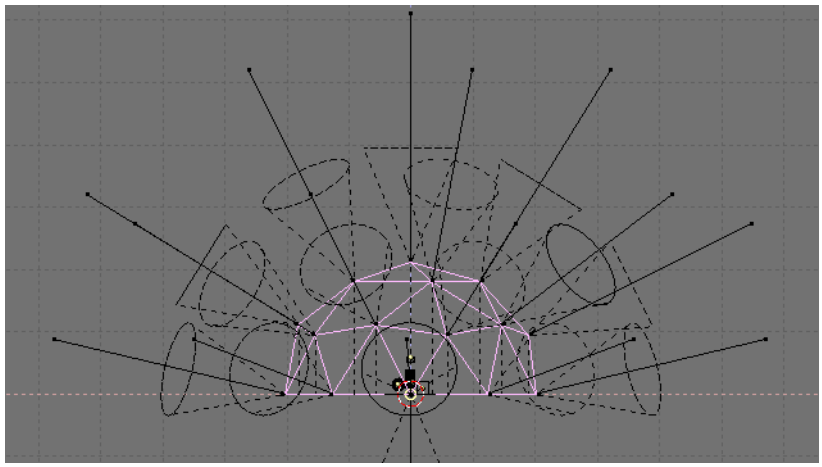


Figure 9-48. Sky dome with duplicated spots.

This is not what we want, since all spots point outwards and the scene is not lit. This is due to the fact that the IcoSphere normals point outward. It is possible to invert their directions by selecting all vertices in EditMode and by pressing the "Flip Normals" button in the Mesh Editing buttons (Figure 9-49).

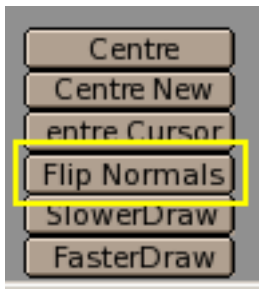


Figure 9-49. Flipping normals.

This leads to the new configuration in Figure 9-50.

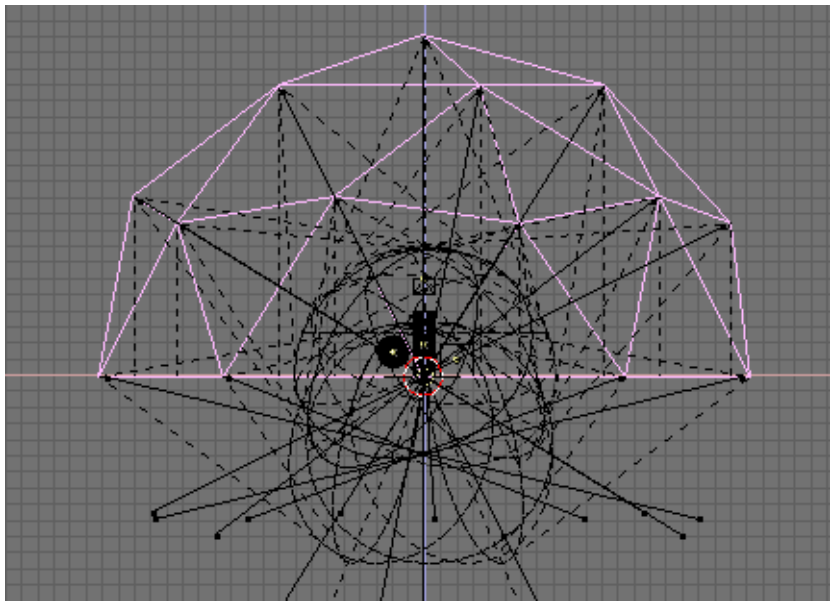


Figure 9-50. Correct sky dome and dupliverted Spot Lights.

To obtain good results select the original Spot Light and change its parameters to a wide angle with soft boundaries (SpotSi=70.0; SpotBl=0.5); with suitable ClipStart and ClipEnd values; in this case 5 and 30, respectively, in any case appropriate values to encompass the whole scene; increase samples to 6 and softness to 12. Decrease energy to 0.1; remember you are using many spots, so each must be weak. (Figure 9-51).

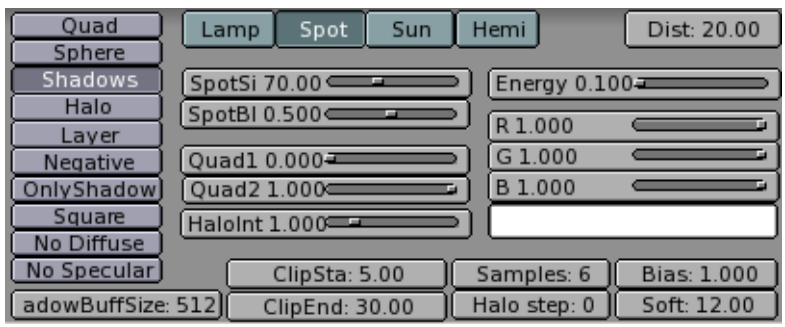


Figure 9-51. Spot Light setup.

Now you can make the rendering. If some materials are given and a world set, the result should be that of Figure 9-52. Note the soft shadows and the 'omnidirectional' lighting. Even better results can be achieved with a level 3 IcoSphere.

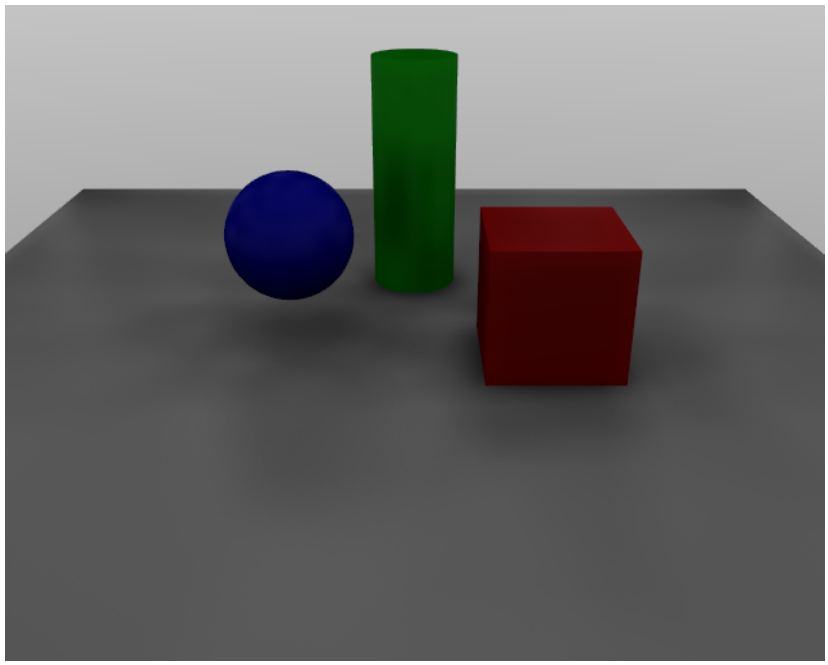


Figure 9-52. Spot Light setup.

This Global Illumination technique effectively substitutes, at a very high computational cost, the Hemi for the above outdoor setting.

It is possible to add a directional light component by faking the Sun either via a single spot or with an Area Light.

An alternative possibility is to make the IcoSphere 'less uniform' by subdividing one of its faces a number of times, as is done for one of the rear faces in Figure 9-53. This is done by selecting one face and pressing the "Subdivide" button, then de-selecting all, re-selecting the single inner small face and subdividing it, and so on.

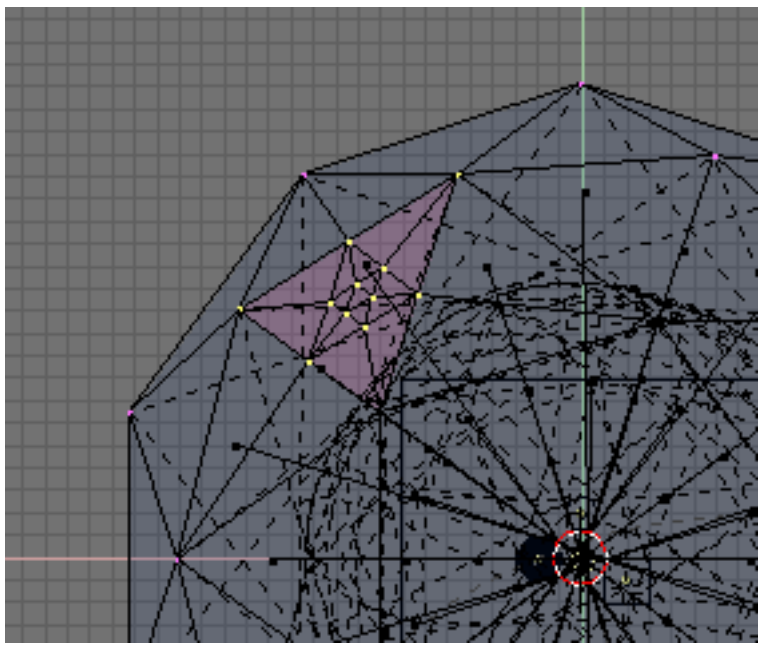


Figure 9-53. Making spots denser in an area.

The result is a very soft directional light together with a global illumination sky dome or, briefly, an anisotropic skydome (Figure 9-54). This is quite good for cloudy conditions, but not so good for clear sunny days. For really clear days, it is better to keep the sky dome separate from the Sun light, so as to be able to use different colours for each.

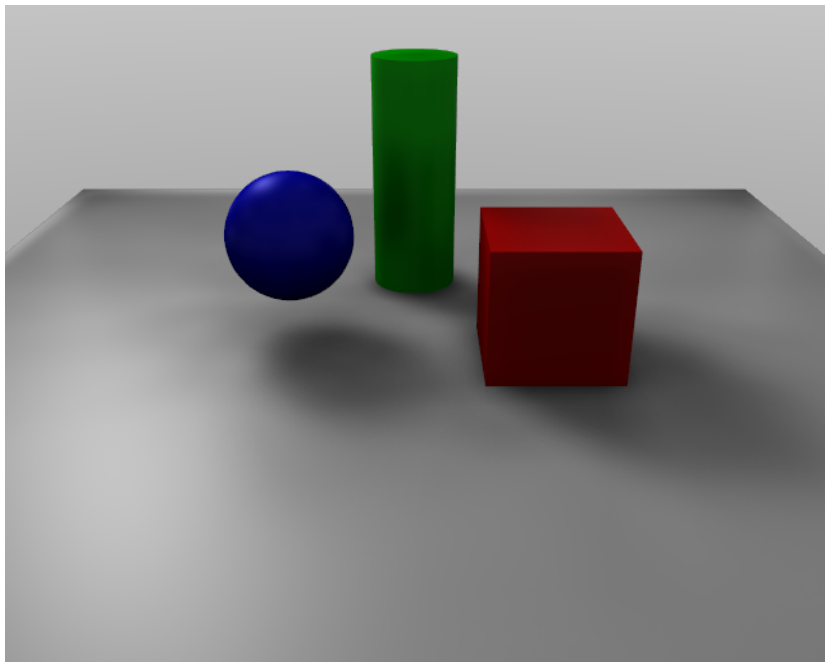


Figure 9-54. Anisotropic skydome render.

Notes

1. Liliana is my grandmother...

Chapter 10. The World and The Universe

Blender provides a number of very interesting settings to complete your renderings with a nice background, and some interesting 'depth' effects. These are accessible via this icon  which brings up the World Buttons shown in Figure 10-1.

Figure 10-1. World Buttons

The World Background

The simplest use of the World Buttons is to provide a nice gradient background for images.

The background buttons (Figure 10-2) allow you to define the colour at the horizon (`HoR, HoG, HoB` buttons) and the colour at the zenith (`ZeR, ZeG, ZeB` buttons).

Figure 10-2. Background colours

These colours are then interpreted differently, according to which Buttons at the top of Figure 10-2 are selected:

- `Blend` - The background colour is blended from Horizon to Zenith. If only this button is pressed, then the gradient occurs from bottom to top of the rendered image regardless of camera orientation.
- `Real` - If this button is also pressed the blending is dependent on camera orientation. The Horizon colour is there exactly at the horizon, that is on the x-y plane, and the zenith colour is used for points vertically above and below the camera.
- `Paper` - If this button is pressed the gradient occurs on Zenith-Horizon-Zenith colours, hence there are two transitions, not one, on the image taking into account camera rotation but keeping the horizon colour to the centre and the zenith colour to the extremes.

The World Buttons also provide some texture buttons. Their basic usage is analogous to Materials textures, except for a couple of differences (Figure 10-3):

- There are only 6 texture channels.

- `Texture mapping` - Only has the `Object` and `View` options, View being the default orientation.
- `Affect` - Texture just affects colour, but in four different ways: it can affect the `Blend` channel, making the Horizon colour appear where the texture is non-zero; it can affect the colour of the `Horizon`, or the colour of the Zenith, up or down (`Zen Up`, `Zen Down`)

Figure 10-3. Texture buttons

Mist

Mist is something which can greatly enhance the illusion of depth in your rendering.

Basically Blender mixes the background colour with the object colour and enhances the strength of the former, the further the object is away from the camera. Mist settings are shown in Figure 10-4.

Figure 10-4. Mist buttons

The `Mist` Button toggles mist on and off, the row of three TogButtons below states the decaying rate of the mist, `Quadratic`, `linear` and `Square Root`. These controls the law which governs the 'strength' of the mist as you get further away from the camera.

The Mist is computed starting from a distance from the camera defined by the `Sta:` button and is computed over a length defined by the `Di:` button. Objects further away from the camera than Sta+Di are completely hidden by the mist.

By default the mist uniformly covers all of the image. For a more 'realistic' effect you might want to have the mist decrease with height (altitude, or z). This is governed by the `Hi:` NumButton. If it is non-zero it states, in Blender units, an interval, around z=0 in which the mist goes from maximum intensity (below) to zero (above).

Finally, the `misi:` NumButton defines Mist intensity, or strength.

Figure 10-5 shows a possible test set up.

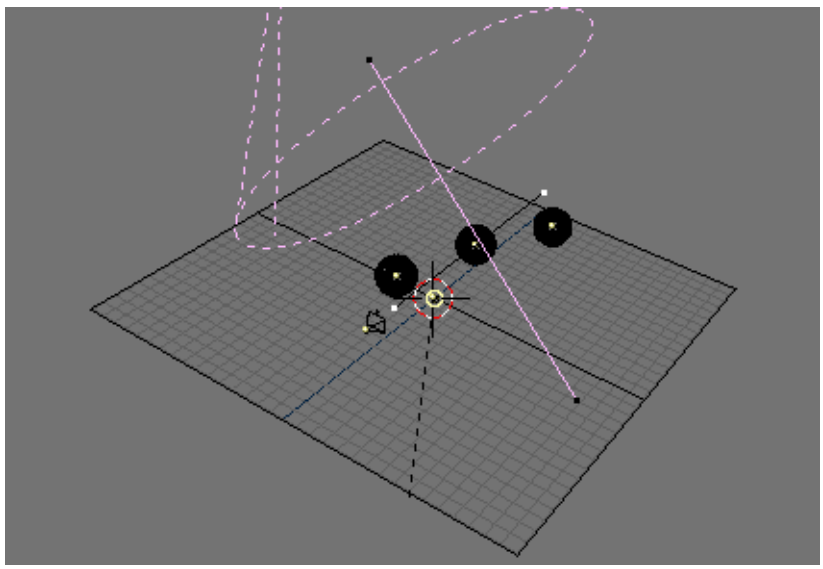


Figure 10-5. Mist test setup

Figure 10-6 shows the results with and without mist. The settings are shown in Figure 10-7; the texture is a plain procedural cloud texture with 'Hard' noise.

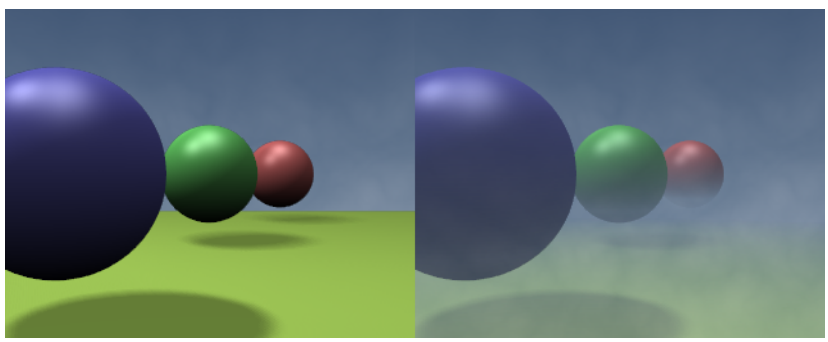


Figure 10-6. Rendering without mist (left) and with mist (right).

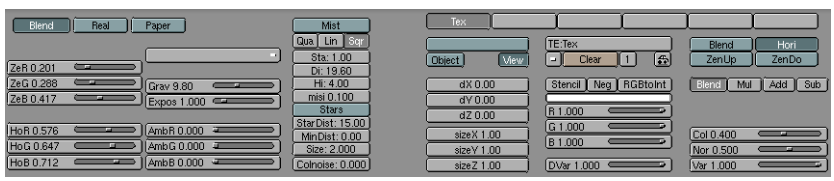


Figure 10-7. World set up.

Mist distances: To see what the mist will actually affect, select your camera, go to EditButtons (**F9**) and hit the `Show Mist` TogButton. The camera will show mist limits as a segment projecting from the camera starting from 'Sta' and of distance 'Di'.

Stars

Stars are randomly placed halo-like objects which appear in the background. Star settings are shown in Figure 10-8.

Figure 10-8. Star buttons

It is very important to understand a few important concepts: `StarDist:` is the *average* distance between stars. Stars are intrinsically a 3D feature, they are placed in space, not on the image!

`Min Dist:` Is the *minimum* distance from the camera at which stars are placed. This should be greater than the distance from the camera to the *furthest* object in your scene, unless you want to risk having stars *in front* of your objects.

The `Size:` NumButton defines the actual size of the star halo. It is better to keep it much smaller than the proposed default, to keep the material smaller than pixel-size and have pin-point stars. Much more realistic.

The `Colnoise:` NumButton adds a random hue to the otherwise plain white stars. It is usually a good idea to add a little ColNoise.

Figure 10-9 Shows the same misty image of Figure 10-7 but with stars added. The Stars settings are shown in Figure 10-10.

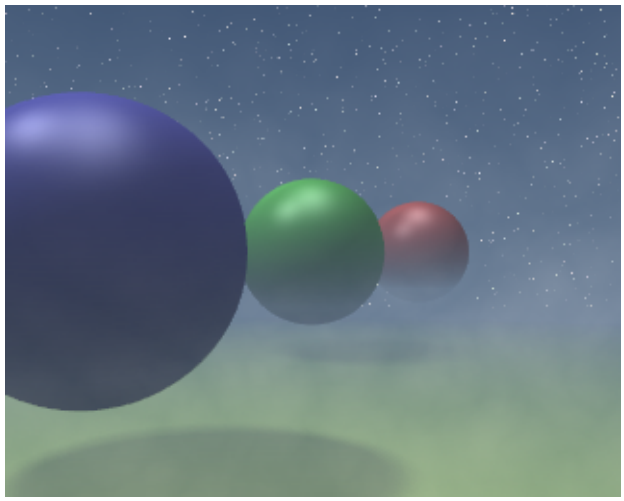


Figure 10-9. Star rendering.

Figure 10-10. Star settings.

Ambient Light

The World Buttons also contain the sliders to define the Ambient light. The effect of Ambient light is to provide a very simple alternative to Global Illumination, inasmuch as it lights up shadows.

Using Ambient light together with other light sources can give convincing results in a fraction of the time required by true GI techniques.

The Ambient light sliders are shown in Figure 10-11.

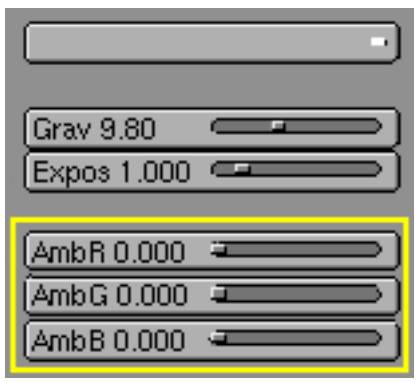


Figure 10-11. Ambient light settings.

Chapter 11. Animation of Undeformed Objects

Objects can be animated in many ways. They can be animated as Objects, changing their position, orientation or size in time; they can be animated by deforming them; that is animating their vertices or control points; or they can be animated via very complex and flexible interaction with a special kind of object: the Armature.

In this chapter we will cover the first case, but the basics given here are actually vital for understanding the following chapters as well.

Three methods are normally used in animation software to make a 3D object move:

- *Key frames* Complete positions are saved for units of time (frames). An animation is created by interpolating an object fluidly through the frames. The advantage of this method is that it allows you to work with clearly visualized units. The animator can work from one position to the next and can change previously created positions, or move them in time.
- *Motion Curves* Curves can be drawn for each XYZ component for location, rotation, and size. These form the graphs for the movement, with time set out horizontally and the value set out vertically. The advantage of this method is that it gives you precise control over the results of the movement.
- *Path* A curve is drawn in 3D space, and the Object is constrained to follow it according to a given time function of the position along the path.

The first two systems in Blender are completely integrated in a single one, the *IPO* (InterPOlation) system. Fundamentally, the IPO system consists of standard motion curves. A simple press of a button changes the IPO to a key system, without conversion, and with no change to the results. The user can work any way he chooses to with the keys, switching to motion curves and back again, in whatever way produces the best result or satisfies the user's preferences.

The IPO system also has relevant implication in Path animations.

IPO Block

The IPO block in Blender is universal. It makes no difference whether an object's movement is controlled or the material settings. Once you have learned to work with object IPOs, how you work with other IPOs will become obvious. Blender does distinguish between different *types* of IPOs. Blender concerns itself only with the type of blocks on which IPOs can work. The interface keeps track of it automatically.

Every type of IPO block has a fixed number of available *channels*. These each have a name (`LocX`, `SizeZ`, etc.) that indicates how they are applied. When you add an IPOCurve to a channel, animation begins immediately. At your discretion (and there are separate channels for this), a curve can be linked directly to a value (`LocX`...), or it can affect a variance of it (`dLocX`...). The latter enables you to move an object as per usual, with the Grabber, without disrupting the IPO. The actual location is then determined by IPOCurves *relative* to that location.

The Blender interface offers many options for copying IPOs, linking IPOs to more than one object (one IPO can animate multiple objects.), or deleting IPO links. The IPOWindow Reference section gives a detailed description of this. This chapter is restricted to the main options for application.

Key Frames

Figure 11-1. Insert Key Menu.

The simplest method for creating an object IPO is with the "Insert key" (**IKEY**) command in the 3DWindow. A Pop-up menu provides a wide selection of options (Figure 11-1). We will select the topmost option: `Loc`. Now the current location X-Y-Z, is saved and everything takes place automatically:

- If there is no IPO block, a new one is created and linked to the object.
- If there are no IPOCurves in the channels `LocX`, `LocY` and `LocZ`, these are created.
- Vertices are then added in the IPOCurves with the exact values of the object location.

We go 30 frames further on (3 x **UPARROW**) and move the object. Again we use **IKEY** and immediately press **ENTER**. The new position is inserted in the IPOCurves. We can see this by slowly paging back through the frames (**LEFTARROW**). The object moves between the two positions.

In this way, you can create the animation by paging through the frames, position by position. Note that the location of the object is *directly* linked to the curves. When you change frames, the IPOs are always re-evaluated and re-applied. You can freely move the object within the same frame, but since you have changed frame, the object 'jumps' to the position determined by the IPO.

The rotation and size of the object are completely free in this example. They can be changed or animated with the "Insert key".

The other options in the Insert Key menu concern other possible IPOs, such as Rotation, Size and any combination of these.

The IPO Curves

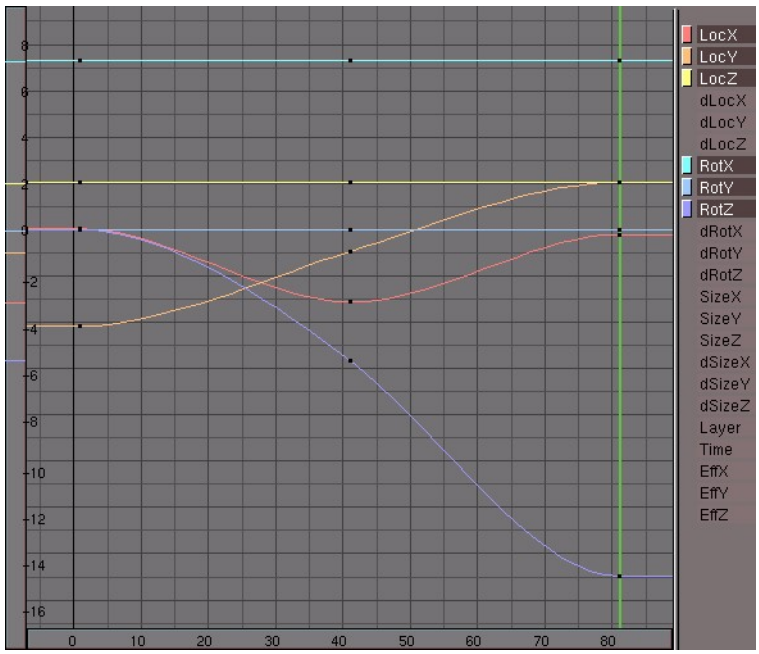


Figure 11-2. The IPO window.

Now we want to see exactly what happened. The first Screen for this is initialised in the standard Blender start-up file. Activate this Screen with **CTRL.LEFTARROW**. At the right we see the IPOWindow displayed (Figure 11-2). You can of course turn any window into an IPO window with the pertinent Window Type menu entry, but it is more handy to have both a 3D window and an IPO window at the same time. This shows all the IPOCurves, the channels used and those available. You can zoom in on the IPOWindow and translate, just as everywhere else in Blender with (**CTRL-MMB**).

In addition to the standard channels, you have the *delta* options, such as `dLocX`. These channels allow you to assign a *relative* change. This option is primarily used to control multiple objects with the same IPO. In addition, it is possible to work in animation 'layers'. You can achieve subtle effects this way without having to draw complicated curves.

Each curve can be selected individually with the **RMB**. In addition, the Grabber and Size modes operate here just as in the 3DWindow. By selecting all curves (**AKEY**) and moving them to the right (**GKEY**), you can move the complete animation in time.

Each curve can be placed in EditMode individually, or it can be done collectively. Select the curves and press **TAB**. Now the individual vertices and *handles* of the curve are displayed. The Bezier handles are coded, just like the curve object:

- Free Handle (black). This can be used any way you wish. Hotkey: **HKEY** (switches between Free and Aligned).
- Aligned Handle (pink). This arranges all the handles in a straight line. Hotkey: **HKEY** (toggles between Free and Aligned).
- Vector Handle (green). Both parts of a handle always point to the previous or next handle. Hotkey: **VKEY**.
- Auto Handle (yellow). This handle has a completely automatic length and direction. Hotkey: **SHIFT-HKEY**.

Handles can be moved by first selecting the middle vertex with **RMB**. This selects the other two vertices as well. Then immediately start the Grab mode with **RMB**-hold and move. Handles can be *rotated* by first selecting the end of one of the vertices and then use the Grabber by means of the **RMB**-hold and move action.

As soon as handles are rotated, the type is changed automatically:

- Auto Handle becomes Aligned.
- Vector Handle becomes Free.

"Auto" handles are placed in a curve by default. The first and last Auto handles always move horizontally, which creates a fluid interpolation.

The IPOCurves have an important feature that distinguishes them from normal curves: it is impossible to place more than one curve segment horizontally. Loops and circles in an IPO are senseless and ambiguous. An IPO can only have 1 value at a time. This is automatically detected in the IPOWindow. By moving part of the IPOCurve horizontally, you see that the selected vertices move 'through' the curve. This allows you to duplicate parts of a curve (**SHIFT-D**) and to move them to another time frame.

It is also important to specify how an IPOCurve must be read *outside* of the curve itself. There are four options for this in the IPOHeader (Figure 11-3).

Figure 11-3. IPO extension options.

Thre effect of each of these can be appreciated in (Figure 11-4).

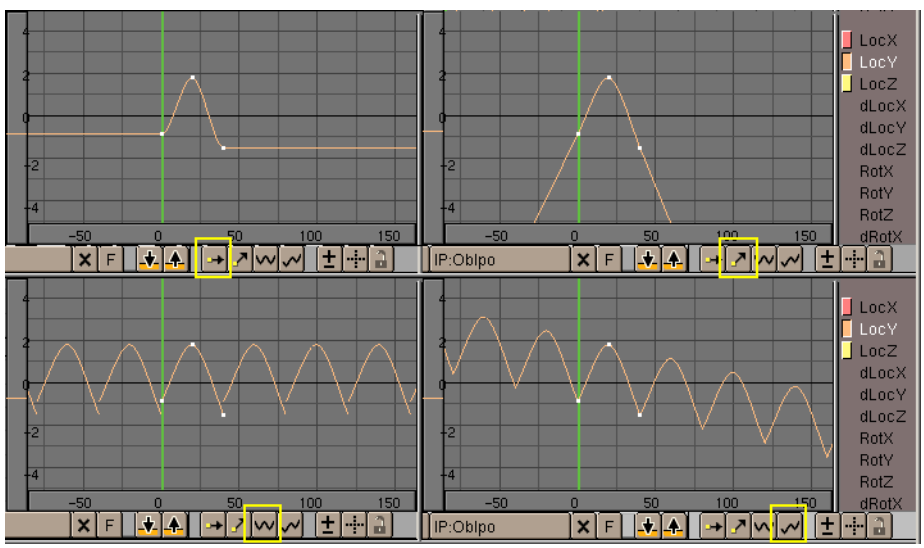


Figure 11-4. Extended IPOs.

From left to right:

-

Extend mode Constant:

The ends of selected IPOCurves are continuously (horizontally) extrapolated. It is the default behaviour.

•

Extend mode Direction:

The ends of the selected IPOCurves continue in the direction in which they ended.

•

Extend mode Cyclic:

The complete width of the IPOCurve is repeated cyclically.

•

Extend Mode Cyclic Extrapolation:

The complete width of the IPOCurve is extrapolated cyclic.

In addition to Beziers, there are two other possible types for IPOCurves. Use the **TKEY** command to select them. A Pop-up menu asks what type the selected IPOCurves must be:

- `Constant` - after each vertex of the curve, this value remains constant. No interpolation takes place.
- `Linear` - linear interpolation occurs between the vertices.
- `Bezier` - the standard fluid interpolation.

The IPO curves need not be set only by Key Framing. They can also be drawn 'by hand'. Use the **CTRL-LMB** command. Here are the rules:

•

There is no IPO block yet (in this window) *and* one *channel* is selected:

a new IPOBlock is created along with the first IPOCurve with one vertex placed where the mouse was clicked.

•

There is already an IPO block, and a *channel* is selected without an IPOCurve:

a new IPOCurve with one vertex is added.

•

There is already an IPO block, and a *channel* is selected withh an existing IPOCurve:

A new point is added to the selected IPOCurve.

This is *not* possible if multiple IPOCurves are selected or in EditMode.

> **Make an object rotate:** This is the best method for specifying axis rotations quickly. Select the object. In the IPOWindow, press one of the "Rot" channels and use **CTRL+LMB** to insert two points. If the axis rotation must be continuous, you must use the button IPOHeader->"Extend mode Directional".

One disadvantage of working with motion curves is that the *freedom* of transformations is limited. You can work quite intuitively with motion curves, but only if this can be processed on an XYZ basis. For a location, this is outstanding, but for a size and rotation there are better mathematical descriptions available: matrices (3x3 numbers) for size and quaternions (4 numbers) for rotation. These could also have been

processed in the channels, but this can quite easily lead to confusing and mathematically complicated situations.

Limiting the *size* to the three numbers XYZ is obvious, but this limits it to a rectangular distortion. A diagonal scaling such as 'shearing' is impossible. Simply working in hierarchies can solve this. A *non*-uniform scaled Parent will influence the rotation of a Child as a 'shear'.

The limitation of the three number XYZ rotations is less intuitive. This so-called Euler rotation is not uniform - the same rotation can be expressed with different numbers - and has the bothersome effect that it is *not* possible to rotate from any position to another, the infamous *gimbal lock*. While working with different rotation keys, the user may suddenly be confronted with quite unexpected interpolations, or it may turn out to be impossible to force a particular axis rotation when making manual changes. Here, also, a better solution is to work with a hierarchy. A Parent will *always* assign the specified axis rotation to the Child. (It is handy to know that the X, Y and Z rotations are calculated one *after* the other. The curve that affects the `RotX` channel, *always* determines the X axis rotation).

Luckily, Blender calculates everything internally with matrices and quaternions. Hierarchies thus work normally, and the Rotate mode does what you would expect it to. Only the IPOs are a limitation here, but in this case the ease of use prevails above a not very intuitive mathematical purity.

IPO Curves and IPO Keys

The easiest way to work with motion curves is to convert them to IPOKeys. We return to the situation in the previous example: we have specified two positions in an object IPO in frame 1 and frame 31 with "Insert Key". At the right of the screen, you can see an IPOWindow. We set the current frame to 21 (Figure 11-5).

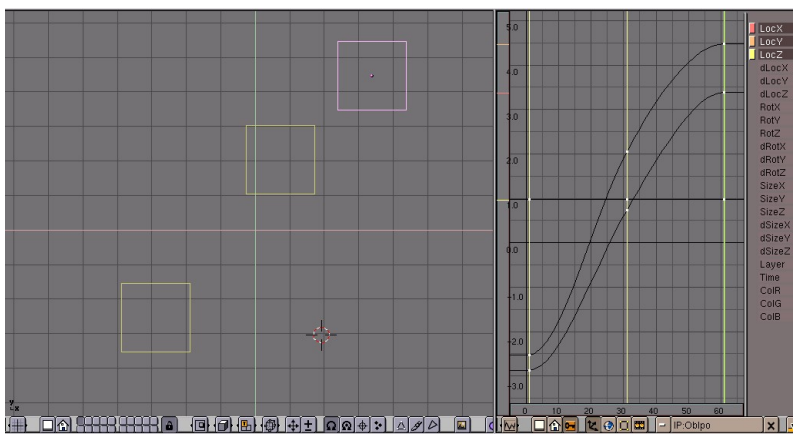


Figure 11-5. Figure

Press **KKEY** while the mouse cursor is in the 3DWindow. Two things will happen now:

- The IPOWindow switches to IPOKey mode.
- The selected object is assigned the "DrawKey" option.

The two actions each have separate meanings.

- The IPOWindow now draws vertical lines through all the vertices of all the visible IPOCurves (IPOS are now black). Vertices with the same 'frame' value are linked

to the vertical lines. The vertical lines (the "IPOKeys") can be selected, moved or duplicated, just like the vertices in EditMode. You can translate the IPOKeys only horizontally.

- The object is not only shown in its current position but 'ghost' objects are also shown at all the Key positions. In addition to now being able to visualize the key positions of the object, you can also modify them *in* the 3DWindow. In this example, use the Grab mode on the object to change the *selected* IPOKeys.

Below are a number of instructions for utilizing the power of the system:

- You can only use the **RMB** to select IPOKeys in the IPOWindow. Border select, and *extend* select, are also enabled here. Select all IPOKeys to transform the complete animation system in the 3DWindow.
- The "Insert Key" always affects *all* selected objects. The IPOKeys for multiple objects can also be transformed simultaneously in the 3DWindow. Use the **SHIFT-K** command: `Show and select all keys` to transform complete animations of a group of objects all at once.
- Use the **PAGEUP** and **PAGEDOWN** commands to select subsequent keys in the 3DWindow.
- You can create IPOKeys with each arrangement of channels. By consciously *excluding* certain channels, you can force a situation in which changes to key positions in the 3DWindow can only be made to the values specified by the visible channels. For example, with only the channel `LocX` selected, the keys can only be moved in the X direction.
- Each IPOKey consists of the vertices that have *exactly* the same frame value. If vertices are moved manually, this can result in large numbers of keys, each having only one curve. In this case, use the **JKEY** ("Join") command to combine selected IPOKeys. It is also possible to assign selected IPOKeys vertices for *all* the visible curves: use **IKEY** in the IPOWindow and choose "Selected keys".
- The DrawKey option and the IPOKey mode can be switched on and off independently. Use the button EditButtons->DrawKey to switch off this option or object. You can switch IPOKey mode on and off yourself with **KKEY** in the IPOWindow. Only **KKEY** in the 3DWindow turns on/off both the DrawKey and IPOKey mode.

Other applications of IPO Curves

There are several other application for IPOs other than just animating an Object movement.

The buttons in Figure 11-6 allow IPO Block type selection, the active one there is the Object IPO described up to now. Then follows Material IPO, World IPO, Vertex Keys IPO, Constraints IPO and Sequence IPO. Not every button is always present. Another one, which replaces the Vertex Keys IPO is the Curve IPO which appears if the selected object is a curve and not a Mesh.

Figure 11-6. The IPO window.

Material IPO is a way of animating a Material. Just as with objects, IpoCurves can be used to specify 'key positions' for Materials. With the mouse in the ButtonsWindow, the command **IKEY** calls up a pop-up menu with options for the various Material variables. If you are in a Material IPO Block then a small Num Button appears next to the red sphere material button in the IPO window toolbar. This indicates

which texture channel is active. The mapping for all 8 channels can be controlled with IpoCurves!

Strictly speaking, with textures two other animations are possible. Since Objects can give texture coordinates on other objects (Each object in Blender can be used as a source for texture coordinates. To do this, the option "Object" must be selected in the green "Coordinates input" buttons and the name of the object must be filled in. An inverse transformation is now performed on the global render coordinate to obtain the *local* object coordinate) it is possible to animate the texture simply by animating the location, size, and rotation of the object.

Furthermore, at each frame, Blender can be made to load another (numbered) Image as a texture map instead of having a fixed one. It is also possible to use SGI movie files or AVI files for this.

Path Animation

A different way to have Objects move in the space is to constrain them to follow a given path.

When objects need to follow a path, or it is too hard to animate a special kind of movement with the keyframe method (Think of a planet following its way around the Sun. Animating that with keyframes is virtually impossible) curve objects can be used for the 3D display of an animation path.

If the Curve object contains more than a single continuous curve only the first curve in the object is then used.

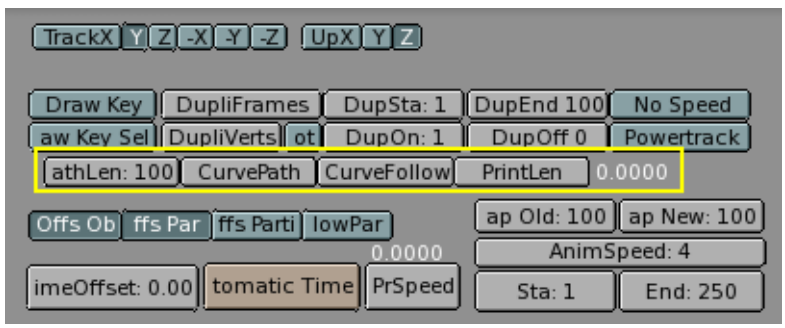


Figure 11-7. The IPO window.

Any kind of curve can become a path by setting the option `CurvePath` Toggle Button in the Animation Buttons window (**F7**) to ON (Figure 11-7).

When a Curve is turned to a Path all Child objects of the Curve move along the specified path. It is a good idea to set the Curve to 3D via the `3D` Toggle Button of the Curve Edit Buttons so that the paths can be freely modelled.

Otherwise, in the ADD menu under Curve->Path, there is a primitive with the correct settings already there. This is a 5th order Nurbs spline, which can be used to create very fluid, continuous movements.

Normally a Path is 100 frames long and it is followed in 100 frames by children. You can make it longer or shorter by varying the `PathLength:` Num Button.

The *speed* along a path is determined with an appropriate curve in the IpoWindow. To see it, the IPO Window Header button with the 'arrow' icon must be pressed in. The complete path runs in the IpoWindow between the vertical values 0.0 and 1.0. Drawing a curve between these values links the time to the position on the path. Backward and pulsing movements are possible with this. For most paths, an IpoCurve must run *exactly* between the Y-values 0.0 and 1.0. To achieve this, use the Number menu

(**NKEY**) in the IpoWindow. If the IpoCurve is deleted, the value of AnimButtons->PathLen determines the duration of the path. A linear movement is defined in this case.

The Speed IPO is a finer way of controlling Path length. The path is long 1 for time IPO, and if the time IPO goes from 0 to 1 in 200 frames then the path is 200 frames long.

Using the option `CurveFollow`, a rotation is also given to the Child objects of the path, so that they permanently point in the direction of the path. Use the "tracking" buttons in the AnimButtons to specify the effect of the rotation (Figure 11-8):

Figure 11-8. Tracking Buttons

`TrackX, Y, Z, -X, -Y, -Z` This specifies the direction axis, i.e. the axis that is placed on the path.

`UpX, UpY, UpZ (RowBut)` Specifies which axis must point 'upwards', in the direction of the (local) positive Z axis. If the `Track` and the `Up` axis coincides, it is deactivated.

Curve paths cannot be given uniform rotations that are perpendicular to the local Z axis. That would make it impossible to determine the 'up' axis.

To visualize these rotations precisely, we must make it possible for a Child to have its own rotations. Erase the Child's rotation with **ALT-R**. Also erase the "Parent Inverse": **ALT-P**. The best method is to 'parent' an *unrotated* Child to the path with the command **SHIFT-CTRL-PKEY**: "Make parent without inverse". Now the Child jumps directly to the path and the Child points in the right direction.

3D paths also get an extra value for each vertex: the 'tilt'. This can be used to specify an axis rotation. Use **TKEY** in EditMode to change the tilt of selected vertices in EditMode, e.g. to have a Child move around as if it were on a roller coaster.

The Time Ipo

With the TimeIpo curve you can manipulate the animation time of objects without changing the animation or the other Ipos. In fact, it changes the mapping of animation time to global animation time (Figure 11-9).

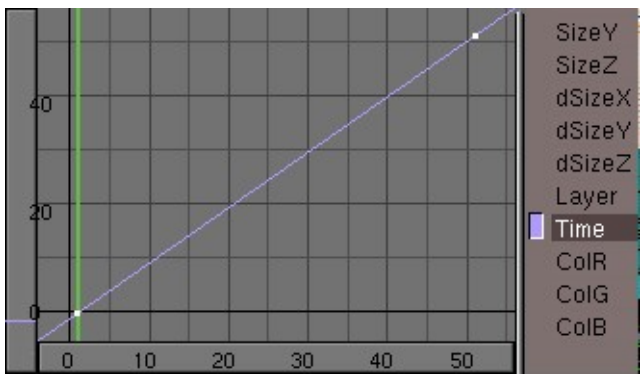


Figure 11-9. Linear time IPO

To grasp this concept, make a simple keyframe-animation of a moving object and create a TimeIpo in the IpoWindow. In frames where the slope of the TimeIpo is positive, your object will advance in its animation. The speed depends on the value of the slope. A slope bigger than 1 will animate faster than the base animation. A slope smaller than 1 will animate slower. A slope of 1 means no change in the animation, negative power slopes allow you to reverse the animation.

The TimeIpo is especially interesting for particle systems, allowing you to "freeze" the particles or to animate particles absorbed by an object instead of emitted. Other possibilities are to make a time lapse or slow motion animation.

> **Multiple Time IPOs:** You need to copy the TimeIpo for every animation system to get a full slow motion. But by stopping only some animations, and continue to animate, for example, the camera you can achieve some very nice effects (like those used to stunning effect in the movie "The Matrix")

Figure 11-10 shows a complex application. We want to make a fighter dive into a canyon, fly next to the water and then rise again, all this by following it with our camera and, possibly, having reflection in the water!

To do this we will need three paths. Path 1 has a fighter parented to it, the fighter will fly following it.

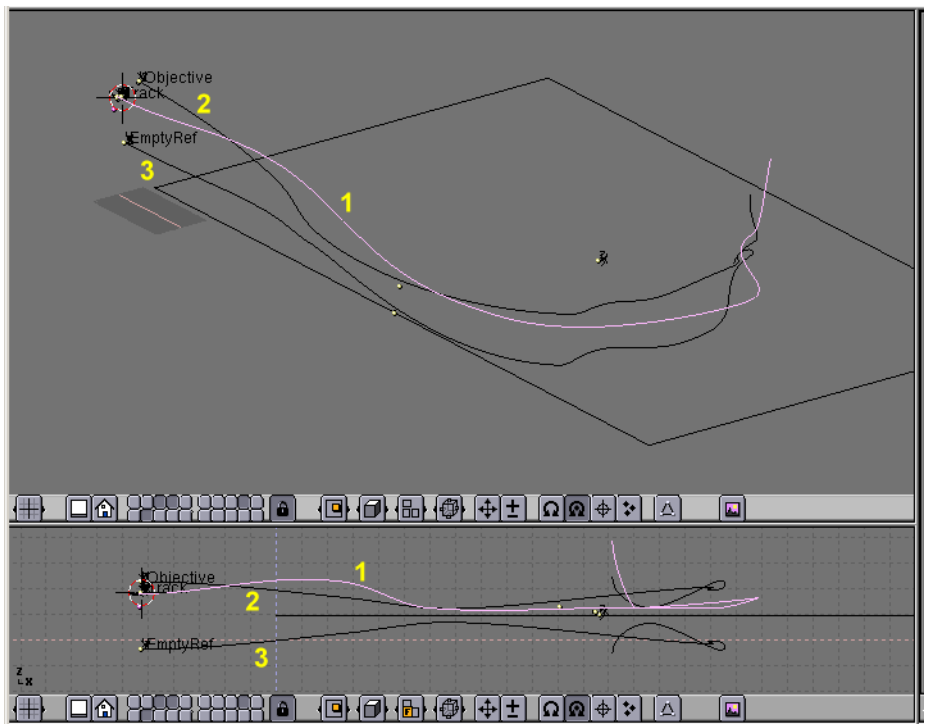


Figure 11-10. Complex path animation

The fighter has an Empty named 'Track' Parented to it in a strategic position. A camera is then parented to another curve, Path 2, and follows it, tracking the 'Track' Empty. The Fighter has a constant time IPO, the camera has not. It goes faster, then slower, always tracking the Empty, and hence the fighter, so we will have very fluid movements of the camera from Fighter side, to Fighter front, other side, back, etc. (Figure 11-11)

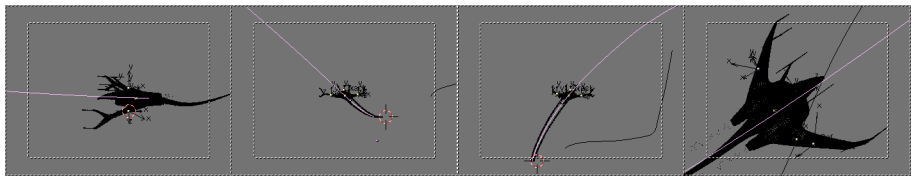


Figure 11-11. Some frames, the camera fluidly trqack the fighter.

Since we want our fighter to fly over a river, we need to set up an Env Map for the water surface to obtain reflections. But the Empty used for the calculations must be always in specular position with respect to the camera... and the camera is moving along a path!

Path 3 is hence created by mirroring path 2 with respect to the water plane, by duplicating it, and using **SKEY, YKEY** in Edit Mode with respect to the cursor, once the cursor is on the plane.

The Empty for the Env Map calculation is then parented to this new path, and the Time IPO of Path 2 is copied to Path 3. Figure 11-12 shows a rendered frame. Some particle systems were used for trails.

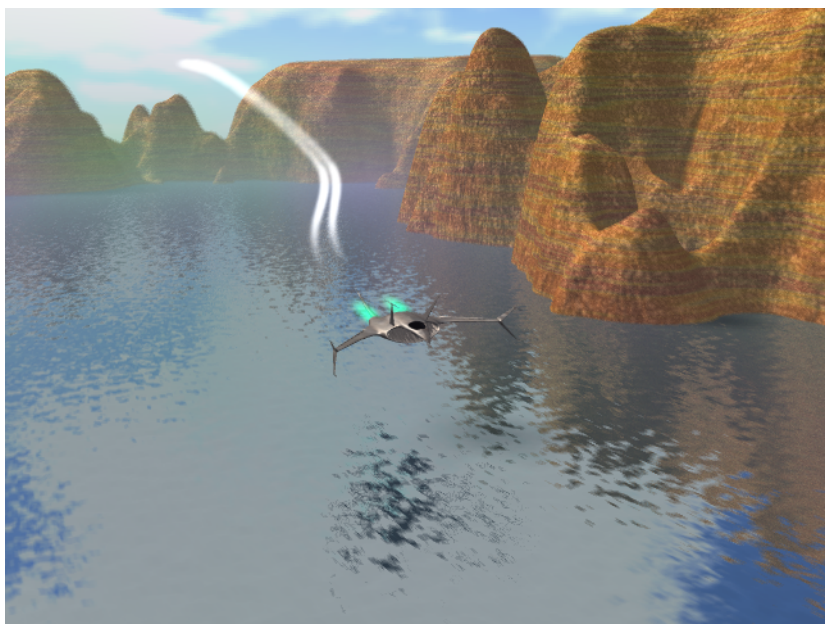


Figure 11-12. A frame of the final animation.

Chapter 12. Animation of Deformations

Animating an Object/Material, whatever as it is is not the only thing you can do in Blender. You can change, reshape, deform your objects in time!

There are many ways of acheiving this actually, and one technique is so powerfull and general there is a full chapter for it: Character animation. The other techniques will be handled here.

Absolute Vertex Keys

VertexKeys, as opposed to Object keys, the specified positions of objects, can also be created in Blender; VertexKeys are the specified positions of vertices *within* an Object. Since this can involve thousands of vertices, separate motion curves are not created for each vertex, the traditional Key position system is used instead. A single IpoCurve is used to determine how interpolation is performed and the times at which a VertexKey can be seen.

VertexKeys are part of the Object Data, not of the Object. When duplicating the Object Data, the associated VertexKey block is also copied. It is not possible to permit multiple Objects to share the same VertexKeys in Blender, since it would not be very practical.

The Vertex Key block is universal and understands the distinction between a Mesh, a Curve, a Surface or a Lattice. The interface and use is therefore unified. Working with Mesh VertexKeys is explained in detail in this section, which also contains a number of brief comments on the other Object Data.

The first VertexKey position that is created is always the *reference* Key. This key defines the texture coordinates. Only if this Key is *active* can the faces and curves, or the *number* of vertices, be changed. It is allowed to assign other Keys a different number of vertices. The Key system automatically interpolates this.

A practical example is given below. When working with VertexKeys, it is very handy to have an IpoWindow open. Use the first Screen from the standard Blender file, for example. In the IpoWindow, we must then specify that we want to see the VertexKeys. Do this using the Icon button with the vertex square ( ). Go to the 3DWindow with the mouse cursor and press **IKEY**. With a Mesh object selected and active. The "Insert Key" menu has several options, the latter being `Mesh`. As soon as this has been selected, a new dialog appears (Figure 11-2) asking for Relative or absolute Vertex Key.

Figure 12-1. Insert Key Menu.

We will choose "Absolute Vertex Key" a yellow horizontal line is drawn in the IpoWindow. This is the first key and thus the *reference* Key. An IpoCurve is also created for "Speed" (Figure 11-3).

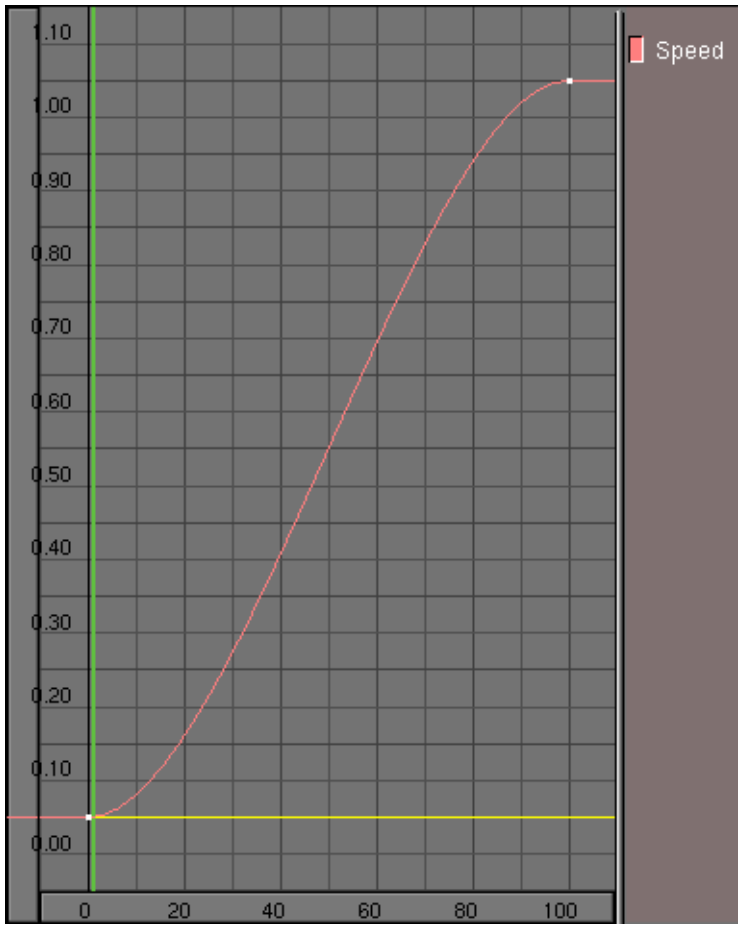


Figure 12-2. Insert Key Menu.

> **Vertex Key creation:** Creating VertexKeys in Blender is very simple, but the fact that the system is very sensitive in terms of its configuration, can cause a number of 'invisible' things to happen. The following rule must therefore be taken into consideration.
>
> As soon as a VertexKey position is inserted it is *immediately active*. All subsequent changes in the Mesh are linked to *this* Key position. It is therefore important that the Key position be added *before* editing begins.

Go a few frames further and again select: **IKEY**, **ENTER** (in the 3DWindow). The second Key is drawn as a light blue line. This is a normal Key; this key and all subsequent Keys affect only the vertex information. Press **TAB** for EditMode and translate one of the vertices in the Mesh. Then browse a few frames back: nothing happens! As long as we are in EditMode, other VertexKeys are *not* applied. What you see in EditMode is *always* the *active* VertexKey.

Leave EditMode and browse through the frames again. We now see the effect of the VertexKey system. VertexKeys can only be selected in the IpoWindow. We always do this *out* of EditMode: the 'contents' of the VertexKey are now temporarily displayed in the Mesh. We can edit the specified Key by starting Editmode.

There are three methods for working with Vertex Keys:

- The 'performance animation' method. This method works entirely in EditMode, chronologically from position to position:
  - Insert Key. The reference is specified.

  - A few frames further: Insert Key. Edit the Mesh for the second position.
  - A few frames further: Insert Key. Edit the Mesh for the third position.
  - Continue the above process...

- The 'editing' method.
  - We first insert all of the required Keys, unless we have eady created the Keys using the method described above.
  - Blender is *not* in EditMode.
  - Select a Key. Now start EditMode, change the Mesh and leave EditMode.
  - Select a Key. Start EditMode, change the Mesh and leave EditMode.
  - Continue the above process....

- The 'insert' method
  - Whether or not there are already Keys and whether or not we are in EditMode does not matter in this method.
  - Go to the frame in which the new Key must be inserted.
  - Insert Key.
  - Go to a new frame, Insert Key.
  - Continue the above process...

While in EditMode, the Keys cannot be switched. If the user attempts to do so, a warning appears.

Each Key is represented by a line which is drawn at a given height. Height is chosen so that the key intersects the "Speed" IPO at the frame at which the Key is taken.

Both the IpoCurve and the VertexKey can be separately selected with **RMB**. Since it would otherwise be too difficult working with them, selection of the Key lines is switched off when the curve is in EditMode. The *channel* button can be used to temporarily hide the curve (**SHIFT-LMB** on "Speed") to make it easier to select Keys.

The Key lines in the IpoWindow, once taken, can be placed at any vertical position. Select the line and use Grab mode to do this. The IpoCurve can also be processed here in the same way as described in the previous chapter. Instead of a 'value', however, the curve determines the interpolation between the Keys, e.g. a sine curve can be used to create a cyclical animation.

During the animation the frame count gives a certain value of the speed IPO, which is used to chose the Key(s) which is/are to be used, possibly with interpolation, to produce the deformed mesh.

The Speed IPO has the standard behaviour of an IPO, also for interpolation. The Key line has three different interpolation types. Press **TKEY** with a Key line selected to to open a menu with the options:

- `Linear`: interpolation between the Keys is linear. The Key line is displayed as a dotted line
- `Cardinal`: interpolation between the Keys is fluid, the standard setting.
- `BSpline`: interpolation between the Keys is extra fluid and includes four Keys in the interpolation calculation. The positions are no longer displayed precisely, however. The Key line is drawn as a dashed line.

Figure 12-3 shows a simple Vertex Key animation of a cylinder. When run the cylinder deforms to a big star, then deforms to a small star, then, since the Speed IPO goes back to 0 the deformation is repeated in reverse order.

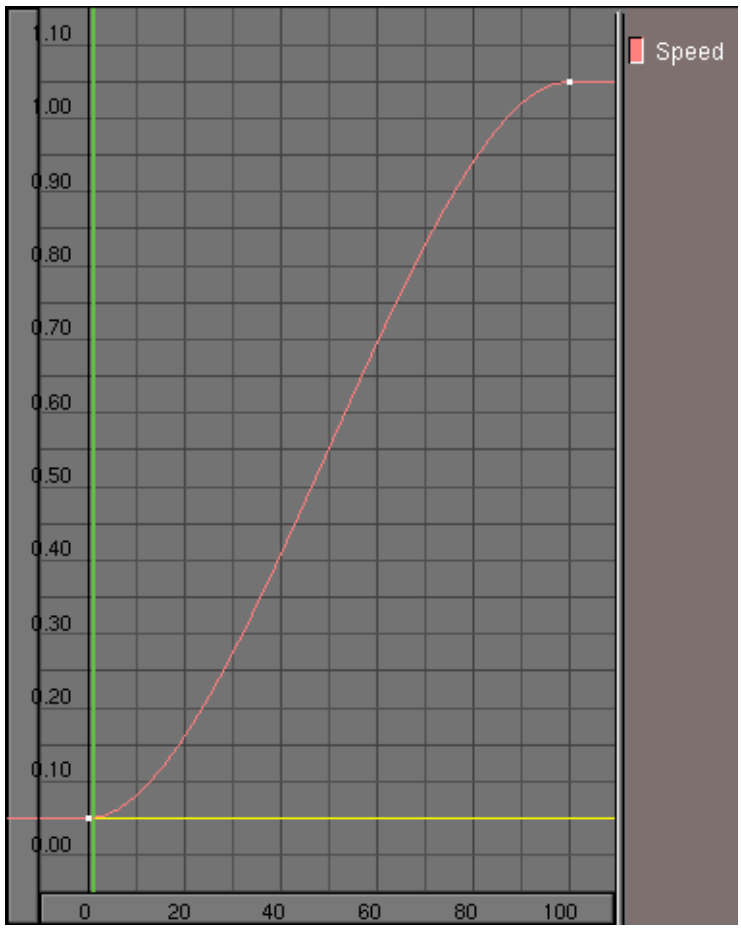


Figure 12-3. Absolute Keys.

Some useful tips:

- Key positions are *always* added with IKEY, even if they are located at the same position. Use this to copy positions when inserting. Two key lines at the same position can also be used to change the effect of the interpolation.
- If *no* Keys are selected, EditMode can be invoked as usual. However, when you leave EditMode, all changes are undone. Insert the Key *in* EditMode in this case.
- For Keys, there is *no* difference between selected and *active*. It is therefore not possible to select multiple Keys.
- When working with Keys with differing numbers of vertices, the faces can become disordered. There are no tools that can be used to specify *precise* sequence of vertices. This option is actually suitable only for Meshes that have only vertices such as Halos.
- The `Slurph` button in the Animation buttons is an interesting option. The Slurph number indicates the interpolation of Keys *per vertex* with a fixed delay. The first vertex comes first and the last vertex has a delay of "Slurph" frames. This effect makes it possible to create very interesting and lively Key framing. Pay special attention to the sequence of the vertices for Meshes. They can be sorted using the

button `Xsort` in the Edit Buttons or made random using the command `Hash` of the same buttons. This must of course be done *before* the VertexKeys are created. Otherwise, unpredictable things can will happen (this is great for Halos though!).

Curve and Surface Keys

As mentioned earlier in this manual, Curve and Surface Keys work exactly the same way as Mesh Keys. For Curves, it is particularly interesting to place Curve Keys in the bevel object. Although this animation is *not* displayed real-time in the 3DWindow, but it will be rendered.

Lattice Keys

Lattice Vertex Keys can be applied in a variety of ways by the user. When combined with "slurping", they can achieve some interesting effects. As soon as one Key is present in a Lattice, the buttons that are used to determine the resolution are blocked.

Relative VertexKeys

Relative Vertex Keys (RVK) works differently inasmuch only the difference between the reference mesh and the deformed mesh is stored. Ths allows for blending several keys together to achieve complex animations.

We will walk through RVK via an example.

We will create a facial animation via RVK. While Absolute Vertex Keys are controlled with only *one* IPO curve, Relative Vertex Keys are controlled by one interpolation curve for every key position, which states 'how much' of that relative deformation is used to produce the deformed mesh. This is why relative keys can be mixed (added, subtracted, etc.).

For facial animation, the base position might be a relaxed position with a slightly open mouth and eyelids half open. Then keys would be defined for left/right eye-blink, happy, sad, smiling, frowning, etc.

The trick with relative vertex keys is that only the vertices that are changed between the base and the key affect the final output during blending. This means it is possible to have several keys affecting the object in different places all at the same time.

For example, a face with three keys: smile, and left/right eye-blink could be animated to smile, then blink left eye, then blink right eye, then open both eyes and stop smiling - all by blending 3 keys. Without relative vertex keys 6 vertex keys would have needed to be generated, one for each target position.

Consider the female head in Figure 12-4

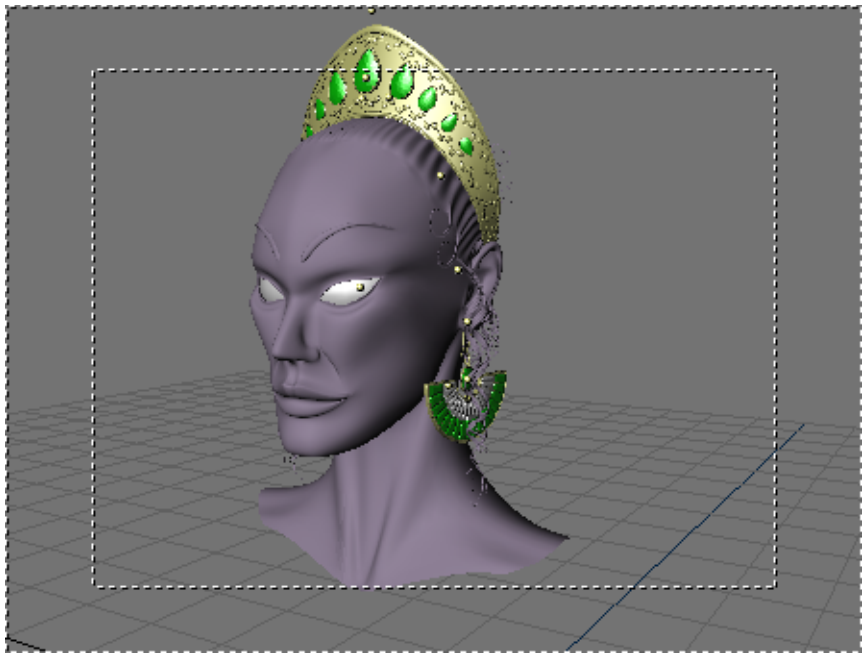


Figure 12-4. The female head we want to animate.

To add a RVK just press **IKEY** and select `Mesh` as for AVK, but, from the pop up menu select `Relative Vertex Keys` This stores the reference Key which will appear as an yellow horizontal line in the IPO window.

Relative keys are defined by inserting further vertex keys. Each time the **IKEY** is pressed and `Mesh` selected a new horizontal line appears in the IPO window. If frame number is augmented each time the horizonta line are placed one above the other. For an easier modelling let's hide all vertices except those of the face Figure 12-5.

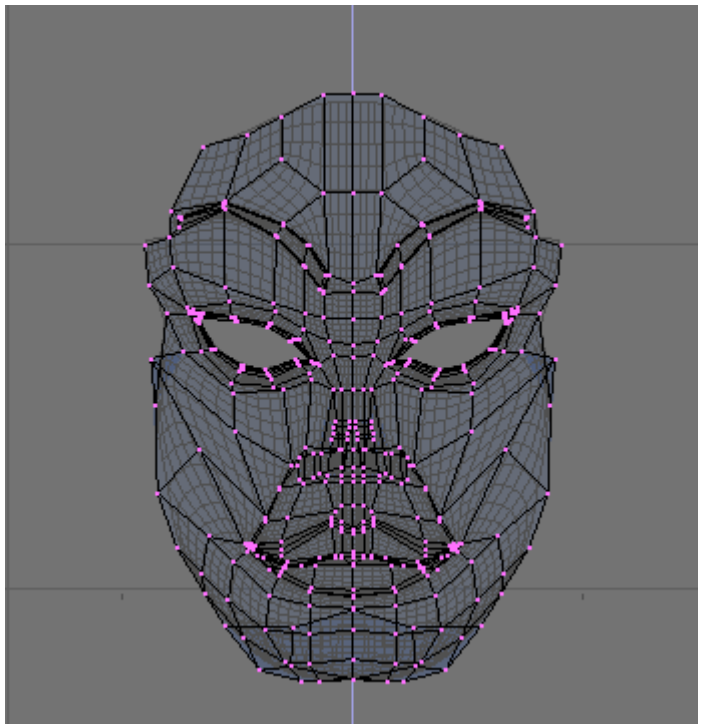


Figure 12-5. The female head we want to animate.

Now move to another frame, say number 5, and add a new Key. A cyan line will appear above the yellow, which now turns orage. Switch to Edit mode and close left eye lid.

When you are done exit from Edit Mode. If you select the reference key you will see the original mesh. If you select your first RVK you will se the deformed one (Figure 12-6).

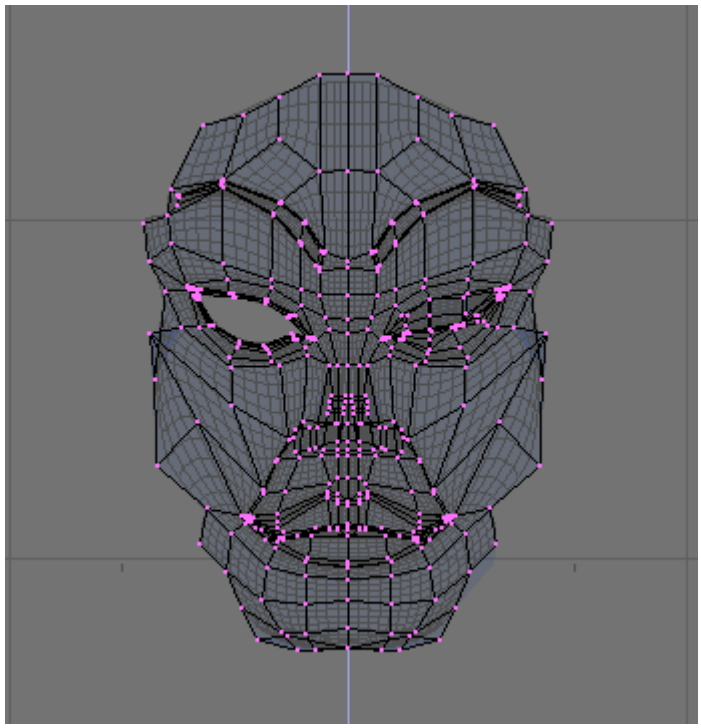


Figure 12-6. Left eye closed.

Repeat the step for the right eye. Beware that the newely inserted key is based on the mesh of the currently *active* key, so it is generally a good idea to select the reference key before pressing **IKEY**

Then add a smile (Figure 12-7).

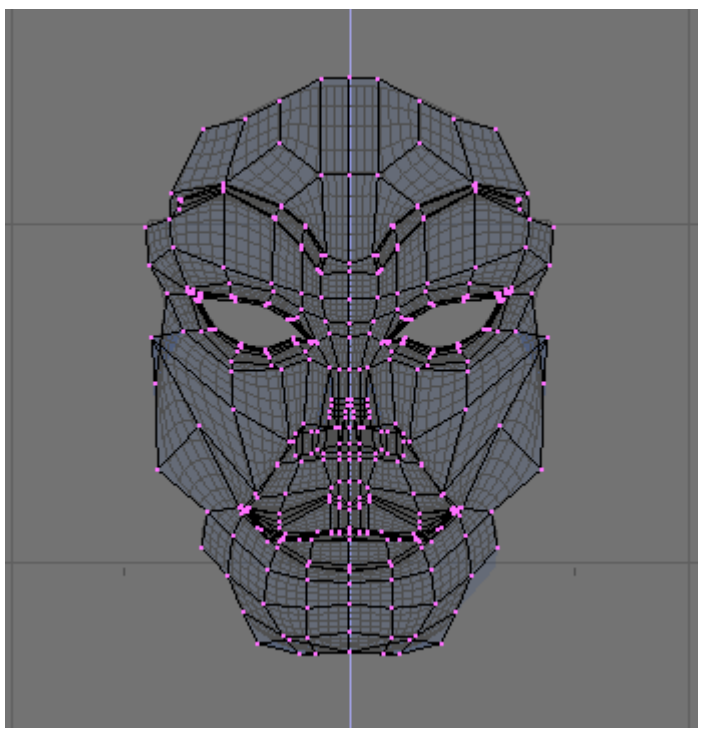


Figure 12-7. Smiling.

Your IPO window will look like Figure 12-8.

Figure 12-8. Smiling.

The vertical order of the vertex KEYs (the blue lines) from bottom to top determines its corresponding IPO curve, i.e. the lowest blue key line will be controlled by the `Key1` curve, the second lowest will be controlled by the `Key2` curve, and so on.

No IPO is present for the reference mesh since that is the mesh which is used if all other Keys have an IPO of value 0 at the given frame.

Select `Key1` and add an IPO with your favorite method. Make it look like Figure 12-9.

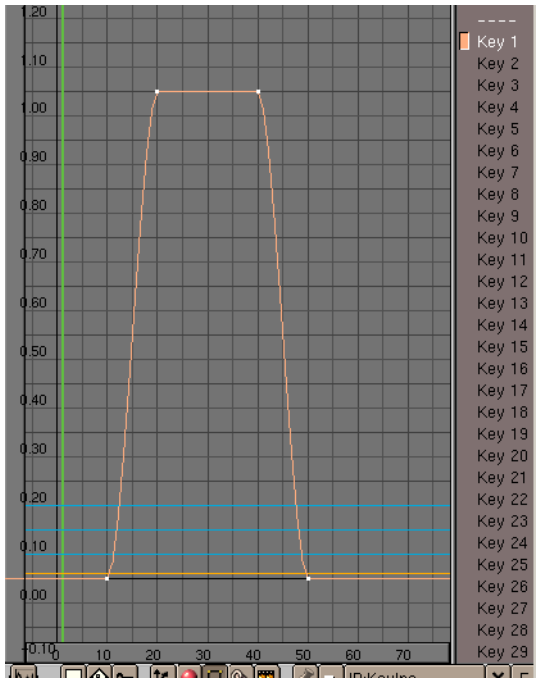


Figure 12-9. The IPO curve of Key 1.

This will make our mesh undeformed up to frame 10, then from frame 10 to frame 20 Key 1 will begin to affect the deformation. From frame 20 to frame 40 Key 1 will completely overcame the reference mesh (IPO value is 1), and the eye will be completely closed. The effect will fade out from frame 40 to frame 50.

You can check with **ALT A**, or by setting the frame numbers by hand. The second option is better, unless your computer is really powerfull!

Copy this IPO by using the down pointing arrow button in the IPO window toolbar Figure 12-10. Select the `Key 2` and paste the curve with the up pointing arrow. Now both keys will have the same influence on the face and both eyeds will close at a time.

Figure 12-10. Clipboard buttons.

> **Panning the Toolbar:** It may happen that the toolbar is longer than the window and some buttons are not shown. You can pan horizzontally all toolbars by clicking **MMB** on them and dragging the mouse.

Add also an IPO for `Key 3` Let's make this different (Figure 12-11).

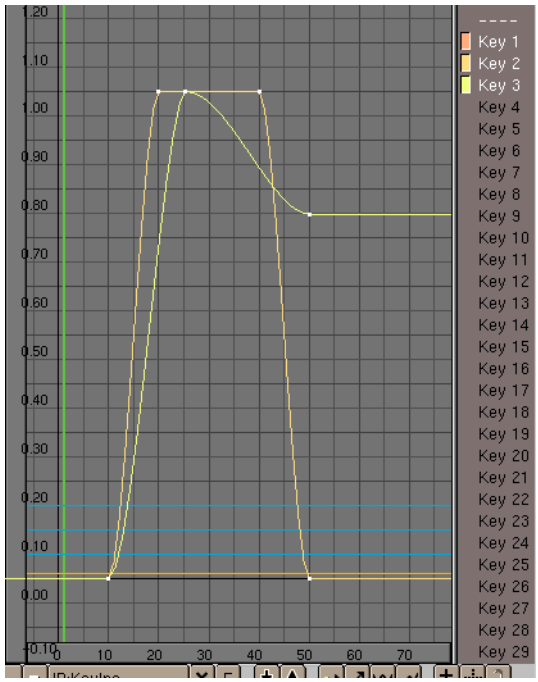


Figure 12-11. All IPOs.

This way the eyes closes and she begins to smile, smile is at maximum with closed eyes, then she smile 'less' while eyes re-open and keeps smiling (Figure 12-12).

Figure 12-12. All IPOs.

The IPO curve for each key controls the blending between relative keys. These curves should be created in the typical fashion. The final position is determined by adding all of the effects of each individual IPO curve.

> **Values out of [0,1] range:** An important part of relative keys is the use of additive or extrapolated positions. For example, if the base position for a face is with a straight mouth, and a key is defined for a smile, then it is possible that the negative application of that key will result in a frown. Likewise, extending the Ipo Curve above 1.0 will "extrapolate" that key, making an extreme smile.

Lattice Animation (x)

Parenting a mesh to a lattice is a nice way to apply deformations to the former while modeling, but it is also a way to make deformations in time!

You can use Lattices in animations in two ways:

- Animate the vertices with vertex keys (or relative vertex keys);
- Move the lattice or the child object of the lattice

The first technique is basically nothing new than what contained in the previus two sections but applied to a lattice which has an object parented to it.

With the second kind you can create animations that squish things between rollers, or achieve the effect of a well-known space ship accelerating to warp-speed.

Make a space ship and add a lattice around the ship. make the lattice with the parameters in Figure 12-13.

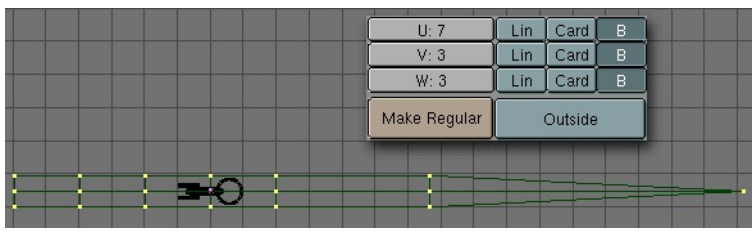


Figure 12-13. Lattice setup

I put the lattice into EditMode for this picture, so you can see the vertices. For working with lattices it is also good to switch on the "Outside" option in the EditButtons for the Lattice, as this will hide the inner vertices of the lattice.

Select the ship, extend the selection to the lattice (holding **SHIFT** while selecting), and press **CTRL-P** to make the lattice the parent of the ship. You should not see any deformation of the ship because the lattice is still regular.

For the next few steps it is important to do them in EditMode. This causes a deformation only if the child object is inside the Lattice. So now select the lattice, enter EditMode, select all vertices (**AKEY**), and scale the lattice along its x-axis (press **MMB** while initiating the scale) to get the stretch you want. The ship's mesh shows immediately the deformation caused by the lattice (Figure 12-14).

Figure 12-14. Stretching

Now edit the lattice in EditMode so that the right vertices have an increasing distance from each other. This will increase the stretch as the ship goes into the lattice. The right ends vertices I have scaled down so that they are nearly at one point; this will cause the vanishing of the ship at the end.

Select the ship again and move it through the lattice to get a preview of the animation. Now you can do a normal keyframe animation to let the ship fly through the lattice.

> **Camera tracking:** With this lattice animation, you can´t use the pivot point of the object for tracking or parenting. It will move outside the object. You will need to vertex-parent an

Empty to the mesh for that. To do so, select the Empty, then the mesh, enter EditMode and select one vertex, then press **CTRL-P**.

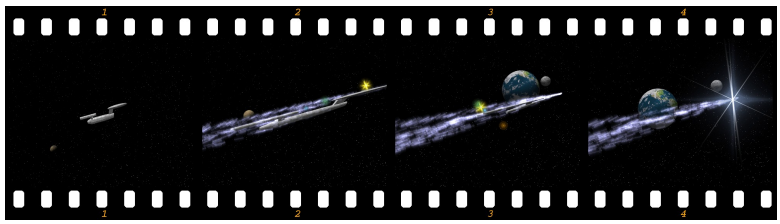


Figure 12-15. Some frames of the resulting animation.

Chapter 13. Character Animation (x)

General Tools

Auto-key

The auto-key feature can be found in the info bar. When it is enabled, Blender will automatically set keyframes when you move objects. This is helpful for people who are not used to explicitly inserting keyframes with **IKEY**. There are two separate toggles for auto-keying: one for object mode and one for pose mode. These two options can be set independently of one another.

Figure 13-1. Auto key options

For Objects

`KeyOB` will set keyframes for objects that are moved in object mode. Users who are familiar with the Blender interface will likely want to leave this option disabled.

For Actions

`KeyAC` sets keyframes for transformations done in pose mode. This ensures that you will not lose a pose by forgetting to insert keyframes. Even users who are familiar with the Blender interface may find this to be a useful feature.

Ipo/Action Pinning

It is now possible to display different ipos in different windows. This is especially valuable while editing actions, which have a different ipo for each bone.

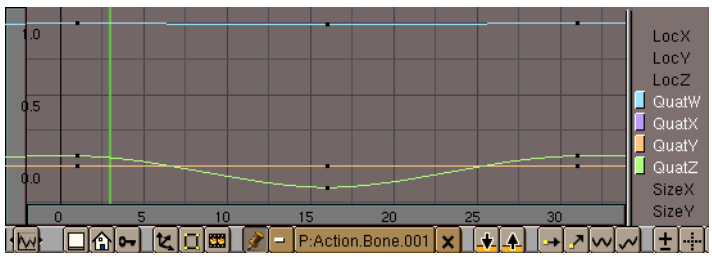


Figure 13-2. Pinned Action IpoWindow

You can "pin" an ipo or action (lock it to the current window) by pressing the pin icon in the header of the window. The contents of the window will stay there, even

when the object is deselected, or another object is selected. Note that the color of the ipo block menu will change, along with the background color of the ipo window. These serve as reminders that the window is not necessarily displaying the ipo of the currently selected object.

Browsing while pinned

The browse menu is still available while a window is pinned. In this case however, changing the current data will not affect the current object; it merely changes which data is displayed.

Armature Object

Creating

A single armature will contain many bones. Consider an armature to be like a skeleton for a living creature. The arms, legs, spine and head are all part of the same skeleton object.

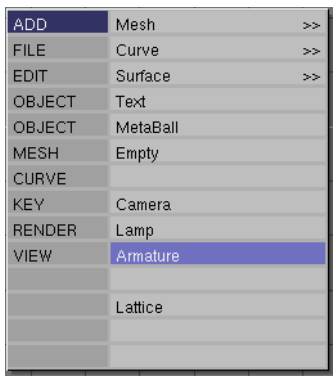


Figure 13-3. Adding an Armature

To create a new armature, select "ADD->Armature" from the toolbox. A new bone will appear with its root at the location of the 3d cursor. As you move the mouse, the bone will resize accordingly. **LMB** will finalize the bone and start a new one that is the child of the previous one. In this way you can make a complete chain. Pressing **ESC** will cancel the addition of the bone.

Adding Bones

You can add another bone to an armature while it is in edit mode by selecting "ADD->Armature" from the toolbox again. This will start the bone-adding mode again, and the new bones you create will be a part of the current armature.

Extruding Bones

You can also extrude bones from existing bones by selecting a bone joint and pressing **EKEY**. The newly created bone will be a child of the bone it is extruded from.

Editing

While in edit mode, you can perform the following operations to the bones in an armature.

Adjusting

Select one or more bone joints and use any of the standard transformation operations to adjust the position or orientation of any bones in the armature. Note that IK chains cannot have any gaps between their bones and as such moving the end point of a bone will move the start point of its child.

You can select an entire IK chain at once by moving the mouse cursor over a joint in the chain and pressing **LKEY**. You can also use the boundary select tool (**BKEY**).

Deleting

You can delete one or more bones by selecting its start and end points. When you do this you will notice the bone itself will be drawn in a highlighted color. Pressing **XKEY** will remove the highlighted bones. Note that selecting a single point is insufficient to delete a bone.

Point Snapping

It is possible to snap bone joints to the grid or to the cursor by using the snap menu accessible with **SHIFT+S**.

Numeric Mode

For more precise editing, pressing **NKEY** will bring up the numeric entry box. Here you can adjust the position of the start and end points as well as the bone's roll around its own axis.

An easy way to automatically orient the z-axis handles of all selected bones (necessary for proper use of the pose-flipped option) is to press **CTRL+N**. Remember to do this before starting to create any animation for the armature.

Undo

While in edit mode, you can cancel the changes you have made in the current editing session by pressing **UKEY**. The armature will revert to the state it was in before editing began.

Joining

It is possible to join two armatures together into a single object. To do this, ensure you are in object mode, select both armatures and press **CTRL+J**.

Renaming

Assigning meaningful names the bones in your armatures is important for several reasons. Firstly it will make your life easier when editing actions in the action window. Secondly, the bone names are used to associate action channels with bones when

you are attempting to re-use actions, and thirdly the names are used when taking advantage of the automatic pose-flipping feature.

Note that bone names need only be unique within a given armature. You can have several bones called "Head" so long as they are all in different armatures.

Basic Naming

To change the names of one or more bones, select the bones in edit mode and switch to the edit buttons with **F9**. A list of all the selected bones should appear.

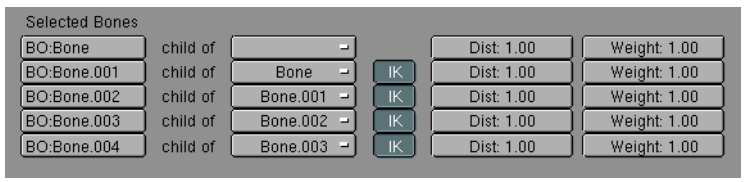


Figure 13-4. EditButtons for an Armature

Change a bone's name by **SHIFT-LMB** in the bone's name box and typing a new name.

It is easier to name the bones by either only editing one bone at a time, or by making sure the "DrawNames" option is enabled in the EditButtons **F9** (see 'Draw Options for Armatures').

Pose Flipping Conventions

Character armatures are typically axially symmetrical. This means that many elements are found in pairs, one on the left and one on the right. If you name them correctly, Blender can flip a given pose around the axis of symmetry, making animation of walk-cycles much easier.

For every bone that is paired, suffix the names for the left and right with either ".L" and ".R" or ".Left" and ".Right". Bones that lie along the axis of symmetry or that have no twin need no suffix. Note that the part of the name preceding the suffix should be identical for both sides. So if there are two hands, they should be named "Hand.R" and "Hand.L".

Basic Parenting

To change parenting relationships within the armature, select the bone that should be the CHILD and switch to the edit buttons window. Next to the bone there should be a menu button labeled "Child Of". To make the bone become the child of another bone, pick the appropriate parent from the list. Note that this is much easier if the bones have been correctly named. To dissolve a parenting relationship, choose the first (blank) entry in the list.

Note that the parenting menu only contains the names of valid parents. Bones that cannot be parents (such as children of the current bone) will not be displayed.

IK Relationship

The IK toggle next to each bone with a parent is used to determine if the IK solver should propagate its effects across this joint. If the IK button is active, the child's start point will be moved to math its parent's end point. This is to satisfy the requirement that there are no gaps in an IK chain. Deactivating the IK button will not restore the child's start point to its previous location, but moving the point will no longer affect the parent's end point.

Setting Local Axes

To get the best results while animating, it is necessary to ensure that the local axes of each bone are consistent throughout the armature. This should be done before any animation takes place.

Clearing Transforms

It is necessary that when the armature object is in its untransformed orientation in object mode, the front of the armature is visible in the front view, the left side is visible in the left view and so on. You can ensure this by orienting the armature so that the appropriate views are aligned and pressing **CTRL+A** to apply size and rotation. Again, this should be done before any animation takes place.

Adjusting Roll Handles

The orientation of the bones' roll handles is important to getting good results from the animation system.

You can adjust the roll angle of a bone by selecting it and pressing **NKEY**. The roll angle is the item at the bottom. The exact number that must be entered here depends on the orientation of the bone.

The Z-axis of each bone should point in a consistent direction for paired bones. A good solution is to have the Z-axes point upwards (or forwards, when the bone is vertically oriented).

This task is much easier if the "Draw Axes" option is enabled in the edit buttons window.

Setting Weights (DEPRECIATED)

The Weight and Dist settings are only used by the automatic skinning which is a depreciated feature.

Object Mode Parenting

When making a child of an armature, several options are presented.

Parent to Bone

In this case, the a popup menu appears allowing you to choose which bone should be the parent of the child(ren) objects.

Parent to Armature

Choosing this option will deform the child(ren) mesh(es) according to their vertex groups. If the child meshes don't have any vertex groups, they will be subject

to automatic skinning. This is very slow, so it is advised to create vertex groups instead.

Parent to Armature Object

Choosing this option will cause the child(ren) to consider the armature to be an Empty for all intents and purposes.

Toggle Buttons for Armatures in the EditButtons F9

Figure 13-5. Draw options for Armatures

Rest Position Button

When this toggle is activated, the armature will be displayed in its rest position. This is useful if it becomes necessary to edit the mesh associated with an armature after some posing or animation has been done. Note that the actions and poses are still there, but they are temporarily disabled while this button is pressed.

Draw Axes Button

When this toggle is activated, the local axes of each bone will be displayed in the 3d view.

Draw Names Button

When this toggle is activated, the names of each bone will be displayed in the 3d view.

Skinning

Skinning is a technique for creating smooth mesh deformations with an armature. Essentially the skinning is the relationship between the vertices in a mesh and the bones of an armature, and how the transformations of each bone will affect the position of the mesh vertices.

Automatic (DEPRECIATED)

If a mesh does not have any vertex groups, and it is made the armature-child of an armature, Blender will attempt to calculate deformation information on the fly. This is very slow and is not recommended. It is advisable to create and use vertex groups instead.

Vertex Weights

Figure 13-6. Vertex Groups

Vertex groups are necessary to define which bones deform which vertices. A vertex can be a member of several groups, in which case its deformation will be a weighted average of the deformations of the bones it is assigned to. In this way it is possible to create smooth joints.

Creating

To add a new vertex group to a mesh, you must be in edit mode. Create a new vertex group by clicking on the "New" button in the mesh's edit buttons.

A vertex group can subsequently be deleted by clicking on the "Delete" button.

Change the active group by choosing one from the pull-down group menu.

Naming

Vertex groups must have the same names as the bones that will manipulate them. Both spelling and capitalization matter. Rename a vertex group by **SHIFT-LMB** on the name button and typing a new name. Note that vertex group names must be unique within a given mesh.

Assigning

Vertices can be assigned to the active group by selecting them and clicking the "Assign" button. Depending on the setting of the "Weight" button, the vertices will receive more or less influence from the bone. This weighting is only important for vertices that are members of more than one bone. The weight setting is not an absolute value; rather it is a relative one. For each vertex, the system calculates the sum of the weights of all of the bones that affect the vertex. The transformations of each bone are then divided by this amount meaning that each vertex always receives exactly 100% deformation.

Assigning 0 weight to a vertex will effectively remove it from the active group.

Removing

Remove vertices from the current group by selecting them and clicking the "Remove" button.

Selection Tools

Pressing the "Select" button will add the vertices assigned to the current group to the selection set. Pressing the "Deselect" button will remove the vertices assigned to the current group from the selection set.

Weight Painting

Weight painting is an alternate technique for assigning vertices to vertex groups. The user can "paint" weights onto the model and see the results in real-time. This makes smooth joints easier to achieve.

Activating

To activate weight-painting mode, select a mesh with vertex groups and click on the weight paint icon.

The active mesh will be displayed in weight-color mode. In this mode dark blue represents areas with no weight from the current group and red represent areas with full weight.

Only one group can be visualized at a time. Changing the active vertex group in the edit buttons will change the weight painting display.

Painting

Weights are painted onto the mesh using techniques similar to those used for vertex painting, with a few exceptions.

The "color" is the weight value specified in the mesh's edit-buttons. The "opacity" slider in the vertex paint buttons is used to modulate the weight.

"Erasing Weight"

To erase weight from vertices, set the weight to "0" and start painting.

Posemode

To manipulate the bones in an armature, you must enter pose mode. In pose mode you can only select and manipulate the bones of the active armature. Unlike edit mode, you cannot add or delete bones in pose mode.

Entering

Enter pose mode by selecting an armature and pressing **CTRL+TAB**. Alternatively you can activate pose mode by selecting an armature and clicking on the pose mode icon in the header of the 3d window. You can leave pose mode by the same method, or by entering edit mode.

Editing

In pose mode, you can manipulate the bones in the armature by selecting them with **RMB** and using the standard transformation keys: **RKEY**, **SKEY** and **GKEY**. Note that you cannot "grab" (translate) bones that are IK children of another bone.

Press **IKEY** to insert keyframes for selected bones.

Clearing a pose

If you want to clear the posing for one or more bones, select the bones and press **ALT+R** to clear rotations, **ALT+S** to clear scaling and **ALT+G** to clear translations. Issuing these three commands will all bones selected will return the armature to its rest position.

Copy/Paste/Flipped

It is frequently convenient to copy poses from one armature to another, or from one action to a different point in the same action. This is where the pose copying tools come into play.

or best results, be sure to select all bones in editmode and press **CTRL+N** to auto-orient the bone handles before starting any animation.

To copy a pose, select one or more bones in pose mode, and click on the "Copy" button in the 3d window. The transformations of the selected bones are stored in the copy buffer until needed or until another copy operation is performed.

To paste a pose, simply click the "Paste" button. If "KeyAC" is active, keyframes will be inserted automatically.

To paste a mirrored version of the pose (if the character was leaning left in the copied pose, the mirrored pose would have the character leaning right), click on the "Paste Flipped" button. Note that if the armature was not set up correctly, the paste flipped technique may not work as expected.

Action Window

Introduction

An action is made of one or more action channels. Each channel corresponds to one

of the bones in the armature, and each channel has an Action Ipo associated with it. The action window provides a means to visualize and edit all of the Ipos associated with the action.

Tip: You can activate the action window with **SHIFT+F12**.

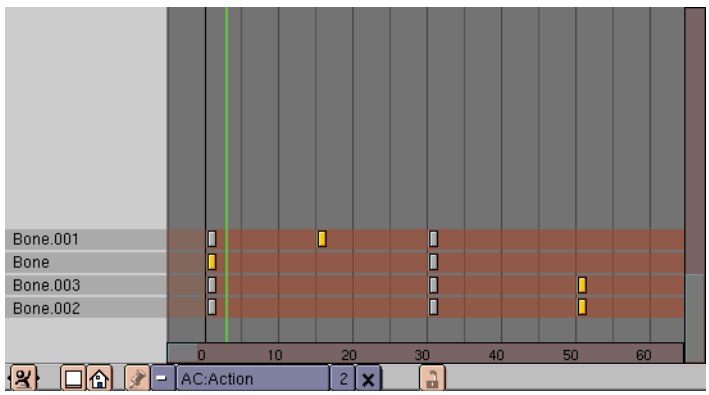
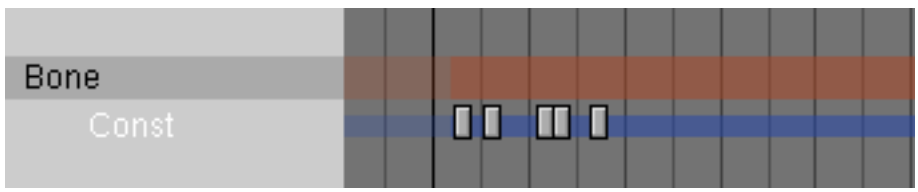


Figure 13-7. ActionWindow

For every key set in a given action ipo, a marker will be displayed at the appropriate frame in the action window. This is similar to the "Key" mode in the ipo window.For action channels with constraint ipos, the will be one or more additional constraint channels beneath each action channel. These channels can be selected independently of their owner channels.



Moving Action Keys

A block of action keys can be selected by either **RMB** on them or by using the boundary select tool (**BKEY**). Selected keys are highlighted in yellow. Once selected, the keys can be moved by pressing **GKEY>** and moving the mouse. Holding **CTRL** will lock the movement to whole-frame intervals. **LMB** will finalize the new location of the keys.

Scaling Action Keys

A block of action keys can be scaled horizontally (effectively speeding-up or slowing-down the action) by selecting number of keys and pressing **SKEY**. Moving the mouse horizontally will scale the block. **LMB** will finalize the operation.

Deleting Action Keys

Delete one or more selected action keys by pressing **XKEY** when the mouse cursor is over the keyframe area of the action window.

Duplicating Action Keys

A block of action keys can be duplicated and moved within the same action by selecting the desired keys and pressing **SHIFT+D**. This will immediately enter grab mode so that the new block of keys can be moved. Subsequently **LMB** will finalize the location of the new keys.

Deleting Channels

Delete one or more entire action or constraint channels (and all associated keys) by selecting the channels in the left-most portion of the action window (the selected channels will be highlighted in blue). With the mouse still over the left-hand portion of the window, press **XKEY** and confirm the deletion. Note that there is no undo so perform this operation with care. Also note that deleting an action channel that contains constraint channels will delete those constraint channels as well.

Baking Actions

If you have an animation that involves constraints and you would like to use it in the game engine (which does not evaluate constraints), you can bake the action by pressing the BAKE button in the Action Window headerbar. This will create a new action in which every frame is a keyframe. This action can be played in the game engine and should display correctly with all constraints removed. For best results, make sure that all constraint targets are located within the same armature.

Action IPO

The action ipo is a special ipo type that is only applicable to bones. Instead of using Euler angles to encode rotation, action ipos use quaternions, which provide better interpolation between poses.

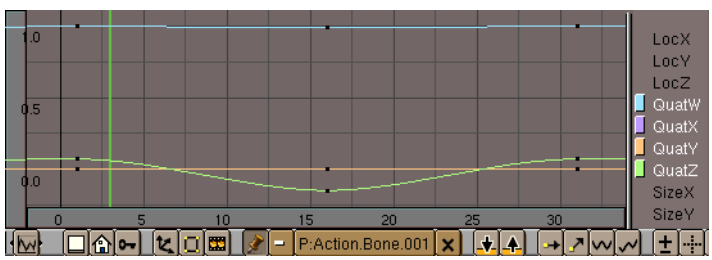


Figure 13-8. ActionIpo

Quaternions

Instead of using a three-component Euler angle, quaternions use a four-component vector. It is generally difficult to describe the relationships of these quaternion chan-

nels to the resulting orientation, but it is often not necessary. It is best to generate quaternion keyframes by manipulating the bones directly, only editing the specific curves to adjust lead-in and lead-out transitions.

Action Actuator

The action actuator provides an interface for controlling action playback in the game engine. Action actuators can only be created on armature objects.

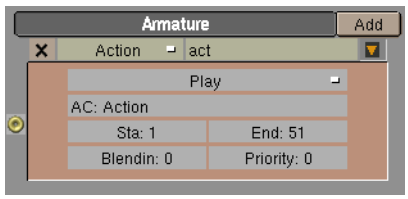


Figure 13-9. Action Actuator

Play Modes

Play

Once triggered, the action will play all the way to the end, regardless of other signals is receives.

Flipper

When it receives a positive signal, the action will play to the end. When it no longer receives a positive signal it will play from its current frame back to the start.

Loop Stop

Once triggered, the action will loop so long as it does not receive a negative signal. When it does receive a negative signal it will stop immediately.

Loop End

Once triggered, the action will loop so long as it does not receive a negative signal. When it does receive a negative signal it will stop only once it has reached the end of the loop.

Property

The action will display the frame specified in the property field. Will only update when it receives a positive pulse.

Blending

By editing the "Blendin" field you can request that Blender generates smooth transitions between actions. Blender will create a transition that lasts as many frames as the number specified in the Blendin field.

Priority

In situations where two action actuators are active on the same frame and they specify conflicting poses, the priority field can be used to resolve the conflict. The action with the lowest numbered priority will override actions with higher numbers. So priority "0" actions will override all others. This field is only important when two actions overlap.

Overlapping Actions

It is now possible to have two non-conflicting action actuators play simultaneously for the same object. For example, one action could specify the basic movements of the body, while a second action could be used to drive facial animation. To make this work correctly, you should ensure that the two actions do not have any action channels in common. In the facial animation example, the body movement action should not contain channels for the eyes and mouth. The facial animation action should not contain channels for the arms and legs, etc.

Python

The following methods are available when scripting the action actuator from python.

getAction()

Returns a string containing the name of action currently associated with this actuator.

getBlendin()

Returns a floating-point number indicating the number of blending frames currently specified for this actuator.

getEnd()

Returns a floating-point number specifying the last frame of the action.

getFrame()

Returns a floating-point number indicating the current frame of playback.

getPriority()

Returns an integer specifying the current priority of this actuator.

getProperty()

Returns a string indicating the name of the property to be used for "Property-Driven Playback".

getStart()

Returns a floating-point number specifying the first frame of the action.

setAction(action, reset)

Expects a string action specifying the name of the action to be associated with this actuator. If the action does not exist in the file, the state of the actuator is not changed.

If the optional parameter reset is set to 1, this method will reset the blending timer to 0. If the reset is set to 0 this method leaves the blending timer alone. If reset is not specified, the blending timer will be automatically reset. Calling this method does not however, change the start and end frames of the action. These may need to be set using setStart and setEnd

setBlendin(blendin)

Expects a positive floating-point number blendin specifying the number of transition frames to generate when switching to this action.

setBlendtime(blendtime)

Expects a floating-point number blendtime in the range between 0.0 and 1.0. This can be used to directly manipulate the internal timer that is used when generating transitions. Setting a blendtime of 0.0 means that the result pose will be 100% based on the last known frame of animation. Setting a value of 1.0 means that the pose will be 100% based on the new action.

setChannel(channelname, matrix)

Accepts a string channelname specifying the name of a valid action channel or bone name, and a 4x4 matrix (a list of four lists of four floats each) specifying an overriding transformation matrix for that bone. Note that the transformations are in local bone space (i.e. the matrix is an offset from the bone's rest position).

This function will override the data contained in the action (if any) for one frame only. On the subsequent frame, the action will revert to its normal course, unless the channel name passed to setChannel is not specified in the action. If you wish to override the action for more than one frame, this method must be called on each frame.

Note that the override specified in this method will take priority over all other actuators.

setEnd(end)

Accepts a floating-point number end, which specifies what the last frame of the action should be.

setFrame(frame)

Passing a floating-point number frame allows the script to directly manipulate the actuator's current frame. This is low-level functionality for advanced use only. The preferred method is to use Property-Driven Playback mode.

setPriority(priority)

Passing an integer priority allows the script to set the priority for this actuator. Actuators with lower priority values will override actuators with higher numbers.

setProperty(propertyname)

This method accepts a string propertyname and uses it to specify the property used for Property-Driven-Playback. Note that if the actuator is not set to use Property-Playback, setting this value will have no effect.

setStart(start)

To specify the starting frame of the action, pass a floating-point number start to this method.

NLAWindow (Non Linear Animation)

Introduction

This window gives an overview of all of the animation in your scene. From here you can edit the timing of ALL ipos, as if they were in the action window. Much of the editing functionality is the same as the Action window.

You can display the NLAWindow with **CTRL+SHIFT+F12**.

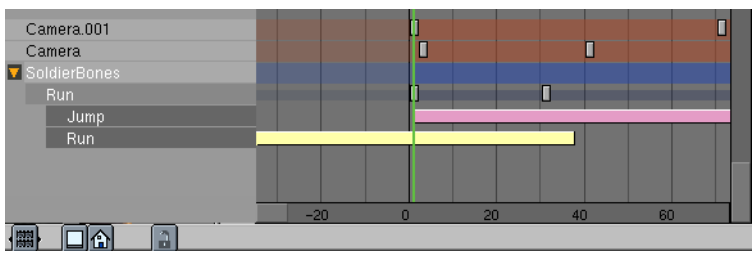


Figure 13-10. NLAWindow

You can also use this window to perform action blending and other Non-Linear Animation tasks. You add and move action strips in a fashion similar to the sequence editor, and generate blending transitions for them.

In the NLA window actions are displayed as a single strip below the object's strip; all of the keyframes of the action (constraint channel keyframes included) are displayed on one line.

To see an expanded view of the action, use the Action window.

Objects with constraint channels will display one or more additional constraint strips below the object strip. The constraint strip can be selected independently of its owner object.

RMB clicking on object names in the NLA window will select the appropriate objects in the 3dWindow. Selected object strips are drawn in blue, while unselected ones are red.

You can remove constraint channels from objects by clicking **RMB** on the constraint channel name and pressing **XKEY**.

Note: Note that only armatures, or objects with ipos will appear in the NLA window.

Working with Action Strips

Action strips can only be added to Armature objects. The object does not necessarily need to have an action associated with it first.

Add an action strip to an object by moving the mouse cursor over the object name in the NLA window and pressing **SHIFT+A** and choosing the appropriate Action to add from the popup menu. Note that you can only have one action strip per line.

You can select, move and delete action strips along with other keyframes in the NLA window.

The strips are evaluated top to bottom. Channels specified in strips later in the list override channels specified in earlier strips.

You can still create animation on the armature itself. Channels in the local action on the armature override channels in the strips. Note that once you have created a channel in the local action, it will always override all actions. If you want to create an override for only part of the timeline, you can convert the local action to an action strip by pressing **CKEY** with your mouse over the armature's name in the NLA window. This removes the action from the armature and puts it at the end of the action strip list.

Action Strip Options

Each strip has several options which can be accessed by selecting the strip and pressing **NKEY**. The options available are as follows:

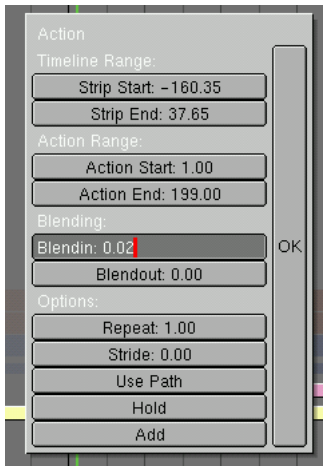


Figure 13-11. Action Strip Options

StripStart/StripEnd

The first and last fame of the action strip in the timeline

ActionStart/ActionEnd

The range of keys to read from the action. The end may be less than the start which will cause the action to play backwards.

Blendin/Blendout

The number of frames of transition to generate between this action and the one before it in the action strip list.

Repeat

The number of times the action range should repeat. Not compatible with "USE PATH" setting.

Stride

The distance (in Blender units) that the character moves in a single cycle of the action (usually a walk cycle action). This field is only needed if "USE PATH" is specified.

Use Path

If an armature is the child of a path or curve and has a STRIDE value, this button will choose the frame of animation to display based on the object's position along the path. Great for walkcycles.

Hold

If this is enabled, the last frame of the action will be displayed forever, unless it is overridden by another action. Otherwise the armature will revert to its rest position.

Add

Specifies that the transformations in this strip should ADD to any existing animation data, instead of overwriting it.

Constraints

Constraints are filters that are applied to the transformations of bones and objects. These constraints can provide a variety of services including tracking and IK solving.

Constraint Evaluation Rules

Constraints can be applied to objects or bones. In the case of constraints applied to bones, any constraints on the armature OBJECT will be evaluated before the constraints on the bones are considered.

When a specific constraint is evaluated, all of its dependencies will have already been evaluated and will be in their final orientation/positions. Examples of dependencies are the object's parent, its parent's parents (if any) and the hierarchies of any targets specified in the constraint.

Within a given object, constraints are executed from top to bottom. Constraints that occur lower in the list may override the effects of constraints higher in the list. Each constraint receives as input the results of the previous constraint. The input to the first constraint in the list is the output of the ipos associated with the object.

If several constraints of the same type are specified in a contiguous block, the constraint will be evaluated ONCE for the entire block, using an average of all the targets. In this way you can constrain an object to track to the point between two other objects, for example. You can use a NULL constraint to insert a break in a constraint block if you would prefer each constraint to be evaluated individually.

Looping constraints are not allowed. If a loop is detected, all of the constraints involved will be temporarily disabled (and highlighted in red). Once the conflict has been resolved, the constraints will automatically re-activate.

Influence

The influence slider next to each constraint is used to determine how much effect the constraint has on the transformation of the object.

If there is only a single constraint in a block (a block is a series of constraints of the same type which directly follow one another), an influence value of 0.0 means the constraint has no effect on the object. An influence of 1.0 means the constraint has full effect.

If there are several constraints in a block, the influence values are used as ratios. So in this case if there are two constraints, A and B, each with an influence of 0.1, the resulting target will be in the center of the two target objects (a ratio of 0.1:0.1 or 1:1 or 50% for each target).

Influence can be controlled with an ipo. To add a constraint ipo for a constraint, open an ipo window and change its type to constraint by clicking on the appropriate icon.

Next click on the Edit Ipo button next to the constraint you wish to work with. If there is no constraint ipo associated with the constraint yet, one will be created. Otherwise the previously assigned ipo will be displayed. At the moment, keyframes for constraint ipos can only be created and edited in the ipo window, by selecting the INF channel and **CTRL+LEFTMOUSE** in the ipo space.

When blending actions with constraint ipos, note that only the ipos on the armature's local action ipos are considered. Constraint ipos on the actions in the motion strips are ignored.

> **Important:** In the case of armatures, the constraints ipos are stored in the current Action. This means that changing the action will change the constraint ipos as well.

Creating Constraints

To add a constraint to an object, ensure you are in object mode and that the object is selected. Switch to the constraint buttons window (the icon looks like a pair of chain links) and click on the "Add" button.

A new constraint will appear. It can be deleted by clicking on the "X" icon next to it. A constraint can be collapsed by clicking on its orange triangle icon. When collapsed, a constraint can be moved up or down in the constraint list by clicking on it at choosing "Move Up" or "Move Down" from the popup menu.

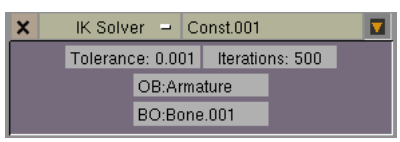
For most constraints, a target must be specified in the appropriate field. In this field you must type in the name of the desired target object. If the desired target is a bone, first type in the name of the bone's armature. Another box will appear allowing you to specify the name of the bone.

Adding Constraints to Bones

To add a constraint to a bone, you must be in pose mode and have the bone selected.

Constraint Types

IK Solver



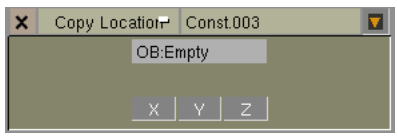
To simplify animation of multi-segmented limbs (such as arms and legs) you can add an IK solver constraint. IK constraints can only be added to bones. Once a target is

specified, the solver will attempt to move the ROOT of the constraint-owning bone to the target, by re-orienting the bone's parents (but it will not move the root of the chain). If a solution is not possible, the solver will attempt to get as close as possible. Note that this constraint will override the orientations on any of the IK bone's parents.

Copy Rotation

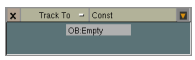
This constraint copies the global transformation of the target and applies it to the constraint owner.

Copy Location



The constraint copies one or more axes of location from the target to the constraint owner.

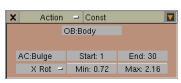
Track To



This constraint causes the constraint owner to point its Y-axis towards the target. The Z-axis will be oriented according to the setting in the anim-buttons window. By default, the Z-axis will be rolled to point upwards.

Action

An action constraint can be used to apply an action channel from a different action to a bone, based on the rotation of another bone or object. The typical way to use this is to make a muscle bone bulge as a joint is rotated. This constraint should be applied to the bone that will actually do the bulging; the target should point to the joint that is being rotated.



The AC field contains the name of the action that contains the flexing animation. The only channel that is required in this action is the one that contains the bulge animation for the bone that owns this constraint

The Start and End fields specify the range of motion from the action.

The Min and Max fields specify the range of rotation from the target bone. The action between the start and end fields is mapped to this rotation (so if the bone rotation is at the Min point, the pose specified at Start will be applied to the bone). Note that the Min field may be higher than the Max.

The pulldown menu specifies which component of the rotation to focus on.

Null

This is a constraint that does nothing at all; it doesn't affect the object's transformation directly. The purpose of a null constraint is to use it as a separator. Remember that if several constraints of the same type follow one another, the actual constraint operation is only evaluated once using a target that is an average of all of the constraints' targets. By inserting a null constraint between two similarly-typed constraints, you can force the constraint evaluator to consider each constraint individually. This is normally only interesting if one or more of the constraints involved have an Influence value of less than 1.0.

Rigging a Hand and a Foot

From the original tutorials written by Lyubomir Kovachev

The Hand

Setting up a hand for animation is a tricky thing. The gestures, the movements of wrists and finders are very important, they express emotional states of the character and interact with other characters and objects. That's why it's very important to have an efficient hand setup, capable of doing all the wrist and fingers motions easily. Here is how I do it:

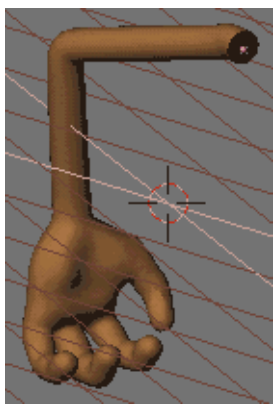


Figure 13-12. The Arm model

We'll use a simple cartoony arm mesh in this tutorial (Figure 13-12).

The following setup uses one IK solver for the movemet of the whole arm and 4 other IK solvers for each finger. The rotation of the wrist is achieved by a simple FK bone.

OK. Take a look at the arm mesh and let's start making the armature.

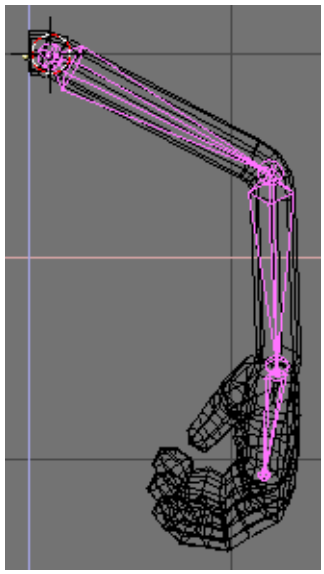


Figure 13-13. Drawing the armature

Position the 3D cursor in the shoulder, go to front view and add an armature. Make a chain of 3 bones - one in the upper arm, the second one in the lower arm and the third one should fit the palm, ending at the beginning of the middle finger. This is called an chain of bones. (Figure 13-13).

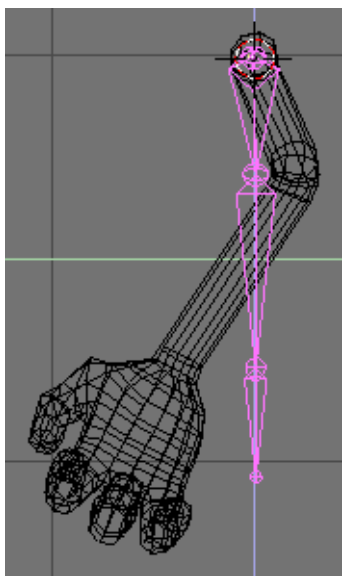


Figure 13-14. Placing the armature in side view.

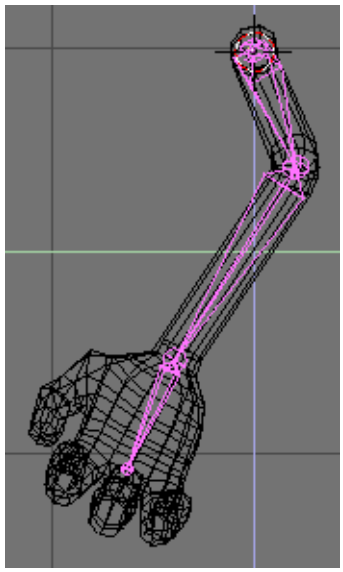


Figure 13-15. Placing the armature in side view.

Now change the view to side view and deit the bones so that they fit in the arm and palm properly (Figure 13-14 (Figure 13-15).

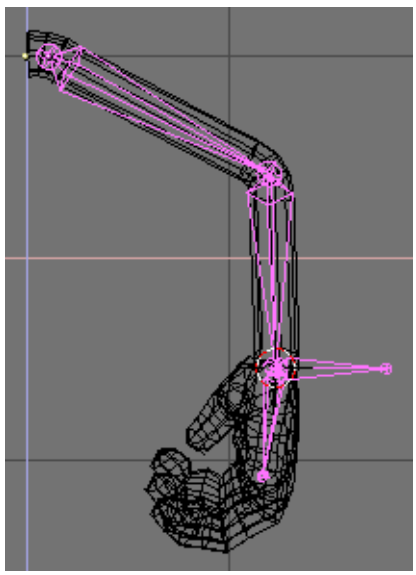


Figure 13-16. Wrist IK solver.

Zoom in the hand and position the cursor at the root of the bone, positioned in the palm. Add a new bone, pointing right, with the same lenght as the palm bone. This will be the IK solver for the arm. (Figure 13-16).

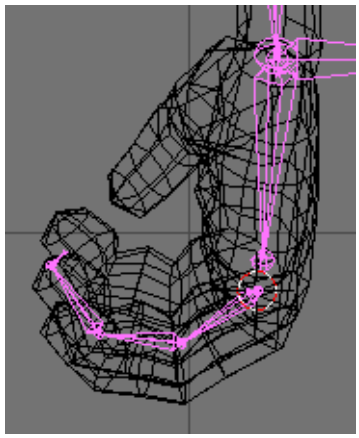


Figure 13-17. Rigging the finger.

Position the 3D cursor at the beginning of the middle finger and in fron view start building a new chain, consisting of 4 bones (Figure 13-17). 3 of them will be the actual bones in the finger, and the fourth bone will be a null bone - this is a small bone, pointing to the palm, that will help turning the whole chain to an IK chain later.

Again, change to side view and re-chape the bones so that they fit the finger well. It could be a tricky part and you may also view the scene using the trackball while reshaping the bones (Figure 13-18).

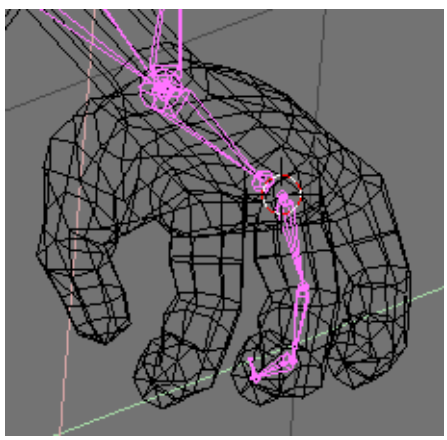


Figure 13-18. Rigging the finger.

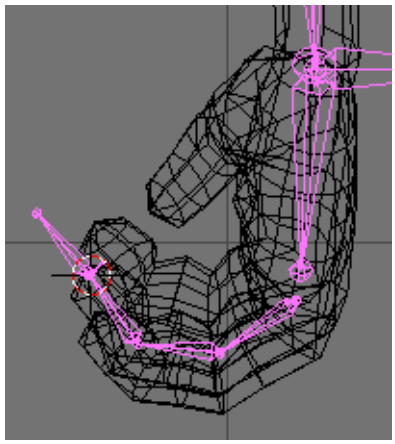


Figure 13-19. Adding the finger IK solver.

Now add the IK solver for this finger chain. Position the 3D cursor at the beginning of the null bone and add bone with the lenght of the other three bones in the finger (Figure 13-19).

Figure 13-20. Rigging the other fingers.

Repeat the same for the creation of the IK chains for the other three fingers. The only difference with the thumb is that it has two actual bones, instead of three. You can just copy and paste the chain and just reshape, reshape, reshape... (Figure 13-20).

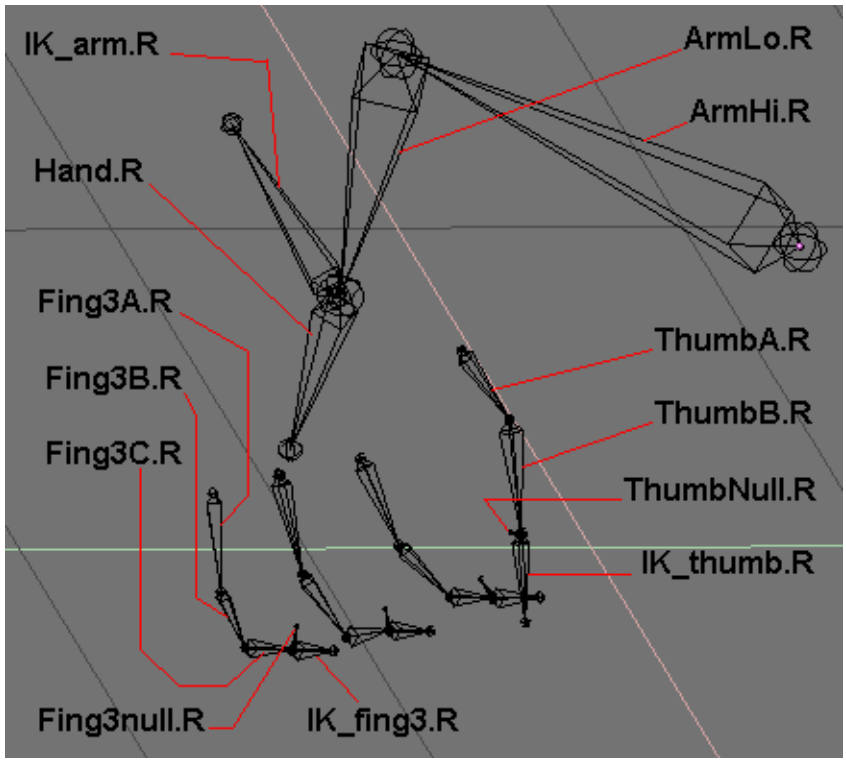


Figure 13-21. Naming overview.

The time has come for the boring part - naming of the bones. You cannot skip this, because you'll need the bone names in the skinning part later. Bones are named as in Figure 13-21.

Note: The names of the bones of finger 1 and finger 2 are not shown here. They are identical to the names of the bones of finger 3, only the number changes.

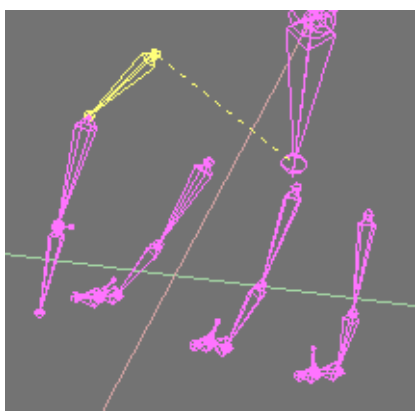


Figure 13-22. Parenting the Thumb.

Now let's do some parenting.

Select the root thumb bone "ThumbA.R" (Figure 13-22) and in the edit menu click in the "child of" field and choose "Hand.R". You've just parented the thumb bone chain to the hnad bone.

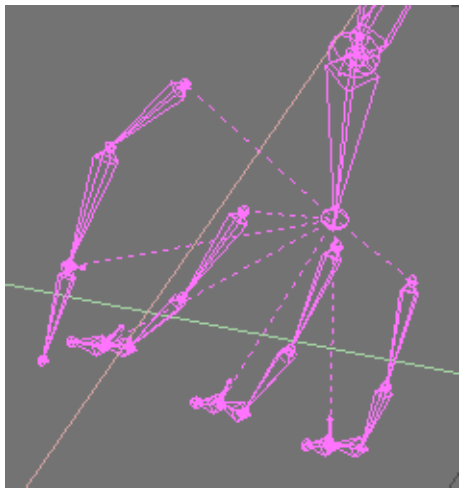


Figure 13-23. Parenting the other fingers.

By repeating the same process parent the following bones (Figure 13-23):

- "Fing1A.R" to "Hand.R"
- "Fing2A.R" to "Hand.R"
- "Fing3A.R" to "Hand.R"
- "IK_thumb.R" to "Hand.R"
- "IK_fing1.R" to "Hand.R"
- "IK_fing2.R" to "Hand.R"
- "IK_fing3.R" to "Hand.R"

Why did we do all this? Why did we parent so much bones to "Hand.R"? Because when you rotate the hand (i.e. "Hand.R") all the fingers will follow the hand. Otherwise the fingers will stay still and only the palm will move and you'll get very weird result.

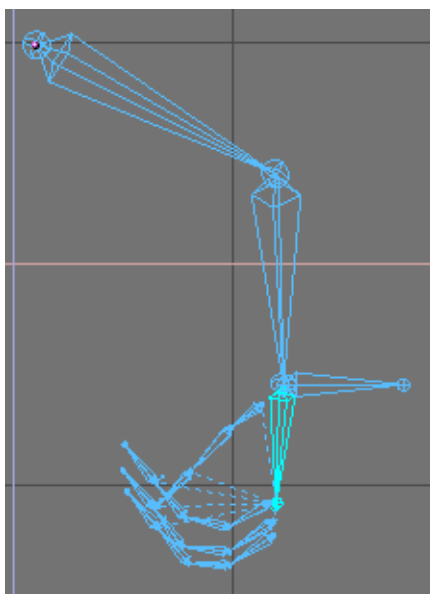


Figure 13-24. Seting the IK solver for the wrist. Selecting the bone.

Time to add constraints. Enter pose mode (Figure 13-24) and open the "Constraints" menu. Choose "Hand.R" and add an IK solver constraint. In the "OB" field type the object name: Armature. The bone went to the center of the armature, but we'll fix this now. In the new "BO" field, that appeared in the constraint window, type the bone name "IK_arm.R". This will be the IK solver bone controlling the arm motion (Figure 13-25).

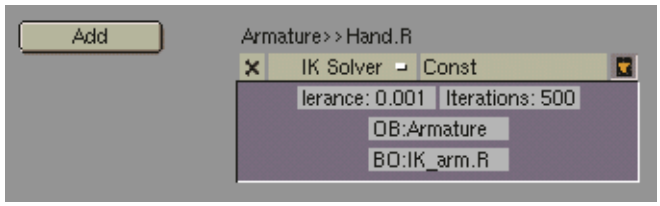


Figure 13-25. Seting the IK solver for the wrist. Setting the Constrain.

Now by repeating the same:

- select "ThumbNull.R" and add IK solver "IK_thumb.R",
- select "Fing1null.R" and add IK solver "IK_fing1.R",
- select "Fing2null.R" and add IK solver "IK_fing2.R",
- select "Fing3null.R" and add IK solver "IK_fing3.R".

You're finished with the bone part. In pose mode select different IK solvers and move them to test the IK chains. Now you can move the fingers, the thumb, the whole arm and by rotating the "Hand.R" bone you can rotate the whole hand.

So let's do the skinning now. It's the part when you tell the mesh how to deform. You'll add vertex groups to the mesh. Each vertex group should be named after the bone that will deform it. If you don't assign vertex groups, the deformation process will need much more CPU power, the animation process will be dramatically slowed down and you'll get weird results. It's highly recommended (almost mandatory) that you use subdivision surfaces meshes for your characters with low vertex count. Otherwise if you use meshes with lots of vertices, the skinning will be much more difficult. Don't sacrifice detail, but model economically, use as less vertices as possible and always use SubSurf.

Parent the Mesh to the Armature, int the Pop-Up select `Armature` and in the following select `Name Groups`. Youre Mesh will be enriched by empty Vertex Groups.

Select the arm mesh, enter edit mode and open the edit buttons window. Notice the small group of buttons with the word "Group" on top. Thanx to the automatic naming feature you have already all the groups you needed created. (Figure 13-26).

Figure 13-26. Vertex group names.

Actually the automatic Grouping scheme has created vertex groups also for the "IK" and "null" bones. These are useless and you can safely delete. Otherwise you can ask Blender not to create Groups at all and create them by yourself before skinning.

Now let's do the tricky part: Select the vertex group "ArmHi.R" from the edit buttons by clicking on the small button with the white minus sign. Now look at the 3D window. Select all the vertices that you want to be deformed by the "ArmHi.R" bone. (Figure 13-27).

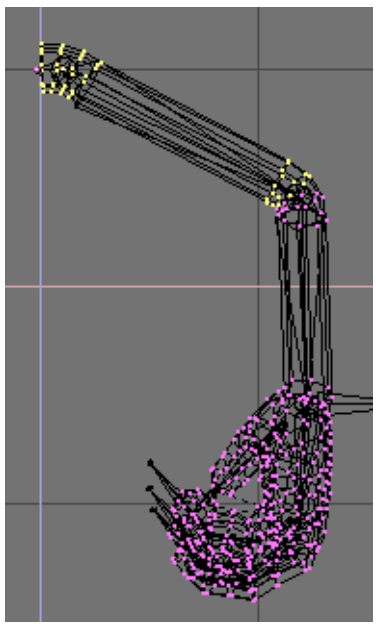


Figure 13-27. ArmHi.R vertex group.

Now press the "Assign" button in the edit buttons window (Figure 13-28). You've just added the selected vertices to the "ArmHi.R" vertex group. These vertices will be deformed by the "ArmHi.R" bone.

Figure 13-28. Assigning vertices to a group.

Repeat the same steps for the other vertex groups: select vertices and assign them to the corresponding group. This is a tricky process. Do it carefully. If you've assigned some vertices to a certain group by mistake, don't worry. Just select the unneeded vertices and press the "Remove" button. You can add a vertex to more than one vertex group. For example the vertices that build joints (of fingers, wrist, elbow, etc.) could be assigned to the two vertex groups that are situated close to it. You can also assign vertices to deform with different strength. The default stregth is 1.000, but you can add vertices with strength 0.500 or less. The l ower the strength value, the less deformation for that vertex. You can make a vertex deform 75% by one bone and 25% by another, or 50% by one and 50% by another. It's all a matter of testing the deformation until you achieve the result you want. In general if your arm model has half-flexed joints (as the model in this tutorial you will get good results without using strenght values different than 1.000. My own rule of thumb when modelling a character is: always model the arms fingers and legs half-flexed, not straight. This is a guarantee for good deformation.

When you're finished adding vertices to vertex groups, exit edit mode and parent the arm to the armature. If you haven't made any mistakes now you'll have a well set up arm with a hand. Select the armature, enter pose mode, select the different IK solvers and test the arm and fingers (Figure 13-29).

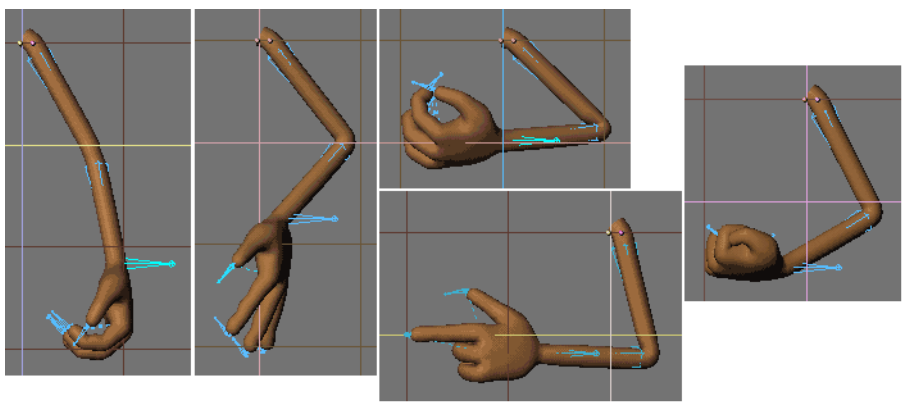


Figure 13-29. Assigning vertices to a group.

The Foot

The setup of legs and feet is maybe the most important thing in the whole rigging process. Bad foot setup may lead to the well known "sliding-feet" effect, which is very annoying and usually ruins the whole animation. A well made complex foot setup must be capable of standing still on the ground while moving the body, and doing other tricky stuff like standing on tiptoe, moving the toes, etc. Now we're going to discuss several different foot setups that can be used for different purposes.

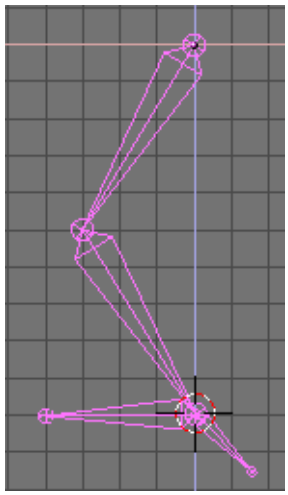


Figure 13-30. A (wrong) leg rig.

First let's see how a bad foot setup looks like (Figure 13-30).

Start building a bone chain of three bones - one for the upper leg, the second one for the lower leg and the third one for foot. Now move the 3D cursor at the heel joint and add another bone - this will be the IK solver. Now add that bone as an IK solver constraint to the foot bone. (Figure 13-31).

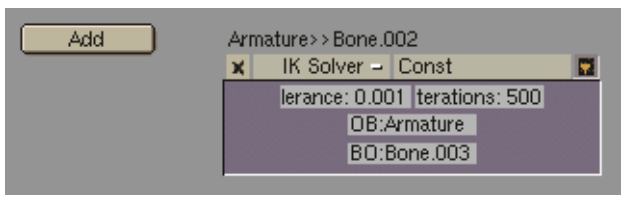


Figure 13-31. Assigning the IK constraint.

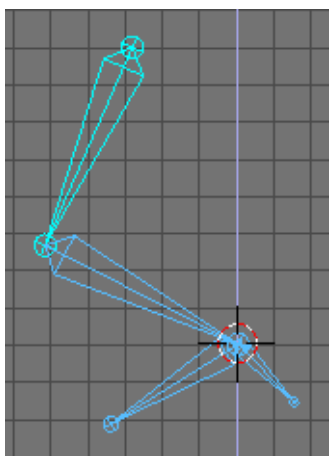


Figure 13-32. THe rig in pose mode.

Test the armature: in pose mode grab the IK solver and move it - it's moving OK. Now grab the first bone in the chain (the upper leg) and move it. The foot is moving too and we don't want this to happen! (Figure 13-32).

Usually in an animation you'll move the body a lot. The upper leg bone is parented to the body and it will be affected by it. So every time you make your character move or rotate his body, the feet will slide over the ground and go under it and over it. Especially in a walkcycle, this would lead to an awful result.

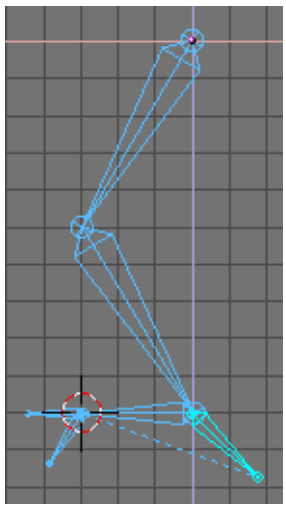


Figure 13-33. Adding a toe and some more IKA.

Now maybe you think this could be avoid by adding a second IK solver at the toes (Figure 13-33). Let's do it. Start a new armature. Add a chain of four bones: upper leg, lower leg, foot and toes. Add two IK solvers - one for the foot and one for the toes. Parent the toe IK solver bone to the foot IK solver bone.

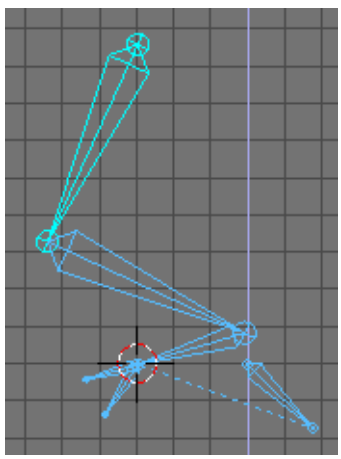


Figure 13-34. Moving the leg.

Test this setup - grab the upper leg bome and move it (Figure 13-34). Well, now the sliding isn't so much as in the previous setup, but it's enough to ruin the animation.

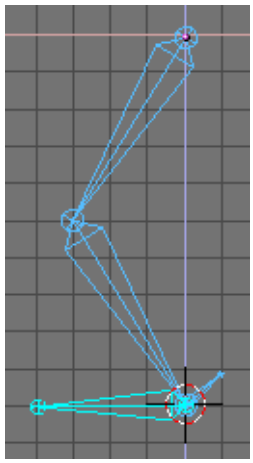


Figure 13-35. Rigging with a null bone.

Start a new armature. Make a chain if three bones - upper leg, lower leg and a null bone. The null bone is a small bone, that we'll add the IK solver to. Now position the 3D cursor at the heel and add the foot bone. Now add the foot bone as an IK solver constraint to the null bone (Figure 13-35).

(You can also add another bone as an IK solver and add a "copy location" constraint to the foot bone, with the IK solver as target bone.)

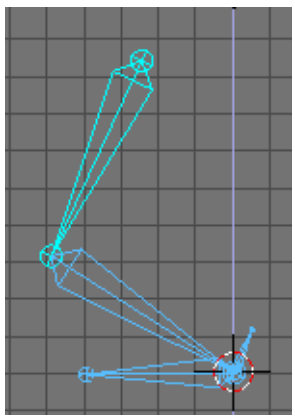


Figure 13-36. Rigging with a null bone.

Test this - now it works. When you move the upper leg the foot stands still (Figure 13-36). That's good. But still not enough. Move the upper leg up a bit more. The leg chain goes up, but the foot stays on the ground. Well, that's a shortcoming of this setup, but you're not supposed the raise the body so much and not move the IK solver up too during animation...

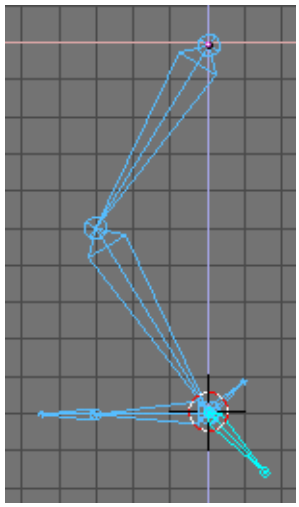


Figure 13-37. Adding the toe.

Again, build a chain of three bones - upper leg, lower leg and null bone. Position the 3D cursor at the heel and add a chain of two bones - the foot bone and the toes bone. Now add an IK solver to the foot bone (Figure 13-37).

Test it. This is a good setup with stable, isolated foot and moving toes. But you still cannot make standing on tiptoe with this setup.

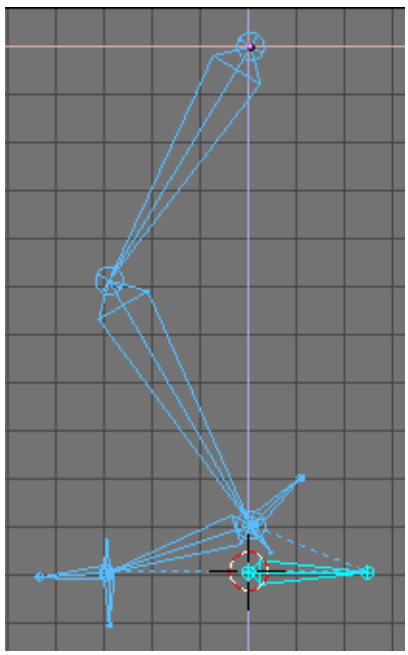


Figure 13-38. Full complete leg rig.

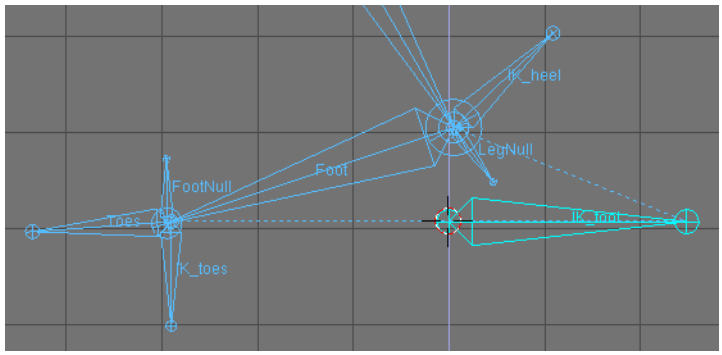


Figure 13-39. Zoom on the foot rig.

Build a chain of three bones - upper leg, lower leg and null bone (name it LegNull) (Figure 13-38). Starting at the heel point, make a second chain of two bones only - foot bone (Foot) and a small null bone (FootNull). Position the 3D cursor at the end of the foot bone and add the toes bone (Toes). From the same point create an IK solver bone (IK_toes). Now position the 3D cursor at the heel and add another IK solver there (IK_heel). Finally, starting somewhere near the heel, add a bigger IK solver (IK_foot) (Figure 13-39).

Now let's add the constraints. Do the following:

- To the bone "Toes" add a copy location contraint with target bone "IK_toes".
- To "FootNull" - an IK solver constraint (target - "IK_toes")
- To "Foot" - copy location (target - "LegNull").
- To "LegNull" - IK solver (target - "IK_heel")

Well, that's it. Now test the armature. Grab "IK_foot" and move it up. Now grab "IK_toes" and move it down. The foot changes it's rotation, but it looks like the toes are disconnected from it. But if you animate carefully you'll always manage to keep the toes from going away from the foot. Now return the armature to it's initial pose. grab "IK_heel" and "LegHi" and move them up. Now the character is standing on his tiptoes. The foot may appear disconnected from the toes again, but you can fix the pose by selecting "IK_heel" only and moving it a bit forward or backwards. This setup may not be the most easy one for animation, but gives you more possibilities that the previous setups. Usually when you don't need to make your character stand on tiptoe, you've better stick to some of the easier setups. You'll never make a perfect setup. You can just improve, but there will always be shortcomings. Feel free to experiment and if you invent something better, don't hesitate to drop an e-mail to: lpk3d@yahoo.com.

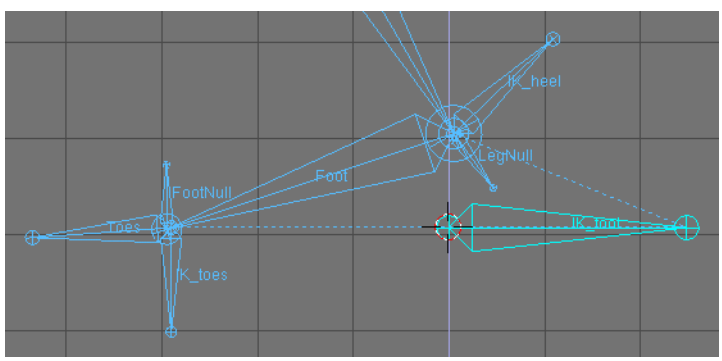


Figure 13-40. Testing the setup.

Rigging Mechanics

Armatures are great also for rigging mechanical stuff, like robots, warriorMechs etc (Figure 13-41).

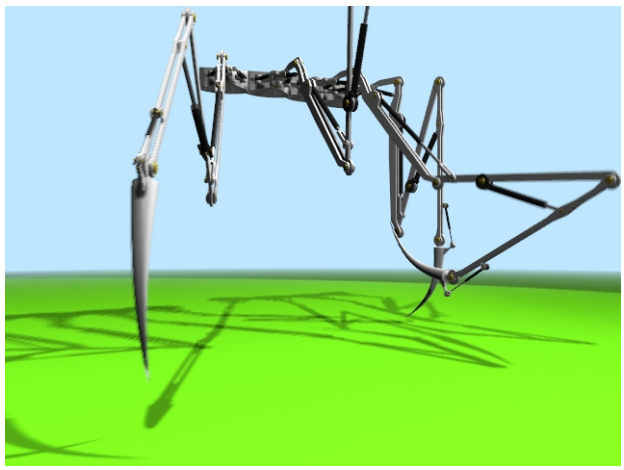


Figure 13-41. Four spider-mech legs.

First step is to create the mesh for the arms. We are not here for organic, we are here for mechanics. So no single mesh thing. The arm/leg/whatever is made of rigid parts, each part is a single mesh, parts moves/rotates one with respect to the other.

Although Figure 13-41 has four spider-like legs arms, each of which have 5 sections, it is clearer to explain the tricks with just a single joint arm.

My suggestion is this, the arm, on the left, made by two equal sections, and the forearm, on the right, made by just one section. Note the cylinders which represents the shoulder (left) the elbow (center) and the wrist (right) (Figure 13-42).

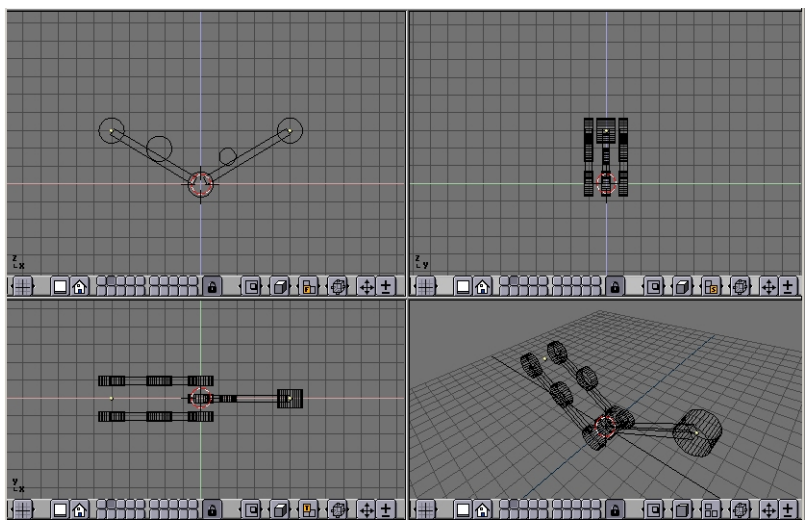


Figure 13-42. The Arm model

The other cylinders in the middle of the arm and forearm are the places where the piston will be linked to.

Note that it is much easier if the axis of mutual rotation (shoulder, elbow, etc.) are exactly on grid points. This is not necessary though, if you master well Blender Snap menu.

Pivot axis

Then add the mechanical axes in the pivot points. Theoretically you should add one at each joint and two for every piston. For the sake of simplicity here there are only the two axes for the piston, made with plain cylinders (Figure 13-43).

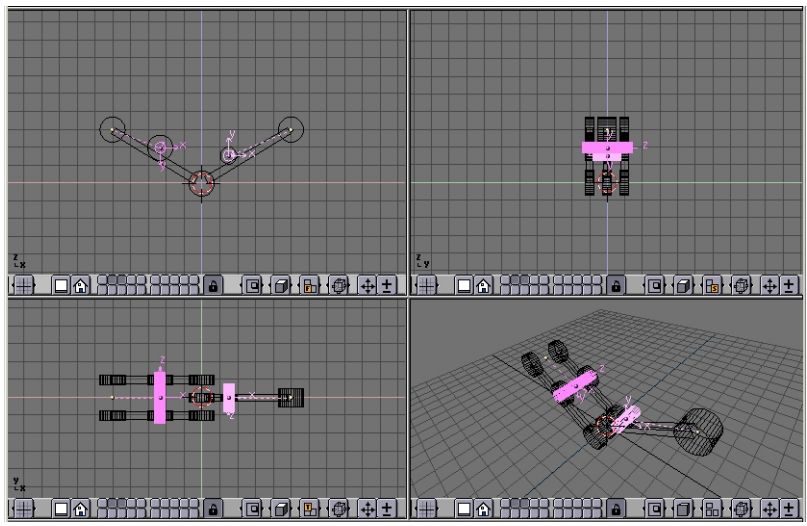


Figure 13-43. The Arm model with pivot axis.

Note two things:

- It is fundamental that the center of the mesh is exactly in the middle and exactly on the axis of rotation of the piston
- Each axis must be parented to the pertinent arm mesh.

The Armature

Now it is time to set up the armature. Just a two bones armature is enough (Figure 13-44).

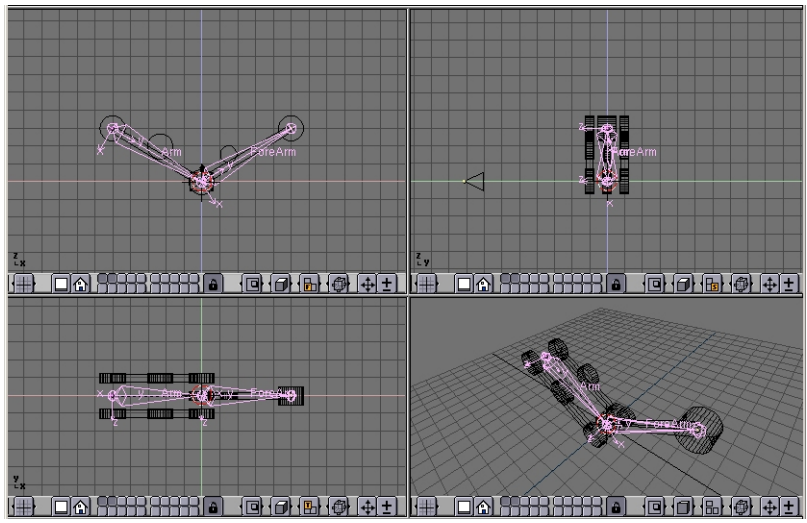


Figure 13-44. The Arm model and its armature

To have an accurate movement, the joints must be precisely set on the pivoting axis (this is why I told you to place such axes on grid points before, so that you can use the Move Selected To Grid feature)

Name the bones smartly (Arm and Forearm, for example). Parent the Arm Mesh to the armature, selecting the 'Bone' option and the Arm bone. Do the same with the forearm mesh and forearm bone.

> **Parent to Bone:** Parent to bone effectvely makes the Object follow the bone without any deformation. This is what should happen for a robot which is made by undeformable pieces of steel!

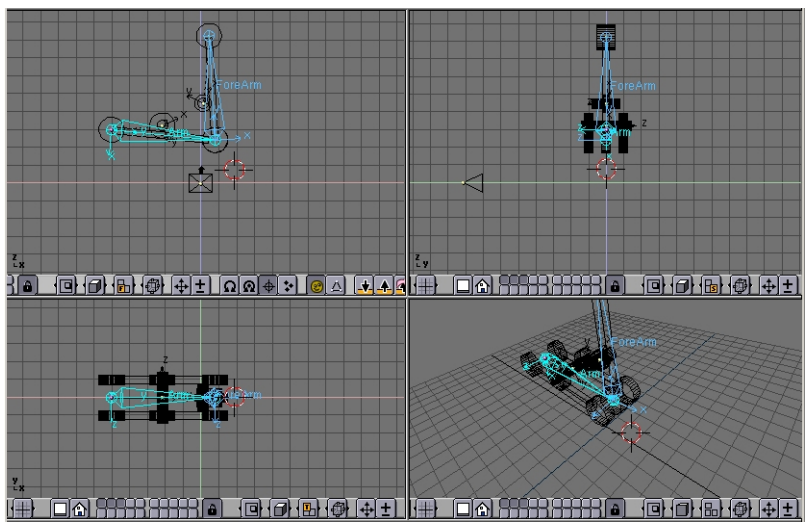


Figure 13-45. The Arm model in Pose Mode

If you switch to pose mode you can move your arm by rotating the bones. (Figure 13-45). You cann add an IKA solver as we did in the previous section if you like.

Hydraulics

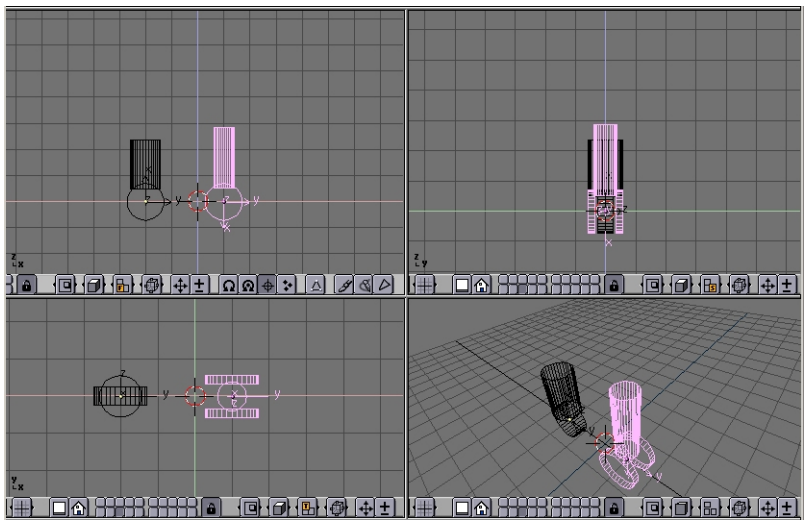


Figure 13-46. Hydraulic piston.

Make a piston with two cylinders, a larger one and a thinner one, with some sort of nice head for linking to the pivoting points (Figure 13-46).

It is is MANDATORY for the two pieces to have the mesh center exactly on the respective pivoting axis.

Place them in the correct position and parent each piston piece to the pertinent mesh representing the axis. (Figure 13-47).

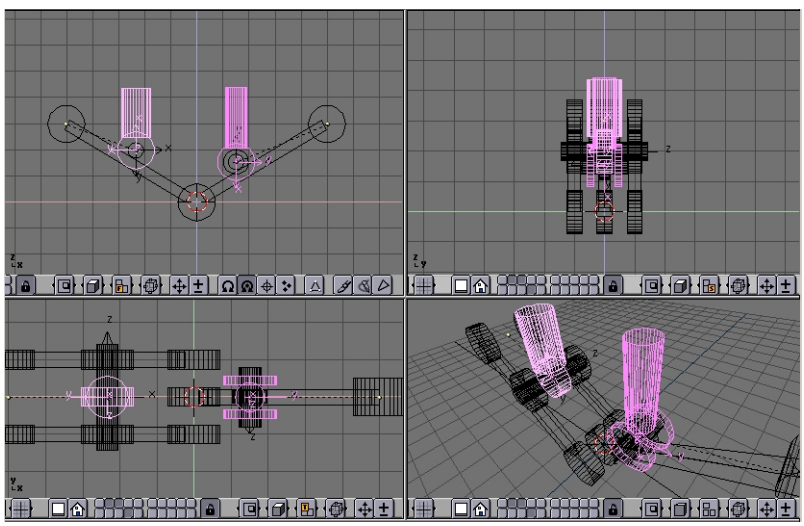


Figure 13-47. Hydraulic piston on the arm.

If you now rotate the two pieces in the position they should have to form a correct STILL image you get a nice piston. (Figure 13-48, left).

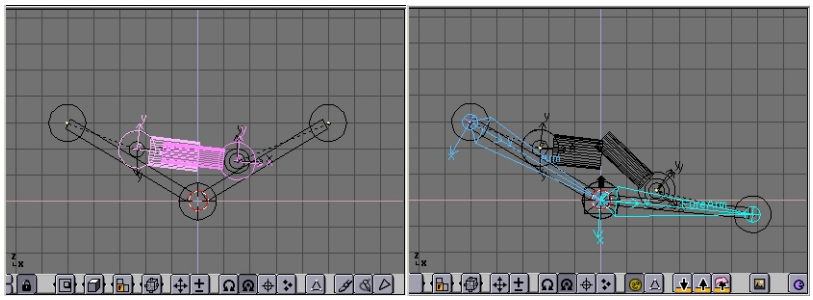


Figure 13-48. Hydraulic piston in pose mode.

But if you switch to pose mode and start moving the Arm/Forearm the piston gets screwed up... (Figure 13-48, right).

To make a working piston you must make each half pison track *the other half piston's pivot axis* This is why the position of all the mesh centers is so critical (Figure 13-49).

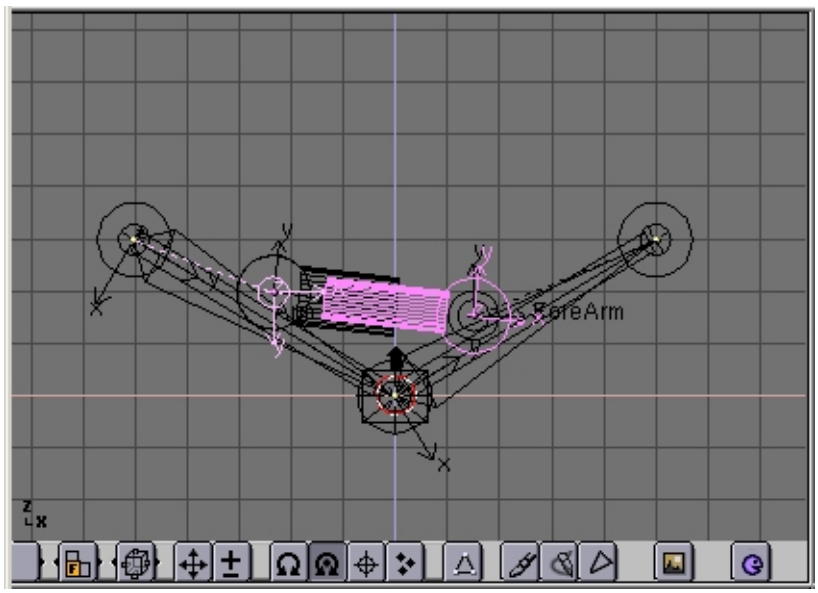


Figure 13-49. Hydraulic piston with mutual tracking.

Select half a piston, select the other half piston's axis mesh, press **CTRL-T**. Beware, this might bring to very funny results. You must experiment with the various track button in the Animation (**F7**) window. The buttons top left TrackX,Y... and pay attention to the axis of the meshes (Figure 13-50).

Figure 13-50. Track settings.

Remember also to press the 'PowerTrack' button for a nicer result (Figure 13-50).

Now, if you switch to pose mode and rotate your bones the piston will extend and contract nicely, as it should in reality. (Figure 13-51).

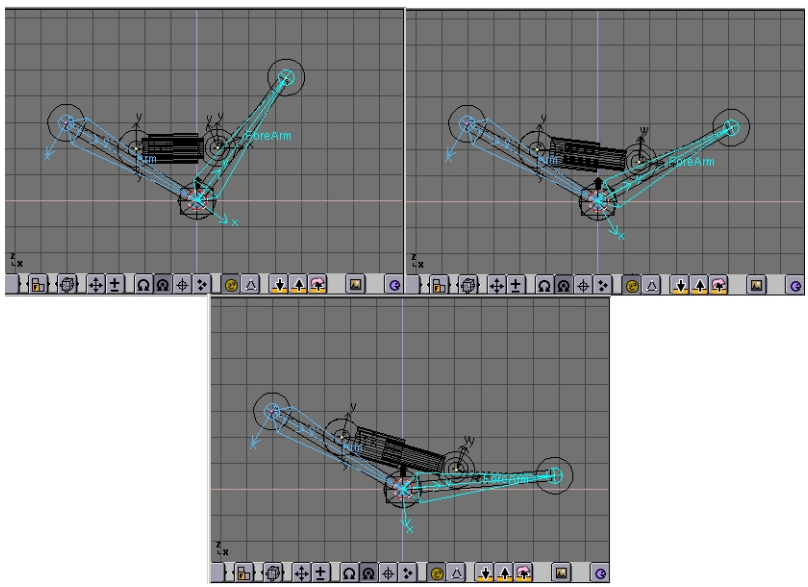


Figure 13-51. Pose Mode for the arm with idraulics.

Next issue now is, since pistons work with pressurized oil which is sent into them, for a really accurate model I should add some tubes. But how to place a nicely deforming tube going from arm to piston? The two ends should stick to two rigid bodies reciprocally rotating. This requires IKA!

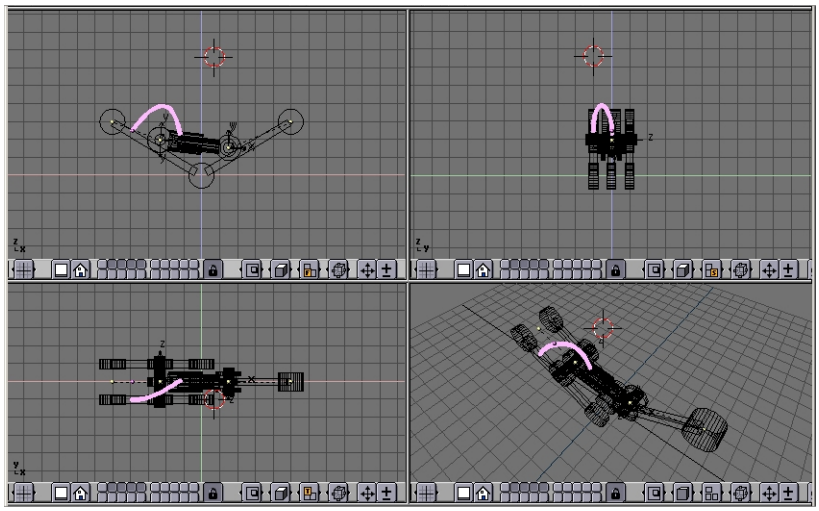


Figure 13-52. Adding a flexible tube.

First add a mesh in the shape of the tube you want to model (Figure 13-52).

Personally I prefer to draw the tube in its bent position as a beveled curve.

This is done by adding a Bezier curve, adding a Bezier circle, and using the Bezier circle as BevOb of the Bezier curve. Then convert that to a mesh **ALT-C** to be able to deform it with an armature.

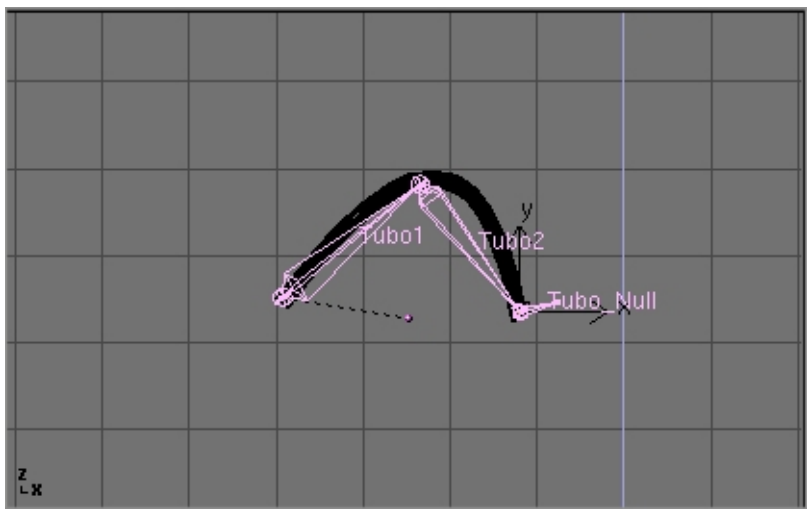


Figure 13-53. Adding the armature to the tube.

Then add an armature. A couple of bones are enough. This armature should go from the tube 'fixed' end to the tube 'mobile' end. Add a third bone which will be used for the Inverse Kinematic solution (Figure 13-53).

Be sure that the armature is parented to the object where the 'fixed' part of the tube is, well, fixed. In this case the robot arm. Add also an Empty at the 'mobile' end of the tube. (Figure 13-54).

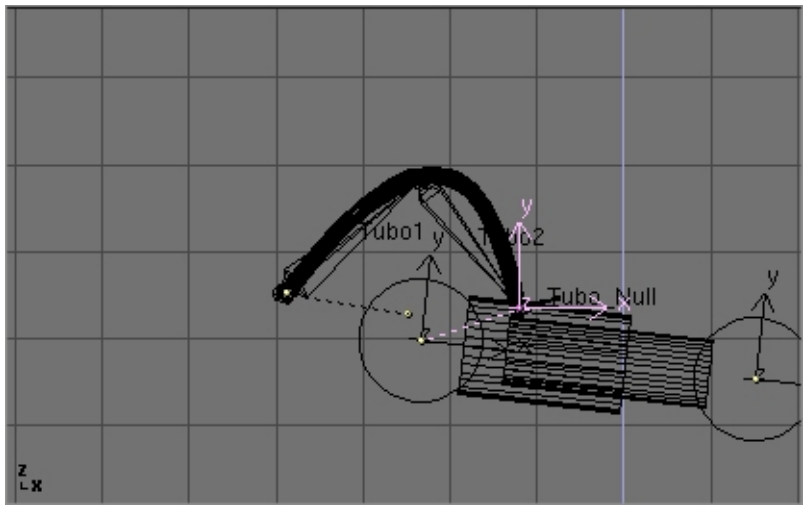


Figure 13-54. The Empty for the IKA solution.

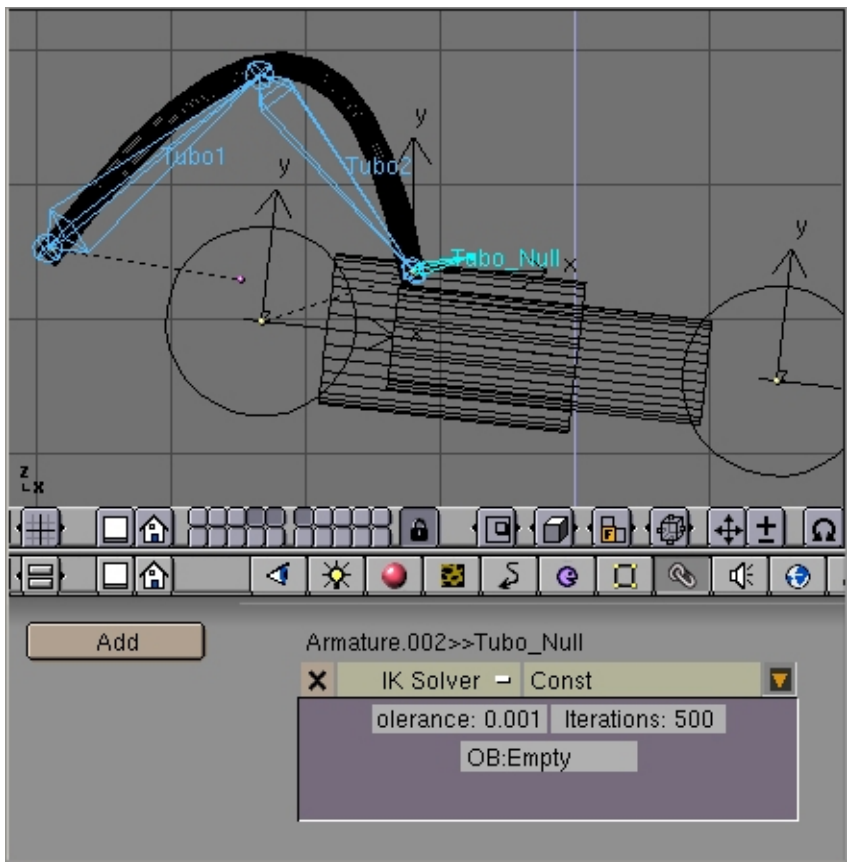


Figure 13-55. IKA constraint.

Parent the Empty to the 'mobile' part of the structure. In this case the outer part of the piston to which the tube is linked. In pose mode go to the 'Constrains' window (chain icon). Select the last bone, the one which starts from where the tube ends, and Add a constrain. Select the 'IK solver' type of constrains and Select the newely created Empty as target Object 'OB:'. (Figure 13-55). You can play with Tolerance and

Iterations if you like.

Lastly, parent the tube to the Armature via the 'Armature' option. Create Vertex groups if you like. Now if, in pose mode, you move the arm, the two parts of the piston keeps moving appropriately, and the Empty follows. This obliges the IKA Armature of the tube to move, to follow the Empty, and, consequently, the Tube to deform (Figure 13-55).

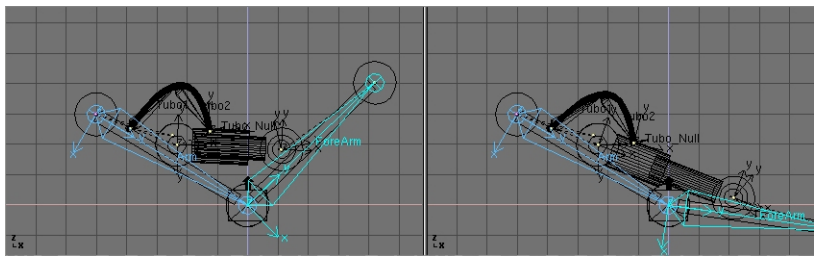


Figure 13-56. Full robot arm in pose mode.

How to setup a walkcycle using NLA

by Malefico

In this tutorial we will try to set up a walkcycle and use it with the PATH option in the Blender NLA Editor. Before starting let me tell you that you will need to have a basic knowledge of the animation tools, (armature set up), in order to follow the text, and have a lot of patience. It is highly recommendable to have read all the precedent NLA related part of the documentation.

We are going to use a character set up like the one explained in "Hand and Foot tutorial", that is with feet bone split up from the leg and using an extra null bone to store the IK solver constraint. For further details please check that chapter !

Before getting any deeper we'll need a character, an armature and a couple of actions for it. If you don't know where to find them just download this blend[1]. Remember: never go out for tutoring without a blend at hand ;-)

When you open the scene you'll find four windows: a 3D window with a character (criticism are not allowed), an action window, and IPO window and finally a NLA window. If you select the armature and check the action window, you'll see three actions defined: "WALKCYCLE", "WAVE_HAND" and "STAND_STILL". In WALKCYCLE and STAND_TILL there are keyframes for almost all control bones while in "WAVE_HAND" there are keyframes only for the arm and hand. This will allow our character to simultaneously wave its hand while walking.

The main idea behind this is to work on each single movement and later on combining everything in the NLA window.

"The path to success"

There are two main ways to animate a walkcycle, first one is to make the character actually advance through the poses of the cycle and the second one is to make the character walk "in situ" thus without real displacement.

The later option though is more difficult to set up, is the best choice for digital animation and it is our choice for this tutorial.

The whole walkcycle will be an "action" for our armature, so let's go an create a new action and switch to "pose mode" to get something like Pose 1 (the so called *"contact pose"*in Figure 13-57.

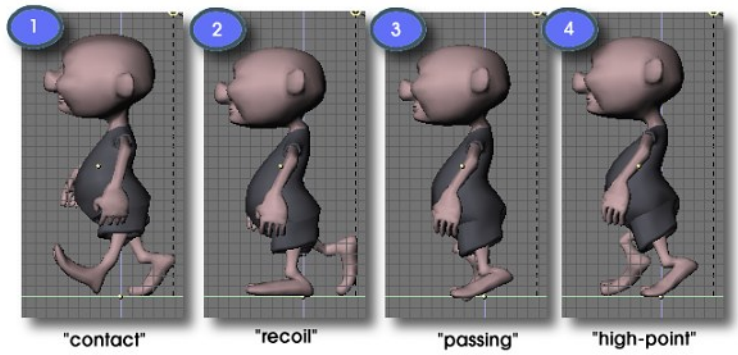


Figure 13-57. Some common poses in a walkcycle.

> **Warning**
>
> There are some details to bear in mind at the time of setting up an armature for walkcycle. As some of you might know, Blender uses a name convention for bones. If we attend to this convention the "Paste Flip Pose" button will be available to paste the "mirror" pose of our model anytime. See this chapter for more information. Also, before parenting your armature to your model, be sure their local axis are aligned to the global axis by selecting them and pressing **CTRL-SHIFT-A**

To animate our walking model we will restrict us to animate a few control bones. In the case of the legs we are going to animate its feet since the IKA solvers will adjust the leg bones better than us. To ensure that feet will move in fixed distances, please activate the Grab Grid option before start moving bones, reduce the grid size if needed.

A nice method is to hide all the bones we are not going to set keyframes for. This way is easier to see the model during animation and keeps our task simple.

Normally a walkcycle involves four poses, which are commonly known as *"contact"*, *"recoil"*, *"passing"*, *and "high-point"*. Take a look at Figure 13-57.

Most important pose is "Contact pose". Most animators agree every walkcycle should start by setting up this pose right. Here the character cover the wider distance it's capable to do in one step. In "Recoil pose", the character is in its lower position, with all its weight over one leg. In "High-point pose", the character is in its higher position, almost falling forwards. "Passing pose" is more like an automatic pose in-between recoil and high-point.

The work routine is as follows:

1. Pose the model in contact pose in frame 1
2. Insert keyframes for the control bones of your armature (those you use for grabbing, mainly IK solvers).
3. Without deselecting them press the "Copy Pose" button. Now the bone's location and rotations have been stored in memory.
4. Go a few frames forward and press "Paste Flip Pose". The flip pose will be pasted in this frame, so if in the previous frame the left leg was forwards now it will be backwards, and viceversa.

5. Now once again select your control bones and insert keyframes for them.
6. Go a few frames forward again (it is recommendable that you use the same number of frames than before, an easy choice is to go just 10 frames every ahead time) and press "Paste Pose", this will paste the initial pose ending the cycle. This way we have achieved a "Michael Jackson" style walkcycle since our character never lift its feet up from the ground.
7. To fix it, go to some intermediate position between the first two poses and move the feet to get something like the Recoil Pose in Figure 13-57, where the waist reaches its lower position.
8. Insert keyframes and copy the pose.
9. Now go to a frame between the last two poses (inverse contact and contact) and insert the flip pose . Insert the required keyframes and we are done.

Tip: If at the contrary you see that the mesh is weirdly deformed, don't panic !, go Edit Mode for the armature, select all bones and press CTRL+N. This will recalculate the direction of bones rolls which is what makes the twisting effect.

You should follow the same routine for all the poses you want to include in your walkcycle. I normally use the contact, recoil , and a high point poses and let Blender to make the passing pose.

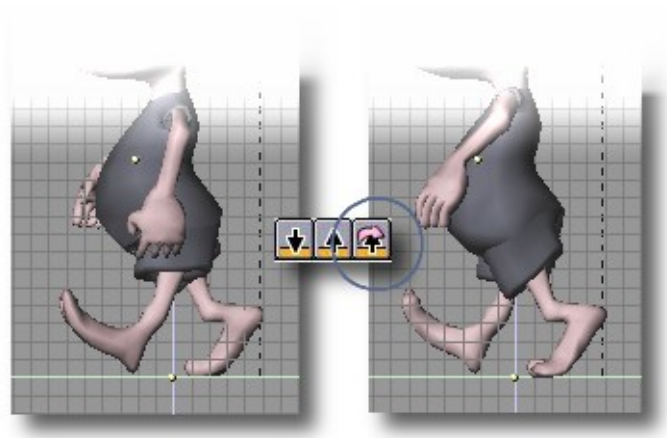


Figure 13-58. Use copy, paste and paste-flip pose buttons to be happy !

Now if you do **ALT-A** you will see our character walking almost naturally.

It will be very useful to count how many Blender Units (B.U.) are covered with each step, which can be done counting the grid squares between both feet in Pose 1. This number is the STRIDE parameter that we are going to use later on in the NLA window.

Now we will focus on make the character actually advance through the scene.

First of all deselect the walkcycle action for our armature so it stops moving when pressing **ALT-A**. To do this, press the little X button besides the action name in the action window.

Then we will create a PATH object for our hero in the ground plane, trying not to make it too curve for now (the more straight the better), once done let's parent the character to the path (and not the other way round). If everything went OK, we will see our character moving stiff along the path when pressing ALT+A.

Now go to the NLA window and add the walkcycle action in a channel as a NLA strip. With the strip selected press **N** and then push the **Use Path**button.

> **Note:** It is convenient that at the moment of adding actions in the NLA window, that no action is selected for the current armature. Why ? Because instead of a NLA strip, we'll see the individual keyframes of the action being inserted in the armature channel and this keyframes will override any prior animation strips we could have added so far. Anyway, if you do insert an action in this way, you can always convert the keyframes into a NLA strip by pressing the **CKEY**.

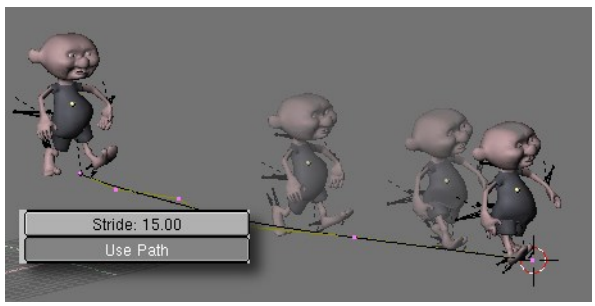


Figure 13-59. A nice stroll

Now if you start the animation again some funny things might happen. This is due to we haven't set the STRIDE parameter.

This value is the number of Blender Units that should be covered by a single walk-cycle and is very important that we estimate it with accuracy. Once calculated we should enter it in the STRIDE box.

If we adjust it well and if the walkcycle was correctly set up, our character should not "slide" across the path.

One way to estimate the Stride value accurately is to count how many grid squares there are between the toes of the feet in Pose 1. This value multiplied by 2 and by the grid scale (normally 1 grid square = 1 B.U. but this could not be the case, for instance in the example 2 grid squares = 1 B.U.) will render the searched STRIDE value.

In the example there are 7.5 squares with GRID=1.0, since the Grid scale is 1.0 we have: STRIDE = 7.5 x 1.0 x 2 = 15

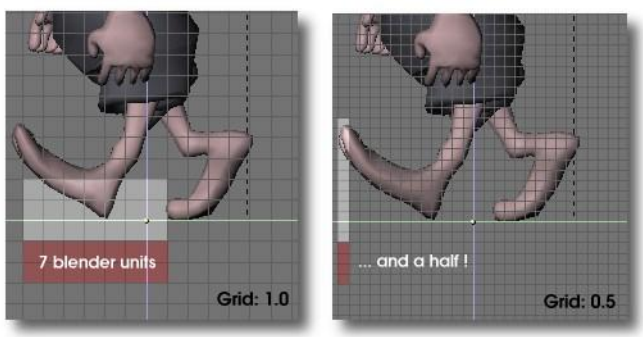


Figure 13-60. Estimating the STRIDE. Refine the grid if needed !

It's likely that we want our character to walk faster or slower or even stop for a while. We can do all this by editing the path's Speed curve.

Select the path and open an IPO window. There we will see a Speed curve normalized between 0 and 1 in ordinates (Y axis) and going from frame 1 to the last in the X axis. The Y coordinate represents the relative position in the path and the curve's slope is the speed of the parented objects. In Edit Mode we will add two points with the same Y coordinate. This "table" represents a pause in the movement and it goes from fram 40 to frame 60 in the figure.

The problem here is that when our character stops because of the pause in the curve, we will see him in a "frozen" pose with a foot on the ground and the other in the air.

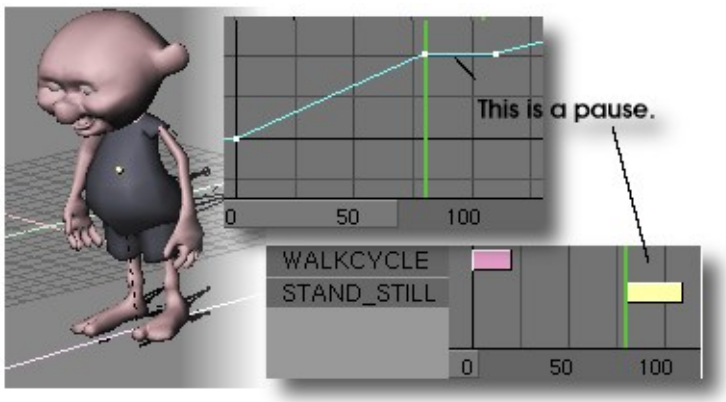


Figure 13-61. Having a rest in the walk

To fix this little problem we will use the NLA window. What we have to do is to insert the "STAND_STILL" action, this is a pose where our character is at rest. I have defined this action as only one frame by erasing all displacements and rotations of the bones. (See Clearing Transformations), and then moving a couple of bones to get a "resting" attitude.

Since the pause is from frame=78 to frame=112 we should insert this "still" action exactly there for it to perfectly fit the pause. For the animation doesn't start nor end briskly we can use the BlendIn and BlendOut options, where we can set the number of frames used to blend actions and in this way doing a more natural transition between them. In this way the character will smoothly change its pose and everything will look fine. If we do use a BlendIn or BlendOut then we should start the action BlendIn frames earlier and finish it BlendOut frames later, because the character should be still moving while changing poses.

We can of course combine different walkcycles in the same path as for instance change from walking to running in the higher speed zone.

In all these situations we will have to bear in mind that the different efects will be added from one NLA strip to the precedent strips and only them. So, the best option is to insert the walkcycle and still strips before any other.

Moving hands while walking

To add actions in the NLA window we have to locate the mouse pointer over the armature's channel and press **SHIFT-A**. A menu with all available actions will pop up. If we don't locate the pointer over an armature channel an error message `ERROR: Not an armature"` will pop up instead.

So, place the pointer over the armature strip and press **SHIFT-A** and add the "WAVE_HAND" action.

As this particular action is just the waving of the left arm to say "hello" during some point in the walkcycle, we will not use the "Use Path" option but move it in time so

it overlaps the arms keyframes from the walkcycle action. Move the pointer over the strip and press **NKEY** or just drag it and scale it to your satisfaction.

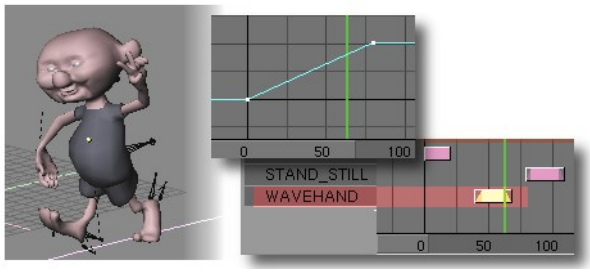


Figure 13-62. Hey guys !

Since this action is the last to be calculated (remember Blender evaluates actions from Top to Bottom in the NLA Editor), it will override any keyframes defined for the bones involved in the precedent actions.

Well, there is no much left to say about NLA and armatures. Now it is time for you to experiment and to show the results of your work to the world. One last recommendation though: it is possible to edit keyframes in the NLA window. We can duplicate frames (**SHIFT-D**), grab keyframes (**GKEY**) and also erase keyframes (**XKEY**), but if you do erase keyframes be careful because they will be lost forever from the currently selected action. So be careful and always convert to NLA strip before erasing anything.

Bye and good luck blenderheads !!

Notes

1. http://malefico3d.com.ar/tutor/gameblends/tutor-nla.zip

Chapter 14. Rendering

Rendering is the final process of CG (short of postprocessing, of course) and is the phase in which the image corresponding to your 3D scene is finally created.

The rendering buttons window is accessed via **F10** or the  button. The rendering buttons are shown in Figure 14-1.

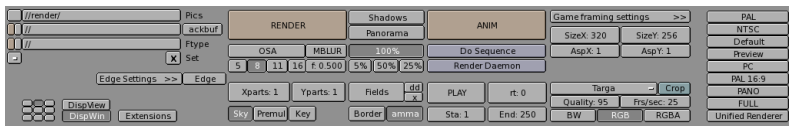


Figure 14-1. Rendering Buttons.

The rendering of the current scene is performed by pressing the big central `RENDER` button, or by pressing **F12**. The result of the rendering is kept in a buffer and shown in its own window. It can be saved by pressing **F3** or via the `File>>Save Image` menu.

The image is rendered according to the dimensions defined in the top of the central-right block of buttons (Figure 14-2).

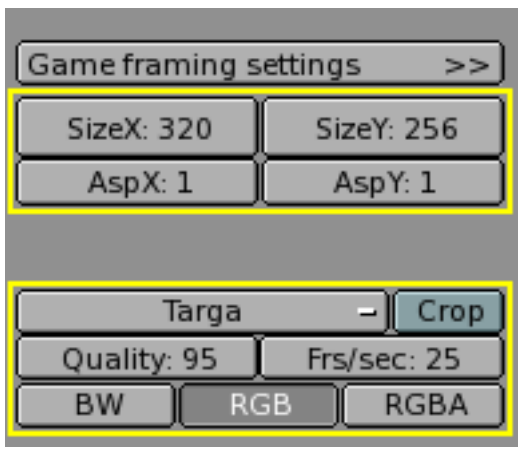


Figure 14-2. Image types and dimensions.

By default the dimensions are 320x256 and can be changed as for any NumButton. The two buttons below define the aspect ratio of the pixels. This is the ratio between the X and Y dimensions of the pixel of the image. By default it is 1:1 since computer screen pixels are square, but can be varied if television shorts are being made since TV pixels are not square. To make life easier the rightmost block of buttons (Figure 14-3) provides some common presets:

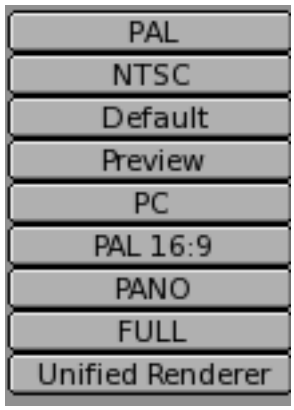


Figure 14-3. Image pre-set dimensions.

- `PAL` 720x576 pixels at 54:51 aspect ratio.
- `NTSC` 720x480 pixels at 10:11 aspect ratio.
- `Default` Same as PAL, but with full TV options, as explained in the following sections.
- `Preview` 640x512 at 1:1 aspect ratio. This setting automatically scales down the image by 50%, to effectively produce a 320x256 image.
- `PC` 640x480 at 1:1 aspect ratio.
- `PAL 16:9` 720x576 at 64:45 aspect ratio, for 16:9 widescreen TV renderings.
- `PANO` Standard panoramic settings 576x176 at 115:100 aspect ratio. More about 'panoramic' renderings in the pertinent section.
- `FULL` 1280x1024 at 1:1 aspect ratio.

Rendering by Parts

It is possible to render an image in pieces, one after the other, rather than all at one time. This can be useful for very complex scenes, where rendering small sections one after the other only requires computation of a small part of the scene, which uses less memory.

By setting values different from 1 in the `Xpart` and `Ypart` in the left column of the central buttons (Figure 14-4) you force Blender to divide your image into a grid of Xpart x Ypart sub-images which are then rendered one after the other and finally assembled together.

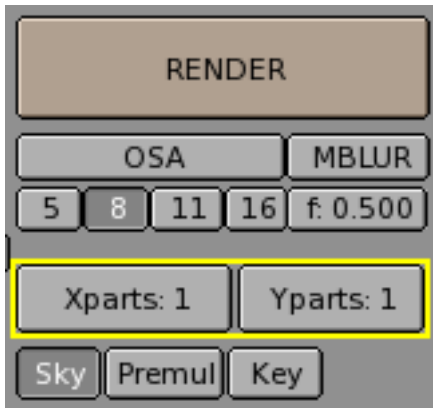


Figure 14-4. Rendering by parts buttons.

Blender cannot handle more than 64 parts.

Panoramic renderings

To obtain nice panoramic renderings, up to a full 360° view of the horizon, Blender provides an automatic procedure.

If the `Xparts` is greater than 1 and the `Panorama` button is pressed (Figure 14-5), then the rendered image is created to be Xparts x SizeX wide and SizeY high, rendering each part by rotating the camera as far as necessary to obtain seamless images.

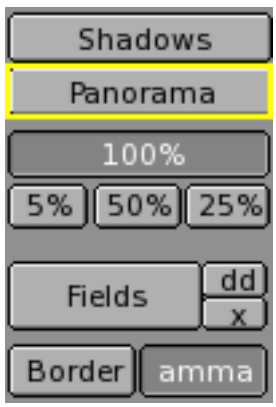


Figure 14-5. Panorama button.

Figure 14-6 shows a test set up with 12 spheres surrounding a camera. By leaving the camera as it is, you obtain the rendering shown in Figure 14-7. By setting Xparts to 3 and selecting Panorama the result is an image three times wider showing one more full camera shot to the right and one full to the left (Figure 14-8).

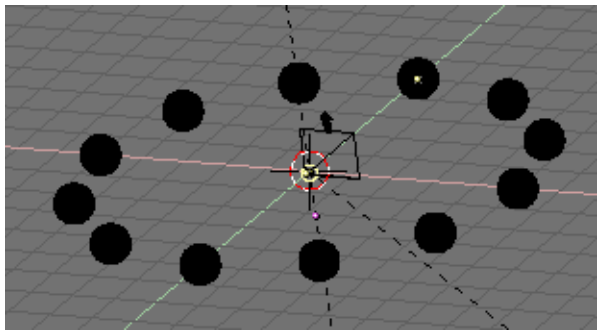


Figure 14-6. Panorama test set up.

To obtain something similar without the Panorama button, the only way is to decrease the camera focal length. For example Figure 14-9 shows a comparable view, obtained with a 7.0 focal length, equivalent to a very wide angle, or fish-eye, lens. Distortion is very evident.

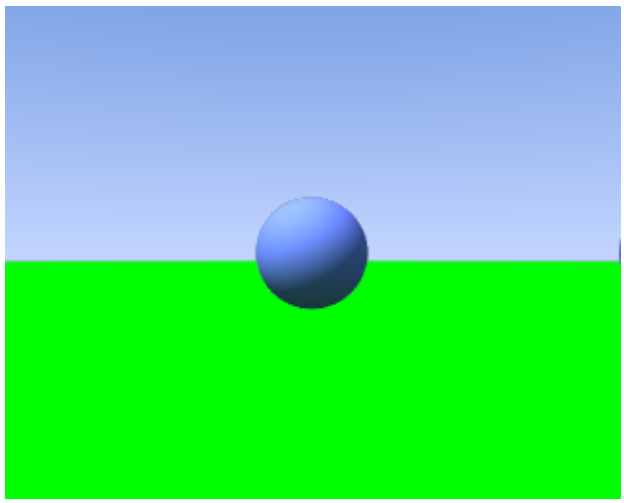


Figure 14-7. Non-panoramic rendering.

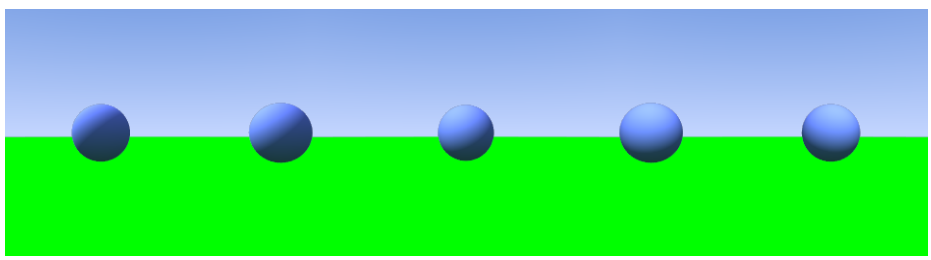


Figure 14-8. Panoramic rendering.

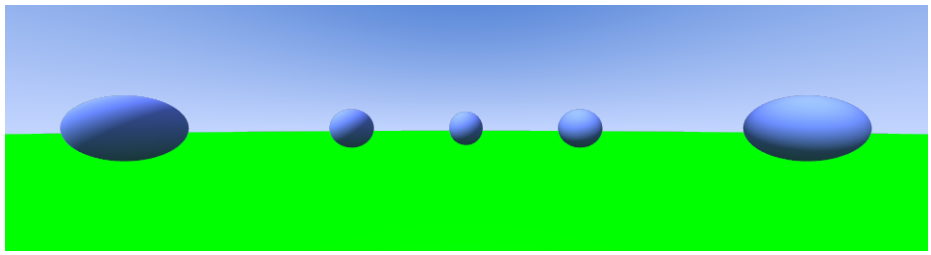


Figure 14-9. Fish-eye rendering.

To obtain a full 360° view some tweaking is necessary. It is known that a focal length of 16.0 corresponds to a viewing angle of 90°. Hence a panoramic render with 4 Xparts and a camera with a 16.0 lens yields a full 360° view, as that shown in Figure 14-10. This is grossly distorted, since a 16.0 lens is a wide angle lens, and distorts at the edges.

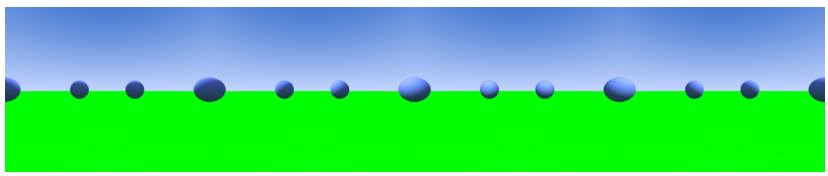


Figure 14-10. Full 360° panorama with 16.0 lenses.

To have undistorted views the focal length should be around 35.0. Figure 14-11 shows the result for a panorama with 8 Xparts and a camera with a 38.5 lens, corresponding to a 45° viewing angle.

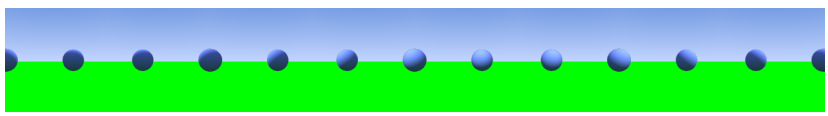


Figure 14-11. Full 360° panorama with 38.5 lenses.

The image is much less distorted, but special attention must be given to proportion. The original image was 320x256 pixels. The panorama in Figure 14-10 is 4 x 320 wide. To keep this new panorama the same width, the SizeX of the image must be set to 160 so that 8 x 160 = 4 x 320. But the camera viewing angle width occurs for the largest dimension, so that, if SizeX is kept to 256 the image spans 45° vertically but less than that horizontally, so that the final result is not a 360° panorama unless SizeX $\geq$ SizeY or you are willing to make some tests.

Antialiasing

A computer generated image is made up of pixels, these pixels can of course only be a single colour. In the rendering process the rendering engine must therefore assign a single colour 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 5 pixels or more, and assigning an 'average' colour to the rendered pixel.

The buttons to control Anti-Aliasing, or OverSAmple (OSA), are below the rendering button (Figure 14-12). By pressing the OSA button antialiasing is activated, by selecting one of the four numeric buttons below it, the level of oversampling (from 5 to 16) is chosen.

Figure 14-12. OSA buttons.

Blender uses a Delta Accumulation rendering system with jittered sampling. The values of 'OSA' (5, 8, 11, 16) are pre-set numbers that specify the number of samples; a higher value produces better edges, but slows down the rendering.

Figure 14-13 shows a rendering with OSA turned off and with 5 or 8 OSA samples.

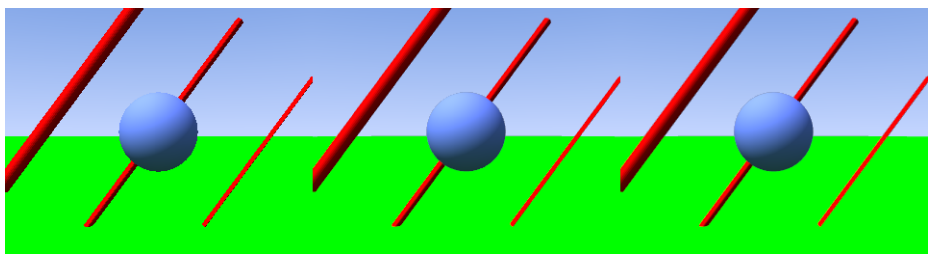


Figure 14-13. Rendering without OSA (left) with OSA=5 (center) and OSA=8 (right).

Output formats

The file is saved in whichever format has been selected in the lower pop-up menu in the center-right buttons (Figure 14-2). From here you can select many image or animation formats (Figure 14-14).

Figure 14-14. Image and animations formats.

The default image type is `Targa`, but, since the image is stored in a buffer and then saved, it is possible to change the image file type after the rendering and before saving using this menu.

By default Blender renders colour (RGB) images (bottom line in Figure 14-2) but Black and White (BW) and colour with Alpha Channel (RGBA) are also possible.

Beware that Blender does *not* automatically add the extension to files, hence any `.tga` or `.png` extension must be explicitly written in the File Save window.

Except for the Jpeg format, which yields lossy compression, all the other formats are more or less equivalent. It is generally a bad idea to use JPG since it is lossy. It is better to use Targa and then convert it to JPG for web publishing purposes, keeping the original Targa.

Anyhow, for what concerns the other formats: `TARGA raw` is uncompressed Targa, uses a lot of disk space. `PNG` is Portable Network Graphics, a standard meant to replace old GIF inasmuch as it is lossless, but supports full true colour images. `HamX` is a self-developed 8 bits RLE format; it creates extremely compact files that can be displayed quickly. To be used only for the "Play" option. `Iris` is the standard SGI format, and `Iris + Zbuffer` is the same with added Zbuffer info.

Finally `Ftype` uses an "Ftype" file, to indicate that this file serves as an example for the type of graphics format in which Blender must save images. This method allows you to process 'color map' formats. The colormap data is read from the file and used to convert the available 24 or 32 bit graphics. If the option "RGBA" is specified, standard colour number '0' is used as the transparent colour. Blender reads and writes (Amiga) IFF, Targa, (SGI) Iris and CDi RLE colormap formats.

- `AVI Raw` - saves an AVI as uncompressed frames. Non-lossy, but huge files.
- `AVI Jpeg` - saves an AVI as a series of Jpeg images. Lossy, smaller files but not as small as you can get with a better compression algorithm. Furthermore the AVI Jpeg format is not read by default by some players.

- `AVI Codec` - saves an AVI compressing it with a codec. Blender automatically gets the list of your available codecs from the operating system and allows you to set its parameters. It is also possible to change it or change its settings, once selected, via the `Set Codec` button which appears (Figure 14-15).
- `QuickTime` - saves a QuickTime animation.

Figure 14-15. AVI Codec settings.

For an AVI animation it is also possible to set the frame rate (Figure 14-15) which, by default, is 25 frames per second.

Rendering Animations

The rendering of an animation is controlled via the righthand column of the central block of buttons (Figure 14-16).

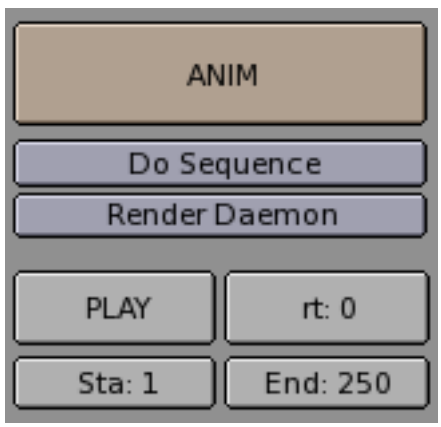


Figure 14-16. Animation rendering buttons.

The `ANIM` button starts the rendering. The first and last frames of the animation are given by the two NumButtons at the foot (`Sta:` and `End:`), and by default are 1 and 250.

By default the 3D scene animation is rendered, to make use of the sequence editor the `Do Sequence` TogButton must be selected.

By default the animation is rendered in the directory specified top left in the rendering buttons window (Figure 14-17). If an AVI format has been selected, then the name will be `####_####.avi` where the '####' indicates the start and end frame of the animation, as 4 digit integers padded with zeros as necessary.

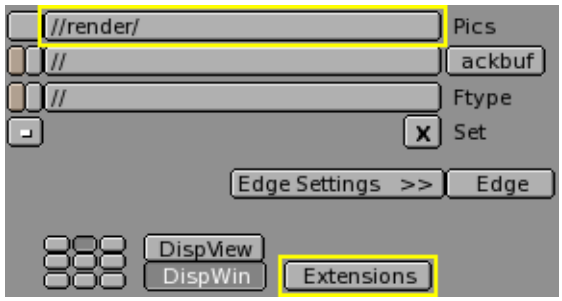


Figure 14-17. Animation location and extensions.

If an image format is chosen, on the other hand, a series of images named ####, ('####' being the pertinent frame number) is created in the directory. If the file name extension is needed, this is obtained by pressing the `Extensions` TogButton (Figure 14-17).

> **Complex animations:** Unless your animation is really simple, and you expect it to render in half an hour or less, it is always a good idea to render the animation as separate Targa frames rather than as an AVI file from the beginning.
>
> This allows you an easy recovery if the power fails and you have to re-start the rendering, since the frames you have already rendered will still be there.
>
> It is also a good idea since, if an error is present in a few frames, you can make corrections and re-render just the affected frames.
>
> You can then make the AVI out of the separate frames with Blender's sequence editor or with an external program.

Motion Blur

Blender's animations are by default rendered as a sequence of *perfectly still* images.

This is unrealistic, since fast moving objects do appear to be 'moving', that is, blurred by their own motion, both in a movie frame and in a photograph from a 'real world camera'.

To obtain such a Motion Blur effect, Blender can be made to render the current frame and some more frames, in between the real frames, and merge them all together to obtain an image where fast moving details are 'blurred'.

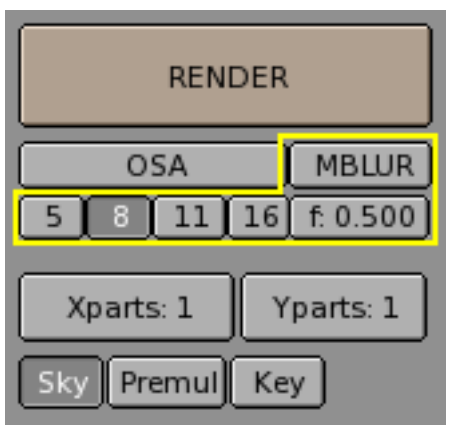


Figure 14-18. Motion Blur buttons.

To access this option select the MBLUR button next to the OSA button (Figure 14-18). This makes Blender render as many 'intermediate' frames as the oversampling number is set to (5, 8, 11 or 16) and accumulate them, one over the other, on a single frame. The number-button Bf: or Blur Factor defines the length of the shutter time as will be shown in the example below. Setting the "OSA" option is unnecessary since the Motion Blur process adds some antialiasing anyway, but to have a really smooth image 'OSA' can be activated too. This makes each accumulated image have anti-aliasing.

To better grasp the concept let's assume that we have a cube, uniformly moving 1 Blender unit to the right at each frame. This is indeed fast, especially since the cube itself has a side of only 2 Blender units.

Figure 14-19 shows a render of frame 1 without Motion Blur, Figure 14-20 shows a render of frame 2. The scale beneath the cube helps in appreciating the movement of 1 Blender unit.

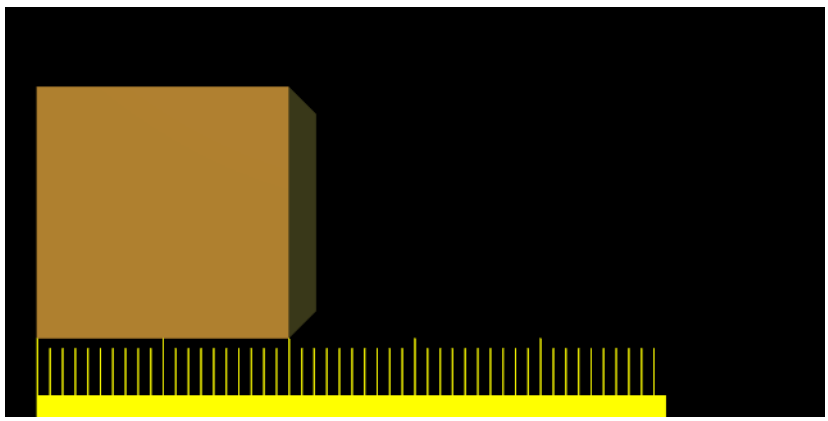


Figure 14-19. Frame 1 of moving cube without motion blur.

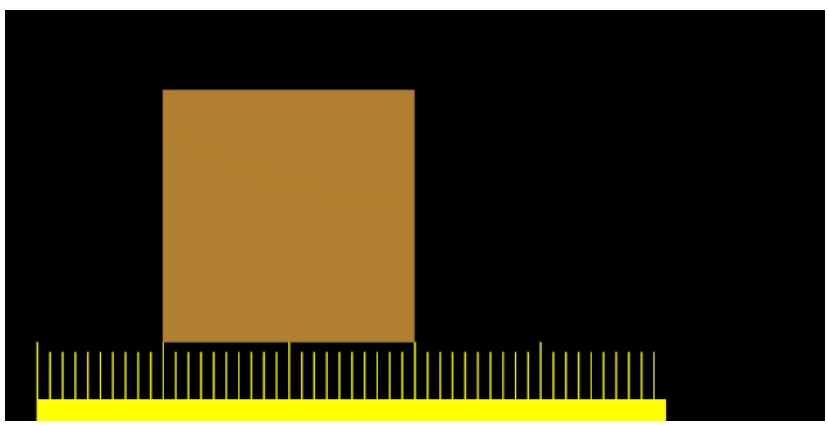


Figure 14-20. Frame 2 of moving cube without motion blur.

Figure 14-21 on the other hand shows the rendering of frame 1 when Motion Blur is set and 8 'intermediate' frames are computed. Bf is set to 0.5; this means that the 8 'intermediate' frames are computed on a 0.5 frame period starting from frame 1. This is very evident since the whole 'blurriness' of the cube occurs on half a unit before and half a unit after the main cube body.

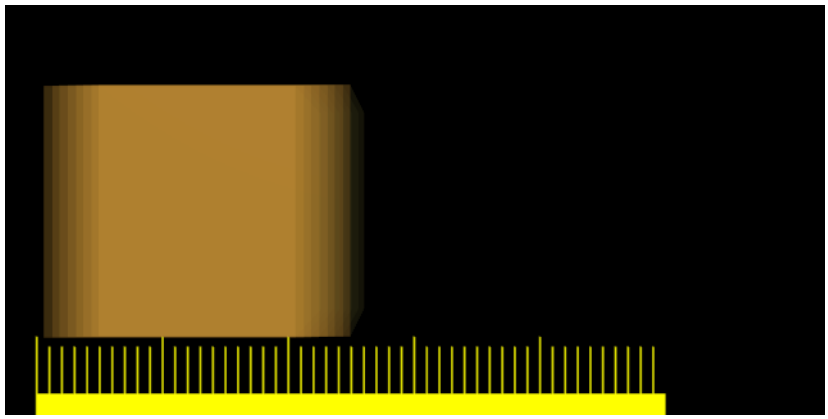


Figure 14-21. Frame 1 of moving cube with motion blur, 8 samples, Bf=0.5.

Figure 14-22 and Figure 14-23 show the effect of increasing Bf values. A value greater than 1 implies a very 'slow' camera shutter.

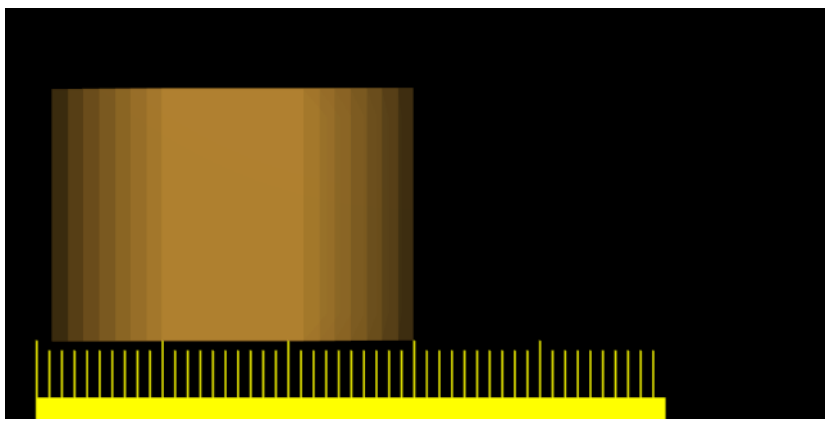


Figure 14-22. Frame 1 of moving cube with motion blur, 8 samples, Bf=1.0.

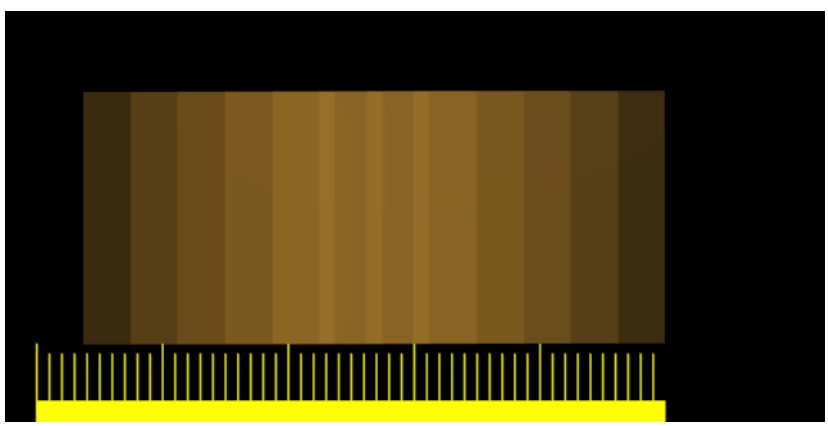


Figure 14-23. Frame 1 of moving cube with motion blur, 8 samples, Bf=3.0.

Better results than those shown can be obtained by setting 11 or 16 samples rather than 8, but, of course, since as many *separate* renders as samples are needed a Motion Blur render takes that many times more than a non-motion blur one.

Better Anti-Aliasing: If Motion Blur is active, even if nothing is moving on the scene, Blender actually 'jitters' the camera a little between an 'intermediate' frame and the next. This implies that, even if OSA is off, the resulting images have nice Anti-Aliasing. An MBLUR obtained Anti-Aliasing is comparable to an OSA Anti-Aliasing of the same level, but generally slower.

This is interesting since, for very complex scenes where a level 16 OSA does not give satisfactory results, better results can be obtained using *both* OSA and MBlur. This way you have as many samples per frame as you have 'intermediate' frames, effectively giving oversampling at levels 25,64,121,256 if 5,8,11,16 samples are chosen, respectively.

Depth of Field

Depth of Field (DoF) is an interesting effect in real world photography which adds a lot to CG generated images. It is also known as Focal Blur.

The phenomenon is linked to the fact that a real world camera can focus on a subject at a given distance, so objects closer to the camera and objects further away will be out of the focal plane, and will therefore be slightly blurred in the resulting photograph.

The amount of blurring of the nearest and furthest objects varies a lot with the focal length and aperture size of the lens and, if skilfully used, can give very pleasing effects.

Blender's renderer does not provide an automatic mechanism for obtaining DoF, but there are two alternative way to achieve it. One relies solely on Blender's internals, and will be described here. The other requires an external sequence plugin and will be outlined in the Sequence Editor Chapter.

The hack to obtain DoF in Blender relies on skilful use of the Motion Blur effect described before, making the Camera move circularly around what would be the aperture of the 'real world camera' lens, constantly pointing at a point where 'perfect' focus is desired.

Assume that you have a scene of aligned spheres, as shown on the the left of Figure 14-24. A standard Blender rendering will result in the image on the right of Figure 14-24, with all spheres perfectly sharp and in focus.

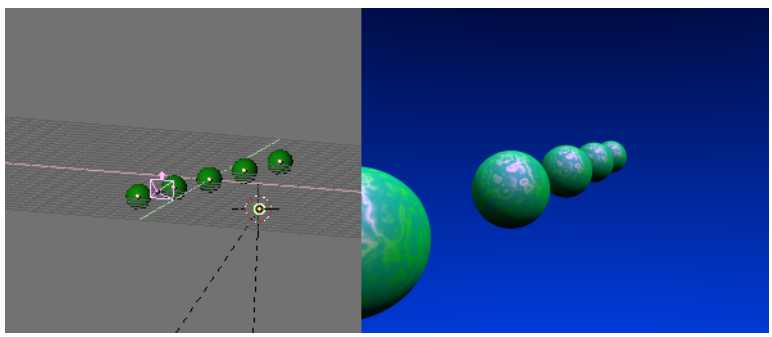


Figure 14-24. Depth of Field test scene.

The first step is to place an Empty (**SPACE**>>ADD>>Empty) where the focus will be. In our case at the center of the middle sphere (Figure 14-25).

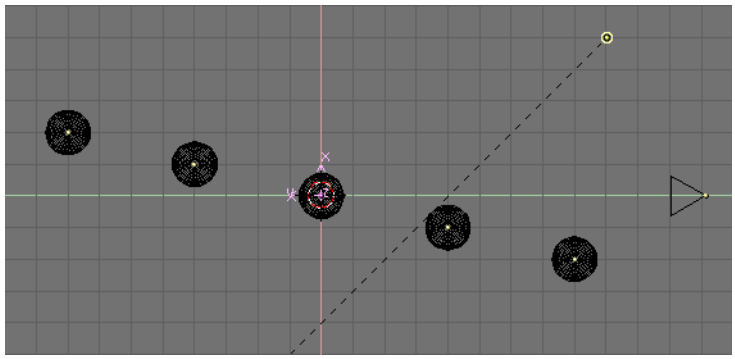


Figure 14-25. Setting the Focus Empty.

Then, assuming that your Camera is already placed in the correct position, place the cursor on the Camera (Select the Camera, **SHIFT+S**>>Curs->Sel) and add a NURBS circle (**SPACE**>>ADD>>Curve>>NURBS Circle).

Out of EditMode (**TAB**) scale the circle. This is very arbitrary, and you might want to re-scale it later on to achieve better results. Basically, the circle size is linked to the physical aperture size, or diaphragm, of your 'real world camera'. The larger the circle the narrower the region with perfect focus will be, and the near and far objects will be more blurred. The DoF blurring will be less evident the smaller the circle.

Now, keeping the circle selected, also select the Empty and press **CTRL+T** to have the circle track the Empty as in Figure 14-26. Since the normal to the plane containing the circle is the local z-axis, you will have to set up tracking correctly so that the local z-axis of the circle points to the Empty (Figure 14-27) and the circle is orthogonal to the line connecting its center to the Empty.

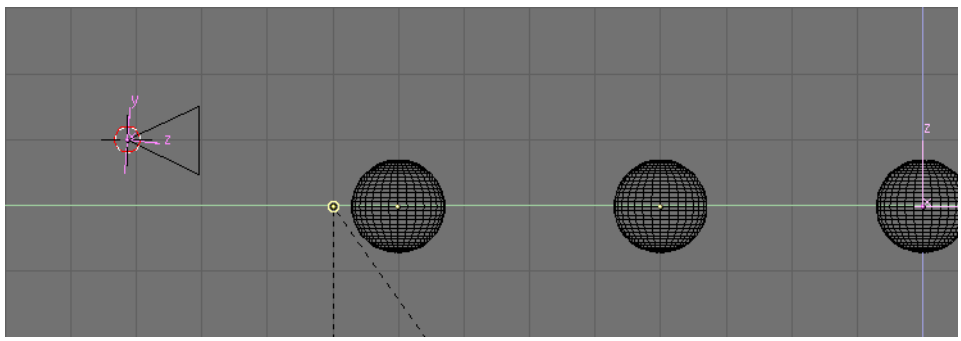


Figure 14-26. NURBS circle tracking the focus Empty.

Figure 14-27. Correct tracking settings for the circle.

Select the Camera and then the circle and parent the Camera to the circle (**CTRL+P**). In the Animation Buttons Window (**F7**) press the `CurvePath` button.

With the circle still selected, open an IPO window (**SHIFT+F6**) select the Curve IPO with the  button. The only available IPO is 'Speed'. **CTRL+LMB** twice at random in the IPO window to add an IPO with two random points. Then set these points numerically by using the new set of buttons which have appeared in the Animation Buttons Window. You should set `Xmin` and `Ymin` to 0, `Xmax` and `Ymax` to 1, then press

the SET button. To complete the IPO editing make it cyclic via the button. The final result should be as shown in Figure 14-28.

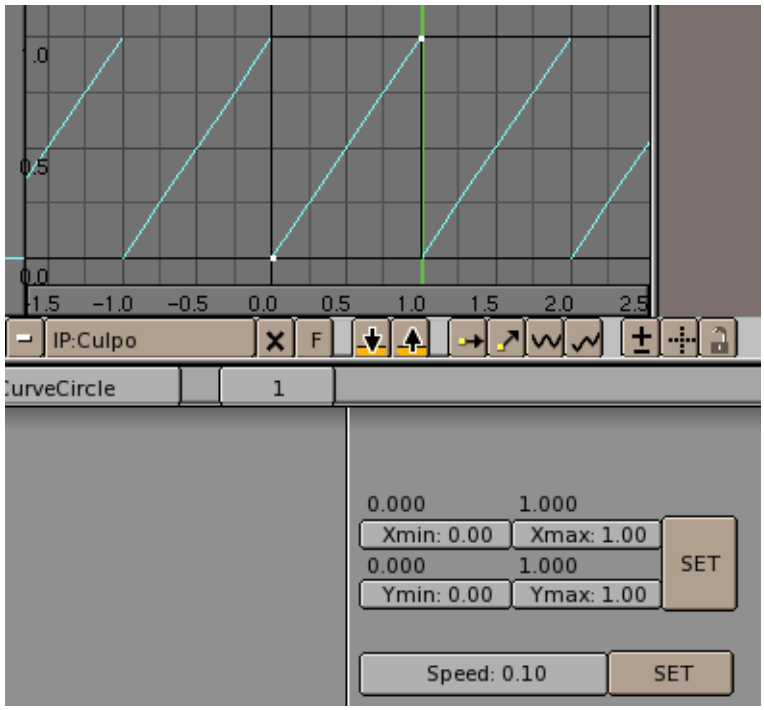


Figure 14-28. Speed IPO for the NURBS circle path.

With these settings we have effectively made the Camera circle around its former position along the NURBS circle path in exactly 1 frame. This makes the Motion Blur option take slightly different views of the scene and create the Focal Blur effect in the end.

There is still one more setting to perform. First select the Camera and then the focal Empty, and make the Camera track the Empty (**CTRL+T**). The Camera will most probably go crazy because it might already have a rotation and it is parented to a circle too, so press **ALT+R** and select `Clear rotation` from the menu which appears to clear all Camera rotations except the tracking.

The Camera should now track the Empty, as in Figure 14-29.

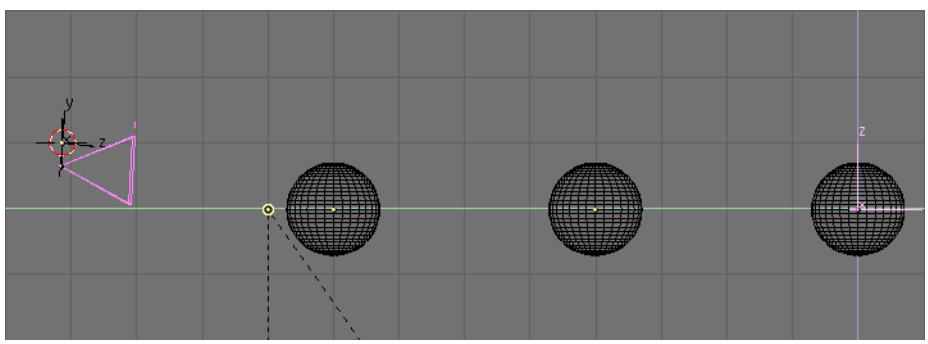


Figure 14-29. Camera tracking the focal Empty.

If you press **ALT+A** now you won't see any movement because the Camera does exactly one full circle path in each frame, so it appears to be still, nevertheless the

Motion Blur engine will detect these moves.

The last touch is then to go to the rendering buttons window (**F10**) and select the `MBLUR` button. You most probably don't need the `OSA` button active, since Motion Blur will implicitly do some antialiasing. It is strongly recommended that you set the Motion Blur factor to 1, since this way you will span the entire frame for blurring, taking the whole circle length. It is also necessary to set the oversamples to the maximum level (16) for best results (Figure 14-30).

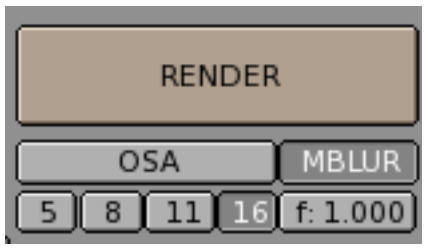


Figure 14-30. Motion blur settings.

A rendering (**F12**) will yield the desired result. This can be much slower than a non-DoF rendering since Blender effectively renders 16 images and then merges them. Figure 14-31 shows the result, to be compared with the one in Figure 14-24. It must be noted that the circle has been scaled much less to obtain this picture than has been shown in the example screenshots. These latter were made with a large radius (equal to 0.5 Blender units) to demonstrate the technique better. On the other hand, Figure 14-31 has a circle whose radius is 0.06 Blender units.

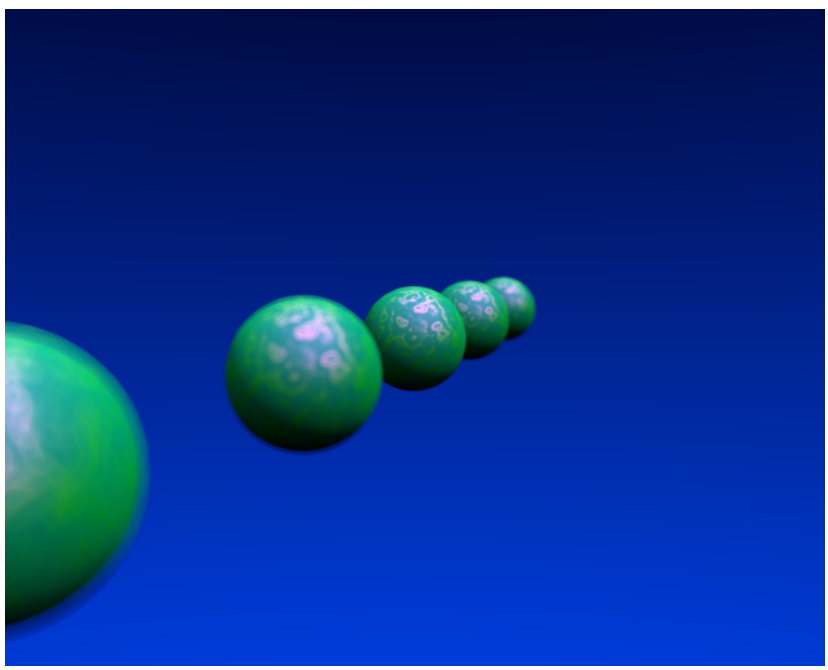


Figure 14-31. Motion blur final rendering.

This technique is interesting and with it it's pretty easy to obtain small degrees of Depth of Field. For big Focal Blurs it is limited by the fact that it is not possible to have more than 16 oversamples.

Cartoon Edges

Blender's new material shaders, as per version 2.28, include nice toon diffuse and specular shaders.

By using these shaders you can give your rendering a comic-book-like or manga-like appearance, affecting the shades of colours, as you may be able to appreciate in Figure 14-32.

Figure 14-32. A scene with Toon materials.

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.

To access this option select the `Edge` button next to the `OSA` button (Figure 14-33). This makes Blender search for edges in your rendering and add an 'outline' to them.

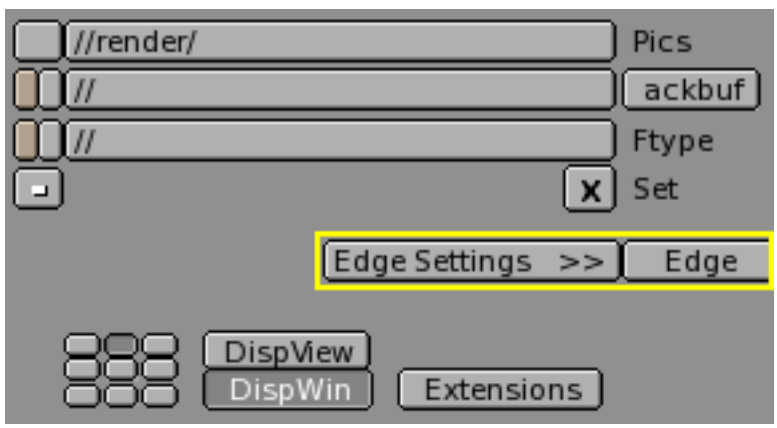


Figure 14-33. Toon edge buttons.

Before repeating the rendering it is necessary to set some parameters. The `Edge Settings` opens a window to set these (Figure 14-34).

Figure 14-34. Toon edge settings.

In this window it is possible to set the edge colour, which is black by default, and its intensity, `Eint` which is an integer ranging from 0 (faintest) to 255 (strongest). The other buttons are useful if the Unified render is used (see next section).

Figure 14-35 shows the same image as Figure 14-32 but with toon edges enabled, of black colour and maximum intensity (Eint=255).

Figure 14-35. Scene re-rendered with toon edge set.

The Unified Renderer

A less well known feature of Blender is the Unified Renderer button in the bottom right corner of the Rendering Buttons (Figure 14-36).

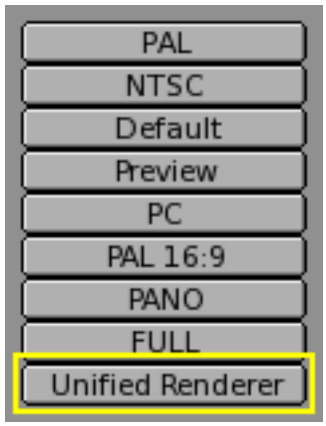


Figure 14-36. The Unified Renderer button.

Blender's *default* renderer is highly optimized for speed. This has been achieved by subdividing the rendering process into several passes. First the 'normal' materials are handled. Then Materials with transparency (Alpha) are taken into account. Finally Halos and flares are added.

This is fast, but can lead to less than optimum results, especially with Halos. The Unified Renderer, on the other hand, renders the image in a single pass. This is slower, but gives better results, especially for Halos.

Furthermore, since transparent materials are now rendered together with the conventional ones, Cartoon Edges can be applied to them too, by pressing the `All` button in the Edge Setting dialog.

If the Unified Renderer is selected an additional group of buttons appears (Figure 14-37).

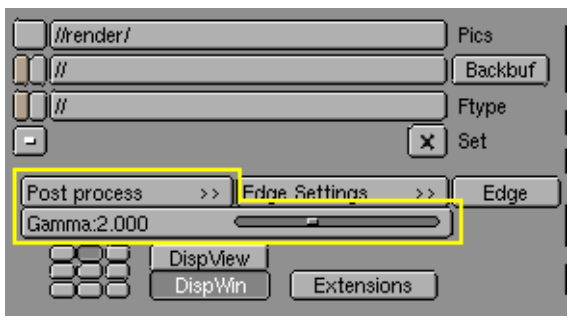


Figure 14-37. Unified Renderer additional buttons.

The `Gamma` slider is related to the OSA procedure. Pixel oversamples are blended to generate the final rendered pixel. The conventional renderer has a Gamma=1, but in the Unified Renderer you can vary this number.

The `Post process` button makes a dialog box appear (Figure 14-38). From this you can control three kinds of post processing: the `Add` slider defines a constant quantity to be added to the RGB colour value of each pixel. Positive values make the image uniformly brighter, negative uniformly darker.

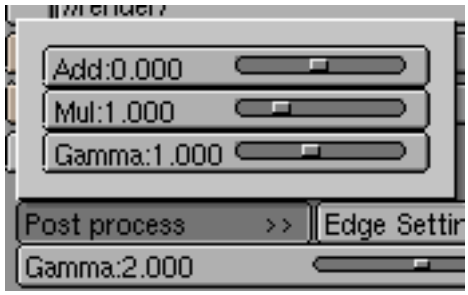


Figure 14-38. Unified Renderer postprocess submenu.

The `Mul` slider defines a value by which all RGB values of all pixels are multiplied. Values greater than 1 make the image brighter, smaller than 1 make the image darker.

The `Gamma` slider does the standard gamma contrast correction of any paint program

Preparing your work for video (x)

Once you mastered the animations trick you will surely start to produce wonderfull animations, Encoded with your favourite codecs and possibly sharion it on the internet with all the community.

But, sooner or later, you will be struck by the desire of building an animation for Television, mayby burning you own DVDs.

To spare you some disappointments here are some tips specifically targeted at Video preparation. The first and principal is to remember the double dashed white line in camera view!

If you render for PC then the whole rendered image, which lies within the *outher* 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 yout TV cathodic ray tube. You are guaranteed that what is within the *inner* dashed rectangle in camera view will be visible on the screen. Everything withing the two rectangles may and may not be visible, depending on the given TV set you will see the video on.

Furthermore the rendering size is strictly dictated by the TV standard. Blender has three pre-set settings for your convenience:

- `PAL` 720x576 pixels at 54:51 aspect ratio.
- `NTSC` 720x480 pixels at 10:11 aspect ratio.
- `PAL 16:9` 720x576 at 64:45 aspect ratio, for 16:9 widescreen TV renderings.

Color Saturation (-)

(to be written)

Rendering to fields (-)

The TV standard prescribes that there should be 25 frames per second (PAL) or 30 frames per second (NTSC). Since the phosporous of the screen do not maintain luminosity too long this could produce a noticeable flickering. To minimize this TV do not represent frames as Computer does but rather represent 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 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 nonnegligible time difference between each field (1/50 or 1/60 of a second) merely rendering a frame the usual way and split it into two half frames does not work. A noticeable jitter of the edges of moving objects would be present.

Figure 14-39. Field Rendering setup.

To optimally handle this issue Blender allows for field rendering. When the `Fields` button is pressed (Figure 14-39). Blender prepares each frame in two passes, on the first it renders only the even lines, the it *advances in time by half time step* and renders all the odd lines.

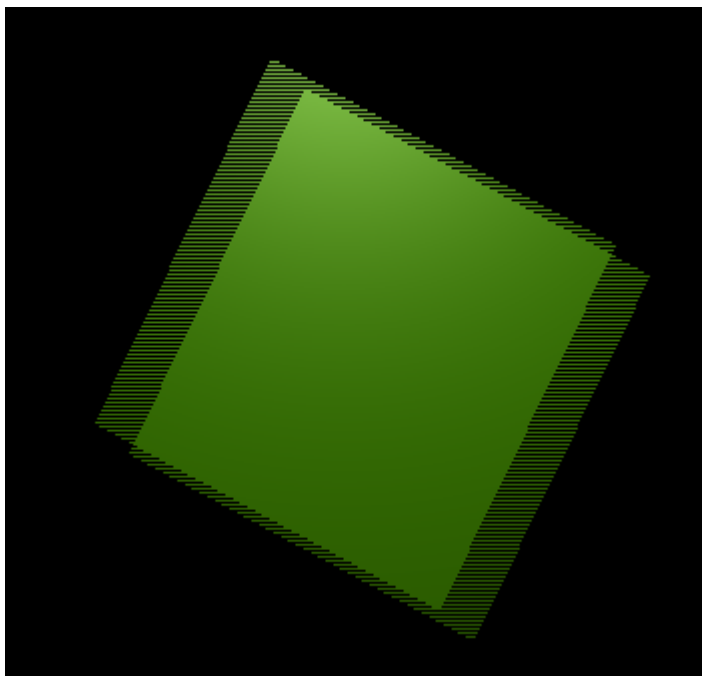


Figure 14-40. Field Rendering result.

This produces odd results on a PC screen (Figure 14-40) but will show correctly on a TV set.

The two buttons next to the `Fields` button forces the rendering of Odd fields first (`Odd`) and disable the half-frame time step between fields (`x`).

Setting up the correct field order (x)

Odd field is to be scanned first on NTSC

Chapter 15. Radiosity (x)

Most rendering models, including ray-tracing, assume a simplified spatial model, highly optimised for the light that enters our 'eye' in order to draw the image. You can add reflection and shadows to this model to achieve a more realistic result. Still, there's an important aspect missing! When a surface has a reflective light component, it not only shows up in our image, it also shines light at surfaces in its neighbourhood. And vice-versa. In fact, light bounces around in an environment until all light energy is absorbed (or has escaped!).

Re-irradiated light carries information about the object which has reirradiated it, notably colour. Hence not only the shadows are 'less black' because of re-irradiated light, but also they tend to show the colour of the nearest, brightly illuminated, object. A phenomenon often referred to as 'colour leaking' (Figure 15-1).

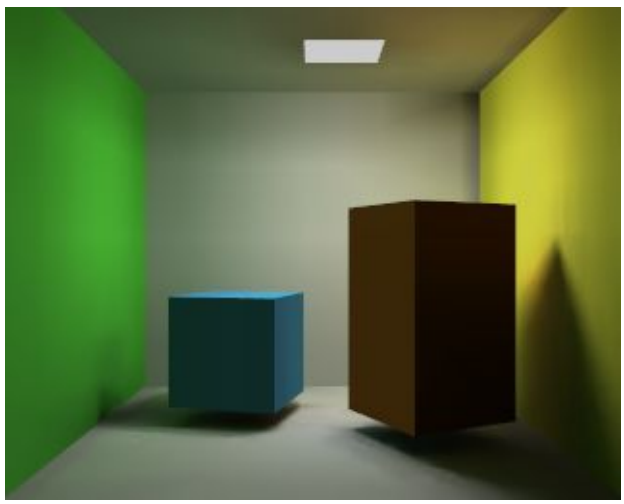


Figure 15-1. Radiosity example

In closed environments, light energy is generated by 'emitters' and is accounted for by reflection or absorption of the surfaces in the environment. The rate at which energy leaves a surface is called the 'radiosity' of a surface. Unlike conventional rendering methods, radiosity methods first calculate all light interactions in an environment in a view-independent way. Then, different views can be rendered in real-time.

In Blender, Radiosity is more of a modelling tool than a rendering tool. It is the integration of an external tool and still has all the properties (and limits) of external tools.

You can run a *radiosity solution* of your scene. The output of such Radiosity solution is a new Mesh Object with vertex colors. These can be retouched with the Vertex-Paint option or rendered using the Material properties "VertexCol" (light color) or "VColPaint" (material color). Even new Textures can be applied, and extra lamps and shadows added.

Currently the Radiosity system doesn't account for animated Radiosity solutions. It is meant basically for static environments, real time (architectural) walkthroughs or just for fun to experiment with a simulation driven lighting system.

The Blender Radiosity method

First, some theory! You can skip to next section if you like, and get back here if questions arise.

During the late eighties and early nineties radiosity was a hot topic in 3D computer graphics. Many different methods were developed, the most successful of these solutions were based on the "progressive refinement" method with an "adaptive subdivision" scheme. And this is what Blender uses.

To be able to get the most out of the Blender Radiosity method, it is important to understand the following principles:

- *Finite Element Method*

  Many computer graphics or simulation methods assume a simplification of reality with 'finite elements'. For a visually attractive (and even scientifically proven) solution, it is not always necessary to dive into a molecular level of detail. Instead, you can reduce your problem to a finite number of representative and well-described elements. It is a common fact that such systems quickly converge into a stable and reliable solution.

  The Radiosity method is a typical example of a finite element method inasmuch as every face is considered a 'finite element' and its light emission considered as a whole.

- *Patches and Elements*

  In the radiosity universe, we distinguish between two types of 3D faces:

  Patches. These are triangles or squares which are able to *send energy*. For a fast solution it is important to have as few of these patches as possible. But, because of the approximations taken the energy is only distributed from the Patch's center, the size should be small enough to make a realistic energy distribution. (For example, when a small object is located above the Patch center, all energy the Patch sends is obscured by this object).

  Elements These are the triangles or squares which *receive energy*. Each Element is associated to a Patch. In fact, Patches are subdivided into many small Elements. When an element receives energy it absorbs part of it (depending on the Patch color) and passes the remainder to the Patch. Since the Elements are also the faces that we display, it is important to have them as small as possible, to express subtle shadow boundaries.

- *Progressive Refinement*

  This method starts with examining all available Patches. The Patch with the most 'unshot' energy is selected to shoot all its energy to the environment. The Elements in the environment receive this energy, and add this to the 'unshot' energy of their associated Patches. Then the process starts again for the Patch NOW having the most unshot energy. This continues for all the Patches until no energy is received anymore, or until the 'unshot' energy has converged below a certain value.

- *The hemicube method*

  The calculation of how much energy each Patch gives to an Element is done through the use of 'hemicubes'. Exactly located at the Patch's center, a hemicube (literally 'half a cube') consist of 5 small images of the environment. For each pixel in these images, a certain visible Element is color-coded, and the transmitted amount of energy can be calculated. Especially with the use of specialized hardware the hemicube method can be accelerated significantly. In Blender, however, hemicube calculations are done "in software".

  This method is in fact a simplification and optimisation of the 'real' radiosity formula (form factor differentiation). For this reason the resolution of the hemicube

(the number of pixels of its images) is approximate and its careful setting is important to prevent aliasing artefacts.

- *Adaptive subdivision*

  Since the size of the patches and elements in a Mesh defines the quality of the Radiosity solution, automatic subdivision schemes have been developed to define the optimal size of Patches and Elements. Blender has two automatic subdivision methods:

  1. Subdivide-shoot Patches. By shooting energy to the environment, and comparing the hemicube values with the actual mathematical 'form factor' value, errors can be detected that indicate a need for further subdivision of the Patch. The results are smaller Patches and a longer solving time, but a higher realism of the solution.

  2. Subdivide-shoot Elements. By shooting energy to the environment, and detecting high energy changes (frequencies) inside a Patch, the Elements of this Patch are subdivided one extra level. The results are smaller Elements and a longer solving time and maybe more aliasing, but a higher level of detail.

- *Display and Post Processing*

  Subdividing Elements in Blender is 'balanced', that means each Element differs a maximum of '1' subdivide level with its neighbours. This is important for a pleasant and correct display of the Radiosity solution with Gouraud shaded faces. Usually after solving, the solution consists of thousands of small Elements. By filtering these and removing 'doubles', the number of Elements can be reduced significantly without destroying the quality of the Radiosity solution. Blender stores the energy values in 'floating point' values. This makes settings for dramatic lighting situations possible, by changing the standard multiplying and gamma values.

- *Rendering and integration in the Blender environment*

  The final step can be replacing the input Meshes with the Radiosity solution (button `Replace Meshes`). At that moment the vertex colors are converted from a 'floating point' value to a 24 bits RGB value. The old Mesh Objects are deleted and replaced with one or more new Mesh Objects. You can then delete the Radiosity data with `Free Data`. The new Objects get a default Material that allows immediate rendering. Two settings in a Material are important for working with vertex colors:

  VColPaint This option treats vertex colors as a replacement for the normal RGB value in the Material. You have to add Lamps in order to see the radiosity colors. In fact, you can use Blender lighting and shadowing as usual, and still have a neat radiosity 'look' in the rendering.

  VertexCol It would have been better to call this option "VertexLight". The vertex-colors are added to the light when rendering. Even without Lamps, you can see the result. With this option, the vertex colors are pre-multiplied by the Material RGB color. This allows fine-tuning of the amount of 'radiosity light' in the final rendering.

The Interface

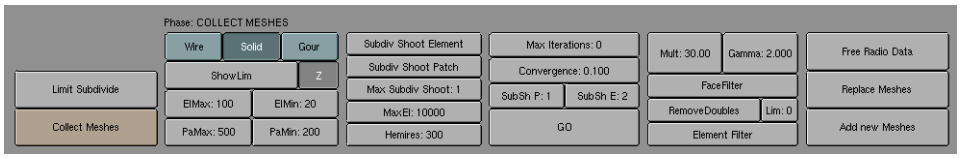


Figure 15-2. Radiosity Buttons

As with everything in Blender, Radiosity settings are stored in a datablock. It is attached to a Scene, and each Scene in Blender can have a different Radiosity 'block'. Use this facility to divide complex environments into Scenes with independent Radiosity solvers.

Radiosity Quickstart

Let's assume you have a scene ready. The first thing to grasp when doing Radiosity is that *no Lamps are necessary*, but some meshes with Emit material property greater than zero are, since these will be the light sources.

You can build the test scene shown in Figure 15-1, it is rather easy, just make a big cube, the room, give different materials to the side walls, add a cube and a stretched cube within and add a plane with an non-zero Emit value next to the roof, to simulate the area light.

Please note that the light emission is governed by the direction of the normals, of a mesh, so the light emitting plane should have a *downward* pointing normal.

Switch to the Radiosity Buttons . The Button window is shown in (Figure 15-2).

1. Select all meshes (**AKEY** and press the button `Collect Meshes`)

   Now the selected Meshes are converted into the primitives needed for the Radiosity calculation. Blender now has entered the Radiosity mode, and other editing functions are blocked until the button "Free Data" has been pressed.

2. Press the button `Gourad` as opposed to `Solid` to have smooth shading. Press the `GO` Button.

   First you will see a series of initialisation steps (at a PIV, it takes a blink), and then the actual radiosity solution is calculated. The cursor counter displays the current step number.

   Theoretically, this process can continue for hours, as photons are shot and bounced. Luckly we are not very interested in the correctness of the solution, instead most environments display a satisfying result within a few minutes. Blender shows the resulting solution as it progresses. When you are satisfied, to stop the solving process, press `ESC`.

3. Now the Gouraud shaded faces display the energy as vertex colors. You can clearly see the 'color bleeding' in the walls, the influence of a colored object near a neutral light-grey surface. In this phase you can do some postprocess editing to reduce the number of faces or filter the colors. These are described in detail in the next section.

4. To leave the Radiosity mode and save the results press "Replace Meshes" and "Free Radio Data". Now we have a new Mesh Object with vertex colors. There's also a new Material added with the right properties to render it (Press **F5** or **F12**).

   Beware, this is a one-way only process. Your original objects are lost... It's better to have saved a copy of your work.

Radiosity Step by Step

Ok, the quickstart might have been too 'superficial' and you want to get deeper insight! Go on reading.

There are few important points to grasp for practical Radiosity:

Only Meshes in Blender are allowed as input for Radiosity. It is important to realize that *each* face in a Mesh becomes a Patch, and thus a potential energy emitter and reflector. Typically, large Patches send and receive more energy than small ones. It is therefore important to have a well-balanced input model with Patches large enough to make a difference! When you add extremely small faces, these will (almost) never receive enough energy to be noticed by the "progressive refinement" method, which only selects Patches with large amounts of unshot energy.

> **Non-mesh Objects:** *Only Meshes* means that you have to convert Curves and and Surfaces to Meshes (**CTRL+C**) before starting the Radiosity solution!

You assign Materials as usual to the input models. The RGB value of the Material defines the Patch color. The 'Emit' value of a Material defines if a Patch is loaded with energy at the start of the Radiosity simulation. The "Emit" value is multiplied with the area of a Patch to calculate the initial amount of unshot energy. Textures in a Material are not taken account of.

Phase 1: Collect Meshes

All selected and visible Meshes in the current Scene are converted to Patches as soon as the `Collect Meshes` button is pressed (Figure 15-3). As a result some Buttons in the interface change color. Blender now has entered the Radiosity mode, and other editing functions are blocked until the button "Free Data" has been pressed. The "Phase" text now says 'Init' and shows the number of Patches and Elements.

> **Emitting faces:** Check the number of "Emit:" patches, if this is zero nothing interesting can happen! You need at least 1 emitting patch to have light and hence a solution.

After the Meshes are collected, they are drawn in a pseudo lighting mode that clearly differs from the normal drawing. The 'collected' Meshes are not visible until `Free Radio Data` has been invoked at the end of the process.

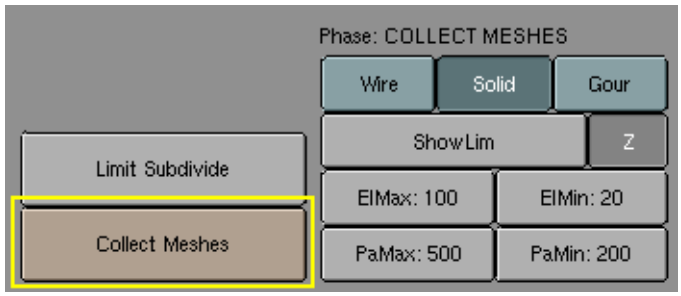


Figure 15-3. Collect Mesh button

Wire, Solid, Gour (RowBut) Three drawmode options are included which draw independent of the indicated drawmode of a 3DWindow. Gouraud display is only performed after the Radiosity process has started.

Press also the `Gour` button, to have smoother results on curved surfaces (Figure 15-4).

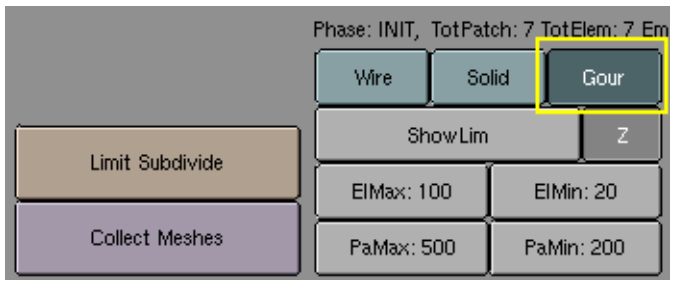


Figure 15-4. Gourad button

Phase 2: Subdivision limits.

Blender offers a few settings to define the minimum and maximum sizes of Patches and Elements (Figure 15-5).

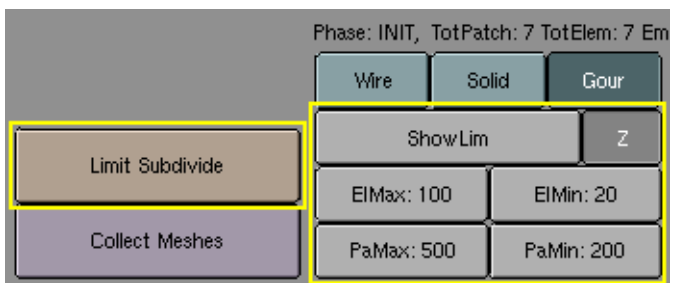


Figure 15-5. Radiosity Buttons for Subdivision

Limit Subdivide (But) With respect to the values "PaMax" and "PaMin", the Patches are subdivided. This subdivision is also automatically performed when a "GO" action has started.

PaMax, PaMin (NumBut)

ElMax, ElMin (NumBut) The maximum and minimum size of a Patch or Element. These limits are used during all Radiosity phases. The unit is expressed in 0.0001

of the boundbox size of the entire environment. Hence, with deafault 500 and 200 settings maximum and minimum Patch size 0.05 of the entire model (1/20) and 0.02 of the entire model (1/50).

ShowLim, Z (TogBut) This option visualizes the Patch and Element limits. By pressing the 'Z' option, the limits are drawn rotated differently. The white lines show the Patch limits, cyan lines show the Element limits.

Phase 3: Adaptive Subdividing

Last settings before starting the analysis (Figure 15-6).

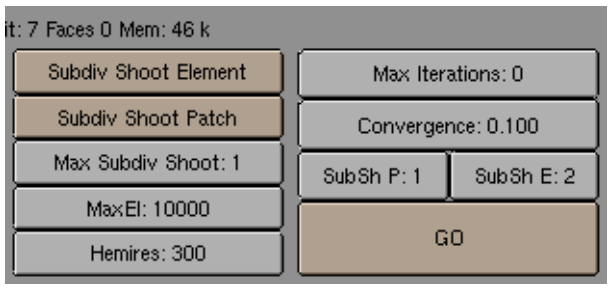


Figure 15-6. Radiosity Buttons

Hemires (NumBut) The size of a hemicube; the color-coded images used to find the Elements that are visible from a 'shoot Patch', and thus receive energy. Hemicubes are not stored, but are recalculated each time for every Patch that shoots energy. The "Hemires" value determines the Radiosity quality and adds significantly to the solving time.

MaxEl (NumBut) The maximum allowed number of Elements. Since Elements are subdivided automatically in Blender, the amount of used memory and the duration of the solving time can be controlled with this button. As a rule of thumb 20,000 elements take up 10 Mb memory.

Max Subdiv Shoot (NumBut) The maximum number of shoot Patches that are evaluated for the "adaptive subdivision" (described below) . If zero, all Patches with 'Emit' value are evaluated.

Subdiv Shoot Patch (But) By shooting energy to the environment, errors can be detected that indicate a need for further subdivision of Patches. The subdivision is performed only once each time you call this function. The results are smaller Patches and a longer solving time, but a higher realism of the solution. This option can also be automatically performed when the "GO" action has started.

Subdiv Shoot Element (But) By shooting energy to the environment, and detecting high energy changes (frequencies) inside a Patch, the Elements of this Patch are selected to be subdivided one extra level. The subdivision is performed only once each time you call this function. The results are smaller Elements and a longer solving time and probably more aliasing, but a higher level of detail. This option can also be automatically performed when the "GO" action has started.

GO (But) With this button you start the Radiosity simulation. The phases are:

1. *Limit Subdivide* When Patches are too large, they are subdivided.
2. *Subdiv Shoot Patch.* The value of "SubSh P" defines the number of times the "Subdiv Shoot Patch" function is called. As a result, Patches are subdivided.

3. *Subdiv Shoot Elem.* The value of "SubSh E" defines the number of times the "Subdiv Shoot Element" function is called. As a result, Elements are subdivided.
4. *Subdivide Elements.* When Elements are still larger than the minimum size, they are subdivided. Now, the maximum amount of memory is usually allocated.
5. *Solve.* This is the actual 'progressive refinement' method. The mousecursor displays the iteration step, the current total of Patches that shot their energy in the environment. This process continues until the unshot energy in the environment is lower than the "Convergence" or when the maximum number of iterations has been reached.
6. *Convert to faces* The elements are converted to triangles or squares with 'anchored' edges, to make sure a pleasant not-discontinue Gouraud display is possible.

This process can be terminated with **ESC** during any phase.

SubSh P (NumBut) The number of times the environment is tested to detect Patches that need subdivision. (See option: "Subdiv Shoot Patch").

SubSh E (NumBut) The number of times the environment is tested to detect Elements that need subdivision. (See option: "Subdiv Shoot Element").

Convergence (NumBut) When the amount of unshot energy in an environment is lower than this value, the Radiosity solving stops. The initial unshot energy in an environment is multiplied by the area of the Patches. During each iteration, some of the energy is absorbed, or disappears when the environment is not a closed volume. In Blender's standard coordinate system a typical emitter (as in the example files) has a relatively small area. The convergence value in is divided by a factor of 1000 before testing for that reason.

Max iterations (NumBut) When this button has a non-zero value, Radiosity solving stops after the indicated iteration step.

Phase 4: Editing the solution

Once the Radiosity solution has been computed there ar still some actions to take (Figure 15-7).

Figure 15-7. Radiosity post process.

Element Filter (But) This option filters Elements to remove aliasing artefacts, to smooth shadow boundaries, or to force equalized colors for the "RemoveDoubles" option.

RemoveDoubles (But) When two neighbouring Elements have a displayed color that differs less than "Lim", the Elements are joined.

Lim (NumBut) This value is used by the previous button. The unit is expressed in a standard 8 bits resolution; a color range from 0 - 255.

FaceFilter (But) Elements are converted to faces for display. A "FaceFilter" forces an extra smoothing in the displayed result, without changing the Element values themselves.

Mult, Gamma (NumBut) The colorspace of the Radiosity solution is far more detailed than can be expressed with simple 24 bit RGB values. When Elements are converted to faces, their energy values are converted to an RGB color using the "Mult" and "Gamma" values. With the "Mult" value you can multiply the energy value, with "Gamma" you can change the contrast of the energy values.

Add New Meshes (But) The faces of the current displayed Radiosity solution are converted to Mesh Objects with vertex colors. A new Material is added that allows immediate rendering. *The input-Meshes remain unchanged.*

Replace Meshes (But) As previous, but the input-Meshes are removed.

Free Radio Data (But) All Patches, Elements and Faces are freed in Memory. You always must perform this action after using Radiosity to be able to return to normal editing.

Radiosity Juicy example

To get definitely away from dry theory and shows what Radiosity can really achieve let's look at an example.

This will actually show you a true Global Illumination scene, with smoother results than the 'Dupliverted Spot Lights' technique shown in the Lighting Chapter to attain something like Figure 15-8.

Figure 15-8. Radiosity rendered Cylon Raider.

Setting up.

We have only two elements in the scene at start up: a Cylon Raider (if you remember Galactica...) and a camera. The Raider has the default grey material, except for the main cockpit windows which are black. For this technique, we will not need any lamps.

The first thing that we will want to add to the scene is a plane. This plane will be used as the floor in our scene. Resize the plane as shown in Figure 15-9 and place it just under the Raider. Leave a little space between the plane and the Raider bottom. This will give us a nice "floating" look.

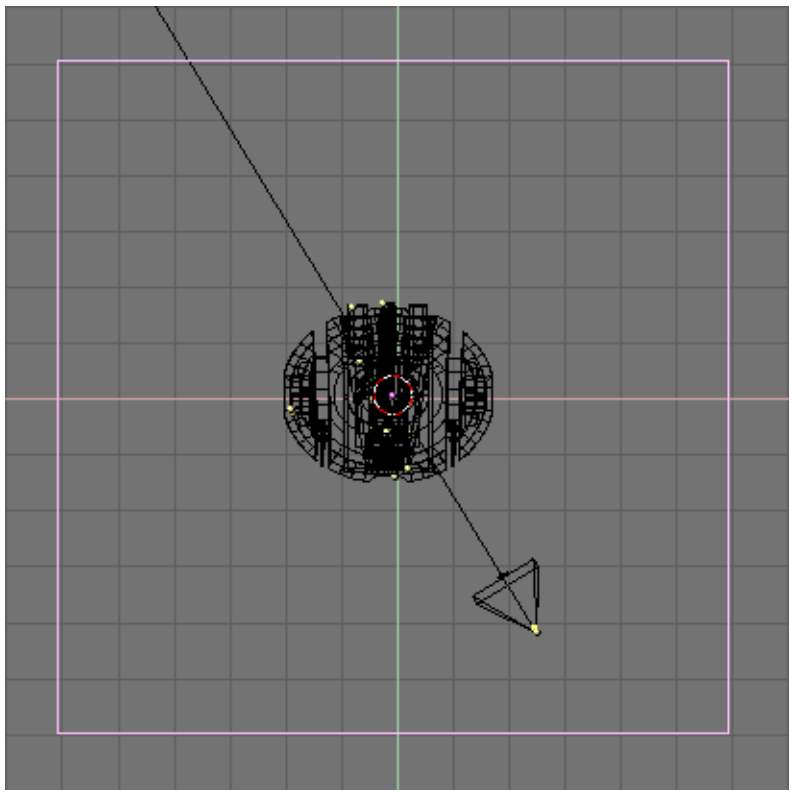


Figure 15-9. Add a plane

Next, you will want to give the plane a material and select a color for it. We will try to use a nice blue. You can use the setting in Figure 15-10 for it.

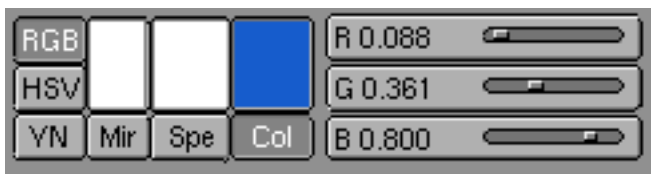


Figure 15-10. Plane colour

The Sky Dome

We want to make a GI rendering, so the next thing that we are going to add is an icosphere. This sphere is going to be our light source instead of the typical lamps. What we are going to do is use its faces as *emitters* that will project light for us in multiple directions instead of in one direction as with a typical, single, lamp. This will give us the desired effect.

To set this up, add an icosphere with a subdivision of 3. While still in EditMode, use the **BKEY** select mode to select the lower portion of the sphere and delete it. This will

leave us with our dome. Resize the dome to better fit the scene and match it up with your plane. It should resemble Figure 15-11.

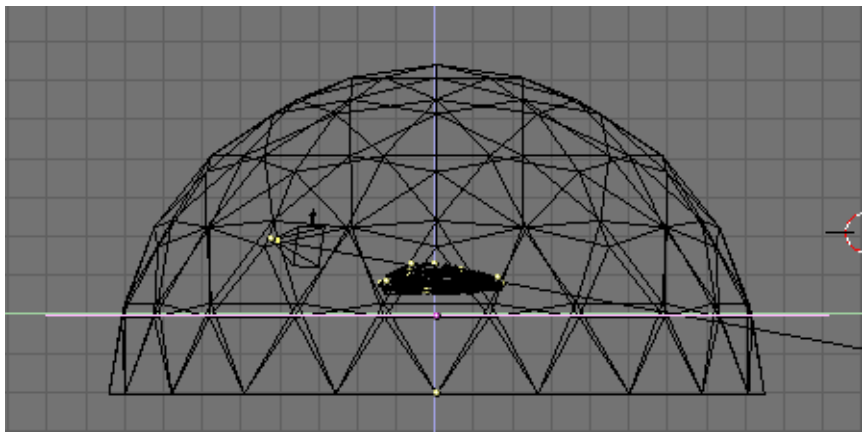


Figure 15-11. Sky dome.

Next, we want to make sure that we have all the vertices of the dome selected and then click on the EditButtons (**F9**) and select `Draw Normals`. This allows us to see in which direction the vertices are "emitting". By default it will be outside, so hit the `Flip Normals` button, which will change the vertex emitter from projecting outward to projecting inward in our dome (Figure 15-12).

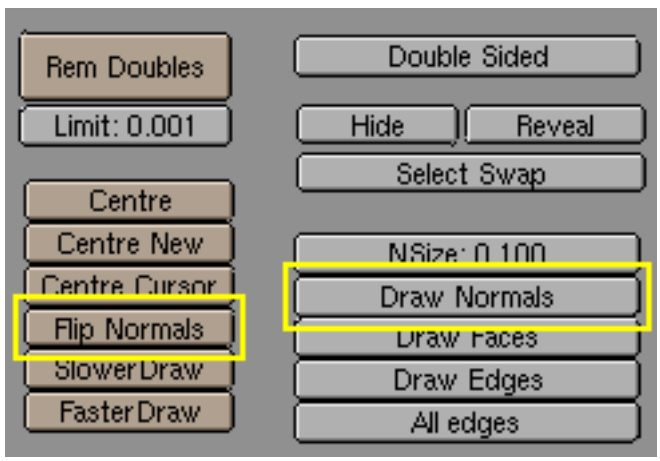


Figure 15-12. Sky dome.

Now that we have created our dome, we need a new material. When you create the material for the dome change the following settings in the MaterialButtons (**F5**):

`Add` = 0.000

`Ref` = 1.000

`Alpha` = 1.000

`Emit` = 0.020

The Emit slider here is the key. This setting controls the amount of light "emitted" from our dome. 0.020 is a good default. Remember that the dome is the bigger part of the scene! you don't want too much light! But you can experiment with this setting to get different results. The lower the setting here though the longer the "solve" time later. (Figure 15-13).

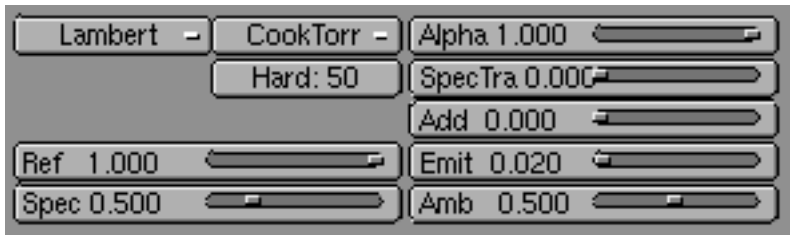


Figure 15-13. Sky dome material.

At this point we have created everything that we need for our scene. The next step will be to alter the dome and the plane from "double-sided" to "single-sided". To achieve this, we will select the dome mesh and then go back to the EditButtons (**F9**). Click the `Double Sided` button and turn it off (Figure 15-14). Repeat this process for the Plane.

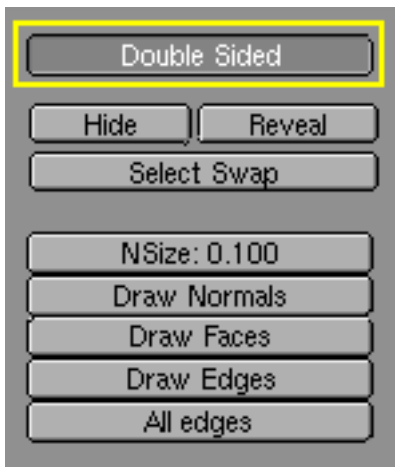


Figure 15-14. Setting Dome and plane 'single sided'.

The Radiosity solution

Now the next few steps are the heart and soul of Global Illumination. Go to side view with **NUM 3** and use **AKEY** to select all of the meshes in our scene. Next hold **SHIFT** and double click on your camera. We do not want this selected. It should look similar to Figure 15-15.

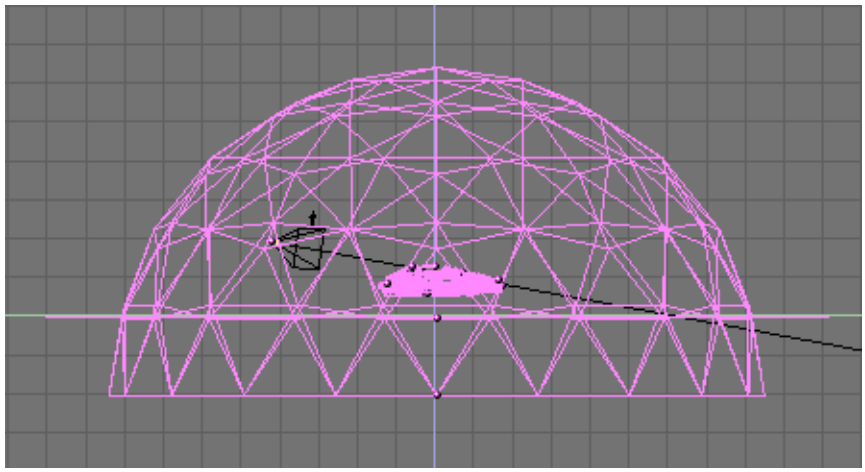


Figure 15-15. Selecting all Meshes.

After selecting the meshes, go to camera view with **NUM 0** and then turn on shaded mode with **ZKEY** so we can see inside our dome.

Now select the Radiosity Buttons ( ). On the left-hand side of the menu, click the `Collect Meshes` button. You should notice a change in your view in the colors. It should look similar to Figure 15-16.

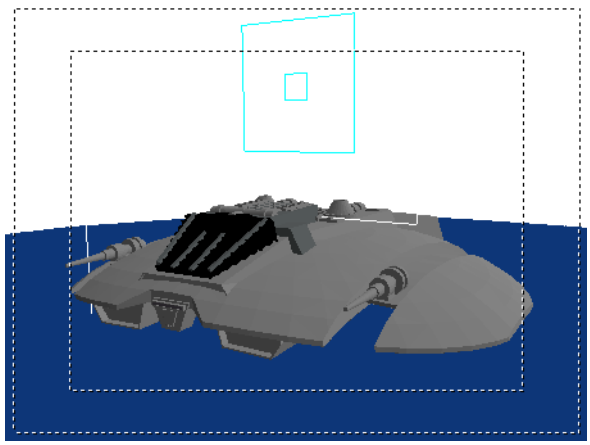


Figure 15-16. Preparing the Radiosity solution.

Next, to keep the Raider smooth like our original mesh, we will want to change from `Solid` to `Gour`. This will give our Raider its nice curves back, in the same way `Set Smooth` would in the EditButtons. You'll also need to change the `Max Subdiv Shoot` to 1 (Figure 15-17). Do not forget this step!

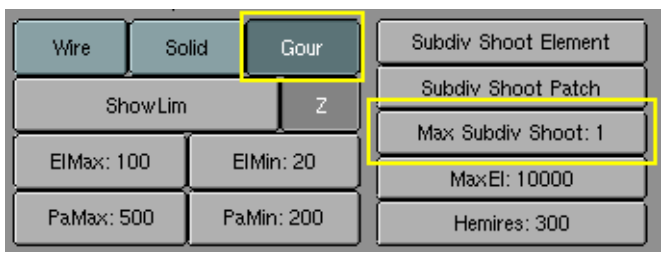


Figure 15-17. Radiosity settings.

After you have set `Gour` and `Max Subdiv Shoot`, click `Go` and wait. Blender will then begin calculating the emit part of the dome, going face by face, thus "solving" the render. As it does this, you will see the scene change as more and more light is added to the scene and the meshes are changed. You will also notice that the cursor in Blender changes to a counter much like if it were an animation. Let Blender run, solving the radiosity problem.

Letting Blender go to somewhere between 50-500 depending on the scene can do, for most cases. The solving time depends on you and how long you decide to let it run... remember you can hit **ESC** at any time to stop the process. This is an area that can be experimented with for different results. This can take from 5 to 10 minutes and your system speed will also greatly determine how long this process takes. Figure 15-18 is our Raider after 100.

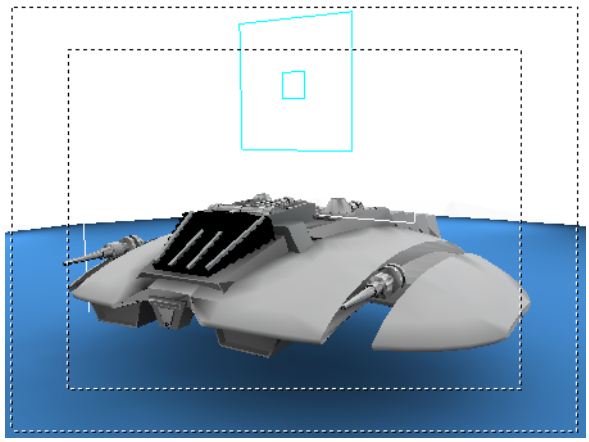


Figure 15-18. Radiosity solution.

After hitting the **ESC** key and stopping the solution, click `Replace Meshes` and then `Free Radio Data`. This finalizes our solve and replaces our previous scene with the new solved radiosity scene.

Now we are ready for **F12** and render (Figure 15-19).

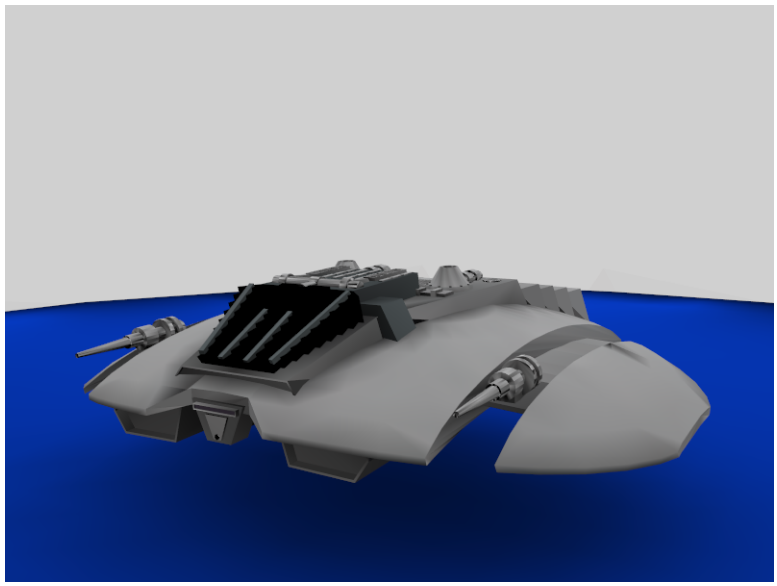


Figure 15-19. Rendering of the radiosity solution.

Texturing

There you go folks! You now have a very clean looking render with soft 360 degree lighting using radiosity. Very nice... But the next thing we want to do is add textures to the mesh. So go back to our main screen area.

Now try selecting your mesh and you will notice that it selects not only the Raider but the plane and dome as well. That is because Radiosity created a new *single* mesh through the solution process. To add a texture though, we only want the Raider.

So, select the mesh and then go into EditMode. In EditMode we can delete the dome and plane since they are no longer needed. You can use the **LKEY** to select the proper vertices and press **XKEY** to delete them. Keep selecting and deleting until you are left with only the Raider. It should look like in Figure 15-20. If we were to render it now with **F12**, we would get just a black background and our Raider. This is nice... but again, we want textures!

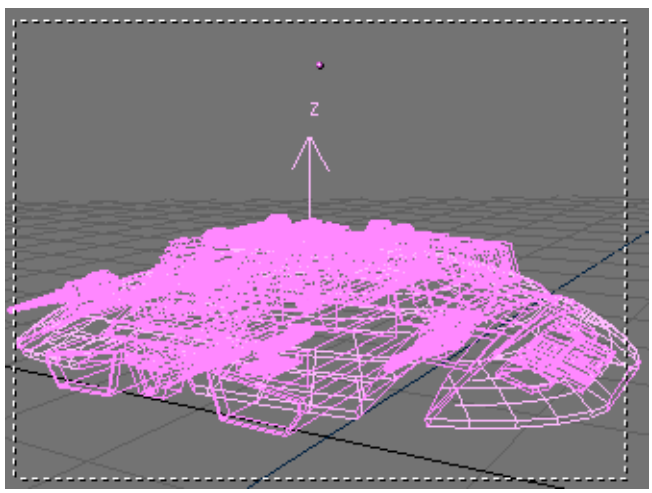


Figure 15-20. The Rider's mesh.

To add textures to mesh, we must separate out the areas that we are going to apply materials and textures to. For the Raider, We want to add textures to the wings and mid-section. To do this select the Raider mesh, and go back into EditMode. Select a vertex near the edge of the wing and then hit the **LKEY** to select linked vertices. Do the same on the other side. Next, click on the mid section of the ship and do the same thing. Select the areas shown in Figure 15-21. When you have those, hit the **PKEY** to separate the vertices selected.

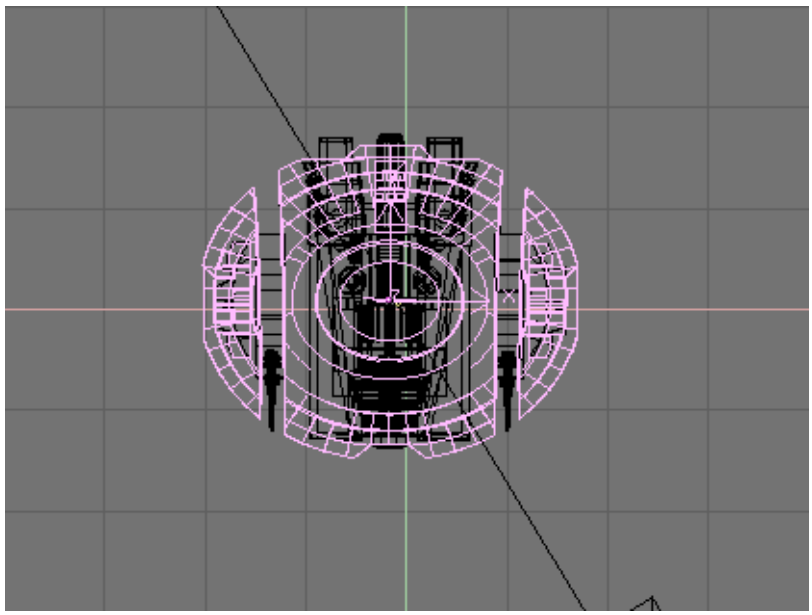


Figure 15-21. Separating the Rider parts to be textured.

We now have our wing section separate and are ready to add the materials and textures. We want to create a new material for this mesh. To get a nice metallic look, we can use the settings in Figure 15-22.

Figure 15-22. "Metallic" material.

Time to add the textures. We want to achieve some pretty elaborate results. We will need two bump-maps to create grooves and two mask for painting and 'decals'. There are hence four textures for the Raider wings to be created, as shown in Figure 15-23.

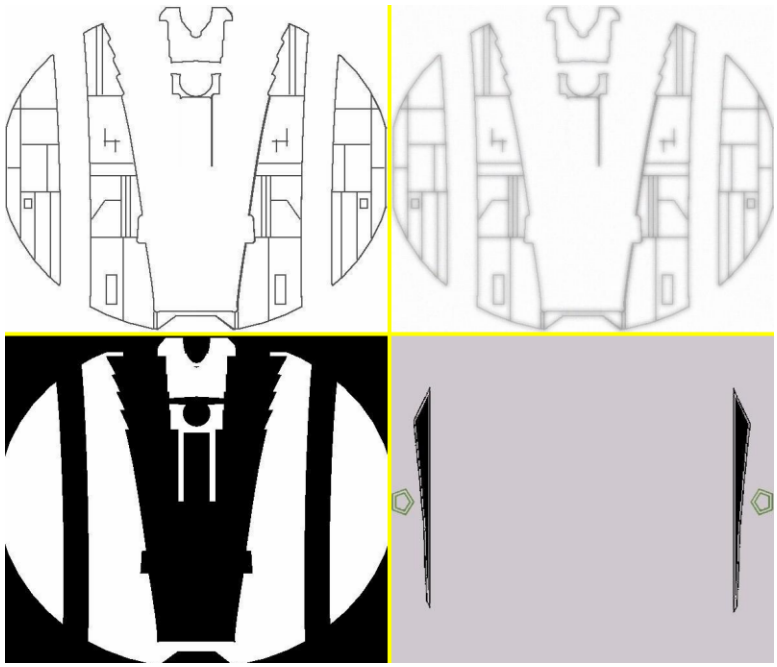


Figure 15-23. Four textures, from upper left corner, clockwise: RaiderBM, RaiderDI, Markings, Raider.

The textures should be placed in four material channels in the rider top mesh. 'RaiderBM' and 'RaiderDI' should be set to a negative `NOR` (Figure 15-24a -click `NOR` twice, it will turn yellow). 'Raider' should be set up as negative `REF` (Figure 15-24b).

> **Which material?:** A Mesh coming from a Radiosity solution tipically has more than one material on it. Theright one, the one where to add trtextures is the one callad *RadioMat*.

Figure 15-24. Texture set-ups.

The result is the desired metallic plating for the hull of the Raider. Finally the fourth texture, 'Markings', is set to `COL` in the MaterialButtons (Figure 15-24c). This will give the Raider its proper striping and insignia. Our rider is quite flat, so the `Flat` projection is adequate. Were it a more complex shape some UV mapping would have been required to attain good results. The material preview for the mesh should look like Figure 15-25.

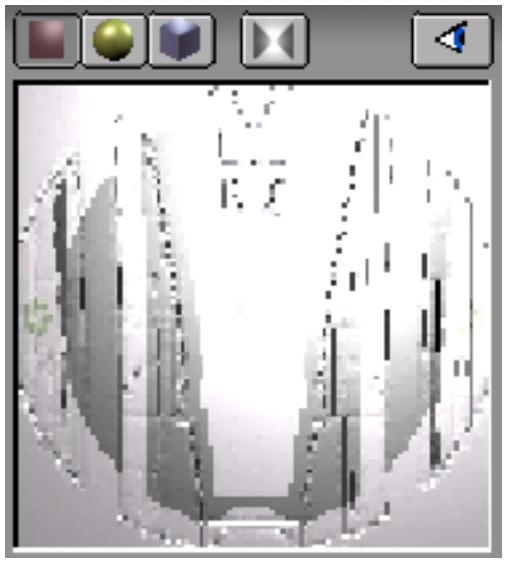


Figure 15-25. Complete material preview.

Our textures to *won't* show up in the rendering right now (except markings) because Nor and Ref type texture reacts to lighting, and there is no light source in the scene! Hence will now need to add a lamp or two, keeping in mind that our ship is still lit pretty well from the radiosity solve, so lamps energy should be quite weak. Once you have your lamps, you try a test render. Experiment with the lamps until you get the results you like.

The final rendering (Figure 15-8) shows a nice well lit Raider with soft texturing.

Chapter 16. Effects

Introduction

There are three kind of effects which can be linked to an Object working during animations.

Effects are added by selecting the object, switching to the Animation Buttons (**F7** or) and by pressing the `New Effect` button of Figure 16-1.

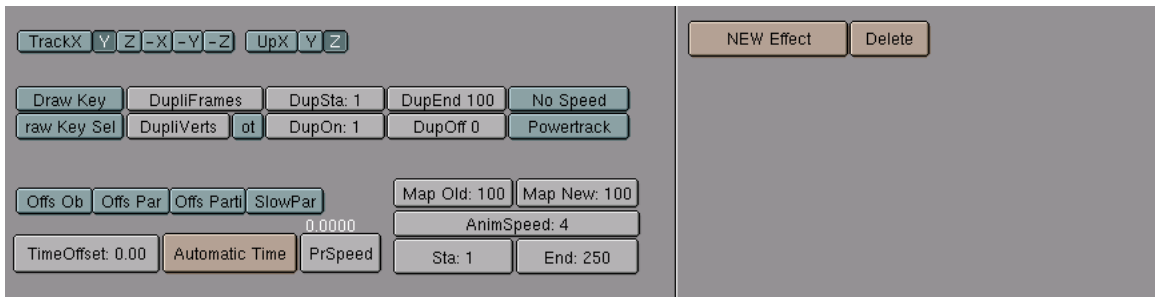


Figure 16-1. Animation Buttons Window

The `Delete` button removes an effect, if one is there, while the drop down list which appears on the right once an effect is added (Figure 16-2) selects the type of effect.

More than one effect can be linked to a single mesh. A row of small buttons, one for each effect, is created beneath the `New Effect` button, allowing you to switch from one to another to change settings.

The three effects are `Build`, `Particles` and `Wave`, the second being the most versatile. The following sections will describe each of them in detail.

Build Effect

The Build effect works on Meshes and causes the faces of the Object to appear, one after the other, over time. If the Material of the Mesh is a Halo Material, rather than a standard one, then the vertices of the Mesh, not the faces, appear one after another.

Figure 16-2. Build Effect

Faces, or vertices, appear in the order in which they are stored in memory. This order can be altered by selecting the Object and pressing **CTRL-F** out of EditMode. This causes faces to be re-sorted as a function of their value (Z co-ordinate) in the local reference of the Mesh.

Note on Reordering: If you create a plane and add the Build effect to see how it works you won't be happy. First, you must subdivide it so that it is made up of many faces, not just one. Then, pressing **CTRL-F** won't do much because the Z-axis is orthogonal to the plane. You must rotate it in EditMode to have some numerical difference between the co-ordinates of the faces, in order to be able to reorder them.

The Build effect only has two NumBut controls (Figure 16-2):

`Len` - Defines how many frames the build will take.

`Sfra` - Defines the start frame of the building process.

Particle Effects

The particle system of Blender is fast, flexible, and powerful. Every Mesh-object can serve as an emitter for particles. Halos (a special material) can be used as particles and with the Duplivert option, so can objects. These dupliverted objects can be any type of Blender object, for example Mesh-objects, Curves, Metaballs, and even Lamps. Particles can be influenced by a global force to simulate physical effects, like gravity or wind.

With these possibilities you can generate smoke, fire, explosions, fireworks, flocks of birds, or even schools of fish. With static particles you can generate fur, grass, and even plants.

A first Particle System

- Reset Blender to the default scene, or make a scene with a single plane added from the topview. This plane will be our particle emitter. Rotate the view so that you get a good view of the plane and the space above it (Figure 16-3).

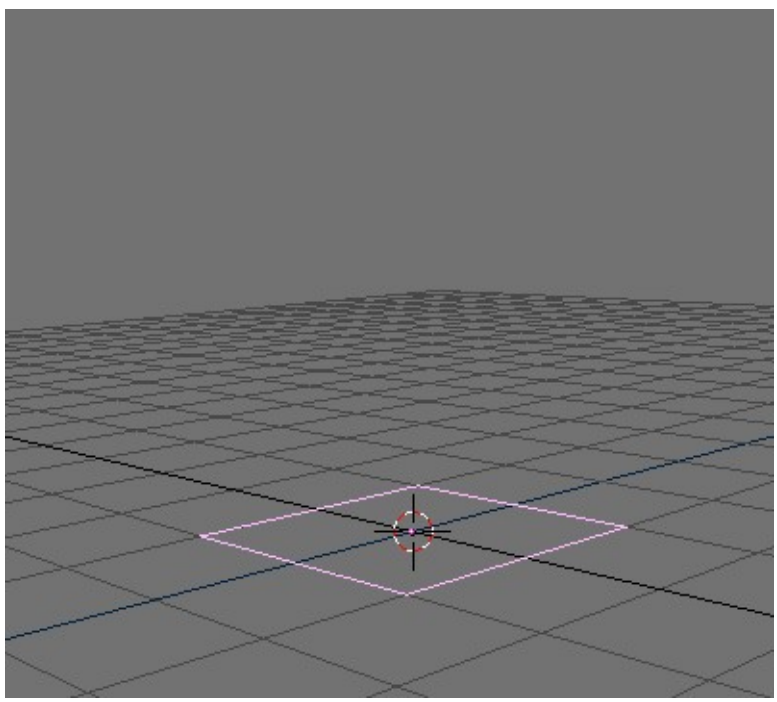


Figure 16-3. The emitter.

- Switch to the AnimButtons (**F7** or) and click the button "NEW Effect" in the middle part of the window. Change the dropdown MenuButton from `Build` to `Particles`. The ParticleButtons are shown in (Figure 16-4).

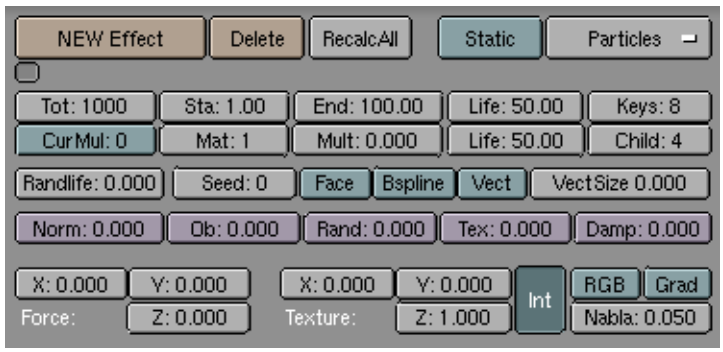


Figure 16-4. The Particle Buttons.

- Set the `Norm:` NumButton to 0.100 with a click on the right part of the button or use **SHIFT-LMB** to enter the value from the keyboard.
- Play the animation by pressing **ALT-A** with the mouse over the 3DWindow. You will see a stream of particles ascending vertically from the four vertices.

Congratulations - you have just generated your first particle-system in a few easy steps!

To make the system a little bit more interesting, it is necessary to get deeper insight on the system and its buttons (Figure 16-5):

- The parameter `Tot:` controls the overall count of particles. On modern speedy CPUs you can increase the particle count without noticing a major slowdown.
- The total number of particles specified in the `Tot:` button are uniformly created along a time interval. Such a time interval is defined by the `Sta:` and `End:` NumButtons, which control the time interval (in frames) in which particles are generated.
- Particles have a lifetime, they last a given number of frames, from the one they are produced in onwards, then disappear. You can change the lifetime of the particles with the `Life:` NumButton
- The `Norm:` NumButton used before made the particles having a starting speed of constant value (0.1) directed along the vertex normals. To make things more "random" you can set the `Rand:` NumButton to 0.1 too. This also makes the particles start whith random variation to the speed.
- Use the `Force:` group of NumButtons to simulate a constant force, like wind or gravity. A `Force: Z:` value of -0.1 will make the particles fall to the ground, for example.

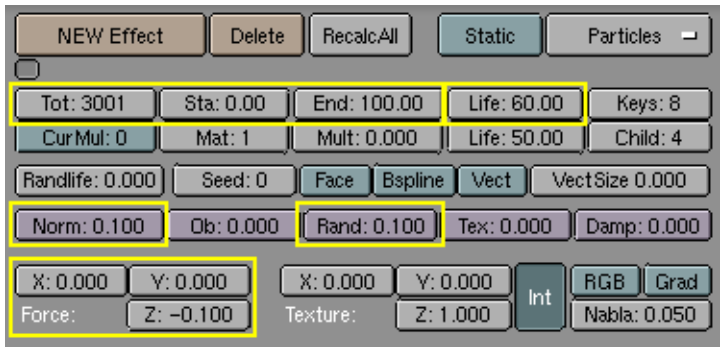


Figure 16-5. Particles settings.

This should be enough to get you started, but don't be afraid to touch some of the other parameters while you're experimenting. We will cover them in detail in the following sections.

Rendering a particle system

Maybe you've tried to render a picture from our example above. If the camera was aligned correctly, you will have seen a black picture with grayish blobby spots on it. This is the standard Halo-material that Blender assigns a newly generated particle system.

Position the camera so that you get a good view of the particle system. If you want to add a simple environment, remember to add some lights. The Halos are rendered without light, unless otherwise stated, but other objects need lights to be visible.

Go to the MaterialButtons (**F5**) and add a new material for the emitter if none have been added so far. Click the Button "Halo" from the middle palette (Figure 16-6).

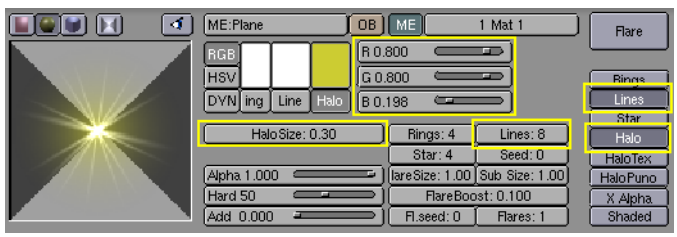


Figure 16-6. Halo settings

The MaterialButtons change to the HaloButtons. Choose `Line`, and adjust `Lines:` to a value of your choosing (you can see the effect directly in the Material-Preview). Decrease `HaloSize:` to 0.30, and choose a color for the Halo and for the lines (Figure 16-6).

You can now render a picture with **F12**, or a complete animation and see thousands of stars flying around (Figure 16-7).

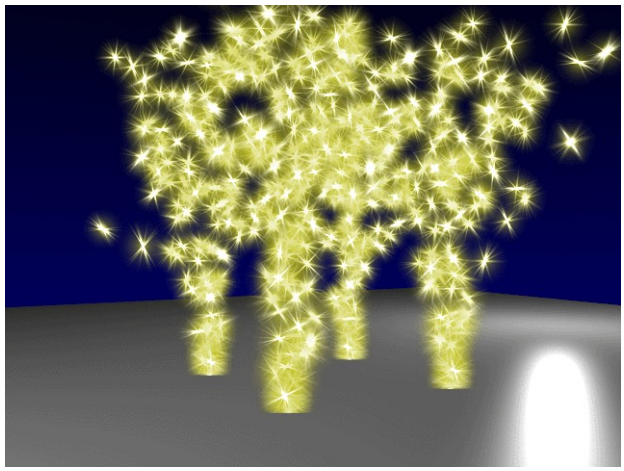


Figure 16-7. Shooting stars

Objects as particles

It is very easy to use real objects as particles, it is exactly like the technique described in the Section called *Dupliframes* in Chapter 17.

Start by creating a cube, or any other object you like, in your scene. It's worth thinking about how powerful your computer is, as we are going to have as many objects, as `Tot:` indicates, in the scene. This means having as many vertices as the number of vertices of the chosen object times the value of `Tot:`!

Scale the newly created object down so that it matches the general scene scale.

Now select the object, then **SHIFT-RMB** the emitter and make it the parent of the cube using **CTRL-P**. Select the emitter alone and check the option "DupliVerts" in the AnimationButtons (**F7**). The dupliverted cubes will appear immediately in the 3DWindow.

Figure 16-8. Setting Dupliverted Particles.

You might want to bring down the particle number before pressing **ALT-A** (Figure 16-8).

In the animation you will notice that all cubes share the same orientation. This can be interesting, but it can also be interesting to have the cubes randomly oriented.

This can be done by checking the option `Vect` in the particle-parameters, which causes the dupli-objects to follow the rotation of the particles, resulting in a more natural motion (Figure 16-8). One frame of the animation is shown in (Figure 16-9).

> **Original Object:** Take care to move the original object out of the cameraview, because it will also be rendered!

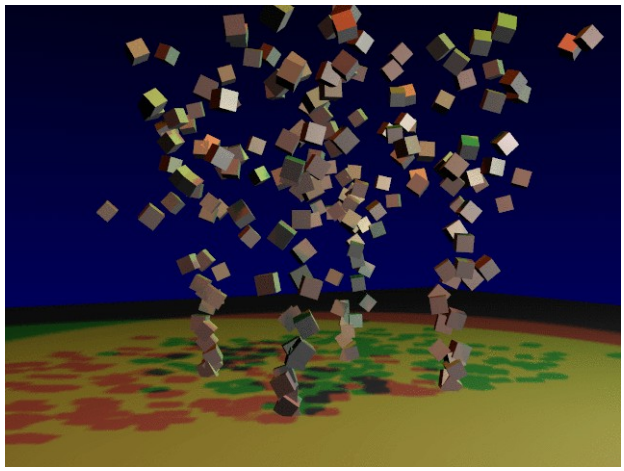


Figure 16-9. Dupliverted particles rendering.

Making fire with particles

The Blender particle system is very useful for making realistic fire and smoke. This could be a candle, a campfire, or a burning house. It's useful to consider how the fire is driven by physics. The flames of a fire are hot gases. They will rise because of their lower density when compared to the surrounding cooler air. Flames are hot and bright in the middle, and they fade and become darker towards their perimeter.

Prepare a simple setup for our fire, with some pieces of wood, and some rocks (Figure 16-10).

TODO

Figure 16-10. Campfire setup.

The particle system

Add a plane into the middle of the stone-circle. This plane will be our particle-emitter. Subdivide the plane once. You now can move the vertices to a position on the wood where the flames (particles) should originate.

Now go to the AnimationButtons **F7** and add a new particle effect to the plane. The numbers given here (Figure 16-11) should make for a realistic fire, but some modification may be necessary, depending on the actual emitter's size.

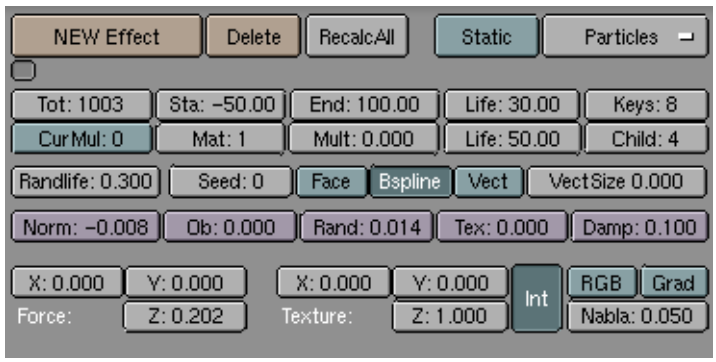


Figure 16-11. Fire particles setup.

Some notes:

- To have the fire burning from the start of the animation make `Sta:` negative. For example, try -50. The value of `End:` should reflect the desired animation length.
- The `Life:` of the particles is 30. Actually it can stay at 50 for now. We will use this parameter later to adjust the height of the flames.
- Make the `Norm:` parameter a bit negative (-0.008) as this will result in a fire that has a bigger volume at its basis.
- Use a `Force: Z:` of about 0.200. If your fire looks too slow, this is the parameter to adjust.
- Change `Damp:` to 0.100 to slow down the flames after a while.
- Activate the "Bspline"-button. This will use an interpolation method which gives a much more fluid movement.
- To add some randomness to our particles, adjust the `Rand:` parameter to about 0.014. Use the `Randlife:` parameter to add randomness in the lifetime of the particles; a really high value here gives a lively flame.
- Use about 600-1000 particles in total for the animation (`Tot:`).

In the 3DWindow, you will now get a first impression of how realistically the flames move. But the most important thing for our fire will be the material.

The fire-material

With the particle emitter selected, go to the MaterialButtons **F5** and add a new material. Make the new material a halo-material by activating the `Halo` button. Also, activate `HaloTex`, located just below this button. This allows us to use a texture later.

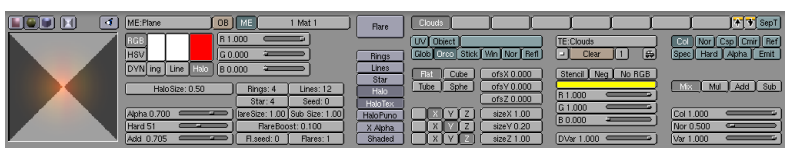


Figure 16-12. Flames Material.

Give the material a fully saturated red color with the RGB-sliders. Decrease the Alpha value to 0.700; this will make the flames a little bit transparent. Increase the `Add` slider up to 0.700, so the Halos will boost each other, giving us a bright interior to the flames, and a darker exterior. (Figure 16-12).

Figure 16-13. Flames Texture.

If you now do a test render, you will only see a bright red flame. To add a touch more realism, we need a texture. While the emitter is still selected, go to the TextureButtons **F6**. Add a new Texture and select the `Cloud`-type. Adjust the "NoiseSize:" to 0.600. (Figure 16-13).

Go back to the MaterialButtons **F5** and make the texture-color a yellow color with the RGB sliders on the right side of the material buttons. To stretch the yellow spots from the cloud texture decrease the "SizeY" value down to 0.30.

A test rendering will now display a nice fire. But we still need to make the particles fade out at the top of the fire. We can achieve this with a material animation of the Alpha and the Halo Size.

An animation for a particle material is always mapped from the first 100 frames of the animation to the lifetime of a particle. This means that when we fade out a material in frame 1 to 100, a particle with a lifetime of 50 will fade out in that time.

Be sure that your animation is at frame 1 (**SHIFT-LEFTARrOW**) and move the mouse over the MaterialWindow. Now press **IKEY** and choose Alpha from the appearing menu. Advance the frame-slider to frame 100, set the Alpha to 0.0 and insert another key for the Alpha with **IKEY**. Switch one Window to an IPOWindow. Activate the MaterialIPOs by clicking on the sphere-icon in the IPOHeader (). You will see one curve for the Alpha-channel of the Material (Figure 16-14).

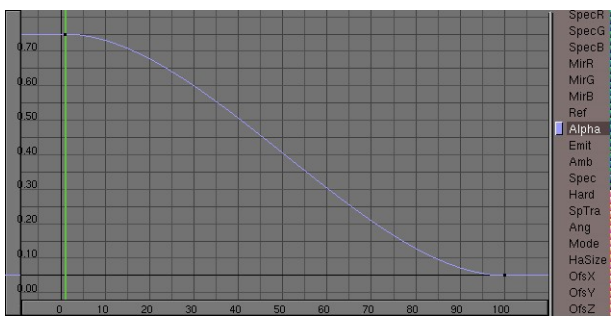


Figure 16-14. Fire Material IPO

Now you can render an animation. Maybe you will have to fine-tune some parameters like the life-time of the particles. You can add a great deal of realism to the scene by animating the lights (or use shadow-spotlights) and adding a sparks particle-system to the fire. Also recommended is to animate the emitter in order to get more lively flames, or use more than one emitter (Figure 16-15).

Figure 16-15. Final rendering.

A simple explosion

This explosion is designed to be used as an animated texture, for composing it with the actual scene or for using it as animated texture. For a still rendering, or a slow motion of an explosion, we may need to do a little more work in order to make it look really good. But bear in mind, that our explosion will only be seen for half a second (Figure 16-16).

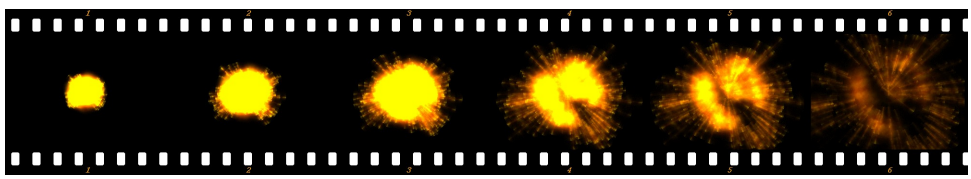


Figure 16-16. The explosion

As emitter for the explosion I have chosen an IcoSphere. To make the explosion slightly irregular, I deleted patterns of vertices with the circle select function in Edit-Mode. For a specific scene it might be better to use an object as the emitter, which is shaped differently, for example like the actual object you want to blow up.

My explosion is composed from two particle systems, one for the cloud of hot gases and one for the sparks. I took a rotated version of the emitter for generating the sparks. Additionally, I animated the rotation of the emitters while the particles were being generated.

The materials

The particles for the explosion are very straightforward halo materials, with a cloud texture applied to add randomness, the sparks too have a very similar material, see Figure 16-17 to Figure 16-19.

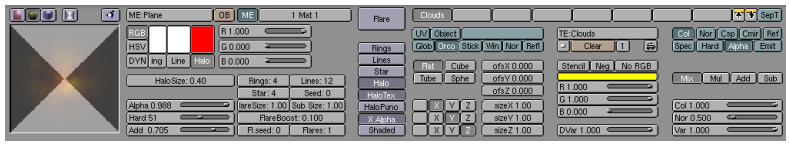


Figure 16-17. Material for the explosion cloud.

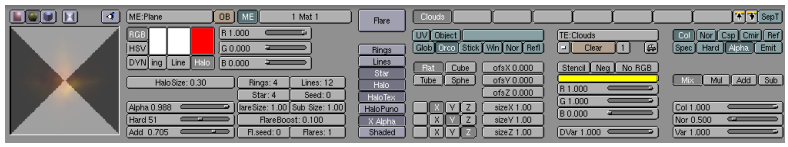


Figure 16-18. Material for the sparks.

Figure 16-19. Texture for both.

Animate the Alpha-value of the Haloparticles from 1.0 to 0.0 at the first 100 frames. This will be mapped to the life-time of the particles, as is usual. Notice the setting of `Star` in the sparks material (Figure 16-18). This shapes the sparks a little bit. We could have also used a special texture to achieve this, however, in this case using the "Star" setting is the easiest option.

The particle-systems

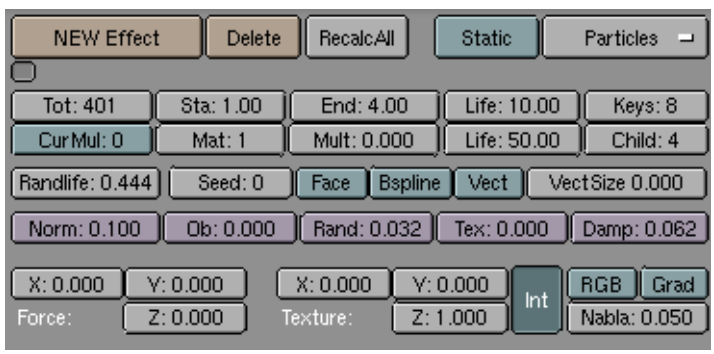


Figure 16-20. Particle system for the cloud

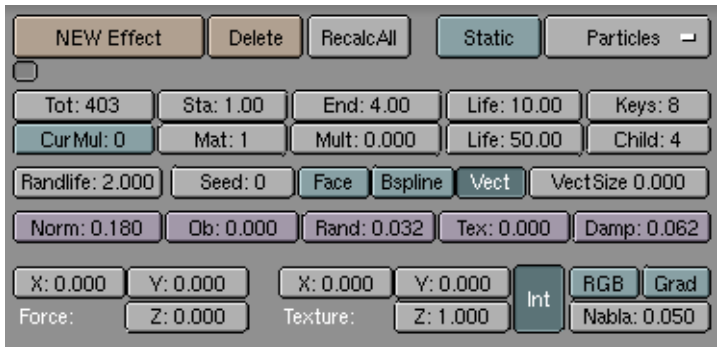


Figure 16-21. Particle system for the sparks

As you can see in (Figure 16-20) and (Figure 16-21), the parameters are basically the same. The difference is the `Vect` setting for the sparks, and the higher setting of `Norm:` which causes a higher speed for the sparks. I also set the `Randlife:` for the sparks to 2.000 resulting in an irregular shape.

I suggest that you start experimenting, using these parameters to begin with. The actual settings are dependent on what you want to achieve. Try adding more emitters for debris, smoke, etc.

Fireworks

A button we have not used so far is the `Mul:` button, located with the particle buttons. The whole third line of buttons is related to this. Prepare a plane and add a particle system to the plane.

Adjust the parameters so that you get some particles flying into the sky, then increase the value of `Mult:` to 1.0. This will cause 100% of the particles to generate child particles when their life ends. Right now, every particle will generate four children. So we'll need to increase the `Child:` value to about 90 (Figure 16-22). You should now see a convincing firework made from particles, when you preview the animation with **ALT-A**.

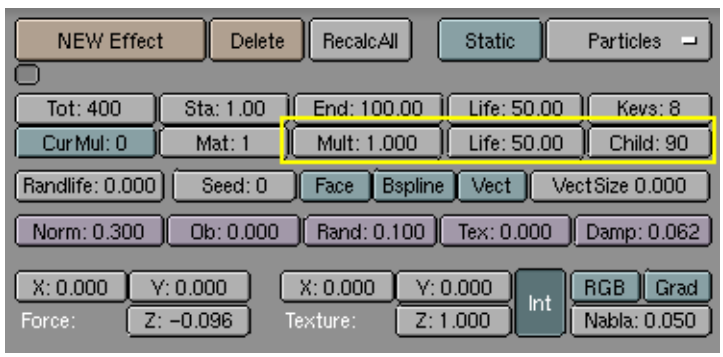


Figure 16-22. Particle Multiplication buttons

When you render the firework it will not look very impressive. This is because of the standard halo material that Blender assigns. Consequently, the next step is to assign a better material.

Ensure that you have the emitter selected and go to the MaterialButtons **F5**. Add a new material with the MenuButton, and set the type to `Halo`.

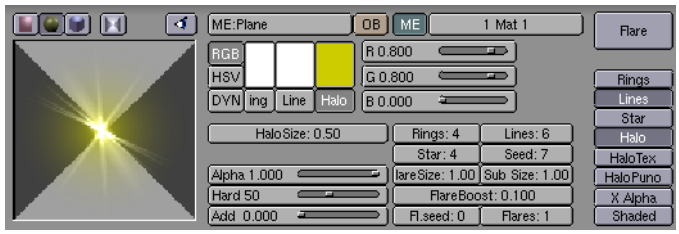


Figure 16-23. Firework Material 1

I have used a pretty straightforward halo material; you can see the parameters in Figure 16-23. The rendered animation will now look much better, yet there is still something we can do.

While the emitter is selected go to the EditButtons **F9** and add a new material index by clicking on the `New` button (Figure 16-24).

Figure 16-24. Adding a second material to the emitter.

Now switch back to the MaterialButtons. You will see that the material data browse in the header has changed color to blue. The button labelled "2" indicates that this material is used by two users. Now click on the "2" button and confirm the popup. Rename the Material to "Material 2" and change the color of the halo and the lines (Figure 16-25).

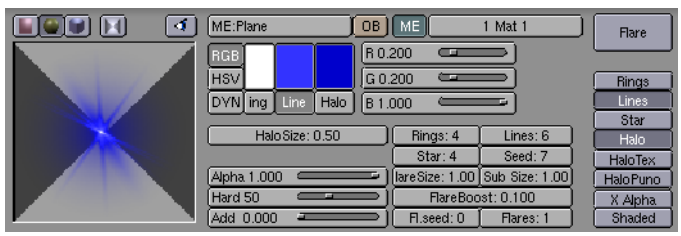


Figure 16-25. Material 2

Switch to the particle parameters and change the `Mat:` button to "2". Render again and you see that the first generation of particles is now using the first material and the second generation the second material! This way you can have up to 16 (that's the maximum number of material indices) materials for particles.

> **Further enhancements:** Beside changing materials you also can use the material IPOs to animate material settings of each different material.

A shoal of fish

Now, we will create a particle system that emits real objects. This kind of particle system can be used to make shrapnel for explosions, or animate groups of animals. We will use the fish from the UV-Texturing tutorial, to create a shoal of fish that can be used to add some life and motion to underwater scenes.

The emitter

Switch to layer three (**3KEY**) to hide the layers with the environment, and add a plane in the sideview window at the 3D-cursor location. Without leaving EditMode, subdivide the plane two times and then leave EditMode.

Go to the AnimationButtons **F7** and add a particle effect to the plane.

Figure 16-26. Fish "Emitter" settings.

Set up your emitter as shown in the picture. I used 30 as total number of particles, and I stopped the generation at frame 30. This is so that every second a new particle is generated. A small amount of randomness should be used. The lifetime of the particles should be long enough to make sure that the particles don't vanish in front of the camera. Activate the Bspline and Vect options; these become important later (Figure 16-26).

Now we have to recover the fish from the UV-Texture tutorial. Press **SHIFT-F1** to append the fish from its file. It will appear textured in the camera view if you have set it to textured mode (**ALT-Z**). If it is too big, scale it down and then move it out of the camera view.

Select the fish and extend your selection to the particle emitter. Press **CTRL-P** to make the emitter the parent of the fish. Now select only the emitter and go to the AnimButtons (**F7**) and switch on Dupliverts. Instances of the fish will appear at the position of every single particle. In case the fish is oriented wrong, select the base object and do a clear rotation with **Alt-R**. Now you can play back the animation in the camera view to see how the fish are moving. Experiment a bit with the particle setting until you get a realistic looking shoal of fish.

Using a Lattice to control the particles

Create a Lattice with the Toolbox. Scale it so that it just covers the shoal of fish. Switch to the EditButtons (**F9**)and set the "U:" resolution of the lattice to something approaching 10 (Figure 16-27). Then select the emitter, extend your selection with the Lattice, and make it the parent of the emitter. You can now deform the Lattice and the particle system will follow. After you have changed something, leave EditMode and do a "Recalc All" for the particle system. This will update them.

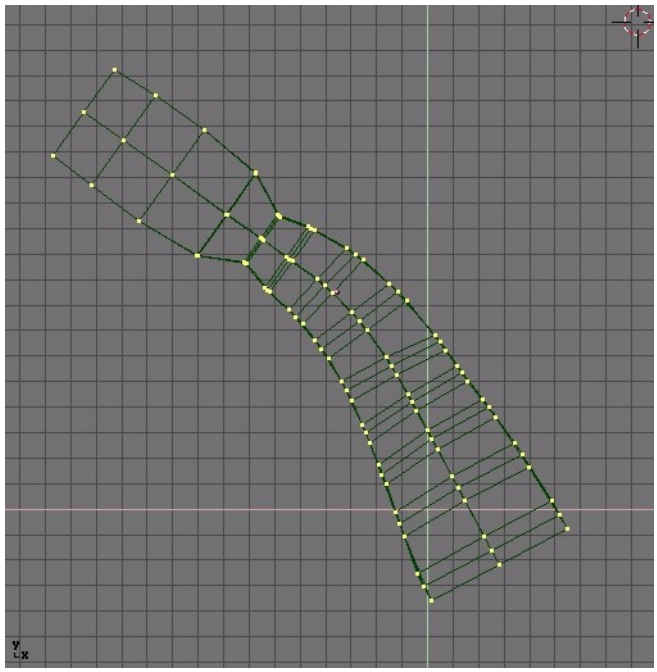


Figure 16-27. Lattice to deform particles path.

With the Lattice you can make curved paths for the fish, or make the shoal extend and join by scaling certain areas of the lattice. (Figure 16-28).

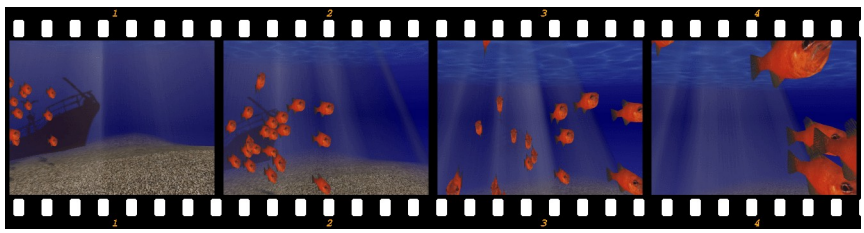


Figure 16-28. Frames from completed animation

Static Particles (-)

Static particles are useful when making objects like fibers, grass, fur and plants.

Wave Effect

The Wave effect adds a motion to the Z co-ordinate of the Object Mesh.

Figure 16-29. Wave Control Panel

The wave effect influence is generated from a given starting point defined by the Sta X and Sta Y NumButs. These co-ordinates are in the Mesh local reference (Figure 16-30).

Figure 16-30. Wave Origin

The Wave effect deformation originates from the given starting point and propagates along the Mesh with circular wavefronts, or with rectilinear wavefronts, parallel to the X or Y axis. This is controlled by two `X` and `Y` toggle buttons. If just one button is pressed fronts are linear, if both are pressed fronts are circular (Figure 16-31).

The wave itself is a gaussian-like ripple which can be either a single pulse or a series of ripples, if the Cycl button is pressed.

Figure 16-31. Wave front type

The Wave is governed by two series of controls, the first defining the Wave form, the second the effect duration.

For what concerns Wave Form, controls are `Speed`, `Height`, `Width` and `Narrow` (Figure 16-32).

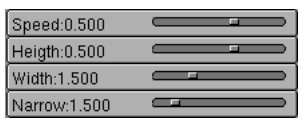


Figure 16-32. Wave front controls

The `Speed` Slider controls the speed, in Units per Frame, of the ripple.

The `Height` Slider controls the height, in Blender Units and along Z, of the ripple (Figure 16-33).

If the Cycl button is pressed, the `Width` Slider states the distance, in Blender Units, between the topmost part of two subsequent ripples, and the total Wave effect is given by the envelope of all the single pulses (Figure 16-33).

This has an indirect effect on the ripple amplitude. Being ripples gaussian in shape, if the pulses are too next to each other the envelope could not reach the z=0 quote any more. If this is the case Blender actually lowers the whole wave so that the minimum is zero and, consequently, the maximum is lower than the expected amplitude value, as shown in Figure 16-33 at the bottom.

The actual width of each gaussian-like pulse is controlled by the `Narrow` Slider, the higher the value the narrower the pulse. The actual width of the area in which the single pulse is significantly non-zero in Blender Units is given by 4 over the Narrow Value. That is, if Narrow is 1 the pulse is 4 Units wide, and if Narrow is 4 the pulse is 1 Unit Wide.

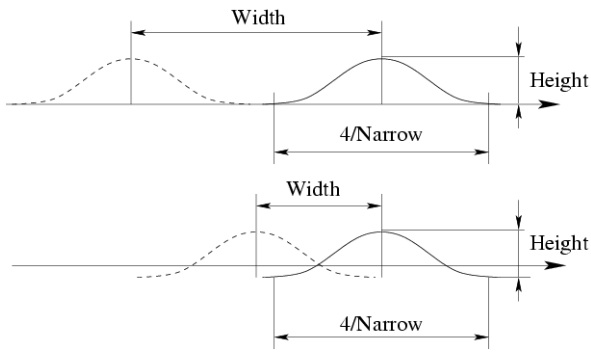


Figure 16-33. Wave front characteristics

> **To obtain a Sinusoidal-like wave:** To obtain a nice Wave effect similar to sea waves and close to a sinusoidal wave it is necessary that the distance between following ripples and the ripple width are equal, that is the "Width" Slider value must be equal to 4 over the "Narrow" Slider value.

The last Wave controls are the time controls. The three NumButs define:

`Time sta` the Frame at which the Wave begins;

`Lifetime` the number of frames in which the effect lasts;

`Damptime` is an additional number of frames in which the wave slowly dampens from the Amplitude value to zero. The Dampening occurs for all the ripples and begins in the first frame after the "Lifetime" is over. Ripples disappear over "Damptime" frames.

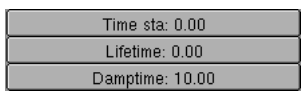


Figure 16-34. Wave time controls

Chapter 17. Special modelling techniques

by Malefico

Introduction

Once we have overcome the "extrusion modelling fever" and started to look at more challenging modelling targets, we might start the searching for alternative methods to do the job. There are a group of modelling techniques in Blender which not only make our modelling job easier but sometimes make it POSSIBLE.

These so called "special" modelling techniques involve not only some vertex manipulation but the use of non-intuituive procedures which require a deeper knowledge or experience from the user than the average beginner.

In this chapter we will describe these techniques in detail and explain their utility in several modelling applications which could not have been solved any other way.

Dupliverts

"Dupliverts" are not a rock band nor a dutch word for something illegal (well maybe it is) but is a contraction for "DUPLIcation at VERTiceS", meaning the duplication of a base object at the location of the vertices of a mesh. In other words, when using Dupliverts on a mesh, on every vertex of it an instance of the base object is placed.

There are actually two approaches to modelling using Dupliverts. Mainly they can be used as an arranging tool, allowing us to model geometrical arrangement of objects (eg: the columns of a greek temple, the trees in a garden, an army of robot soldiers, the desktops in a classroom). The object can be of any object type which Blender supports.

The second approach is using them to model an object starting from a single part of it (eg: the spikes in a club, the thorns of a sea-urchin, the tiles in a wall, the petals in a flower)

We are going to discuss both approaches to examine all our options

Dupliverts as arranging tool

All you need is a base object (eg: the "tree" or the "column") and a mesh with its vertices following the pattern you have in mind.

I will use a simple scene for the following part. It consists of a camera, the lamps, a plane (for the floor) and a strange man I modelled after a famous Magritte's character. If you don't like surrealism you will find this part extremely boring.

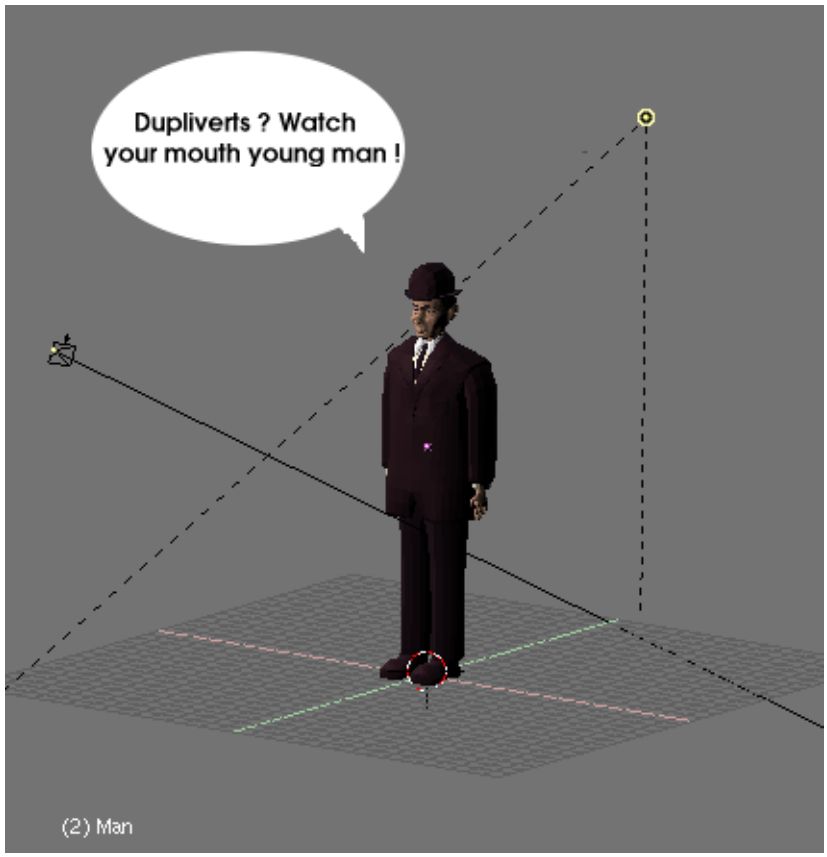


Figure 17-1. A simple scene to play with

Anyway, the man will be my "base object". It is a good idea that he will be at the center of coordinates, and with all rotations cleared. Move the cursor to the base object's center, and From Top View add a mesh circle, with 12 vertices or so.

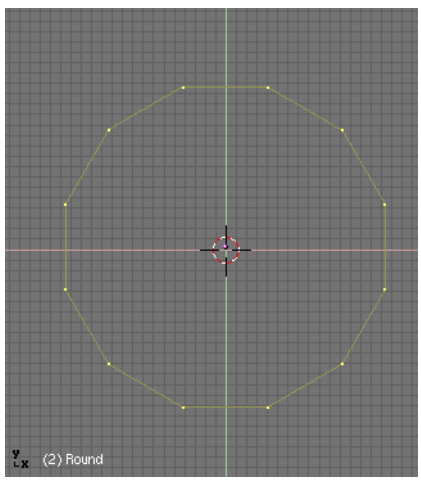


Figure 17-2. The parent mesh can be any primitive

Out of Edit Mode, select the base object and add the circle to the selection (order is very important here). Parent the base object to the circle by pressing **CTRL-P**. Now, the circle is the parent of the character. We are almost done.

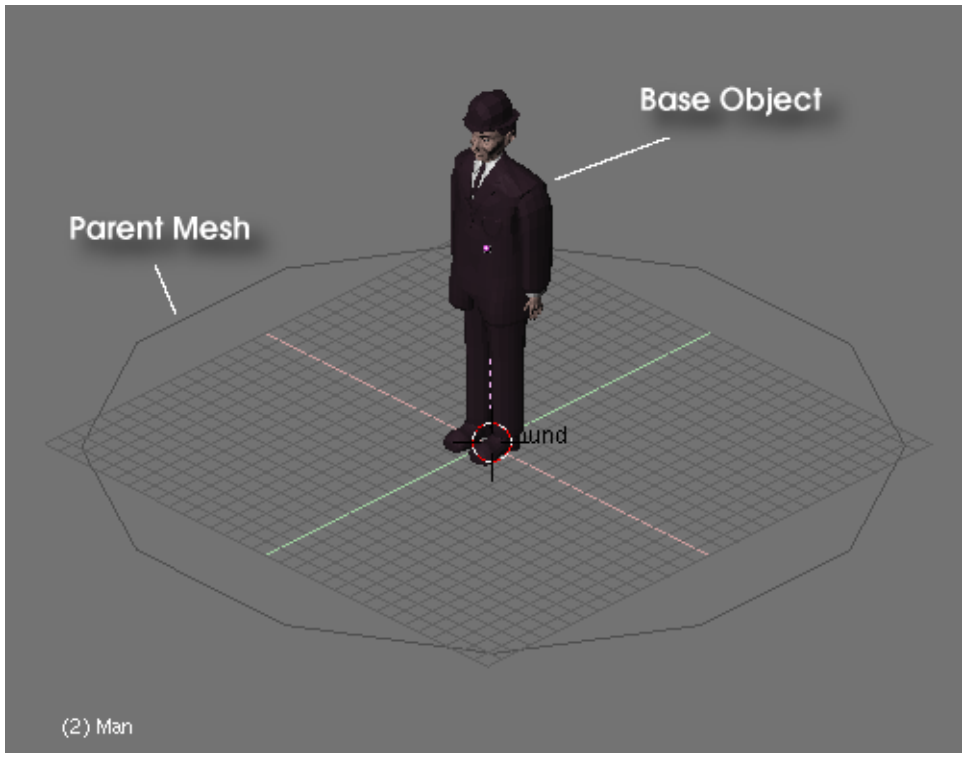


Figure 17-3. The man is parented to the circle

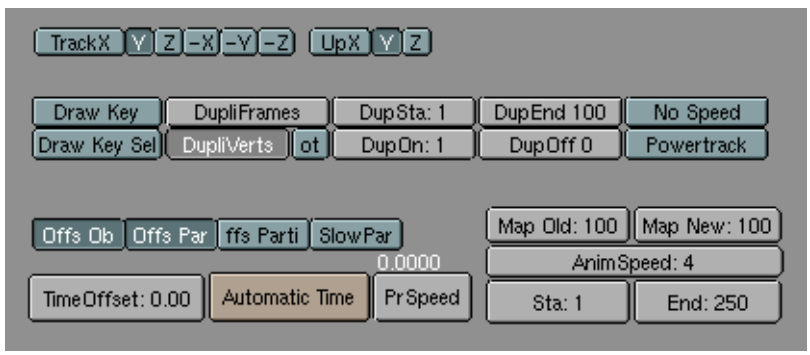


Figure 17-4. The Animation Buttons

Now select only the circle, switch the ButtonsWindow to the AnimButtons (**F7**) and select the option“DupliVerts”.

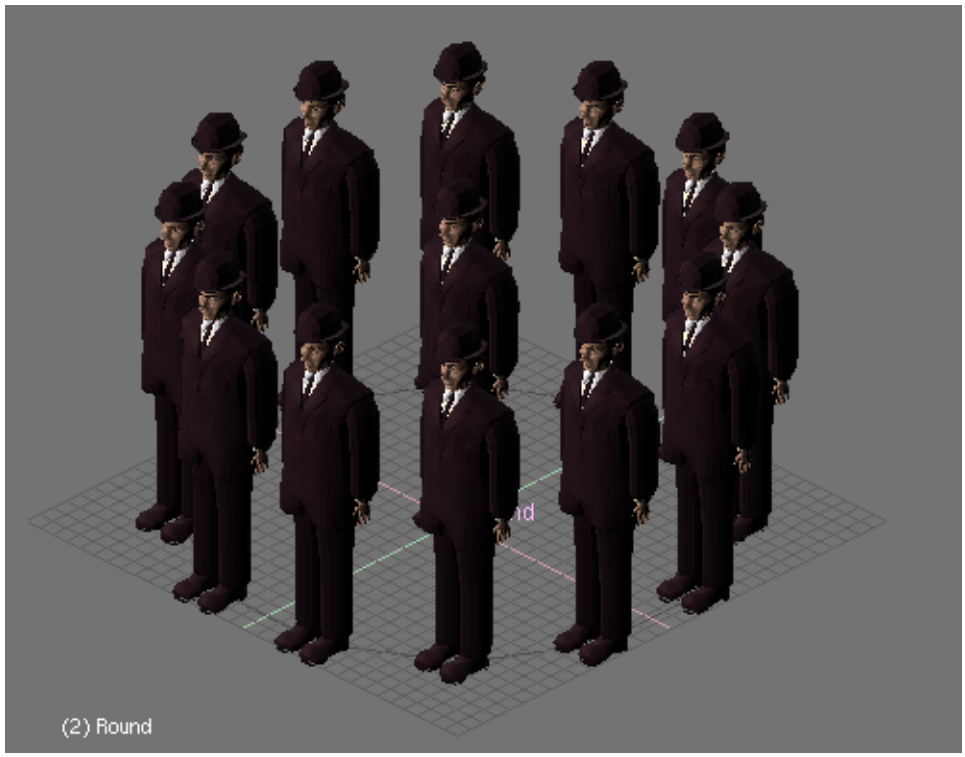


Figure 17-5. In every vertex of the circle a man is placed

WOW, isn't it great ?. Don't worry about the object at the center. It is still shown in the 3D-views, but it will *NOT* be rendered. You can now select the base object, change (scale, rotate, EditMode)[1] it and all dupliverted objects will reflect the changes. But the more interesting thing to note is that you can also edit the parent circle.

Select the circle and scale it. You can see that the misterious men are uniformly scaled with it. Now enter the EditMode for the circle, select all vertices **AKEY** and scale it up about three times. Leave EditMode and the dupliverted objects will update. This time they will still have their original size but the distance between them will have changed. Not only we can scale in EditMode, but we can also delete or add vertices to change the arrangement of men.

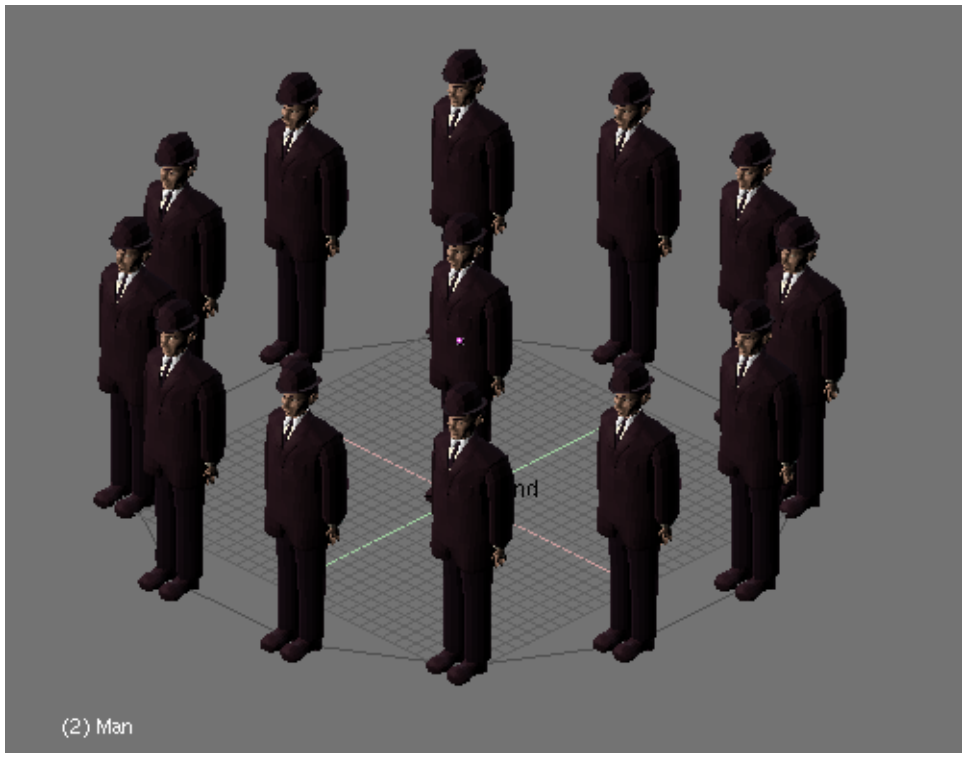


Figure 17-6. Changing the size of the circle in Edit Mode

Select all vertices and duplicate them. Now scale the new vertices outwards to get a second circle around the original. Leave Edit Mode, and a second circle of men will appear.

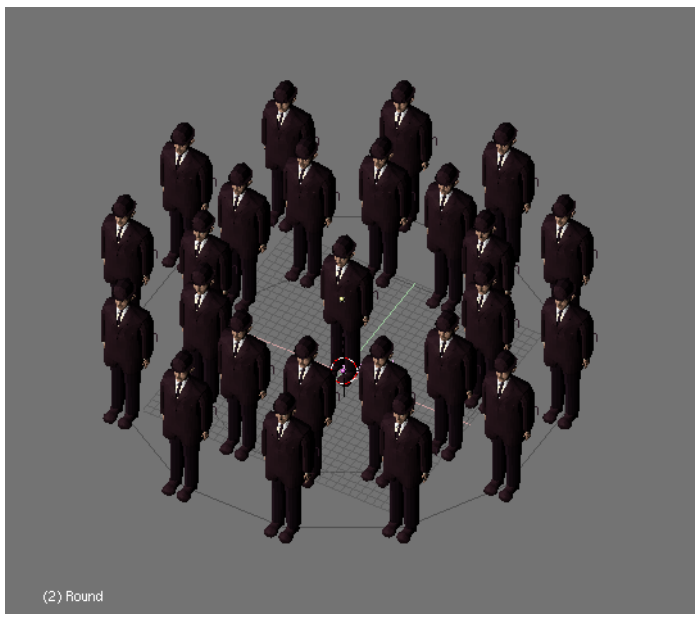


Figure 17-7. A second row of Magritte's men

Until now all Magritte's men were facing the camera, ignoring each other. We can get more interesting results using the "Rot" option next to the duplivert button. With this option active, we can rotate the dupliverted objects according to the face-normals of

the parent object. More precisely, the dupliverted objects axis are aligned with the normal at the vertex location.

Which axis is aligned (X, Y or Z) depends on what is indicated in the TrackX,Y,Z buttons and the UpX,Y,Z buttons. Trying this with our surrealist buddies, will lead to wierd results depending on these settings.

The best way to figure out what will happen is first of all aligning the "base" and "parent" objects' axis with the World axis. This is done selecting both objects and pressing **CTRL-A**, and click the "Apply Size/Rot?" menu.

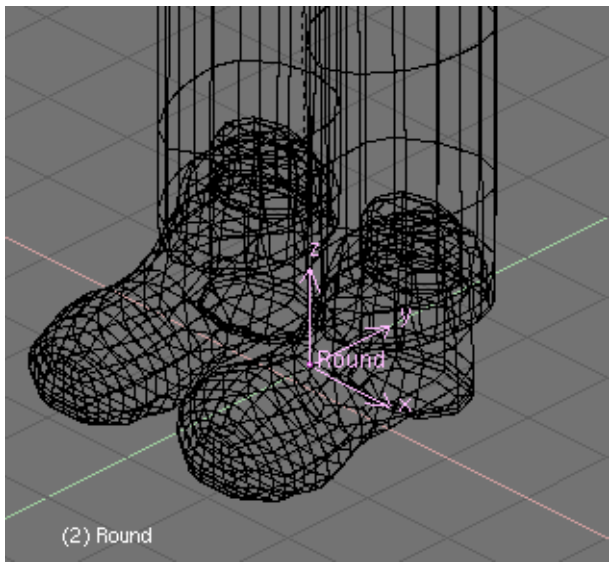


Figure 17-8. Show object's axis to get what you want

Then make the axis of the base object and the axis and normals in the parent object visible (in this case, being a circle with no faces, a face must be defined first for the normal to be visible (actually to exist at all))

Now select the base object (our magritte's man) and play a little with the Anim buttons. Note the different alignment of the axis with the different combinations of UpX,Y,Z and TrackX,Y,Z.

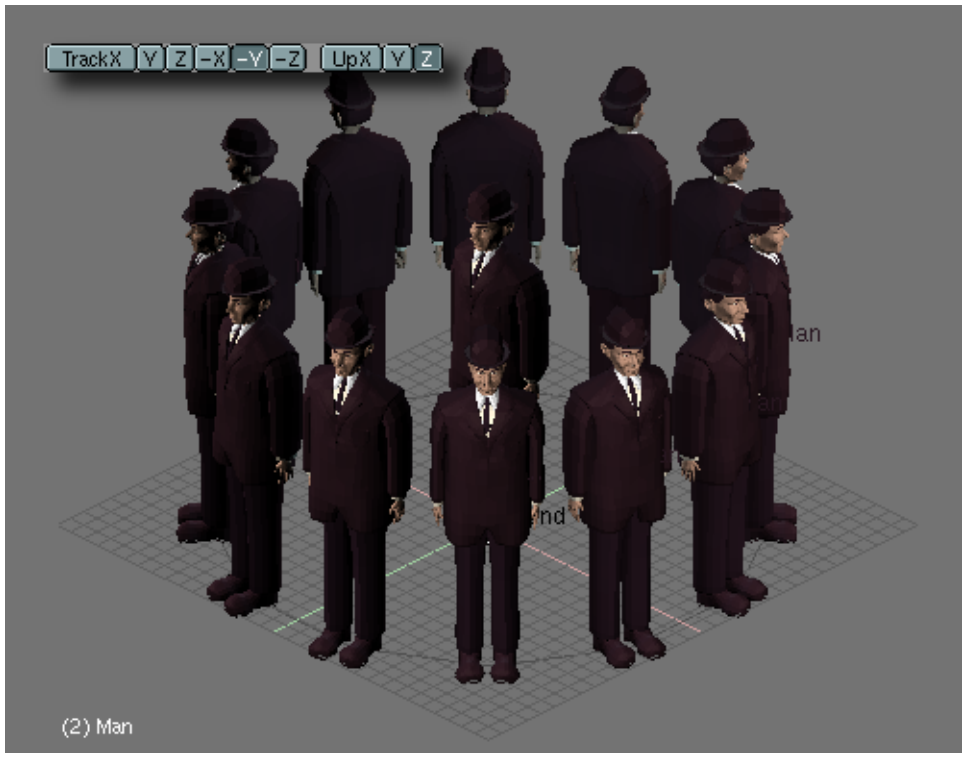


Figure 17-9. Negative Y Axis is aligned to vertex normal (pointing to the circle's center)

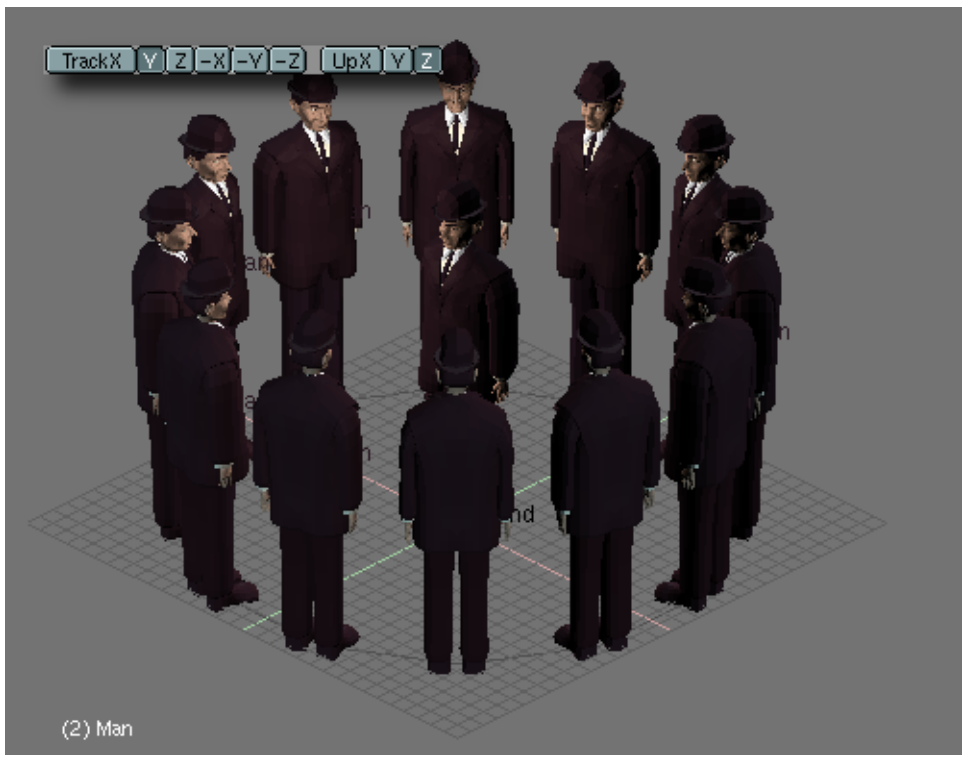


Figure 17-10. Positive Y axis is aligned to normal

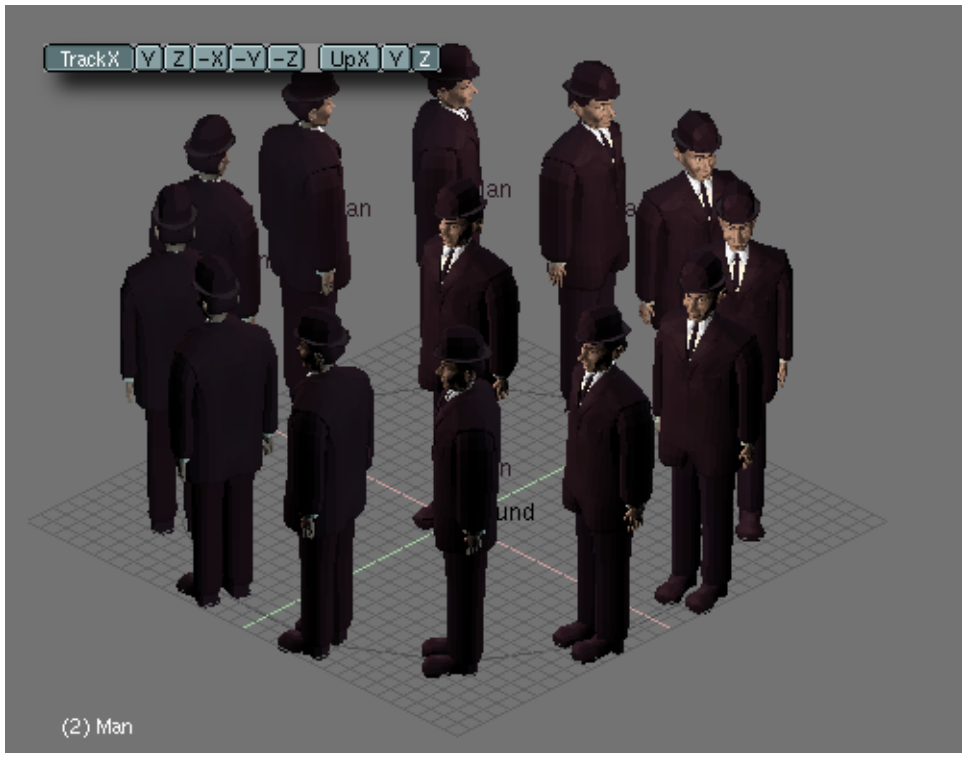


Figure 17-11. Positive X axis is aligned to normal

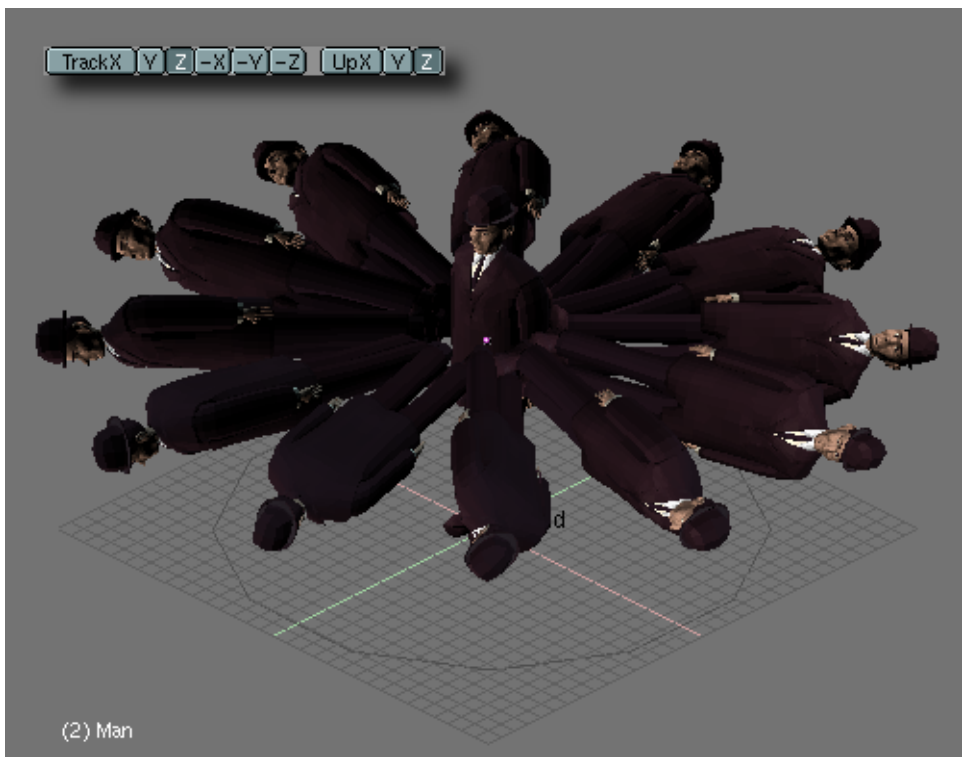


Figure 17-12. Positive Z axis is aligned to normal (wierd, huh ?)

Dupliverts to model a single object

Very interesting models can be done using Dupliverts and a standard primitive.

Starting from a cube in Front View, and extruding a couple of times I have modelled something which looks like a tentacle when Subsurfs are activated. Then I added an Icosphere with 2 subdivisions.

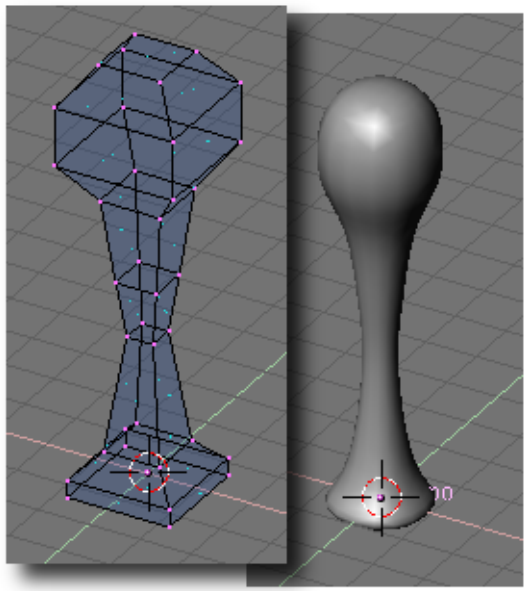


Figure 17-13. Strange tentacle (?) and subsurfed version

I had special care to be sure that the tentacle was located at the sphere center, and that both the tentacle axis and the sphere axis were aligned with the world axis as above.

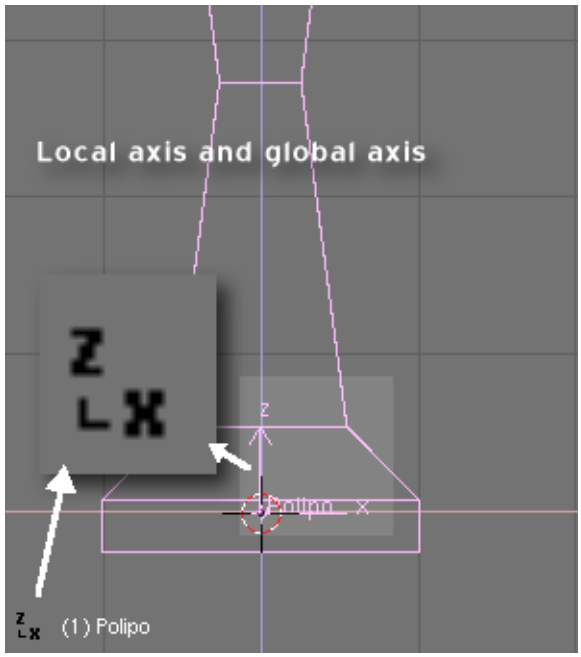


Figure 17-14. Strange tentacle (?) and subsurfed version

Now, I simply make the icosphere the parent of the tentacle. Select the icosphere alone and made it "Duplivert" in the AnimButtons.

Press the “Rot” button to rotate the tentacles.

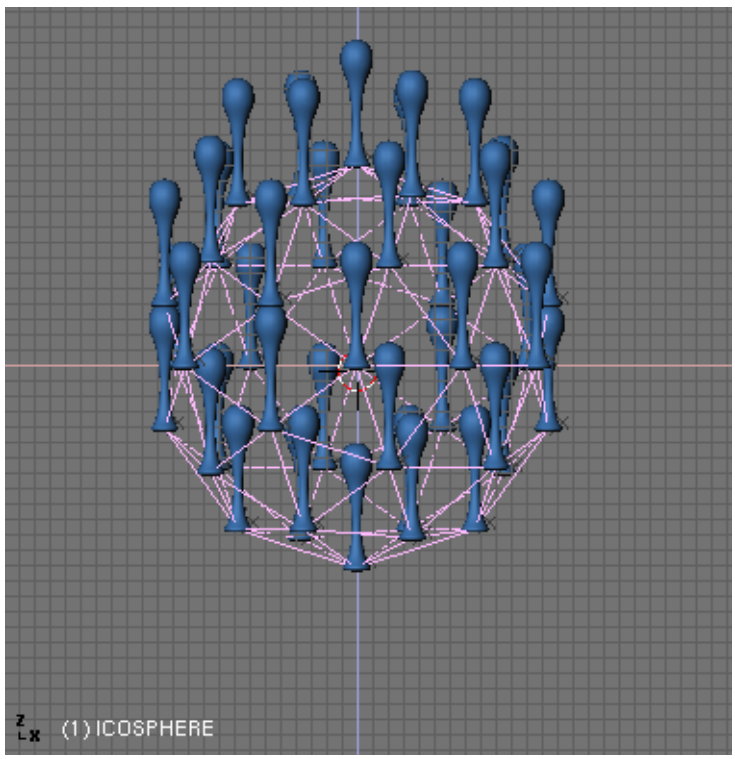


Figure 17-15. Dupliverts not rotated

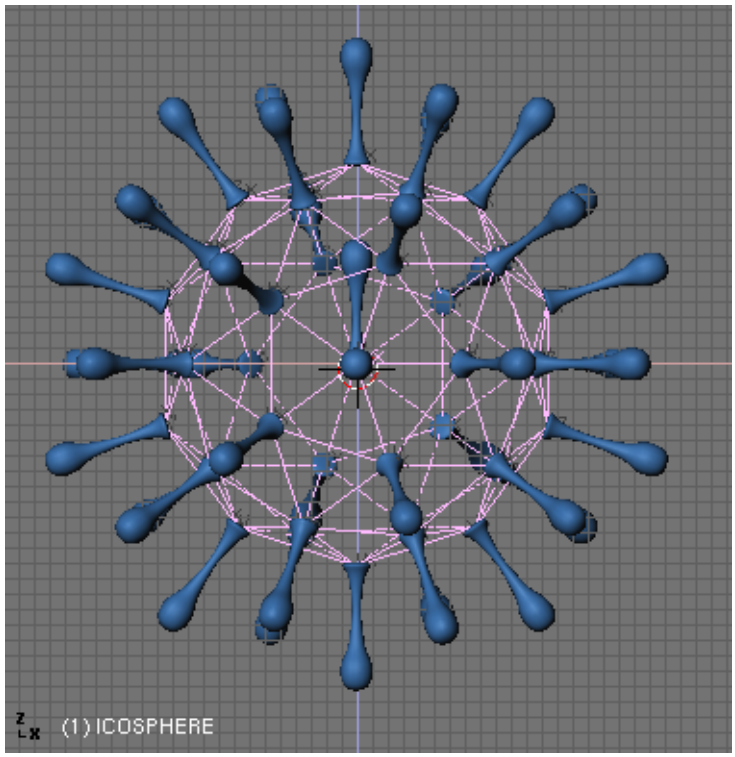


Figure 17-16. Dupliverts rotated

Once again to make the tentacle point outwards we have to take a closer look to its axis. When applying Rot, Blender will try to align one of the tentacle axis with the normal vector at the parent mesh vertex.

Again, the base mesh is not rendered, so you probably would like to add an extra renderable sphere to complete the model.

You can experiment in EditMode with the tentacle, moving its vertices off the centre of the sphere, but the object's center should always be at the sphere's center in order to get a simmetrical figure. However take care not to scale up or down in one axis in ObjectMode since it would lead to unpredictable results in the dupliverted objects when applying the "Rot" button.

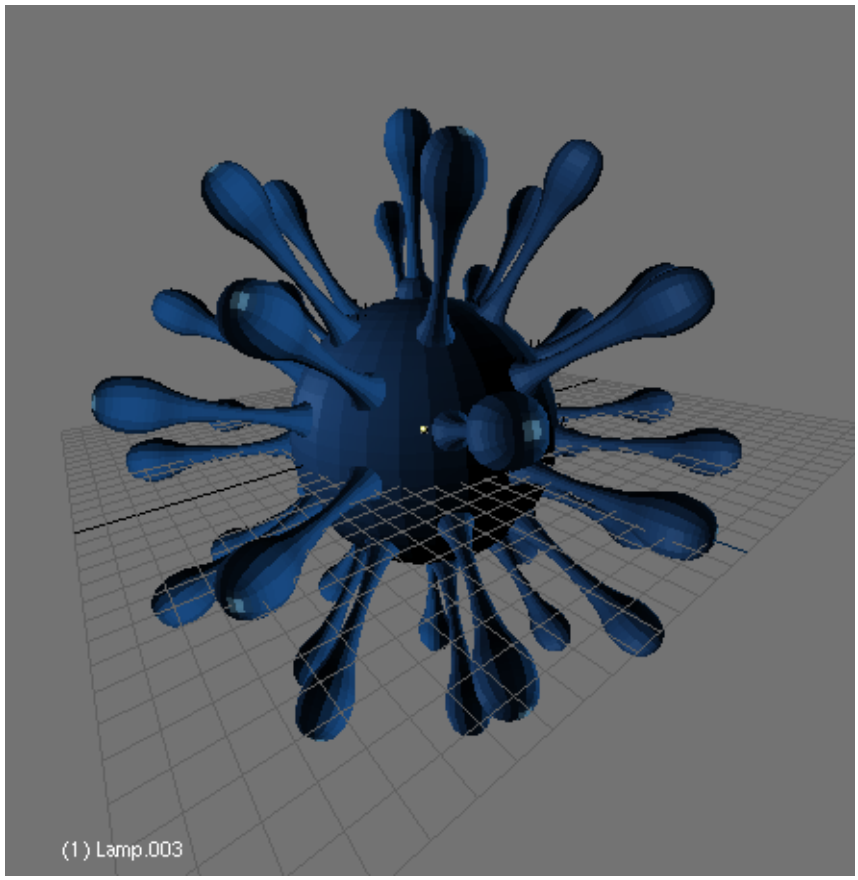


Figure 17-17. Our model complete

Once you're done with the model and you are happy with the results, you can select the tentacle and press **SHIFT-CTRL-A** and click on the "Make duplis real ?" menu to turn your virtual copies into real meshes.

Dupliframes

You can consider Dupliframes in two different ways: an arranging or a modelling tool. In a way, Dupliframes are quite similar to Dupliverts. The only difference is that with Dupliframes we arrange our objects by making them follow a curve rather than using the vertex of a mesh.

Dupliframes or Frame Duplication is a very useful modelling technique for objects which are repeated along a path, such as the wooden sleepers in a railroad, the boards in a fence or the links in a chain, but also for modelling complex curve objects like corkscrews, seashells and spirals.

Modelling using Dupliframes

We are going to model a chain with its links using Dupliframes

First things come first. To explain the use of dupliframes as a modelling technique, we will start by modelling a single link. To do this, add in front view a Curve Circle (Bezier or Nurbs, whatever). In Edit Mode, subdivide it once and move the vertices a little to fit the link's outline.

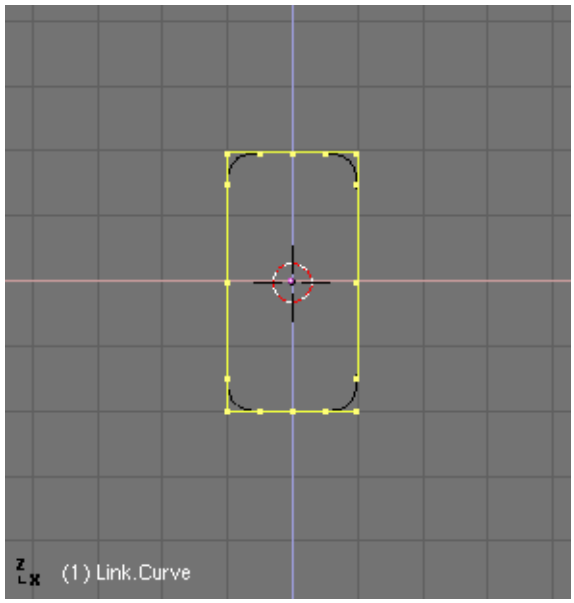


Figure 17-18. Link's outline

Leave Edit Mode and add a Surface Circle object. NURBS-surfaces are ideal for this purpose, because we can change the resolution easily after creation, and if we need to, we can convert them to a mesh object. It is very important that you do not confuse Curve Circle and Surface Circle. The first one will act as the shape of the link but it will not let us do the skinning step later on. The second one will act as a cross section of our skinning.

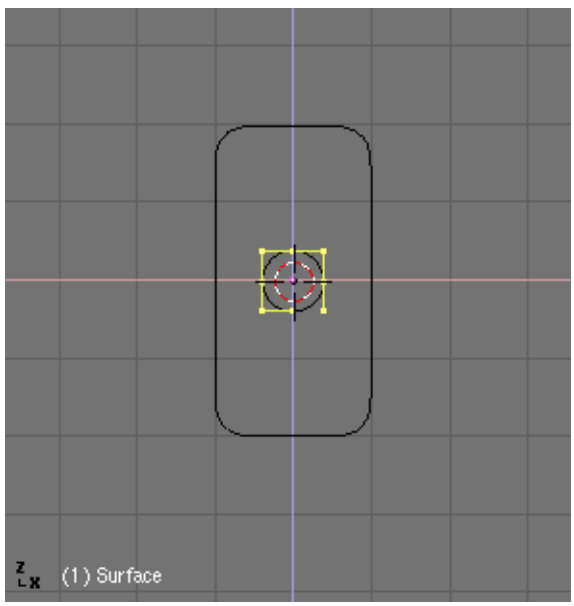


Figure 17-19. Link's cross section

Now parent the circle surface to the circle curve (the link's outline). Select the curve and in the Anims buttons press CurvePath and CurveFollow.

Figure 17-20. Curve's settings: Curve Follow

It probably happens that the circle surface will appear dislocated. Just select it and press **ALT-O** to clear the origin.

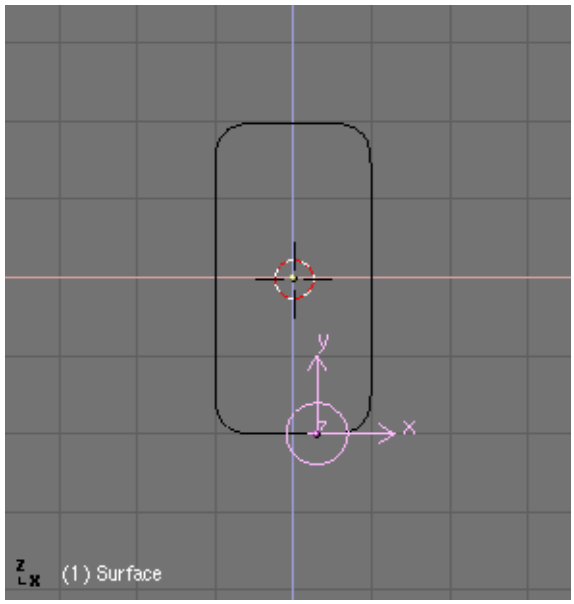


Figure 17-21. Erasing origin

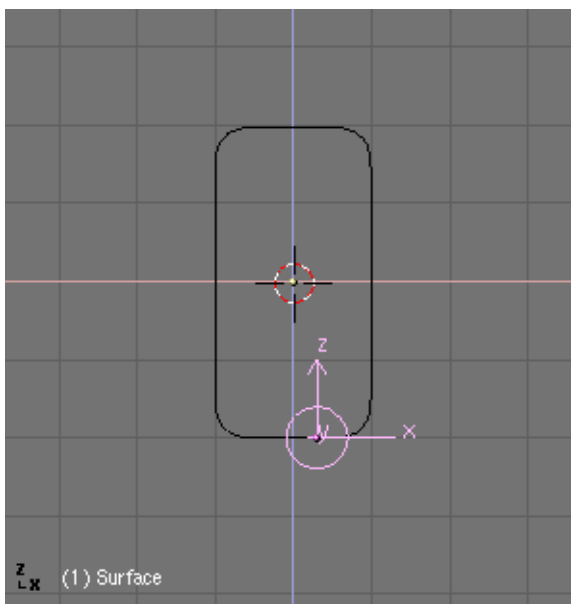


Figure 17-22. Aligning object's axis to World axis

If you hit **ALT-A** the circle will follow the curve. Now you probably will have to adjust the TRackX,Y,Z and UpX,Y,Z animation buttons, to make the circle go perpendicular to the curve path.

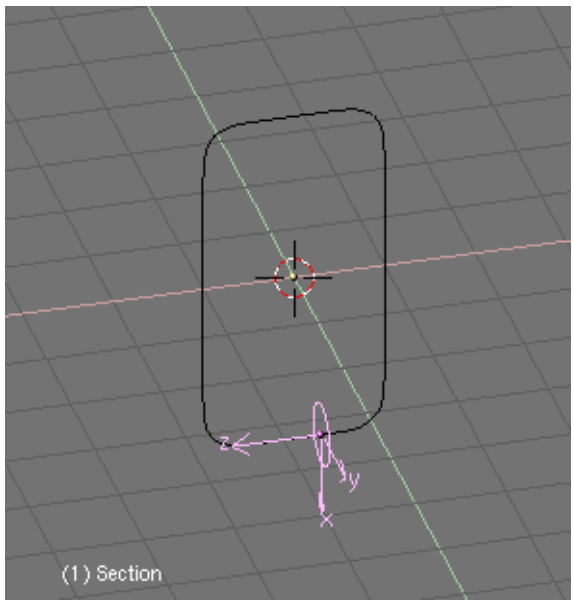


Figure 17-23. Tracking the right axis

Are you done ?, well, now select the Surface Circle and go to Animation Buttons and press Dupliframes. A number of instances of the circular cross section will appear along the curve path.

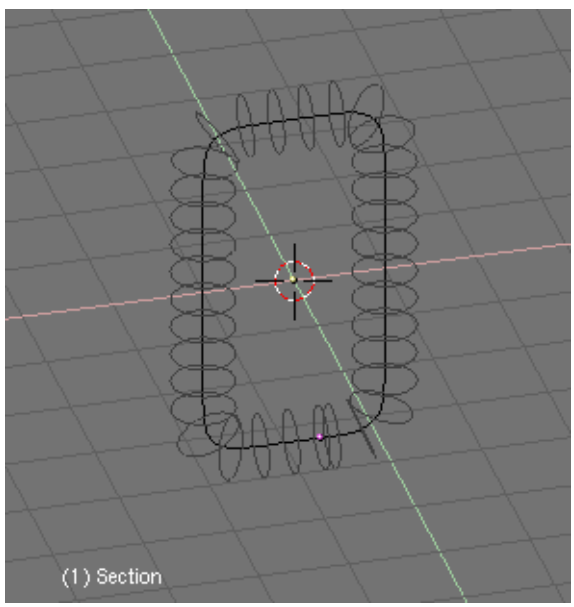


Figure 17-24. Dupliframes !

You can adjust the number of circles you want to have with the DupSta, DupEnd, DupOn and DupOff buttons. These buttons control the Start and End of the duplica-

tion, the numebr of duplicates each time and also the Offset between duplications. If you want the link to be opened, you can try a different setting for DupEnd.

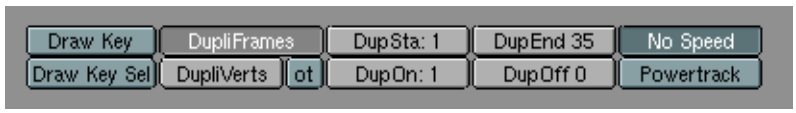


Figure 17-25. Values for dupliframes. Note "DupEnd: 35" will end link befor curve's end.

To turn the structure into a real NURBS-object, select the Surface Circle and press **CTRL-SHIFT-A**. A pop-up menu will appear prompting "OK? Make Dupli's Real".

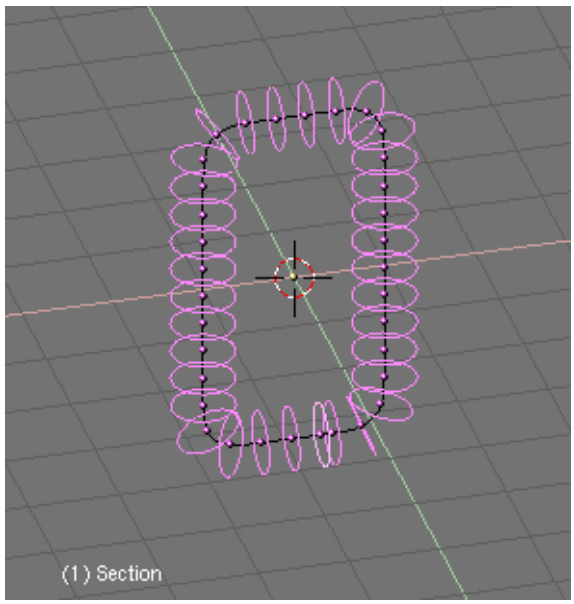


Figure 17-26. Values for dupliframes. Note "DupEnd: 35" will end link befor curve's end.

Do not deselect anything. We now have a collection of NURBS forming the outline of our object, but so far they are not skinned, so we cannot see them in a shaded preview or in a rendering. To achieve this, we need to join all the rings to one object. Without deselecting any rings, press **CTRL-J** and confirm the pop-up menu request. Now, enter EditMode for the newly created object and press **AKEY** to select all vertices. Now we are ready to skin our object. Press **FKEY** and Blender will automatically generate the solid object. This operation is called "Skinning".

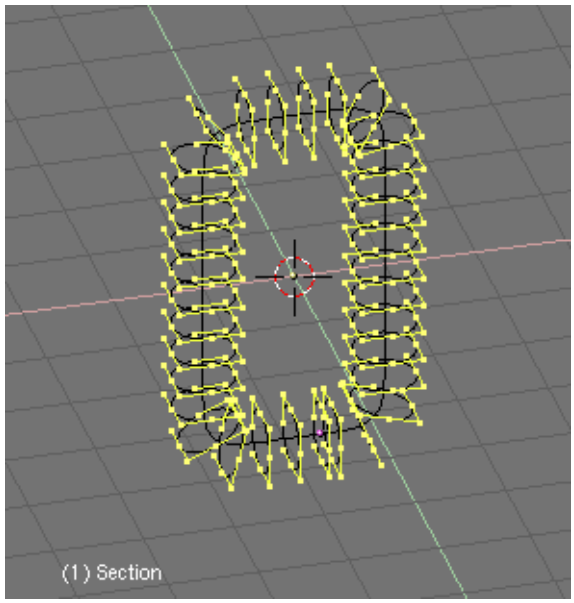


Figure 17-27. Skinning the link.

When you leave EditMode, you can now see the object in a shaded view. But it is very dark. To correct this, enter EditMode and select all vertices, then press **WKEY**. Choose "Switch Direction" from the menu and leave EditMode. The object will now be drawn correctly.

The object we have created is a NURBS object. This means that you can still edit it. Even more interestingly, you can also control the resolution of the NURBS object via the EditButtons.

Here you can set the resolution of the object using "ResolU" and "ResolV", so you can adjust it for working with the object in a low resolution, and then set it to a high resolution for your final render. NURBS objects are also very small in filesize for saved scenes. Compare the size of a NURBS scene with the same scene in which all NURBS are converted (**ALT-C**) to meshes.

Finally you can delete the curve we used to give the shape of the link, since we will not use it anymore.

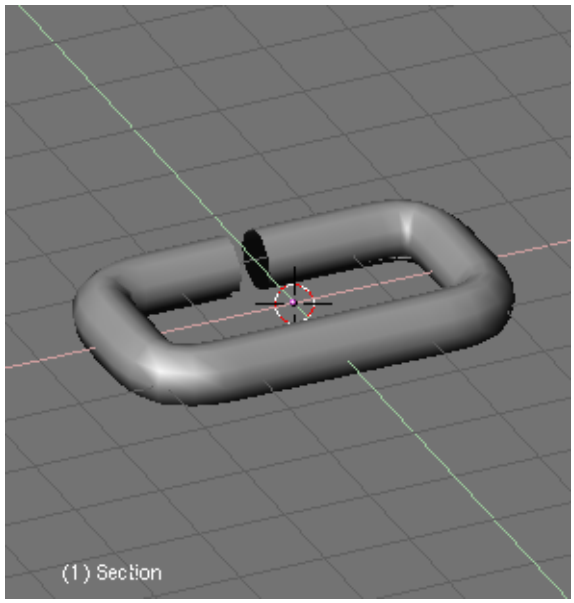


Figure 17-28. Values for dupliframes. Note "DupEnd: 35" will end link befor curve's end.

Arranging objects with Dupliframes

Now we will continue modelling the chain itself. For this, just add a Curve Path (we could use a different curve but this one gives better results). In Edit Mode, move its vertices until get the desired shape of the chain. If not using a Curve Path, you should check the button 3D in the Edit Buttons to let the chain be real 3D.

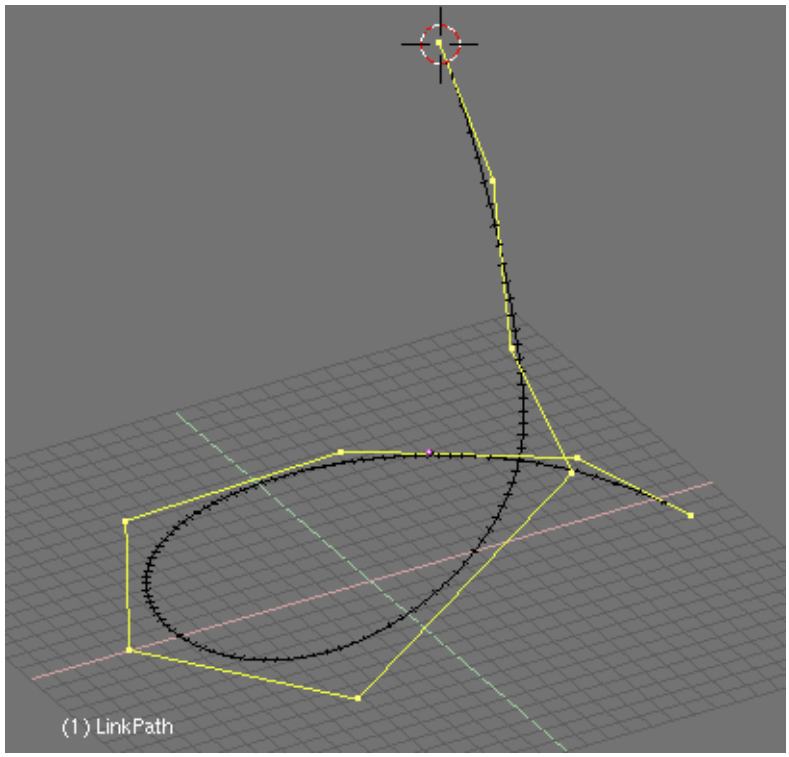


Figure 17-29. Using a curve path to model the chain.

Select the object "Link" we modelled in the previous step and parent it to the chain curve. Since we are using a Curve Path the option "CurvePath" in the AnimButtons will be automatically activated, however the "CurveFollow" option will not, so you will have to activate it.

Figure 17-30. Curve settings.

If the link is dislocated, select it and press **ALT-O** to clear the origin. Until now we have done little more than animate the link along the curve. This can be verified by playing the animation with **ALT-A**.

Now, with the link selected once again go to the AnimButtons. Here, activate the option "DupliFrames" as before. Play With the "DupSta:", "DupEnd:" and "DupOf:" NumButtons. Normally we are going to use "DupOf: 0" for a chain. If using "DupOf: 0" the links are too close from each other you should change the value PathLen for the path curve to a minor value, and correspondingly change the DupEnd: value for the link to that number.

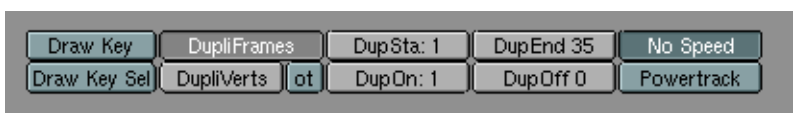


Figure 17-31. Adjusting the dupliframes.

We need that the link rotates along the curve animation, so we have each link rotated 90 degrees respect the preceding one in the chain. For this, select the link and press Axis in the Edit Buttons to reveal the object's axis. Insert a rotation keyframe in the axis which was parallel to the curve. Move 3 or 4 frames ahead and rotate along that axis pressing "R" followed by "X-X" (X twice),Y-Y, or Z-Z to rotate in the local X,Y or Z axis.

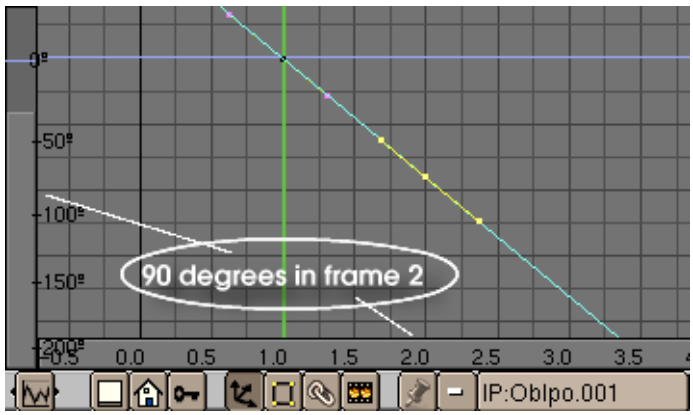


Figure 17-32. Rotating the link.

Open an IPO window to edit the rotation of the link along the path. Press the "Extrapolation Mode" so the link will continually rotate until the end of the path. You can edit the IPO rotation curve to make the link rotate exactly 90 degrees every one, two or three links (each link is a frame). Use the "N" key to locate a node exactly at X=2.0 and Y=9.0, which correspond to 90 degrees in 1 frame (since frame 1 to 2).

Now we got a nice chain !

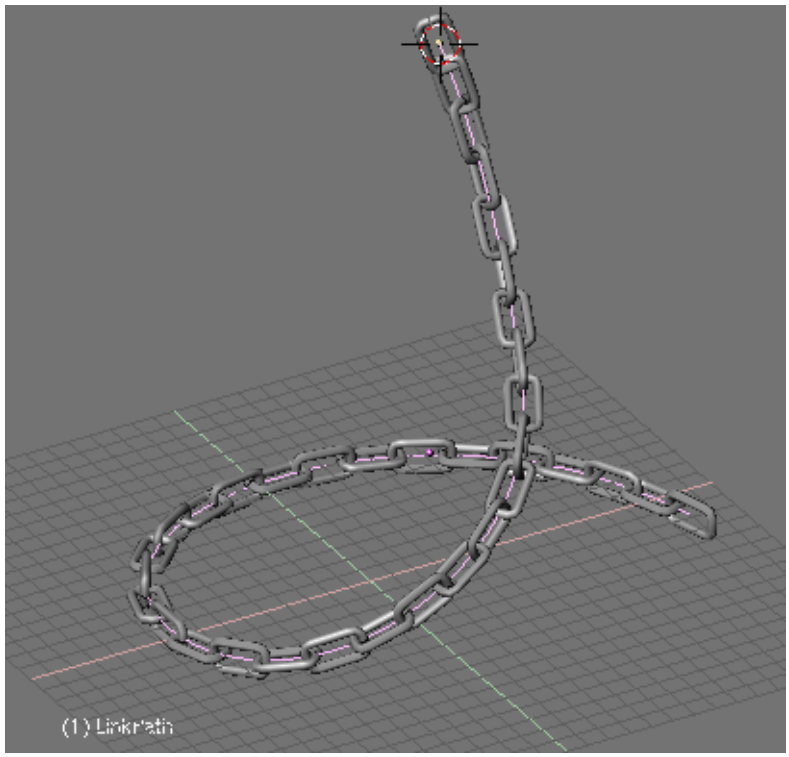


Figure 17-33. Dupliframed chain.

More Animation and Modelling

You are not limited to use Curve Paths to model your stuff. These were used just for our own convenience, however in some cases there are no need of them.

In Front View add a surface circle (you should know why by now). Subdivide once, to make it look more like a square. Move and scale some vertices a little to give it a trapezoid shape.

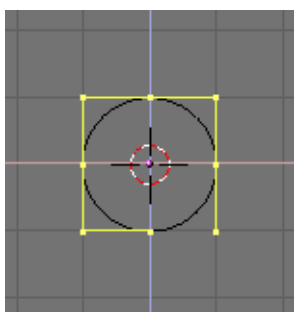


Figure 17-34. Figure

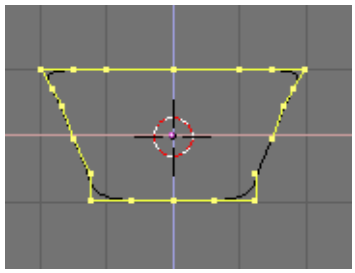


Figure 17-35. Figure

Then rotate all vertices a few degrees. Grab all vertices and displace them some units right or left in X (but at the same Z location). You can use **CTRL-K** to achieve this precisely. Leave Edit Mode.

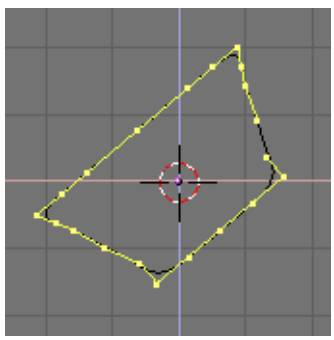


Figure 17-36. Figure

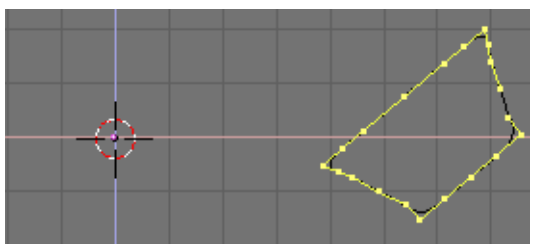


Figure 17-37. Figure

From now on, the only thing we are going to do is editing IPO animation curves. So you can call this "Modelling with Animation" if you like. We will not enter Edit Mode for the surface in any moment.

Switch to Top View. Insert a keyframe for rotation at frame 1, go ahead 10 frames and rotate the surface 90 degrees over its new origin. Insert one more keyframe. Open an IPO window, and set the rotation IPO to Extrapolation Mode.

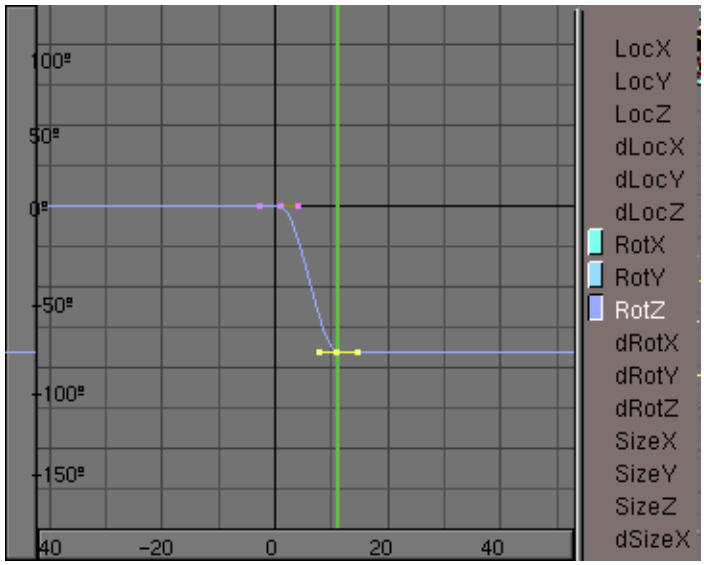


Figure 17-38. Using a curve path to model the chain.

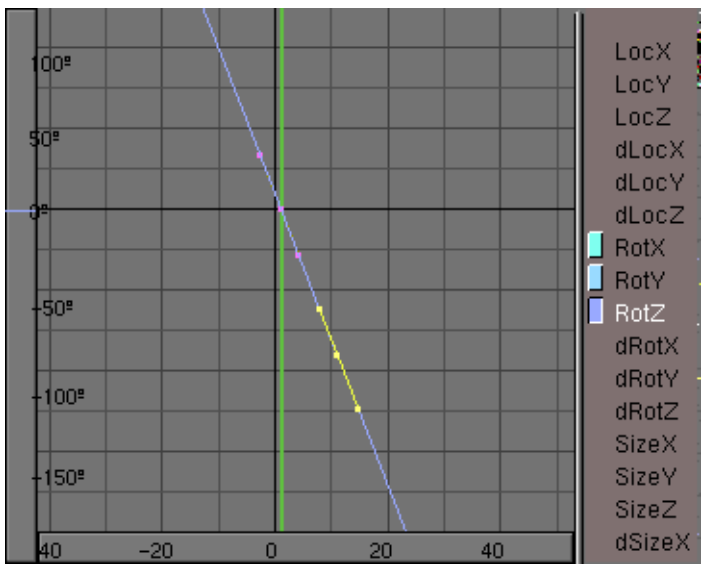


Figure 17-39. Using a curve path to model the chain.

Go back to frame 1 and insert a keyframe for Location. Switch to Front View. Go to frame 11 (just press Cursor Up) and move the surface in Z a few grid units. Insert a new keyframe for Location. In the IPO window set the LocZ to Extrapolation Mode.

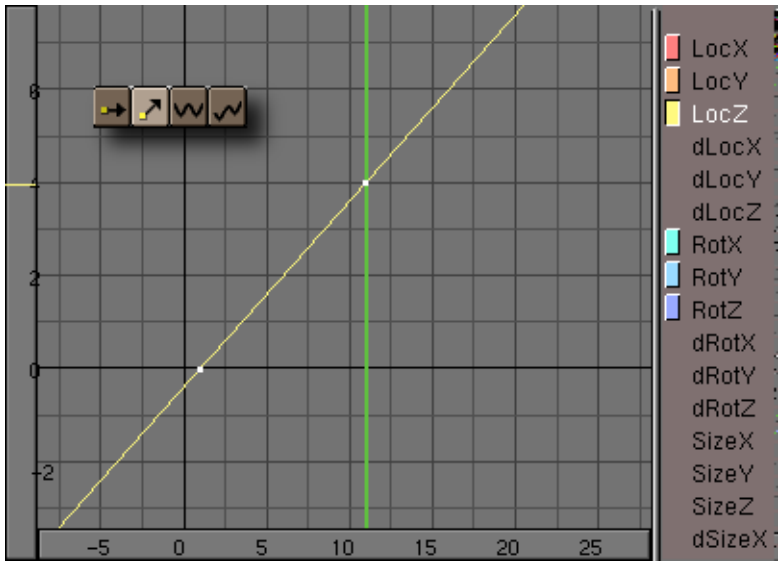


Figure 17-40. Using a curve path to model the chain.

Now, of course, go to the Animation buttons and press Dupliframes. You can see how our surface is ascending spirally thru the 3D space forming something like a spring. This is nice, however we want more. Deactivate Dupliframes to continue.

In frame 1 scale the surface to nearly zero and insert a keyframe for Size. Go ahead to frame 41, and clear the size with **ALT-S**. Insert a new keyframe for size. This IPO will not be in extrapolation mode since we don't want it scales up at infinitum, right ?

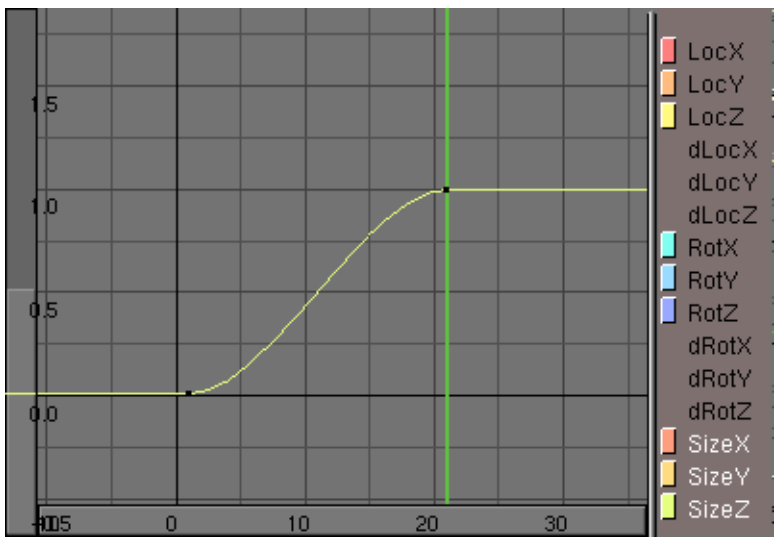


Figure 17-41. Using a curve path to model the chain.

If you now activate Dupliframes you will see a beautiful outline of a corkscrew. Once again the last steps are: Make Duplis Real, Joining teh surfaces, Select all vertices and skinning, Switch direction of normal if needed and leave Edit Mode.

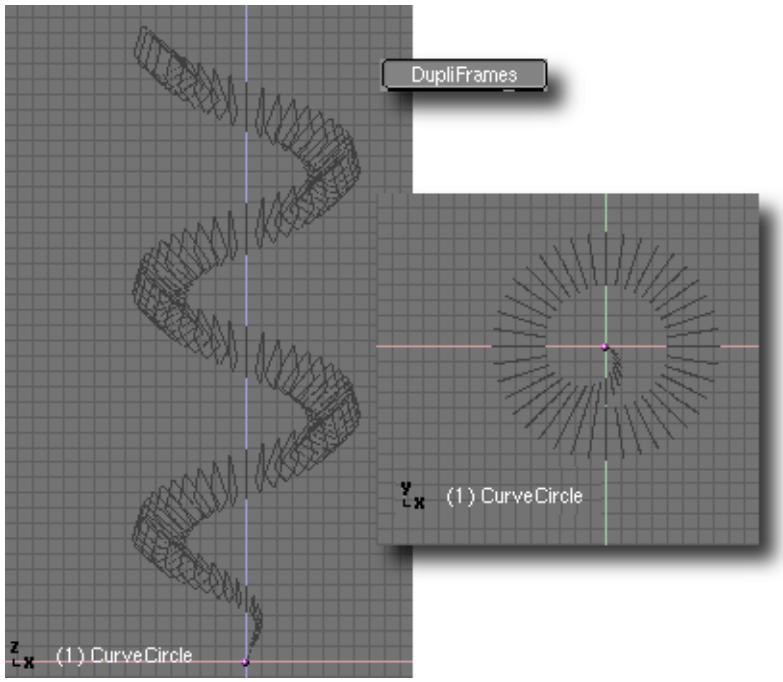


Figure 17-42. Using a curve path to model the chain.

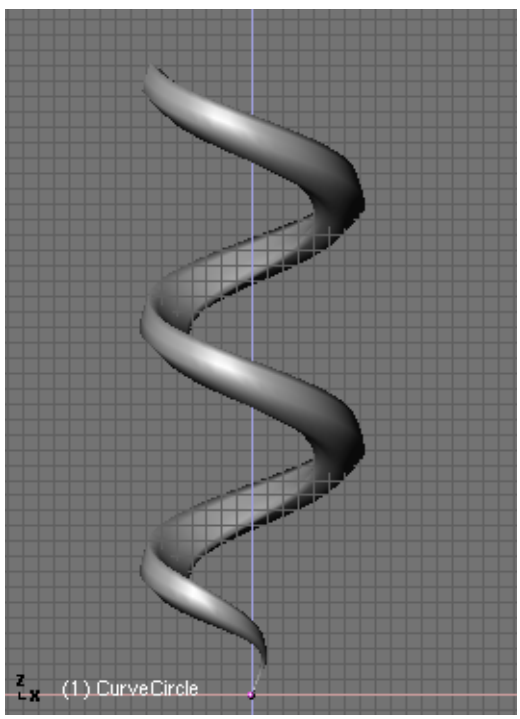


Figure 17-43. Using a curve path to model the chain.

You can see this was a rather simple example. With more IPO curve editing you can achieve very interesting and complex models. Just use your imagination.

Modelling with lattices

A Lattice consists of a non-renderable three-dimensional grid of vertices. Their main use is to give extra deformation to any child object they might have. These child objects can be Meshes, Surfaces and even Particles.

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:

1. First of all: It's easier. Since your mesh could have a zillion vertices, scaling, grabbing and moving them could be a hard task. Instead, if you use a nice simple lattice your job is simplified to move a couple of vertices.
2. It's nicer. The deformation you get looks a lot better !
3. It's FAST !. You can put all or several of your child objects in a hidden layer and deform them all at once.
4. It's a good practice. A lattice can be used to get different versions of a mesh with minimal extra work and consumption of resources. This leads to an optimal scene design, minimizing the amount of modelling job. A Lattice does not affect the texture coordinates of a Mesh Surface. Subtle changes to mesh objects are easily facilitated in this way, and do not change the mesh itself.

How does it work ?

A Lattice always begins as a 2 x 2 x 2 grid of vertices (which looks like a simple cube). You can scale it up and down in Object Mode and change its resolution thru the EditButtons->U,V,W.

After this initial step you can deforme the Lattice in EditMode. If there is a Child Object, the deformation is continually displayed and modified. Changing the U,V,W values of a Lattice returns it to a uniform starting position.

Now we are going to see a very simple case in which having a lattice will simplify and speed up our modelling job.

I have modelled a very simple fork using a plane subdivided couple of times. It looks really ugly but it's all I need. Of course it is completely flat from a Side View. Wow, it is REALLY ugly. The only important detail is that it has been subdivided enough to ensure a nice deformation in the Lattice step. You cannot bend a two vertices segment !

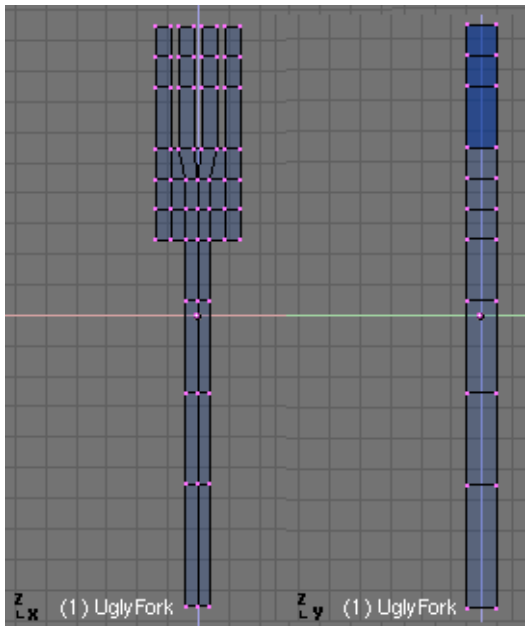


Figure 17-44. An ugly fork.

In Top View, now add a Lattice. Before changing its resolution, scale it up so it completely envolves the fork's width. This is very important. Since I want to keep the lattice vertices count low (it doesn't make sense it has the same number of vertices than the mesh, right ?) I need to keep resolution low but still set the lattice to convenient size.

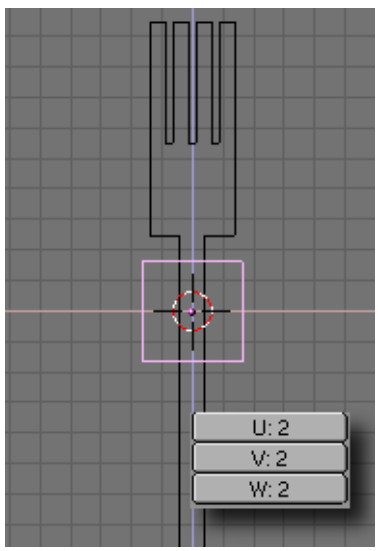


Figure 17-45. A 2x2x2 Lattice.

Adjust the Lattice resolution to complete the fork's lenght.

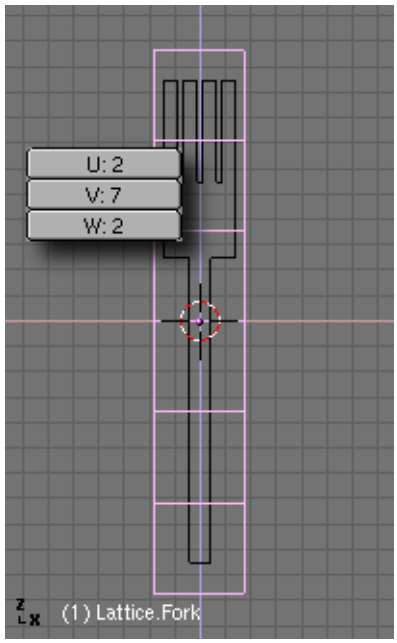


Figure 17-46. Use a suitable resolution, but don't exaggerate.

Now, we are ready for the fun part. Parent the fork to the lattice, by selecting the fork and the lattice and pressing **CTRL-P**. Enter Edit Mode for the lattice and start selecting and scaling vertices. You might want to scale in X or Y axis separately to have more control over the lattice depth (to avoid making the fork thicker or thinner).

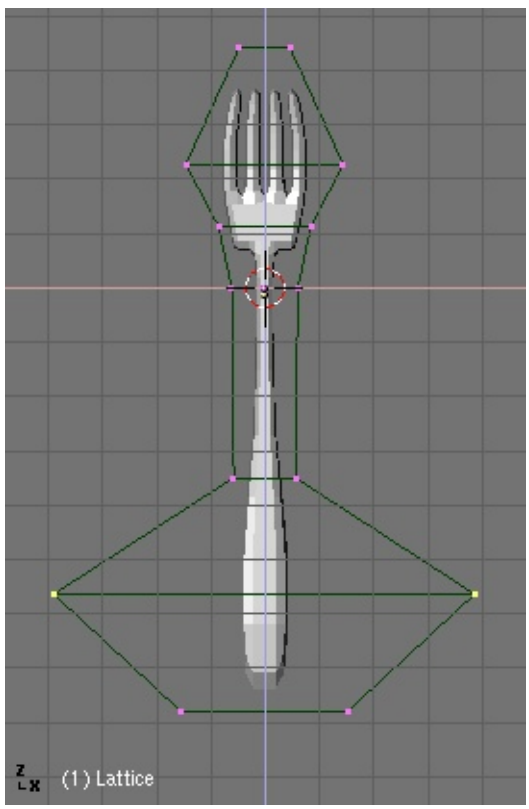


Figure 17-47. Deforming the lattice is a pleasure !.

Note that if you move the fork up ad down inside the lattice, the deformation will apply in different parts of the mesh.

Once you're done in Front View, switch to Side View. Select and move different vertices sections to give the fork the suitable bends.

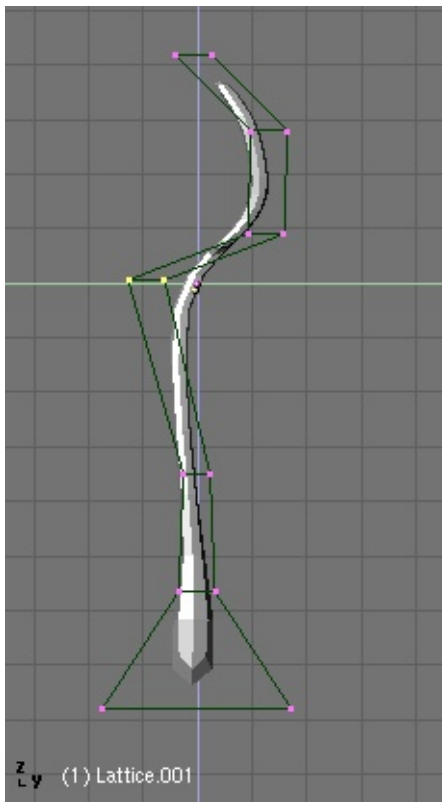


Figure 17-48. Bending things.

It was quick, wasn't it ?

You can get rid of the lattice now if you're not adding any other child object. But before doing it, you might want to keep your deformations !. Just select the fork and press **CTRL-SHIFT-A** and click on the "Apply Lattice Deform ?" menu entry.

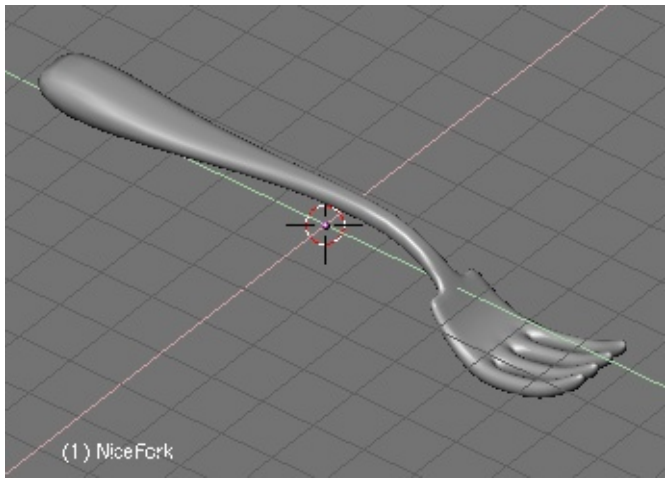


Figure 17-49. A nice fork.

You can use a lattice to model an object following another object's shape. For instance take a look at the following scene. I have modelled a bottle, and now I would like to confine a character inside it. He deserves it.

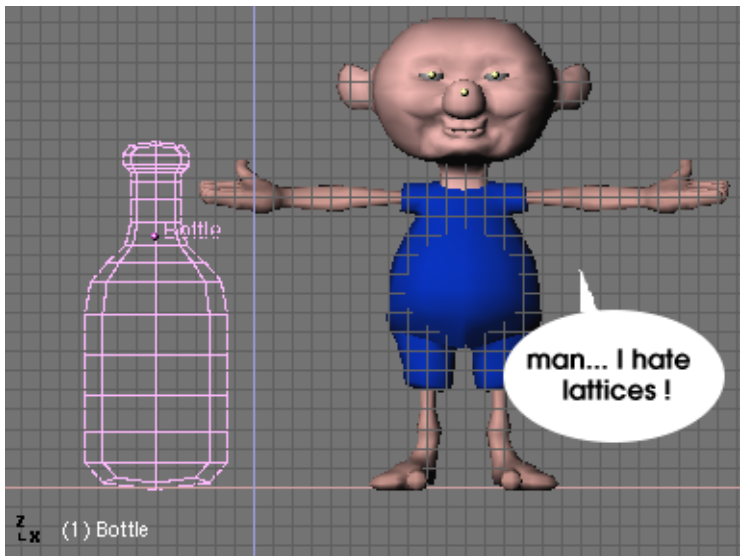


Figure 17-50. Poor guy...

Add a lattice to envolve the character. I didn't use a too high resolution for the lattice. I scaled it in X and Y to fit the lattice to the character.

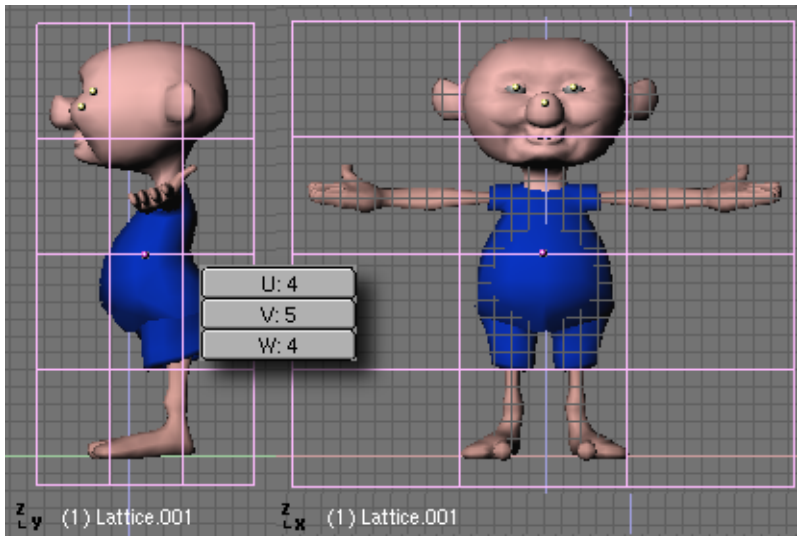


Figure 17-51. Bending things.

Parent the character to the lattice, and then scale the lattice again to fit the dimensions of the bottle.

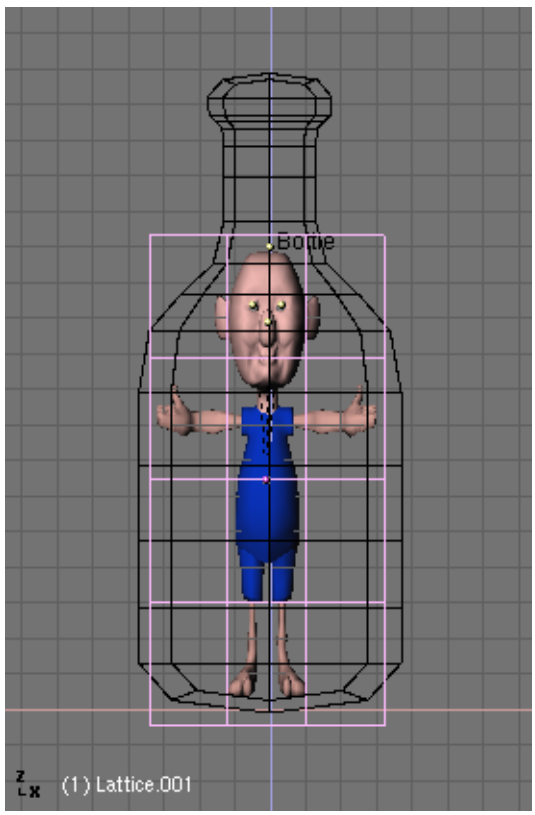


Figure 17-52. Scale the lattice to fit the bottle.

Now enter Edit Mode for the lattice. Press the Outside button in the Edit Buttons to switch off the inner vertices of the lattice. We will switch them on later. Move and scale the vertices in front and side views until the character perfectly fits the bottle's shape.

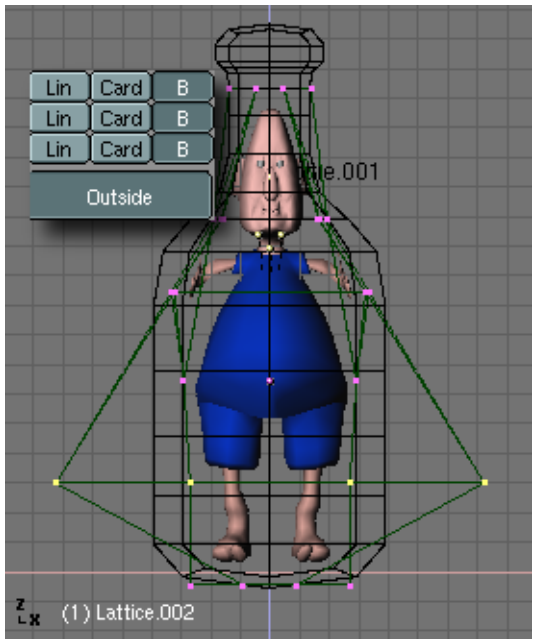


Figure 17-53. Edit comfortable.

You can select the lattice and do the modelling in one 3D window using Local View and see the results in another window using Global View to make your modelling comfortable.

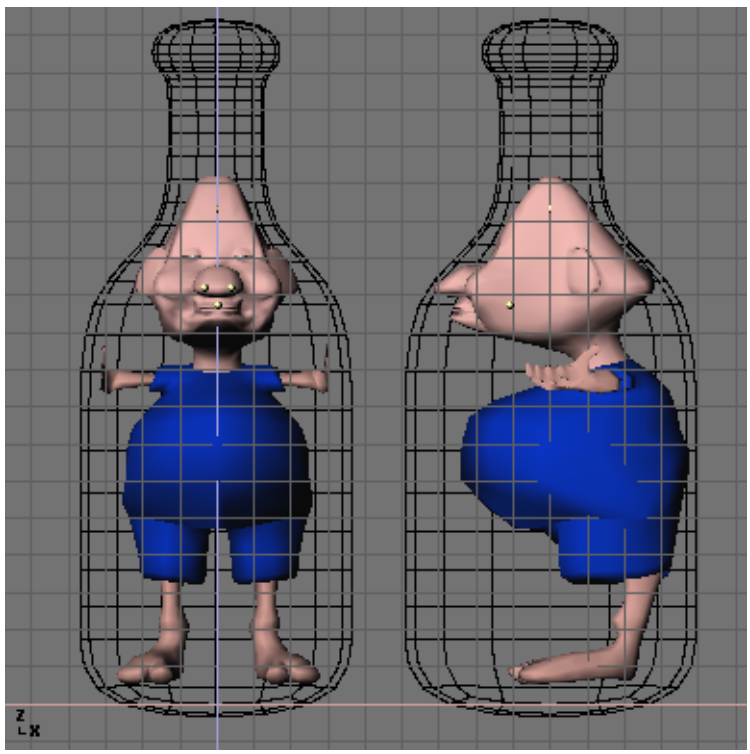


Figure 17-54. Claustrophobic ?.

Hadn't we used a lattice it would have taken a lot more of vertex picking-and-moving work to deform the character.

Since lattices also supports RVK for vertex animation, quite interesting effects can be achieved with this tool.

Figure 17-55. Final Render. Believe me, he deserved it !

Lattices can be used in many applications which require a "liquid-like" deformation of a mesh. Think of a genie coming out of his lamp, or a cartoon character with its eyes popping out exaggeratedly. And have fun !

Resources

- *Dupliframes modeling: A Ride Through the Mines* - http://www.vrotvrot.com/xoom/tutorials/mineRide/mineride.html

Notes

1. and also ObjectMode, however scaling in ObjectMode could bring up some problems when applying Rotation to dupliverts as we will see soon
2. http://www.vrotvrot.com/xoom/tutorials/mineRide/mineride.html

Chapter 18. Volumetric Effects

Although Blender exhibits a very nice Mist option in the World Settings to give your images some nice depth, you might want to create true volumetric effects; mists and clouds and smoke which really looks like they occupy some space.

Figure 18-1 shows a setup with some columns in a circular pattern, whith some nice material of your choiche for columns and soil and a World defining sky colour.

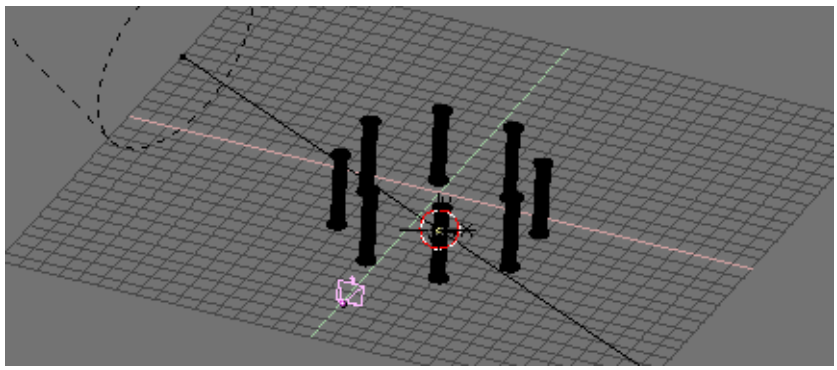


Figure 18-1. Columns on a plain.

Figure 18-2 shows, the relative rendering, whereas Figure 18-3 there is a rendering with Blender's built-in Mist. Mist setting in this particular case are: Linear Mist, Sta=1, Di=20, Hig=5

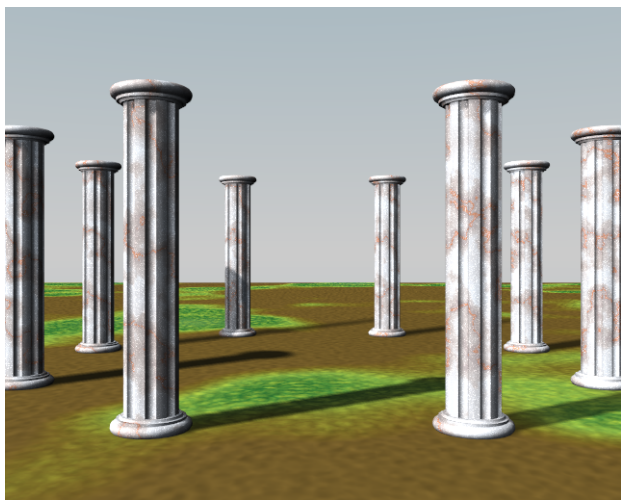


Figure 18-2. A plain rendering.

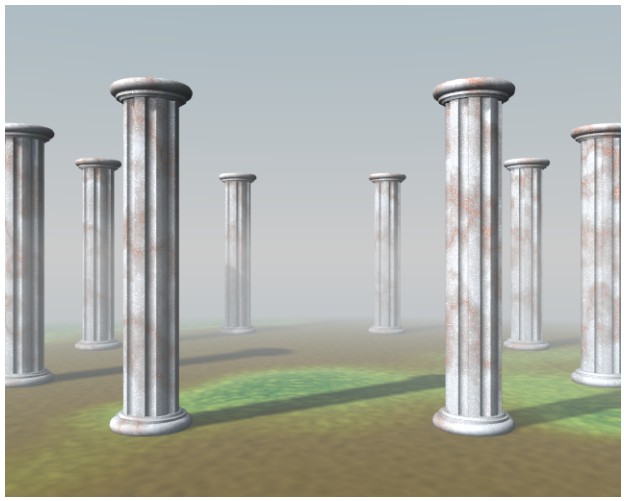


Figure 18-3. A rendering with builin Blender Mist.

But we want to create some truly cool, swirling, and, most important non-uniform mist. Blender Built in textures, clouds for example, are intrinsecally 3D, but are rendered only when mapped onto a 2D surface. We will achieve a 'volumetric' like rendering by 'sampling' the texture on a series of mutually parallel planes. Each of our planes will hence exibits a standard Blender texture on its 2D surface, but the global effect will be of a 3D object. This concep will be clearer as the example proceeds.

With the camera in the default position, turn to front view, and add a plane in front of the camera, with its center aligned with the camera wiewing direction. In side view move the plane where you want your volumetric effect to terminate. In our case somewhere beyond the furthest column. Scale the plane so that it encompasses all camera's viewing angle (Figure 18-4). It is important to have a camera in the default position, that is pointing along the y axis since we need the planes to be orthogonal to the direction of sight. We will anyway be able to move it later on.

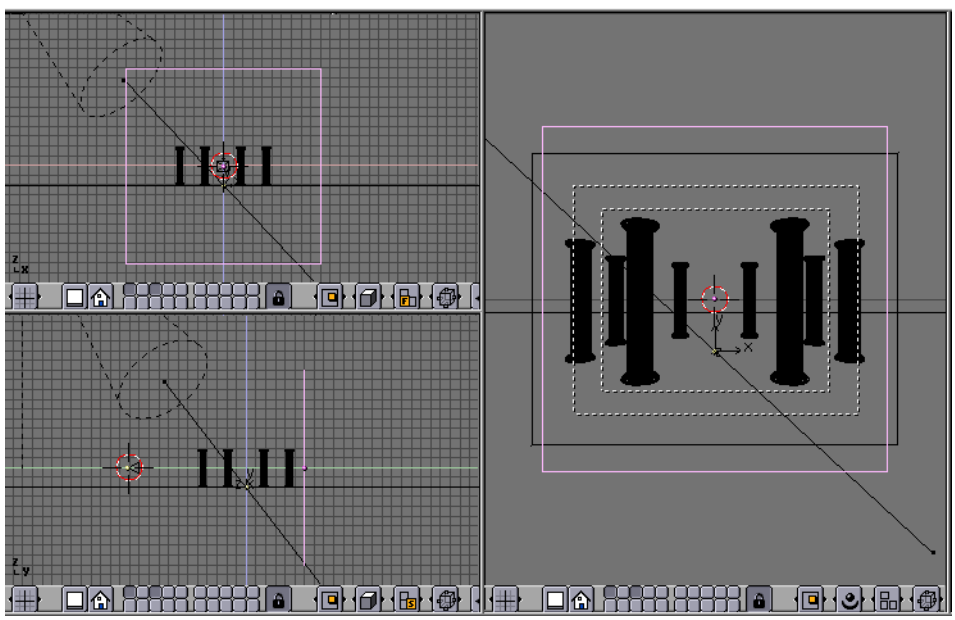


Figure 18-4. The plane setup.

After having checked that we're at frame 1, let's place a Loc keyframe (**IKEY**). We

should now move to frame 100, move the plane much nearer to the camera, and set another Loc Keyframe. Now, in the animation buttons (**F7**) Press the DupliFrame button.

The 3D window, in side view, will show something like Figure 18-5. This is not good because planes are denser at the begin and at the end of the sweeep. With the plane still selected change to an IPO window (**SHIFT F6**). There will be three Loc IPOs, only one of which non-constant. Select it, swithc to EditMode (**TAB**) and select both control points. Now turn them from smooth to sharp with (**VKEY**) Figure 18-6.

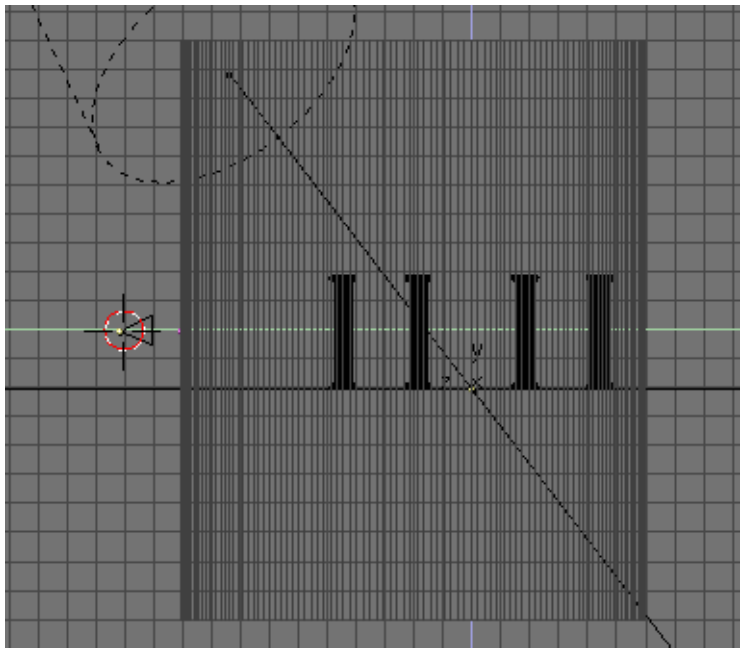


Figure 18-5. The Dupliframed plane.

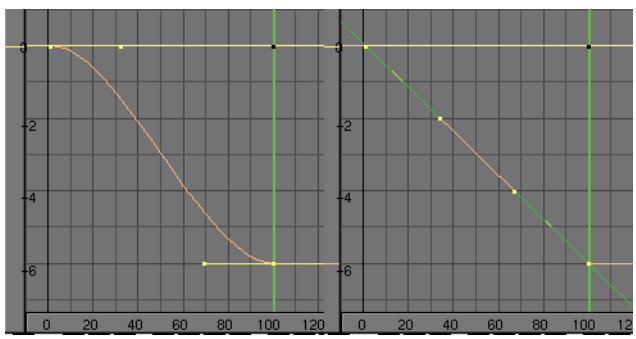


Figure 18-6. Reshaping the Dupliframed Plane IPO.

The planes will now look as in Figure 18-7. Parent the Dupliframed planes to the camera (select the plane, **SHIFT** select the camera, **CTRL P**. You have now a series of planes automatically following the camera, always oriented perpendicularly to it. From now on you could move the camera if you so wish.

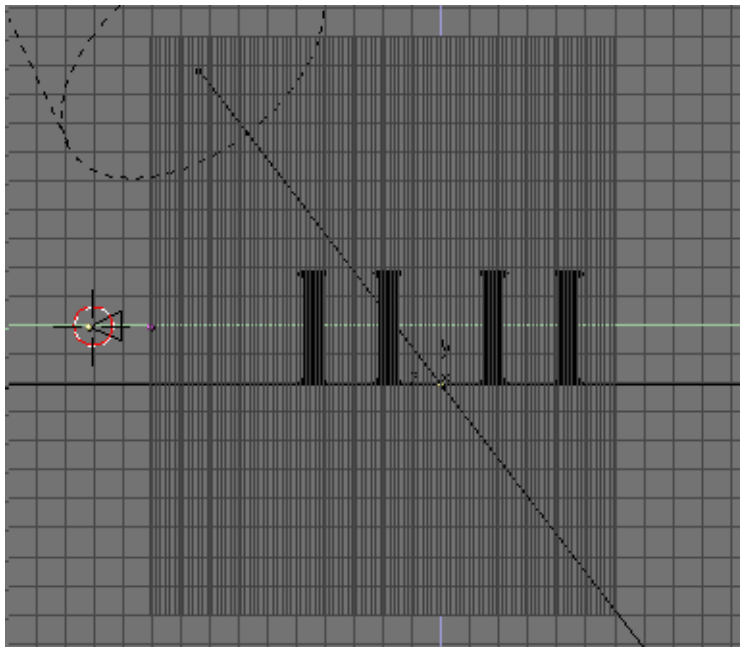


Figure 18-7. Reshaping the Dupliframed Plane IPO.

Figure 18-8. Basic Material settings.

Now we must add the Mist material itself. The material should be Shadeless and cast no shadows to avoid undesired effects. It should have an small Alpha value (Figure 18-8) A material like this would basically act like Blender's built in mist, hence we would have no advantage in the resulting image. The drawback is that computing 100 transparent layer is very CPU intensive, expecially if one desires the better results of the Unified Renderer.

> **Quick previews:** You can use the `DupOff:` NumButton in the Animation Buttons window to turn off some of the planes and hence have a faster, lower quality preview of what you are doing. For the final rendering you will then turn DupOff back to 0.
>
> Pay attention to the Alpha value! the lesser planes you use the thinner will be the mist, so your final rendering will be much more 'Misty' than your previews!

The true interesting stuff comes when you add textures. We will need at least two: One to limit the Mist in the vertical dimension and keep it on the ground; The second to make it non uniform and with some varying hue.

As a first texture Add a Blend Linear texture, with a very simple colorband, going from pure white, Alpha=1 at a position 0.1 to pure white, Alpha=0 at a position 0.9 (Figure 18-9). Add this as a Alpha only `Mul` texture (Figure 18-10). To make our mist consistent as the Camera moves, and the planes follows, we have to set it `Global`. This will be true also for all other textures and will make the planes sample a fixed 3D volumetric texture. If you are planning an animation you will see a static mist, with respect to the scene, while the camera moves. Whichever other texture setting would show a Mist wich is static with respect to the camera, hence being always the same while the camera moves, which is higly unrealistic.

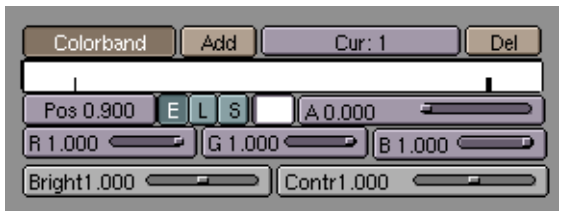


Figure 18-9. Basic Material settings.

Figure 18-10. Basic Material settings.

If you want to anyway have a moving, swirling changing mist you can do so by animating the texture, as will be explained later on.

The Blend texture operates on X and Y directions, so if you want it to span vertically in the Global coordinates you will have to remap it (Figure 18-10). Please note that the blending from Alpha=1 to the Alpha=0 will occurr from global z=0 to global z=1 unless additional offsets and scalings are added. For our aim the standard settings are OK.

If you now do a rendering, it doesn't matter where your camera, and planes, are. The mist will be thick below z=0, non-existent above z=1 and fading in between. If you're puzzled by this apparent complexity, think of what you would have got with a regular Orco texture and non-parented planes. If you had to move the camera, expecially in animations, the results would become very poor as soon as the planes are not perpendicular to the camera any more. To end up with no mist at all if the camera were to become parallel to the planes!.

The second texture is the one giving the true edge on the built in mist. Add a Cloud texture, make its Noise Size=2, Noise Depth=6 and Hard Noise on (Figure 18-11). Add colorband to this too, going from pure white with Alpha=1 at Positoion 0 to a pale bluish gray with Alpha = 0.8 at a position of about 0.15, to a pinkish hue with Alpha 0.5 around position 0.2, ending to a pure white, Alpha=0 color at position 0.3. Of course you might want to go to greenish-yellow for swamp mists etc.

Figure 18-11. Cloud texture settings.

Use this texture on both `Col` and `Alpha` as a `Mul` texture, keeping al other settings to default. If you now render the scene the bases of your columns will be now masked by a cool mist (Figure 18-12). Please note that the Unified renderer gives much better results here.

Figure 18-12. Cloud texture settings.

If you are planning an animation and want your Mist to be animated like if it were moved by a wind it is this latter texture you must work on. Add a Material texture IPO, be sure to select the correct texture channel and add some IPO to the OfsX, OfsY and OfsZ properties.

Chapter 19. Sequence Editor

An often underestimated function of Blender is the Sequence Editor. It is a complete video editing system that allows you to combine multiple video channels and add effects to them. Even though it has a limited number of operations, you can use these to create powerful video edits (especially when you combine it with the animation power of Blender!) And, furthermore, it is extensible via a Plugin system quite alike the Texture plugins.

Learning the Sequence Editor

This section shows you a practical video editing example exhibiting most of the Sequence Editor built in features. We will put toghether several Blender made animations to obtain some stuning effects. One frame of the resulting edited animation is in Figure 19-1.

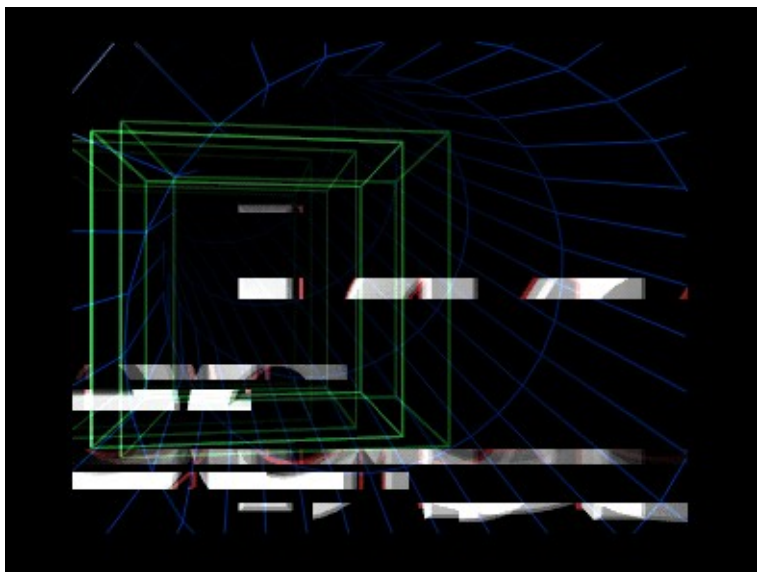


Figure 19-1. Final result.

First animation: two cubes

Let's start with something simple and see where it leads. Start a clean Blender and remove the default plane. Split the 3D window and switch one of the views to the camera view with **NUM 0**. In the top-view, add a cube and move it just outside of the dotted square that indicates the camera view. Figure 19-2.

> **TV limitations:** When you are planning to show your work on television, note the inner dotted square. Since not all televisions are the same, there is always a part of the picture that is 'cut off'. The inner square indicates which area is guaranteed to be viewable. The area between the dotted lines is referred to as the 'overscan area'.

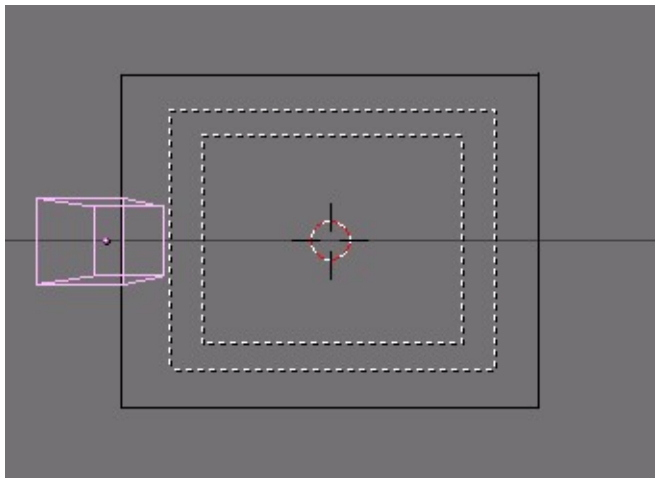


Figure 19-2. Moving the cube out of the camera view.

We want to create a simple animation of the cube where it moves into view, rotates once and then disappears. Set the animation end to 61 (set the `End:` value in the Render Buttons window - **F10**) and insert a LocRot keyframe on frame 1 with **IKEY** and selecting `LocRot` from the menu which appears. This will store both the location and the rotation of the cube on this frame.

Go to frame 21 (press **ARROW_UP** twice) and move the cube closer to the camera. Insert another keyframe. On Frame 41, keep the cube on the same location but rotate it 180 degrees and insert another keyframe.

Finally on frame 61 move the cube out of view, to the right and insert the last keyframe.

Keyframe checking: To check, select the cube and press **KKEY** to show all keyframes in the 3D window. If you want, you can easily make changes by selecting a keyframe with **PAGEUP** or **PAGEDOWN** (the active keyframe will be displayed as a brighter yellow color than the other keyframes) and moving or rotating the cube. With the keys displayed, you do not need to re-insert the keyframes - they are automatically updated (Figure 19-3).

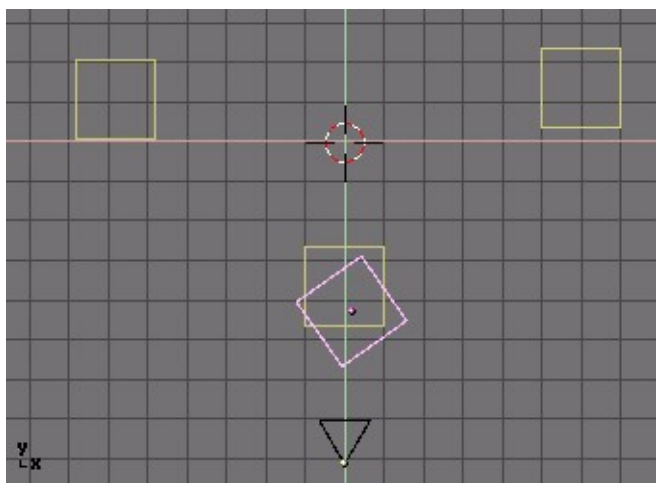


Figure 19-3. Defining keyframes for the cube

We will need two versions of the animation: one with a solid material and one with a wireframe. For the material, we can use a plain white lit by two bright lamps - a white one and a blue one with an energy value of two (Figure 19-4).

For the wireframe cube, set the material type to 'Wire' and change the color to green (Figure 19-5).

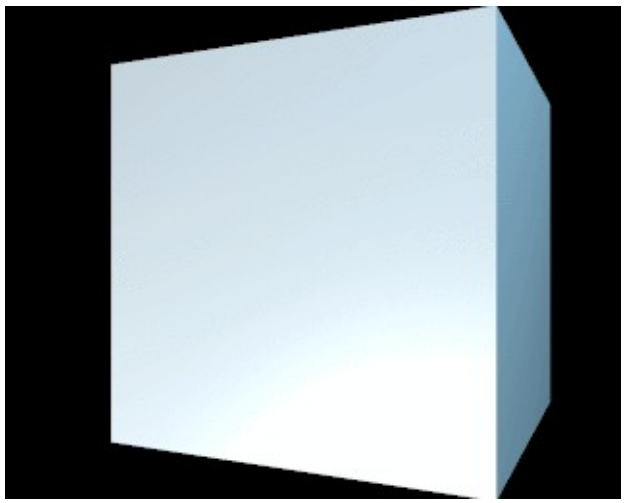


Figure 19-4. A rendering of the solid cube.

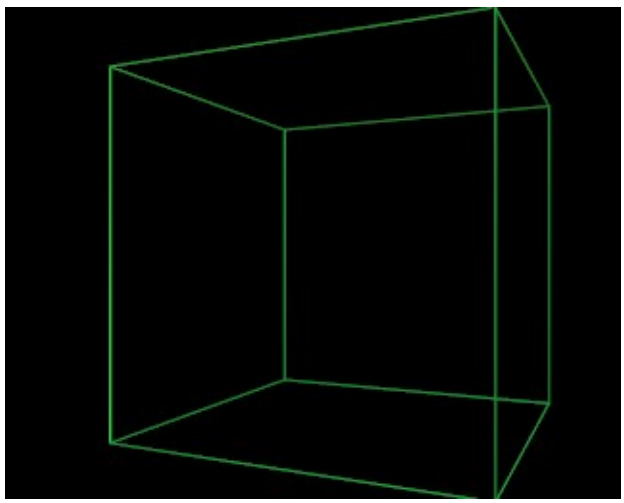


Figure 19-5. And a rendering of the wireframe cube.

Enter an appropriate filename (for example 'cube_solid.avi') in the 'Pics' field of the Render Buttons window (**F10**) (Figure 19-6).

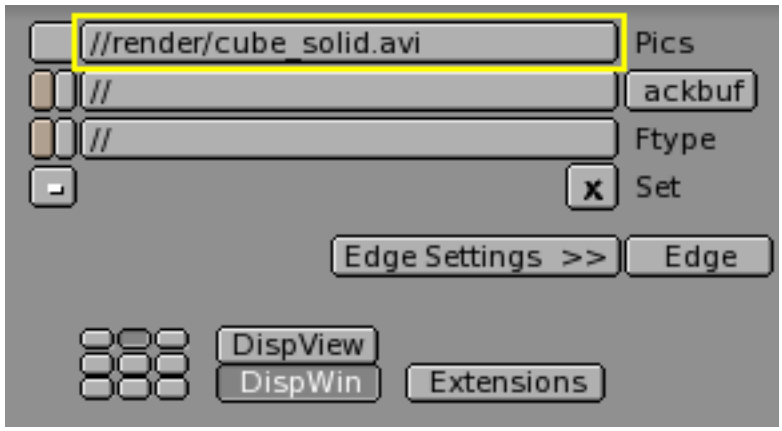


Figure 19-6. Set the animation output filename.

Render the animation with the white solid cube. This will save it to your disk. Save it as an AVI file. Use AVI Raw if possible, because it yelds an higher quality - compression should be the last thing in the editing process - otherwise, if short of disk space use AVI Jpeg or AVI Codec, the first being less compressed and hence often of higher quality.

Now change the material to the green wireframe, render the animation again and save the result as cube_wire.avi.

You now have a 'cube_solid.avi' and 'cube_wire.avi' on your hard disk. This is enought for our first sequence editing.

First Sequence: delayed wireframes

The first sequence will use only the wireframe animation - twice - to create an interesting effect. We will create multiple layers of video, give them a small time offset and add them together. This will simulate the 'glowing trail' effect that you see on radar screens.

Start a clean Blender file and change the 3D window to a Sequence Editor window by pressing **SHIFT F8** or by selecting the Sequence Editor icon Video Sequence Editor from the window header.

Add a movie to the window by pressing **SHIFT-A** and selecting 'Movie' (Figure 19-7). From the File Select window select the wireframe cube animation that you made before.

Figure 19-7. Adding a video strip

After you have selected and loaded the movie file, you see a blue strip that represents it. After adding a strip, you are automatically in grab mode. The start and end frame are now displayed in the bar.

Take a closer look at the Sequence Editor screen now. Horizontally you see the time value. Vertically, you see the video 'channels'. Each channel can contain an image, a movie or an effect. By layering different channels on top of each other and applying effects, you can mix different sources together. If you select a video strip, its type, length and filename will be printed at the bottom of the window.

Grab your video strip and let it start at frame 1. Place it in channel 1, that is on the bottom row (Figure 19-8).

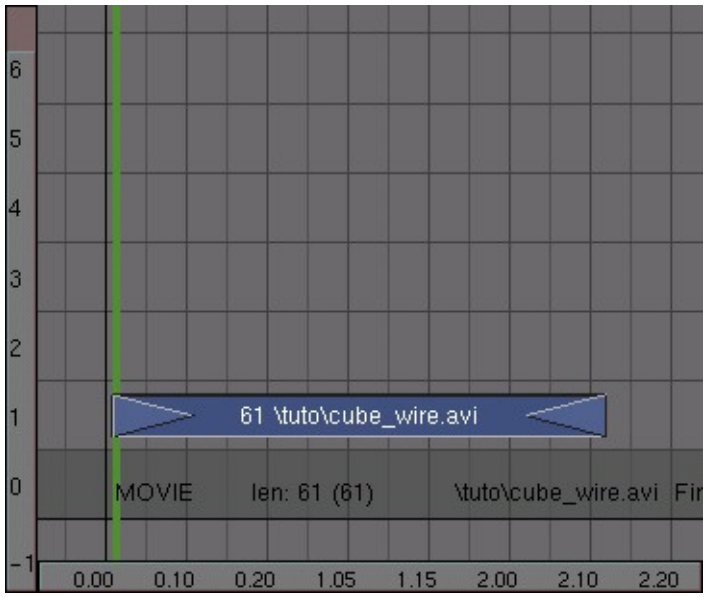


Figure 19-8. Placing the strip.

> **Lead-in, Lead-out and stills:** You can add *lead-in* and *lead-out* frames by selecting the triangles at the start and end of the strip (they will turn purple) and dragging them out. In the same way, you can define the 'length' in frames of a still image.

Duplicate the movie strip with **SHIFT D**, place the duplicate in channel 2 and shift it one frame to the right. We now have two layers of video on top of each other, but only one will display. To mix the two layers you need to apply an effect to them.

Select both layers and press **SHIFT-A**. Select ADD from the menu that pops up (Figure 19-9).

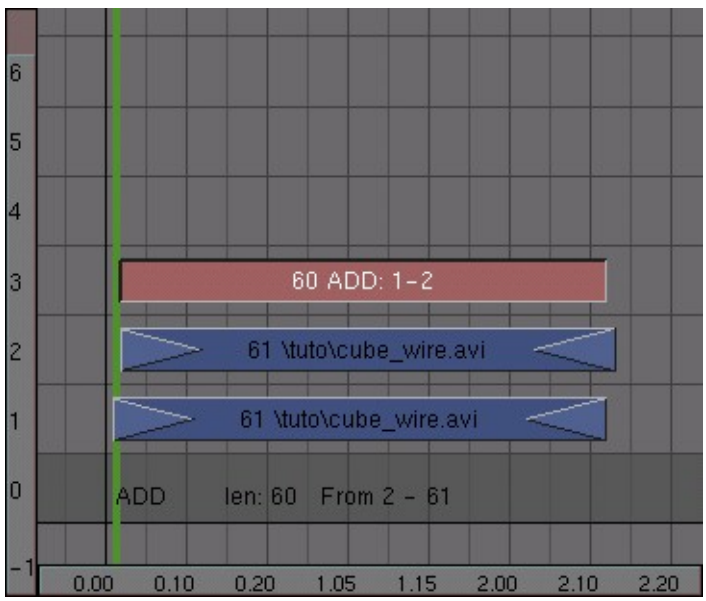


Figure 19-9. Mixing two video strips

To see what's happening split the sequence editor window and select the image button in the header (Figure 19-10). This will activate the automatic preview (Figure 19-11). If you select a frame in the sequence editor window with the strips, the preview will be automatically updated (with all the effects applied!).

Figure 19-10. Sequence Editor preview button.

If you press **ALT A** in the preview window, Blender will play back the animation. (Rendering of effects for the first time takes a lot of processing time, so don't expect a real-time preview!).

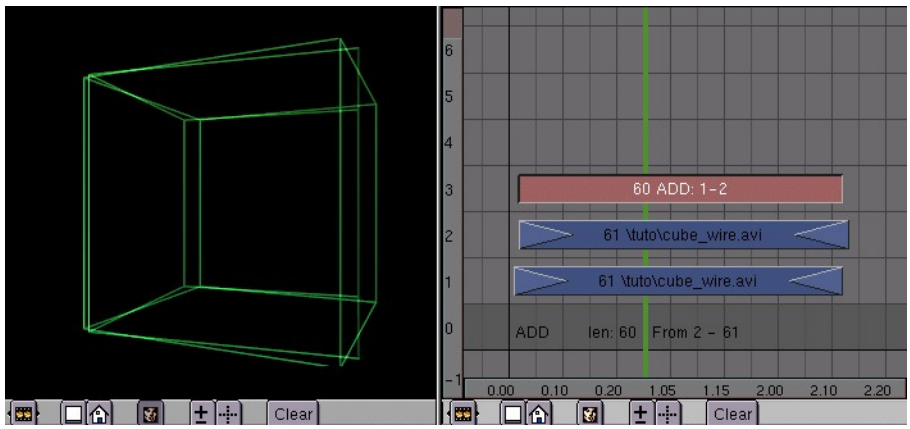


Figure 19-11. Adding a preview window.

> **Windowless preview:** If you do not like the separate render window, switch to the Render Buttons (**F10**) and select DispView in the bottom left.

Now its time to add some more mayhem to this animation. Duplicate another movie layer and add it to the ADD effect in video channel 3. Repeat this once and you will have four wireframe cubes in the preview window (Figure 19-12).

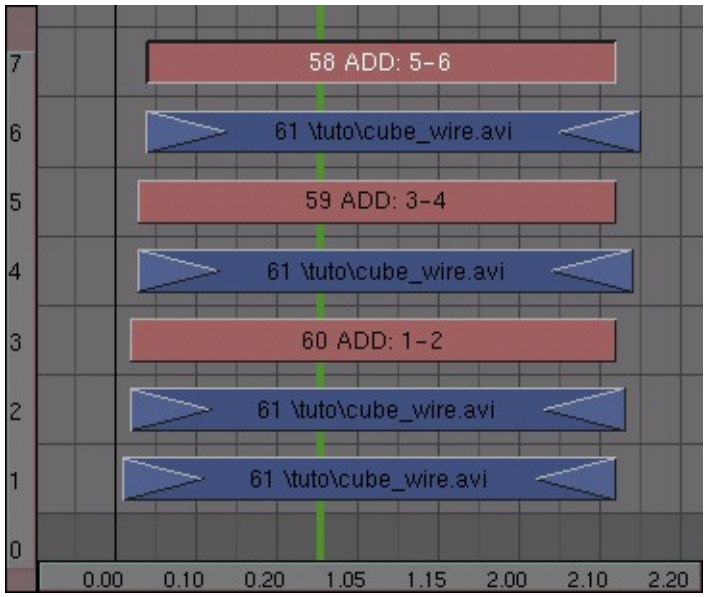


Figure 19-12. Sequence with 4 wireframe cube strips added together.

All the cubes have the same brightness now, but I would like to have a falloff in brightness. This is easily arranged: open an IPO window somewhere (**F6**) and select the sequence icon in its header (Figure 19-13).

Figure 19-13. Sequence IPO button.

Select the first add strip (the one in channel 3), hold down **CTRL** and click **LMB** in the IPO window on a value of 1. This sets the brightness of this add operation to maximum. Repeat this for the other two add strips, but decrease the value a bit for each of them, say to around 0.6 and 0.3 (Figure 19-14).

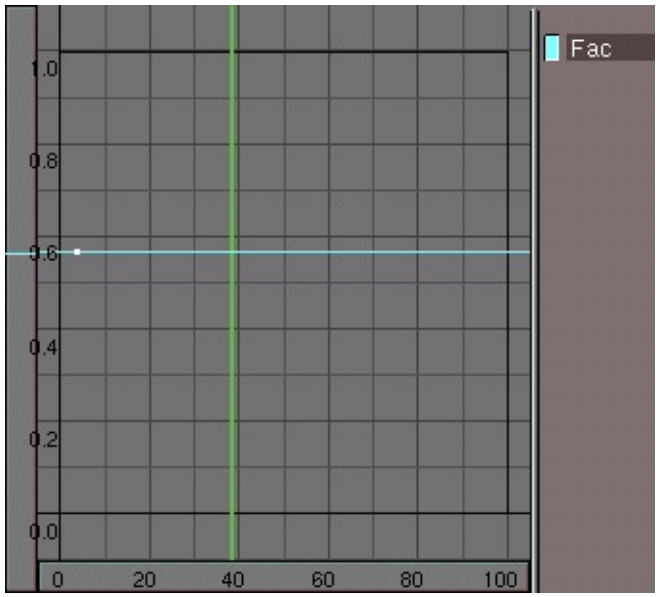


Figure 19-14. Defining the brightness of a layer with an IPO

Depending on the ADD values that you have just set, your result should look something like what is shown in Figure 19-15.

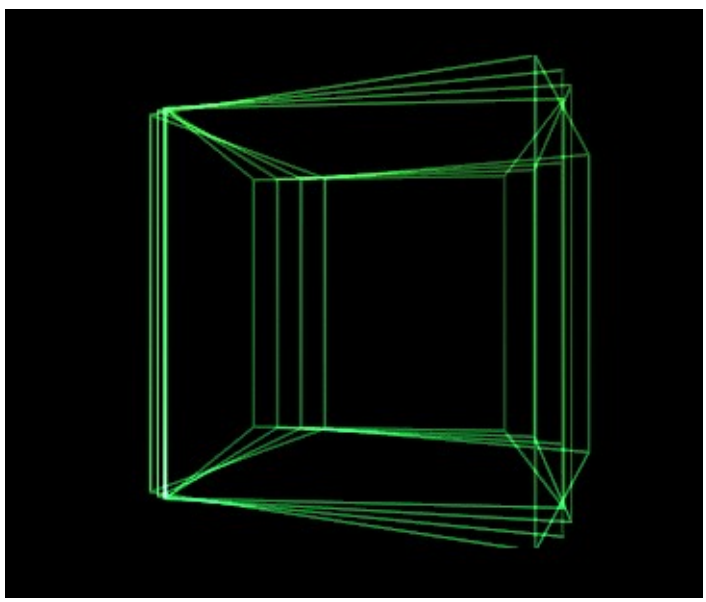


Figure 19-15. Four wireframe cubes combined with fading effects.

Now we already have 7 strips and we have only just begun with our animation! You can imagine that the screen can quickly become very crowded indeed. To make your project more manageable, select all strips (**AKEY** and **BKEY** work here, too!), press **MKEY** and press ENTER or click on the `Make Meta` pop up. The strips will now be combined into a meta-strip, and can be copied or moved as a whole.

With the meta strip selected, press N and enter a name, for example 'Wire/Delay', to better remember what it is Figure 19-16.

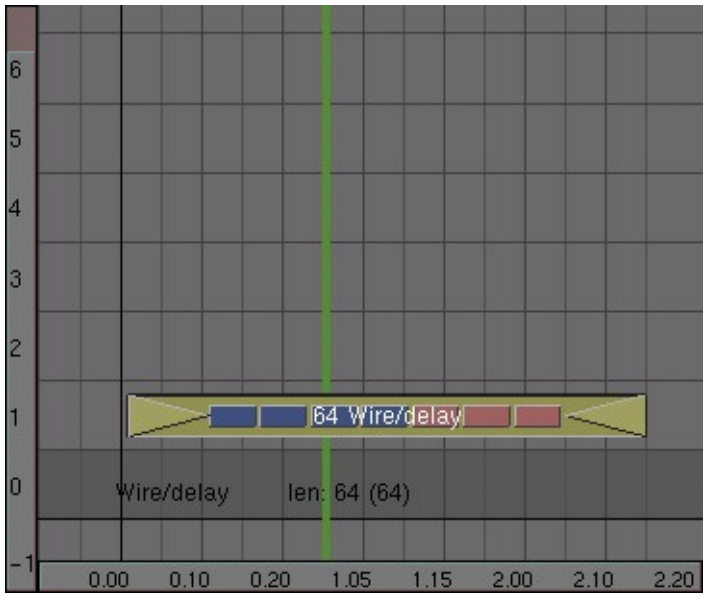


Figure 19-16. Named META strip

Second animation: A delayed solid cube

Now it is time to use some masks. We want to create two areas in which the animation plays back with 1 frame time difference. This creates a very interesting glass-like visual effect.

Start by creating a black and white image like this one. You can use a paint program or do it in Blender. The easiest way to do this in Blender is to create a white material with an emit value of 1 or a shadeless white material on some beveledCurve Circles(Figure 19-17). In this way, you do not need to set up any lamps. Save the image as mask.tga.

Figure 19-17. Animation mask.

Switch to the sequence editor and move the meta strip that we made before out of the way (we will reposition it later). Add the animation of the solid cube (SHIFT+A, 'Movie'). Next, add the mask image. By default a still image will get a length of 50

frames in the sequence editor. Change it to match the length of the cube animation by dragging out the arrows on the side of the image strip with the right mouse button.

Now select both strips (hold down **SHIFT**), press **SHIFT+A** and add a SUB (subtract) effect. (Figure 19-18).

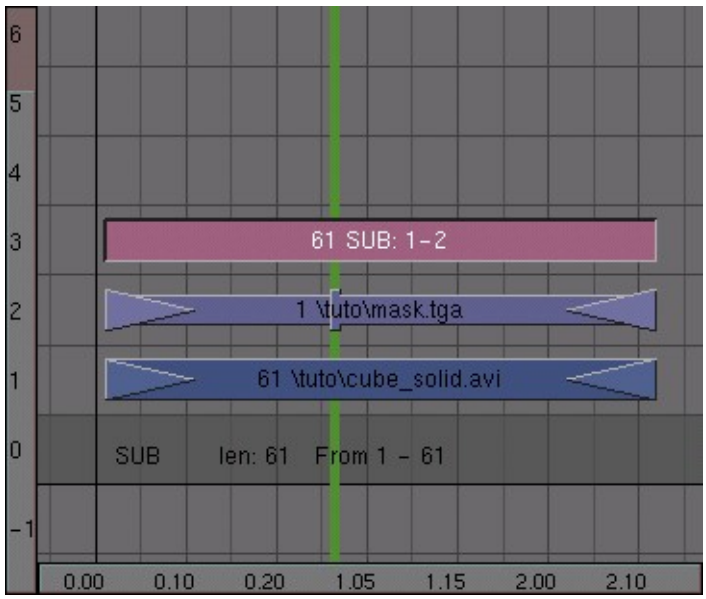


Figure 19-18. Subtracting the mask from the video.

In the preview window you will now see the effect; the areas where the mask is white have been removed from the picture (Figure 19-19).

Figure 19-19. Mask subtracted.

This effect is ready now; select all three strips and convert them into a META strip by pressing **MKEY**

Now do the same, except that you don't use the SUB effect but the MUL (multiply) effect (Figure 19-20). This time you will only see the original image where the mask image is white. Turn the three strips of this effect into a meta strip again.

Figure 19-20. Mask multiplied.

For the final step I have to combine the two effects together. Move one of the meta strips above the other one and give it a time offset of one frame. Select both strips and add an ADD effect (Figure 19-21).

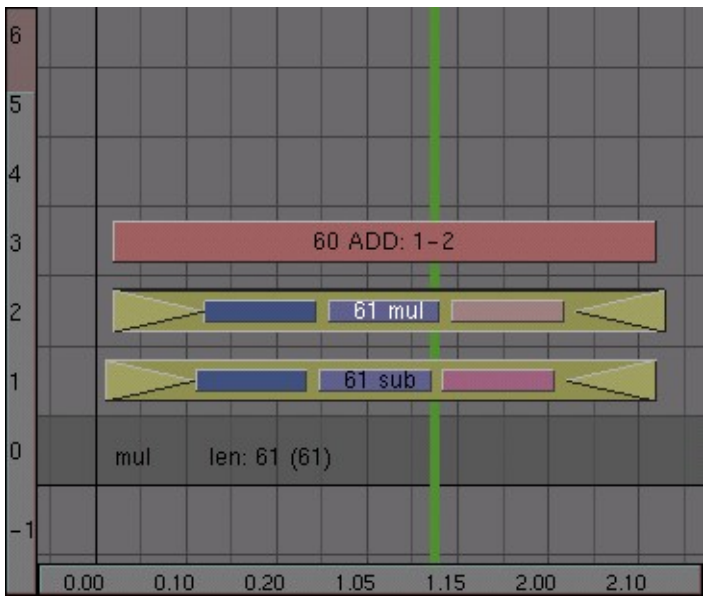


Figure 19-21. Adding the two effects

In the preview window you can now see the result of the combination of the animation and the mask (Figure 19-22).

When you are ready, select the two meta strips and the ADD effect and convert them into a new meta strip. (That's right! You can have meta strips in meta strips!)

> **Getting into a Meta Strip:** To edit the contents of a meta strip, select it and press **TAB**. The meta strip will 'explode' to show its components and background will turn yellow/greenish to indicate that you are working inside a meta strip. Press **TAB** again to return to normal editing.

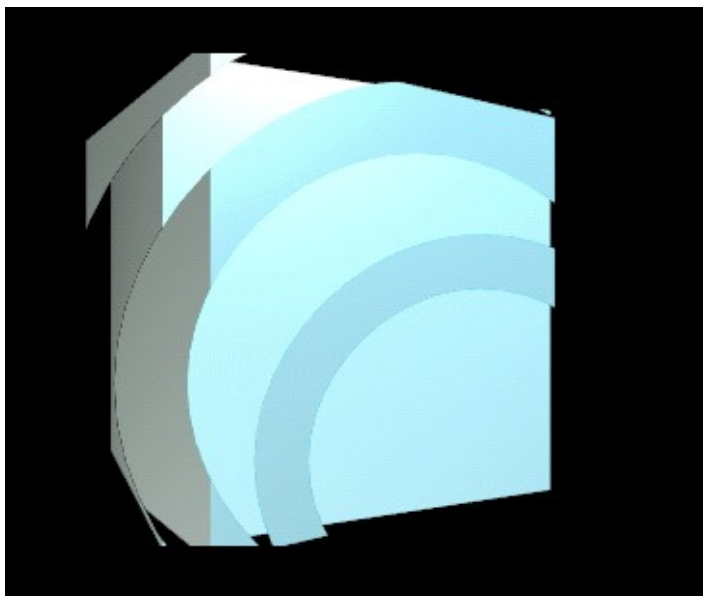


Figure 19-22. Two time-shifted layers.

Third animation: a tunnel

We want a third 'effect'to further enrich our animation; a 3D 'tunnel' to be used as a background effect. This is really simple to create. First save your current work - you will need it later!

Start a new scene (**CTRL-X**) and delete the default plane. Switch to front view (**NUM 1**). Add a 20-vertex circle about 10 units under the z=0 line (the pink line in your screen) (Figure 19-23).

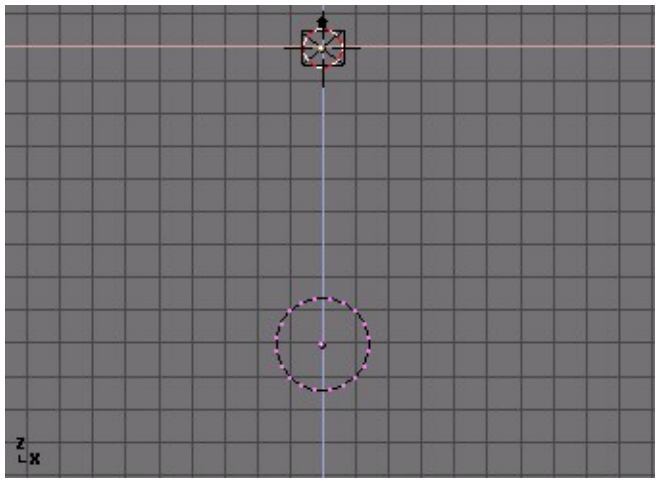


Figure 19-23. Adding a 20-vertex circle.

While still in editmode, switch to side view (**NUM 3**) and snap the cursor to the origin by locating it roughly at the x,y,z=0 point and pressing **SHIFT-S**. Select `Curs>>Grid`.

We want to turn the circle into a circular tube, or torus. For this, we will use the Spin function. Go to the Edit Buttons window (F9) and enter a value of 180 in the `Degr` NumButton and enter '10' in the `Steps` one. Pressing `Spin` will now rotate the selected vertices around the cursor at 180 degrees and in 10 steps. (Figure 19-24).

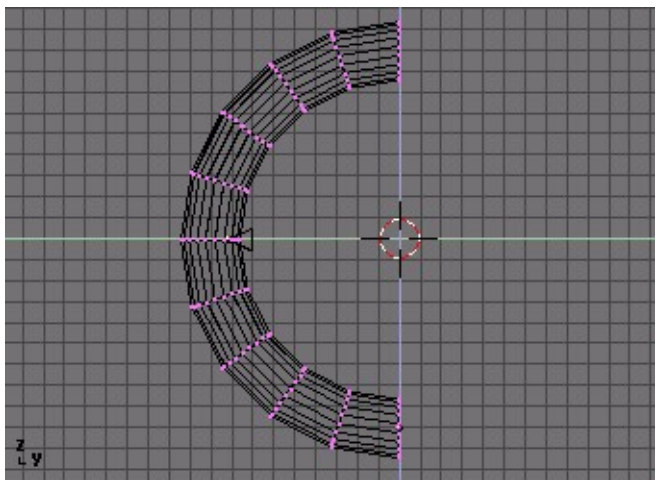


Figure 19-24. Spinning the circle around the cursor

Leave editmode (**TAB**). With the default settings, Blender will always rotate and scale around the object's center which is displayed as a tiny dot. This dot is yellow when the object is unselected and pink when it is selected. With the cursor still in the origin, press the `Center Cursor` button in the Edit Buttons window to move the object center to the current cursor location. Now press **RKEY** and rotate the tube 180 degrees around the cursor.

Now it's time to move the camera into the tunnel. Open another 3D window and switch it to the camera view (NUMPAD+0). Position the camera in the side view window to match Figure 19-25, the camera view should now match Figure 19-26.

> **Missing edges:** If not all of the edges of the tunnel are showing, you can force Blender to draw them by selecting 'All Edges' in the Edit Buttons window (**F9**).

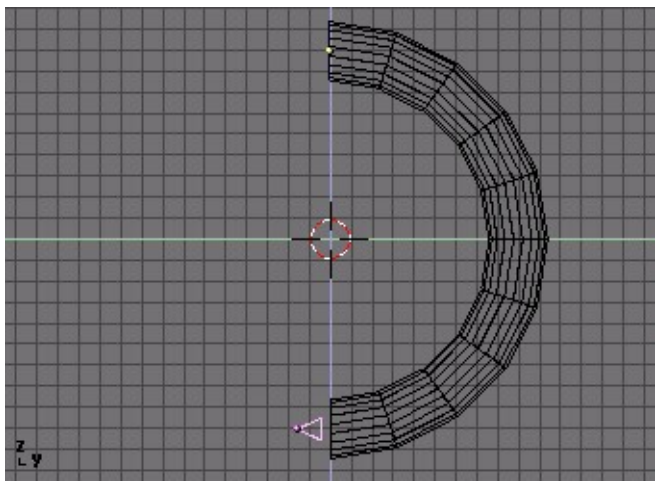


Figure 19-25. Camera inside the tunnel.

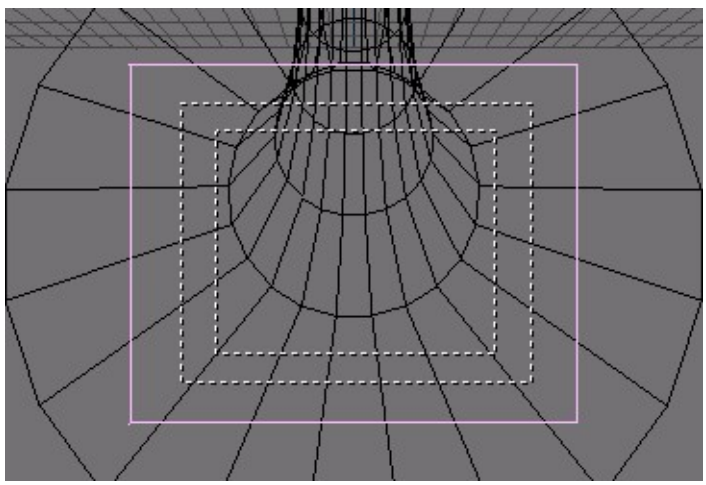


Figure 19-26. Camera view of the tunnel interior.

To save ourselves some trouble, I want to render this as a looping animation. I can then add as many copies of it as I like to the final video compilation.

There are two things to keep in mind when creating looping animations. First, make sure that there is no 'jump' in your animation when it loops. For this, you have to be careful when creating the keyframes and when setting the animation length. Create two keyframes: one with the current rotation of the tube on frame 1, and one with a rotation of 90 degrees (hold down **CTRL** while rotating) on frame 51. In your animation frame 51 is now the same as frame1, so when rendering you will need to leave out frame 51 and render from 1 to 50.

Please note that the number 90 degrees is not chosen carelessly, but because the tunnel is periodic with period 18°, hence you must rotate it by a multiple of 18°, and 90° is it, to guarantee that frame 51 is exactly the same than frame 1.

Second, to get a *linear* motion you need to remove the ease-in and ease-out of the rotation. These can be seen in the IPO window of the tube after inserting the rotation

keyframes. The IPO smoothly starts and end, much like a cosine function. We want it to be straight. To do so select the rotation curve, enter editmode (**TAB**) and select all vertices (**AKEY**) and press **VKEY** ('Vector') to change the curve into a linear one (Figure 19-27).

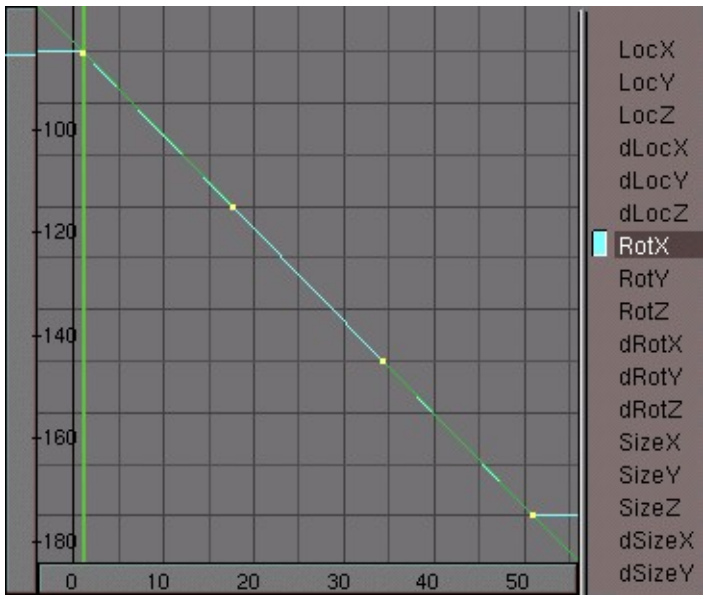


Figure 19-27. Tunnel rotation IPO without ease-in and ease-out.

To create a more dramatic effect, select the camera while in camera view mode (Figure 19-28). The camera itself is displayed as the solid square. Press **RKEY** and rotate it a bit. If you now play back your animation it should loop seamlessly.

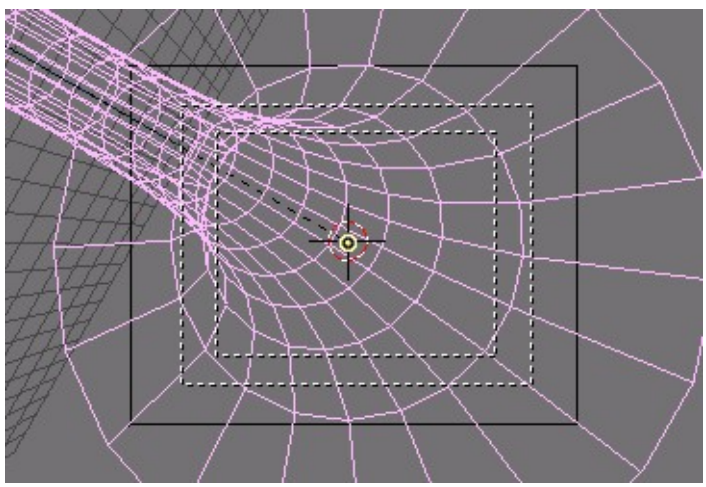


Figure 19-28. Figure Rotate the camera to get a more dramatic effect

For the final touch, add a blue wireframe material to the tube and add a small lamp on the location of the camera. By tweaking the lamp's 'Dist' value (attenuation distance) you can make the end of the tube disappear in the dark without having to work with mist. (Figure 19-29).

When you are satisfied with the result, render your animation and save it as 'tunnel.avi'.

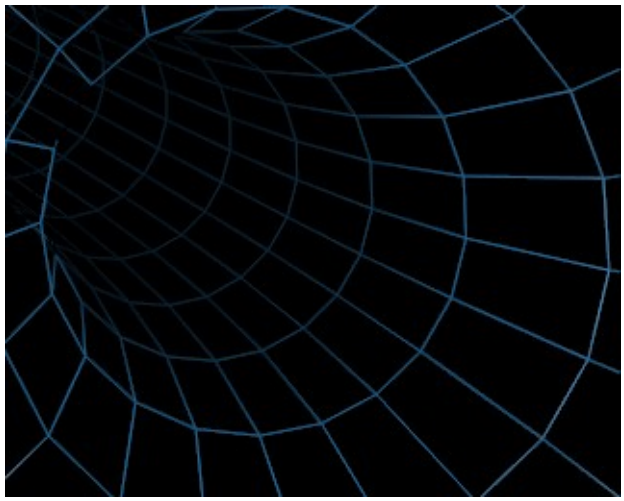


Figure 19-29. Figure A groovy tunnel.

Second sequence: Using the tunnel as a backdrop

Reload your video compilation Blender file. The tunnel that we made in the last step will be used as a backdrop for the entire animation. To make it more interesting I will modify an ADD effect to change the tunnel into a pulsating backdrop. Prepare a completely black picture and call it 'black.tga' (try pressing **F12** in an empty Blender file. Save with **F3**, but make sure that you have selected the TGA file format in the Render Buttons window). Add both black.tga and the tunnel animation and combine them with an ADD effect (Figure 19-30).

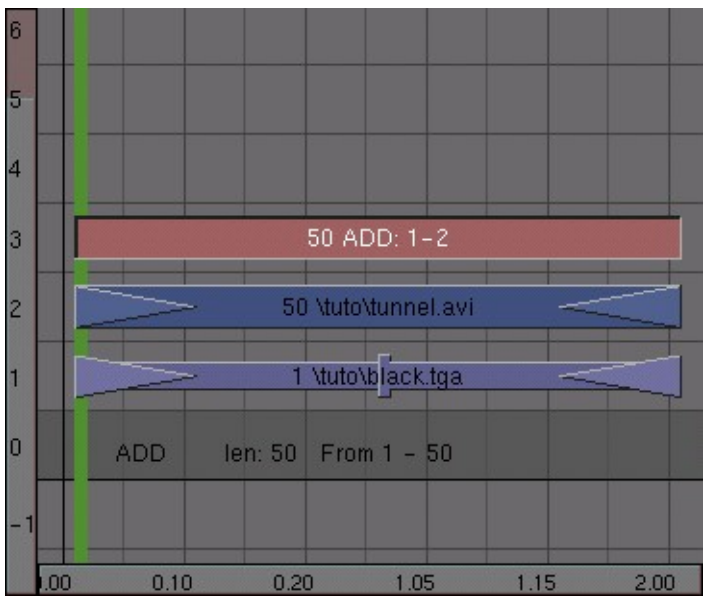


Figure 19-30. Setting up the backdrop effect.

Now with the ADD effect selected, open an IPO window and select the Sequence Editor button in its header. From frame 1-50, draw an irregular line by holding down **CTRL** and left-clicking. Make sure that the values are between 0 and 1 (Figure 19-31).

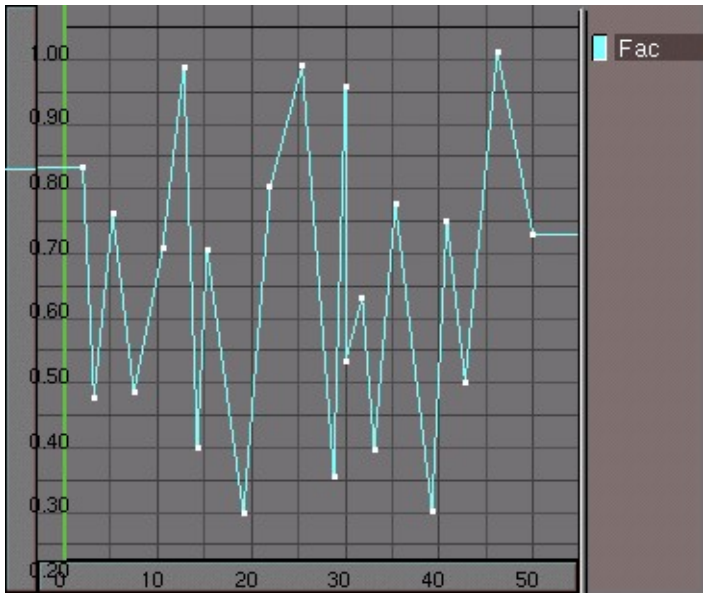


Figure 19-31. Adding randomnes with a irregular Ipo

When you are ready, take a look at the result in a preview screen and change the animation into a meta strip.

Save your work!

Fourth Animation: a jumping logo

Let's create some morerandomness and chaos! Take a logo (We can just add a text object) and make it jump through the screen. Again, the easiest way to do this is to add vertices directly into the IPO window (select a LocX, LocY or LocZ channel first), but this time you may need to be a bit more careful with the minimum and maximum values for each channel. Don't worry about the looks of this one too much - the next step will make is hardly recognizable anyway. (Figure 19-32).

Figure 19-32. Jumping logo

Save the animation as 'jumpylogo.avi'.

Fifth Animation: particle bars

Our last effect will use an animated mask. By combining this with the logo of the previous step, I will achieve a streaking effect that introduces the logo to our animation. This mask is made by using a particle system. To set one up switch to side view, add a plane to your scene and while it is still selected switch to the Animation Buttons window (**F7**). Select 'New effect' and then change the default effect (build) to 'Particles'. Change the system's settings as indicated in Figure 19-33.

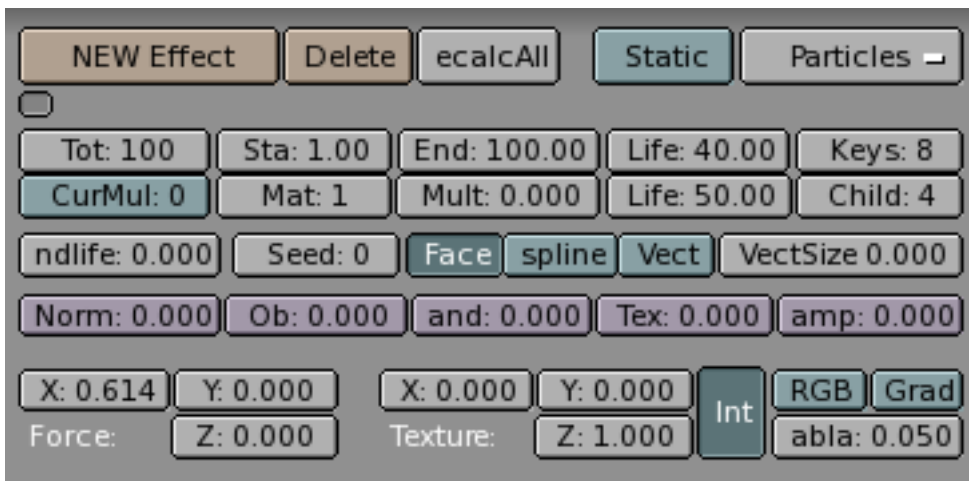


Figure 19-33. Particle system settings.

Press **TAB** to enter editmode, select all vertices and subdivide the plane twice by pressing **WKEY** and selecting `Subdivide` from the pop-up menu.

Next switch to front view and add another plane. Scale it along the X-axis to turn it into a rectangle (press **SKEY** and move your mouse horizontally. Then click **MMB** to

scale along the indicated axis only). Give the rectangle a white material with an emit value of one.

Now you need to change the particles into rectangles by using the dupliverts function. Select rectangle, then particle emitter and parent them. Select only the plane and in the left part of the animation buttons window, select the DupliVerts button. Each particle is now replaced by a rectangle (Figure 19-34).

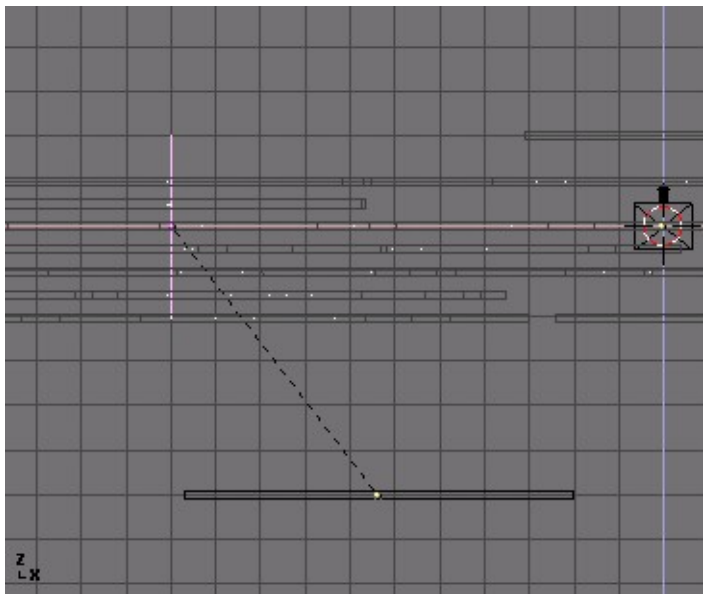


Figure 19-34. Dupliverted rectangles

I now add some mist as a quick hack to give the rectangles each a different shade of gray. Go to the World Buttons window with **FKEY** , click on the button in its header and select `Add New`. The world settings will now appear.

By default, the sky will now be rendered as a gradient between blue and black. Change the horizon colors (HoR, HoG, HoB) to pure black (Figure 19-35).

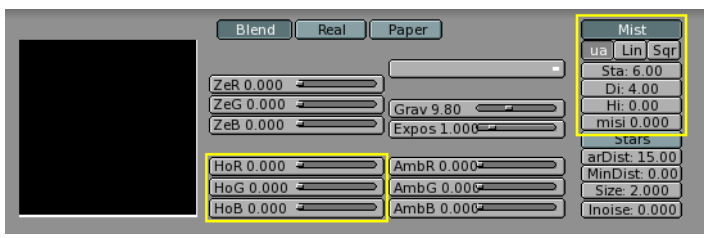


Figure 19-35. Figure Setting up mist.

To activate rendering of mist activate the Mist button in the middle of the screen. When using mist, you have to indicate on which distance from the camera it works. Select the camera, switch to the Edit Buttons window and enable `ShowLimits`. Now switch to top view and return to the World Buttons window. Tweak the `Sta:` and `Di:` (Start, Distance, respectively) parameters so that the mist covers the complete width of the particle stream (Figure 19-35 and Figure 19-36).

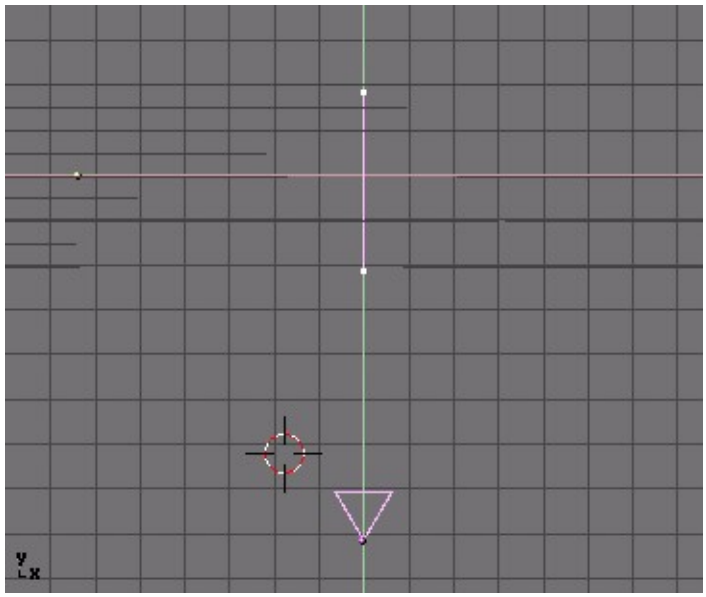


Figure 19-36. Setting the mist parameters

Set the animation length to 100 frames and render the animation to disk. Call the file 'particles.avi' (Figure 19-37).

Figure 19-37. Rendered particle rectangles.

Third sequence: Combining the logo and the particle bars

By now you know the drill: reload your compilation project file, switch to the Sequence Editor window and add both 'particles.avi' and 'logo.avi' to your project. Combine them together with a MUL effect. Since the logo animation is 50 frames and the particles animation is 100 frames, you'll need to duplicate the logo animation once and apply a second MUL effect to it (Figure 19-38 and Figure 19-38).

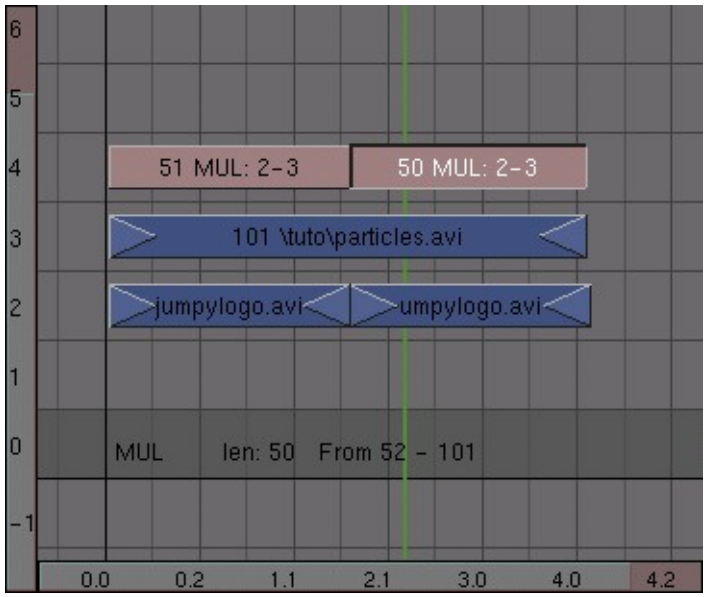


Figure 19-38. Use the logo animation twice

Combine these three strips into one meta strip. If you're feeling brave you can make a few copies and give them a small time offset just like with the wireframe cube.

Figure 19-39. The particles animation combined with the logo animation

Sixth Animation: zooming logo

If you would combine all your animations so far you would get a really wild video compilation, but if this was your company's presentation you would want to present the logo in a more recognizable way. The final part of our compilation will therefore be an animation of the logo that zooms in very slowly. Prepare this one and save it as 'zoomlogo.avi'. Also prepare a white picture and save it as 'white.tga'.

We will now use the CROSS effect to first make a rapid transition from black to white, then from white to our logo animation. Finally, a transition to black will conclude the compilation.

Start off by placing black.tga in channel 1 and white.tga in channel 2. Make them both 20 frames long. Select them both and apply a cross effect. The cross will gradually change the resulting image from layer 1 to layer 2. In this case, the result will be a transition from black to white (Figure 19-40).

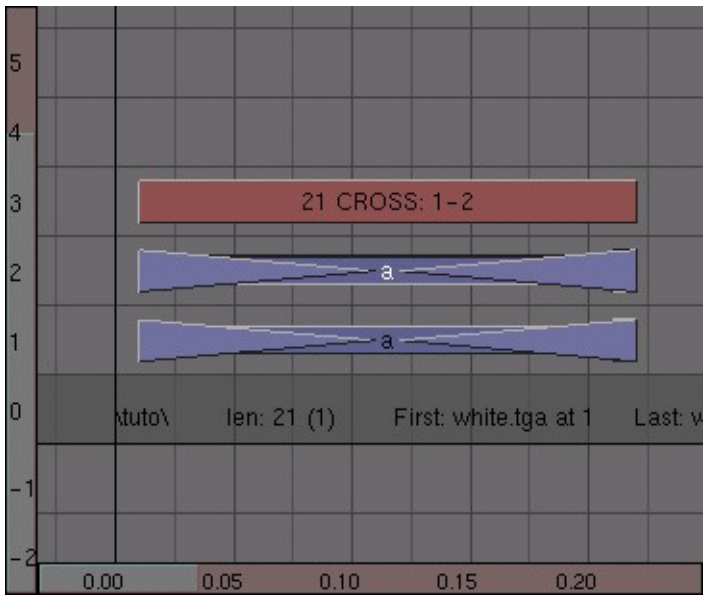


Figure 19-40. Black-white transition.

Next, add a duplicate of white.tga to layer 1 and place it directly to the right of black.tga. Make it about half as long as the original. Place the logo zoom animation in layer 2 and add a cross effect between the two. At this point, the animation looks like a white flash followed by the logo zoom animation (Figure 19-41).

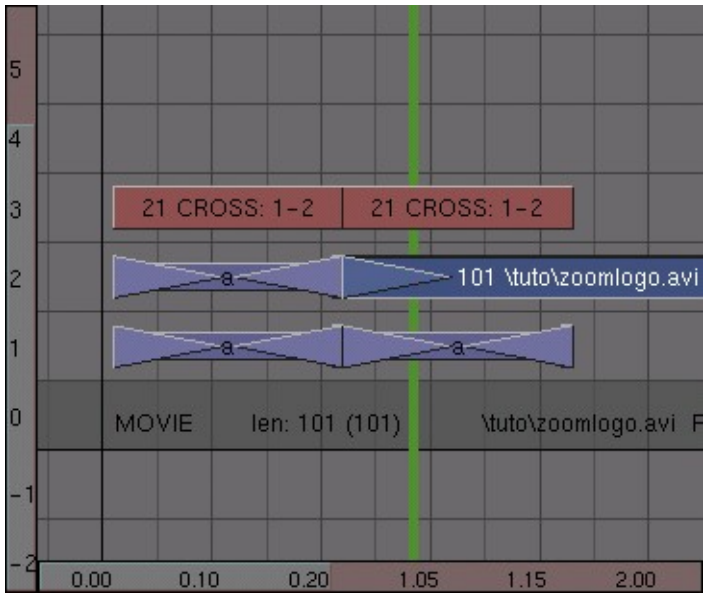


Figure 19-41. Figure White-video transition

The last thing that you need to do is to make sure that the animation will have a nice transition to black at the very end. Add a duplicate of black.tga and apply another cross effect. When you are ready, transform everything into a meta strip (Figure 19-42).

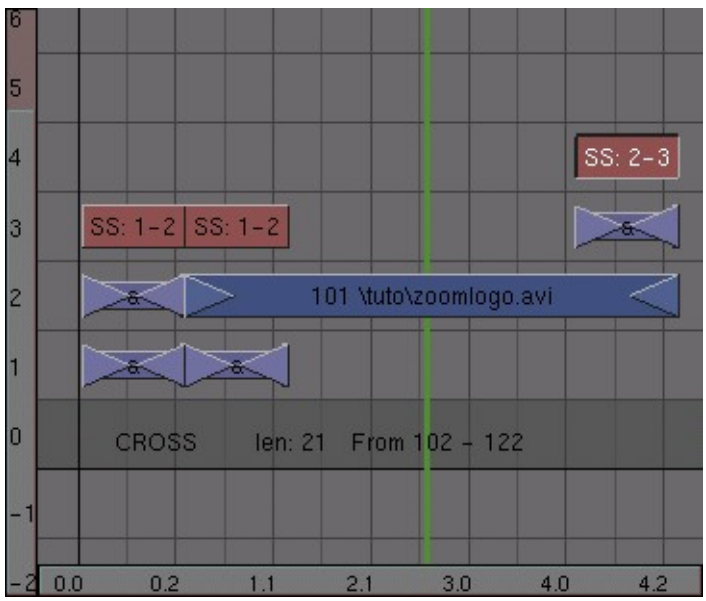


Figure 19-42. Video-black transition

Assembling everything so far

We're at the end of our work! It's time add some of the compilations that we have made so far and see how our work looks. The most important thing to remember while creating your final compilation is that when rendering your animation, the sequence editor only 'sees' the top layer of video. This means that you have to make

sure that it is either a strip that is ready to be used, or it should be an effect like ADD that combines several underlying strips.

The foundation of the compilation will be the fluctuating tunnel. Add a some duplicates of the tunnel meta strip and place them in channel one. Combine them into one meta strip. Do not worry about the exact length of the animation yet; you can always duplicate more tunnel strips.

On top of that, place the delayed wireframe cube in channel 2. Add channel 1 to channel two and place the add effect in channel 3 (Figure 19-43).

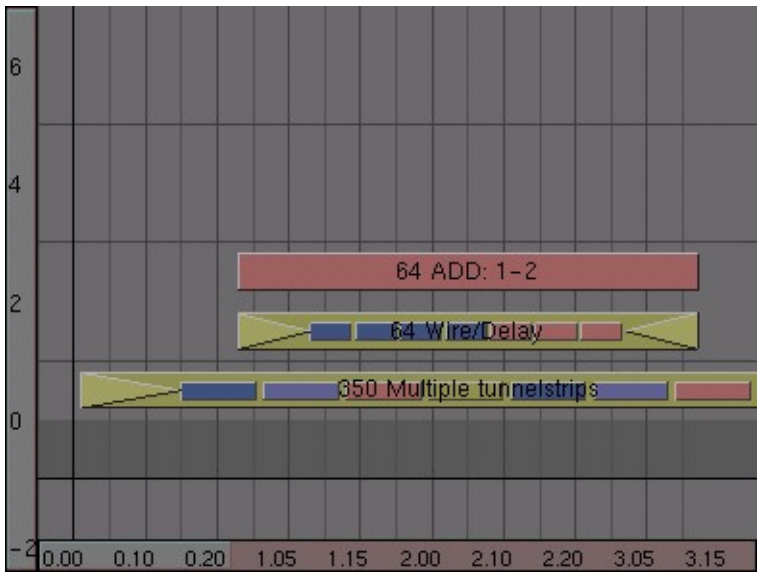


Figure 19-43. Combining the tunnel and the wireframe cube

Now we also want to add the solid cube animation. Place it in channel 4, overlapping with the wireframe animation in channel 2. Add it to the tunnel animation in layer one. This is where things are starting to get a little tricky; if you would leave it like this, the animation in channel 5 (the solid cube together with the tube) would override the animation in channel 2 (the wireframe cube) and the wireframe cube would become invisible as soon as the solid cube shows up. To solve this, add channel 3 to channel 5 (Figure 19-44).

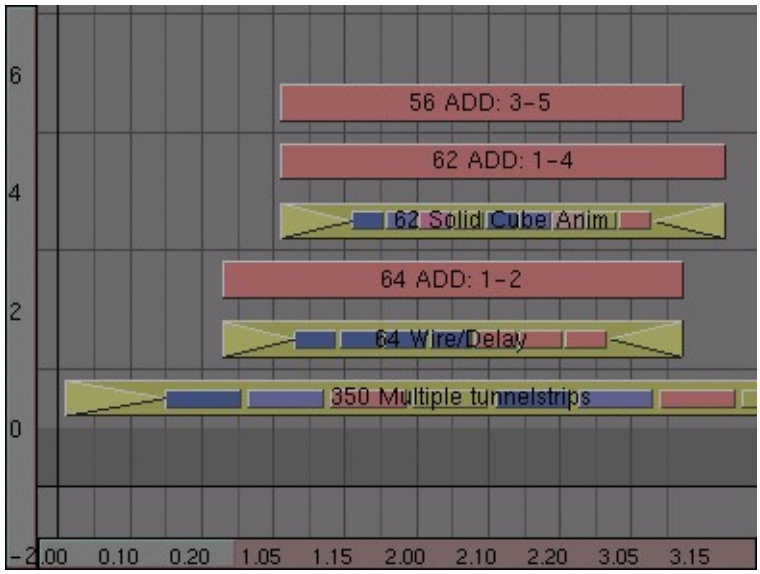


Figure 19-44. Combining the tunnel, wireframe and solid cube.

You will often need to apply some extra add operations to fix missing parts of video. This will most likely become apparent after you have rendered the final sequence.

Slide the Sequence Editor window a bit to the left and add the meta strip with the particle/logo animation in it. Place this strip in layer 2 and place an add effect in layer 3. For some variation, duplicate the wireframe animation and combine it with the add in layer 3 (Figure 19-45).

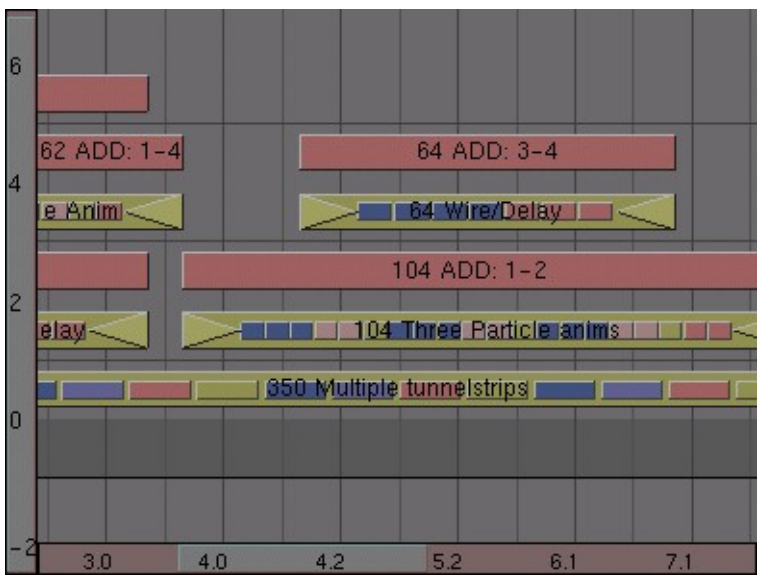


Figure 19-45. Adding the particle/logo animation

Now go to the end of the tunnel animation strip. There should be enough place to put the logo zoom animation at the end and still have some space left before it (Figure 19-46). If not, select the tunnel strip, press **TAB** and add a duplicate of the animation to the end. Press **TAB** again to leave meta edit mode.

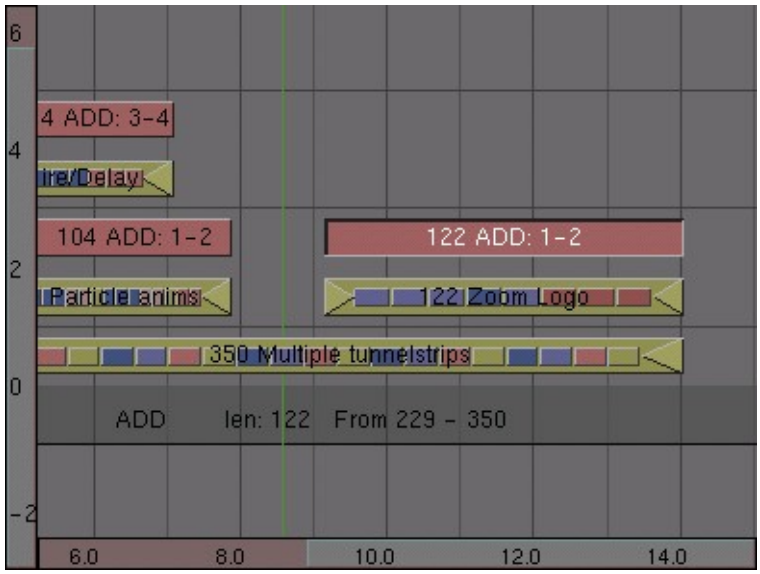


Figure 19-46. Adding the logo zoom animation.

If there is still some space left, wecan add a copy of the solid cube animation. To get it to display correctly, you will have to apply two add channels to it: one to combine it with the particle logo animation and one to combine it with the logo zoom animation (Figure 19-47).

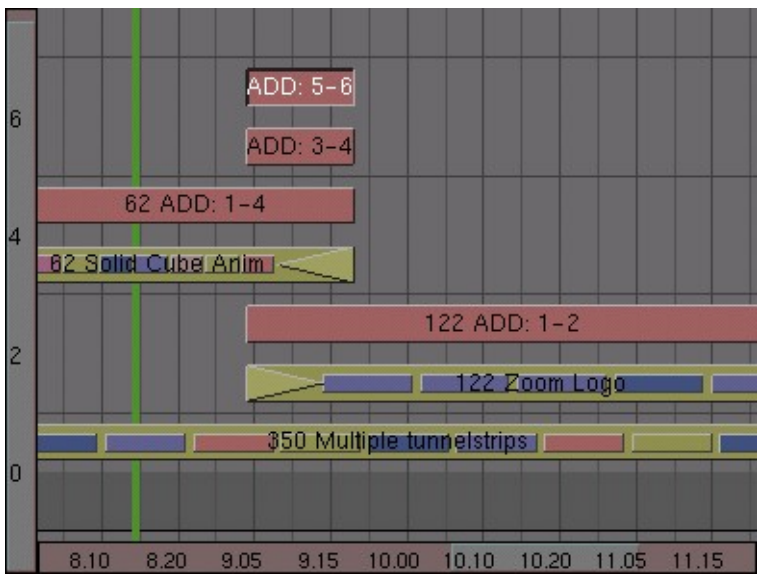


Figure 19-47. Adding one last detail

Figure 19-48 shows the complete sequence.

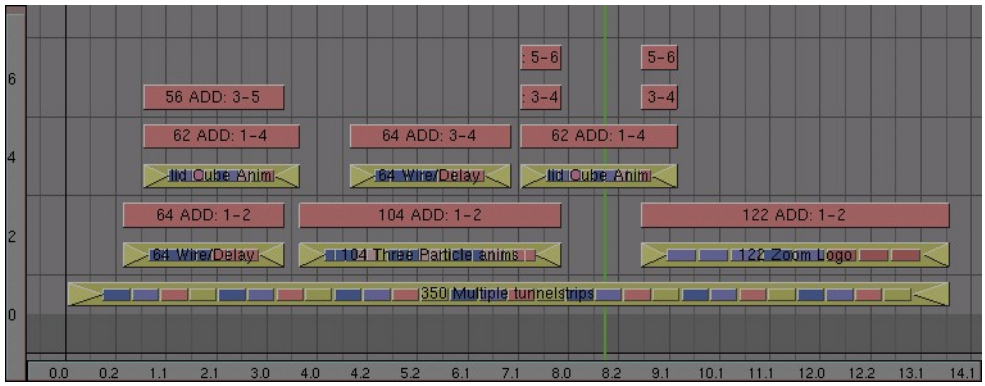


Figure 19-48. The complete sequence

Conclusion

We are now ready to render our final video composition! To tell Blender to use the Sequence Editor information while rendering, select the 'Do Sequence' button in the Render Buttons window. After that, rendering and saving your animation works like before (be sure not to overwrite any of your AVI of the sequence!).

Sequence Editor Plugins (-)

TBW

Chapter 20. Python Scripting

Blender has a very powerful yet often overlooked feature. It exhibits an internal full fledged Python interpreter.

This allows any user to add functionalities by writing a Python script. 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. It was expressely designed to be usable as an extension language for applications that need a programmable interface, and this is why Blender uses it.

Blender has a "Text window" among its windows types accessible via the button of the window type menu or via **SHIFT F11**.

The newley opened Text window is grey and empty, with a very simple toolbar (Figure 20-1). From left to right there are the standard Window type selection button and the fullscreen button, followed by a toggle button which shows/hides the line numbers for the text and the regular select button.

Figure 20-1. Text Toolbar.

The select button ( ) allows to select which Text buffer is to be displayed, as well as allowing to create a new buffer or loading a text file.

Once a text buffer is in the Text window, this behaves as a very simple text editor. Typing on the keyboard produces text in the text buffer. As usual pressing, **LMB** dragging and releasing **LMB** selects text. The following keyboard commands apply:

- **ALT C** or **CTRL C** - Copy the marked text into a buffer;
- **ALT X** or **CTRL X** - Cut out the marked text into a buffer;
- **ALT V** or **CTRL V** - Paste the text from buffer to the cursor in the textwindow;
- **ALT S** - Saves the text as a textfile, a FileWindow appears;
- **ALT O** - Loads a text, a FileWindow appears;
- **SHIFT ALT F** or **RMB** - Pops up the Filemenu for the TextWindow;
- **ALT J** - Pops up a NumButton where you can specify a linenumber the cursor will jump to;
- **ALT P** - Executes the text as a Python script;
- **ALT U** or **CTRL U** - Undo;
- **ALT R** or **CTRL R** - Redo;
- **ALT M** - Converts the content of the text window into 3D text (max 100 chars);

Blender's cut/copy/paste buffer is *separate* from Window's clipboard. So normally you *cannot* cut/paste/copy out from/into Blender. To access your Windows clipboard use **SHIFT-CTRL-C SHIFT-CTRL-V**

To delete a text buffer just press the 'X' button next to the buffer's name, just as you do for materials, etc.

The most notable keystroke is **ALT P** which makes the content of the buffer being parsed by the internal Python interpreter built into Blender.

The next section will present an example of Python scripting. Before going on it is worth noticing that Blender comes with only the bare Python interpreter built in,

and with a few Blender-specific modules, those described in the Section called *API Reference*.

to have access to the standard Python modules you need a complete working python install. You can download this from http://www.python.org. Be sure to check on http://www.blender.org which is the *exact* Python version which was built into BLender to prevent compatibility issues.

Blender must also be made aware of *where* this full Python installation is. This is done by defining a `PYTHONPATH` environment variable.

Setting `PYTHONPATH` on Win95,98,Me

Once you have installed Python in, say, `C:\PYTHON22` you must open the file `C:\AUTOEXEC.BAT` with your favorite text editor, add a line:

```
SET PYTHONPATH=C:\PYTHON22;C:\PYTHON22\DLLS;C:\PYTHON22\LIB;C:\PYTHON22\LIB\LIB-
TK
```

and reboot the system.

Setting `PYTHONPATH` on WinNT,2000,XP

Once you have installed Python in, say, `C:\PYTHON22` Go on the "My Computer" Icon on the desctop, **RMB** and select `Properties`. Select the `Advanced` tab and press the `Environment Variables` button.

Below the System Variables box, (the second box), hit `New`. If you are not an administrator you might be unable to do that. In this case hit `New` in the upper box.

Now, in the Variable Name box, type PYTHONPATH, in the Variable Value box, type:

```
SET PYTHONPATH=C:\PYTHON22;C:\PYTHON22\DLLS;C:\PYTHON22\LIB;C:\PYTHON22\LIB\LIB-
TK
```

Hit OK repeatedly to exit from all dialogs. You may or may not have to reboot, dependingo on the OS.

Setting `PYTHONPATH` on Linux and other UNIXes

Normally you will have Python already there. if not, install. You will have to discover where it is. This is easy, yust start a python interactive shell opening a shell and typing `python` in there. Type the following commands:

```
>>> import sys
>>> print sys.path
```

and note down the output, it should look like

```
[”, ’/usr/local/lib/python2.2’, ’/usr/local/lib/python2.2
/plat-linux2’, ’/usr/local/lib/python2.0/lib-tk’, ’/usr/lo
cal/lib/python2.0/lib-dynload’, ’/usr/local/lib/python2.0/
site-packages’]
```

Add this to your favourite `rc` file as an evironment variable setting. For example, add in your `.bashrc` the line

```
export PYTHONPATH=/usr/local/lib/python2.2:/usr/local/lib/
python2.2/plat-linux2:/usr/local/lib/python2.2/lib-tk:/usr
/local/lib/python2.2/lib-dynload:/usr/local/lib/python2.0/
site-packages
```

all on a single line. Open a new login shell, or logoff and login again.

> **Other usages for the Text window:** The text window is handy also when you want to share your .blend files with the community or with your friends. A Text window can be used to write in a README text explaining the contents of your blender file. Much more handy that having it on a separate application. Be sure to keep it visible when saving!
>
> If you are sharing the file with the community and you want to share it under some licence you can write the licence in a text window.

A working Python example

Now that you've seen that Blender is extensible via Python scripting and that you've got the basics of script handling and how to run a script, and before smashing your brain with the full python API reference contained in next section let's have a look to a quick and dirty working example.

We will present a tiny script to produce polygons. This indeed duplicates somewhat the ADD>>Mesh>>Circle menu entry, but will create 'filled' polygons, not just the outline.

To make the script simple yet complete it will exhibit a Graphical User Interface (GUI) completely written via Blender's API.

Headers, importing modules and globals.

The first 32 lines of code are reported in Example 20-1.

Example 20-1. Script header

```
001 ###########################################################
002 #
003 # Demo Script for Blender Manual
004 #
005 ##########################################################S68
006 # This script generates polygons. It is quite useless
007 # since you can do polygons with ADD->Mesh->Circle
008 # but it is a nice complete script example, and the
009 # polygons are 'filled'
010 ###########################################################
011
012 ###########################################################
013 # Importing modules
014 ###########################################################
015
016 import Blender
017 from Blender import NMesh
018 from Blender.BGL import *
019 from Blender.Draw import *
020
```

```
021 import math
022 from math import *
023
024 # Polygon Parameters
025 T_NumberOfSides = Create(3)
026 T_Radius        = Create(1.0)
027
028 # Events
029 EVENT_NOEVENT = 1
030 EVENT_DRAW    = 2
031 EVENT_EXIT    = 3
032
```

After the necessary comments with the description of what the script does there is (lines 016-022) the importing of Python modules.

`Blender` is the main Blender Python API module. `NMesh` is the module providing access to Blender's meshes, while `BGL` and `Draw` give access to the OpenGL constants and functions and to Blender's windowing interface, respectively. The `math` module is Python's mathematical module.

The polygons are defined via the number of sides they have and their radius. These parameters have values which must be defined by the user via the GUI hence lines (025-026) creates two 'generic button' objects, with their default starting value.

Finally, the GUI objects works witha and generates events. Events identifier are integers left to the user to define. It is usually a good practice to define mnemonic names for events, as it is done here in lines (029-031).

Drawing the GUI.

The code responsible for drawing the code should reside in a `draw` function (Example 20-2).

Example 20-2. GUI drawing

```
033 ########################################################
034 # GUI drawing
035 ########################################################
036 def draw():
037  global T_NumberOfSides
038  global T_Radius
039  global EVENT_NOEVENT,EVENT_DRAW,EVENT_EXIT
040
041  ########## Titles
042  glClear(GL_COLOR_BUFFER_BIT)
043  glRasterPos2d(8, 103)
044  Text("Demo Polygon Script")
045
046  ######### Parameters GUI Buttons
047  glRasterPos2d(8, 83)
048  Text("Parameters:")
049  T_NumberOfSides = Number("No. of sides: ", EVENT_NOEVENT, 10, 55, 210, 18,
050                     T_NumberOfSides.val, 3, 20, "Number of sides of out polygon");
051  T_Radius        = Slider("Radius: ", EVENT_NOEVENT, 10, 35, 210, 18,
052                     T_Radius.val, 0.001, 20.0, 1, "Radius of the polygon");
053
054  ######### Draw and Exit Buttons
055  Button("Draw",EVENT_DRAW , 10, 10, 80, 18)
056  Button("Exit",EVENT_EXIT , 140, 10, 80, 18)
057
```

Lines (037-039) merely grant access to global data. The real interesting stuff starts from lines (042-044). The OpenGL window is initialized, and the current position set to x=8, y=103. The origin of this reference is the lower left corner of the script window. Then the title `Demo Polygon Script` is printed.

A further string is written (lines 047-048), then the input buttons for the parameters are created. The first (lines 049-050) is a NumButton, exactly alike those in the varouis Blender ButtonWindows. For the meaning of all the parameters pleas refer to the API reference. Basically ther eis the button label, the event generated by the button, its location (x,y) and its dimensions (width, height), its value, the minimum and maximum allowable values and a text string which willl appear as an help while hoovering on the button.

Lines (051-052) defines a slider, with a very similar sintax. Lines (055-056) finally creates a `Draw` butto which will create the polygon and an `Exit` button.

Managing Events.

The GUI is not drawn, and would not work, untill a proper event handler is written and registered (Example 20-3).

Example 20-3. Handling events

```
058 def event(evt, val):
059  if (evt == QKEY and not val):
060   Exit()
061
062 def bevent(evt):
063  global T_NumberOfSides
064  global T_Radius
065  global EVENT_NOEVENT,EVENT_DRAW,EVENT_EXIT
066
067  ######### Manages GUI events
068  if (evt == EVENT_EXIT):
069   Exit()
070  elif (evt== EVENT_DRAW):
071   Polygon(T_NumberOfSides.val, T_Radius.val)
072    Blender.Redraw()
073
074 Register(draw, event, bevent)
075
```

Lines (058-060) defines the keyboard event handler, here responding to the **QKEY** with a plain `Exit()` call.

More interesting are lines (062-072), in charge of managing the GUI events. Every time a GUI button is used this function is called, with the event number defined within the button as a parameter. The core of this function is ence a "select" structure executing different codes accordingly to the event number.

As a last call, the `Register` function is invoked. This effectively draws the GUI and starts the event capturing cycle.

Mesh handling

Finally, Example 20-4 shows the main function, the one creating the polygon. It is a rather simple mesh editing, but shows many important points of the Blender's internal data structure

Example 20-4. Script header

```
076 ##########################################################
077 # Main Body
078 ##########################################################
079 def Polygon(NumberOfSides,Radius):
080
081 ######### Creates a new mesh
082 poly = NMesh.GetRaw()
083
084 #########Populates it of vertices
085 for i in range(0,NumberOfSides):
086   phi = 3.141592653589 * 2 * i / NumberOfSides
087   x = Radius * cos(phi)
088   y = Radius * sin(phi)
089   z = 0
090
091   v = NMesh.Vert(x,y,z)
092   poly.verts.append(v)
093
094 #########Adds a new vertex to the center
095 v = NMesh.Vert(0.,0.,0.)
096 poly.verts.append(v)
097
098 #########Connects the vertices to form faces
099 for i in range(0,NumberOfSides):
100  f = NMesh.Face()
101  f.v.append(poly.verts[i])
102  f.v.append(poly.verts[(i+1)%NumberOfSides])
103  f.v.append(poly.verts[NumberOfSides])
104  poly.faces.append(f)
105
106 #########Creates a new Object with the new Mesh
107 polyObj = NMesh.PutRaw(poly)
108
109 Blender.Redraw()
```

The first important line here is number (082). Here a new mesh object, `poly` is created. The mesh object is constituted of a list of vertices and a list of faces, plus some other interesting stuff. For our purpouses the vertices and faces lists are what we need.

Of course the newely created mesh is empty. The first cycle (lines 085-092) computes the x,y,z location of the `NumberOfSides` vertices needed to define the polygon. Being a flat figure it is z=0 for all.

Line (091) call the `NMesh` method `Vert` to create a new vertex object of coords (x,y,z). Such an object is then appended (line 096) in the `poly` Mesh `verts` list.

Finally (lines 095-096) a last vertex is added in the center.

Lines (099-104) now connects these vertices to make faces. It is not required to create all vertices beforehand and then faces. You can safely create a new face as soon as all its vertices are there.

Line (100) creates a new face object. A face object has its own list of vertices `v` (up to 4) defining it. Lines (101-103) appends three vertices to the originally empty `f.v` list. The vertices are two subsequent vertices of the polygon and the central vertex. These

vertices must be taken from the Mesh `verts` list. finally line (104) appends the newly created face to the `faces` list of our `poly` mesh.

Conclusions

If you create a `polygon.py` file containing the above described code and load it into a Blender text window as you learned in the previous section and press **ALT P** in that window to run it you will see the script disappearing and the window turn grey. In the lower left corner the GUI will be drawn (Figure 20-2)

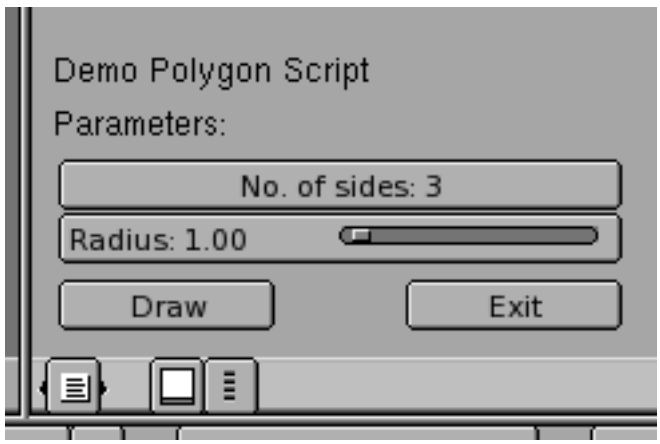


Figure 20-2. The GUI of our example.

By selecting, for example, 5 vertices and a radius 0.5, and by pressing the `Draw` button a pentagon will appear on the xy plane of the 3D window (Figure 20-3)

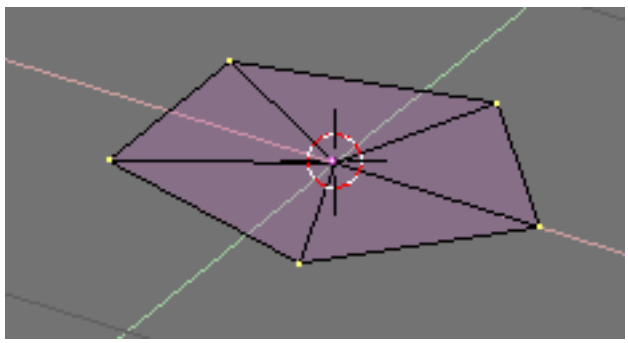


Figure 20-3. The result of our example script.

API Reference

This section reports an in-depth reference for Blender's Python API

Here it is :)[3]

Notes

1. http://www.python.org

2. http://www.blender.org
3. http://www.blender.org/modules/documentation/228PythonDoc/Blender-module.html

Chapter 21. TBW


PLUGINS!!!!

Chapter 22. TBW

NOTHING HERE (YET :))

Chapter 23. Interactive 3d

Introduction (-)

(to be written)

Designing for interactive environments (-)

(to be written)

Physics (-)

(to be written)

Logic Editing (-)

(to be written)

Sensors (-)

(to be written)

Always (-)

(to be written)

Keyboard (-)

(to be written)

Mouse (-)

(to be written)

Touch (-)

(to be written)

Collision (-)

(to be written)

Near (-)

(to be written)

Radar (-)

(to be written)

Property (-)

(to be written)

Random (-)

(to be written)

Ray (-)

(to be written)

Message (-)

(to be written)

Controllers (-)

(to be written)

And (-)

(to be written)

Or (-)

(to be written)

Expression (-)

(to be written)

Python (-)

(to be written)

Actuators (-)

(to be written)

Motion (-)

(to be written)

Constraint (-)

(to be written)

IPO (-)

(to be written)

Camera (-)

(to be written)

Sound (-)

(to be written)

Property (-)

(to be written)

Edit Object (-)

(to be written)

Scene (-)

(to be written)

Random (-)

(to be written)

Message (-)

(to be written)

CD (-)

(to be written)

Game (-)

(to be written)

Visibility (-)

(to be written)

Exporting to standalone applications (-)

(to be written)

Chapter 24. Usage of Blender 3D Plug-in

Introduction

The Blender 3D Plug-in allows you to publish interactive 3D Blender productions for web browsers on different computer platforms. On most platforms content will be handled by the Blender 3D Plug-in for Netscape. Besides being a plug-in for Netscape itself, the Netscape plug-in can also be used in Mozilla.

For Internet Explorer on Windows, we have created an ActiveX control. The Blender 3D Plug-in ActiveX control can also be used to publish content in other applications that support OLE/COM. Among those are Word, PowerPoint and Macromedia Director.

Functionality

The Blender 3D Plug-in is able to display two kinds of Blender files: regular Blender files and Publisher Blender files. Regular Blender files are created with the free Blender Creator and Publisher Blender files are created with Blender Publisher viable to those that own a Publisher license. When the plug-in displays a regular Blender file, the Blender logo's are displayed on top of the content. Owners of a Blender Publisher license can generate Publisher Blender files. This new file format supports compression for faster download, signing to signify file ownership and locking so that your content can not be altered. Another important advantage of Publisher files is that the plug-ins will not display the Blender logo's. In addition, a Publisher license enables you to create custom loading animations that replace the build-in loading animation

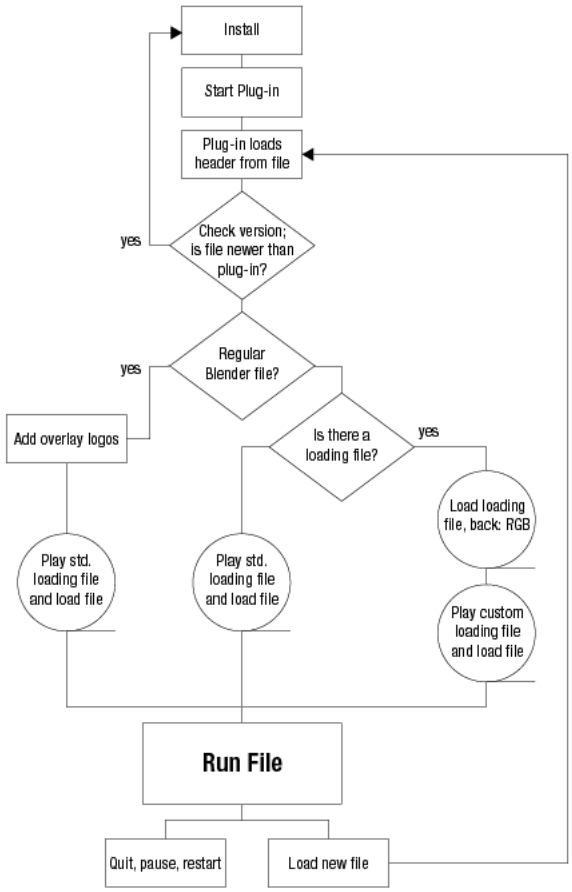


Figure 24-1. Blender 3D Plug-in functionality diagram

Figure 24-1 gives you an overview of how the plug-ins will process the different file types. When the plug-in is loaded, it determines whether a custom loading animation is requested. If so, it will commence downloading this file while displaying a solid color (if it is not already in the cache on the client system). The color can be specified in the HTML code or else, if missing from HTML, the background color of the HTML page is chosen. At download completion, the plug-in will download the main Blender file to be displayed while displaying the custom loading animation. The main Blender file must be a Publisher Blender file. If not, the plug-in will not play the file downloaded.

If a custom loading animation was not specified, the plug-in will download the main Blender file while displaying the build-in Blender loading animation. After completion, the plug-in displays the file downloaded with or without logo's depending on the file type (regular Blender or Publisher Blender file).

3D Plug-in installation

The installation procedure of the Blender 3D Plug-in depends on the type of plug-in and on the operating system. The Active X control can be installed automatically from the HTML page when the HTML code is properly written. For details, read the Section called *Embedding Blender 3D Plug-in in web pages* on how to embed the plug-ins in HTML. This installation process also includes automatic updates when a new plug-in becomes available.

Netscape

Downloading and installing the Netscape Blender 3D Plug-in is almost done automatically. If the plug-in is not available for your browser, you will be redirected to the Blender 3D Plug-in download page. There, you will find instructions on how to proceed.

In case there are problems when installing the plug-in, please read the FAQ section in the Section called *Blender 3D Plug-in FAQs*.

Creating content for the plug-ins

When creating content for the plug-in, there is not much difference with creating and running content inside Blender or as a stand-alone game. There is one difference however. The plug-in might have dimensions that do not correspond to the settings made in the Blender file. This might create a difference in aspect ratios. The plug-in will match the two as good as possible.

The next figures show the situation of a perfect match. In the 3D view of Figure 24-2, the outer dotted rectangle shows the area that is to be displayed. The size and shape of this rectangle is set by changing the value in the SizeX and SizeY buttons in the DisplayButtons (F10) of Blender. The size of the plug-in in Figure 24-3 has been set to the same values.

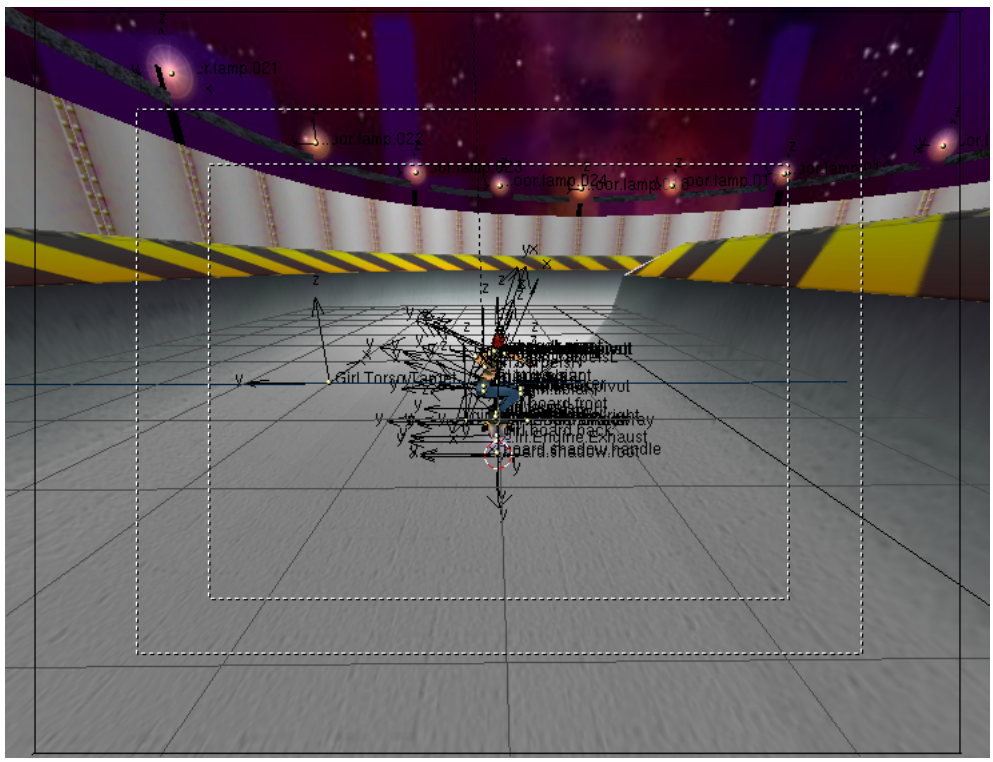


Figure 24-2. File in 3D view

Figure 24-3. Perfect match in the plug-in

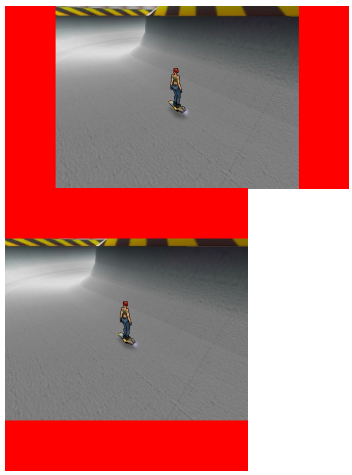


Figure 24-4. Framing of content

In the images in Figure 24-4, the aspect ratio of the plug-in does not match the Blender file. You'll notice that the plug-in or stand-alone player will try to show the content as big as possible without distortion of the content. The extra areas within the plug-in are drawn in a solid color (red in the figures) that can be either set in the HTML or in the Blender user interface.

How the plug-in or stand-alone player solves the difference in aspect ratio can be controlled in Blender. In the Display Buttons (F10) click on the "Game framing settings" button. You now can select three options: Stretch, Expose or Bars. If you select Bars the extra areas are filled with the color you set with the color sliders.

You can have different settings for different scenes. So you can have a 3D scene with bars and an overlay scene that is stretcht to fit to the output size.

If you select for a single 3D scene the settings as they are shown in Figure 24-5 you'll get results similar to the pictures in Figure 24-4.

Figure 24-5. GUI for Framing in the DisplayButtons (F10)

If you select "Expose", the plug-in or stand alone player will simply show more of the 3D environment. This will usually produce the most natural results but if you have enemies coming from the top or from the sides the user might see them pop up.

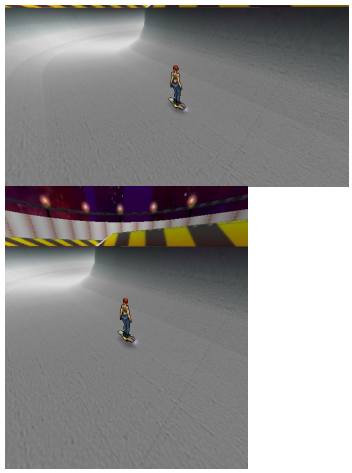


Figure 24-6. Extended framing

And finally, if you select "Stretch", the plug-in or stand-alone player will simply stretch the image horizontally or vertically to fit. This will distort the image somewhat (See Figure 24-7) but you'll never see bars or more from the 3D world as defined in Blender.

Figure 24-7. Stretched framing

Jumping to another HTML page

It is possible to have the browser load a new URL from within your Blender file. Send a message with the "To:" field set to "host_application" and the "Subject:" "load_url". The message's "Body:" should contain the full URL you want the browser to load.

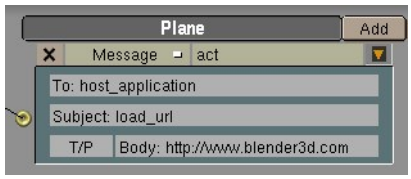


Figure 24-8. Browsing HTML pages from within the Plug-in

Creating a custom loading animation

The file `load.blend` (download at http://www.blender3d.com/plugin/blend/load.blend) is an example of a custom loading animation for the Blender 3D Web Plug-in. The selected object has all of the important logic for showing file loading progress. It has a property called "progress". The value of that property drives the Ipo animation curve of the object, causing its size to increase in the Z direction. This is the most convenient way to use the loading information, because it's easy to set up and preview an Ipo animation.

The only tricky part is to get the file loading progress information being sent by the plug-in. The plug-in sends game messages with the subject "progress" as the file loads. Each message has a body which is a floating point value between 0.0 and 1.0 sent as a text string. The value 0.0 means that none of the file is loaded yet. 1.0 means that the file is completely loaded. This information is extracted by the Python script "progress.py", which gets all "progress" messages from the message sensor (in case more than one message is received within a single cycle of the game logic). It evaluates the body of the last message, converting it back to a numerical value and assigns the value to the "progress" property of the object.

The camera has some logic attached which causes it to send artificial progress messages for testing the animation. This logic should be deleted from the camera before the file is actually put into use.

Tips:

1. The purpose of a loading animation is to occupy the viewer's attention while a larger file is loading. It should be as small as possible so that it loads very quickly. Textures, especially *.tga images, increase file size a lot. Use JPEG images or avoid images completely. Save your file using Blender's file compression tool.
2. Most of the complexity in this example is for showing the download progress of the larger file. Showing the loading progress is very reassuring to the viewer, but not absolutely necessary. You can use any real-time Blender scene as a loading animation.

Embedding Blender 3D Plug-in in web pages

To embed the Blender 3D Plug-in on your web pages, you will need to add some HTML code to your web pages. Also, you will want to add a link to the Blender 3D Plug-in page (http://www.blender3d.com/plugin/) where users can install the plug-in if it is not already installed on their system. Click-able images that you can use to forward users to the download page are available at the Blender 3D Plug-in page.

The current version (2.28) of the Active X control (the Internet Explorer plug-in) is now able to support multiple plug-ins on a one HTML page. The Netscape version does not support this however. Therefore, you are still advised not to put two plug-ins on a HTML page. This will be fixed in one of our forthcoming releases.

Insert the following HTML tag into your web page and change the parameters to suit your content:

```
<p>
<object
  classid="clsid:5DB05CB8-7751-469D-A1DD-45C8C201C013"
  id=Blender3DPlugin
  width = 640
  height = 480
  codebase="http://www.blender.nl/plugin/Blender3DPlugin.cab#Version=2,25,4,0">

<param name="blenderURL" value="http://www.yoursite.com/yourproduction.blend">
<param name="loadingURL" value="http://www.yoursite.com/yourloadinganimation.blend">
<param name="ForeColor" value=65280>
<param name="BackColor" value=255>
<param name="useFileBackColor" value=1>
<param name="frameRate" value=20>

<EMBED
  type="application/x-blender-plugin"
  PLUGINSPAGE="http://plugin.blender.nl"
  name="NPBlender"
  WIDTH=640
  HEIGHT=480
  SRC="http://www.yoursite.com/yourproduction.blend"
  loadingURL="http://www.yoursite.com/yourloadinganimation.blend"
  ForeColor=65280
  BackColor=255
  useFileBackColor=1
  frameRate=20>
</EMBED>
</object>
</p>
```

This code works for both the ActiveX control and the Netscape plug-ins.

The part between the `<object>` and the `<EMBED>` tags relates to the ActiveX control.

- The classid is the unique identifier of the Blender 3D Plug-in.
- The id is the name of the plug-in on the page. This can be used to identify the plug-in in Javascript.
- The width and height parameters allow you to set the dimensions of the plug-in on the page. In the example, width and height are given in pixels. You can also specify the dimensions in percentages of the size of the page (e.g. width = "50%").
- The codebase is the URL the ActiveX control will be downloaded from when it is not installed on the system the page is viewed on. The version number after the hash sign (#) should read the minimum version of the ActiveX control needed to view your content. Internet Explorer will compare this version to the version of the ActiveX control installed on the system. If the ActiveX control installed is older, the newer version is downloaded and installed automatically.

The `<object>` tag is followed by a list of parameters:

- blenderURL (required) is the URL of the Blender file to be viewed.
- loadingURL (optional) is the URL of the custom loading animation.
- ForeColor (optional) is the color to be shown while the custom loading animation is downloaded.
- BackColor (optional) is the color used to draw the extra areas when the aspect ratio of the plug-in does not match the aspect ratio set in the Blender file.

- useFileBackColor (optional) read the color to draw the extra areas with from the Blender file. If neither BackColor or useFileBackColor are set the HTML background color is used to draw the extra areas.
- frameRate (optional) is the maximum number of frames per second. When your animation does not need to be viewed at maximum frame rate possible (e.g. web banners), set this value to a meaningful maximum. With lower frame rates the client system will remain more responsive. For other types of content (e.g. games) you probably want to set this value to the maximum of 100 or leave out the parameter in which case the plug-in will use 100 as well.

The Netscape plug-in settings can be found between the `<EMBED>` and `</EMBED>` tags. Most of the parameters are the same as those of the ActiveX control.

Parameters that differ are:

- PLUGINSPAGE (optional) The URL of the web page displayed by the browser when the plug-in is not installed on the client system.
- name (required) The id is the name of the plug-in on the page. This can be used to identify the plug-in in Javascript. It is the equivalent of the id of he Active control.
- SRC (required) is the URL of the Blender file to be viewed. The equivalent of the ActiveX blenderURL parameter.

Color values should be passed to the ActiveX control in a format known as OLE_COLOR. The red, green and blue components of the color are stored in a single value. To determine a BGR value, specify blue, green and red (each of which has a value from 0 - 255) in the following formula: BGR value = (blue * 65536) + (green * 256) + red

Some parameters of the plug-ins can be dynamically accessed (through Javascript for example). This means that you can interact with the plug-in from your HTML code. For instance, you can change the URL of the Blender file from the HTML page with a button by adding the following to your HTML page:

```
                    <form>
<input type="button"
value="load other production"
onClick="Blender3DPlugin.blenderURL='http://www.yoursite.com/other.blend';">
</form>
```

This feature is currently not available for the Netscape plug-ins. We are planning to add Javascript capabilities in the next release.

Another very powerful option is to send messages to the Blender file running inside the plug-in. This way you can change the behaviour of your production from the HTML. The following form allows you to type in messages in the HTML and send them to the plug-in:

```
<p>
<form onSubmit="Blender3DPlugin.SendMessage(
     the_to.value,
     the_from.value,
     the_subject.value,
     the_body.value); return false;">

<table><tr>

<td>to:</td>
<td><input type="text" name="the_to" value="" size=30></td></tr>
```

```
<tr><td>from:</td>
<td><input type="text" name="the_from" value="" size=30></td></tr>

<tr><td>subject:</td>
<td><input type="text" name="the_subject" value="" size=30></td></tr>

<tr><td>body:</td>
<td><input type="text" name="the_body" value="" size=30></td></tr>

<tr><td> </td>
<td>
<input type="button" value="send"
     onClick="Blender3DPlugin.SendMessage(the_to.value,
     the_from.value, the_subject.value, the_body.value);">
</td></tr>

</table>
</form>
</p>
```

Embedding the ActiveX control in other applications

If Blender content is played in Internet Explorer, it is played using the Blender 3D Plug-in Active X Control. Active X is a Microsoft technology that allows Active X controls to run inside a host application. Internet Explorer is one example of such a host but there are many more. Another application that can use Active X controls is the popular PowerPoint application. For detailed information on Active X controls see Microsoft's Active X pages[3].

When a control is needed for a certain type of playback, it is automatically downloaded from a secure location, the control installs itself, and then becomes available in Windows. The advantage of Active X controls is that although they are generally downloaded and used in Internet Explorer, once they exist on your system any Windows application that understands Active X can use the controls to extend their applications.

Embedding Blender content in PowerPoint

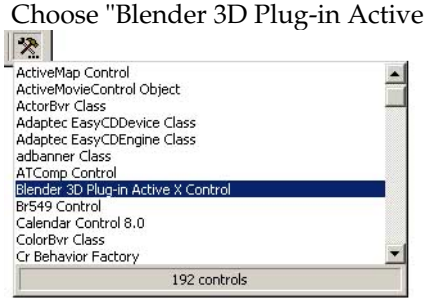
As example how to embed Blender content into other applications using the Active X control, we use here Microsofts Powerpoint.

Steps to embed Blender content in PowerPoint 2000:

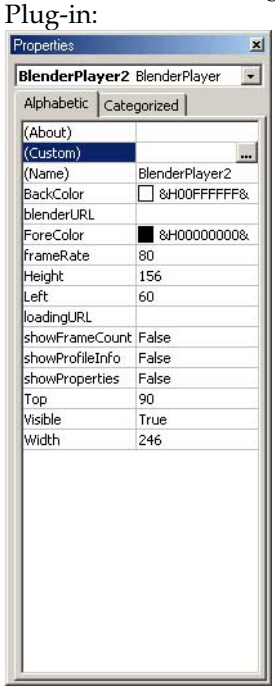
1. Start PowerPoint 2000
2. Open the Active X Control Toolbox by choosing View > Toolbars > Control Toolbox.
3. The item in the bottom right corner is the "More Controls" button.

Click on the button to have a list of Active X Controls appear that are installed on the machine.

4. Choose "Blender 3D Plug-in Active X Control" from the list:

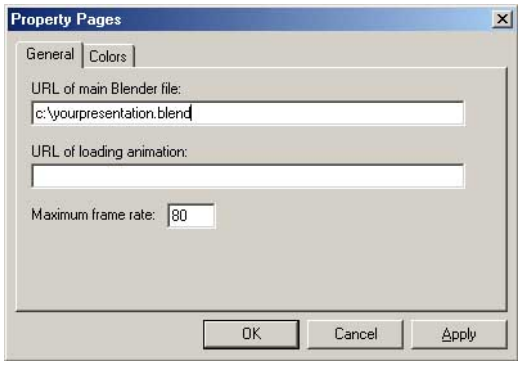


5. You will notice that the cursor changes to a crosshair. This indicates that you can draw a rectangle to define the location and dimensions of the control in your presentation.
6. Press the left mouse button at the desired location and drag the mouse (while keeping the button pressed) to define its size.
7. After you have created the control, right click on the control you have just created. This will bring up a window listing the properties of the Blender 3D Plug-in:



8. You can choose to edit the properties in this window or you can choose to open a separate property window by selecting custom and clicking on the button

with the three dots ("..."):



9. Enter the URL of the presentation or a direct path as shown in the image. If you have created a custom loading animation (Blender Publisher license owners only) enter the URL or file path of that presentation as well. You can also enter a maximum frame rate. By reducing the frame rate, you can reduce the load the plug-in puts on the system.

Blender 3D Plug-in FAQs

Blender 3D Plug-in FAQs

1. All plug-ins:

Q: The plug-in has loaded, but it doesn't respond to the keyboard or mouse. What's wrong?

A: You have to click on the plug-in to give it the keyboard/mouse focus.

Q: Playback is very slow, what can I do?

A: Like all other 3D applications, you need a 3D accelerated videocard to get good performance. Also make sure you have installed the latest drivers for your card. In addition, make sure that the HTML code contains no frameRate parameter or that is is set to a high value (see the Section called *Embedding Blender 3D Plug-in in web pages*).

2. ActiveX specific:

Q: Installation on the client system fails, what can I do?

A: When the ActiveX control refuses to install it could be that Internet Explorer on the client system has been configured with strict security settings that prevent ActiveX controls being downloaded from the Internet. The security settings of Internet Explorer can be accessed in the Internet Options available from the Tools menu. If security settings are not the cause, check the windows version.

- On Windows 2000, the user type determines whether she or he is allowed to install ActiveX controls downloaded from the Internet. Depending on the Windows 2000 configuration, only Administrator type users are generally allowed to install the ActiveX control.

- On Windows98 occasionally, the plug-in refuses to install. Often, the client system has outdated or corrupt DCOM drivers. This can be solved by downloading the updated DCOM98 drivers for Windows 98 (version 1.3) from http://www.microsoft.com/com/dcom/dcom98/download.asp.

3. Netscape specific:

Q: What are the known problems of the Netscape plug-in?

A: There are some problems which are currently under investigation and will be fixed soon:

- Netscape/Linux window resize. Resizing the Netscape window can crash the plug-in under Linux.
- Netscape/Linux sound disabled. The Netscape/Linux version of the plug-in does not have sound support in this release.
- Netscape/Linux: 'exception in thread "main"'. This is a known problem; we are working on it. Just click on ok and everything will run fine.
- Netscape/Linux: no OpenGL software rendering. If you do not have a 3D accelerated X-server the plug-in may fail.

Notes

1. http://www.blender3d.com/plugin/blend/load.blend
2. http://www.blender3d.com/plugin/
3. http://www.microsoft.com/com/tech/activex.asp
4. http://www.microsoft.com/com/dcom/dcom98/download.asp

Chapter 25. Python Scripting for interactive environments (-)

(game engine specific Python subjects)

(to be written)

Chapter 26. Blender windows - general introduction

This section describes the general functions of the mouse and keyboard, both of which work uniformly throughout the Blender interface. Each Blender window also offers a number of specific options. These options are described in the following sections.

The Mouse

Each time you place the mouse cursor over the edge of a Blender window, the mouse cursor changes shape. When this happens, the following mouse keys are activated:

LMB (hold-move)

Drag the window edge horizontally or vertically. The window edge always moves in increments of 4 pixels, making it relatively easy to move two window edges so that they are precisely adjacent to each other, thus joining them.

MMB or RMB

A PopupMenu prompts for "Split Area" or "Join Areas".

Choosing "Split Area", Blender allows you to indicate the exact split point or to cancel "split" by pressing ESC. "Split" divides the window into two windows, creating an exact copy of the original window.

Choosing "Join Areas", Windows with a shared edge are joined if possible. The active Window remains.

If there is no Header in the Window, the PopupMenu contains also the item "Add Header".

The Window Header

Blender window headers offer the following extra options in combination with mouse keys:

LMB on the header

The entire Blender window pops to the foreground.

CTRL LMB on the header

The entire Blender window pops to the background.

MMB (hold-move) on the header

If the Blender window is not wide enough to display the entire header, the **MMB** key can be used to horizontally shift the header.

RMB on the header

A PopupMenu appears asking for "Top", "Bottom" or "No Header". That way the header can be moved to the top or the bottom of the Blender window or can be hidden.

You can add a header to a Window by pressing the middle Mouse over an edge of a headerless Window.

Chapter 27. HotKeys In-depth Reference

Window HotKeys

Certain window managers also use the following hotkeys. So **ALT-CTRL** can be substituted for **CTRL** to perform the functions described below if a conflict arises.

CTRL+LEFTARROW

Go to the previous Screen.

CTRL+RIGHTARROW

Go to the next Screen.

CTRL+UPARROW or CTRL+DOWNARROW

Maximise the window or return to the previous window display size.

SHIFT+F4

Change the window to a DataView

SHIFT+F5

Change the window to a 3DWindow

SHIFT+F6

Change the window to an IpoWindow

SHIFT+F7

Change the window to a ButtonsWindow

SHIFT+F8

Change the window to a SequenceWindow

SHIFT+F9

Change the window to an OopsWindow

SHIFT+F10

Change the window to an ImageWindow

SHIFT+F11

Change the window to a TextWindow

SHIFT+F12

Change the window to a SoundWindow

Universal HotKeys

The following HotKeys work uniformly in all Blender windows:

ESC

- This key always cancels Blender functions without changes.
- *or*: FileWindow, DataView and ImageSelect: back to the previous window type.
- *or*: the RenderWindow is *pushed* to the background (or closed, that depends on the operating system).

SPACE

Open the main menu of Blender, the Toolbox.

TAB>

Start or quit EditMode.

F1

Loads a Blender file. Changes the window to a FileWindow.

SHIFT+F1

Appends parts from other files, or loads as Library-data. Changes the window to a FileWindow, making Blender files accessible as a directory.

F2

Writes a Blender file. Change the window to a FileWindow.

SHIFT+F2

Exports the scene as a DXF file

CTRL+F2

Exports the scene as a VRML1 file

F3KEY

Writes a picture (if a picture has been rendered). The fileformat is as indicated in the DisplayButtons. The window becomes a FileWindow.

CTRL+F3

Saves a screendump of the active window. The fileformat is as indicated in the DisplayButtons. The window becomes a FileWindow.

SHIFT+CTRL+F3

Saves a screendump of the whole Blender screen. The fileformat is as indicated in the DisplayButtons. The window becomes a FileWindow.

F4

Displays the LampButtons (if a ButtonsWindow is available).

F5

Displays the MaterialButtons (if a ButtonsWindow is available).

F6

Displays the TextureButtons (if a ButtonsWindow is available).

F7

Displays the AnimButtons (if a ButtonsWindow is available).

F8

Displays the RealtimeButtons (if a ButtonsWindow is available).

F9

Displays the EditButtons (if a ButtonsWindow is available).

F10

Displays the DisplayButtons (if a ButtonsWindow is available).

F11

Hides or shows the render window.

F12

Starts the rendering of the active camera.

LEFTARROW

Go to the previous frame.

SHIFT+LEFTARROW

Go to the first frame.

RIGHTARROW

Go to the next frame.

SHIFT+LEFTARROW

Go to the last frame.

UPARROW

Go forward 10 frames.

DOWNARROW

Go back 10 frames.

ALT+AKEY

Change the current Blender window to Animation Playback mode. The cursor changes to a counter.

ALT-SHIFT+AKEY

The current window, *plus* all 3DWindows go into Animation Playback mode.

ALT+EKEY

Start or leave EditMode.

IKEY

Insert Key menu. This menu differs from window to window.

JKEY

Toggle the render buffers. Blender allows you to retain two different rendered pictures in memory.

NKEY

Number buttons. These buttons differ depending on the type of Blender window. Numeric information for the active selection can be visualised and specified using these buttons.

CTRL+OKEY

Opens the last saved file.

QKEY

"OK? Quit Blender". This key closes Blender. "Blender quit" is displayed in the console if Blender is properly closed.

ALT-CTRL+TKEY

TimerMenu. This menu offers access to information about drawing speed. The results are displayed in the console.

CTRL+UKEY

"OK" Save User defaults". The current project (windows, objects, etc.), including UserMenu settings are written to the default file that will be loaded every time you start Blender or set it to defaults by pressing **CTRL-X**.

CTRL+WKEY

Write file. This key combination allows you to write the Blender file without opening a FileWindow.

ALT+WKEY

Write Videoscape file. Changes the window to a FileWindow.

CTRL+XKEY

Erase All. Everything (except the render buffer) is erased and released. The default scene is reloaded.

EditMode HotKeys - General

TAB or ALT+E

This button starts and stops EditMode.

AKEY

Select/Unselect all.

BKEY

Border Select. In EditMode, this function only works on the vertices. It works as described in the previous section.

BKEY-BKEY

Circle Select. If you press BKEY a second time after starting Border Select, Circle Select is invoked. This mode selects vertices with **LMB** and deselects vertices with **MMB**. Use **NUM+** or **NUM-** to adjust the circle size. Leave Circle Select with **RMB** or ESC.

NKEY

NumberMenu. In EditMode Mesh, Curve, Surface: The location of the active vertex is displayed.

PKEY

SeParate. All selected vertices, edges, faces and curves are removed from the Edit-Mode and placed in a new Object. This operation is the opposite of Join (CTRL+J).

CTRL+PKEY

"Make Vertex Parent". Select 1 or 3 vertices from a Mesh, Curve or Surface. Now this Object becomes the Vertex Parent of the selected Objects. If only 1 vertex is selected, only the *location* of this vertex determines the Parent transformation; the rotation and dimensions of the Parent do not play a role here. If three vertices are selected, it is a 'normal' Parent relationship in which the 3 vertices determine the rotation and location of the Child *together*. This method produces interesting effects with Vertex Keys. In EditMode, other Objects can be selected with **CTRL+RMB**.

CTRL+SKEY

Shear. In EditMode this operation enables you to make selected forms 'slant'. This always works via the horizontal screen axis.

UKEY

(For Font Objects: ALT+U) Reload Original Data. When starting EditMode, the original ObData block is saved. This option enables you to restore the previous situation. By continually leaving EditMode while working (TAB-TAB) you can refresh this 'undo buffer'.

WKEY

Specials PopupMenu. A number of *tools* are included in this PopupMenu as an alternative to the EditButtons. This makes the buttons accessible as *shortcuts*, e.g. EditButtons->Subdivide is also 'WKEY, 1KEY'.

SHIFT-DKEY

Add Duplicate. The selected vertices (faces, curves, etc.) are copied. Grab mode starts immediately thereafter.

SHIFT+WKEY

Warp. Selected vertices can be bent into curves with this option. It can be used to convert a plane into a tube or even a sphere. The centre of the circle is the 3DCursor. The mid-line of the circle is determined by the horizontal dimensions of the selected vertices. When you start, everything is already bent 90 degrees. Moving the mouse up or down increases or decreases the extent to which *warping* is done. By zooming in/out of the 3Dwindow, you can specify the maximum degree of *warping*. The CTRL limitor increments warping in steps of 5 degrees.

EditMode Mesh Hotkeys

EKEY

Extrude Selected. "Extrude" in EditMode transforms all the selected *edges* to *faces*. If possible, the selected faces are also duplicated. Grab mode is started directly after this command is executed.

FKEY

Make Edge/Face. If 2 vertices are selected, an *edge* is created. If 3 or 4 vertices are selected, a *face* is created.

SHIFT+FKEY

Fill selected. All selected vertices that are bound by *edges* and form a closed polygon are filled with triangular *faces*. Holes are automatically taken into account. This operation is 2D; various layers of polygons must be filled in succession.

ALT+F

Beauty Fill. The edges of all the selected triangular faces are switched in such a way that equally sized faces are formed. This operation is 2D; various layers of polygons must be filled in succession. The Beauty Fill can be performed immediately after a Fill.

HKEY

Hide Selected. All selected vertices and faces are temporarily hidden.

SHIFT+H

Hide Not Selected: All *non*-selected vertices and faces are temporarily hidden.

ALT+H

Reveal. All temporarily hidden vertices and faces are drawn again.

LKEY

Select Linked. If you start with an *unselected* vertex near the mouse cursor, this vertex is selected, together with all vertices that share an edge with it.

SHIFT+L

Deselect Linked. If you start with a *selected* vertex, this vertex is deselected, together with all vertices that share an edge with it.

CTRL+L

Select Linked Selected. Starting with *all* selected vertices, all vertices connected to them are selected too.

CTRL+N

Calculate Normals Outside. All normals from selected faces are recalcultated and consistently set in the same direction. An attempt is made to direct all normals 'outward'.

SHIFT-CTRL+N

Calculate Normals Inside. All normals from selected faces are recalculated and consistently set in the same direction. An attempt is made to direct all normals 'inward'.

CTRL+T

Make Triangles. All selected faces are converted to triangles.

XKEY

Erase Selected. A PopupMenu offers the following options:

- "Vertices": all vertices are deleted. This includes the edges and faces they form.
- "Edges": all edges with both vertices selected are deleted. If this 'releases' certain vertices, they are deleted as well. Faces that can no longer exist as a result of this action are also deleted.
- "Faces": all faces with all their vertices selected are deleted. If any vertices are 'released' as a result of this action, they are deleted.
- "All": everything is deleted.
- "Edges and Faces": all selected edges and faces are deleted, but the vertices remain.
- "Only Faces": all selected faces are deleted, but the edges and vertices remain.

YKEY

Split. This command 'splits' the selected part of a Mesh without deleting faces. The split parts are no longer bound by *edges*. Use this command to control *smoothing*. Since the split parts have vertices at the same position, selection with LKEY is recommended.

EditMode Curve Hotkeys

CKEY

Set the selected curves to cyclic or turn cyclic off. An individual curve is selected if at least one of the vertices is selected.

EKEY

Extrude Curve. A vertex is added to the selected end of the curves. Grab mode is started immediately after this command is executed.

FKEY

Add segment. A segment is added between two selected vertices at the end of two curves. These two curves are combined into 1 curve.

HKEY

Toggle Handle *align/free*. Toggles the selected Bezier *handles* between *free* or *aligned*.

SHIFT+H

Set Handle *auto*. The selected Bezier *handles* are converted to *auto* type.

CTRL+H

Calculate Handles. The selected Bezier curves are calculated and all *handles* are assigned a type.

LKEY

Select Linked. If you start with an *non*-selected vertex near the mouse cursor, this vertex is selected together with all the vertices of the same curve.

SHIFT+L

Deselect Linked. If you start with a *selected* vertex, it is deselected together with all the vertices of the same curve.

TKEY

Tilt mode. Specify an extra axis rotation, i.e. the *tilt*, for each vertex in a 3D curve.

ALT+TKEY

Clear Tilt. Set all axis rotations of the selected vertices to zero.

VKEY

Vector Handle. The selected Bezier *handles* are converted to *vector* type.

WKEY

The special menu for curves appears:

- Subdivide. Subdivide the selected vertices
- Switch direction. The direction of the selected curves is reversed. This is manly for Curves that are used as *paths*!

XKEY

Erase Selected. A PopupMenu offers the following options:

- "Selected": all selected vertices are deleted.
- "Segment": a curve segment is deleted. This only works for single segments. Curves can be split in two using this option. Or use this option to specify the cyclic position within a cyclic curve.
- "All": delete everything.

EditMode Font Hotkeys

SHIFT+TAB

The ASCII code for TAB.

RIGHTARROWKEY

Move text cursor 1 position forward

SHIFT+RIGHTARROWKEY

Move text cursor to the end of the line.

LEFTARROWKEY

Move text cursor 1 position backwards.

SHIFT+LEFTARROWKEY

Move text cursor to the start of the line

DOWNARROWKEY

Move text cursor 1 line forward

SHIFT+DOWNARROWKEY

Move text cursor to the end of the text.

UPARROWKEY

Move text cursor 1 line back.

SHIFT+UPARROWKEY

Move text cursor to the beginning of the text

ALT+U

"Reload Original Data" (undo). When EditMode is started, the original text is saved. You can restore this original text with this option.

ALT+V

Paste text. The text file "/tmp/.cutbuffer" is inserted at the cursor location.

EditMode Surface Hotkeys

CKEY

Toggle Cyclic menu. A PopupMenu asks if selected surfaces in the 'U' or the 'V' direction must be cyclic. If they were already cyclic, this mode is turned off.

EKEY

Extrude Selected. This makes *surfaces* of all the selected *curves*, if possible. Only the edges of surfaces or loose curves are candidates for this operation. Grab mode is started immediately after this command is completed.

FKEY

Add segment. A segment is added between two selected vertices at the ends of two curves. These two curves are combined into 1 curve.

LKEY

Select Linked. If you start with an *non*-selected vertex near the mouse cursor, this vertex is selected together with all the vertices of the same curve or surface.

SHIFT+L

Deselect Linked. If you start with a *selected* vertex, this vertex is deselected together with all vertices of the same curve or surface.

SHIFT+R

Select Row. Starting with the last selected vertex, a complete row of vertices is selected in the 'U' or 'V' direction. Selecting "Select Row" a second time with the same vertex switches the 'U' or 'V' selection.

WKEY

The special menu for curves appears:

- Subdivide. Subdivide the selected vertices
- Switch direction. This will switch the normals of the selected parts.

XKEY

Erase Selected. A PopupMenu offers the following choices:

- "Selected": all selected vertices are deleted.
- "All": delete everything.

Armature Hotkeys

TABKEY

????

EKEY

Extrude Armature. An extra Bone is added to the selected Bone. Grab mode is started immediately thereafter.

CTRL+K

????

CTRL+P

Make Parent. If the Parent is an Armature, a PopupMenu offers three options:

- "Use Bone": One of the Bones becomes the parent. The Object will not be deformed. A popup permits to select the bone. This is the option if you are modeling a robot or machinery
- "Use Armature": The whole armature is used as parent for deformations. THis is the choiche for organic beings.
- "Use Object": Standard parenting.

VertexPaint Hotkeys

SHIFT+K

All vertex colours are erased; they are changed to the current drawing colour.

UKEY

Undo. This undo is 'real'. Pressing Undo twice returns you to the previous situation.

WKEY

"Shared Vertexcol": The colours of all faces that share vertices are blended.

FaceSelect Hotkeys

TAB

Switches to EditMode, selections made here will show up when switching back to FaceSelectMode with **TAB**.

AKEY

Selects all faces.

BKEY

Border select.

RKEY

Calls a menu allowing to rotate the UV coordinates or the VertexCol.

UKEY

Calls the "UV Calculation" menu. The following modes can the applied to the selected faces:

- `Cube :` Cubical mapping, a number button asks for the cubemap size
- `Cylinder :` Cylindrical mapping, calculated from the center of the selected faces
- `Sphere :` Spherical mapping, calculated from the center of the selected faces
- `Bounds to x:` UV coordinates are calculated from the actual view, then scaled to a boundbox of 64 or 128 pixels in square
- `Standard x:` Each face gets default square UV coordinates

- `From Window :` The UV coordinates are calculated using the projection as displayed in the 3DWindow

Chapter 28. Windows Reference

The InfoWindow

InfoToolbar

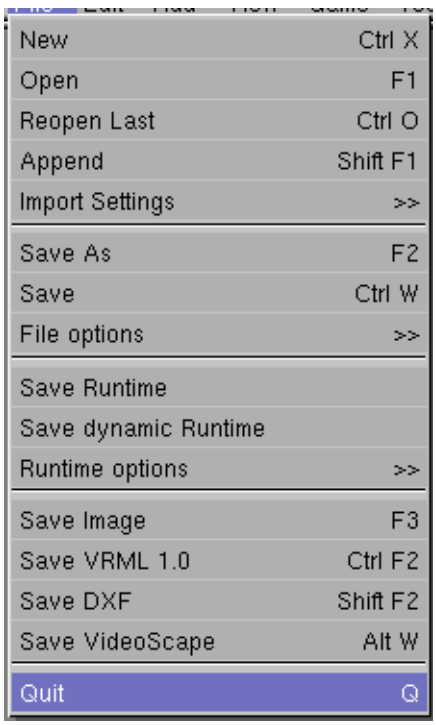
WindowType

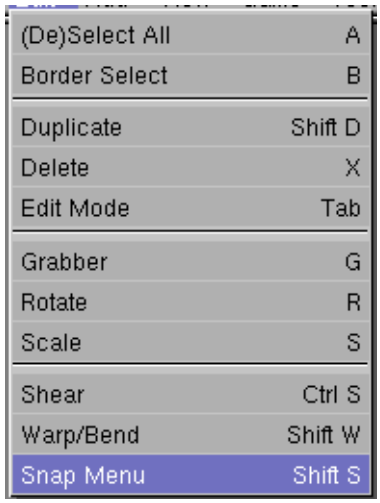
As in all Blender window headers, the first button allows you to configure the window type.

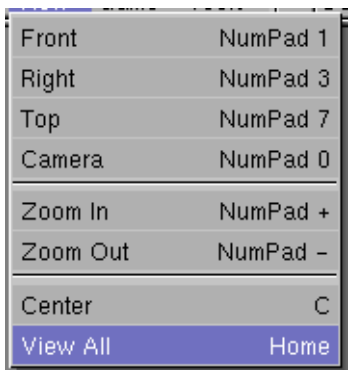
Menus

The  hide/shows the menus.

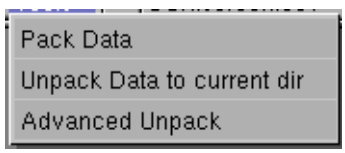
With the menus you can easily access common commands, like saving and loading.







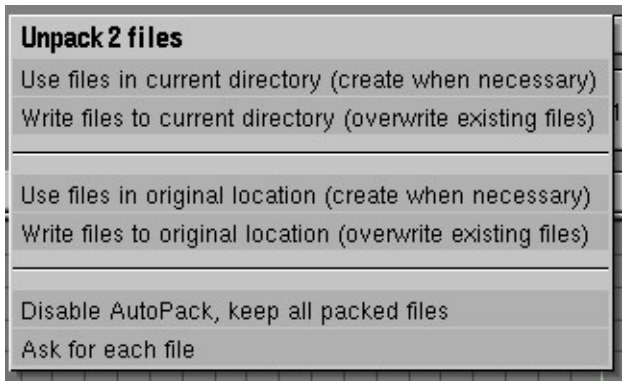
The menu entries are very common and doing the things they are labeled. A special thing is the Tools menu:



With *Pack Data* you can pack Images, Fonts and Sounds into the Blend-file, allowing to distribute easily your files as a single file. As a sign that your file is packed, a little icon of a parcel appears in the menu-bar.

Unpack Data into current dir unpacks into the current directory on your harddisk. It creates directories for textures, sounds and fonts there.

The *Advanced Unpack* menu gives you more control to unpack files. The entries are self-explaining.



Screen Browse

Allows you to select a different Screen from a list. The option "Add New" creates an exact copy of the current Screen. The copy is 'invisble': only the name on the adjacent button changes. HotKey for next or previous Screen: ALT+ARROWLEFT or ALT+ARROWRIGHT

SCR:

Assign a new and unique name to the current Screen and then adds the new name to the list in alphabetical order.

Delete Screen (But)

`Delete Current Screen?` The current Screen is deleted and released.

Scene Browse

Select a different scene from a list. This button is blocked in EditMode.

`Add New` displays a PopupMenu with four options:

- "Empty": create a completely empty scene.
- "Link Objects": all Objects are linked to the new scene. The *layer* and *selection flags* of the Objects can be configured differently for each Scene.
- "Link ObData": duplicates Objects only. ObData linked to the Objects, e.g. Mesh and Curve, are not duplicated.
- "Full Copy": everything is duplicated.

SCE:

Assigns a new and unique name to the current Scene and places the new name in the list in alphabetical order.

Delete Scene (But)

"OK? Delete Current Scene". This button deletes and releases the current Scene without deleting Objects. Objects without users are not written to a file.

The information text

The standard text is:

- www.blender.org: the location at which the software can be obtained.
- 2.28: the version of Blender. This manual is applicable to the V2.x series only.
- Ve: 4 Fa: 1: the number of *vertices* and *faces* in the current 3DWindow. If in doubt, use NUMPAD-9 to count the *vertices* and *faces* again.
- Ob: 2-1: the number of *total* Objects and of *selected* Objects in the current 3DWindow.
- La: 0: the number of lamps in the current 3Dwindow.
- Mem: 2.03M: the amount of memory in use in Megabytes, not including fragmented memory.
- Plane (or similar): the name of the *active* Object.

Changes in EditMode:

- Ve: 0-4 Fa: 0-1: The first numeric value is the number of *selected* vertices, the second is the *total* number of vertices. The numeric values for the second variable apply to *faces* and have the same significance.
- Plane (or similar): the name of the *active* Object.

During and after rendering:

The values for the totals are changed to reflect the rendered picture. These values may be different from the totals displayed in the 3DWindow.

- Time: 00.01.56 (00:44): the pure rendering time for the last picture rendered in minutes/seconds/hundredths of a second and the actual extra rendering time in seconds/hundredths of a second. Excessive swap time or poor file accessibility can cause high 'extra rendering time'.

The Maximize button

Clicking the Maximize button will switch blenter in/out of full screen mode.

The Toolbox button

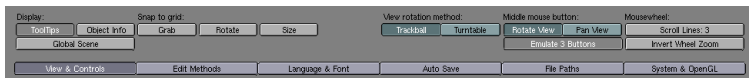
Clicking the Toolbox icon will pop up the Toolbox, like **SPACE** does.

The Info Buttons

The Info Window allows you to configure personal settings. These settings are automatically loaded from the file $HOME/.B.blend each time Blender is started. Personal settings cannot be written to a file other than .B.blend. The HotKey CTRL-U can be used to overwrite the file .B.blend.

The bottom line of buttons selects the various settings:

View and Controls:



- `ToolTips` - Switch Tooltips on and off.
- `ObjectInfo` - Shows Object name and frame in 3D viewport
- `Grab,Rotate,Scale` - Makes relevant action take place in steps, as if **CTRL** were pressed.
- `Clobal Scene` - Forces the current scene to be displayed in all windows
- `Trackball, Turnable` - Sets the behaviour of 3D window rotation method
- `Rotate View, Pan View`- Sets **MMB** functionality.
- `Emulate 3D buttons` - Makes **ALT LMB** equivalent to **MMB**
- `Scroll Lines:` - Controls wheel mouse scrolling
- `Invert Wheel Zoom` - toggles the Wheel directio to zoom in-zoom out

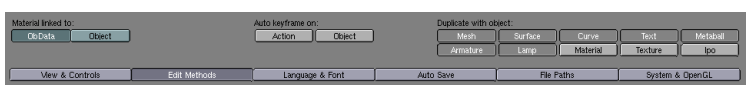
By default, the following *limitors* apply to *grabbing rotating* and *scaling* (press the keys after click and hold with the mouse):

- (no key): fluid change
- **SHIFT**: finer control
- **CTRL**: large grid steps
- **SHIFT-CTRL**: small grid steps

The following alternatives can also be used (when the corresponding toggle button is ON):

- (no key): large grid steps
- **SHIFT**: small grid steps
- **CTRL**: fluid change
- **SHIFT-CTRL**: finer control

Edit Methods:



- `ObData, Object` - Toggles where Material data is linked to.
- `Action, Object` - ????
- `Duplicate with object:` - This series of button specifies which data is really duplicated, and which is merely linked, when a object is duplicated (**SHIFT D**

One of Blender's most advanced features is its use of object-oriented functions. Blender allows you to reuse (i.e. link) data blocks to construct compact and efficient structures.

When one of these buttons is pressed, the indicated DataBlock is duplicated instead of linked when using **SHIFT-D**. The **ALT-D** command always makes a copy of the selected Objects with all other data linked.

The most commonly use of *links* is when activating the *Duplicate* commands: **ALT+D** create a copy of the selected Objects, reusing (i.e. linking to) all other data, including Meshes and Materials. **SHIFT+D** create a copy of the selected Objects, using these button settings to determine whether links to other data are created or duplicates of other data are created.

Language and Fonts:

- `International Fonts` - Toggles the usage of international, antialiased, fonts.
- `Select font` - Allows for font customization
- `Font size:` - Sets UI font size.
- `Language:` - Sets the language.
- `Tooltips, Buttons, Toolbox` - Sets which part of the UI is to be translated.

Auto Save:

Blender can save 'temp' files at regular intervals as a temporary backup or as extra protection against disasters. The files are identical to Blender files saved in the normal manner. If Blender is in EditMode when this function is used, only the original Data are saved, without saving the Data with which you are working. Blender saves 'temp' files in the specified 'temp' directory with the name "<process-id>.blend". This results in unique names for all 'temp' files, allowing multiple Blenders to simultaneously write 'temp' files on the same computer. When Blender is closed down, the file is renamed "quit.blend", making it easy to retrieve work in progress if the user inadvertently quits Blender. Blender writes files very quickly, which means that waiting time is kept to a minimum, allowing the user to continue working within a split second after saving files of 1-2 Mb.

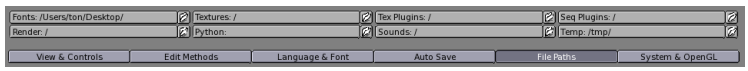
- `Auto Save Temp Files` - Enable saving temporary .blend files.
- `Minutes:` - Time in minutes between auto saving

- `Open recent` - Opens last saved temporary file
- `Versions` - Sets how many .blend# files are to be kept as history every time a regular save is performed.

  If 'Versions' has a value of '2' and the file 'rt.blend' is being written:

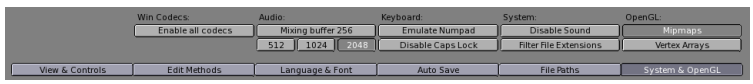
  - rt.blend2 is deleted
  - rt.blend1 is renamed to rt.blend2
  - rt.blend is renamed to rt.blend1
  - rt.blend is rewritten.

File Paths:



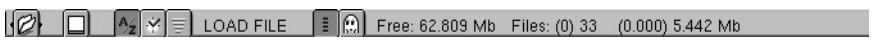
Defines all the directory where items are to be looked for/ saved by default, their name is self-explanatory.

System and OpenGL:



- `Enable all Codecs` - Enables all available codecs for rendering.
- `Mxing buffer:` - Sets the dimension of the Audio mixing buffer
- `Emulate Numpad` - Forces regular 0 to 9 keys to act as NumPad 0-9 keys (if you are on a laptop...)
- `Disable Caps Lock` - Disables caps lock when entering Text.
- `Disable Sound` - Disables sound from being played.
- `File Filter Extensions` - Makes only file with extension to appear in image select windows
- `Mipmaps` - Turns OpenGL MipMapping ON/OFF
- `VertexArrays` - Turns OpenGL vertex arrays to ON/OFF

The FileWindow



FileToolbar

WindowType

As with every window header, the first button left enables you to set the window type.

Full Window

The second button expands the window to full screen or returns to the previous window display size; returns to the original screen setting (**CTRL-UPARROW**).

Sort Alpha

Files are sorted alphabetically. Directories are always sorted first.

Sort Time

Files are sorted by creation date.

Sort Size

Files are sorted by size.

Action type

Indicates what type of FileWindow this is. It is extremely important to continually verify that the correct `LOAD` or `SAVE` option is selected.

File list format

Indicates whether the file names are displayed in long or short format.

Hide dot-files

The ghost button hides dot-files (filenames with a leading dot).

Info

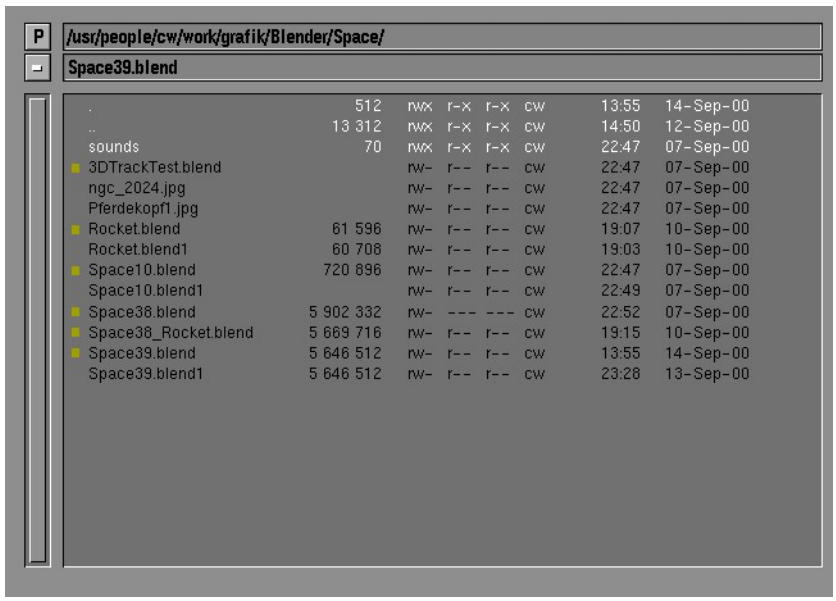
The header provides extra information about the status of the selected directory.

- Free: 81.106 Mb: the free disk space available
- Files: (0) 72: the number of selected files between parentheses, followed by the total number of files.
- (0.000) 8.600 Mb: the total number of bytes in the selected files between parentheses, followed by the total for the entire directory.

FileWindow

The FileWindow is generally called up to read and write files. However, you can also use it to manage the entire file system. It also provides a handy overview of the internal structure of a Blender file, since it is the window used to browse a .blend file content when an append action **SHIFT-F1** is called for.

There are 4 'modes' for the FileWindow



- FileManager: the standard mode.
- FileSelect: the FileHeader shows the action to be performed (Load, Save, etc.).
- DataSelect: display the Blender data system as files.
- DataBrowse: like DataSelect, but now as an alternative to a PopupMenu.

The FileWindow is optimised for reuse. The second and subsequent times the same directory is called up, the file system is not re-read. This saves considerable time, but can sometimes cause confusion if other processes have written files to the directory (the directory is always read again after Blender writes to it). If you have any doubts about the validity of the current display, press **DOTKEY**.

button

Displays the parent directory. You can also use: **PKEY**.

Directory Name

The text right to the shows the current directory. You can also create a new directory. When you leave the text line (with LeftMouse or ENTER), you will be asked: "OK? Make dir".

Preset Directories

The file $HOME/.Bfs contains a number of presets that are displayed in this menu. If a file is read or written, the directory involved is temporarily added to the menu.

File Name

The file name can be entered here. This text line can also be used to select files using *wildcards*. Example: enter '*.tga', then press **ENTER**. All files with the extension '.tga' are selected.

FileSelect

Blender commands such as F1 (read file) and F2 (write file) invoke a FileWindow in FileSelect mode. This mode is used to indicate a single file, i.e. the file in the FileName button. Press ENTER to initiate the action, e.g. read a Blender file. Use ESC to cancel the action. The FileManager functions also work in FileSelect mode. These only work on *selected* files. Standard functions in this window:

LMB

Indicates the *active* file. Its name is placed in the FileName button.

MMB

Indicates the *active* file and closes the FileWindow with the message OK.

RMB

Select files. For functional reasons, a **RMB** Select here does not indicate the *active* file!

ENTER

Closes the FileWindowperforming the desired action, returns with an OK message.

ESC

Closes the FileWindow with no further action.

PAGEDN

Scrolls down one page.

PAGEUP

Scrolls up one page.

HOME

Scrolls to the first file.

END

Scrolls to the last file.

NUM+

Automatic file-number increase. If there is a number in the active file name (such as rt01.tga), this number is incremented by one. This is quite handy when reading or writing sequential files.

NUM-

Automatic file number decrease (see previous description).

SLASH

Make the current directory the root directory: "/"

PERIOD

Re-read the current directory.

EKEY

For the *active* file: start the Unix editor defined in environment variable $WINEDITOR. For viewing text files.

IKEY

For the *active* file: start the Unix image viewer definde in environment variable $IMAGEEDITOR . For viewing or Editing images.

Read Libraries

Blender allows you to read in, append, or *link* (parts of) other files. If 'Load Library' is selected by invoking the File Whindow with **SHIFT-F1** the FileSelect appears in a special mode. Blender files are now highlighted as directories. They are accessible as a directory as well; they then display a situation in much the same way as DataView

displays the internal Blender structure. Now you can select any number of blocks you wish using RightMouse and append them to the current structure with a simple ENTER. The complete 'underlying' structure is included: thus, if an Object is selected, the associated Ipo, ObData, Materials and Textures are included as well. If a Scene is selected, the entire Scene is appended to the current structure, Objects and all.

You can specify how you want this to be appended in the FileSelect Header:

Append

External blocks become a normal part of the current structure, and thus of the file as well, if the file is saved. Appending a second time causes the entire selection to be added again in its entirety. Since block names must be unique, the name of the appended blocks will differ slightly from the existing name (only the number in the name).

Link

This is the 'normal' use of Libraries. The specified blocks are added to the current structure here as well, but Blender remembers the fact that they are Library blocks. When the file is saved, only the name of the Library block and the name of the file from which the blocks were copied are saved, thus keeping the original file compact. When you read the file again, Blender reads the original file and then reads all the Library blocks from the other file(s). The names of the files and the blocks must not be changed, however. Blender keeps track of what Library blocks have already been read. Appending the same blocks twice has no consequences.

This enables more animators working on a project. Therefore a linked Object cannot be changed from the scene it is imported in.

FileManager

The FileManager function only works with selected files. The active file does not play a role here. Most of these commands do expect two FileWindows to be open. Commands such as **RKEY** (remove) and **TKEY** (touch) also work with a single Window. Note: when we say *files* here, we also mean directories.

AKEY

Select/deselect all files.

BKEY

Backup files to the other FileWindow. This allows files to be copied without changing the file date.

CKEY

Copy files to the other FileWindow. (Unix: cp -r) read.

LKEY

Link files to the other FileWindow. (Unix: ln)

MKEY

Move files to the other FileWindow. (Unix: mv)

RKEY

Remove files. For safety's sake: only empty directories. (Unix rm -f)

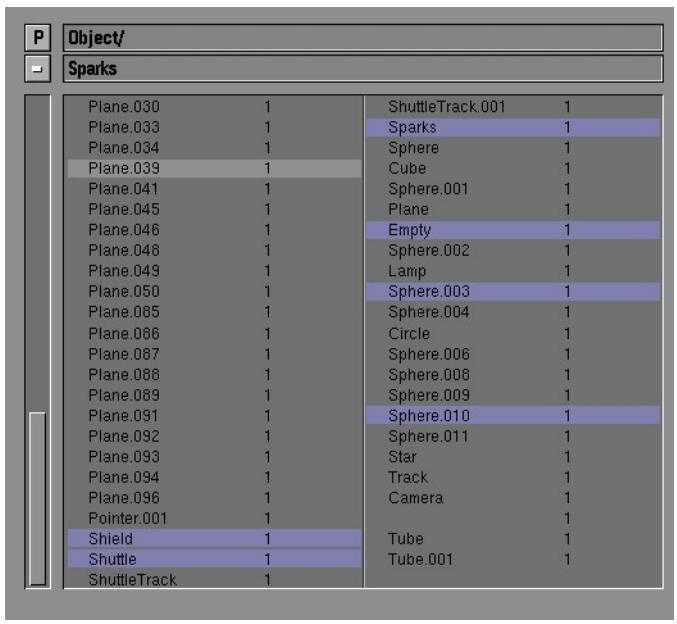
SHIFT-R

Remove files recursively. This deletes the entire contents of directories. (Unix: rm -rf)

TKEY

Update modification times of files. (Unix: touch)

DataView and DataBrowse



The DataView window can be invoked with **SHIFT-F4**. It allows you to view the entire internal Blender structure as a file system. Each DataBlock is listed as a file name. They are sorted by type in directories.

Currently, the functions are limited to:

- Select Objects by name. Click RightMouse on the file name. A LeftMouse click on a name makes the Object *active*. Note that an activated Object can also reside in a hidden *layer*.

- Setting and deleting Fake Users. (Press FKEY). A Fake User ensures that a DataBlock is always saved in a file, even if it has no users.
- Link Materials (**CTRL-L**). The links to *selected* Materials are all replaced by a link to the *active* Material (LeftMouse, in the FileName button). This link option will be expanded to other DataBlock types.

PopupMenus that contain more than 24 items and are thus unmanageably large, are replaced by the DataBrowse windows. Standard functions in this window:

LMB

Display the *active* DataBlock. This is placed in the FileName button.

MMB

Display the *active* DataBlock and close the DataBrowse with an OK message.

ENTER

Close the DataBrowse, return with an OK message.

ESC

Close the DataBrowse with no further action.

PAGEDN

Scroll down one page.

PAGEUP

Scroll up one page.

The 3DWindow

3D Toolbar

WindowType

As with every window header, the first button left allows you to set the window type.

Full Window

Maximise the window, or return it to its original size; return to the old screen setting (**CTRL-UPARROW**).

Home

All Objects in the visible layers are displayed completely, centered in the window (**HOME**).

Layers buttons

These 20 buttons show the available layers. In fact, a layer is nothing more than a *visibility flag*. This is an extremely efficient method for testing Object visibility. This allows the user to divide the work functionally.

For example: Cameras in layer 1, temporary Objects in layer 20, lamps in layers 1, 2, 3, 4 and 5, etc. All hotkey commands and tools in Blender take the layers into account. Objects in 'hidden' layers are treated as *unselected*.

Use **LMB** for selecting, **SHIFT-LMB** for *adding/removing* to/from the group of selected layers.

Hotkeys: **1KEY** , **2KEY** , etc. **0KEY** for layers 1,2,3,4, etc. Use **ALT-1**, **ALT-2** , ... **ALT-0**) for layers 11, 12, ... 20. Here, as well, use **SHIFT** + Hotkey *adding/removing* to/from the group of selected layers.

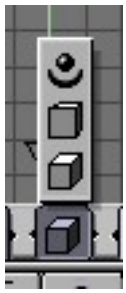
Lock

Every 3DWindow has it's own layer setting and active Camera. This is also true for a Scene: here it determines what layers - and what camera - are used to render a picture. The *lock* option links the layers and Camera of the 3DWindow to the Scene and vice versa: the layers and Camera of the Scene are linked to the 3DWindow. This method passes a layer change directly to the Scene and to all other 3DWindows with the "Lock" option ON. Turn the "Lock" OFF to set a layer or Camera *exclusively* for the current 3DWindow. All settings are immediately restored by turning the button back ON.

LocalView

LocalView allows the user to continue working with complex Scenes. The currently selected Objects are taken separately, centered and displayed completely. The use of 3DWindow layers is temporarily disabled. Reactivating this option restores the display of the 3DWindow in its original form. If a picture is rendered from a LocalView, only the Objects present are rendered *plus* the visible lamps, according to the layers that have been set. Activating a new Camera in LocalView does not change the Camera used by the Scene. Normally, LocalView is activated with the hotkey **NUM/**.

View Mode



A 3DWindow offers 3 methods for 3D display:

- Orthonormal. Blender offers this method from every view, not just from the X, Y or Z axes.
- Perspective. You can toggle between *orthonormal* and *perspective* with the **NUM5**.
- Camera. This is the *view* as rendered (**NUM0**).

View Direction

These pre-sets can be used with either *ortho* or *perspective*. Respectively, these are the:

- Top View, hotkey **NUM7**
- Front View, hotkey **NUM1**
- Right View, hotkey **NUM3**

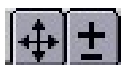
The hotkeys combined with **SHIFT** give the opposite view direction. (Down View, Back View, Left View)

Draw Mode

Set the drawing method. Respectively:

- BoundBox. The quickest method, for animation previews, for example.
- WireFrame.
- Solid. Zbuffered with the standard OpenGL lighting. **ZKEY** toggles between WireFrame and Solid.
- Shaded. This is as good an approach as is possible to the manner in which Blender renders - with Gouraud shading. It displays the situation from a single frame of the Camera. **SHIFT+Z** toggles, use **CTRL+Z** to force a recalculation.
- Textured.

Objects have their own Draw Type, independent of the window setting (see Edit-Buttons>>DrawType). The rule is that the *minimum* (more economic) DrawMode is displayed.



View Move

Move the mouse for a *view* translation. This is an alternative for **SHIFT-MMB**.

View Zoom

Move the mouse vertically to zoom in and out of the 3DWindow. This is an alternative for **CTRL+MMB**.

These buttons determine the manner in which the Objects (or vertices) are *rotated* or *scaled*.

Around Center

The midpoint of the *boundbox* is the center of rotation or scaling. Hotkey: **COMMA**.

Around Median

The median of all Objects or vertices is the center of rotation or scaling.

Around Cursor

The 3DCursor is the midpoint of rotation or scaling. Hotkey: **DOT**.

Around Individual Centers

All Objects rotate or scale around their own midpoints. In EditMode: all vertices rotate or scale around the Object midpoint.

EditMode

This button starts or terminates EditMode (**TAB** or **ALT+E**).

VertexPaint

This button starts or terminates VertexPaintMode (**VKEY**).

TexturePaint

This button starts or terminates TexturePaintMode.

FaceSelect

This button starts or the FaceSelect mode (**FKEY**).

Proportional Vertex Editing Tool

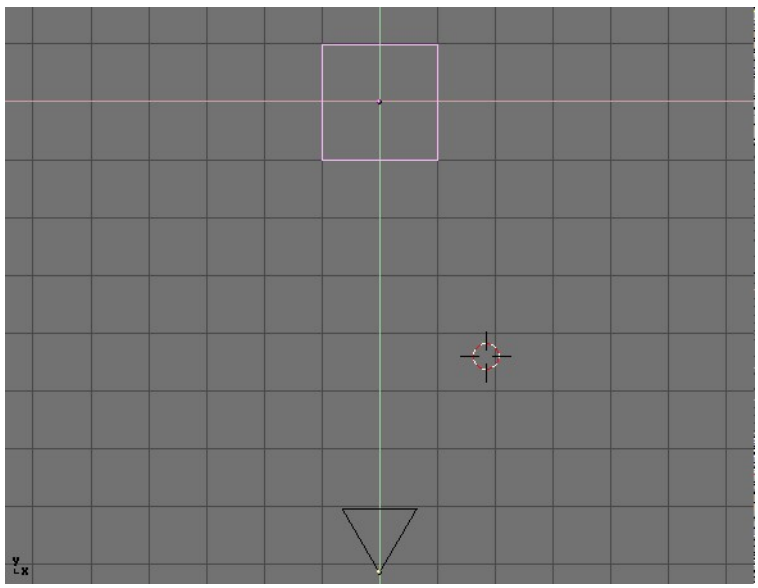
The Proportional Editing tool can be activated, only in editmode, with the Icon in 3D window header, or **OKEY**. The Proportional Editing tool is then Available in Editmode for all Object types. This tool works like a 'magnet', you select a few vertices and while editing (grab, rotate, scale) the surrounding vertices move proportinal with it. Use **NUM+** and **NUM-** keys or **MW** to adjust the area of influence, this can be done "live" while editing.

You can choose between a sharp fallof and a smooth falloff.

OpenGL Renderer

A **LMB** renders the actuall view in OpenGL. **CTRL-LMB** renders a animation in OpenGL. The rendered pictures are saves as in the DisplayButtons indicated.

3DWindow



The standard 3DWindow has:

- A grid. The dimensions (distance between the gridlines) and resolution (number of lines) can be set with the ViewButtons. This grid is drawn as infinite in the presets of *ortho* ViewMode (Top, Front, Right view). In the other *views*, there is an finite 'floor'. Many Blender commands are adjusted to the *dimension* of the grid, to function as a standard unit. Blender works best if the total 'world' in which the user operates continually falls more or less within the total grid floor (whether it is a space war or a logo animation).
- Axes in colour codes. The reddish line is the X axis, the green line is the Y axis, the blue line is the Z axis. In the Blender universe, the 'floor' is normally formed by the X and Y axes. The height and 'depth' run along the Z axis.
- A 3D cursor. This is drawn as a black cross with a red/white striped circle. A LeftMouse click moves the 3DCursor. Use the SnapMenu (**SHIFT-S**) to give the 3Dcursor a specific location. New Objects are placed at the 3D cursor location.
- Layers (visible in the header buttons). Objects in 'hidden' layers are not displayed. All hotkey commands and tools in Blender take the layers into account: Objects in the 'hidden' layers are treated as *not* selected. See the following paragraph as well.
- ViewButtons. Separate variables can be set for each 3Dwindow, e.g for the *grid* or the *lens*. Use the **SHIFT-F7** hotkey or the WindowType button in the 3DHeader. The ViewButtons are explained in detail elsewhere in this manual.

The Mouse

The mouse provides the most direct access to the 3DWindow. Below is a complete overview:

LMB

Position the 3DCursor.

CTRL-LMB

In EditMode: create a new vertex.

Press LMB and drag

These are the Gestures. Blender's gesture recognition works in three ways:

- Draw a straight line: start *translation* mode (Grabber)
- Draw a curved line: start *rotation* mode.
- Draw a V-shaped line: start *scaling* mode.

MMB

Rotate the direction of view of the 3DWindow. This can be done in two ways (can be set in the UserMenu):

- the *trackball* method. In this case, where in the window you start the mouse movement is important. The rotation can be compared to rotating a ball, as if the mouse grasps and moves a tiny miniscule point on a ball and moves it. If the movement starts in the middle of the window, the *view* rotates along the horizontal and vertical window axes. If the movement begins at the edge of the window, the *view* rotates along the axis perpendicular to the window.
- the *turntable* method. A horizontal mouse movement always results in a rotation around the global Z axis. Vertical mouse movements are corrected for the view direction, and result in a combination of (global) X and Y axis rotations.

SHIFT-MMB

Translate the 3DWindow. Mouse movements are always corrected for the view direction.

CTRL-MMB

Zoom in/out on the 3DWindow.

RMB

Select Objects or (in EditMode) vertices. The last one selected is also the *active* one. This method guarantees that a maximum of 1 Object and 1 vertex are always selected. This selection is based on graphics (the wireframe).

SHIFT-RMB

Adds/removes from selection Objects or (in EditMode) vertices. The last one selected is also the *active* one. Multiple Objects or *vertices* may also be selected. This selection is based on graphics too (the wireframe).

CTRL-RMB

Select Objects on the Object-centers. Here the wireframe drawing is not taken into account. Use this method to select a number of identical Objects in succession, or to make them active.

SHIFT-CTRL-RMB

Adds/removes from selection Objects. The last Object selected is also the *active* one. Multiple Objects can be selected.

ALT-CTRL-RMB

In Edit Mode: edge select.

SHIFT-ALT-CTRL-RMB

In Edit Mode: add/removes edges from selection.

Press RMB and drag

Select and start *translation* mode, the Grabber. This works with all the selection methods mentioned.

NumPad

The numeric keypad on the keyboard is reserved for *view* related hotkeys. Below is a description of all the keys with a brief explanation.

NUM/

LocalView. The Objects selected when this command is invoked are taken separately and displayed completely, centered in the window. See the description of 3DHeader>>LocalView.

NUM*

Copy the rotation of the *active* Object to the current 3DWindow. Works as if this Object is the camera, without including the translation.

NUM- , NUM+

Zoom in, zoom out. This also works for Camera ViewMode.

NUM.

Center and zoom in on the selected Objects. The *view* is changed in a way that can be compared to the LocalView option.

NUM5

Toggle between *perspective* and *orthonormal* mode.

NUM9

Force a complete recalculation (of the animation systems) and draw again.

NUM0

View from the current *camera*, or from the Object that is functioning as the *camera*.

CTRL-NUM0

Make the *active* Object the *camera*. Any Object can be used as the camera. Generally, a Camera Object is used. It can also be handy to let a spotlight function temporarily as a camera when directing and adjusting it. **ALT+NUM0** Revert to the previous *camera*. Only Camera Objects are candidates for 'previous camera'.

NUM7

Top View. (along the negative Z axis, Y up)

SHIFT-NUM1

Down View. (along the positive Z axis, Y up)

NUM1

Front View. (along the positive Y axis, Z up)

SHIFT-NUM1

Back View. (along the negative Y axis, Z up)

NUM3

Right View. (along the negative X axis, Z up)

SHIFT-NUM3

Left View. (along the positive X axis, Z up)

NUM2 NUM8

Rotate using the *turntable* method. Depending on the view, this is a rotation around the X and Y axis.

NUM4 NUM6

Rotate using the *turntable* method. This is a rotation around the Z axis.

SHIFT-NUM2 SHIFT-NUM8

Translate up or down; corrected for the view.

SHIFT-NUM4 SHIFT-NUM6

Translate up or down; correct for the view.

Hotkeys

This list contains all the hotkeys that can be used in a 3D window. EditMode hotkeys are described in the following paragraph.

HOME

All Objects in the visible layer are displayed completely, centered in the window.

PAGEUP

Select the next Object Key. If more than one Object Key is selected, the selection is shifted up cyclically. Only works if the AnimButtons->DrawKey is ON for the Object.

SHIFT-PAGEUP

Adds to selection the next Object Key.

PAGEDOWN

Select the previous Object Key. If more than one Object Key is selected, the selection is shifted up cyclically. Only works if the AnimButtons->DrawKey is ON for the Object.

SHIFT-PAGEDOWN

Adds to selection the previous Object Key.

ACCENT

(To the left of the **1KEY** in US keyboard) Select all layers.

SHIFT-ACCENT

Revert to the previous layer setting.

AKEY

Select All / deselect All. If any Object or vertex is selected, everthing is always deselected first.

SHIFT-A

This is the AddMenu. In fact, it is the ToolBox that starts with the 'ADD' option. When Objects are added, Blender starts EditMode immediately if possible.

CTRL-A

"Apply size/rot". The rotation and dimensions of the Object are assigned to the ObData (Mesh, Curve, etc.). At first glance, it appears as if nothing has changed, but this can have considerable consequences for animations or texture mapping. This is best illustrated by also having the axis of a Mesh Object be drawn (EditButtons->Axis). Rotate the Object and activate Apply. The rotation and dimensions of the Object are 'erased'.

CTRL-SHIFT-A

If the active Object is automatically duplicated (see AnimButtons->DupliFrames or AnimButtons->Dupliverts), a menu asks "Make dupli's real?". This option *actually* creates the Objects. If the active Mesh Object is deformed by a Lattice, a menu asks "Apply Lattice deform?". Now the deformation of the Lattice is assigned to the vertices of the Mesh.

BKEY

Border Select. Draw a rectangle with the LeftMouse; all Objects within this area are *selected*, but *not* made active. Draw a rectangle with the RightMouse to *deselect* Objects. In orthonormal ViewMode, the dimensions of the rectangle are displayed, expressed as global coordinates, as an extra feature in the lower left corner. In Camera ViewMode, the dimensions that are to be rendered according to the DisplayButtons are displayed in *pixel* units.

SHIFT-B

Render Border. This only works in Camera ViewMode. Draw a rectangle to render a smaller cut-out of the standard window frame. If the option DisplayButtons->Border is ON, a box is drawn with red and black lines.

CKEY

Centre View. The position of the 3DCursor becomes the new centre of the 3DWindow.

SHIFT-C

CentreZero View. The 3DCursor is set to zero (0,0,0) and the *view* is changed so that all Objects, including the 3Dcursor, can be displayed. This is an alternative for **HOME**.

ALT-C

Convert Menu. Depending on the *active* Object, a PopupMenu is displayed. This enables you to convert certain types of ObData. It only converts in one direction, everything ultimately degrades to a Mesh! The options are:

- "Font -> Curve"
- "MetaBall -> Mesh" The original MetaBall remains unchanged.
- "Curve -> Mesh"
- "Surface -> Mesh"

CTRL-C

Copy Menu. This menu copies information from the *active* Object to (other) *selected* Objects.

- Fixed components are:
  - "Copy Loc": the X,Y,Z location of the Object. If a Child is involved, this location is the relative position in relation to the Parent.
  - "Copy Rot": the X,Y,Z rotation of the Object.
  - "Copy Size": the X,Y,Z dimension of the Object.
  - "DrawType" : see EditButtons->DrawType.
  - "TimeOffs" : see AnimButtons->TimeOffs.
  - "Dupli": all Duplicator data from the AnimButtons

- If applicable:
  - "Copy TexSpace": The texture space.
  - "Copy Particle Settings": the complete particle system from the AnimButtons.

- For Curve Objects:
  - "Copy Bevel Settings": all beveling data from the EditButtons.

- Font Objects:
  - "Copy Font Settings": font type, dimensions, spacing.
  - "Copy Bevel Settings": all beveling data from the EditButtons.

- Camera Objects:

- "Copy Lens": the lens value.

SHIFT-D

"Add Duplicate". The selected Objects are copied. The settings in the UserMenu (Duplication/Linking presets) determine what DataBlocks are also copied or are linked. Blender then automatically starts Grab Mode (**GKEY**).

ALT-D

"Add Linked Duplicate". The selected Objects are copied. The DataBlocks linked to Objects remain linked. Blender then automatically starts Grab Mode (**GKEY**).

CTRL-D

Draw the (texture) Image as wire. This option has a limited function. It can only be used for 2D compositing.

ALT-E

Start / stop EditMode. Alternative hotkey: TAB.

SHIFT-F

Fly Mode. Only from Camera ViewMode. The mouse cursor jumps to the middle of the window. It works as follows:

- Mouse cursor movement indicates the view direction.
- **LMB** (repeated): Fly faster.
- **MMB** (repeated): Fly slower.
- **LMB** and **MMB** toghether: Set speed to zero.
- **CTRL**: translation downwards (negative Z).
- **ALT**: translation upwards (positive Z).
- **ESC**: Camera back to its starting position, terminate Fly Mode.
- **SPACE**: Leave the Camera in current position, terminate Fly Mode.

(Be careful when looking straight up or down. This causes confusing turbulence.)

CTRL-F

Sort Faces. The *faces* of the *active* Mesh Object are sorted, based on the current view in the 3DWindow. The leftmost *face* first, the rightmost last. The sequence of *faces* is important for the Build Effect (AnimButtons).

GKEY

Grab Mode. Or: the *translation* mode. This works on selected Objects and *vertices*. Blender calculates the quantity and direction of the translation, so that they correspond *exactly* ù with the mouse movements, regardless of the ViewMode or view direction of the 3DWindow. Alternatives for starting this mode:

- **RMB**
- **LMB** to draw a straight line.

The following options are available in *translation* mode:

- Limitors:
  - **CTRL**: in increments of 1 grid unit.
  - **SHIFT**: fine movements.
  - **SHIFT-CTRL**: in increments of 0.1 grid unit.

- **MMB** togglse: A short *click* restricts the current translation to the X,Y or Z axis. Blender calculates which axis to use, depending on the already initiated mouse movement. Click MiddleMouse again to return to unlimited translation.
- **ARROWS**: These keys can be used to move the mouse cursor exactly 1 pixel.
- Grabber can be terminated with:
  - **LMB SPACE** or **ENTER**: move to a new position.
  - **RMB** or **ESC**: everything goes back to the old position.

- Switching mode:
  - **GKEY**: starts Grab mode again.
  - **SKEY**: switches to Size mode.
  - **RKEY**: switches to Rotate mode.

ALT+G

Clear location. The X,Y,Z locations of selected Objects are set to zero.

IKEY

Insert Object Key. A *keyposition* is inserted in the current frame of all *selected* Objects. A PopupMenu asks what key position(s) must be added to the IpoCurves.

- "Loc": The XYZ location of the Object.
- "Rot": The XYZ rotation of the Object.
- "Size": The XYZ dimensions of the Object
- "LocRot": The XYZ location and XYZ rotation of the Object.
- "LocRotSize": The XYZ location, XYZ rotation and XYZ dimensions of the Object.
- "Layer": The layer of the Object.

- "Avail": A position is only added to all the current IpoCurves, that is curves which already exists.
- "Effector": (only for Ika Objects) the end-effector position is added.
- "Mesh ", "Lattice ", "Curve" or "Surface": depending on the type of Object, a VertexKey can be added

CTRL-J

Join Objects. All *selected* Objects of the same type are added to the *active* Object. What actually happens here is that the *ObData* blocks are combined and all the selected Objects (except for the *active* one) are deleted. This is a rather complex operation, which can lead to confusing results, particularly when working with a lot of linked data, animation curves and hierarchies.

KKEY

Show (as) Keys. The DrawKey option is turned ON for all selected Objects. If all of them were already ON, they are all turned OFF. #->$11

SHIFT-K

A PopupMenu asks: "OK? Show and select all keys". The DrawKey option is turned ON for all selected Objects, and all Object-keys are selected. This function is used to enable *transformation* of the entire animation system.

LKEY

Local Menu. Makes library linked objects local for the current scene.

SHIFT-L

Select Links menu. This menu enables you to select Objects that share links a DataBlock with the *active* Object. Use this function to obtain an overview of the sometimes quite complicated linked structures Blender can create.

- "Object Ipo": all Objects with the same Object Ipo are selected.
- "Object Data": all Objects with the same ObData (Mesh, Curve, etc.) are selected.
- "Current Material": all Objects with the same *active* Material are selected (this is the Material in MaterialButtons).
- "Current texture": all Objects with the same *active* Texture are selected (this is the Texture in MaterialButtons).

CTRL-L

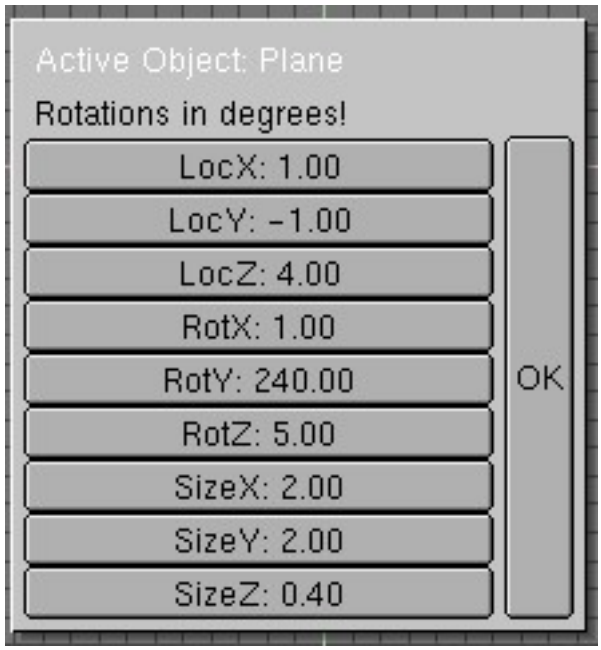
Make Links menu. This menu is used to copy links between the *active* Object and selected Objects. Only the links (the references) are copied; the contents of the DataBlocks involved are not changed. The result of this operation is easy to see in the OopsWindow.

- "To Scene": the selected Objects are also linked to another Scene. A second PopupMenu asks the user to specify a Scene. Objects that appear in more than one Scene are displayed with a blue null point.
- "Object Ipo": all selected Objects are given a link to the Object Ipo of the *active* Object.
- "Mesh data", "Curve data", "Font data", etc.: all selected Objects are given a link to the ObData of the *active* Object. Objects must be the same type!
- "Materials": all selected Objects are given links that are identical to the Material(s) of the *active* Object. The entire Material situation is copied. If the *active* Object does not have any Materials, all Material links for the selected Objects are erased.

MKEY

Move to Layer. This hotkey calls up a menu that can be used to view or change the layer settings of all the *selected* Objects. If the selected Objects have different layers, this is 'OR'ed in the menu display. Use ESC to exit the menu. Press the "OK" button or **ENTER** to change the layer seting. The hotkeys (**ALT**-)(**1KEY**, **2KEY**, ... - **0KEY**) work here as well (see 3DHeader).

NKEY



These buttons allow the user to enter numbers for the position, rotation and dimensions of the *active* Object. Use **ESC** to exit the menu without changing the Object. Press the "OK" button or **ENTER** to assign the changes to the Object.

ALT-O

Clear Origin. The 'Origin' is erased for all Child Objects, which causes the Child Objects to move to the exact location of the Parent Objects.

CTRL-P

Make Parent. The *active* Object becomes the Parent of the selected Objects. All transformations of the Parent are now passed on to the Children. This allows you to create complex hierarchies. As part of the 'Make Parent' operation, an *inverse* of the Parent transformation is calculated and stored in the Child Object. This *inverse* may make it seem as if all transformations remained unchanged *after* Make Parent was executed. Depending on the type of Object, special Parent relationships can be selected.

ALT-P

Clear Parent. All selected Child Objects are unlinked from the Parents. A PopupMenu asks you to make a selection:

- "Clear Parent": the selected Child Objects are unlinked from the Parent. since the transformation of the Parent disappears, this can appear as if the former Children themselves are transformed.
- "... and keep transform": the Child Objects are unlinked from the Parent, and an attempt is made to assign the current transformation, which was determined in part by the Parent, to the (former Child) Objects.
- "Clear Parent inverse": The inverse matrix of the Parent of the selected Objects is erased. The Child Objects remain linked to the Objects. This gives the user complete control over the hierarchy.

RKEY

Rotate mode. Works on selected Objects and *vertices*. In Blender, a rotation is by default a rotation perpendicular to the screen, regardless of the view direction or View-Mode. The degree of rotation is *exactly* linked to the mouse movement. Try moving around the rotation midpoint with the mouse. The rotation midpoint is determined by the state of the 3DHeader->Around buttons.

XKEY , YKEY and ZKEY

While in rotation mode, these keys switch to a global axis rotation around the corresponding axis. Alternatives for starting *rotation* mode:

- **LMB** (click-hold-draw): draw a C-shaped curve. The following options are available in *rotation* mode:
- Limitors:
  - **CTRL**: in increments of 5 degrees.
  - **SHIFT**: finer control.
  - **SHIFT**-**CTRL**: finer control with increments of 1 degree.

- **MMB** toggles the current rotation axis, which is perpendicular to the screen, with the two other axes, which are vertical and horizontal on the screen. The mouse movements here follow the two rotation axes. Click MiddleMouse again to return to a rotation axis perpendicular to the screen.
- **ARROWS**: These keys move the mouse cursor exactly 1 pixel.
- Switching mode:
  - **RKEY**: starts Rotate mode again.
  - **SKEY**: switches to Size mode.
  - **GKEY**: switches to Grab mode.

- Rotation mode can be terminated with:
  - **LMB SPACE ENTER**: finalizes the rotation.
  - **RMB** or **ESC**: everything returns to the former state.

ALT-R

Clear Rotation. The X,Y,Z rotations of selected Objects are set to zero.

SKEY

Size mode or *scaling* mode. Works on selected Objects and *vertices*. The degree of *scaling* is *exactly* linked to the mouse movement. Try to move from the (rotation) midpoint with the mouse. The midpoint is determined by the settings of the 3DHeader->Around buttons. Alternatives for starting scaling mode:

- LeftMouse (click-hold-draw) draw a V-shaped line.

The following options are available in scaling mode:

- Limitors:
  - **CTRL**: in steps of 0.1.
  - **SHIFT-CTRL**: in steps of 0.01.

- **MMB** restricts the scaling to the X, Y or Z axis. Blender calculates the appropriate axis based on the already initiated mouse movement. Click MiddleMouse again to return to free scaling.
- **ARROWS**: These keys move the mouse cursor exactly 1 pixel.
- **XKEY** Make the horizontal scaling negative; this is also called an X-flip.
- **YKEY** Make the vertical scaling negative; this is also called a Y-flip.
- Switching mode:
  - **SKEY**: starts Size mode again.
  - **RKEY**: switches to Rotate mode.
  - **GKEY**: switches to Grab mode.

Terminate Size mode with:

- **LMB SPACE** or **ENTER**: finalizes scaling.
- **RMB** or **ESC**: everything returns to its previous state.

ALT-S

Clear Size. The X,Y,Z dimensions of selected Objects are set to 1.0.

SHIFT-S

Snap Menu. This menu offers a number of options for precisely specifying the position of Objects or vertices.

- "Sel -> Grid": all selected Objects or vertices are moved to the closest grid position.
- "Sel -> Curs": all selected Objects or vertices are moved to the 3DCursor position.
- "Curs -> Grid": the 3DCursor is moved to the closest grid position.
- "Curs -> Sel": the 3DCursor is moved to the selected Object. If there are multiple Objects or vertices, the 3DCursor is set to the midpoint. The 3DHeader->Around buttons determine what type of midpoint is meant here.

Tip: use the PopupMenu hotkeys **1KEY**, **2KEY**, etc. to select the options.

TKEY

Texture space mode. The position and dimensions of the texture space for the selected Objects can be changed in the same manner as described above for Grab and Size mode. To make this visible, the *drawingflag* EditButtons->TexSpace is set ON. A PopupMenu asks you to select: "Grabber" or "Size".

CTRL-T

Make Track. All selected Objects are given a rotation *constraint* directed at the *active* Object. The type of rotation and which axis is up and which one is directed at the Track Object are specified in the AnimButtons.

ALT-T

Clear Track. The *tracking* is turned off for all selected Objects. A PopupMenu allows you to maintain the current *tracking* as the rotation in the Objects.

UKEY

Single User Menu. Use this menu to manage the (multi-) user structure. These operations only work on selected Objects and the DataBlocks that are linked to them.

- "Object": if other Scenes also have a link to this Object, the link is deleted and the Object is copied. The Object now only exists in the current Scene. The links *from* the Object remain unchanged.
- "Object & ObData": Similar to the previous command, but now the ObData blocks with multiple links are copied as well. All selected Objects are now present in the current Scene only, and each has a unique ObData (Mesh, Curve, etc.).
- "Object & ObData & Materials+Tex": Similar to the previous command, but now Materials and Textures with multiple links are also copied. All selected Objects are now unique. They have unique ObData and each has a unique Material and Texture block.
- "Materials+Tex": Only the Materials and Textures with multiple links are copied.

VKEY

Start or exit VertexPaint Mode.

ALT-V

Object-Image Aspect. This hotkey sets the X and Y dimensions of the selected Objects in relation to the dimensions of the Image Texture they have. Use this hotkey when making 2D Image compositions and multi-plane designs to quickly place the Objects in the appropriate relationship with one another.

ALT-W

Write Videoscape. The selected Mesh Object is saved as an ASCII file. Use the FileWindow to enter a file name. Blender adds the extension ".obj" to the Videoscape file.

XKEY

Erase Selected. All selected Objects are deleted from the Scene and, if they are not linked to other Scenes, they are released. The ObData and other linked DataBlocks remain.

ZKEY

DrawMode Solid ON/OFF. This is Zbuffered with the standard OpenGL lighting.

SHIFT-Z

DrawMode Shaded ON. This drawing mode, which is Zbuffered and Gouraud shaded, approaches the way in which Blender renders as closely as possible. It shows the situation from a single Camera frame.

CTRL+Z

Turn DrawMode Shaded ON *and* force a recalculation of DrawMode Shaded.

The IpoWindow

IpoHeader

WindowType

As with every window header, the first button enables you to set the window type. Full Window (IconTog) Maximise the window or return it to the previous window display size; return to the old screen setting (**CTRL-UPARROW**).

Home

All visible curves are displayed completely, centered in the window (**HOME**).

IpoKeys

This is a drawing mode for the animation curves in the IpoWindow (the IpoCurves). Yellow vertical lines are drawn through all the vertices of the curves. Vertices of different curves at the same location in 'time' are joined together and can easily be selected, moved, copied or deleted. This method adds the ease of traditional *key* framing to the animation curve system. For Object-Ipos, these IpoKeys can also be drawn and transformed in the 3DWindow. Changes in the 3D position are processed in the IpoCurves immediately. #->$11

Ipo Type

Depending on the *active* Object, the various Ipo systems can be specified with these buttons. From left to right in the image (in Blender window they are not all present at all times and not in this order).

Object Ipo

Settings, such as the location and rotation, are animated for the *active* Object. All Objects in Blender can have this Ipo block.

Material Ipo

Settings of the *active* Material are animated for the *active* Object. A NumBut appears as an extra feature immediately to the right when this button is selected. This button indicates the number of the active Texture *channel*. Eight Textures, each with its own mapping, can be assigned per Material. Thus, per Material-Ipo, 8 curves in the row `OfsX, OfsY, ...Var` are available.

World Ipo

Used to animate a number of settings for the WorldButtons. World too has several texture channels.

VertexKey Ipo

If the *active* Object has a VertexKey, the keys (Absolute or Relative) are drawn as horizontal lines. Only one IPOcurve is available to interpolate between the Absolute keys, or as many curves as Keys are allowed for Relative Keys.

Constrain Ipo

If the *active* Object has a *constrain* its influence value can ba animated via an IPO. Each constrain has its own IPO. used to display the speed-Ipo.

Sequence Ipo

The active Sequence Effect can have an IpoCurve.

Action Ipo

The active Armature IPOcurves are shown.

Speed Ipo

If the *active* Object is a *path* Curve, this button can be used to display the speed (time) IPO.

Camera Ipo

The active camera IPO curves are shown.

Lamp Ipo

If the *active* Object is a Lamp, this button can be used to animate light settings, comprehensive of textures.

The DataButtons can be used to control the Ipo blocks themselves.

Ipo Browse

Choose another Ipo from the list of available Ipos. The option "Add New" makes a complete copy of the current Ipo. This is not visible; only the name in the adjacent button will change. Only Ipos of the same type are displayed in the menu list.

IP:

Give the current Ipo a new and unique name. After the new name is entered, it appears in the list, sorted alphabetically.

Users

If this button is displayed, there is more than one user for the Ipo block. Use the button to make the Ipo "Single User".

Unlink Ipo

The current Ipo is unlinked.

Copy to Buffer

All selected IpoCurves are copied to a temporary buffer.

Paste from Buffer

All selected *channels* in the IpoWindow are assigned an IpoCurve from the temporary buffer. The rule is: the sequence in which they are copied to the buffer is the sequence in which they are pasted. A check is made to see if the number of IpoCurves is the same.

Extend mode Constant

The end of selected IpoCurves are horizontally extrapolated.

Extend mode Direction

The ends of selected IpoCurves continue extending in the direction in which they end.

Extend mode Cyclic

The full length of the IpoCurve is repeated cyclically.

Extend mode Cyclic Extrapolation

The full length of the IpoCurve is extrapolated cyclic.

View Zoom

Move the mouse horizontally or vertically to zoom in or out on the IpoWindow. This is an alternative for **CTRL-MMB**.

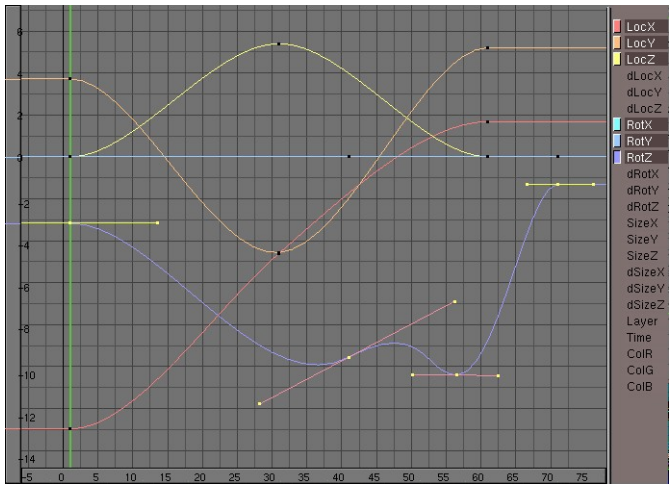
View Border

Draw a rectangle to indicate what part of the IpoWindow should be displayed in the full window.

Lock

This buton locks the update of the 3DWindow while editing in the IpoWindow, so you can see changes maked to the IPO in realtime inthe 3DWindow. This option works extremely well with relative vertex keys.

IpoWindow



The IpoWindow shows the contents of the Ipo block. Which one depends on the Ipo Type specified in the header. The standard IpoWindow has a grid with the time expressed horizontally in frames and vertical values that depend on the *channel*. There are 2 sliders at the edge of the IpoWindow. How far the IpoWindow is zoomed in can be seen on the sliders, which can also be used to move the *view*. The right-hand part of the window shows the available *channels*.

To make it easier to work with rotation-IpoCurves, they are displayed in degrees (instead of in radials). The vertical scale relation is: 1.0 'Blender unit' = 10 degrees.

In addition to the IpoCurves, the VertexKeys are also drawn here. These are horizontal blue lines; the yellow line visualises the *reference* Key.

Each *channel* can be operated with two buttons:

IpoCurve Select

This button is only displayed if the *channel* has an IpoCurve. The button is the same colour as the IpoCurve. Use the button to select IpoCurves. Multiple buttons can be (de)selected using SHIFT+LeftMouse.

Channel Select

A *channel* can be selected whether there is an IpoCurve or not. Only IpoCurves of selected channels are drawn. Multiple *channels* can be (de)selected using SHIFT+LeftMouse.

The Mouse

LMB and drag

These are the Gestures. Blender's gesture recognition works in two ways here:

- Draw a straight line: start *translation* mode (Grabber)
- Draw a V-shaped line: start *scaling* mode.

CTRL-LMB

Create a new vertex. These are the rules:

- There is no Ipo block (in this window) *and* one *channel* is selected: a new IpoBlock is created along with the first IpoCurve with one vertex.
- There is already an Ipo block, and a *channel* is selected without an IpoCurve: a new IpoCurve with one vertex is added
- Otherwise a new vertex is simply added to the selected IpoCurve.
- This is *not* possible if multiple IpoCurves are selected or if you are in EditMode.

MMB

Depending on the position within the window:

- On the *channels*; if the window is not high enough to display them completely, the visible part can be shifted vertically.
- On the sliders; these can be moved. This only works if you are zoomed in.
- The rest of the window; the *view* is translated.

CTRL-MMB

Zoom in/out on the IpoWindow. You can zoom horizonally or vertically using horizontal and vertical mouse movements.

RMB

Selection works the same here as in the 3DWindow: normally one item is selected. Use **SHIFT** to add/remove from the selection.

- If the IpoWindow is in IpoKey mode, the IpoKeys can be selected.
- If at least 1 of the IpoCurves is in EditMode, only its vertices can be selected.
- VertexKeys can be selected if they are drawn (horizontal lines)
- The IpoCurves can be selected.

RMB and drag

Select and start *translation* mode, i.e. the Grabber. The selection can be made using any of the four selection methods discussed above.

SHIFT-RMB

Adds/removes from the selection.

The HotKeys

NUM- , NUM+

Zoom in, zoom out.

PAGEUP

Select the next IPO Key. If more than one IpoKey is selected, the selection is shifted cyclically.

SHIFT-PAGEUP

Add the next IpoKey to the selection.

PAGEDOWN

Select the previous IPO Key. If more than one Object Key is selected, the selection is shifted cyclically.

SHIFT-PAGEDOWN

Add the previous IPO Key to the selection.

HOME

All visible curves are displayed completely, centered in the window.

TAB

All selected IpoCurves go into or out of EditMode. This mode allows you to transform individual vertices.

AKEY

Select All / deselect All. If any item is selected, first everything is deselected. Placing the mouse cursor above the *channels*, (de)selects all channels where there is a curve.

BKEY

Border select. Draw a rectangle with the LeftMouse; all items that fall within this rectangle are *selected*. Draw a rectangle with the RightMouse to *deselect*.

CKEY

If one vertex or one IpoKey is selected, the current *frame* number is set to this position.

SHIFT-D

Duplicate IPO. All selected vertices or IpoKeys are copied. Then *translation* mode is started automatically.

GKEY

Translation mode (the Grabber). This works on selected curves, keys or vertices. Alternatives for starting this mode:

- **RMB** and drag.
- **LMB** and draw a straight line.

The following options are available in *translation* mode:

- Limitors:
- **CTRL** increments of 1 frame or vertical unit.
- **SHIFT-CTRL** increments of 0.1 frame or vertical unit.
- **MMB** restricts the current translation to the X or Y axis. Blender calculates which axis to use, based on the already initiated mouse movement. Click MiddleMouse again to restore unlimited translation.
- **ARROWS**:
- With these keys the mouse cursor can be moved exactly 1 pixel.
- Grabber can be terminated with:
- **LMB SPACE** or **ENTER**: move to the new position.
- **RMB** or **ESC**: everything returns to the old position.

HKEY

Toggle Handle *align / free*.

SHIFT-H

Set Handle *auto*. The selected Bezier *handles* are converted to *auto* type.

IKEY

Insert Key. Vertices can be added to the visible curves in the IpoWindow. A Popup-Menu asks you to make a choice:

- "Current Frame"; all visible curves get a vertex on the current frame.
- "Selected Keys"; (only in IpoKey mode) all selected IpoKeys get vertices for each visible curve, including IpoCurves that are not part of the IPO Key.

JKEY

Join vertices. Selected vertices or IpoKeys can be joined. A PopupMenu asks you to make a choice:

- "All Selected"; all selected vertices are replaced by a new one.
- "Selected doubles": all selected vertices that are closer to each other than 0.9 *frame* are joined.

KKEY

IPO Key mode ON/OFF. If the Ipo block is Object Ipo type, the Objects are redrawn with the option DrawKey ON (see the explanation under IpoHeader).

RKEY

Recording

mode. The X and Y movements of the mouse are linked to the height of the IpoCurve. Thus, this works with a maximum of two selected *channels* or IpoCurves. The old curve is completely deleted; the new one becomes a 'linear' type. You cannot change parts of curves with *recording*. The scale at which this happens is determined by the *view* of the IpoWindow. A PopupMenu asks you to make a choice:

- "Still"; the current frame is used as the starting point.
- "Play anim"; the animation starts, allowing you to see the correlation with other animation systems.

During *recording* mode, the **CTRL** must be held down to actually start recording. Press **SPACEKEY** or **ENTER** or LeftMouse to stop *recording*. Use **ESCKEY** to undo changes.

SKEY

Scaling mode. This works on selected IpoCurves and vertices. The degree of *scaling* is *precisely* linked to the mouse movement. Try to move from the (rotation) midpoint with the mouse. In IpoKey mode, you can only *scale* horizontally. Alternatives for starting *scaling* mode:

- **LMB** draw a sharp angle; a V-shaped line. The following options are available in *scaling* mode:
- Limitors:
  - **CTRL**: in increments of 0.1.
  - **SHIFT-CTRL**: in increments of 0.01.

- **MMB** limits *scaling* to the X or Y axis. Blender calculates which axis to use based on the already initiated mouse movement. Click MiddleMouse again to return to free *scaling*.
- **ARROWS** These keys allow you to move the mouse cursor exactly 1 pixel.
- **XKEY** Make the horizontal *scaling* negative, the X-flip.
- **YKEY** Make the vertical *scaling* negative, the Y-flip.
- Terminate size mode with: **LMB SPACE** or **ENTER**: to finalize scaling.

  RMB or **ESC** : everything returns to its previous state.

SHIFT-S

Snap Menu.

- "Horizontal": The selected Bezier handles are set to horizontal.
- "To next": The selected handle or vertex is set to the same (Y) value as the next one.
- "To frame": The selected handles or vertices are set to the exact frame values.
- "To current frame": The selected handle or vertex is moved to the current frame.

TKEY

If an IpoCurve is selected: "Ipo Type". The type of selected IpoCurves can be changed. A PopupMenu asks you to make a choice:

- "Constant": after each vertex of the curve, this value remains constant, and is not interpolated.
- "Linear": linear interpolation occurs between the vertices.
- "Bezier": the vertices get a *handle* (i.e. two extra vertices) with which you can indicate the curvature of the interpolation curve.

If a Key is selected: "Key Type". The type of selected Keys can be changed.

- "Linear": linear interpolation occurs between the Keys. The Key line is displayed as a broken line.
- "Cardinal": fluent interpolation occurs between the Keys; this is the default.

- "BSpline": extra fluent interpolation occurs between the Keys, four Keys are now involved in the interpolation calculation. Now the positions *themselves* cannot be displayed precisely, however. The Key line is shown as a broken line.

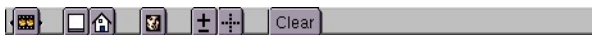
VKEY

Vector Handle. The selected Bezier *handles* are converted to *vector* type.

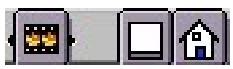
XKEY

Erase selected. The selected vertices, IpoKeys or IpoCurves are deleted. If there are selected VertexKeys, they are also deleted.

The SequenceWindow



Sequence Toolbar



WindowType

As with every window header, the first button allows you to set the type of window.

Full Window

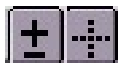
Maximise the window, or return to the previous window display size; return to the previous screen setting (**CTRL-UPARROW**).

Home

All visible Sequences are completely displayed, centered in the window (**HOME**).

DisplayImage

The window shows the result of the Sequences, i.e. a picture.



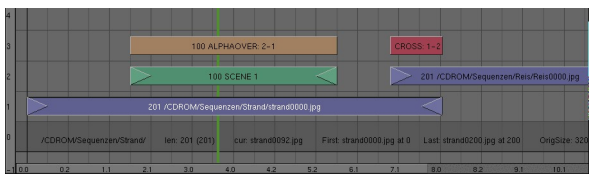
View Zoom

Move the mouse to zoom into or out of the SequenceWindow. This is an alternative for **CTRL-MMB** and **MW**. View Border (IconBut) Draw a rectangle to indicate what part of the SequenceWindow should be displayed in the full window.

Clear

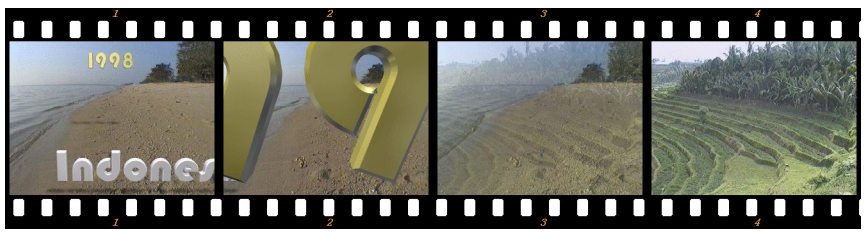
Force a clear of all buffered images in memory.

SequenceWindow



In the SequenceWindow you arrange your scenes, external pictures or movie files for the postproduction of your animation. The Strip in row 1 is a sequence of numbered jpeg-pictures. It will play for a few seconds and then `SCENE 1` in row 2 will be superimposed with the `ALPHAOVER` effect in row 3. The `ALPHAOVER` generates some shadows, which can be seen in the film-strip below. In `SCENE 1` (which is a normal Blender-scene) the titeling is done with the usual animation features from Blender, which makes that system a very flexible titler.

In the end a transition of strip in row 1 and the folowing in row 2 is done with the `CROSS` effect. The numbers 1-2 are representing the row numbers the effect applies to. The result is a smooth fade between the two strips in row one and two.



The mouse

LMB

The position of the mouse cursor becomes the current frame. If DisplayImage is ON, the Sequences at this position are read or calculated at this position.

MMB and drag

Depending on the position within the window:

- On the sliders; this can be moved.
- The rest of the window; the *view* is translated.

CTRL-MMB and drag

Zoom in/out on the IpoWindow. For ease of use, you can only zoom horizontally. Use the HeaderButton if you must zoom vertically as well.

RMB

Selection works the same way here as in the 3DWindow: normally a maximum of one Sequence strip is selected. Use **SHIFT** to extend or reduce the selection (*extend* select).

RMB and drag

Selects something and immediately starts *translation* mode, i.e. the Grabber.

SHIFT-RMB

Extend

selection.

The hotkeys

NUM+ , NUM-

Zoom in, zoom out.

SHIFT-NUM+

Insert gap. One second is inserted at the current frame position. This only applies to strips that are totally to the right of the current frame.

ALT-NUM+

Insert gap. As above, but now 10 seconds are inserted.

SHIFT-NUM-

Remove gaps. All strips that are completely to the right of the current frame and do *not* start past the last frame are repositioned so that there is no longer an empty space.

NUM.

The last selected strip is displayed completely.

HOME

All visible Sequences are displayed completely, centered in the window.

AKEY

Select All / deselect All. If any strip is selected, everything is first deselected.

SHIFT-A

Add sequence. A PopupMenu asks you to make a choice. The first three are possible sources:

- "Images"; Specify with FileSelect (with **RMB** select!), what pictures will form a strip. If only 1 picture is selected, the strip is lengthened to 50 frames. Directories can also be specified; each directory then becomes a separate strip in the SequenceWindow.
- "Movie"; Specify with FileSelect (with **LMB** or **RMB**!) what movie will comprise a strip.
- "Audio"; Specify with FileSelect (with **LMB** or **RMB**!) what WAV file will comprise a strip.
- "Scene"; A PopupMenu asks you to specify the Scene that is to be inserted as a strip. The Scene is then rendered according to its own settings and processed in the Sequence system.

The following menu options are *effects* that work on pictures; two strips must be selected for this. The order of selection determines how the effects are applied.

- "Plugin"; a File Window let you choose a sequence plugin.
- "Cross"; a fluent transition from strip 1 to strip 2.
- "GammaCross"; This is a gamma-corrected cross. It provides a more 'natural' transition in which lighter parts are inserted before darker parts.
- "Add"; two strips are added together.
- "Sub"; the second strip is subtracted from the first.
- "Mul"; the strips are multiplied.
- "AlphaOver"; the second strip, with its *alpha*, is placed over the first. pictures with *alpha* are normally 32 bits.

- "AlphaUnder"; the first strip is placed behind the second, with the *alpha* of the second strip.
- "AlphaOverDrop"; like "AlphaOver", but now with a drop shadow.

BKEY

Border select. Draw a rectangle with the LeftMouse; all strips that fall within this rectangle are *selected*. Draw a rectangle with the RightMouse to *deselect*.

CKEY

If one of the ends of a strip is selected (the triangles), the current frame is moved to this end. In all other cases, the Change menu is invoked. This menu allows you to change specific characteristics of the *active* strip.

If this is an Image strip:

- "Change Images". The FileSelect appears and new pictures can be specified.

If this is an Effect strip:

- "Switch a-b"; change the sequence of the effect.
- "Recalculate"; force a recalculation of the effect.
- "Cross, Gammacross, Add, ..."; change the type of effect.

If this is a Scene strip:

- "Update Start and End"; the start and end frame of the Scene is processed in the strip again.

ALT-D

Add Duplicate. All selected strips are copied. Immediately thereafter, *translation* mode is started. The images in an Image strip are reused; they take up no extra memory.

FKEY

"Set Filter". An extra Y filter can only be activated in Movie strips. This filter is for a stable video display with no flickering.

GKEY

Translation mode (the Grabber). This works on selected strips or the (triangular) ends of selected strips. Alternative for starting this mode: **RMB** and drag;. The following options are available in *translation* mode:

- Limitors: **SHIFT** with finer translation.
- **MMB** limits the current translation to the X or Y axis. Blender calculates which axis to use based on the already initiated mouse movement. Click MiddleMouse again to return to unlimited translation.

- **ARROWS**: These keys can be used to move the mouse cursor exactly 1 pixel.
- Grabber can be terminated with:
  - **LMB SPACE** or **ENTER**: move to the new position.
  - **RMB** or **ESC**: everything returns to the old position.

MKEY

Make Meta. The selected strips are combined into a Meta strip. This only occurs if no unselected strips are linked to the selection by effects. Use **TAB** to view the contents of a Meta or to leave the Meta. Metas can be inside other Metas, and behave exactly like a normal Sequence strip. When Metas are duplicated, their contents are *not* linked!

ALT-M

Un-Meta. The selected Meta is 'unpacked' again.

SKEY

Snap menu. The PopupMenu offers you a choice: "Sequence to frame"; the selected strips are placed with their starting point on the current frame.

XKEY

Delete Sequence. The selected strips are deleted.

The OopsWindow

Oops Toolbar

WindowType

As with every window header, the first button allows you to set the window type.

Full Window

Maximise the window, or return to the previous window display size; return to the previous screen setting (**CTRL-UPARROW**).

Home

All visible blocks are displayed completely, centered in the window (**HOME**).

View Zoom

Move the mouse to zoom in or out of the OopsWindow. This is an alternative to **CTRL-MMB**.

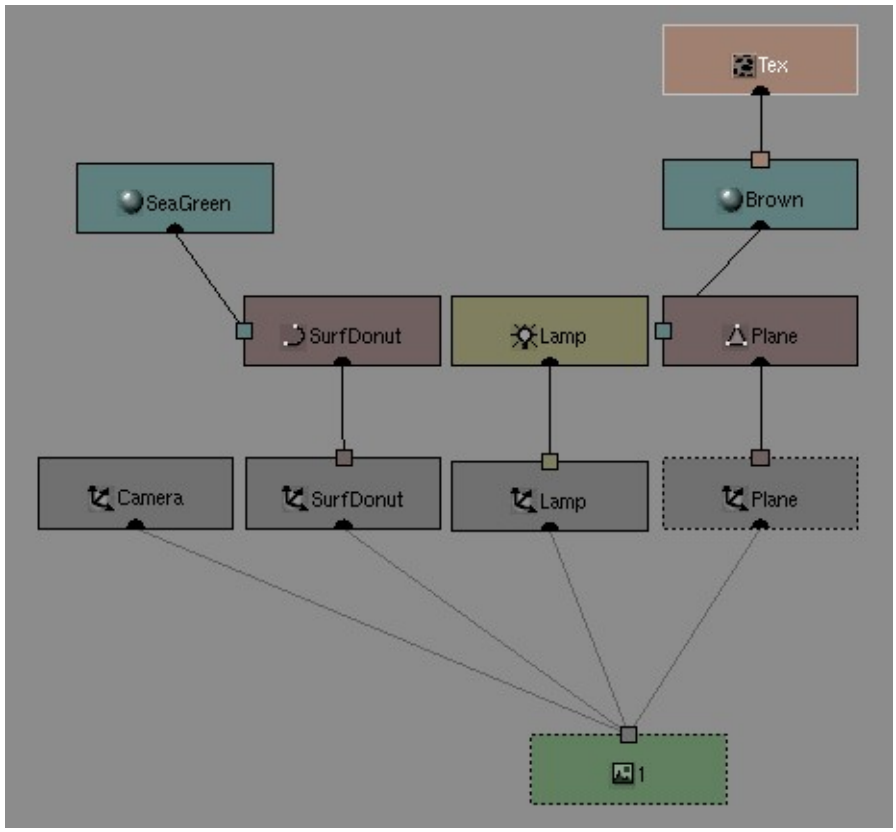
View Border

Draw a frame to indicate what part of the OopsWindow should be displayed in the full window.

Visible Select

This row of buttons determines what types of DataBlocks must be displayed. Relations between blocks are only shown if both blocks are visible. These are:

- Lay: the *layer* setting of the Scene determines what Objects are drawn.
- Scene: all Scenes present are displayed.
- Object: all Objects of all visible Scenes are displayed, possibly limited by the "Lay" option.
- Mesh
- Curve: this is also for Surface and Font blocks.
- MetaBall
- Lattice
- Lamp
- Material
- Texture
- Ipo
- Library



OopsWindow

The OopsWindow gives a schematic overview of the current structure. Blender is based on an Object-Oriented Programming System (OOPS!). The building blocks are universal DataBlocks that are assigned relationships using *links*. The different DataBlocks in the OopsWindow can be recognised by an icon and the colour. The DataBlocks have a clearly visible 'entrance' and 'exit' between which *link* lines are drawn. The current Scene and the *active* Object have a frame with a dashed line.

The functionality of the OopsWindow is limited to visualisation. Links are created using the available HotKeys (CTRL-L in the 3DWindow) and with the DataButtons in the Headers. Selected Objects are also selected in the OopsWindow, and vice versa.

In the accompanying example, we see the Scene at the bottom, with four linked Object blocks. The Object blocks have links to the specific ObData; such as Mesh, Surface, Lamp. The Materials ("Brown" and "SeaGreen") are linked to the ObData and the Texture is linked to a Material.

The mouse

LMB and drag

These are the Gestures. Blender's gesture recognition works in two ways here:

- Draw a straight line: start *translation* mode.
- Draw a V-shaped line: start *scaling* mode.

MMB and drag

The *view* is translated.

CTRL-MMB and drag

Zoom in or out of the OopsWindow.

RMB

Select works here in the normal fashion: normally a maximum of one DataBlock is selected. Use SHIFT to enlarge or reduce the selection (*extend* select).

RMB and drag

Select and start *translation* mode, the Grabber.

SHIFT-RMB

Add/remove from selection.

CTRL-RMB

This selects and *activates* a DataBlock. This only works for Scenes and Objects.

The HotKeys

NUM+ NUM-

Zoom in, zoom out.

HOME

All visible blocks are displayed completely, centred in the window.

1KEY, 2KEY...0KEY

The visible *layers* of the current Scene can be set. e **SHIFT** for *extend* select.

AKEY

Select All / deselect All. If one block is selected, everything is first eselected.

BKEY

Border select. Draw a rectangle with the **LMB**; all blocks that ll within this rectangle are *selected*. Draw a ctangle with the **RMB** to *deselect* the blocks.

GKEY

Translation mode (the Grabber). This works on selected blocks. Alternatives for starting this mode:

- **RMB** and drag
- **LMB** and drag to draw a straight line.

The following options are available in *translation* mode:

- **MMB** restricts the current translation to the X or Y axis. Blender calculates which axis to use based on the already initiated mouse movement. Click **MMB** again to restore unlimited translation.
- **ARROWS**: The mouse cursor can be moved exactly 1 pixel with these keys.
- Grabber terminates with:
  - **LMB SPACE ENTER**: move to a new position.
  - **RMB** or **ESC**: everything returns to the old position.

LKEY

Select Linked Forward. All DataBlocks that are linked by a selected DataBlock are also selected. In this way, the entire underlying structure can be selected, starting with a selected Scene block.

SHIFT-L

Select Linked Backward. All *users* of selected DataBlocks are selected. This allows you to visualise what Objects the Material uses, starting with a selected Material block.

SKEY

Scaling mode. This works on selected blocks. Only the length of the links, i.e. the distance between the DataBalocks, can be transformed. The degree of *scaling* corresponds *exactly* to the mouse movement. Try to move the (rotation) midpoint with the mouse. Alternatives for starting *scaling* mode: - **LMB** and drag to draw a sharp angle; a V-shaped line. The following options are available in *scaling* mode:

- **MMB** restricts the *scaling* to the X or Y axis. Blender calculates which axis to use based on the already initiated mouse movement. Click **MMB** again to return to free *scaling*.
- **ARROWS**: These keys move the mouse cursor exactly 1 pixel.
- Exit size mode with:
  - **LMB SPACE** or **ENTER**: to finalize scaling.
  - **RMB** or **ESC**: everything returns to its previous state.

SHIFT-S

Shuffle Oops. An attempt is made to minimise the length of the *link* lines for selected DataBlocks using parsed toggling.

ALT-S

Shrink Oops. The length of the *link* lines for the selected DataBlocks is shortened without causing the blocks to overlap.

The Action Window

Action Toolbar

WindowType

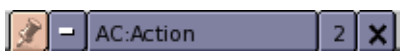
As with every window header, the first button allows you to set the window type.

Full Window

Maximise the window, or return to the previous window display size; return to the previous screen setting (**CTRL-UPARROW**).

Home

All visible blocks are displayed completely, centered in the window (**HOME**).



Pin

Causes this Action window to show the current data block regardless of what object is selected or active.

Action Browse

Choose another Action from the list of available Actions. The option "Add New" makes a complete copy of the current Action. This is not visible; only the name in the adjacent button will change.

AC:

Give the current Action a new and unique name. After the new name is entered, it appears in the list, sorted alphabetically.

Users

If this button is displayed, there is more than one user for the Action block. Use the button to make the Action "Single User".

Unlink Action

The current Action is unlinked.

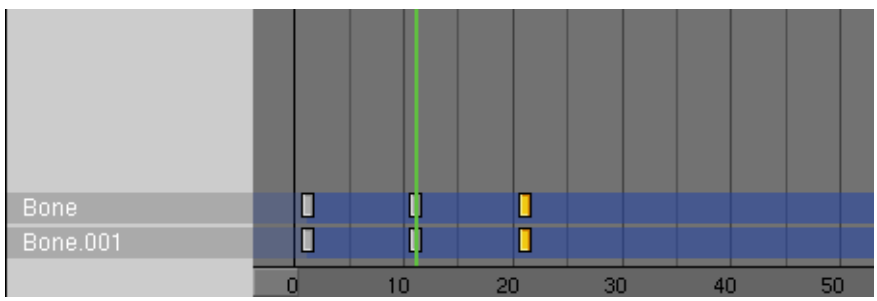
Bake

Generate an action based of the current action where the constraints effects are converted into IPO Keys.

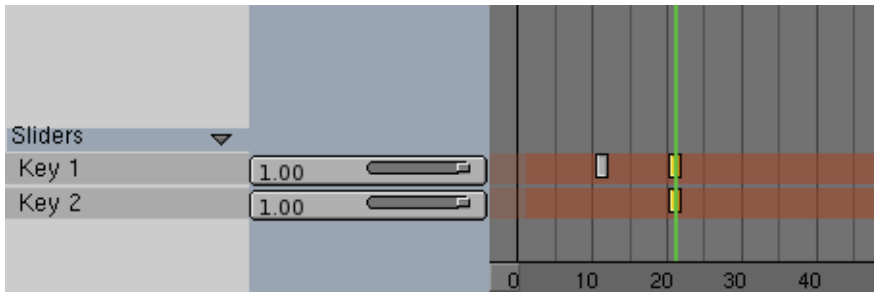
Lock

This buton locks the update of the 3DWindow while editing in the ActionWindow, so you can see changes maked to the Action in realtime in the 3DWindow.

ActionWindow



The ActionWindow gives an overview of the current Armature Keys and Relative Vertex Keys of the Object. It presents the time, in frames, on an horizontal axis and, vertically, as many stripes are there are bones in the armature or Relative Vertex Keys.



If the object is an armature bone Keys are represented as grey rectangle at the pertinent frame, yellow if selected. If it is a Mesh, Rectangles are present where a Key is assigned a given value. THis can be assigned via IPO or via the sliders in the Action-Window.

The mouse

LMB

Sets current frame

MMB and drag

The *view* is translated.

CTRL-MMB and drag

Zoom in or out of the ActioWindow, this happens only horizontally.

RMB

Select a strip (if clicked on strip name) or a Key if clicked on a Key. Use SHIFT to enlarge or reduce the selection (*extend* select).

SHIFT-RMB

Add/remove from selection.

The HotKeys

HOME

All visible blocks are displayed completely, centred in the window.

AKEY

Select All / deselect All. If one block is selected, everything is first eselected.

BKEY

Border select. Draw a rectangle with the **LMB**; all blocks that fall within this rectangle are *selected*. Draw a ctangle with the **RMB** to *deselect* the blocks.

CKEY

Centers view at current frame.

SHIFT-D

Duplicates the selected Keys. Duplicates are automatically in Grab mode.

GKEY

Translation mode (the Grabber). This works on selected blocks and only horizontally, to change frame. The following options are available in *translation* mode:

- **ARROWS**: The mouse cursor can be moved exactly 1 pixel with these keys.
- **CTRL**: The Keys are displaced by 1 frame steps.
- **SHIFT-CTRL**: The Keys are displaced by 0.1 frame steps.
- Grabber terminates with:
  - **LMB SPACE ENTER**: move to a new position.
  - **RMB** or **ESC**: everything returns to the old position.

TKEY

Allows to define the type of interpolation for the selected strips:

- "Constant" is picewise constant (abrupt changes).
- "Linear" is linear interpolation (abrupt changes in derivative).
- "Bezier" is default fluid interpolation.

XKEY

Delete selected Keys.

The Non Linear Animation Window

NLA Toolbar

WindowType

As with every window header, the first button allows you to set the window type.

Full Window

Maximise the window, or return to the previous window display size; return to the previous screen setting (**CTRL-UPARROW**).

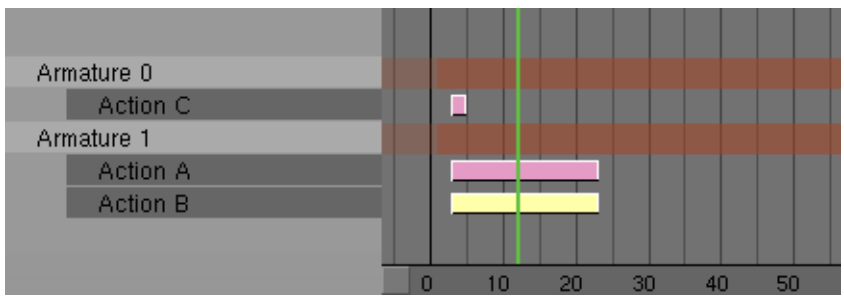
Home

All visible blocks are displayed completely, centered in the window (**HOME**).

Lock

This buton locks the update of the 3DWindow while editing in the ActionWindow, so you can see changes maked to the Action in realtime in the 3DWindow.

NLAWindow



The NLAWindow gives an full overview of all the Armature Objects and allows a very fine and flexible control of each armature action, effectively allowing for action blending somewhat akin to how Relative Vertex Keys work on meshes.

The NLA window presents the time, in frames, on the horizontal axis and one strip for each armature, each armature strip can have as many action substrips as desired. It is important to have unlinked any normal Action fromthe armature when working with NLA because normal actions take precedence over NLA controls.

The mouse

LMB

Sets current frame

MMB and drag

The *view* is translated.

CTRL-MMB and drag

Zoom in or out of the NLAWindow, this happens only horizontally.

RMB

Select an Armature Strip (if clicked on strip name) or an Action Strip if clicked on it. Use SHIFT to enlarge or reduce the selection (*extend* select).

SHIFT-RMB

Add/remove from selection.

The HotKeys

HOME

All visible blocks are displayed completely, centred in the window.

AKEY

Select All / deselect All. If one block is selected, everything is first eselected. It behaves differently depending if the cursor is on the left (selects all armatures) or on the right (selects all actions) of the NLAWindow.

BKEY

Border select. Draw a rectangle with the **LMB**; all blocks that fall within this rectangle are *selected*. Draw a ctangle with the **RMB** to *deselect* the blocks.

SHIFT-D

Duplicates the selected Action(s). Duplicates are automatically in Grab mode and are assigned to new sub-strips.

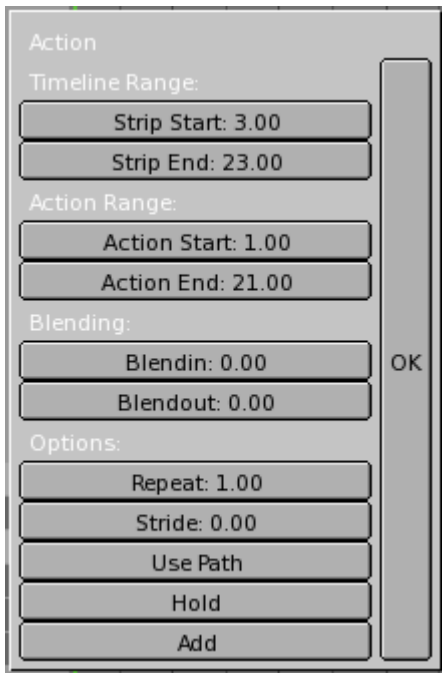
GKEY

Translation mode (the Grabber). This works on selected Actions and only horizontally, to change frame. The following options are available in *translation* mode:

- **ARROWS**: The mouse cursor can be moved exactly 1 pixel with these keys.
- **CTRL**: The Keys are displaced by 1 frame steps.
- **SHIFT-CTRL**: The Keys are displaced by 0.1 frame steps.
- Grabber terminates with:
  - **LMB SPACE ENTER**: move to a new position.
  - **RMB** or **ESC**: everything returns to the old position.

NKEY

Brings up the "Numerical" window settings for the selected Action.



- "Strip Start" and "Strip End" defines the Action Strip placement. If the interval is greater than the actual Action duration the Action is performed slower to match the required duration, otherwise faster.
- "Action Start" and "Action End" defines the Action timeline "Windowing". The Actions are defined in their normal way and their duration is, by default, fa "Window" of frames going from first to last key. With this sliders it is possible to vary the Action "Windowing".
- "BlendIn" and "BlendOut" defines a number of frames at the begginning and at the end of the strip of "Reduced influence" of the Action. By carefully setting these and by letting action stripls slightly overlap you can blend fluidly different actions.
- "Repeat" Makes the strip contain as many copies of the action as desired. Great for Walkcycles.
- "Stride" in Walkcycles defines the length (in Blender Units) of a stride.
- "Use Path" Makes Blender use the Path to which the armature is parented, and its length, to make the Armature move according to Stride definition.
- "Hold" Makes the last pose to be held forever, instead than reverting to original state.
- "Add" MakesBlending addictive.

TKEY

Allows to define the type of interpolation for the selected strips:

- "Constant" is picewise constant (abrupt changes).

- "Linear" is linear interpolation (abrupt changes in derivative).
- "Bezier" is default fluid interpolation.

XKEY

Delete selected Keys.

The Text Window

Text Toolbar

WindowType

As with every window header, the first button allows you to set the window type.

Full Window

Maximise the window, or return to the previous window display size; return to the previous screen setting (**CTRL-UPARROW**).

Line Numbers

This button toggles showing of the line numbers on and off.

Text Browse

Choose another Text from the list of available Buffers. The option "Add New" opens a new empty buffer. The option "Open New" turns the Text Window into a FIle Selection Window an allows to load a Text Buffer from the disk.

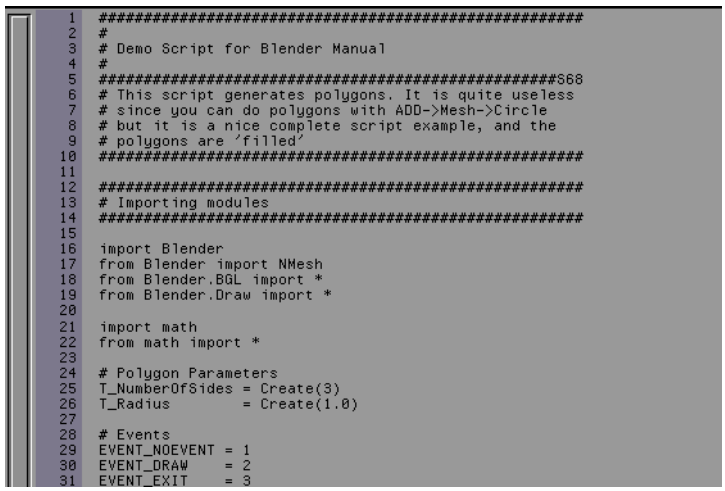
TX:

Give the current Text buffer a new and unique name. After the new name is entered, it appears in the list, sorted alphabetically.

Unlink Text

The current Text Buffer is unlinked.

TextWindow



```
 1  ##################################################
 2  #
 3  # Demo Script for Blender Manual
 4  #
 5  ##################################################S68
 6  # This script generates polygons. It is quite useless
 7  # since you can do polygons with ADD->Mesh->Circle
 8  # but it is a nice complete script example, and the
 9  # polygons are 'filled'
10  ##################################################
11
12  ##################################################
13  # Importing modules
14  ##################################################
15
16  import Blender
17  from Blender import NMesh
18  from Blender.BGL import *
19  from Blender.Draw import *
20
21  import math
22  from math import *
23
24  # Polygon Parameters
25  T_NumberOfSides = Create(3)
26  T_Radius        = Create(1.0)
27
28  # Events
29  EVENT_NOEVENT = 1
30  EVENT_DRAW    = 2
31  EVENT_EXIT    = 3
```

The TextWindow is a simple but useful Texteditor, fully integrated into Blender. The main purpose of it is to write Python scripts, but it is also very useful to write comments in the Blendfile or to instruct other users the purpose of the scene.

The mouse

LMB

Sets the cursor position, defines a selection.

SHIFT-LMB

Adds/remove from selection.

MMB

Pan / translates window.

RMB

Opens a menu:

- "New" - Creates a new empty buffer.
- "Open" - Turns window in File Selection Window for loading a text buffer fromdisk.
- "Save" - Save text buffer to disc.
- "Save As" - Turns window in File Selection Window for saving the current text buffer to disc.

The HotKeys

ALT-C or CTRL-C

Copy the marked text into a temporary buffer

SHIFT-ALT-F

Opens the same menu as **RMB**

ALT-J

Asks for a line number and makes the cursor jump to it.

ALT-M

Converts the text in the buffer into a 3D text object (Max 1000 chars.).

ALT-O

Opens a Text buffer.

ALT-P

Executes the Text as a Python script.

ALT-R

Redo.

ALT-S

Saves the Text buffer.

ALT-U

Undo.

ALT-V or CTRL-V

Paste the marked text from the temporary buffer

ALT-X or CTRL-X

Cut the marked text into a temporary buffer

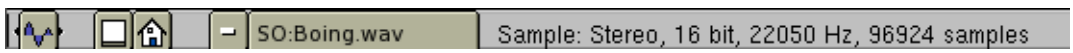
Windows

Blender's temporarybuffer is separated from Window's clipboard. To access Window's clipboard use **SHIFT-CTRL-C**, **SHIFT-CTRL-V**, **SHIFT-CTRL-X**

The SoundWindow

The SoundWindow is currently most useful for the realtime part of Blender, which is not covered by this manual.

It is used to load and visualize sounds. You can grab and zoom the window like every other window in Blender.



SoundHeader

WindowType

As with every window header, the first button allows you to set the type of window.

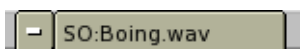
Full Window

Maximise the window, or return to the previous window display size; return to the previous screen setting (**CTRL-UPARROW**).

Home

All visible Sequences are completely displayed, centered in the window (**HOME**).

SoundBrowse



Choose another Audio Stream from the list of available ones. The option "Add New" opend a File selection window to open a new AUdio file.

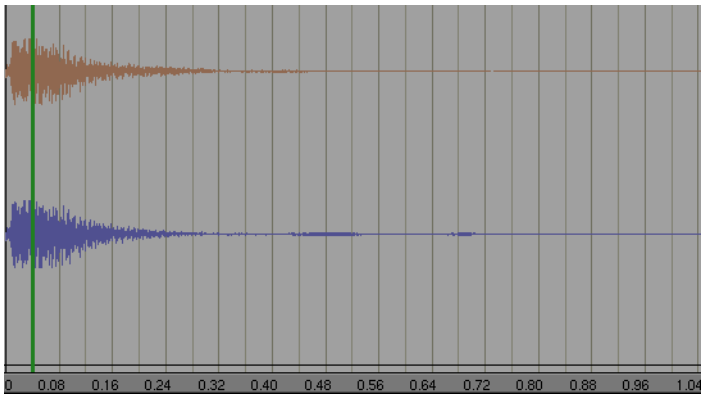
SO:

Give the current Audio Stream a new and unique name. After the new name is entered, it appears in the list, sorted alphabetically.

InfoText

Provides some info about the currently activeAudio Stream.

The Audio Window



The sound Audio Window represents the wave shape. Differently by allother Blender Time windows the time scale is here in seconds, not frames.

The ImageWindow

ImageHeader

WindowType

As with every window header, the first button allows you to set the type of window.

Full Window

Maximise the window, or return to the previous window display size; return to the previous screen setting (**CTRL-UPARROW**).

Home

The picture is displayed completely, centred in the window (**HOME**).

Square UV-polygons

This option keeps the UV polygons square while UV-texturing

Clip UV with imagesize

Limits the UV-polygons to the imagesize

The Image blocks can be specified using the DataButtons.

Image Browse

Select an Image from the list provided.

IM:

Give the current Image a new and unique name. After the new name is entered, it is displayed in the list alphabetically.

Unlink

Unlinks the curent image.

Pack

Packs the current image within the .blend file.

Load

Load a new Image. A Image Select Windows appears. The labelless button nearby let you specify the imagevia a standard File Select Window.

Replace

Replaces the current image wit a new one. A Image Select Windows appears. The labelless button nearby let you specify the imagevia a standard File Select Window.

RefMap

Uses the current image as a Reflection Map, ignoring UV coordinates.

Tile

Sets the image to tile mode. This way you can make a repeating pattern with a small part of an image. With **SHIFT-LeftMouse** you indicate which part of the image should be used.

PartsX and PartsY

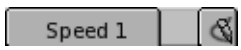
Defines the dimension of the tile mode.

Anim

Enables the texture-animation.

Animation start and end

Controls the start and end of the texture-animation.



Speed

Sets the speed of the animation, in frames per second.

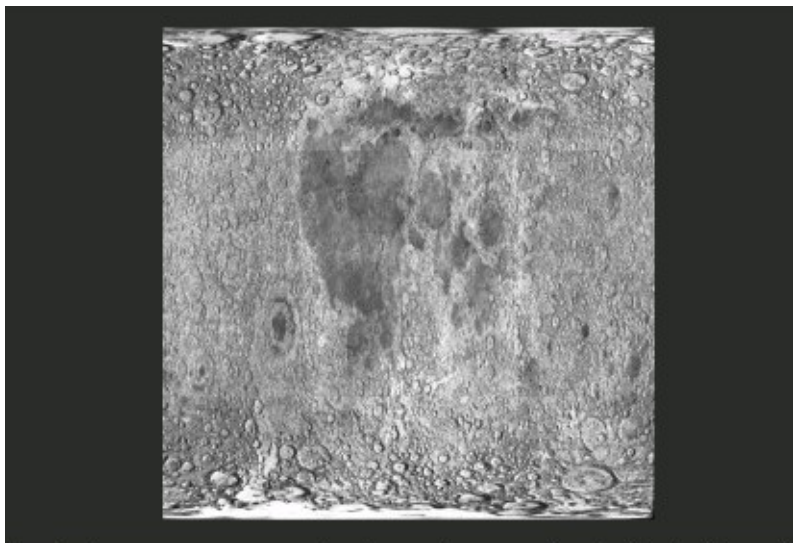
Texture Paint

Enables Texture Paint mode.

Lock

When activated, changes made in the ImageWindow are shown in realtime in the 3DWindows.

ImageWindow



Images in Blender are also DataBlocks. The ImageWindow is used for visualisation and UV-texturing.

The use of the mouse and HotKeys is:

MMB

Translate the *view*.

NUM+ NUM- MW

Zoom in, zoom out.

HOME

The picture is displayed completely, centred in the window.

CTRL-N

Replace Image Names. A menu asks you to enter an old and a new file name. All file names for Images with the old name or a name which starts with corresponding characters are replaced by the new name. This utlility is especially useful for changing directories. Example: "old" = /data/, "new" = /pics/textures/. The file name "/data/marble.tga" is replaced by "/pics/textures/marble.tga".

The ImageSelectWindow

ImageSelectHeader

WindowType

As with every window header, the first button allows you to set the type of window.

Full Window

Maximise the window, or return to the previous window display size; return to the old screen setting (**CTRL-UPARROW**).

Remove

Delete the ".Bpib" help file in the current directory. A new ".Bpib" is only created once the directory is read again.

Dirview

Indicates whether the left part, where the directories are displayed, is shown.

Info

Indicates whether the lower part, where information about the active picture is displayed, is shown.

Images

Obsolete.

Magnify

The active picture is displayed twice as large.

ImageSelectWindow

In parts of the Blender interface where pictures can be loaded, you generally have the option of using a FileSelect window or the ImageSelect window. For the most part, the functionality is the same. The ImageSelect window reads the directory and examines each file to see if it is a recognisable image. After the entire directory is scanned, the images are loaded and displayed as a thumbnail and saved in the ".Bpib" file. If a ".Bpib" file already exists, it is first read in and compared to the contents of the directory.

P

Displays the parent directory (**PKEY**).

DirName:

This text box displays the current directory.

Preset Directories

The file $HOME/.Bfs contains a number of pre-sets that are shown in this menu. While a file is being read or written, the directory involved is temporarily added to the menu.

FileName:

The file name can be entered here.

Status Icons.

The different phases of ImageSelect:

- Was a ".Bpib" file found?
- Was the directory scanned completely?
- Have all the pictures been read in?

The mouse and HotKeys

LMB

Activate a file. The file name is placed in the FileName button.

MMB

Activate a file and return to the previous window type.

RMB

Select a file.

ENTER

Close the ImageSelectWindow; return with a OK message.

ESC

Close the ImageSelectWindow; no action is performed.

PAGEDN

Scroll down one page.

PAGEUP

Scroll up one page.

PKEY

Go to the parent directory.

The Animation playback window

To allow you to view sequences of rendered frames or AVIs, Blender has a simple, but efficient animation playback option. This playback is invoked with the "PLAY" button in the DisplayButtons. This button plays all of the numbered frames specified in the DisplayButtons->pics TextBut.

An alternative for starting the animation window is to type -a in the command line: blender -a . Blender first reads all the files in memory and then displays them as a flip book. Check in advance to make sure sufficient memory is available; you can see this with the FileWindow. Use **ESC** to stop the reading process.

The commands available in the playback window are:

ESC

Close the window.

ENTER

Start playback.

LEFTARROW , DOWNARROW

Stops the playback; if playback is already stopped, moves 1 frame back.

RIGHTARROW , UPARROW

Stops the playback; if playback is already stopped, moves 1 frame forward.

NUM0

Sets the playback at the first frame and switches 'cyclical' playback off. Pressing this key again turns cyclical playback on again and starts the playback at the beginning.

PAD1 to PAD9

The playback speed. 60, 50, 30, 25, 20, 15, 12, 10 and 6 frames per second, respectively.

LMB

Move the mouse horizontally through the playback window to scroll through the frames.

Chapter 29. Buttons Reference

The ButtonsWindow

ButtonsHeader

WindowType

As with every window header, the first button enables you to set the window type.

Full Window

Maximise the window, or return to the previous window display size; return to the old screen setting. HotKey: (ALT-)CTRL+UPARROW

Home

The optimal *view* settings for the current window are restored. HotKey: HOMEKEY. The following series of buttons determine the type of ButtonsWindow. In order, they are:

ViewButtons

The 3DWindow settings for the window.

LampButtons

The settings of the *active* Lamp Object. HotKey: F4.

MaterialButtons

The settings of the *active* Material linked by the *active* Object. HotKey F5.

TextureButtons

The settings of the *active* Texture. This can be the Texture for a Material, Lamp or World. HotKey: F6.

AnimButtons

The (animation) settings of the *active* Object, including Particle Effects. HotKey: F7.

RealTimeButtons

Settings for real time 3D content. HotKey: **F8**

WorldButtons

The settings of the World linked by the current Scene.

EditButtons

The settings and EditMode options of the ObData linked by the *active* Object. HotKey: F9.

Face/PaintButtons

The VertexPaint and FaceSelect options. Only for an *active* Mesh Object.

RadiosityButtons

The Radiosity options.

ScriptButtons

Buttons to assign scripts.

DisplayButtons

The settings of the Scene. HotKey: F10. Later sections deal with each button in detail.

This group of DataButtons is drawn based on the type of ButtonsWindow and the availability of the data to be visualised. The header for the DisplayButtons is shown here as an example.

Scene Browse

Choose a different scene from the list of available scenes.

SCE:

Give the current Scene a new and unique name.

Delete Scene

"OK? Delete Current Scene"

Current Frame.

The current frame number is displayed as a fixed part of the ButtonsHeader.

ButtonsWindow

A buttonsWindow offers a number of extra facilities:

MiddleMouse

If space is available, this can be used to horizontally or vertically move the *view*.

CTRL+MiddleMouse

Within certain limits, a ButtonsWindow can be enlarged or reduced.

NUM+

Zoom in.

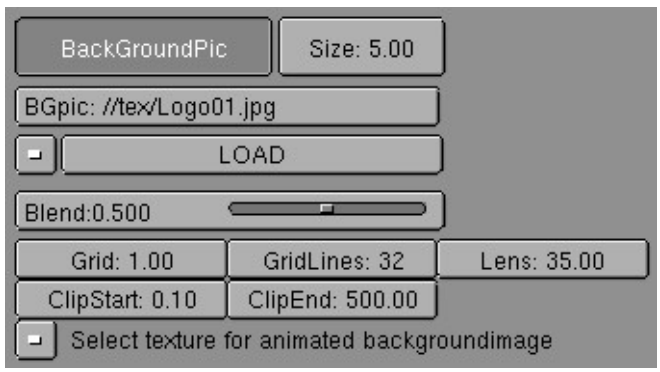
NUM-

Zoom out.

HOMEKEY

The optimal *view* settings for the current window are restored. If there is only one 3Dwindow in the current Screen, the NumPad commands for the 3DWindow also work in the ButtonsWindow.

The ViewButtons



These buttons are not global; independent variables can be set for each 3Dwindow independently, for the *grid* or the *lens*, for example. Use the HotKey SHIFT+F7 in the 3DWindow or the WindowType button in the 3DHeader.

Use SHIFT+F5 to restore the 3DWindow.

BackGroundPic

This option displays a picture in the background of the 3DWindow. Standard Image blocks are used; re-using an Image does not consume any additional memory. The BackGroundPic is only drawn in *ortho* and Camera view. It is always centered around the global nulpoint. In camera View, it is completely displayed in the viewport.

Size:

The size in Blender units for the width of the BackGroundPic. Only of interest in *ortho*.

ImageBrowse

Select an existing Image from the list provided.

LOAD

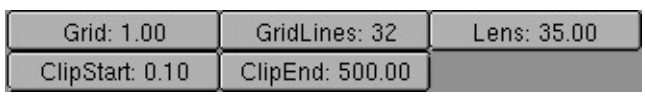
The window changes into an ImageSelectWindow. Use this to specify the picture to be used as the BackGroundPic. The picture is added to the Blender structure as an Image block.

Blend

The factor with which the grey of the 3DWindow is blended with the background picture.

TextureBrowse

Specify a Texture to be used as the BackGroundPic. This only works for Image Textures. The TextureButtons have extensive options for an animated Image Texture, which allows you to achieve an animated BackGroundPic. Use this option for *rotoscoping*, for example. This is a technique in which video images are used as examples or as a basis for 3D animation.



Grid:

The distance between two grid lines. This value is also used to define minimum and maximum values for several buttons in Blender.

GridLines:

The number of lines that comprise the grid in *perspective* or Camera view. A value of zero means no grid at all.

Lens:

The 'lens' used for the *perspective* view. This is independent of the Camera.

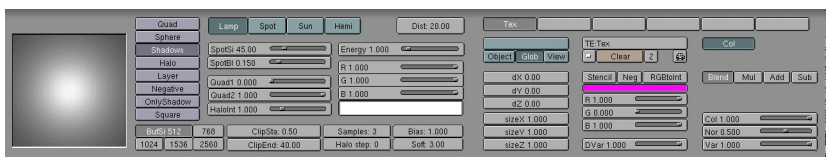
ClipStart:

The start clipping value in *perspective* view mode.

ClipEnd:

The distance beyond which items are no longer displayed in *perspective* view mode. The ClipStart and ClipEnd limits also determine the resolution (actually the density) of the OpenGL zbuffer. Try to keep these values as close together as possible, so that the zbuffer can distinguish small differences.

LampButtons



The settings in these ButtonsWindow visualise the Lamp DataBlock. The LampButtons are only displayed if the *active* Object is a Lamp. HotKey for LampButtons: F4.

The DataButtons in the Header indicate what Lamp block is visualised.

Lamp Browse

Choose another Lamp block from the list provided.

LA:

Give the current Lamp a new and unique name.

Users

If the Lamp Block is used by more than one Object, this button shows the total number of Objects. Press the button to make the Lamp "Single User". Then an exact copy is made.

Lamp options

Quad

The distance from the lamp is in inverse quadratic proportion to the intensity of the light. An inverse linear progression is standard (see also the buttons "Dist", "Quad1" and "Quad2").

Sphere

The lamp only sheds light within a spherical area around the lamp. The radius of the sphere is determined by the "Dist" button.

Shadows

The lamp can produce shadows. Shadow calculations are only possible with the Spot lamps. The render option "Shadows" must also be turned ON in the DisplayButtons. See also the *shadowbuffer* buttons later in this section.

Halo

The lamp has a halo. This only works with Spot lamps. The intensity of the halo is calculated using a conic section. With the option "Halo step:" it also uses the shadow buffer (volumetric rendering). The scope of the spot halo is determined by the value of "Dist".

Layer

Only Objects in the same layer(s) as the Lamp Object are illuminated. This enables you to use selective lighting, to give objects an extra accent or to restrict the effects of the lamp to a particular space. It also allows to you keep rendering times under control.

Negative

A lamp casts ’negative’ light.

OnlyShadow

For spot lamps (with "Shadow" ON), only the shadow is rendered. Light calculations are not performed and where there are shadows, the value of "Energy" is reduced.

Square

Spotlamps can have square Spotbundles with this option. For a better control over shadows and for slide projector effects.

Lamp types

Lamp

The standard lamp, a point light source.

Spot

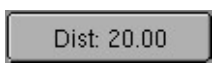
The lamp is restricted to a conical space. The 3DWindow shows the form of the spot-light with a broken line. Use the sliders "SpotSi" and "SpotBl" to set the angle and the intensity of the beam.

Sun

The light shines from a constant direction; the distance has no effect. The position of the Lamp Object is thus unimportant, except for the rotation.

Hemi

Like "Sun", but now light is shed in the form of half a sphere, a *hemisphere*. This method is also called *directionalambient*. It can be used to suggest cloudy daylight.



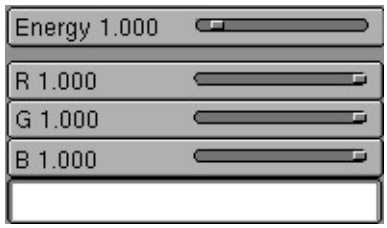
Dist

For the lamp types "Lamp" and "Spot", the distance affects the intensity of the light. The standard formula is used for this:

D = "Dist" button, a = distance to the lamp. Light intensity = D/(D + a).

This is an inverse linear progression. With the option "Quad", this becomes:

Light intensity = D/(D + a*a).



Energy

The intensity of the light. The standard settings in Blender assume that a minimum of two lamps are used.

R, G, B

The red, green and blue components of the light.

SpotSi

The angle of the beam measured in degrees. Use for shadow lamp beams of less than 160 degrees.

SpotBl

The softness of the spot edge.

Quad1, Quad2

The light intensity formula of a Quad Lamp is actually: Light intensity = D / (D + (quad1 * a) + (quad2 * a * a)) D = "Dist" button. a = distance to the lamp. The values of "quad1" and "quad2" at 1.0 produces the strongest quadratic progression. The values of "quad1" and "quad2" at 0.0 creates a special Quad lamp that is insensitive to distance.

HaloInt

The intensity of the spot halo. The scope of the spot halo is determined by "Dist".

Shadow Buffer

Blender uses a *shadowbuffer* algorithm. From the spotlight, a picture is rendered for which the distance from the spotlight is saved for each pixel. The shadow buffers are compressed, a buffer of 1024x1024 pixels requires, on average, only 1.5 Mb of memory.

This method works quite quickly, but must be adjusted carefully. There are two possible side effects:

- *Aliasing* . The shadow edge has a block-like progression. Make the spot beam smaller, enlarge the buffer or increase the number of *samples* in the buffer.
- *Biasing* . Faces that are in full light show *banding* with a block-like pattern. Set the "Bias" as high as possible and reduce the distance between "ClipSta" and "ClipEnd".

Bufsi 512, 768, 1024, 1536, 2560 (RowBut)

The size of the buffer in pixels. The value of DisplayButtons->Percentage (100%, 75%, ...) is multiplied by this.

ClipSta, ClipEnd

Seen from the spot lamp: everything closer than ClipSta always has light; everything farther away than ClipEnd always has shadow. Within these limits, shadows are calculated. The smaller the shadow area, the clearer the distinction the lamp buffer can make between small distances, and the fewer side effects you will have. It is particularly important to set the value of ClipSta as high as possible.

Samples

The shadow buffer is 'sampled'; within a square area a test is made for shadow 3*3, 4*4 or 5*5 times. This reduces *aliasing*.

Halo step

A value other than zero in the button "Halo step" causes the use of the shadow detection (volumetric rendering) for Halos. Low values cause better results and longer rendering times. A value of "8" works fine in most cases.

Soft

The size of the sample area. A large "Soft" value produces broader shadow edges.

Texture Channels

Texture name

A Lamp has six *channels* with which Textures can be linked. Each *channel* has its own *mapping*, i.e. the manner in which the texture works on the lamp. The settings are in the buttons described below.

Mapping: coordinates input.

Each Texture has a 3D coordinate (the texture coordinate) as input. The starting point is always the global coordinate of the 3D point that is seen in the pixel to be rendered. A lamp has three options for this.

Object Name

The name of the Object that is used for the texture coordinates. If the Object does not exist, the button remains empty.

Object

Each Object in Blender can be used as a source for texture coordinates. To do this, an inverse transformation is applied to the global coordinate, which gives the *local* Object coordinate. In this way, the texture is linked with the position, size and rotation of the Object.

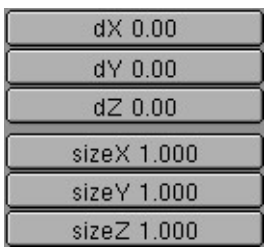
Glob

The global coordinate is passed on to the texture.

View

The *view* vector of the lamp; the vector of the global coordinate to the lamp, is passed on to the texture. If the lamp is a Spot, the *view* vector is normalised to the dimension of the spot beam, allowing use of a Spot to project a 'slide'.

Mapping: transform coordinates.



Use these buttons to adjust the texture coordinate more finely.

dX, dY, dZ

The extra translation of the texture coordinate.

sizeX, sizeY, sizeZ

The extra scaling of the texture coordinate.

Texture Block

TE:

The name of the Texture block. The name can be changed with this button.

Texture Browse

Select an existing Texture from the list provided, or create a new Texture Block.

Clear

The link to the Texture is erased.

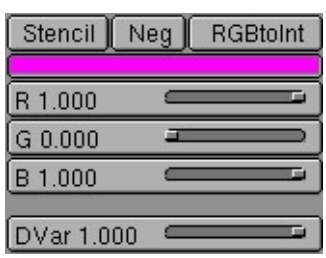
Users

If the Texture Block has multiple users, this button shows the total number of users. Press the button to make the Texture "Single User". Then an exact copy is made.

Auto Name

Blender assigns a name to the Texture.

Mapping: Texture input settings.



These buttons pass extra information to the Texture.

Stencil

Normally, textures are executed one after the other and placed over each other. A second Texture *channel* can completely replace the first. This option sets the *mapping* to *stencil* mode. No subsequent Texture can have an effect on the area the current Texture affects.

Neg

The inverse of the Texture is applied.

RGBtoInt

With this option, an RGB texture (affects colour) is used as an Intensity texture (affects a value).

R, G, B

The colour with which an Intensity texture blends with the current colour.

DVar

The value with which the Intensity texture blends with the current value.

Mapping: output to.

Col

The texture affects the colour of the lamp.

Mapping: output settings.

These buttons adjust the output of the Texture.

Blend

The Texture mixes the values.

Mul

The Texture multiplies the values.

Add

The Texture adds the values.

Sub

The Texture subtracts the values.

Col

The extent to which the texture affects the colour.

Nor

The extent to which the texture affects the normal (not important here).

Var

The extent to which the texture affects the value (a variable, not important here).

Material Buttons

The settings in this ButtonsWindow visualise the Material DataBlock. The Material-Buttons are only displayed if the *active* Object has a Material. Hotkey: F5.

The DataButtons in the Header indicate what Material block is visualised.

Material Browse

Select another Material from the list provided, or create a new block.

MA:

Give the current Material a new and unique name.

Users

If the Material block is used by more than one Object, this button indicates the total number of users. Press the button to make the Material "Single User". An exact copy is created.

Remove Link

Delete the link to the Material.

Auto Name

Blender assigns a name to the Material.

Copy to buffer

The complete contents of the Material and all the *mapping* is copied to a temporary buffer.

Copy from buffer

The temporary buffer is copied to the Material.

Preview settings.

Plane

The preview plane only shows the X-Y coordinates.

Sphere

In the sphere-preview the Z axis is the vertical axis for the preview sphere; the X and Y axes revolve around this axis.

Cube

The cubic preview shows the material preview mapped on three sides of a cube, allowing to see the three possible mappings.

Background

Use this button to select a light or a dark background.

Refresh

Use this button to refresh the material-preview. This is mostly needed after changing frames while having a material-Ipo.

These buttons specify what the Material block is linked to, or must be linked to. By linking Materials directly to Objects, each Object is rendered in its own Material.

ME:

This Button indicates the block to which the Material is linked. This button can only be used to give the block another name. Possible blocks are:

- ME: Material is linked to a Mesh (ObData) block.
- CU: Material is linked to a Curve, Surface or Font (ObData) block.
- MB: Material is linked to a MetaBall (ObData) block.
- OB: Material is linked to the Object itself.

OB

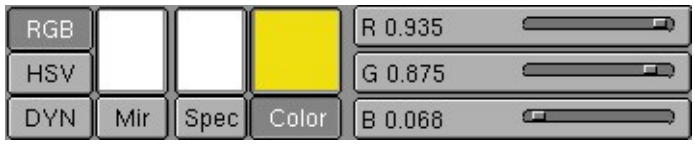
Use this button to link the current Material to the *Object*. Any link to the ObData block remains in effect. Links can be removed with the Header button:

"Remove Link" ME or CU or MB

Use this button to link the current Material to the *ObData* of the Object. Any link to the Object block remains in effect. Links can be removed with the Header button: "Remove Link"

1 Mat 1

An Object or ObData block may have more than one Material. This button can be used to specify which of the Materials must be displayed, i.e. which Material is *active*. The first digit indicates how many Materials there are; the second digit indicates the number of the *active* Material. Each face in a Mesh has a corresponding number: the 'Material index'. The number of *indices* can be specified with the EditButtons. Curves and Surfaces also have Material indices.



RGB

Most colour sliders in Blender have two pre-set options: in this case, the colour is created by mixing Red, Green, Blue.

HSV

The colour sliders mix colour with the Hue, Saturation, Value system. 'Hue' determines the colour, 'Saturation' determines the amount of colour in relation to grey and 'Value' determines the light intensity of the colour.

DYN

Adjust parameters for the dynamics optons. The following buttons specify what type of color is visualised in the sliders:

Mir

The mirror colour of the Material. This affects a environment or reflection map.

Spec

Specularity, the colour of the sheen.

Color

The basic colour of the Material.

R, G, B

These mix the colour specified.

Ref

Reflectivity. The degree to which the Material reflects the basic colour when light falls on it.

Alpha

The degree of coverage, which can be used to make Materials transparent. Use the option "ZTransp" to specify that multiple transparent layers can exist. Without this option, only the Material itself is rendered, no matter what faces lie behind it. The transparent information is saved in an *alphalayer*, which can be saved as part of a picture (see DisplayButtons).

Emit

The Material 'emits light', without shedding light on other faces of course.

Ambient

The degree to which the global Ambient colour is applied, a simple form of environmental light. The global Ambient can be specified in the World block, using the WorldButtons. Ambient is useful for giving the total rendering a softer, more coloured atmosphere.

Zoffset

This button allows you to give the face to be rendered an artificial forward offset in Blender's Zbuffer system. This only applies to Materials with the option "ZTransp". This option is used to place cartoon figures on a 3D floor as images with alpha. To prevent the figures from 'floating', the feet and the shadows drawn must be placed partially beneath the floor. The Zoffset option then ensures that the entire figure is displayed. This system offers numerous other applications for giving (flat) images of spatial objects the appropriate 3D placement.

Spec

The degree of sheen (specularity) the material has.

Hard

The hardness of the specularity. A large value gives a hard, concentrated sheen, like that of a billiard ball. A low value gives a metallic sheen.

SpTr

This button makes areas of the Material with a sheen opaque. It can be used to give transparent Materials a 'glass' effect.

Add

This option adds some kind of glow to transparent objects, but only works with the unified renderer.

TexFace

A texture assigned with the UVEditor gives the color information for the faces.

NoMist

The Material is insensitive to "Mist" (see WorldButtons).

Traceable

This term stems from Blender's ray-trace past. It specifies whether or not shadow lamps can 'see' the current Material. Turn the "Traceble" option OFF to prevent undesired shadows.

Shadow

This button determines whether the Material can receive a shadow, i.e. whether a shadow calculation is needed.

Shadeless

This button makes the Material insensitive to light or shadow.

Wire

Only the *edges* of faces are rendered (normal rendering!). This results in an exterior that resembles a wire frame. This option can only be used for Meshes.

VCol Light

If the Mesh vertex has colours (see EditButtons), they are added to the Material as extra *light*. The colours also remain visible without lamps. Use this option to render *radiosity*-like models.

VCol Paint

If the Mesh vertex has colours, this button replaces the basic colour of the Material with these colours. Now light must shine on the Material before you can see it. < point>Halo (TogBut) Instead of rendering the faces, each vertex is rendered as a halo. The *lensflare* effect is a part of the halo. This option change certain MaterialButtons (see the following section).

ZTransp

Transitional Zbuffers can only render opaque faces. Blender uses a modified method to Zbuffer transparent faces. This method requires more memory and calculation time than the normal Zbuffer, which is why the two systems are used alongside each other.

Zinvert

The Material is rendered with an inverse Zbuffer method; front and back are switched.

Env

Environment option. The Material is not rendered and the Zbuffer and render buffers are 'erased' so that the pixel is delivered with Alpha = 0.0.

OnlyShadow

This option determines the *alpha* for transparent Materials based on the degree of shadow. Without a shadow the Material is not visible.

Texture name

A Material has eight *channels* to which Textures can be linked. Each *channel* has its own *mapping*, which is the effect the texture has on the material.

Copy to buffer

The complete *mapping* settings are copied to a temporary buffer.

Copy from buffer

The contents of the temporary buffer are copied to the *mapping* settings.

SepT

Separate Textures. This option forces only the current *channel* to be rendered with its corresponding Texture.

Mapping: coordinates as input.

Each Texture has a 3D coordinate (the texture coordinate) as input. The starting point is generally the global coordinate of the 3D point that can be seen in the pixel to be rendered. A Material has the following Mapping options:

UV

The U-V coordinates of a face or Nurbs surface from an Object make up the texture coordinates. U-V is a commonly used term for specifying the mathematical space of a flat or curved surface.

Object

Every Object in Blender can be used as a source for texture coordinates. For this, the Object's inverse transformation is applied to the global coordinate, which gives the *local* Object coordinate. This links the texture to the position, dimension and rotation of the Object. Generally, an Empty Object is used to specify the exact location of a Texture, e.g. to place a logo on the body of an airplane. Another commonly used approach is to have the 'Texture Object' move to achieve an animated texture.

Object Name

The name of the Object used for the texture coordinates. If the Object does not exist, the button remains empty.

Glob

The global coordinate is passed on to the texture.

Orco

The standard setting. This is the *originalcoordinate* of the Mesh or another ObData block.

Stick

Sticky texture. Blender allows you to assign a texture coordinate to Meshes, which is derived from the manner in which the Camera view sees the Mesh. The screen coordinate (only X,Y) for each vertex is calculated and saved in the Mesh. This makes it appear as if the texture is projected from the Camera; the texture becomes "sticky" (see also "Make Sticky" in the EditButtons). Use "Sticky" to precisely match a 3D object with an Image Texture. Special *morphing* effects can also be achieved.

Win

The screen coordinate (X,Y) is used as a texture coordinate. Use this method to achieve 2D *layering* of different Images.

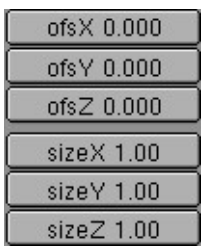
Nor

The normal vector of the rendered face is used as a texture coordinate. Use this method to achieve *reflectionmapping*, which is the suggestion of mirroring using a specially pre-calculated Image.

Refl

The reflection vector of the rendered face is used as a texture coordinate. This vector points in a direction that makes the face appear to be mirrored. Use this option to suggest a reflected surface with procedural textures such as "Marble" or "Clouds" and of course for the use with the EnvMap texture.

Mapping: transform coordinates.



Use these buttons to more finely adjust the texture coordinate.

dX, dY, dZ

The extra translation of the texture coordinate. sizeX, sizeY, sizeZ (NumBut) The extra scaling of the texture coordinate.

Mapping: 3D to 2D

For Image Textures only; this determines the manner is which the 3D coordinate is converted to 2D.

Flat

The X and Y coordinates are used directly.

Cube

Depending on the normal vector of the face, the X-Y or the X-Z or the Y-Z coordinates are selected. This option works well for stones, marbles and other regular textures,

Tube

This creates a tube-shaped mapping. The Z axis becomes the central axis, X and Y revolve around it.

Sphere

This causes a sphere-shaped mapping. The Z axis becomes the central axis, X and Y revolve around it.

Mapping: switch coordinates.

The three rows of buttons indicate the new X, Y and Z coordinates. Normally, the X is mapped to X, the Y to Y and Z to Z. The first button switches a coordinate completely off.

Texture Block

TE:

The name of the Texture block. The name can be changed with this button. Texture Browse (MenuBut) Select an existing Texture from the list provided, or create a new Texture Block.

Clear

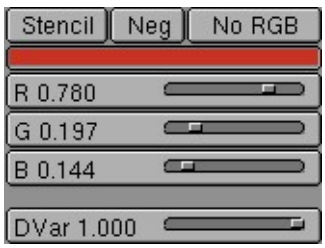
The link to the Texture is erased.

Users

If the Texture Block has multiple users, this button displays the total number of users. Press the button to make the Texture "Single User". An exact copy is made.

Auto Name

Blender assigns a name to the Texture.



Mapping: Texture input settings.

These buttons pass extra information to the Texture.

Stencil

Normally, textures are executed one after the other and laid over one another. A second Texture *channel* can completely replace the first. With this option, the *mapping* goes into *stencil* mode. No subsequent Texture can have an effect on the area the current Texture affects.

Neg

The effect of the Texture is reversed.

RGBtoInt

With this option, an RGB texture (affects colour) is used as an Intensity texture (affects a value).

R, G, B

The colour with which an Intensity texture blends with the current colour.

DVar

The value with which the Intensity texture blends with the current value.

Mapping: output to.

Col

The texture affects the basic colour of the material.

Nor

The texture affects the rendered normal. Only important for Image textures. The "Stucci" texture does this itself.

Csp

The texture affects the *specularity* colour of the material.

Cmir

The texture affects the *mirror* colour of the material, filtered with Mir-RGB sliders.

Ref

The texture affects the value of the material's *reflectivity*. There are three settings; the third setting reverses the effect

Spec

The texture affects the value of *specularity* of the material. There are three settings.

Hard

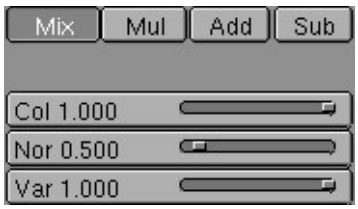
The texture affects the *hardness* value of the material. There are three settings.

Alpha

The texture affects the *alpha* value of the material. There are three settings.

Emit

The texture affects the "Emit" value of the material. There are settings.



Mapping: output settings.

These buttons change the output of the Texture.

Mix

The Texture blends the values or colour.

Mul

The Texture multiplies values or colour.

Add

The Texture adds the values or colour.

Sub

The Texture subtracts values or colour.

Col

The extent to which the texture affects colour.

Nor

The extent to which the texture affects the normal (not important here).

Var

The extent to which the texture affects a value (a variable).

The MaterialButtons, Halos

If a Material has the option "Halo" ON, a number of buttons change to specific halo settings. *Lensflares* can also be created here. Halos are rendered on the 3D location of the vertices. These are small, transparent round spots or pictures over which circles and lines can be drawn. They take Blender's Zbuffer into account; like any 3D element, they can simply disappear behind a face in the forefront.

Halos are placed over the currently rendered background as a separate layer, or they give information to the *alphalayer*, allowing halos to be processed as a post-process.

Only Meshes and Particle Effects can have halos. A Mesh with a halo is displayed differently in the 3DWindow; with small dots at the position of the vertices. Halos cannot be combined with 'ordinary' faces within one Mesh. Only one Material can be used per 'halo' Mesh.

Flare

Each halo is now also rendered as a *lensflare*. This effect suggests the reflections that occur in a camera lens if a strong light source shines on it. A Flare consists of three layers:

- the ordinary halo, which has a 3D location, and can thus disappear behind a face.
- the basic Flare, which is the same halo, but possibly with other dimensions. This is placed over the entire rendering as a post-process.
- the sub Flares, multi-coloured dots and circles, that are also placed over the entire rendering as a post-process.

The "HaloSize" value not only determines the dimensions, but is also used to determine the visibility - and thus the strength - of the Flare rendered in the post-process. This way, a Flare that disappears slowly behind a face will decrease in size at a corresponding speed and gradually go out.

Rings

Determines whether rings are rendered over the basic halo.

Lines

Determines whether star-shaped lines are rendered over the basic halo.

Star

Instead of being rendered as a circle, the basic halo is rendered in the shape of a star. The NumBut "Star" determines the number of points the star has.

Halo

Turn this option OFF to return to a normal Material.

HaloTex

Halos can be given textures in two ways:

- "HaloTex" OFF: the basic colour of each halo is determined by the texture coordinate of the halo-vertex.
- "HaloTex" ON: each halo gets a complete texture area, in which, for example, an Image texture is displayed completely in each basic halo rendered.

HaloPuno

The vertex normal ("Puno" in Blender's turbo language) is used to help specify the dimension of the halo. Normals that point directly at the Camera are the largest; halos with a normal that point to the rear are not rendered. If there are no vertex normals in the Mesh (the Mesh only consists of vertices) the normalised local coordinate of the vertex is used as the normal.

XAlpha

Extreme Alpha. Halos can 'emit light'; they can add colour. This cannot be expressed with a normal *alpha*. Use this option to force a stronger progression in the alpha.

Ring

With this option ON, the colour of the rings can be mixed with the RGB sliders.

Line

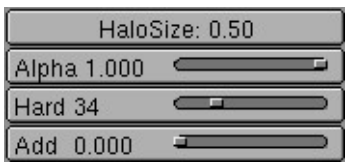
With this option ON, the colour of the lines can be mixed with the RGB sliders.

Halo

With this option ON, the colour of the basic halo can be mixed with the RGB sliders.

R, G, B

Use these sliders to mix the indicated colour.



HaloSize

The dimension of the halo.

Alpha

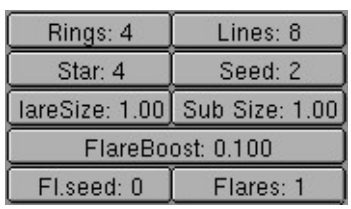
The degree of coverage of the halo.

Hard

The hardness of the halo, a large value gives a strong, concentrated progression.

Add

Normally, the colour of halos is calculated during rendering, giving a light emitting effect. Set the "Add" value to 0.0 to switch this off and make black or 'solid' halos possible as well.



Rings

The number of rings rendered over the basic halo.

Lines

The number of star-shaped lines rendered over the basic halo.

Star

The number of points on the star-shaped basic halo.

Seed

'Random' values are selected for the dimension of the *rings* and the location of the *lines* based on a fixed table. "Seed" determines an offset in the table.

FlareSize

The factor by which the post-process basic Flare is larger than the halo.

SubSize

The dimension of post-process sub Flares, multicolured dots and circles.

FlareBoost

This gives the Flare extra strength.

Fl.seed

The dimension and shape of the sub Flares is determined by a fixed table with 'random' values. "Fl.seed" specifes an offset in the table.

Flares

The number of sub Flares.

The TextureButtons

The settings in this ButtonsWindow visualise the Texture DataBlock. These buttons are only displayed if:

- the active Material has a Texure (see MaterialButtons).
- the active Lamp has a Texture (see LampButtons).
- the World block has a Texture (see WorldButtons)

Blender automatically selects the correct setting if the TextureButtons are called up from the MaterialButtons, LampButtons or WorldButtons. Hotkey: F6. Each Texture

has a 3D coordinate (the texture coordinate) as input. What happens here is determined by the type of texture:

- Intensity textures: return one value. The *previewrender* in this window shows this as grey values.
- RGB textures: returns three values; they always work on colour.
- Bump textures: returns three values; they always work on the normal vector. Only the "Stucci" and "Image" texture can give normals.

The DataButtons in the Header indicate what Texture block is visualised.

Texture Browse

Select another Texture from the list provided, or create a new block.

TE:

Give the current Texture block a new and unique name.

Users

If the Texture block has more than one user, this button shows the total. Press the button to make the Texture "Single User". An exact copy is then created.

Remove Link

Delete the link to the Texture.

Auto Name

Blender assigns a name to the Texture.

The standard TextureButtons

This group of buttons determines the type of user from which the Texture must be displayed. Blender automatically selects the correct settings if the TextureButtons are invoked from the MaterialButtons, LampButtons or WorldButtons.

MA:

The name of the DataBlock that has a (possible) link to the Texture.

Mat

The Texture of the active Material is displayed.

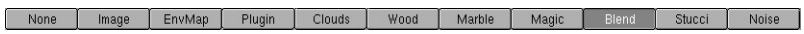
World

The Texture of the World block is displayed.

Lamp

The Texture of the Lamp is displayed.

This group of buttons shows the *channels*. In this example, we see that of the eight available *channels* for the Material, only the first is linked to a Texture.



The program includes nine types of textures. These are described in detail later in this manual.

Default Vars

Changes the values set for the type of texture to the standard values.

Bright

The 'brightness' of the colour or intensity of a texture. In fact, a fixed number is added or subtracted.

Contr

The 'contrast' of the colour or intensity of a texture. This is actually a multiplication.

Colorband

Use this option to create a smooth colour progression. Intensity textures are thus changed into an RGB texture. The use of Colorband with a sharp transistion can cause *aliasing*.

Colorband

Switches the use of Colorband on or off.

Add

Adds a new colour to the Colorband.

Cur:

The active colour from the Colorband.

Del

Delete the active colour.

Pos:

The position of the active colour. Values range from 0.0 to 1.0. This can also be entered using LeftMouse (hold-move) in the Colorband.

E, L, S

The interpolation type with which colours are mixed, i.e. 'Ease', 'Linear' and 'Spline'. The last gives the most fluid progression.

A, R, G, B

The Alpha and RGB value of the active colour.

Image texture

The Image texture is the most frequently used and most advanced of Blender's textures. The standard bump-mapping and perspective-corrected MipMapping, filtering and anti-aliasing built into the program guarantee outstanding image quality (set the DisplayButtons->OSA ON for this). Because pictures are two-dimensionaal, you must specify in the *mapping* buttons how the 3D texture coordinate is converted to 2D; *mapping* is a part of the MaterialButtons.

InterPol

This option interpolates the pixels of an Image. This becomes visible when you enlarge the picture. Turn this option OFF to keep the pixels visible - 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.

UseAlpha

Use the *alphalayer* of the Image.

CalcAlpha

Calcualte an *alpha* based on the RGB values of the Image.

NegAlpha

Reverses the *alpha* value.

MipMap

Generates a series of pictures, each half the size of the former one. This optimises the filtering process. When this option is OFF, you generally get a sharper image, but this can significantly increase calculation time if the filter dimension becomes large.

Fields

Video frames consist of two different images (fields) that are merged by horizontal line. This option makes it possible to work with field images. It ensures that when 'Fields' are rendered (DisplayButtons->Field) the correct field of the Image is used in the correct field of the rendering. MipMapping cannot be combined with "Fields".

Rot90

Rotates the Image 90 degrees when rendered.

Movie

Movie files (AVIs supported by Blender, SGI-movies) and 'anim5' files can also be used for an Image. To do this, set the "Frames" NumBut to the total number of frames.

Anti

Graphic images such as cartoons and pictures that consist of only a few colours with a large surface filling can be *anti*-aliased as a built in pre-process.

St Field

Normally, the first field in a video frame begins on the first line. Some *framegrabbers* do this differently!

Filter

The filter size used by the options "MipMap" and "Interpol".

Load Image

The (largest) adjacent window becomes an ImageSelectWindow. Specify here what file must be read to become an Image.

...

This small button does the same thing, but now simply gives a FileSelect.

ImageBrowse

You can select a previously created Image from the list provided. Image blocks can be reused without taking up extra memory.

File Name

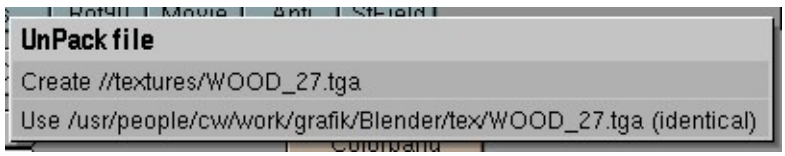
Enter a file name here, after which a new Image block is created.

Users

Indicates the number of users for the Image. The "Single User" option cannot be activated here. It has no significance for Images.

Pack

Indicates the packing of the image. Pressed (grey) means the image is packed into the Blend-file. Klicking on the Button packs or unpacks the image. If a unpack option is triggered the unpack-menu pops up.



Reload

Force the Image file to be read again.

The following options determine what happens if the texture coordinate falls outside the Image.

Extend

Outside the Image the colour of the edge is extended.

Clip

Outside the Image, an *alpha* value of 0.0 is returned. This allows you to 'paste' a small logo on a large object.

ClipCube

The same as "Clip", but now the 'Z' coordinate is calculated as well. Outside a cube-shaped area around the Image, an *alpha* value of 0.0 is returned.

Repeat

The Image is repeated horizontally and vertically.

Xrepeat

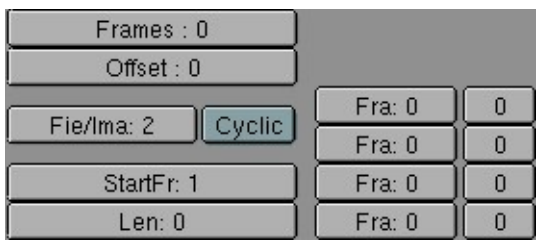
The (extra) degree of repetition in the X direction.

Yrepeat

The (extra) degree of repetition in the Y direction.

MinX, MinY, MaxX, MaxY

Use these to specify a *cropping*, it appears that the Image actually becomes larger or smaller.



Frames

This activates the animation option; another image file (in the same Image block) will be read per rendered frame. Blender tries to find the other files by changing a number in the file name. Only the rightmost digit is interpreted for this. For example: 01.ima.099.tga + 1 becomes 01.ima.100.tga. The value of "Frames" indicates the total number of files to be used. If the option "Movie" is ON, this value must also be set. Now, however, a frame is continually taken from the same file.

Offset

The number of the first picture of the animation.

Fie/Ima

The number of *fields* per rendered frame. If no *fields* are rendered, even numbers must be entered here. (2 fields = 1 frame).

Cyclic

The animation Image is repeated cyclically.

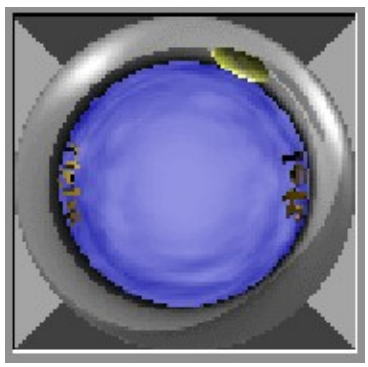
StartFr:

The moment - in Blender frames - at which the animation Image must start.

Len

This button determines the length of the animation. By assigning "Len" a higher value than "Frames", you can create a *still* at the end of the animation. The "Fra:"-buttons allow you to create a simple montage within an animation Image. The left button, "Fra" indicates the frame number, the right-hand button indicates how long the frame must be displayed.

Environment Maps



Blender allows three types of environment maps:

Static

The map is only calculated once during an animation or after loading a file.

Dynamic

The map is calculated each time a rendering takes place. This means moving Objects are displayed correctly in mirroring surfaces.

Load

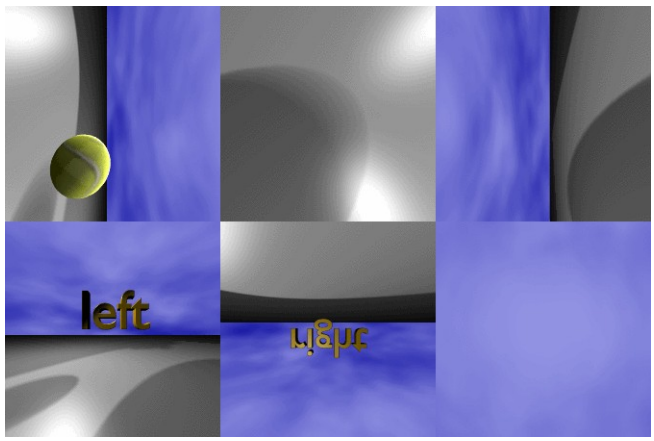
When saved as an image file, environment maps can be loaded from disk. This option allows the fastest rendering with environment maps.

Free Data

This action releases all images associated with the environment map. This is how you force a recalculation when using a Static map.

Save EnvMap

You can save an environment map as an image file, in the format indicated in the DisplayButtons (F10).



These buttons are drawn when the environment map type is "Load". The environment map image then is a regular Image block in the Blender structure.

Load Image

The (largest) adjacent window becomes an ImageSelectWindow. Specify here what file to read in as environment map.

...

This small button does the same thing, but now gives a FileSelect.

ImageBrowse

You can select a previously loaded map from the list provided. EnvMap Images can be reused without taking up extra memory.

File Name

Enter an image file name here, to load as an environment map. Users (But) Indicates the number of users for the Image.Reload (But) Force the Image file to be read again.

Ob:

Fill in the name of an Object that defines the center and rotation of the environment map. This can be any Object in the current Scene.

Filter:

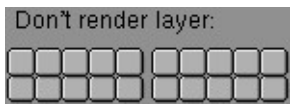
With this value you can adjust the sharpness or blurriness of the reflection.

Clipsta, ClipEnd

These values define the clipping boundaries when rendring the environment map images.

CubeRes

The resolution in pixels of the environment map image.



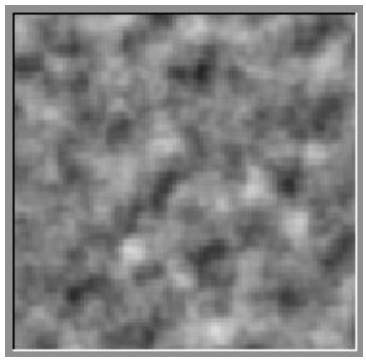
Don't render layer

Indicate with this option that faces that exist in a specific layer are NOT rendered in the environment map.

Plugin texture

Plugins are pieces of compiled C-code which can be loaded by runtime, to extend a programms features. After choosing "Load Plugin" you get a FileWindow which lets you choose a plugin. The plugins are plattform specific, so be sure to load a plugin for your operating system.

Clouds texture



"Clouds" is a *proceduraltexture*. This means that each 3D coordinate can be translated directly to a colour or a value. In this case, a three-dimensional table with pseudo random values is used, from which a fluent interpolation value can be calculated with each 3D coordinate (thanks to Ken Perlin for his masterful article "An Image Synthesizer", from the SIGGRAPH proceedings 1985). This calculation method is also called ~(Perlin) *Noise*.

Default

The standard Noise, gives an Intensity.

Color

The Noise gives an RGB value.

NoiseSize

The dimension of the Noise table.

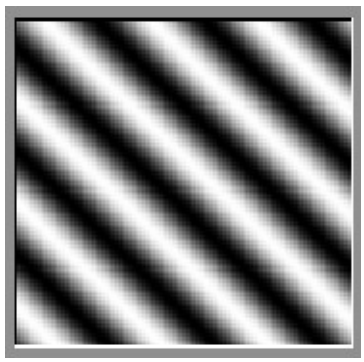
NoiseDepth

The depth of the Cloud calculation. A higher number results in a long calculation time, but also in finer details.

Soft Noise, Hard Noise

There are two methods available for the Noise function.

Wood texture



"Wood" is also a *proceduraltexture*. In this case, bands are generated based on a sine formula. You can also add a degree of turbulence with the Noise formula. It returns an Intensity value only.

Bands

The standard Wood texture.

Rings

This suggests 'wood' rings.

BandNoise

Applying Noise gives the standard Wood texture a certain degree of turbulence.

RingNoise

Applying Noise gives the rings a certain degree of turbulence.

NoiseSize

The dimension of the Noise table.

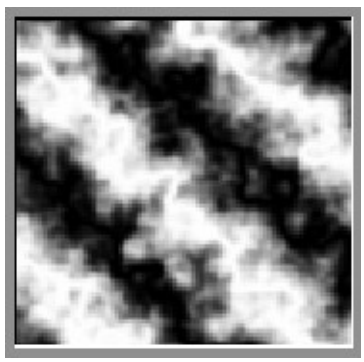
Turbulence

The turbulence of the "BandNoise" and "RingNoise" types.

Soft Noise, Hard Noise

There are two methods available for the Noise function.

Marble texture



"Marble" is also a *proceduraltexture*. In this case, bands are generated based on a sine formula and Noise turbulence. It returns an Intensity value only.

Soft, Sharp, Sharper

Three pre-sets for soft to more clearly defined Marble.

NoiseSize

The dimensions of the Noise table.

NoiseDepth

The depth of the Marble calculation. A higher value results in greater calculation time, but also in finer details.

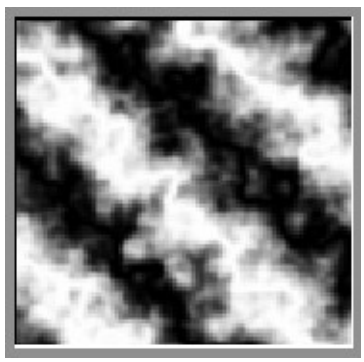
Turbulence

The turbulence of the sine bands.

Soft Noise, Hard Noise

The Noise function works with two methods.

Magic texture



"Magic" is a procedural texture. The RGB components are generated independently with a sine formula.

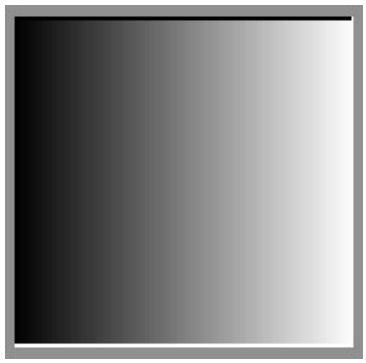
Size

The dimensions of the pattern.

Turbulence

The strength of the pattern.

Blend texture



This is also a *proceduraltexture*. It generates a progression in Intensity.

Lin

A linear progression.

Quad

A quadratic progression. Ease (RowBut) A flowing, non-linear progression.

Diag

A diagonal progression.

Sphere

A progression with the shape of a three-dimensional ball.

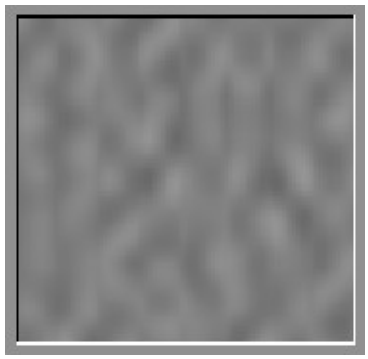
Halo

A quadratic progression with the shape of a three-dimensional ball.

Flip XY

The direction of the progression is flipped a quarter turn.

Stucci texture



This *proceduraltexture* generates Noise-based normals.

Plastic

The standard Stucci. Wall In, Wall out (RowBut) This is where Stucci gets it name. This is a typical wall structure with holes or bumps.

NoiseSize (NumBut) The dimensions of the Noise table.

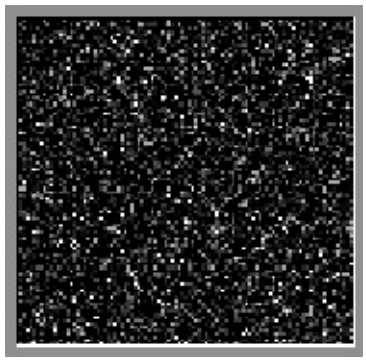
Turbulence

The depth of the Stucci.

Soft Noise, Hard Noise

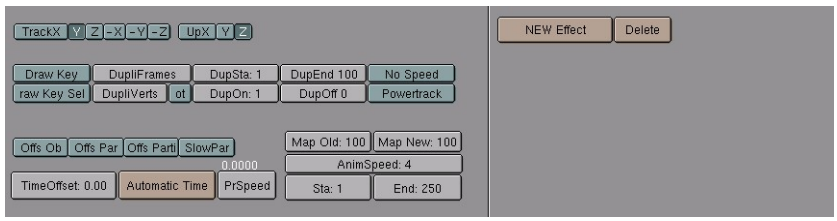
There are two methods available for working with Noise.

Noise texture



Although this looks great, it is not *PerlinNoise*! This is a true, randomly generated Noise. This gives a different result every time, for every frame, for every pixel.

The AnimButtons



This ButtonsWindow visualises settings associated with animations, most of which are part of the Object DataBlock. It can also be used to create Effects: like the 'Build'- and the 'Particles'-Effect. Hotkey: F7.

The typical 'browse' MenuBut is missing here. Link Objects to other Scenes with the LinkMenu (CTRL+L).

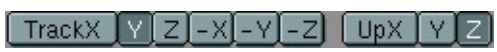
OB:

Give the Object block a new and unique name. The Object is inserted again, sorted alphabetically.

Users

If the Object block has multiple users, this button shows the total number of users. Press the button to make the Object "Single User". An exact copy is then created (exlusive the Object block).

Tracking buttons



In Blender, Objects can be assigned a rotation *constraint*:

- Objects that always point in the direction of other Objects: CTRL+T, or "Make Track".
- Objects as Children of a Curve path, where the curve determines the rotation ("Follow" button).
- Particles can give rotations to Objects (see AnimButtons, Effects). Because Objects have a rotation of their own, it is advisable to first erase this using ALT+R. If the Object is a Child, then erase the "Parent Inverse" as well using ALT+P.

Use these buttons to indicate how *tracking* must work:

TrackX, Y, Z, -X, -Y, -Z

Specifies the direction axis; the axis that, for example, must point to the other Object.

UpX, UpY, UpZ

Specify what axis must point 'up', in the direction of the (global) positive Z axis. If the "Track" axis is the same as the "Up" axis, this is turned off.

PowerTrack

This option completely switches off the Object's own rotation and that of its Parents. Only for Objects that 'track' to another Object.

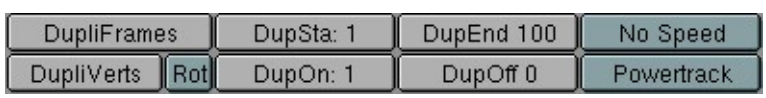
DrawKey

If Objects have an Object Ipo, they can be drawn in the 3Dwindow as *keypositions*. Key positions are drawn with this option ON *and* the IpoKeys ON (in the IpoHeader). Hotkey: KKEY.

DrawKeySel

Limits the drawing of *Objectkeys* to those selected.

Duplicators



Blender can automatically generate Objects without actually creating them. To do this, an animation system must be created first. A 'virtual' copy of the Object will then be placed on every frame specified. It is also possible to have a virtual copy placed on each *vertex* (or particle). This can be used as a modeling tool as well. To do this, select the duplicated Objects and press CTRL-SHIFT+A ("Make Dupli's Real").

DupliFrames

No matter how the Object moves, with its own Object Ipos or on a Curve path, a copy of the Object is made for every frame from "DupSta" to "DupEnd". The "DupliFrames" system is built for the specified frame interval.

DupliVerts

Child Objects are duplicated on all vertices of this Object (only with Mesh).

DupSta, DupEnd

The start and end frame of the duplication.

DupOn, DupOff

Empty positions can be specified with the option "DupliFrames". For example: "DupOn" on '2', "DupOff" on '8' sets two copies on every 10 frames. The duplicated Objects move over the animation system like a sort of train.

No Speed

The "DupliFrames" are set to 'still', regardless of the current frame.

Slurph

This option is only available if there are VertexKeys. The "Slurph" value specifies a fixed delay for the interpolation of Keys *pervertex*. The first vertex comes first, the last vertex has a delay of "Slurph" frames. This effect makes quite special and realistic Key framing possible.

Watch the sequence of vertices carefully with Meshes. The sequence can be sorted with the commands EditButtons->Xsort and EditButons->Hash. Naturally, it is important that this occurs *before* the VertexKeys are created, because otherwise quite unpredictable things can occur (however, it can be nice for Halos).

Relative Keys

This button toggles between using standard vertex keyframing and the use of relative vertex keys. Relative vertex keys allowing mix, add or subtract multiple vertex key positions independently. Best suited for facial expression animations.

OffsOb

The "TimeOffset" value works on its own Object Ipo.

OffsPar

The "TimeOffset" value works on the Parent relationship of the Object.

OffsPart

The "TimeOffset" value works on the Particle Effect.

SlowPar

The value of "TimeOffset" is used to create a 'delay' in the Parent relationship. This delay is cumulative and depends on the previous frame. When rendering animations, the complete sequence must always be rendered, starting with the first frame.

TimeOffset

Depending on the previously mentioned pre-sets, the animation is shifted a number of frames. This does not work for VertexKeys.

Automatic Time

This generates automatic "TimeOffset" values for all *selected* Objects. The start value is the value of the "TimeOffset" button. A requestor pops up and asks for the size of the interval. Blender looks at the Object's screen coordinates in the nearest 3DWindow and calculates the offset values from left to right.

PrSpeed

The speed of the Object is printed.

Map Old, Map New

This button can be used to modify the internal time calculation. "Map Old" gives the previous value in frames; "Map New" specifies the number of frames that must be rendered. Only the mutual relations between these values are important. Use this only to speed up or slow down the entire animation system. The absolute value 'frame' now becomes relative, which can be quite confusing if the animation must still be modified.

AnimSpeed

The maximum speed of the real-time animation playback, expressed in hundredths of a second.

Sta, End

The start and end frame of an animation to be rendered or played real-time.

These buttons are only displayed if the active Object is a Curve.

PathLen

The length of the Curve path in frames, if there is no Speed Ipo.

CurvePath

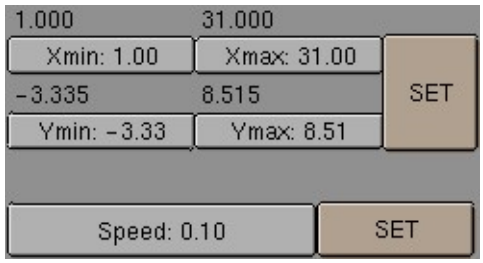
Specifies that the Curve becomes a *path*. Children of this Curve now move over the curve. All Curves can become a *path*, but a 5th order Nurbs curve works best. It has no problems with movement and rotation discontinuity.

CurveFollow

The Curve path passes a rotation to the Child Objects. The 'Tracking' buttons determine which axis the path follows. In EditMode, horizontal lines are also drawn for a 3D curve. This determines the *tilt*, which is an extra axis rotation of the Child Objects. The *tilt* can be changed using the TKEY. Curve paths cannot give uniform perpendicular (aligned with the local Z axis) rotations. In that case, the 'up' axis cannot be determined.

PrintLen

The length of the path is printed in Blender units.



These buttons are displayed if an IpoWindow is present in the same Screen.

Xmin, Xmax, Ymin, Ymax

The numbers above these buttons specify the boundbox of all the visible curves in the IpoWindow. Use the buttons to enter a new value.

Set

The new values of the boundbox are assigned to the visible curves in the IpoWindow.

Speed

In certain cases, the exact speed of a translation caused by Object Ipos must be determined. Proceed as follows to do this:

- In the IpoWindow, make only the LocX, LoxY, LocZ curves visible.
- Set the IpoKey option ON (KKEY in the IpoWindow).
- Select the keys that must be assigned a particular speed.
- Only keys that already have a speed and direction can be changed. If the speed is 0.0, nothing happens.
- Press the "Set" Button.

Anim Effects: Build

Three effects are currently built in: "Build", "Particles" and "Wave". Effects are a fixed part of the Object; they cannot have any links or multiple users.

New Effect

Create a new Effect.

Delete

Delete the Effect.

Build

Select an effect. The Build Effect works on Meshes, which are built up face by face over time. It also works on the vertices in Halo Meshes. The sequence in which this happens can be specified in the 3DWindow with CTRL+F: "Sort Faces" (not in Edit-Mode). The *faces* of the *active* Mesh Object are sorted. The current face in the 3DWindow is taken as the starting point. The leftmost *face* first, the rightmost *face* last.

Len

The total time the building requires.

SFra

The frame number on which the Effect starts.

Anim Effects: Particles

Particles are halos (or Objects if the option "DupliVerts" is ON) that are generated more or less according to laws of physics. Use Particles for smoke, fire, explosions, a fountain, fireworks or a school of fish! With the Static option it is also possible to make fur or even plants.

A Particle system is pre-calculated as a pre-process (this can take some time). They can then be viewed in the 3DWindow in real time. Particles are a full-fledged part of Blender's animation system. They can also controlled by Lattices. Only Meshes can have Particles.

Recalc All

Recalc the particle-system after changing the animation of the emitter mesh. This updates the particle-system.

Static

Making static particles. Particles now don't animate or move anymore, they follow the Object's transformation. Static particles are generated one at each 'frame' for the entire 'Life' value. Use the "step" option to control this; step=2 means a particle at every two frames.

Tot

The total number of Particles. Particles require quite a bit of memory (not in the file!) and rendering time, so specify this value carefully.

Sta, End

The start and end frame between which Particles are generated.

Life

The life span of each Particle.

Keys

Not all Particle locations are calculated and remembered for each frame for the entire particle system. This is only done for a fixed number of *key* positions between which interpolations are performed. A larger number of "Keys" gives a more fluid, detailed movement. This makes significant demands on the memory and time required to calculate the system.

CurMul

Particles can 'multiply themselves' at the end of their lives. For each generation, certain particle settings are unique. This button determines which generation is displayed.

Mat

The Material used for the current generation of Particles.

Mult

This determiness whether the particles multiply themselves. A value of 0.0 switches this off. A value of 1.0 means that each Particle multiplies itself. The particle system itself ensures that the *total* number of Particles is limited to the "Tot" value.

Life

The age of the Particles in the following generation.

Child

The number of children of a Particle that has multiplied itself.

RandLife

A factor that ascribes the age of Particles a (pseudo) random variation.

Seed

The offset in the random table.

Face

With this option particles are not only emited from vertices, but also from the faces of the mesh.

Bspline

The Particles are interpolated from the *keys* using a B-spline formula. This give a much more fluid progression, but the particles no longer pass exactly through the *key* positions.

Vect

This gives particles a rotation direction. This can be seen in the Halo rendering. Particles that duplicate Objects now also give a rotation to these Objects. VectSize (TogBut) The extent to which the speed of the "Vect" Particle works on the dimensions of the Halo.

Norm

The extent to which the vertex normal of the Mesh gives the Particle a starting speed. If the Mesh has no faces (and thus no vertex normals) the normalised *local* vertex coordinate is used as the starting speed.

Ob

The Extent to which the speed of the Object gives the Particle a starting speed. This makes a rotating cube become a sort of 'sprinkler'.

Rand

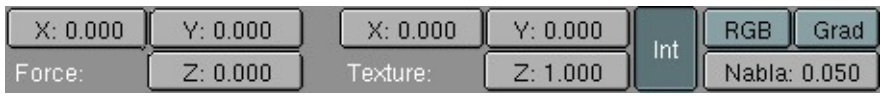
The extent to which a (pseudo) random value gives the Particle a starting speed.

Tex

The extent to which the Texture gives the Particle a starting speed. For this, only the last Texture of the Material is used, in *channel* number 8.

Damp

Use of damping reduces the speed, like a sort of friction.



Force X, Y, Z

A standard, continually present force. This can simulate the effect of gravity or wind.

Texture X, Y, Z

A standard force that works on a Particle, determined by the texture. Textures can have an effect on the movement of Particles. The 3D coordinate of the Particle is passed to the texture per Particle *key*.

Int

The Intensity that is passed back from the texture is used as a factor for the standard texture force (previous three buttons).

RGB

The colour of the texture has a direct effect on the speed of the Particle: Red on the X, Green on the Y and Blue on the Z component of the speed.

Grad

The *gradient* of the texture is calculated. This is the mathematical derivitive. Four samples of the texture are combined to produce a speed vector. With *procedural* textures, such as Clouds, this method gives a very beautiful, turbulent effect. Set the number of "Keys" as high as possible to see the sometimes rather subtle twisting.

Nabla

The dimension of the area in which the *gradient* is calculated. This value must be carefully adjusted to the frequency of the texture.

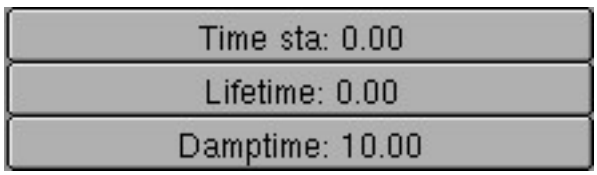
Anim Effects: Wave

The Wave Effect adds a animated Wave to a Mesh. It is not limited to flat objects but can also be used to make a sphere 'wobble'.

The Wave Effect can be accessed from the AnimButtons **F7** while the mesh is active. Choose 'NEW Effect' and change it with the MenuButton to 'Wave'.

Wave Type

Per default you have then a XY Wave on your Object. With the Buttons X and Y you can enable or disable the wave generation for an axis, look at the image below for the three basic effects. The Button "Cycl" makes the generation cyclic in the animation.



Time Sta

When (in frames of the animation) the wave generation should start.

Lifetime

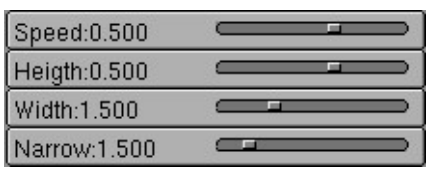
How long (in frames) a wave exists

Damptime

How many frames the wave schould extenuate.

Sta X, Sta Y

Starting Position of the Wave



Speed

Speed the Wave travels, can also be negative.

Height

Amplitude of the Wave.

Width

Width of the wave (wavelenght)

Narrow

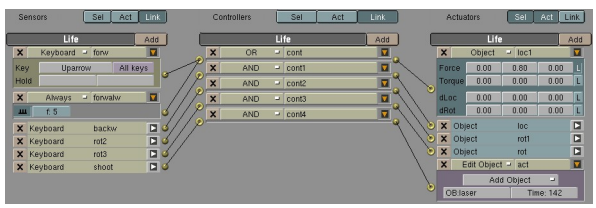
How narrow the next wave follows.

Realtime Buttons

The RealtimeButtons are meant for making interactive 3D-animations in Blender. Blender acts then as a complete developement tool for interactive worlds including a gameEngine to play the worlds. All is done without compiling the game or interactive world. Just press **PKEY** and it runs in realtime. This manual does not cover the realtime part of Blender, because it is a complex process which needs the full attention of a seperate book. Hoever we like to give you a overview what can be done how with Blender. Visit our website www.blender.nl to see the latest developements of the gameEngine and find tutorials giving you a start in interactive 3D-graphics.

The RealtimeButtons can be logical seperated in to parts. The left part contains global settings for elements of the game.

This includes settings for general physics, like the damping or mass. Here you also define if an object should be calculated with the build in physic or should be handeled static or forming a level. Here you can also define properties of game objects, these properties can carry values which describe attributes of the object like variables in a programming language.



The right part of the RealtimeButtons is the comand center to add game logic to your objects and worlds. It consists of the sensors, controlers and actuators.

Sensors are like the senses of a lifeform, the react on keypresses, collisions, contact with materials, timer events or values of properties.

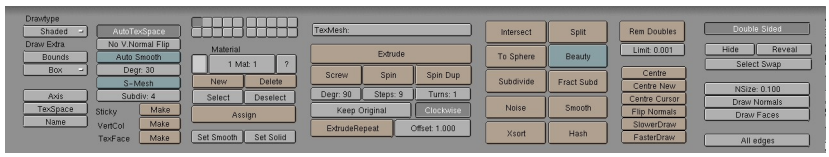
The controlers are collecting events from the sensors and are able to calculate them to a result. Simple actuators are just doing a AND for example to test if a key is pressed and a certain time is over. There are also OR actuators and you also can use python-scripting to do more complex stuff.

The actuator are then actually do things on the objects. This can be applying forces to objects to move or rotate them, playing predefined animations (via IPOs) or adding new objects.

The logic is connected (wired) with the mouse amongst the sensors, controlers and actuators. After that you are immediately able to play the game! If you discover something in the game you don´t like, just stop, edit and restart. This way you get fantastic turnaround times in your developement.

EditButtons

EditButtons, general, F9KEY



The settings in this ButtonsWindow visualise the ObData blocks and provide tools for the specific EditModes. Certain buttons are redrawn depending on the type of ObData. The types are: Mesh, Curve, Surface, Text, MetaBall, Lattice, Ika and Camera. This section describes the buttons that appear for nearly all ObData. Later in the text, the buttons are grouped per ObData type. A complete overview of all HotKeys for EditMode is provided in the 3Dwindow section.

The DataButtons in the Header specify what block is visualised. Mesh is used as an example here, but the usage of the other types of ObData is identical.

Mesh Browse

Select another Mesh from the list provided.

ME:

Give the current block a new and unique name. The new name is inserted in the list, sorted alphabetically.

Users

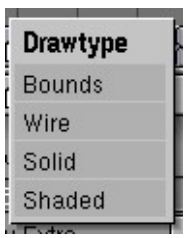
If the block is used by more than one Object, this button shows the total number of Objects. Press the button to change this to "Single User". An exact copy is then created.

OB:

Give the current Object a new and unique name. The new name is inserted in the list, sorted alphabetically.

This group of buttons specifies Object characteristics. They are displayed here for ease.

DrawType



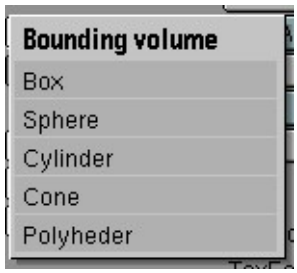
Choose a preference for the standard display method in the 3D window from the list provided. The "DrawType" is compared with the "DrawMode" set in the 3Dheader; the least complex method is the one actually used.

The types, in increasing degree of complexity, are:

- Bounds. A bounding object in the dimensions of the object is drawn.
- Wire. The wire model is drawn.
- Solid. Zbuffered with the standard OpenGL lighting.
- Shaded. This display, which uses Gouraud shading, is the best possible approach to the manner in which Blender renders. It depicts the situation of a single frame from the Camera. Use CTRL+Z to force a recalculation.

The "Draw Extra" options are displayed above the selected DrawType.

BoundBox



A bounding object is displayed in the dimensions of the object.

Box

With this MenuButton you can choose between different bound-objects.

Axis

The axes are drawn with X, Y and Z indicated.

TexSpace

The texture space. This can be different from the BoundBox. It is displayed with broken lines.

Name

The name of the Object is printed at the Object centre.

Do Centre

Each ObData has its own local 3D space. The null point of this space is placed at the Object centre. This option calculates a new, centred null point in the ObData. This may change texture coordinates.

Centre New

As above, but now the Object is placed in such a way that the ObData appears to remain in the same place.

Centre Cursor

The new null point of the object is the 3D-Cursor location.

The *layer* setting of the Object. Use SHIFT+LeftMouse to activate multiple layers.

Material *indices*

Objects and ObData can be linked to more than one Material. This can be managed with these buttons.

1 Mat 1

This button can be used to specify which Material should be shown, i.e. which Material is *active*. The first digit indicates the amount of Materials, the second digit indicates the *index* number of the *active* Material. Each *face* in a Mesh has a corresponding number: the 'Material index'. The same is true of Curves and Surfaces.

?

In EditMode, this Button indicates what *index* number, and thus what Material, the selected items have.

New

Make a new *index*. The current Material is assigned an extra link. If there was no Material, a new one is created.

Delete

Delete the current *index*. The current Material gets one less link. The already used *index* numbers are modified in the ObData.

Select

In EditMode, everything is selected with the current *index* number.

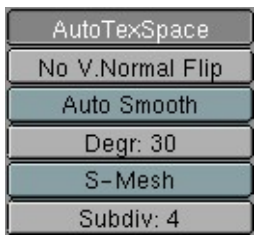
Deselect

In EditMode, everything is deselected with the current *index* number.

Assign

In EditMode, the current *index* number is assigned to the selected items.

EditButtons, Mesh



AutoTexSpace

This option calculates the texture area automatically, after leaving EditMode. You can also specify a texture area yourself (Outside EditMode, in the 3DWindow; TKEY), in which case this option is turned OFF.

No V.Normal Flip

Because Blender normally renders double-sided, the direction of the normal (towards the front or the back) is automatically corrected during rendering. This option turns this automatic correction off, allowing "smooth" rendering with faces that have sharp angles (smaller than 100 degrees). Be sure the face normals are set consistently in the same direction (CTRL+N in EditMode).

AutoSmooth

Automatic smooth rendering (not faceted) for meshes. Especially interesting for imported Meshes done in other 3D applications. The Button "Set smooth" also has to be activated to make "Auto Smooth" work. The smoothing isn´t displayed in the 3D Window.

Degr:

Determines the degree in which faces can meet and still get smoothed by "Auto Smooth".

S-Mesh

The S-Mesh option turns a Mesh Object into a S-Mesh. S-Mesh means procedural smooth subdivision of Mesh objects.

Subdiv:

Number of subdivisions for S-Meshes.

Make Sticky

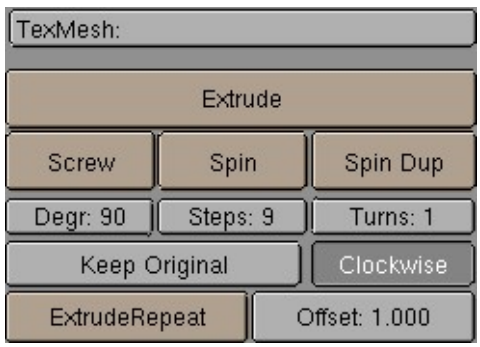
Blender allows you to assign a texture coordinate to Meshes that is derived from the way the Camera view sees the Mesh. The screen coordinates (only X,Y) are calculated from each vertex and these coordinates are stored in the Mesh. As if the texture is permanently projected and fixed on the Mesh as viewed from the Camera; it becomes "sticky". Use "Sticky" to match a 3D object exactly with the Image Texture of a 3D picture. This option also allows you to create special *morphing* effects. If the image is already "sticky", the button allows you to remove this effect.

Make VertCol

A colour can be specified per vertex. This is required for the VertexPaint option. If the Object DrawType is "Shaded", these colours are copied to the vertex colours. This allows you to achieve a *radiosity*-like effect (set MaterialButtons->VertCol ON). If the Mesh is "Double Sided", this is automatically turned off.

Make TexFace

Assigns a texture per face. WIll be automaticly set when you use the UV-Editor to texture a realtime modell.



TexMesh

Enter the name of another Mesh block here to be used as the source for the texture coordinates. *Morphing*-like effects can then be achieved by distorting the *active* Mesh. For example, a straight stream of water (as an animated texture) can be placed in a winding river.

Extrude

The most important of the Mesh tools: Extrude Selected. "Extrude" in EditMode converts all selected *edges* to *faces*. If possible, the selected faces are also duplicated. Grab mode starts immediately after this command is executed. If there are multiple

3DWindows, the mouse cursor changes to a question mark. Click at the 3DWindow in which "Extrude" must be executed. HotKey: EKEY.

Screw

This tool starts a repetitive "Spin" with a screw-shaped revolution on the selected vertices. You can use this to create screws, springs or shell-shaped structures.

Spin

The "Spin" operation is a repetitively rotating "Extrude". This can be used in every view of the 3DWindow, the rotation axis is always through the 3DCursor, perpendicular to the screen. Set the buttons "Degr" and "Steps" to the desired value. If there are multiple 3DWindows, the mouse cursor changes to a question mark. Click at the 3DWindow in which the "Spin" must occur.

Spin Dup

Like "Spin", but instead of an "Extrude", there is duplication.

Degr

The number of degrees the "Spin" revolves.

Steps

The total number of "Spin" revolutions, or the number of steps of the "Screw" per revolution.

Turns

The number of revolutions the "Screw" turns.

Keep Original

This option saves the selected original for a "Spin" or "Screw" operation. This releases the new vertices and faces from the original piece.

Clockwise

The direction of the "Screw" or "Spin", clockwise, or counterclockwise.

Extrude Repeat

This creates a repetitive "Extrude" along a straight line. This takes place perpendicular to the view of the 3DWindow.

Offset

The distance between each step of the "Extrude Repeat". HotKey: WKEY.

Intersect

Select the faces (vertices) that need an intersection and press this button. Blender now intersects all selected faces with each other.

Split

In EditMode, this command 'splits' the selected part of a Mesh without removing faces. The split sections are no longer connected by *edges*. Use this to control *smoothing*. Since the split parts can have vertices at the same position, we recommend that you make selections with the LKEY. HotKey: YKEY.

To Sphere

All selected vertices are blown up into a spherical shape, with the 3DCursor as a midpoint. A requester asks you to specify the factor for this action. HotKey: WKEY.

Beauty

This is an option for "Subdivide". It splits the faces into halves lengthwise, converting elongated faces to squares. If the face is smaller than the value of "Limit", it is not longer split in two. Subdivide (But) Selected faces are divided into quarters; all edges are split in half. HotKey: WKEY.

Fract Subd

Fractal Subdivide. Like "Subdivide", but now the new vertices are set with a random vector up or down. A requestor asks you to specify the amount. Use this to generate landscapes or mountains.

Noise

Here Textures can be used to move the selected vertices up a specific amount. The local vertex coordinate is used as the texture coordinate. Every Texture type works

with this option. For example, the Stucci produce a landscape effect. Or use Images to express this in relief.

Smooth

All edges with both vertices selected are shortened. This flattens sharp angles. HotKey: WKEY.

Xsort

Sorts the vertices in the X direction. This creates interesting effects with VertexKeys or 'Build Effects' for Halos.

Hash

This makes the sequence of vertices completely random.

Rem Doubles

Remove Doubles. All selected vertices closer to one another than "Limit" are combined and redundant faces are removed.

Flip Normals

Toggles the direction of the face normals. HotKey: WKEY.

SlowerDraw, FasterDraw.

When leaving EditMode all edges are tested to determine whether they must be displayed as a wire frame. Edges that share two faces with the same normal are never displayed. This increases the recognisability of the Mesh and considerably speeds up drawing. With "SlowerDraw" and "FasterDraw", you can specify that additional or fewer edges must be drawn when you are not in EditMode.

Double Sided

Only for display in the 3Dwindow; can be used to control whether double-sided faces are drawn. Turn this option OFF if the Object has a negative 'size' value (for example an X-flip).

Hide

All selected vertices are temporarily hidden. HotKey: HKEY.

Reveal

This undoes the "Hide" option. HotKey: ALT+H.

Select Swap

Toggle the selection status of all vertices.

NSize

The length of the face normals, if they have been drawn.

Draw Normals

Indicates that the face normals must be drawn in EditMode.

Draw Faces

Indicates that the face must be drawn (as Wire) in EditMode. Now it also indicates whether faces are selected.

AllEdges

After leaving EditMode, all edges are drawn normally, without optimisation.

EditButtons, Curve and Surface

These options convert selected curves.

Poly

A polygon only gives a linear interpolation of vertices.

Bezier

Vertices in a Bezier curve are grouped in threes; the *handles*. The most frequently used curve type for creating letters or logos.

Bspline

(Obsolete.-cw-)

Cardinal

(Obsolete.-cw-)

Nurb

A Nurbs curve is mathematically quite 'pure'. For example: it can be used to create perfect circles.

Nurbs curves have *knots*, a row of numbers that specify the exact curve. Blender offers three pre-sets for this:

Uniform U, V

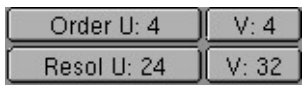
Sets the *knots* to create a uniform distribution. Use this for closed curves or surfaces.

Endpoint U, V

Sets the *knots* so that the first and last vertices are always included.

Bezier U, V

Sets the *knots* table in such a way that the Nurbs behave like a Bezier.



Order U, V

The *order* is the 'depth' of the curve calculation. Order '1' is a point, order '2' is linear, order '3' is quadratic, etc. Always use *order* '5' for Curve paths. Order '5' behaves fluently under all circumstances, without annoying discontinuity in the movement.

ReslolU, V

The resolution in which the interpolation occurs; the number of points that must be generated between two vertices in the curve.

Set Weight

Nurbs curves have a 'weight' per vertex; the extent to which a vertex participates in the interpolation. This button assigns the "Weight" value to all selected vertices.

Weight

The weight that is assigned with "Set Weight".

1.0, sqrt

A number of pre-sets that can be used to create pure circles and spheres.

DefResolU

The standard resolution in the U direction for curves.

Set

Assigns the value of "DefResolU" to all selected curves.

Back

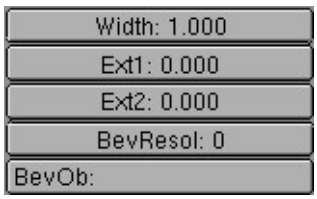
Specifies that the back side of (extruded) 2D curves should be filled.

Front

Specifes that the front side of (extruded) 2D curves should be filled.

3D

The curve may now have vertices on each 3D coordinate; the front and back side are never rendered.



These buttons are only drawn for Curve and Font Objects.

Width

The interpolated points of a curve can be moved farther apart or brought closer together.

Ext1

The depth of the extrude.

Ext2

The depth of the standard bevel.

BevResol

The resolution of the standard bevel; the bevel eventually becomes a quarter circle.

BevOb

The 'bevel' Object. Fill in the name of another Curve Object; this now forms the bevel. For each interpolated point on the curve, the 'bevel Object' is, as it were, extruded and rotated. With this method, for example, you can create the rails of a roller coaster with a 3D curve as the base and two small squares as bevels. Set the values "ResolU" of both Curves carefully, given that this beveling can generate many faces.

Hide

All selected vertices are hidden temporarily.

Reveal

This undoes the "Hide" operation.

Select Swap

Toggle the selection status of all vertices.

Subdivide

Create new vertices or *handles* in curves.

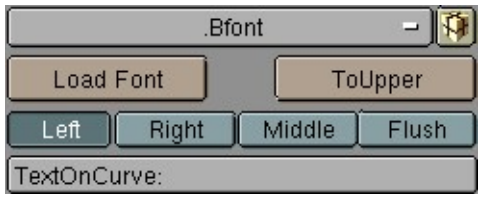
NSize

This determines the length of the 'tilt' lines in 3DCurves.

Spin

This button is only available for Surface Objects. It makes selected Nurb curves a surface of revolution. The rotation axis runs perpendicular to the screen through the 3DCursor.

EditButtons, Font



Font type

Select a font from the list.

Load Font

In the FileSelect Window, this specifies an "Adobe type 1" file that must be read.

Pack Font

To pack the font into the *.blend

ToUpper

In EditMode, changes all letters into capitals or, if there are no small letters, changes all capitals to small letters.

Left

All text is left-aligned.

Right

All text is right-aligned.

Middle

The text is centered.

Flush

The text is spread out to full length; the length of the longest sentence.

TextOnCurve

Enter the name of a Curve Object here; this now forms the line along which the text is placed.

Size

The letter size.

Linedist

The distance between two lines of text.

Spacing

The size of the space between two letters.

Yoffset

This shifts the text up or down. For adjusting "TextOnCurve".

Shear

Changes the letters to italics.

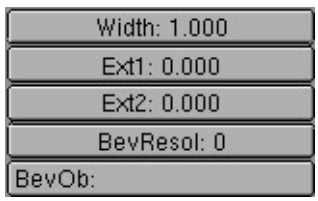
Xoffset

This moves the text left or right. For adjusting "TextOnCurve".

Ob Family

You can create fonts yourself within a Blender file. Each letter from this Font Object is then replaced by any Object you chose, and is automatically duplicated. This means that you can type with Objects! Objects to be considered as letters must belong to the same 'family'; they must have a name that corresponds to the other letter Objects and with the name that must be entered in this button. Important: set the option AnimButtons->DupliVerts ON! For example:

- "Ob Family" = Weird.
- The Objects that are to replace the letters a and b are called 'Weirda' and 'Weirdb', respectively.



Width

The interpolated points of a text can be moved farther apart or brought closer together.

Ext1

The depth of the extrude.

Ext2

The depth of the standard bevel.

BevResol

The resolution of the standard bevel; the bevel eventually becomes a quarter circle.

BevOb

The 'bevel' Object. Fill in the name of a Curve Object; this now forms the bevel. For each interpolated point on the curve, the 'bevel Object' is, as it were, extruded and rotated. Set the values "ResolU" of both Text and Curve carefully, given that this beveling can generate many faces.

EditButtons, MetaBall

WireSize

Determines the resolution of the MetaBall displayed in the 3DWindow. Be careful with small values, as they use a lot of memory.

RenderSize

The resolution of the rendered MetaBall.

Threshold

This value determines the global 'stiffness' of the MetaBall.

Always

In EditMode, the MetaBall is completely recalculated during *transformations*.

HalfRes

The MetaBall is calculated in half resolution.

Fast

The MetaBall is only recalculated after the *transformation*.

In EditMode these Buttons apply to the active Ball:

Stiffness

The stiffness can be specified separately per 'ball', only for the *active* ball.

Len

MetaBall elements can also be tube-shaped. This button specifies the length of the *active* ball.

Negative

The active 'ball' has a negative effect on the other balls.

Ball

The standard type.

TubeX, TubeY, TubeZ

The active 'ball' becomes a tube; in the X, Y of Z direction.

EditButtons, Lattice

Meshes and Surfaces can be deformed with Lattices, provided the Lattice is the Parent of the Mesh or Surface.

U, V, W

The three dimensions of the Lattice. If a new value is entered here, the Lattice is placed in a regular, standard position.

Lin, Card, B

The manner in which the deformation is calculated can be specified per dimension of the Lattice. The options are: Linear, Cardinal spline and B spline. The last option gives the most fluid deformation.

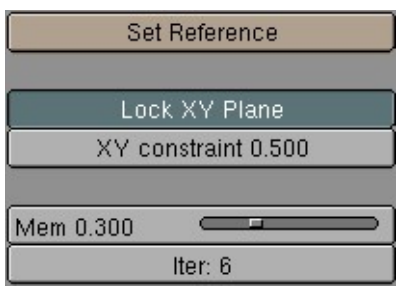
Make Regular

This option sets the Lattice in a regular, standard position.

Outside

This type Lattice only displays vertices on the outside. The inner vertices are interpolated linearly. This makes working with large Lattices simpler.

EditButtons, Ika



Set Reference

The reference position of an Ika determines the position from which the Ika is calculated towards the user-specified position. This results in a sort of 'memory', a rest mode to which the Ika can always return. A slightly bent form works best as a reference. This position is also evaluated if the IKA has a Skeleton deformation; this is the state in which *no* deformation occurs.

Lock XY Plane

With this option you are limiting the effector to the XY-plane, to avoid annoying Y-axis flips. This type is default now and much stabler to work with. Known problem: rotating an Ika (with RKEY) is not well defined, so it is better to disable inverse kinematics first with TABKEY and then rotate it.

XY Constraint

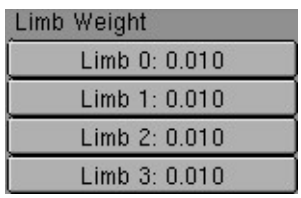
Amount of constrain to the XY Plane.

Mem

This is the extent to which the reference position has an effect on the Ika setting. Set this value to 0.0 to create a completely slack chain.

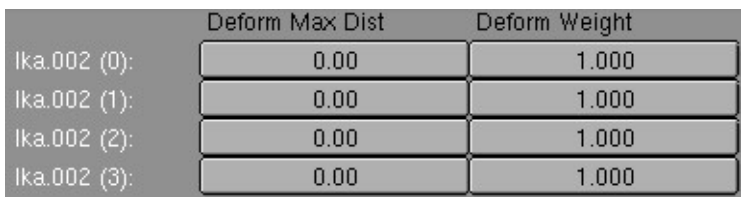
Iter

The number of iterations of the Ika calculation. To achieve a natural expression, this value can be kept low. During animation or transformation, the Ika then moves to the desired position slowly.



Limb Weight

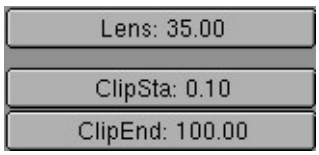
These numbers give a relationship factor per *limb* for how stiff or heavy the limb is in relation to other limbs.



Skeleton Weight

A Skeleton deformation can consist of multiple Ika Objects. These numbers determine the extent to which each Ika contributes to deformation.(Deform Weight) and how far this influence reaches (Deform Max Dist). A "Deform Max Dist" of zero works with an global fall-off, like in older Blender versions.

EditButtons, Camera



Lens

This number is derived from the lens values of a photo camera: ’120’ is telelens, ’50’ is normal, ’28’ is wide angle.

ClipSta, ClipEnd

Everything that is visible from the Camera’s point of view between these values is rendered. Try to keep these values close to one another, so that the Zbuffer functions optimally.

DrawSize

The size in which the Camera is drawn in the 3DWindow.

Ortho

A Camera can also render orthogonally. The distance from the Camera then has no effect on the size of the rendered objects.

ShowLimits

A line that indicates the values of "ClipSta" and "ClipEnd" is drawn in the 3Dwindow near the Camera.

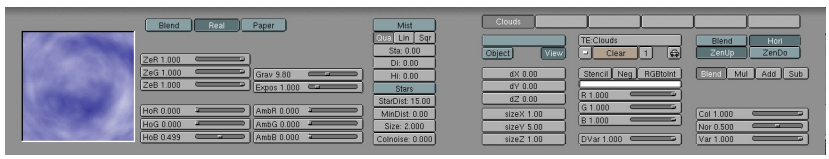
ShowMist

A line that indicates the area of the 'mist' (see WorldButtons) is drawn near the Camera in the 3Dwindow.

Constraint Buttons

Sound Buttons

The WorldButtons



The settings in this ButtonsWindow visualise the World DataBlock. It is linked to a Scene, and can therefore be reused by other Scenes. This block contains the settings for standard backgrounds ('sky'), mist effects and the built-in star generator. The *ambient* colour and *exposure* time can be set here as well.

World Browse

Select another World from the list provided, or create a new block.

WO:

Give the current World block a new and unique name.

Users

If the World block has more than one user, this button shows the total number of users. Press the button to make the World "Single User". An exact copy is then created.

Remove Link

Delete the link to the World.

Sky options.

Where the *alpha* in the rendering is less than 1.0, a *sky* colour is filled in. The *alpha* is then no longer usable for post-processing (unless the *sky* is black).

Blend

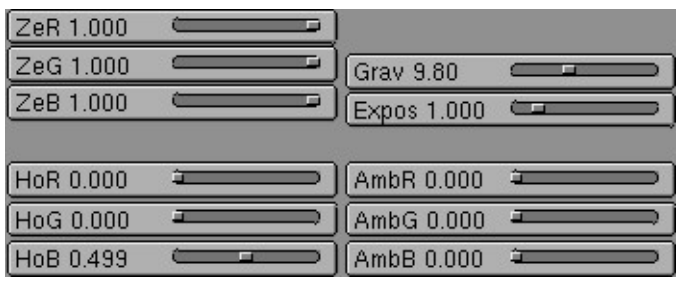
This option renders the background, e.g. a *sky*, with a natural progression. At the bottom of the image is the horizon colour, at the top, the colour of the zenith. The progression is not linear, but bent in the shape of a ball, depending on the *lens* value of the Camera.

Real

The option "Real" makes the position of the horizon *real*; the direction in which the camera is pointed determines whether the horizon or the zenith can be seen. This also influences the generated texture coordinates.

Paper

This option makes the "Blend" (if this is selected) or the texture coordinates completely flat, at 'viewport' level.



ZeR, ZeG, ZeB

The colour of the *zenith*. This is the point directly above or directly below an observer (on the earth!).

HoR, HoG, HoB

The colour of the *horizon*.

AmbR, AmbG, AmbB

The colour of the environmental light, the *ambient*. This is a rather primitive way to make the entire rendering lighter, or to change the colour temperature.

Grav

This slider defines the gravity for the realtime part of Blender.

Expos

The lighting time, *exposure*. In fact, this causes a global strengthening or reduction in all the lamps. Use this to give the rendering more contrast.

Mist

Activates the rendering of *mist*. All rendered faces and halos are given an extra *alpha* value, based on their distance from the camera. If a 'sky' colour is specified, this is filled in behind the *alpha*.

Qua, Lin, Sqr

Determines the progression of the mist. Quadratic, linear or inverse quadratic (square root), respectively. "Sqr" gives a thick 'soupy' mist, as if the picture is rendered under water.

Sta

The start distance of the mist, measured from the Camera.

Di

The depth of the mist, with the distance measured from "Sta".

Hi

With this option, the mist becomes thinner the higher it goes. This is measured from Z = 0.0. If the value of "Hi" is set to zero, this effect is turned off.

Stars

Blender has an automatic star generator. These are standard halos that are only generated in the *sky*. With this option ON, stars are also drawn in the 3DWindow (as small points).

StarDist

The average distance between two stars. Do not allow this value to become too small, as this will generate an overflow.

MinDist

In reality, stars are light years apart. In the Blender universe, this distance is much smaller. To prevent stars from appearing too close to the Camera, you can set a "MinDist" value. Stars will never appear within this distance.

Size

The average screen dimensions of a star.

ColNoise

This value randomly selects star colour.

Texture Channels

Texture name

A World has six *channels* with which Textures can be linked. This is only used for the *sky*. Each *channel* has its own *mapping*; i.e. the effect the texture has on the *sky*. The settings are in the buttons described below.

Mapping: coordinates input.

Each Texture has a 3D coordinate (the texture coordinate) as input. A sky has three options for this.

Object Name

The name of the Object that is used as a source for the texture coordinates. If the Object does not exist, the button remains empty.

Object

Each Object in Blender can be used as a source for texture coordinates. To accomplish this, an inverse transformation is used to obtain the *local* Object coordinate. This links the texture to the position, dimensions and rotation of the Object.

View

The *view* vector of the camera is passed on to the texture.

Mapping: transform coordinates.

Use these buttons to more finely adjust the buttons texture coordinate.

dX, dY, dZ

The extra translation of the texture coordinate.

sizeX, sizeY, sizeZ

The extra scaling of the texture coordinate.

TE:

The name of the Texture block. Use this button to change the name.

Texture Browse

Choose an existing Texture from the list provided, or create a new Texture Block.

Clear

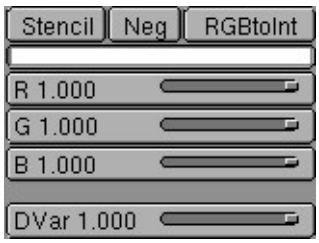
The link to the Texture is erased.

Users

If the Texture Block has more than one user, this button shows the total number of users. Press the button to make the Texture "Single User". An exact copy is then created.

Auto Name

Blender assigns a name to the Texture.



Mapping: Texture input settings.

These buttons pass extra information on to the Texture.

Stencil

Textures are normally executed one after the other and layed over one another. A second Texture *channel* can completely replace the first one. This option sets the *mapping* to *stencil* mode. No subsequent Texture can operate where this Texture is operating. Neg (TogBut) The Texture operation is reversed.

RGBtoInt

This option causes an RGB texture (works on colour) to be used as an Intensity texture (works on a value).

R, G, B

The colour that an Intensity texture blends with the current colour.

DVar

The value that an Intensity texture blends with the current value.

Mapping: output to.

Blend

The texture works on the colour progression in the sky.

Hori

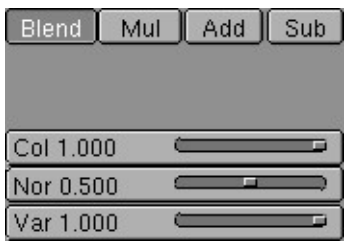
The texture works on the colour of the horizon.

ZenUp

The texture works on the colour of the zenith above.

ZenDown

The texture works on the colour of the zenith below.



Mapping: output settings.

These buttons adjust the output of the Texture.

Blend

The Texture blends the values.

Mul

The Texture multiplies the values.

Add

The Texture adds the values.

Sub

The Texture subtracts the values.

Col

The extent to which the texture works on colour.

Nor

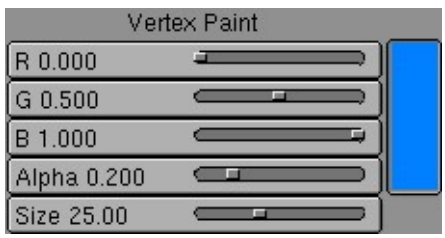
The extent to which the texture works on the normal (not applicable here).

Var

The extent to which the texture works on a value (a single variable).

Paint/Face Buttons

In Blender, the vertices of a Mesh can be assigned a colour, using EditButtons->"Make VertCol". Then, you can change the colour manually, as if you are painting the Mesh (start vertexPaint mode with VKEY in the 3DWindow). This ButtonsMenu has no HotKey, and can only be invoked in the ButtonsHeader with the 'brush' IconBut. The second part of these Buttons is to set drawmodes needed for the UV-Editor.



R, G, B

The active colour used for painting.

Alpha

The extent to which the vertex colour changes while you are painting.

Size

The size of the brush, which is drawn as a circle during painting.

Mix

The manner in which the new colour replaces the old when painting: the colours are mixed.

Add

The colours are added.

Sub

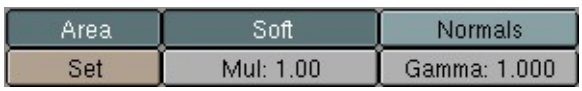
The paint colour is subtracted from the vertex colour.

Mul

The paint colour is multiplied by the vertex colour.

Filter

The colours of the vertices of the painted face are mixed together, with an "alpha" factor.



Area

In the *back* buffer, Blender creates an image of the painted Mesh, assigning each face a colour number. This allows the software to quickly see what faces are being painted. Then, the software calculates how much of the face the brush covers, for the degree to which paint is being applied. You can set this calculation with the option "Area".

Soft

This specifies that the extent to which the vertices lie within the brush also determine the brush's effect.

Normals

The vertex normal (helps) determine the extent of painting. This causes an effect as if painting with light.

Set

The "Mul" and "Gamma" factors are applied to the vertex colours of the Mesh.

Mul

The number by which the vertex colours can be multiplied.

Gamma

The number by which the clarity of the vertex colours can be changed.

The "Face Select"-Buttons are meant for use with the UVEditor, especially usefull with the realtime engine but also to use UV-Textures for rendering. They become active if you enter the FaceSelectMode with **FKEY** or with the FaceSelectIcon in the 3DWindow header. These buttons display the settings for the *active face* when in FaceSelect mode (**FKEY** or the FaceSelect icon in the 3DWindow header).

When multiple faces are selected you have to use the button "Copy Drawmode" to assign the settings to all selected faces. More about the FaceSelect mode and using UV texture coordinates you can read in the UV-texturing chapter.

Tex

Faces with this attribute are rendered textured in the textured view and the realtime engine. If no texture is assigned to the face it will be rendered in a bright purple.

Tiles

Images can have a tile-mode assigned. In the ImageWindow header you can indicate how many tiles an Image will be sibdivided in. This button tells Blender to use this tilemode for the active face.

Light

The faces with this attribute are calculated with light in the realtime engine and the shaded views.

Invisible

This attribute makes faces invisible.

Collision

Faces with this attribute are taken into account for the real-time collision detection.

Shared

In Blender vertex colors are stored in each Face, thus allowing a different color for individual faces without having to add vertices. With this option, you can make sure that vertex colors are blended across faces if they share vertices.

Twoside

Faces with that attribute are rendered twosided.

ObColor

Each Object in Blender has an RGB color that can be animated with Ipo-curves. With this option the realtime engine uses this "ObColor" instead of the vertex colors.

Halo

The faces are rendered as halos, which means the normals always pointing to the camera.

Shadow

Faces with this option set acting as shadow in the realtime-engie. In fact the face 'drops' on the floor. So you have to make sure the normal of the face points to the Z-axis. The face has to be located in the center of the Object (or slightly above). Best effect gives a texture with an alpha channel.

Opaque

The color of the textured face is normally rendered as color.

Add

This option makes the face being rendered transparant. The color of the face is added to what has already being drawn, thus achieving a bright 'lightbeam'-like effect. Black areas in the texture are transparent, withe is full bright.

Alpha

Depending on the alpha channel of the image texture, the polygon is rendered transparant.

To copy the drawmodes from the active to the selected faces use these Buttons.

Copy DrawMode

Copy the drawmode.

Copy UV+tex

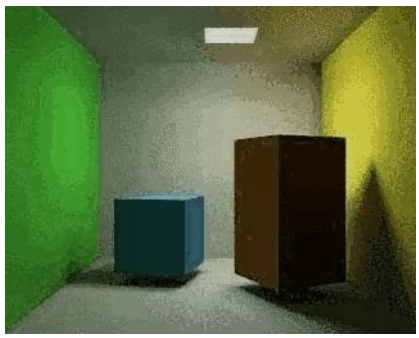
Copys UV information and textures.

Copy VertCol

Copys the vertex colors.

Introduction to radiosity

Most rendering models, including ray-tracing, assume a simplified spatial model, highly optimised for the light that enters our 'eye' in order to draw the image. You can add reflection and shadows to this model to achieve a more realistic result. Still, there's an important aspect missing! When a surface has a reflective light component, it not only shows up in our image, it also shines light at surfaces in its neighbourhood. And vice-versa. In fact, light bounces around in an environment until all light energy is absorbed (or has escaped!).



In closed environments, light energy is generated by 'emittors' and is accounted for by reflection or absorption of the surfaces in the environment. The rate at which energy leaves a surface is called the 'radiosity' of a surface.

Unlike conventional rendering methods, radiosity methods first calculate all light interactions in an environment in a view-independent way. Then, different views can be rendered in real-time.

In Blender, Radiosity is more of a modeling tool than a rendering tool. It is the integration of an external tool and still has all the properties (and limits) external tools.

The output of Radiosity is a Mesh Object with vertex colors. These can be retouched with the VertexPaint option or rendered using the Material properties "VertexCol" (light color) or "VColPaint" (material color). Even new Textures can be applied, and extra lamps and shadows added.

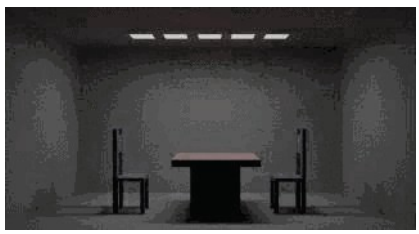
Currently the Radiosity system doesn't account for animated Radiosity solutions. It is meant basicall for static environments, real time (architectural) walkthroughs or just for fun to experiment with a simulation driven lighting system.

[subsection]GO! A quickstart

1. Load the file called `"radio.blend"` from the CD. You can see the new RadioButtons menu displayed already.
2. Press the button "Collect Meshes". Now the selected Meshes are converted into the primitives needed for the Radiosity calculation. Blender now has entered the Radiosity mode, and other editing functions are blocked until the button "Free Data" has been pressed.

3. Press the button "GO". First you will see a series of initialisation steps (at a P200, it takes a few seconds), and then the actual radiosity solution is calculated. The cursor counter displays the current step number. Theoretically, this process can continue for hours. Fortunately we are not very interested in the precise correctness of the solution, instead most environments display a satisfying result within a few minutes. To stop the solving process: press ESC.
4. Now the Gouraud shaded faces display the energy as vertex colors. You can clearly see the 'color bleeding' in the walls, the influence of a colored object near a neutral light-grey surface. In this phase you can do some postprocess editing to reduce the number of faces or filter the colors. These are described in detail in the next section.
5. To leave the Radiosity mode and save the results press "Replace Meshes" and "Free Radio Data". Now we have a new Mesh Object with vertex colors. There's also a new Material added with the right proprties to render it (Press F5 or F12).

The same steps can be done with the examples "room.blend" and "radio2.blend".



[subsection]The Blender Radiosity method

During the later eighties and early nineties radiosity was a hot topic in 3D computer graphics. Many different methods were developed. The most successful solutions were based at the "progressive refinement" method with an "adaptive subdivision" scheme. (Recomended further reading: the web is stuffed with articles about radiosity, and almpost every recent book about 3D graphics covers this area. The best still is "Computer Graphics" by Foley & van Dam et al.).

To be able to get the most out of the Blender Radiosity method, it is important to understand the following principles:

Finite Element Method

Many computer graphics or simulation methods assume a simplification of reality with 'finite elements'. For a visual attractive (and even scientifically proven) solution, it is not always necessary to dive into a molecular level of detail. Instead, you can reduce your problem to a finite number of representative and well-described elements. It is a common fact that such systems quickly converge into a stable and reliable solution. The Radiosity method is a typical example of a finite element method.

Patches and Elements

In the radiosity universe, we distinguish between two types of 3D faces:

1. Patches. These are triangles or squares which are able to *send energy*. For a fast solution it is important to have as few of these patches as possible. But, because the energy is only distributed from the Patch's center, the size should be small enough to make a realistic energy distribution. (For example, when a

small object is located above the Patch center, all energy the Patch sends then is obscured by this object).

2. Elements. These are the triangles or squares used to *receive energy*. Each Element is associated to a Patch. In fact, Patches are subdivided into many small Elements. When an element receives energy it absorbs part of it (depending on the Patch color) and passes the remainder to the Patch. Since the Elements are also the faces that we display, it is important to have them as small as possible, to express subtle shadow boundaries.

Progressive Refinement

This method starts with examining all available Patches. The Patch with the most 'unshot' energy is selected to shoot all its energy to the environment. The Elements in the environment recieve this energy, and add this to the 'unshot' energy of their associated Patches. Then the process starts again for the Patch NOW having the most unshot energy.

This continues for all the Patches until no energy is received anymore, or until the 'unshot' energy has converged below a certain value.

The hemicube method

The calculation of how much energy each Patch gives to an Element is done through the use of 'hemicubes'. Exactly located at the Patch's center, a hemicube consist of 5 small images of the environment. For each pixel in these images, a certain visible Element is color-coded, and the transmitted amount of energy can be calculated. Especially by the use of specialized hardware the hemicube method can be accellerated significantly. In Blender, however, hemicube calculations are done "in software".

This method is in fact a simplification and optimisation of the 'real' radiosity fomula (form factor differentiation). For that reason the resolution of the hemicube (the number of pixels of its images) is important to prevent aliasing artefacts.

Adaptive subdivision

Since the subdivision of a Mesh defines the quality of the Radiosity solution, automatic subdivision schemes have been developed to define the optimal size of Patches and Elements. Blender has two automatic subdivision methods:

1. Subdivide-shoot Patches. By shooting energy to the environment, and comparing the hemicube values with the actual mathematical 'form factor' value, errors can be detected that indicate a need for further subdivision of the Patch. The results are smaller Patches and a longer solving time, but a higher realism of the solution.
2. Subdivide-shoot Elements. By shooting energy to the environment, and detecting high energy changes (frequencies) inside a Patch, the Elements of this Patch are subdivided one exta level. The results are smaller Elements and a longer solving time and probably more aliasing, but a higher level of detail.

Display and Post Processing

Subdividing Elements in Blender is 'balanced', that means each Element differs a maximum of '1' subdivide level with its neighbours. This is important for a pleasant and correct display of the Radiosity solution with Gouraud shaded faces.

Usually after solving, the solution consists of thousands of small Elements. By filtering these and removing 'doubles', the number of Elements can be reduced significantly without destroying the quality of the Radiosity solution.

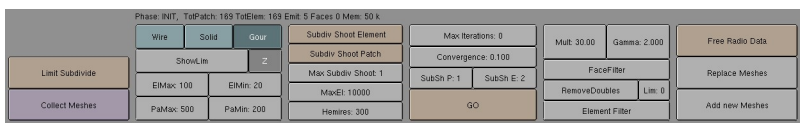
Blender stores the energy values in 'floating point' values. This makes settings for dramatic lighting situations possible, by changing the standard multiplying and gamma values.

Rendering and integration in the Blender environment

The final step can be replacing the input Meshes with the Radiosity solution (button "Replace Meshes"). At that moment the vertex colors are converted from a 'floating point' value to a 24 bits RGB value. The old Mesh Objects are deleted and replaced with one or more new Mesh Objects. You can then delete the Radiosity data with "Free Data". The new Objects get a default Material that allows immediate rendering. Two settings in a Material are important for working with vertex colors:

-
- VColPaint. This option treats vertex colors as a replacement for the normal RGB value in the Material. You have to add Lamps in order to see the radiosity colors. In fact, you can use Blender lighting and shadowing as usual, and still have a neat radiosity 'look' in the rendering.
- VertexCol. This option better should have been called "VertexLight". The vertex-colors are added to the light when rendering. Even without Lamps, you can see the result. With this option, the vertex colors are pre-multiplied by the Material RGB color. This allows fine-tuning of the amount of 'radiosity light' in the final rendering.

[subsection]The RadiosityButtons



As with everything in Blender, Radiosity settings are stored in a datablock. It is attached to a Scene, and each Scene in Blender can have a different Radiosity 'block'. Use this facility to divide complex environments into Scenes with independent Radiosity solvers.

Phase 1: preparing the models

Only Meshes in Blender are allowed as input for Radiosity. It is important to realize that **each** face in a Mesh becomes a Patch, and thus a potential energy emittor and reflector. Typically, large Patches send and receive more energy than small ones. It is

therefore important to have a well-balanced input model with Patches large enough to make a difference!

When you add extremely small faces, these will (almost) never receive enough energy to be noticed by the "progressive refinement" method, which only selects Patches with large amounts of unshot energy.

You assign Materials as usual to the input models. The RGB value of the Material defines the Patch color. The 'Emit' value of a Material defines if a Patch is loaded with energy at the start of the Radiosity simulation. The "Emit" value is multiplied with the area of a Patch to calculate the initial amount of unshot energy.

Textures in a Material are not taken account for.

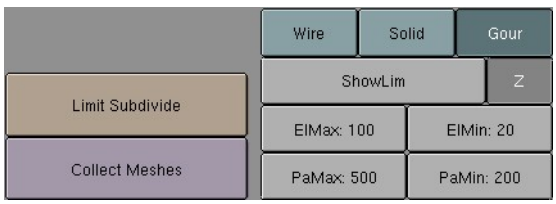
Collect Meshes

All selected and visible Meshes in the current Scene are converted to Patches. As a result some Buttons in the interface change color. Blender now has entered the Radiosity mode, and other editing functions are blocked until the button "Free Data" has been pressed. The "Phase" text prints the number of input patches. Important: check the number of "emit:" patches, if this is zero nothing interesting can happen!

Default, after the Meshes are collected, they are drawn in a pseudo lighting mode that clearly differs from the normal drawing. The 'collected' Meshes are not visible until "Free Radio Data" has been invoked.

Phase 2: subdivision limits.

Blender offers a few settings to define the minimum and maximum sizes of Patches and Elements.



Collect Meshes

You can always restart the entire Radiostiy process with this button.

Limit Subdivide

With respect to the values "PaMax" and "PaMin", the Patches are subdivided. This subdivision is also automatically performed when a "GO" action has started.

PaMax, PaMin, ElMax, ElMin

The maximum and minimum size of a Patch or Element. These limits are used during all Radiosity phases. The unit is expressed in 0,0001 of the boundbox size of the entire environment.

ShowLim, Z

This option visualizes the Patch and Element limits. By pressing the 'Z' option, the limits are drawn rotated differently. The white lines show the Patch limits, cyan lines show the Element limits.

Wire, Solid, Gour

Three drawmode options are included which draw independent of the indicated drawmode of a 3DWindow. Gouraud display is only performed after the Radiosity process has started.

Phase 3: adaptive subdividing, GO!

Hemires

The size of a hemicube; the color-coded images used to find the Elements that are visible from a 'shoot Patch', and thus receive energy. Hemicubes are not stored, but are recalculated each time for every Patch that shoots energy. The "Hemires" value determines the Radiosity quality and adds significantly to the solving time.

MaxEl

The maximum allowed number of Elements. Since Elements are subdivided automatically in Blender, the amount of used memory and the duration of the solving time can be controlled with this button. As a rule of thumb 20,000 elements take up 10 Mb memory.

Max Subdiv Shoot

The maximum number of shoot Patches that are evaluated for the "adaptive subdivision" (described below) . If zero, all Patches with 'Emit' value are evaluated.

Subdiv Shoot Patch

By shooting energy to the environment, errors can be detected that indicate a need for further subdivision of Patches. The subdivision is performed only once each time

you call this function. The results are smaller Patches and a longer solving time, but a higher realism of the solution. This option can also be automatically performed when the "GO" action has started. Subdiv Shoot Element (But) By shooting energy to the environment, and detecting high energy changes (frequencies) inside a Patch, the Elements of this Patch are selected to be subdivided one exta level. The subdivision is performed only once each time you call this function. The results are smaller Elements and a longer solving time and probably more aliasing, but a higher level of detail. This option can also be automatically performed when the "GO" action has started.

GO

With this button you start the Radiosity simulation. The phases are:

1.
2. Limit Subdivide. When Patches are too large, they are subdivided.
3. Subdiv Shoot Patch. The value of "SubSh P" defines the number of times the "Subdiv Shoot Patch" function is called. As a result, Patches are subdivided.
4. Subdiv Shoot Elem. The value of "SubSh E" defines the number of times the "Subdiv Shoot Element" function is called. As a result, Elements are subdivided.
5. Subdivide Elements. When Elements are still larger than the minimum size, they are sibdivided. Now, the maximum amount of memory is usually allocated.
6. Solve. This is the actual 'progressive refinement' method. The mousecursor displays the iteration step, the current total of Patches that shot their energy in the environment. This process continues until the unshot energy in the environment is lower than the "Convergence" or when the maximum number of iterations has been reached.
7. Convert to faces. The elements are converted to triangles or squares with 'anchored' edges, to make sure a pleasant not-discontinue Gouraud display is possible.

This process can be terminated with ESC during any phase.

SubSh P

The number of times the environment is tested to detect Patches that need subdivision. (See option: "Subdiv Shoot Patch").

SubSh E

The number of times the environment is tested to detect Elements that need subdivision. (See option: "Subdiv Shoot Element").

Convergence

When the amount of unshot energy in an environment is lower than this value, the Radiosity solving stops. The initial unshot energy in an environment is multiplied by the area of the Patches. During each iteration, some of the energy is absorped, or disappears when the environment is not a closed volume. In Blender's standard coordinate system a typical emittor (as in the example files) has a relative small area.

The convergence value in is divided by a factor of 1000 before testing for that reason .

Max iterations

When this button has a non-zero value, Radiosity solving stops after the indicated iteration step.

Phase 4: editing the solution

Element Filter

This option filters Elements to remove aliasing artefacts, to smooth shadow boundaries, or to force equalized colors for the "RemoveDoubles" option.

RemoveDoubles

When two neighbouring Elements have a displayed color that differs less than "Lim", the Elements are joined.

Lim

This value is used by the previous button. The unit is expressed in a standard 8 bits resolution; a color range from 0 - 255.

FaceFilter

Elements are converted to faces for display. A "FaceFilter" forces an extra smoothing in the displayed result, without changing the Element values themselves.

Mult, Gamma

The colorspace of the Radiosity solution is far more detailed than can be expressed with simple 24 bit RGB values. When Elements are converted to faces, their energy values are converted to an RGB color using the "Mult" and "Gamma" values. With the "Mult" value you can multiply the energy value, with "Gamma" you can change the contrast of the energy values.

Add New Meshes

The faces of the current displayed Radiosity solution are converted to Mesh Objects with vertex colors. A new Material is added that allows immediate rendering. The input-Meshes remain unchanged.

Replace Meshes

As previous, but the input-Meshes are removed.

Free Radio Data

All Patches, Elements and Faces are freed in Memory. You always must perform this action after using Radiosity to be able to return to normal editing.

ScriptButtons

ScriptLinks - Linking scripts to Blender

Python scripts can be attached to DataBlocks with the ScriptButtons window, and assigned events that define when they should be called.

The ScriptButtons are accessed via the icon in the the ButtonsWindow header.

ScriptLinks can be added for the following DataBlocks

- Objects - Available when an Object is active
- Cameras - Available when the active Object is a Camera
- Lamps - Available when the active Object is a Lamp
- Materials - Available when the active Object has a Material
- Worlds - Available when the current scene contains a World

When you are able to add a script link an icon appears in the header, similar to the ones that are used in the IPO Window. Selecting one of the icons brings up the ScriptLink buttons group in the left of the ScriptButtons window.

DataBlocks can have an arbitrary number of ScriptLinks attached to them - additional links can be added and deleted with the "New" and "Del" buttons, similar to Material Indices. Scripts are executed in order, beginning with the script linked at index one.

When you have at least one scriptlink the Event type and link buttons are displayed. The link button should be filled in with the name of the Text object to be executed. The Event type indicates at what point the script will be executed,

- **FrameChanged** - This event is executed every time the user changes frame, and during rendering and animation playback. To provide more user interaction this script is also executed continuously during editing for Objects.
- **Redraw** - This event is executed every time Blender redraws its Windows.

Scripts that are executed because of events being triggered receive additional input by objects in the Blender module.

- The **Blender.bylink** object is set to True to indicate that the script has been called by a ScriptLink (as opposed to the user pressing Alt-P in the Text window).
- The **Blender.link** object is set to contain the DataBlock which referenced the script, this may be a Material, Lamp, Object, etc.

- The **Blender.event** object is set to the name of the event which triggered the ScriptLink execution. This allows one script to be used to process different event types.

Scene ScriptLinks

The ScriptLink buttons for Scenes are always available in the ScriptButtons, and function exactly in the manner described above. Events available for SceneScriptLinks are:

- **FrameChanged** - This event is executed every time the user changes frame, and during rendering and animation playback.
- **OnLoad** - This event is executed whenever the Scene is loaded, ie. when the file is initially loaded, or when the user switches to the current scene.
- **Redraw** - This event is executed every time Blender redraws its Windows.

The RenderingButtons

This button field contains all the settings and commands that involve displaying and rendering. All of these data are part of the Scene block. They must thus be set separately for each Scene within a Blender file. Hotkey: F10.

In the header of the ButtonsWindow you get a SceneBrowse for that reason.

Pics

Enter the name of the directory to which the rendered image must be written if the "ANIM" command is given and, when required, the first few letters of the file name. Blender automatically assigns files a number, frame 1 becoming 0001. In this example, pictures are written as the files /render/0001, /render/0002 etc. If you enter "/render/rt" in the button, the files will be called /render/rt0001, /render/rt0002... Blender creates the specified directories itself if they do not already exist. The small square button to the left of the TextBut is used to invoke a FileSelect. Use it to select the output directory, and possibly a file name, and press ENTER to assign this to the TextBut .

Backbuf

Enter a name of a image file to be used as a background. Blender uses the rendered *alpha* to determine the extent to which this background is visible. A code can be processed into the file name, which allows an already rendered animation to be used as a background. Be sure that a '#' is placed at the end. This is replaced by the current (four-digit) frame number. For example: /render/work/rt.# becomes /render/work/rt.0101 at frame 101. The two small buttons to the left of the TextBut invoke an ImageSelect or a FileSelect Window. Specify the file and press ENTER to assign it to the TextBut.

BackBuf

Activate the use of a background picture.

Ftype

Use an "Ftype" file, to indicate that this file serves as an example for the type of graphics format in which Blender must save images. This method allows you to process 'color map' formats. The colormap data are read from the file and used to convert the available 24 or 32 bit graphics. If the option "RGBA" is specified, standard colour number '0' is used as the transparent colour. Blender reads and writes (Amiga) IFF, Targa, (SGI) Iris and CDi RLE colormap formats. Here, as well, the two small button to the left of the TextBut can be used to invoke an ImageSelect or a FileSelect window.

Extensions

Filename extensions (*.xxx) will be added to the filename, needed mostly for Windows systems.

Each Scene can use another Scene as a "Set". This specifies that the two Scenes must be integrated when rendered. The current Lamps then have an effect on both Scenes. The render settings of the Set, such as the Camera, the *layers* and the World used, are replaced by those of the current Scene. Objects that are linked to both Scenes are only rendered once. A Set is displayed with light gray lines in the 3DWindow. Objects from a Set cannot be selected here.

Window Location

These nine buttons visualise the standard position in which the Render Window appears on the screen. The setting in the example specifies that the Window must be at the top in the middle.

DispView

The rendering can also be displayed in the 3Dwindow instead of in a separate window. This occurs, independent of the resolution, precisely within the render borders of Camera *view*. Use F11 to remove or recall this image.

DispWin

The rendering occurs in a separate window. Use F11 to move this window to the foreground or background.

Edge

In a post-render process an outline will be added to the objects in the rendering. Together with special prepared materials, this causes a cartoon-like picture.

Eint:

Sets the intensity for the edge-rendering. Too high values causes outlining of single polygones.

Shift

With the unified renderer the outlines are shifted a bit.

All

Also consider transparent faces for edge-rendering with the unified renderer

RENDER

Start the rendering. This is a 'blocking' process. A square mouse cursor indicates that Blender is busy. Hotkey: F12. Rendering can also take place in the 'background'. Use the command line for this:

```
blender -b file.blend -f 100
```

This will render frame 100 and save the images to disk.

OSA

OverSampling. This option turns anti-aliasing on, to achieve 'soft' *edges* and perfectly displayed Image textures. OSA rendering generally takes 1.5 to 2 times longer than normal rendering.

5, 8, 11, 16

Blender uses a Delta Accumulation rendering system with *jitteredsampling*. These numbers are pre-sets that specify the number of *samples*; a higher value produces better *edges*, but slows down the rendering.

MBLUR

This option mimics a natural (or long) shutter time by accumulating multiple frames.

- The number-button "Bf:" defines the length of the shutter time.
- The value of "Osa" (5,8,11,16) defines the number of accumulated images.
- Setting the "OSA" option makes each acumulated image having antialising.

Xparts, Yparts

OSA rendering of large images, in particular, can take up a lot of memory. In addition to all the shadow buffers and texture maps and the faces themselves, this takes up 10 to 16 bytes per pixel. For a 2048x1024 picture, this requires a minimum of 32 Mb free memory. Use this option to subdivide the rendering into 'parts'. Each part is rendered independently and then the parts are combined. The "Xparts" are particularly important when rendering "Ztransp" faces.

Sky

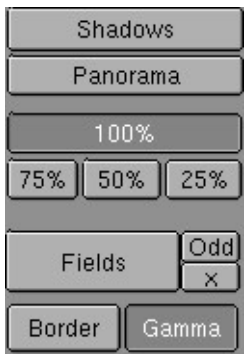
If a World has 'sky', this is filled in in the background. The alpha is not altered, but the transparent colours 'contaminate' the background colours, which makes the image less suitable for post-processing.

Premul

'Sky' is never filled in. The *alpha* in the picture is delivered as "Premul": a white pixel with *alpha* value 0.5 becomes: (RGBA bytes) 128, 128, 128, 128. The colour values are thus multiplied by the alpha value in advance. Use "Premul" alpha for post-processing such as filtering or *scaling*. Remember to select the "RGBA" option before saving. When Blender reads RGBA files, "Premul" is considered the standard.

Key

'Sky' is never filled in. The *alpha* and colour values remain unchanged. A white pixel with an *alpha* value of 0.5 becomes: (RGBA bytes) 255, 255, 255, 128. What this means is especially clear when rendering Halos: the complete transparency information is in the (hidden) *alpha* layer. Many drawing programs work better with "Key" alpha.



Shadows

This turns shadow rendering on. Shadow can only be created by Spot Lamps.

Panorama

Blender can render panoramas. To do this, a number of pictures are rendered, where the number in question corresponds with the value of "Xparts". For each 'part', the Camera is rotated in such a way that a continuous panorama is created. For the width

of the panorama of the Camera Lens, adjust the "Xparts" and the "SizeX" for the picture. The total width of the picture, in pixels becomes: Xparts * SizeX. These are the settings for a 360 degree panorama: Xparts = 8, SizeX = 720, lens = 38.6.

100%, 75%, 50%, 25%

These pre-sets allow you to render smaller pictures. It also affects the size of 'shadow buffers'.

Fields

Specifies that two separate *fields* are rendered. Each field is a complete picture. The two fields are merged together in such a way that a 'video frame' is created.

Odd

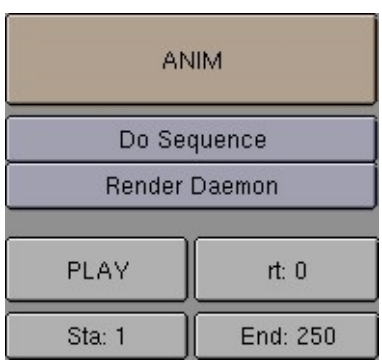
This option indicates that the first field in a video frame begins on the first line. x (TogBut) With "Field" rendering, this switches the time difference between the two fields off (0.5 frame).

Border

This allows you to render a small cut-out of the image. Specify a render 'border' with SHIFT+B in the 3DWindow (in Camera *view* of course). A cut-out is always inserted in a complete picture, including any "BackBuf" that may be present. Set the option "Crop" ON to turn this off.

Gamma

Colours cannot be normally added together without consequences, for example when rendering anti-aliasing. This limitation is caused by the way light is displayed by the screen: the colour value 0.4 does not appear half as strong as 0.8 (in actuality it is nearly 0.56!). This can be solved by assigning the display-hardware an extremely high gamma correction: gamma 2.3 or even higher. This gives a really pale image with 'washed out' dark tints to which dithering must be applied. Blender renders everything internally already gamma-corrected. This produces a more stable anti-aliasing for the eye, i.e. anti-aliasing that does not 'swim'. To see this difference, render a "Shadeless" white plane *with* OSA - and with and without "Gamma". The only time this option should be set to OFF is when Blender is used for *imagecomposition*.



ANIM

Start rendering a sequence. This is a 'blocking' process. A square mouse cursor indicates that Blender is busy. Animations can also be rendered in the 'background'. Use the command line for this: blender -b file.blend -a.

Do Sequence

Specifies that the current Sequence strips must be rendered. To prevent memory problems, the pictures of the complete Sequence system are released per rendering, except for the current frame.

RenderDeamon

Indicates to the external network render utility that the current Scene must be rendered.

Play

This starts an animation playback window. All files from the "Pics" directory are read and played.

rt

For debugging purposes.

Sta, End

The start and end frame of an ANIM rendering.

SizeX, SizeY

The size of the rendering in pixels. The actual value is also determined by the percentage buttons (100%, 75%, etc.).

AspX, AspY

The pixel relationship. The pixels in monitors and video cards are not usually exactly square. These numbers can be used to specify the relative dimension of a pixel.

These buttons specify the graphics file format in which images are saved.

AVI raw

Uncompressed AVI files. AVI is a commonly used format on Windows plattforms.

AVI jpeg

JPEG compessed AVI files.

Frs/sec

Framerate for the AVI formats.

Targa

This is a commonly used, RLE-compressed standard.

TgaRaw

Raw Targa output as alternative for standard (RLE) Targa. Needed for some broken software.

Iris

The standard for SGI software.

JPEG

This lossy format can produce strong compression. Use "Quality" to indicate how much compression you want.

Cosmo

Only for SGI: specifies that Cosmo hardware must be used to compress SGI Movies.

HamX

A self-developed 8 bits RLE format. Creates extremely compact files that can be displayed quickly. To be used only for the "Play" option.

Ftype

This switches the Ftype option ON. See the description of the "Ftype" TextBut.

Movie

Only for SGI: Blender writes SGI movies.

Quality

Specifies the quality of the JPEG compression. Also for Movies.

MaxSize

For Movies: the average maximum size of each Movie frame in Kbytes. The compression factor can then vary per frame.

Crop

Specifies that the "Border" rendering must not be inserted in the total image. For Sequences, this switches the automatic picture *scaling* off. If the pictures are enlarged, the outside edges are cut off.

BW

After rendering, the picture is converted to black & white. If possible, the results are saved in an 8 bit file.

RGB

The standard. This provides 24 bit graphics.

RGBA

If possible (not for JPEG), the *alpha* layer is also saved. This provides 32 bit graphics.

IRIZ

Only for Iris format graphics: the Zbuffer is added to the graphics as a 32 bit extra layer.

A number of presets: (In the future, this will be replaced by a user-defined script).

PAL

The European video standard: 720 x 576 pixels, 54 x 51 aspect.

Default

Like "PAL", but here the render settings are also set.

Preview

For preview rendering: 320 x 256 pixels.

PC

For standard PC graphics: 640 x 480 pixels.

PAL 16:9

Wide-screen PAL.

PANO

A standard Panorama setting.

FULL

For large screens: 1280 x 1024 pixels.

Appendix A. Hotkeys Quick Reference Table

This Appendix is an automated import of Joeri Kassenaar's original work, re-organized as a text database by Bastian Salmela.

Symbols

Works In		Works If		Causes	
	3D Window		An Object is selected		A Menu to Appear
	IPO Window		Mesh Vertex/vertices is/are selected		A Toolbox to Appear
	Sequence Window		IPO keys are selected		A File Select dialog
	Image Select Window		Sequence Strip is selected		A Menu to Appear
	Text Edit Window		To be on a Button Window	**Uses**	
	Object Mode		Data Selected		**LMB**
	Edit Modes		A Curve is selected		**RMB**
	Object or Edit Modes		IPO Handles selected		
	Pose Mode		A Render Window exists		
	Object Oriented Window		An Armature is selected		
	Render Window		An Object is selected. An Image Select Window Exists		
			In Object Mode		

TAB

Key	Works in	Action	Works if
TAB		Object mode / Edit mode	
TAB		Toggle Meta strip	
TAB		Object / Edit (unposed) mode	

Key	Works in	Action	Works if
CTRL-TAB		Object / Pose Mode	
SHIFT-TAB		Tab in Text Mode	

NUMPAD

Key	Works in	Action	Works if
. Del		Local view w/o moving cursor	
/		Local view & cursor / previous view	
*****		Rotate view to object orientation	
+ / -		Zoom in/out	
+ / -		Zoom in/out	
0		Increase PVE, (Grab/Rot./Scale)	
CTRL-0		Set Camera View	
ALT-0		Restore last camera to view	
1 3 7		Front / Right / Top view	
CTRL-1 3 7		Back / Left / Bottom view	
2 4 6 8		Rotate view	
5		Perspec-tive/orthographic view	
SHIFT-7		Zoom view to fit all objects	
9		Redraw	

NUMBERS

Key	Works in	Action	Works if
~		Display all layers	
SHIFT-~		Display all layers / previous layers	

Key	Works in	Action	Works if
CTRL-~		Lock/Unlock layers & camera to scene	
0-9		Swap layers 1-10	
SHIFT-0-9		Add/remove layers 1-10 to layer setting	
ALT-0-9		Swap layers 11-20	
SHIFT-ALT-0-9		Add/remove layers 11-20 to layer setting	
-		Swap layer 11	
SHIFT--		Add/remove layer 11 to layer setting	
=		Swap layer 12	
SHIFT-=		Add/remove layer 12 to layer setting	

Comma and Period

Key	Works in	Action	Works if
,		Rotation/scaling around bounding box	
.		Rotation/scaling around cursor [cursor]	

Arrow Keys

Key	Works in	Action	Works if
[up]/[dn]		10 frames forward/backward	
[rt]/[lt]		1 frame forward/backward	
SHIFT-[up]/[rt]		Jump to last frame	

Key	Works in	Action	Works if
SHIFT-[dn]/[lt]		Jump to first frame	
CTRL-[up]/[dn]		Active window full/small screen	
CTRL-[rt]/[lt]		Screen configuration forward/backward	

Arrow Keys - Grab/Rotate/Scale behaviour

Key	Works in	Action	Works if
[up]		Adjust arrow direction	
[up]		Adjust arrow direction (Grab coarse adjust)	
SHIFT-[up]		Adjust (grab fine adjust)	
CTRL-[up]		Adjust (grab coarse grid snap center)	
SHIFT-CTRL-[up]		Adjust (grab fine grid snap center)	

Mouse

Key	Works in	Action	Works if
lMB		Place cursor/gesture/vertex paint	
CTRL-lMB		Add vertex	
mMB		Trackball	
mMB		Translate view	
SHIFT-mMB		Translate view (see numpad c-2468)	
CTRL-mMB		Zoom view (see number +/-)	
rMB		Select	
SHIFT-rMB		Add to selection	

Key	Works in	Action	Works if
CTRL-rMB		Select object by closest object center	
SHIFT-CTRL-rMB		Add selection by closest object center	
CTRL-rMB		Select object & affect active object	
SHIFT-CTRL-rMB		Add to selection & affect active object	

A

Key	Works in	Action	Works if
A		Select / deselect All	
A		Select / deselect All	
SHIFT-A		Add menu	
CTRL-A		Apply location and rotation	
ALT-A		Play Animation	
SHIFT-CTRL-A		Apply Lattice / Duplicate	
SHIFT-ALT-A		Play Anim in current and all 3d windows	

B

Key	Works in	Action	Works if
B		Border select / deselect	
B		Border select / deselect	
BB		Circle Border select / deselect	
SHIFT-B		Define render Border	

C

Key	Works in	Action	Works if
C		Center windows around 3d cursor	
C		Snaps current frame to selected key	
C		Change images	
SHIFT-C		Cursor on origin, window on home	
CTRL-C		Copy menu	
ALT-C		Convert menu	

D

Key	Works in	Action	Works if
D		Duplicate	
SHIFT-D		Duplicate	
CTRL-D		Display alpha of images as wire	
ALT-D		Add data-linked duplicate	

E

Key	Works in	Action	Works if
E		Extrude (by grabbing-G returns if Rot./Scale)	
ER		Extrude by Rotating	
ES		Extrude by Scaling	

F

Key	Works in	Action	Works if

Key	Works in	Action	Works if
F		Make edge/face. / Connect curve	
F		Face select display on / off	
CTRL-F		Flip selected triangle edeges	
CTRL-F		Sort faces	
ALT-F		Beauty refill	
ALT-F		Make first base	
SHIFT-ALT-F		Save and open text files menu	

G

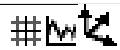
Key	Works in	Action	Works if
G		Grabber	
G		Grabber	
ALT-G		Clear location	
ALT-G		Clear location	

H

Key	Works in	Action	Works if
H		Handle type: align / free	
H		Hide selected vertices	
SHIFT-H		Handle type: auto (see also V)	
SHIFT-H		Hide deselected vertices	
CTRL-H		Automatic Handle calculation	

I

Key	Works in	Action	Works if

Key	Works in	Action	Works if
I		Insert Keyframe menu	

J

Key	Works in	Action	Works if
J		Swap render page of render window	
SHIFT-J		Join selected triangles to quads	
CTRL-J		Join selected objects (see also P)	
CTRL-J		Join selected keys	

K

Key	Works in	Action	Works if
K		Draw/hide object keys	
K		Show keys/show curves	
CTRL-K		Add skeleton to an IKA	
SHIFT-K		Clear vertexpaint colors	

L

Key	Works in	Action	Works if
L		Make local menu (see also U)	
L		Select vertices linked to cursor	
L		Select linked objects	
SHIFT-L		Select linked menu	
CTRL-L		Make link menu	

Key	Works in	Action	Works if
CTRL-L		Select vertices linked to selected vertex	
ALT-L		Make local menu	

M

Key	Works in	Action	Works if
M		Move to layer(s)	
M		Make a meta strip	

N

Key	Works in	Action	Works if
N		Number menu (numeric loc/rot/size entry)	
SHIFT-N		Recalculate normals outside	
SHIFT-CTRL-N		Recalculate normals inside	

O

Key	Works in	Action	Works if
O		Clear origin	
O		Normal / Proportional vertex edit (PVE)	
SHIFT-O		Sharp / Smooth falloff for PVE	
CTRL-O		Open file	

P

Key	Works in	Action	Works if

Key	Works in	Action	Works if
P		Separate vertices into objects (see also J)	
CTRL-P		Make parent	
CTRL-P		Make vertex parent	
ALT-P		Clear Parent menu	
ALT-P		Play a script in Text Edit window	
SHIFT-CTRL-P		Make parent without inverse	

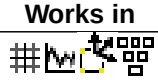
Q

Key	Works in	Action	Works if
Q		Quit Blender menu	

R

Key	Works in	Action	Works if
R		Rotate	
SHIFT-R		Select Row of nurbs	
ALT-R		Clear rotation	
ALT-R		Clear rotation	

S

Key	Works in	Action	Works if
S		Scale	
SHIFT-S		Snap-to menu	
CTRL-S		Shear	
ALT-S		Shrink/Fatten [rmb] function	
ALT-S		Clear Size	

Key	Works in	Action	Works if
S X		Mirror X	
S Y		Mirror Y	
SHIFT-ALT-S		Select text menu.	

T

Key	Works in	Action	Works if
T		Texture space menu (grab & rotate textures)	
T		Tilt of 3d curve (see F9 & press 3D button)	
CTRL-T		Convert to Triangles	
CTRL-T		Make Track-to	
ALT-T		Clear Track-to	
CTRL-ALT-T		Blenchmark	

U

Key	Works in	Action	Works if
U		Single User menu	
U		Reload data buffer. (undo)	
CTRL-U		Save current file as user default	

V

Key	Works in	Action	Works if
V		VertexPaint on / off	
V		Vector handle (see also H)	
ALT-V		Object resize to materials-texture aspect	

Key	Works in	Action	Works if
SHIFT-ALT-V		View menu for positioning cursor	

W

Key	Works in	Action	Works if
W		Special edit menu	
SHIFT-W		Warp selected vertices around cursor	
CTRL-W		Write file	
ALT-W		Write selected as videoscape format	

X

Key	Works in	Action	Works if
X		Erase menu	
CTRL-X		Delete all, reload default file (see also U)	

Y

Key	Works in	Action	Works if
Y		Split selected from the rest	

Z

Key	Works in	Action	Works if
Z		Wire / Solid view	
Z		Zoom/trans. render window (see F12 & F11)	
SHIFT-Z		Wire / Shaded view	

Key	Works in	Action	Works if
CTRL-Z		Shaded calculation view	
ALT-Z		Solid / Textured (potato) view	

Appendix B. Python Reference


(Python)

Appendix C. Command Line Arguments

Blender can also be run from the command line. In the following I will assume the you are in Blender's directory or that Blender executable is in your PATH. THis second option is the preferred one. Please refear to your friendly OS manual to know what a PATH is and how to have Blender in it.

Plain `blender` runs the whole thing just as if you clicked the Blender icon

Adding command line arguments can force different behaviours. The general syntax is.

```
blender [options] [file]
```

The basic option to remember is `-h`, that is the help option... once you remember this Blender will tell you all the others... a sort of magic word.

```
blender -h
```

Yields:

```
Blender V 2.28
Usage: blender [options ...] [file]

Render options:
  -b <file>     Render <file> in background
    -S <name>   Set scene <name>
    -f <frame>  Render frame <frame> and save it
    -s <frame>  Set start to frame <frame> (use with -a)
    -e <frame>  Set end to frame (use with -a)<frame>
    -a          Render animation

Animation options:
  -a <file(s)>  Playback <file(s)>
    -p <sx> <sy>        Open with lower left corner at <sx>, <sy>
    -m          Read from disk (Don't buffer)

Window options:
  -w            Force opening with borders
  -W            Force opening without borders
  -p <sx> <sy> <w> <h>  Open with lower left corner at <sx>, <sy>
                        and width and height <w>, <h>

Game Engine specific options:
  -g fixedtime          Run on 50 hertz without dropping frames
  -g vertexarrays       Use Vertex Arrays for rendering (usually faster)
  -g noaudio            No audio in Game Engine
  -g nomipmap           No Texture Mipmapping
  -g linearmipmap       Linear Texture Mipmapping instead of Nearest (default)

Misc options:
  -d            Turn debugging on
  -noaudio      Disable audio on systems that support audio
  -h            Print this help text
  -y            Disable OnLoad scene scripts, use -Y to find out why its -
y
  -R            Register .blend extension
```

Let's analyze these.

Render Options

The most important set of options. THey allow you to do `background rendering`. This implies that you render an image without having to run the Blender GUI. This uses some less memory and, usually, much less CPU time.

It is important to note that there are very few parameters which can be passed via the command line. Image dimensions, file tipes etc. need to have been set interactively before via the GUI.

To render a still image (assuming the still is in frame number 1):

```
blender -b yourfile.blend -f 1
```

To render an animation going from frame 1 to frame 100:

```
blender -b yourfile.blend -s 1 -e 100 -a
```

Animation Options:

These allows to play back an animation:

```
blender -a yourfile.avi
```

Useful if you used Blender built in Jpeg Codec and don't know how to play back the AVI.

Window Options

These options forces Blender to open its window with specified dimensions.

`-W` forces the default full-screen opening.

`-w` forces a non-full screen opening. If no other parameters are given the window occupies all the screen the same, but it is not 'maximized'. if the `-p` switch is given the the position and dimension of the window can be specified.

```
blender -w -p 128 128 1024 768
```

Forces Blender to open as a 1024x768 window with its lower left corner in (128,128)

Other Options

The Game Engine options are not covered in this manual.

The Miscellaneous Options turns on debugging features (`-d`), suppress audio (`-noaudio`), disable automatic running of 'On Load' scripts (`-y`) register '.blend' extension in the Operating System (`-R`) and provides the list of all options (`-h`).

Appendix D. Supported videocards

by Florian Findeiss

Graphics Compatibility

Blender requires a 3D accelerated graphics card that supports OpenGL. We strongly recommend making sure you are using the latest version of the drivers for your graphics card before attempting to run Blender. See the Upgrading section below if you are unsure how to upgrade your graphics drivers. Additionally here are some tips to try if you are having trouble running Blender, or if Blender is running with very low performance.

Most consumer graphics cards are optimized for 16-bit color mode (High Color). Try changing the color mode you are using in the Display Properties. Some cards many not be able to accelerate 3D at higher resolutions, try lowering your display resolution in the Display Properties. Some cards may also have problems accelerating 3D for multiple programs at a time - make sure Blender is the only 3D application running. If Blender runs but displays incorrectly, try lowering the hardware acceleration level in the Performance tab of the Advanced Display Properties .

Upgrading your Graphics Drivers

Graphics cards are generally marketed and sold by a different company than the one that makes the actual chipset that handles the graphics functionality. For example, a Diamond Viper V550 actually uses an NVidia TNT2 chipset, and a Hercules Prophet 4000XT uses a PowerVR Kyro chipset. Often both the card manufacturer and the chipset maker will offer drivers for your card, however, we recommend always using the drivers from the chipset maker, these are often released more frequently and of a higher quality. If you are not sure what chipset is in your graphics card consult the section on determining your graphics chipset. Once you know what chipset your graphics card uses, find the chipset maker in the table below, and follow the link to that companies driver page. From there you should be able to find the drivers for your particular chipset, as well as further instructions about how to install the driver.

Determing your graphics chipset

The easiest way of finding out what graphics chipset is used by your card is to consult the documentatioon (or the box) that came with your graphics card, often the chipset is listed somewhere (for example on the side of the box, or in the specifications page of the manual, or even in the title, ie. a "Leadtek WinFast GeForce 256"). If you are unable to find out what chipset your card uses from the documentation, follow the steps below. If you don't know what graphics card you have, go to the Display Properties dialog, select the Settings tab, and look for the Display field, where you should see the names of your monitor and graphics card. Often the graphics card will also display its name/model and a small logo when you power on the computer. Once you know what graphics card you h ave, the next step is to determine what chipset is used by the card. One way of finding this out is to look up the manufacturer in the card manufacturers table and follow the link to the manufacturers website, once there find the product page for your card model; somewhere on this page it should list the chipset that the card is based on. Now that you know the chipset your card uses, you can continue with the instructions in the Upgrading section.

Dispkay Properties

The display properties dialog has many usefull settings for changing the functioning of your graphics card. To open the display properties dialog, go to **Start Menu** -> **Settings** -> **Control Panel** and select the Display icon, or right-click on your desktop and select Properties.

Advanced display properties

The advanced display properties dialog has settings for controlling the function of your graphics driver, and often has additional settings for tweaking the 3D accelera-

tion. To open the advanced display properties dialog open the Display Properties as described above, then open the Settings tab, and click on the Advanced button in the lower right corner.

Table D-1. Card Manufacturers

Company	Commonly used chipset
3Dfx	3Dfx
AOpen	NVidia/SiS
Asus	NVidia
ATI	ATI
Creative	NVidia
Diamond Multimedia	NVidia/S3
Elsa	NVidia
Gainward	NVidia/S3
Gigabyte	NVidia
Hercules	NVidia/PowerVR
Leadtek	3DLabs/NVidia
Matrox	Matrox
Videologic	PowerVR/S3

Table D-2. Chipset Manufacturers

Company	Chipsets	Driver Page
3Dfx	Banshee/Voodoo	http://www.voodoof
3DLabs	Permedia	http://www.3dlabs.c
ATI	Rage/Radeon	http://mirror.ati.com
Intel	i740/i810/i815	http://developer.inte
Matrox	G200/G400/G450	http://www.matrox.c
NVidia	Vanta/Riva 128/Riva/TNT/GeForce	
PowerVR	KYRO/KYRO II	http://www.powervr
Rentition	Verite	http://www.micron.c

Company	Chipsets	Driver Page
S3 Graphics	Savage	http://www.s3graph
SiS	300/305/315/6326	http://www.sis.com/
Trident Microsystems	Blade/CyberBlade	http://www.tridentm

Graphics Compatibility Test Results

In the table good (or bad) performance refers to the speed of general 3d drawing and is a indication of how well a game will perform. Good (or bad) interactivity refers to how fast the interface responds on the graphics card, and is an indication of how well the graphics card works for creating and editing files. All tests are carried out with the latest drivers we could find. If results on your system do not match ours make sure you are using the latest drivers, as described in the Upgrading section.

Table D-3. Card Types

Chipset Manufacturer	Chipset Model	Windows 98	Windows 2000
3Dfx	Banshee	Works (very poor performance)	-untested-
3Dfx	Voodoo 3000	Good performance, Poor interactivity	Good performance, Poor interactivity
3Dfx	Voodoo 5500	Works (good performance)	-untested-
ATI	All-In-Wonder 128	Works (poor performance)	-untested-
ATI	Rage II 3D	Works (poor performance)	-untested-
ATI	Rage Pro 3D	Works (poor performance)	-untested-
ATI	Radeon DDR VIVO	Good performance, Good interactivity	Good performance, Good interactivity
Matrox	Millennium G200	Ok performance, Extremely poor interactivity, Some drawing errors	Ok performance, Very poor interactivity, Some drawing errors
Matrox	Millennium G400	Good performance, Poor interactivity	Good performance, Poor interactivity
Matrox	Millennium G450	Good performance, Very poor interactivity	Good performance, Very poor interactivity
NVidia	TNT	Good performance, Good interactivity	Good performance, Good interactivity
NVidia	Vanta	Good performance, Good interactivity	Good performance, Good interactivity

Chipset Manufacturer	Chipset Model	Windows 98	Windows 2000
NVidia	TNT2	Good performance, Good interactivity	Good performance, Good interactivity
NVidia	GeForce DDR	Works (good performance)	-untested-
NVidia	GeForce 2	Works (good performance)	-untested-
PowerVR	Kyro	Good performance, Good interactivity, Some drawing errors	Good performance, Good interactivity, Some drawing errors
Rendition	Verite 2200	Works (poor performance), Some drawing errors	-untested-
S3	Virge	Ok performance, Good interactivity	-untested-
S3	Trio 64	Works (poor performance)	-untested-
S3	Savage 4	-untested-	Works (poor performance)
SiS	6326	Works (poor performance)	-untested-

Appendix E. Documentation changelog

Appendix F. Blender changelog

2.28a

Known Problems

- The shadowbuffer size (LampButtons) has to be a value that can be divided by 64, especially for cases when you render 75% or 25% of the size.
- The Hemi lamp still doesn't support the new shaders.

General Fixes

- Windows versions now use the user 'home' directory for saving settings. This includes the Blender .B.blend file (for startup), which for backwards compatibility still is being read from the old location, if there's none available in the new location.
- Edge select now only works with CTRL-ALT-RightMouse. (ALT-RightMouse causes several window managers to act).
- Toon shading: now works with Sun lights as well.
- Toon shading options in MaterialButtons have a proper range now.
- Bug fix: the menu option 'reopen last' caused a crash (all systems).
- Bug fix: volume envelopes for audio strips now work in all spaces.

Python

Most work has been done on python for this update:

- Many debug messages have been removed.
- Added argv to the builtin sys Python module. Currently only argv[0] is there. This fixes the weird error that can make correct scritps fail.
- Fixed scriptlinking. Only when a script was linked to an Object, this functionality worked. Now scriptlinking works on other types of Objects and global scriptlinking works now too. This was an incompatibility problem.
- Changes in the Object module:
  - Added buildParts() method. This method forces computation of the particle system
  - Fix argument parsing in the getLocation() and similar methods. These functions were not compatible with the old implementation. Now they need to have a list of values.
  - Fixed crash in the getTracked() method.
  - Fixed crash in the getParent() method. Problem occurred when getParent was called the second time on the same Object.
  - Linking a Mesh datablock to an Object was not possible. Fixed
- Changes in the Ipo module:
  - Removed the pop up dialog "relative / absolute" when an ipo is changed.

    - Added EvaluateCurveOn(position, time) method. This method returns the value of the ipo curve number position at the given time.

- Changes in the NMesh module:
  - Added the addMaterial (mat) method.
  - Added an optional argument to the update() method. If the value in the argument is 1, the mesh normals are recalculated. Otherwise, no recalculation is done.
  - Fixed crash in the getVertexInfluences. This crash occurred when no bone was linked to the vertex.
  - Fixed bug #399 - crash in the Face(vertexlist) method.
  - Fixed crash when trying to rename newly created meshes.
- Changes in the Material module:
  - Missing specR, specG and specB variables. These variables where available in the previous Python API.
- Changes in the Image module:
  - Added variables to expose the image width, height and depth parameters.
- Changes in the Text module:
  - The filename attribute didn't check for NULL - causing incorrect behaviour.
- Changes in the Armature module:
  - Fix bug #433 - Bone.setQuat does not work. The method changed the head instead of the quat.
- Changes in the Scene module:
  - Added an optional argument to the method update(). When the value is 1, a full update is done to all objects. Otherwise, only the base list of objects is updated.
- Some cleanup to make modules more consistent internally.
- A lot of documentation updates to many modules.

2.28

General

- Included new default .B.blend file in Blender, with Material, Texture, World and 'draw faces' and 'draw edges' options set.
- When ButtonsWindow shows Material options (**F5**), selecting a Lamp automatically switches to Lamp shading settings (**F4**). And vice versa.
- Tooltips for window headers have been upgraded
- Typing a '\' in swedish keyboards works now.
- Fixed slow file reading of files with vertex deform groups.
- BUG FIX: When the preset directory menu in FileSelect was empty, it causing a nasty drawing error. (Typical when you fire up Blender and didn't save yet).
- BUG FIX: saving VRML 1.0 files had incorrect UV coordinates.

- Installation for MS Windows now works properly for all versions.
- Tooltips (buttons) now wait 0.3 seconds before popping up.
- Added mouse wheel support in Text Editor.

New features - Win32 and OSX

- Support for Quicktime has been added. Quicktime can be used to import (gif, tiff, psd) images and movies as textures, and allows you to render animations to the Quicktime movie format. (.mov)
- Please note! The selected Quicktime codec does *not* get saved in the blendfile. Each time you startup Blender you have to (re)select the codec from the codec dialog. Background rendering is possible, it displays a codec dialog first when you start a render process. And on OSX Blender freezes when you press the "Options" button in the codec dialog !!
- The rendermenu now shows which codec is selected for rendering.

Audio (new!)

- Blender now allows playback of audio, in 3D window for syncing, as well in the Sequence editor.
- Editing with audio strips in the Sequence editor
- Animation is synced to the audio (framedrop)
- Audio is also played in all other windows (including 3D windows)
- Check the full features of this system in this article: http://intrr.org/blender/audiosequencer.html
- Known problem: the Ipos for audio fading only playback in Sequencer
- Output to movies with audio is not supported yet; a "Mixdown" feature for writing the mix into a file is provided instead

Python (new!)

- The whole implementation was redone in pure C, as a mix of new code and updated parts of old bpython files;
- Some bugs were fixed, like material updating in NMesh;
- Already existing modules now have more data exposed: more variables and more method functions;
- The file selector is back, now in the Window module;
- There are new modules: Armature, Effect, Metaball and World;
- Active development, a mailing list where people interested in Blender Python scripting are invited to participate: http://www.blender.org/mailman/listinfo/bf-python
- API reference documentation available in beta state.
- Expect in the period after 2.28 a lot of interesting import/export modules. - For 2.29 is scheduled to have pulldown menus to activate scripts.

3DWindow editing

- ALT+RIGHTMOUSE in Mesh edit mode selects both vertices of an edge (this is not real edge-selection, but it's close!).
- Added 'draw edges' mode in F9 menu, it displays vertex selection in edges.
- Changed ugly 'draw faces' option into a nice transparent.
- Settings like 'draw faces' and 'draw edges' are now saved in a file.
- "insert Mesh key" (**I**) gives option to choose between Relative or Absolute (normal) keys.
- Reversed change in **SHIFT+O** menu behavior.
- in edit mode; press **O** to toggle proportional editing
- in edit mode; press **SHIFT+O** to toggle sharp/smooth proportional editing
- outside of edit mode; the **SHIFT+O** toggles subsurf on/off
- BUG FIX: **SHIFT+O** caused crash when no Object was active
- New "Group selection" menu (**SHIFT-G**), which offers the following options:
  1. *Children* - Selects all direct children of the active object
  2. *Immediate Children* - Select all children, children's children etc. of the active object
  3. *Activate Parent* - Makes the parent of the current object active
  4. *Objects on shared layers* - Selects all objects that share at least one layer with the active object
- Added 'Textured Solid' to the Draw mode popup (**DKEY**).
- "Turn table" view rotation now allows (with vertical movement) a twisted rotation as well. This to fix bug #364, causing a deviation error. Feedback on this feature might result in making this another user option though.
- Selecting objects has a different frontbuffer drawing handling now. Instead of drawing everything again, in all windows, it restricts to the active and previous active object. When there are more objects to be redrawn, it does a normal swap-buffers. Result is it all redraws & selects a lot faster in complex scenes.
- BUG FIX: Typing special characters in 3d Text Object didn't work.
- BUG FIX #149. Rendering the current view (view3d header, render) didn't render solid drawmode.
- Spherical mapping in FaceSelect mode works again (**U**).

Action Window

- Fixed 'HOME' button, it didn't work at all
- You now can zoom out much further
- BUG FIX: opening a new Action Window, or scaling it, displayed with wrong window matrix.
- Relative Vertex Keys (RVKs) now show up here as well, with options:
- RVK sliders. Pressing the little triangle next to the word 'sliders' in the channel names opens them up.
- NKEY in the area where the key block names are allows the user to change the name of the keyblock, and the max and min values of the RVK sliders.

- ability to visualize the keyframes for the IpoCurves when the object is selected.
- right mouse can be used to select the keys
- border select in the main area can be used to border select keys.
- **A** selects/deselects all of the keys
- **G** and **S** can be used to grab or scale the key selections.
- **X** deletes the selected keys.
- **D** duplicated the selected keys.
- **V**, **H**and **SHIFT-H** change the bezier handles for the selected keys.

NLA Window

- BUG FIX, #178, deleting action strips caused memory to be garbled, which caused a non-readable .blend file.
- BUG FIX, #152, data-browse didn't work for a **SHIFT+A** "add action"

Rendering

- Lamps have options to not give 'Diffuse' or 'Specular' light (F4 menu)
- New shaders (from Tuhopuu) integrated in Blender:
- MaterialButtons: layout changed a bit, the 'shader' options now are located together.
- Shader options are separated in 'diffuse' and 'specular'. You can combine them freely.
- Diffuse Lambert: is the old shader
- Diffuse Oren Nayar: new shader, gives sandy/silky/skinny material well
- Diffuse Toon: for cartoon render
- Specular Phong: new spec, traditional 70ies spec
- Specular CookTorr: a reduced version of cook torrance shading, does off specular peak well (is the old spec in Blender)
- Specular Blinn: new spec, same features as CookTorr, but with extra 'refraction' setting
- Specular Toon: new spec for cartoon render
- Default Blender starts with settings that render compatible, also for old files.
- Works correctly in shaded view and preview-render
- BUG FIX: In render window, pressing **Z** now allows zooming again.
- BUG FIX: 2.27 didn't render in background mode (blender -b).
- Blender in background render mode now reacts to **CTRL+C** again. You'll have to do it twice... first **CTRL+C** will send an internal break (exit render loops)
- New feature: rendering large images now displays correctly zoomed down in a render window. Not zooming down caused unpredictable slowdowns at various systems.
- F11 now nicely pops/pushes a render window again.
- DispView mode (render in 3d window) now doesn't disappear anymore after rendering

- While rendering, pressing a single **ESC** immediately halts rendering. It now works as in versions 2.23 and older, without causing slow rendering.
- BUG FIX: Memory leak which caused Blender to hog up more and more memory when rendering long animations. (An oldie!).

Quicktime support

- Updated Quicktime code so settings can be stored in the blendfile, works compatible for OSX and Windows. This enables Blender to;
- have scenes with different codec settings. (same as avicodec)
- render directly without dialog.
- batch/background render to Quicktime movies.

Radiosity

- BUG FIX #370; when over 16 materials are used for 'collect meshes', it used an illegal index number, causing errors in rendering.

OpenGL related: (mostly for OSX and Nvidia cards)

- Material preview (**F4**, **F5**) now nicely shows scanline progress of rendering
- Using the GUI in general is much faster now, by properly caching 'glFlush' and swapbuffer calls.
- Geforce in 'full scene antialias' mode doesn't garble the Blender interface anymore (Windows).
- Iconified windows do not get window focus anymore. (X11)

OSX related

- Vertex painting and Face selection modes now correctly work.
- BUG FIX: Selecting vertices in ImageWindow (UV coords) didn't display.
- A hand cursor now indicates you can move a window edge.
- New shiny Blender icon.
- Saving a screenshot for Blender works with **ALT+CTRL+F3** key. (Normal key is **CTRL+F3**, but OSX uses it).
- Cursor now moves with arrow keys again, while grabbing/rotating/scaling
- BUG FIX: After rendering, and pressing **ESC** or **F11**, the main window was inactive for input.

Source code

- `interface.c` has an extensive API doc now, with full description of the GUI system in Blender. Can be found in `bf-blender/blender/docs/`

- Soundsystem library moved to 'intern' (out of the game engine)
- New dependency to SDL library for compiling. Both system SDL (with sdl-config) and an SDL library in `/lib/` work.
- Compiling with gcc under Windows works now.
- And heaps and heaps of Makefile/projectfile/automake improvements.

2.27

New features - all platforms

- International Language support for the Blender interface. A translation system for tooltips, buttons and menus, with support for non-latin character sets.
- User definable (antialiased) fonts for the interface. (Freetype)
- A restyled userpreferences menu, which adds more space for all the new buttons.
- Autoskinning for Armatures, when parenting a Mesh to an Armature it gives a menu.
- In editmode, selected vertices can be merged with **Alt-M**. The merge option is also added to the **W** menu.
- Shadowbuffers now can have any size. (with number-button)
- Hotkeys are added for subsurfing objects. **Shift-O** toggles subsurf mode for an object and **Ctrl-1** to **Ctrl-4** will set the subsurf division with the keyboard.
- Freetype2 support added for importing almost every font format for a 3D text object.
- Pressing **Alt-M** in the texteditor window generates a 3D text from the current textfile. (up to 1000 characters)
- The texteditor window is enhanced with a rightmouse menu for opening and saving files.
- The windowtype button has been redesigned, it now displays a textual description
- A small button has been added that toggles the visibility of the pulldown menu titles.
- Rename/delete menu's are added to the file and image browsers. Try **N** and **R**/**X**.

New features - Win32 and OSX

- Support for Quicktime has been added. Quicktime can be used to import (gif, tiff, psd) images and movies as textures, and allows you to render animations to the Quicktime movie format. (.mov)
- Please note! The selected Quicktime codec does *not* get saved in the blendfile. Each time you startup Blender you have to (re)select the codec from the codec dialog. Background rendering is possible, it displays a codec dialog first when you start a render process. And on OSX Blender freezes when you press the "Options" button in the codec dialog !!
- The rendermenu now shows which codec is selected for rendering.

New features - Win32 only

- New application and installer icons.
- Clipboard support to copy/paste text from and to the texteditor. Try with **Alt-Shift-C** and **Alt-Shift-V**.
- A button has been added to the top header which allows runtime switching between windowed and fullscreen mode. Hotkeys for switching are **Alt-DOWNARROW** for windowed and **Alt-UPARROW** for fullscreen mode. Commandline options -w/-W for the two modes are also available.
- During animation playback, the framenumber gets displayed in the animwindow header.
- (Experimental) A button has been added which allows you to list all available codecs in the avi codec dialog, instead of listing only certified codecs.

Sourcecode

- All Dutch comments have been translated to English.

Bugfixes

Check the bug tracker for a complete listing: http://projects.blender.org/tracker/index.php?group_id=9&atid=125 [3]

A shortlist:

- Lattices deform correctly again
- Plugins for Windows work again
- OpenGL context bug in GHOST. Fixes crashes in combination with Linux drivers, the r200 DRI most notably.

Still not working

- FaceSelect mode in OSX causes bad redrawing of the backbuffer (it flashes)
- The Blender210 Python API. The only API which is available in 2.27 is the one that was used in 2.25
- New build system using configure (this is partially working but needs more work)
- Physics support in the game engine

2.26

New features

- Mouse wheel support for all windows. Two user settings are added in the top window for mouse wheel usage. The zoom direction [WZoom] and the number of lines that get scrolled [WLines] can be set here. The mouse wheel also works

to increase/decrease the size of the circle while selecting or doing proportional scaling.

- A "fake user" button next to the databrowse buttons has been added, to keep the data when it has been unlinked from an object. (Basically a shortcut for pressing **Shift+F4** and **F**)
- Added support for manipulating Ipo bezier handles through the action window. Selecting action keys and pressing **V**, **H**, or **Shift+H** modifies the handles of the keys in the same way it does in the IPO window.
- Pressing **C** in the action window scrolls the window so that the current frame is in the center.
- Extra selection support for the action window, including:
  - border select initiated in the channel names border selects the channels and constraint channels.
  - right click or border select initiated in the horizontal scroll causes blender to select all keys for the selected frames.
  - right click or border select in the vertical scroll causes blender to select all keys for the channel or constraint channels that are to the left of the selection.
- X-ray bones support for showing armature bones in shaded mode (disabled depth test). The new x-ray button is added in the edit buttons when the armature is selected.
- Pressing **T** in the action window changes the Ipo type (constant/linear/bezier) for the Ipo curves owned by the selected channels
- Simple shaded+wire and solid+wire draw modes. It basically draws the wireframe after drawing the model in solid/shaded/textured mode. The user can set this per object using the "Wire" button in the edit buttons window.
- A header button for drawing line numbers in the texteditor.
- Ctrl keystrokes for cut/copy/paste/undo/redo in the texteditor.
- NTSC render preset (Image size - 720x480, Aspect ratio - 10x11, 30 fps)

Bugfixes

- Make switching to bottom, back, and left view (**Shift+Numpad 7**, **Shift+Numpad 1** and **Shift+Numpad 3**) work when in camera view.
- Full screen by default under Unix.
- Ignore WM_DELETE_WINDOW on IRIX.
- Icons in the imageselect window get drawn properly.
- Alleviation of the symptoms of the "slow render bug" by polling less often for the escape key in the inner loops of the render code (makes the escape key less responsive though).
- RGBA Targa files get displayed correctly in The GIMP

Workarounds

- A small modification is made that prevents closing the renderwindow on Mac OSX, so the GUI won't become unresponsive.

- The Blender210 Python API. The only API which is available in 2.26 is the one that was used in 2.25
- New build system using configure (this is partially working but needs some work)
- Physics support in the game engine

Still Not working

- Plugins in MacOSX

Notes

1. http://intrr.org/blender/audiosequencer.html
2. http://www.blender.org/mailman/listinfo/bf-python
3. http://projects.blender.org/tracker/index.php?group_id=9&atid=125

Appendix G. Troubleshooting (-)

Appendix H. About the Blender Documentation Project

About the Blender Documentation Project (-)

(to be written)

Contributors (-)

This documentation is the effort of many people, besides those on the cover page, and it is the result of a long evolution of Blender documentation dating back many years and issues which were, themselves, the result of the effort of many people.

The following is an alphabetical list of all the people which contributed directly, by writing, or indirectly, by having written something in previous documentations later merged here.

- Matthew R. Duncan
- Martin Kleppmann
- Jason Nairn
- Martin Poirier
- Randall Rickert
- Willem Zwarthoed

How to submit your changes

If you would like to contribute changes or correction to the Blender documentation but you don't have write access to the CVS repository, then we prefer to receive them as *patches* to the DocBook/XML files. Assuming some basic knowledge about CVS usage, this is easy. Here's how you do it:

First of all, before you start working check out the latest CVS tree and work on those files directly. Changes are made to the CVS repository almost daily and you don't want to work on outdated files. Updates are fast as only the changes are downloaded to your workstation.

When you're ready, you issue the following command:

```
cvs diff -u -w BlenderManual.xml > BlenderManual.xml.diff
```

The differences between your local copy and the copy in the CVS repository will be stored in `BlenderManual.xml.diff`. If you want to submit multiple patches, please do a diff for each file separately and send them to the DocBoard maillist.

Getting your system ready for DocBook/XML (-)

(to be written) DocBook environment, XML editors, ...

Learning DocBook/XML

by Martin Middleton

Learning DocBook can seem a little scary at first, just like learning Blender. We're not going to go into too much detail about DocBook. There is an excellent on-line reference entitled *DocBook: The Definitive Guide* , available on-line. It can be found it

at http://docbook.org/tdg/en/html/docbook.html. The purpose of this section is to teach just enough about DocBook to make the reader dangerous.

One thing to be aware of: there are actually two versions of DocBook that are in widespread use. One is for DocBook using SGML. The other is for DocBook using XML. While there are only a few differences between them, we will confine our discussion to the XML version of DocBook.

A Basic DocBook File

We will begin with an example DocBook document and explain what each piece of the document does. For the rest of this discussion refer to A Docbook example, BlenderManual.xml below.

```
<?xml version="1.0"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN" "docbook-xml-4.2/docbookx.dtd"
[

<!ENTITY bver "2.27">

<!ENTITY chapter_introduction   SYSTEM "chapters/introduction/chapter_introduction.xml"
<!ENTITY chapter_installation   SYSTEM "chapters/introduction/chapter_installation.xml"
<!ENTITY chapter_interface      SYSTEM "chapters/interface/chapter_interface.xml">

<!--
Everything between the line above and the line below is treated as a comment.
-->

<!ENTITY FIXME '<inlinegraphic format="PNG" fileref="gfx/general/undercon.png"/>'>
<!ENTITY NEEDHELP '<inlinegraphic format="PNG" fileref="gfx/general/undercon.png"/>'>

]>

<book>
  <!-- PART 1 -Introduction -->
  <part>
    &chapter_introduction;
    &chapter_installation;
    &chapter_interface;
  </part>
</book>
```

The XML Declaration

At the top of the file is the following line:

```
<?xml version="1.0"?>
```

This is known as the XML declaration. All XML documents should begin with an XML declaration.

The DOCTYPE Declaration

The second line is the Document Type Declaration.

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN" "docbook-xml-4.2/docbookx.dtd"
[
```

This tells the XML interpreter several things. The second word `book` tells the interpreter what type of document this file is. In this case it is a book, which is a concept from the DocBook DTD.

The end of this line contains information about the DTD we are using. The string has a bunch of information about the DTD. Most of the information doesn't matter to you, just know that we currently use DocBook version 4.2.

The DocBook Hierachy

The basic hierarchy that we use for DocBook looks something like this:

```
<set>               // A Set of Books
  <book>            // A Book
    <part>          // A division of a Book. Each division can contain a num-
ber of elements, such as chapters.
      <chapter>     // A chapter of a book, can also be its own file
        <section>   // A section of a chapter, chapter 1 section 1 will be de-
noted by 1.1 or something like that
          <section>  // sections can be nested creating subsections and sub-
subsections, etc.
            <para>   // a paragraph, the place text actually resides. Para-
graphs can be at any level below set,
       // text can also reside in lists, glossentries, etc.
```

Entities

The lines after the Document Type declaration define "Entities" for the DocBook document. Entities are basically handy shortcuts.

```
<!ENTITY bver "2.27">

<!ENTITY chapter_introduction   SYSTEM "chapters/introduction/chapter_introduction.xml"
<!ENTITY chapter_installation   SYSTEM "chapters/introduction/chapter_installation.xml"
```

The first line declares an entity called `bver`, which is just the text "2.27". In order to to put that piece of text anywhere in the document all you have to do is write `&bver;` For example here: 2.28.

Below the general entities are SYSTEM entities. These refer to system resources. In this case they are used to locate the different files that will come together to form the book. The first entity, `chapter_introduction` contains the introductory chapter of the book. The content of this chapter is stored in the file `chapter_introduction.xml` which is located in the `chapters/introduction` subdirectory. Notice that the path supplied is a local path, that is `chapter_introduction.xml` and `chapter_installation.xml` must be in the same directory in order for the DocBook interpreter to work properly.

The Actual Document

After all the entities are declared, the book can be put together. It begins with a <book> tag. A book is required to have a title, which is surrounded by <title> tags. When this document is interpreted, the interpreter substitutes the contents of the file referred to by <chapter_introduction>, chapters/introduction/chapter_introduction.xml, into the main XML file, BlenderManual.xml. This is followed by the contents of chapters/introduction/chapter_installation.xml.

Each tag must be closed with a tag of the same name except that there is a forward slash between the open bracket and the name of the tag. Toward the end of the BlenderManual.xml file the book tag is closed off with </book>. Since the first tag that was opened is now closed, the XML document is finished.

Common Tags

This section discusses some common tags and explains how they are generally used. There are a lot of tags available (300+) which means that most documents will use only a small subset of all the tags. Detailed tag explanations can be found at http://docbook.org/tdg/en/html/docbook.html. Some of the tags are obviously designed for very specific purposes such as <glosssee> which refers one glossary entry to another glossary entry. Others are less specific, such that several options will achieve basically the same stylistic interpretation.

A good example of this are the tags <literal>, <userinput>, and <computeroutput>. Under most circumstances these are interpreted in basically the same way.

However, it is important to realize that this is simply how the interpreter *chooses* to format these tags. Another interpreter, or another style sheet for the same interpreter, could display these tags in very different ways. Therefore, it is important to use the tag that most closely matches your meaning. Use <userinput> when specifying something a user must input, <computeroutput> when specifying the output of a machine, and <literal> when expressing a literal value.

When listing tags below, specific examples of the contexts in which they are useful will be provided. In reality there aren't too many tags which need to be used. One can simply tell a reader that a specific word is a filename without putting it in a <filename> tag. However, the richness of the document increases when more tags are supplied and it is easier to tell very specifically what something is if a section or piece of text is clearly identified.

Document Structure

As mentioned above there is a basic structure to DocBook documents. The tags that give structure to a DocBook document are outlined below:

<set>

The top level of the DocBook documentation. A set of books. It is unlikely that you will ever have to work with a <set> tag, it is included here for general reference purposes.

<book>

This is exactly the same as a physical book. It is a volume that may be distributed on its own or as part of a set. In the Blender Manual documentation, the <book> tag is only used in the top-level file BlenderManual.xml.

`<chapter>`

Exactly what the name suggests - a chapter in a book. This can be a separate file, or part of the overall book file. Each chapter in the Blender documentation is generally a separate file.

`<section>`

This is a part of a chapter. By nesting `<section>` tags it is possible to create subsections and sub-subsections. Generally, when writing a file that will be part of a larger file, it is of the section variety.

`<para>`

This is a paragraph. As a rule of thumb, nearly all text entered should be between `<para>` tags. When there are two different paragraphs, they must be enclosed in two separate sets of paragraph tags. Paragraph tags, like paragraphs, cannot be nested.

Linking

One of the features of DocBook is the ability to create links between sections, text, whatever at will. This means that documentation can be produced with very little replication. If the reader is to be able to easily reference a section as background material, a link to it can be provided. If the reader already knows the background material they can chose not to follow the link.

Setting up a Target

In order to create a link in a document two things are needed. The first is an instruction for DocBook to create a link from a piece of text. The second is a target for the link. To create a target, a tag must have an id. For example, to link to a specific section, change the `<section>` tag to `<section id="section.link">` where “section.link” is just an arbitrary identifier.

Note: The identifier that is used as a target should be text with period separators. If it is a section it is probably best to use something of the form “something.section”, etc. Be sure to follow the recommendations of the Blender Documentation Board for the Section called *Cross references* creating section names if creating documents for the Blender Manual.

Links

Just as there are multiple ways to put targets into your document, there are multiple ways of linking to those targets (as there are multiple ways to do anything in DocBook). Here are the most useful link tags:

`<link>`

Just a plain old link. In order to link to a specific section the syntax would look something like `<link linkend="specific.section">`**`clickable text`**`</link>`. This will allow a user to click on “clickable text” which will take them to the specific section you have identified as “specific section”.

`<ulink>`

This tag allows you to link to a Uniform Resource Identifier (URI) outside of the document itself. Generally this will be a Uniform Resource Locator (URL)

where a reader can get more information about a subject in your document. The usage looks like this: `<ulink url="http://www.blender.com">` **`Blender.org`**`</ulink>`. If you need to link to a local file, follow this convention `<ulink url="file://localhost/path/to/filename">`**`Some text here`**`</ulink>` to be sure that it can be viewed in most browsers.

`<xref>`

Xref is much like link except that you can't put any text between the opening and closing tags. You can specify what text will be clickable in two ways. The first is to supply whatever you are linking to with an `XRefLabel`. Thus, the example of a section target above would become `<section id="section.link" xreflabel="label">` . The other way can be used when dealing with a tag that requires you to have a `<title>` such as chapter. Assume that we have a chapter defined as follows:

```
<chapterid="specific.chapter">
  <title id="specific.chapter.title">Title</title>
</chapter>
```

In order to reference this chapter with an xref you would simply write `<xref linkend="specific.chapter" endterm="specific.chapter.title"/>`. The word "Title" would appear as clickable in the midst of your text. The same technique can be used with `<link>`.

> **Note:** Please notice the trailing slash at the end of the "xref" example in the preceding paragraph. Failure to include this slash can cause hours of pain when trying to debug parsing errors in your documents. The same is true of the "anchor" tag.

Lists

There are a total of seven types of lists that you can put in DocBook documents. We will only discuss 3 of them. They are:

Itemized List

A plain old bulleted list. Here is an example of an itemized list.

```
<orderedlist>
  <listitem>
    <para>
      Download the latest version of Blender.
    </para>
  </listitem>
  <listitem>
    <para>
      Study the Blender Manual.
    </para>
  </listitem>
  <listitem>
    <para>
      Create an awesome gallery.
    </para>
  </listitem>
```

```
  <listitem>
    <para>
      Write my Blender Screenplay.
    </para>
  </listitem>
</orderedlist>
```

Here is what it looks like after it has been formatted:

- Download the latest version of Blender.
- Study the Blender Manual.
- Create an awesome gallery.
- Write my Blender Screenplay.

Ordered List

A numbered list. Here is an example:

```
<orderedlist>
  <listitem>
    <para>Start Blender</para>
  </listitem>
  <listitem>
    <para>
      Type <userinput>SPACEBAR</userinput> to open the Add Menu.
    </para>
  </listitem>
  <listitem>
    <para>
      Next, click with the LMB (left mouse button) on the menu-entry <quote>MESH</q
      different possibilities for adding mesh-objects.
    </para>
  </listitem>
  <listitem>
    <para>
      Select <quote>Plane</quote>, the Toolbox disappears and a Plane ap-
pears at the 3D cursor. We are currently in the EditMode
      where we can edit the shape of the mesh itself.
    </para>
  </listitem>
</orderedlist>
```

Here is what the example above looks like when formatted.

1. Start Blender
2. Type **SPACEBAR** to open the Add Menu.
3. Next, click with the LMB (left mouse button) on the menu-entry “MESH”. The menu changes and will show the different possibilities for adding mesh-objects.
4. Select “Plane”, the Toolbox disappears and a Plane appears at the 3D cursor. We are currently in the EditMode where we can edit the shape of the mesh itself.

Variable List

This is a list that appears as a series of terms and their definition. It is similar to the style that a glossary is in, and is the style of this very list. Here is an example:

```
<variablelist>
  <varlistentry>
    <term><sgmltag>&lt;quote&gt;</sgmltag></term>
    <listitem>
      <para>
        This puts the text between the quote tags in
        quotes. The same effect can be derived by putting
        double quotes (") around a word in a text mode.
        However, using tags is more proper and will ensure that
        the quotes are properly represented no matter what
        country your document is published in.
      </para>
    </listitem>
  </varlistentry>
  <varlistentry>
    <term><sgmltag>&lt;emphasis&gt;</sgmltag></term>
    <listitem>
      <para>
        Emphasizes the text through the use of italics or
        boldface type.
      </para>
    </listitem>
  </varlistentry>
  <varlistentry>
    <term><sgmltag>&lt;footnote&gt;</sgmltag></term>
    <listitem>
      <para>
        Inserts a footnote<footnote>
        <para>Which looks something like this.</para>
        </footnote> at the point in the text where the opening
        tag is and then the actual text of the footnote (which
        is contained between the footnote tags, and generally
        between para tags) at the bottom of the page.
      </para>
    </listitem>
  </varlistentry>
</variablelist>
```

As I mentioned above, there are four other types of list that I have not mentioned here. These are `CalloutList`, `GlossList`, `SegmentedList` and `SimpleList`. For more details on this list go to the DocBook documentation[3].

Formatting

There are a few other stylistic tags that are helpful. They are listed below in no particular order.

`<quote>`

This puts the text between the quote tags in quotes. The same effect can be derived by putting double quotes (") around a word in a free text. However, using

tags ensures that the quotes are properly represented no matter what country your document is published in.

`<emphasis>`

Emphasizes the text through the use of italics or boldface type.

`<email>`

Inserts the email address between the tags into the document, hyper-text linked if possible.

`<userinput>`

Gives a consistent form to the text that a user should enter into a computer program.

`<computeroutput>`

Gives a consistent form to the text that a computer application might output.

`<note>`

Separates the text in note from the rest of the document and calls attention to it.

`<Warning>`

Much like a note, but calls greater attention to the text of the warning.

`<Tip>`

Like a warning and a Note, but calls attention to the text in a different way. Which one of these tags you choose should depend on the actual meaning of the text you want to enclose with these tags. If it is a note, use note; a tip, use tip, etc. `<caution>` and `<important>` also fall into this class of tags.

`<filename>`

This gives a consistent formatting to filenames and directories. A filename should just be enclosed in `<filename>` tags, whereas a directory should have an opening filename tag that looks like this: `<filename class="directory">`.

`<literal>`

Presents the content of this tag as a literal value. Generally used on numbers rather than text you want to appear literally on the screen.

An additional useful bit of formatting is the ability to display literal text without any sort of interpretation of your markup. There are several ways to do this, the one I have chosen looks like this:

```
<screen>
  <![CDATA[
   <chapter id="specific.chapter">
     <title id="specific.chapter.title">Title</title>
   </chapter>
  ]]>
</screen>
```

Here is what the above example looks like once formatted.

The Blender Documentation Project styleguide

To maintain consistency throughout all Blender Documentation, prospective authors are kindly requested to abide by the present Style Guide.

General Guidelines

Blender documentation should be written in clear, concise, idiomatic and correct English. It should be adequately subdivided into chapter, sections and subsections.

The logical subdivision of Core documentation is dictated by the Documentation Board.

The logical subdivision of contributed Tutorials is left to each author, but it is strongly advised that the authors provide a list of up to five keywords from the Section called *Keywords* to classify their work. Please type Keywords exactly since they will be indexed for searches.

Tutorials should contain an abstract of up to 300 words briefly describing topic, aims and contents for quick browsing.

Images Guidelines

The use of images in the documentation is essential. The PNG and JPG formats are highly preferred. GIF and other non-free formats are prohibited. Uncompressed formats like TGA are discouraged.

Floating Images

Float images should bear a caption and be referenced in the text. Please refrain from wording like *the next picture* or *the following figure*. Use cross references. Cross references are described the Section called *Cross references*

The usage of unreferenced images is discouraged. If you have an image you don't know how to reference either the image is unnecessary or your text is unclear.

The Documentation, both Core and Tutorials, is to be published both on the web and as printable matter. Dimension/resolutions should be, at maximum, as reported in Table H-1 (see.. crossreferences!)

Table H-1. Image resolution/dimensions

Width [pixels]	Height [pixels]	Resolution [dpi]
1000	768	150
800	600	125
640	512	100

These resolutions/sizes guarantee that printed images won't be larger than 17cm and hence fill an A4 printed page.

The use of images larger than 800 pixels is strongly discouraged anyway, since they are too wide for comfortable reading on a web browser.

One of Blender's most prominent features is that the Interface is fully OpenGL and scalable. This is great, but will unfortunately result in many differences if a number of users resort to screen dumps to show peculiar material settings, texture settings and the like.

To allow for both clarity and uniformity you should dimension the interface so that the RED slider in the material window is 18 pixels high. This is what Blender produces if you use a 1024x768 screen resolution and press the 'home' button to set the

default button size.

If you use a larger resolution, please drop down to 1024x768 for screen capturing. I hope no one out there has smaller screens!

Screen captures must come in a LOSSLESS format, so use PNG.

Inline Images

Inline images showing Blender icons are welcome and make description much clear. Since these are STANDARD images, please refrain from making your own, but download the complete set provided by the Documentation Board

Inline Smileys

Official Documentation is not a place for smileys. (Humour is another matter, of course you can be humorous!)

Tables

Tables are an appropriate way to display lots of data in a clear structured way. They can be a valid alternative to screen dumps to show some setups.

Tables, like float figures must have a caption and must be referenced in the text.

Code

DocBook has its nice environment for Python/C/whateverlanguage code chunks.

Code chunks, like float figures must have a caption and must be referenced in the text.

Documentation Style in Practice

This Section shows in detail an example of what we expect your XML to be.

Images Examples

Here's how to insert images in your doc

Floating Images

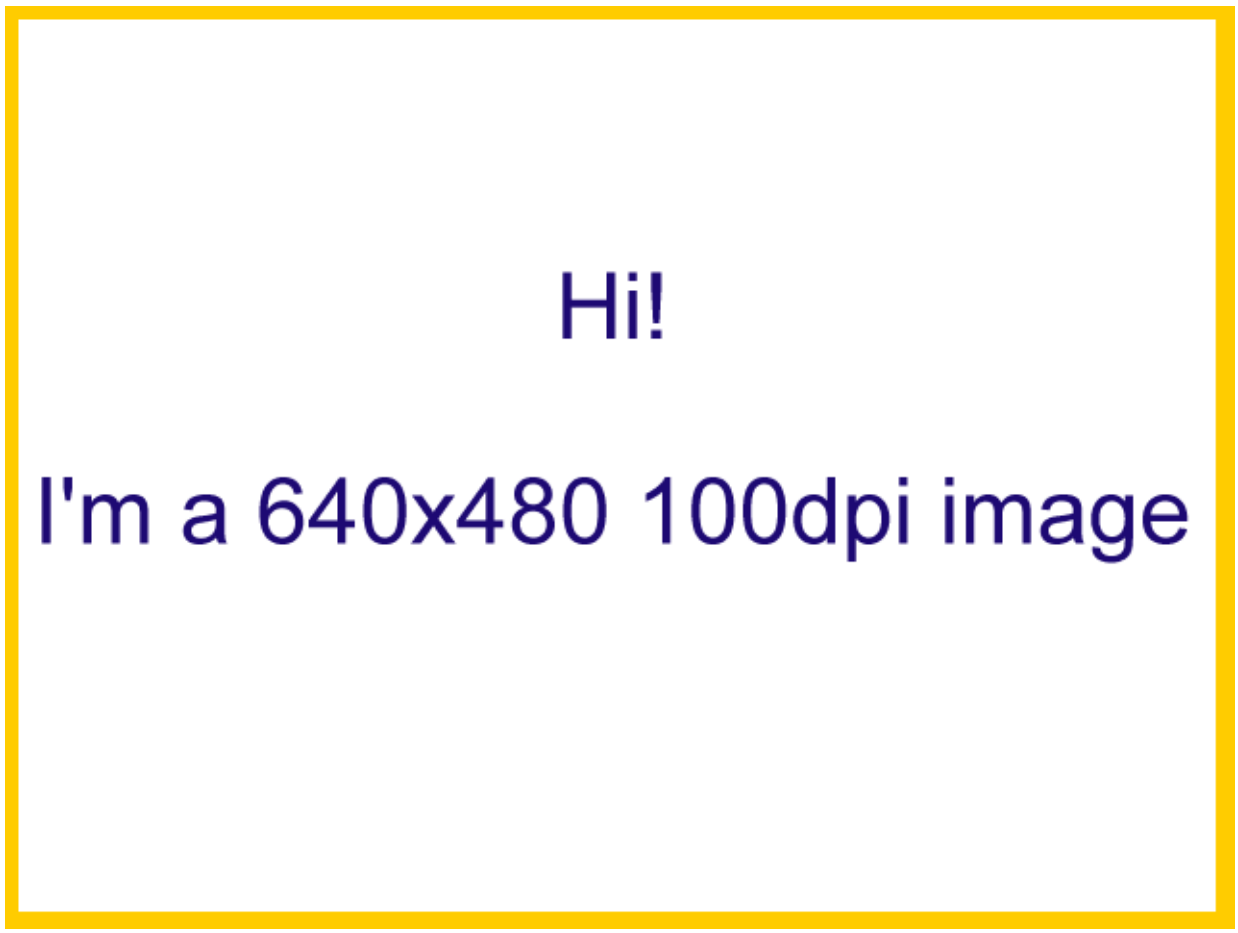


Figure H-1. A demo float image, with its appropriate caption

The Figure H-1 figure is included with the code in Example H-1. Please pay attention to the `Float="1"` attribute. This is strongly encouraged, since it produces much better output in Hard Copy printout.

Example H-1. How to include a Float Figure

```
<figure id="BSG.F.001" float="1">
  <title>A demo float image, with its appropriate caption</title>
  <graphic fileref="gfx/appendix_documentation_project/demoimage1.png"/>
</figure>
```

Inline Images

Inline Images like  are obtained via Example H-2

Example H-2. How to include an Inline Figure

```
<guiicon>
  <inlinegraphic fileref="gfx/appendix_documentation_project/demoimage2.png"></inlinegr
</guiicon>
```

Tables

Tables as in Table H-1 (see.. crossreferences across pages!) are obtained via Example H-3

Example H-3. How to include a Table

```
<table frame="all" id="BSG.T.001"><title>Sample Table</title>
  <tgroup cols="3" align="center" colsep="1" rowsep="1">
    <thead>
      <row>
        <entry>Width [pixels]</entry>
        <entry>Height [pixels]</entry>
        <entry>Resolution [dpi]</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry>1000</entry>
        <entry>768</entry>
        <entry>150</entry>
      </row>
      <row>
        <entry>800</entry>
        <entry>600</entry>
        <entry>125</entry>
      </row>
      <row>
        <entry>640</entry>
        <entry>512</entry>
        <entry>100</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

Yeah! I've seen easier things... (even LaTeX!)

Code

All the above examples of XML code where printed out by encapsulating them as shown in Example H-4

Example H-4. How to include Code

```
<example id="BSG.L.004"><title>How to include Code</title>
<programlisting><![CDATA[
    ****YOUR CODE HERE****
]]>
</programlisting>
</example>
```

The `<![CDATA[` and `]]>` pair disable the XML interpretation of anything in between, it is necessary only if you actually key in XML code (like I'm doing) and is superfluous for other programming languages.

If you want to have inline code like `this` and the ones above you should use a `<literal>text</literal>` pair, with or without a `CDATA` wrapper.

Cross references

If you noticed those `id="TheObjectLabel"` attributes in Figures, tables and code listings, you have already understood half of how cross-referencing works.

Those `id="something"` attributes are unique labels identifying those objects. The other half of the task is understand how to use them for actual referencing.

This is done via a plain `<xref linkend="TheObjectLabel"/>`.

Since labels are to be UNIQUE it is advisable to follow the guidelines provided in the relative appendix the Section called *Cross Reference Labelling Guidelines*.

Keywords

Keywords are an essential tool for categorizing and efficient searching. The following list is of course not static. if you feel something is missing please warn the authors.

- Animation
  - Armatures
    - Forward Kinematics
    - Inverse Kinematics
  - Keyframes Animation
  - Path Animation
  - Relative Vertex Keys Animation
  - Vertex Keys Animation
- Interface
- Lighting
  - Fake Global Illumination
  - Radiosity
- Modeling
  - Bezier Splines

    - Meshes
    - NURBS
    - SubSurfed Meshes

- Materials
- Realtime
- Rendering
- Sequence Editor
- Texturing
    - Built in Procedural
    - Plugins
    - UV Texturing

- World Settings

Cross Reference Labelling Guidelines

Labels are a delicate matter inasmuch they must be unique throughout all the Blender Documentation Project to allow for cross referencing from one chapter to another etc.

It is thus highly advisable to stick to a given standard:

For Core documentation

`PID.CID.[F|T|E|D|L].AID.userdefined`

For Tutorials

`TID.AID.userdefined`

Where

- `PID` is a two letter Publication Identifier;
- `CID` is a three letter Chapter Identifier;
- `[F|T|E|D|L]` is a letter defining the Type of object labelled, that is:
    - Figure;
    - Table;
    - Equation;
    - Document Data (i.e. a Section, subsection, paragraph etc);
    - Listate;

- `AID` is a three letter Author Identifier;
- `userdefined` are (up to) ten letters for the author to define;
- `TID` is a three letter Tutorial Identifier;

`PID` and `CID` are provided by the DocBoard.

`TID` and `AID` are to be agreed between the author and the DocBoard in a preliminary phase.

The `userdefined` is completely up to the author, who is responsible for keeping the whole label unique.

Notes

1. http://docbook.org/tdg/en/html/docbook.html
2. http://docbook.org/tdg/en/html/docbook.html
3. http://www.oasis-open.org/committees/docbook/documentation/index.shtml

Appendix I. The Documentation Licenses

Open Content License

This is the License for This Style Guide as well as for the whole Blender Core Documentation. You are kindly requested to produce any contribution to the Blender Documentation Project using this License. If your contribution is outside Core documentation (as a tutorial, demo .blend file, images or animation) and you wish to use a more restrictive License, you can use the Section called *Blender Artistic License*.

The source of the Open Content License is http://opencontent.org/opl.shtml. It is repeated below for easier reference.

```
OpenContent License (OPL)
Version 1.0, July 14, 1998.

This document outlines the principles underlying the OpenContent (OC)
movement and may be redistributed provided it remains unaltered.
For legal purposes, this document is the license under which
OpenContent is made available for use.

The original version of this document may be found at
http://opencontent.org/opl.shtml

LICENSE

Terms and Conditions for Copying, Distributing, and Modifying

Items other than copying, distributing, and modifying the Content
with which this license was distributed (such as using, etc.) are
outside the scope of this license.

1. You may copy and distribute exact replicas of the OpenContent
(OC) as you receive it, in any medium, provided that you conspicuously
and appropriately publish on each copy an appropriate copyright notice
and disclaimer of warranty; keep intact all the notices that refer to
this License and to the absence of any warranty; and give any other
recipients of the OC a copy of this License along with the OC. You may
at your option charge a fee for the media and/or handling involved in
creating a unique copy of the OC for use offline, you may at your
option offer instructional support for the OC in exchange for a fee,
or you may at your option offer warranty in exchange for a fee. You
may not charge a fee for the OC itself. You may not charge a fee
for the sole service of providing access to and/or use of the OC via
a network (e.g. the Internet), whether it be via the world wide web,
FTP, or any other method.

2. You may modify your copy or copies of the OpenContent or any portion
of it, thus forming works based on the Content, and distribute such
modifications or work under the terms of Section 1 above, provided that
you also meet all of these conditions:

a) You must cause the modified content to carry prominent notices
stating that you changed it, the exact nature and content of the
changes, and the date of any change.

b) You must cause any work that you distribute or publish, that in
whole or in part contains or is derived from the OC or any part
thereof, to be licensed as a whole at no charge to all third parties
under the terms of this License, unless otherwise permitted under
applicable Fair Use law.

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the OC, and
```

```
can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the OC, the distribution of the whole must be on the terms of this
License, whose permissions for other licensees extend to the entire
whole, and thus to each and every part regardless of who wrote it.
Exceptions are made to this requirement to release modified works
free of charge under this license only in compliance with Fair Use
law where applicable.

3. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to copy,
distribute or modify the OC. These actions are prohibited by law
if you do not accept this License. Therefore, by distributing or
translating the OC, or by deriving works herefrom, you indicate
your acceptance of this License to do so, and all its terms and
conditions for copying, distributing or translating the OC.

NO WARRANTY

4. BECAUSE THE OPENCONTENT (OC) IS LICENSED FREE OF CHARGE, THERE
IS NO WARRANTY FOR THE OC, TO THE EXTENT PERMITTED BY APPLICABLE
LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS
AND/OR OTHER PARTIES PROVIDE THE OC "AS IS" WITHOUT WARRANTY OF
ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE OC IS WITH YOU.
SHOULD THE OC PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE
YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

5. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY
MIRROR AND/OR REDISTRIBUTE THE OC AS PERMITTED ABOVE, BE LIABLE
TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL
OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO
USE THE OC, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED
OF THE POSSIBILITY OF SUCH DAMAGES.
```

Blender Artistic License

This is the License designed for Tutorials, .blend example files, still images and animations. It is more restrictive than the Blender Documentation License and is thought to protect more the intellectual rights of the artist producing the Tutorial/Blend File/Still/Animation. (Loosely adapted from Perl Artistic License)

Authors can of course choose the less restrictive Blender Documentation License, but no material will be hosted on the Foundation Site unless it abides to one of these two licenses.

It is highly advisable to prepare a zipped package containing, besides the Tutorial etc. files, a file called `LICENSE` containing this license.

If you are distributing a .blend file alone it is still advisable to add a LICENSE file, otherwise you can add the license in a text buffer of Blender and make sure it shows up when opening the file.

If you are releasing binary files like printable tutorials and you don't want to include the whole license (why wouldn't you?) or are releasing single images or animations, you can also add, in the text, in a corner of the image or in the last frames the wording:

```
(C) *Year* *your name* - released under Blender Artistic License - www.blender.org
```

Herebelow, the License Itself:

```
The "Blender Artistic License"

Preamble:

The intent of this document is to state the conditions under which a
Tutorial guide, a Blender file, a still image or an animation (in the
following all four will be addressed as 'Item') may be copied, such
that the Copyright Holder maintains some semblance of artistic control
over the development of the Item, while giving the users of the pack-
age
the right to use and distribute the Item in a more-or-less customary
fashion, plus the right to make reasonable modifications.

Definitions:

 "Item" refers to the collection of files distributed by the
 Copyright Holder, and derivatives of that collection of files
 created through textual modification, binary modification,
 image processing, format translation and/or modifications
 using Blender.

 "Standard Version" refers to such a Item if it has not been
 modified, or has been modified in accordance with the wishes
 of the Copyright Holder as specified below.

 "Copyright Holder" is whoever is named in the copyright or
 copyrights for the Item.

 "You" is you, if you're thinking about copying or distributing
 this Item.

 "Reasonable copying fee" is whatever you can justify on the
 basis of media cost, duplication charges, time of people involved,
 and so on.  (You will not be required to justify it to the
 Copyright Holder, but only to the computing community at large
 as a market that must bear the fee.)

 "Freely Available" means that no fee is charged for the Item
 itself, though there may be fees involved in handling the Item.
 It also means that recipients of the Item may redistribute it
 under the same conditions they received it.

1. You may make and give away verbatim copies of the Standard Version of
this Item without restriction, provided that you duplicate all of the
original copyright notices and associated disclaimers.

2. You may apply any modification derived from the Public Domain or from
the Copyright Holder.  An Item modified in such a way shall still be
considered the Standard Version.

3. You may otherwise modify your copy of this Item in any way, provided
that you insert a prominent notice in each changed file - except images -

stating how and when you changed that file, that you keep note in a separate
text file of any change/deletion/addition of images, and provided that
you do at least ONE of the following:

    a) place your modifications in the Public Domain or otherwise make them
    Freely Available, such as by posting said modifications to Usenet or
    an equivalent medium, or placing the modifications on a major archive
    site such as Blender Foundation www.blender.org, or by allowing the
    Copyright Holder to include your modifications in the Standard Ver-
sion
```

```
    of the Package.

    b) use the modified Item only within your corporation or organization.

    c) make other distribution arrangements with the Copyright Holder.

4. You may distribute this Item electronically, provided that you do at
least ONE of the following:

    a) distribute a Standard Version together with instructions (in a
    README file, in a text window of Blender) on where to get the
    Standard Version.

    b) make other distribution arrangements with the Copyright Holder.

5. If this Item is a Tutorial documentation or a still image you can
redistribute it as hard copy, provided that you do at least ONE of
the following:

    a) distribute a printout of Standard Version with at most mere
    typesetting changes, stating clearly who is the Copyright
    holder and where to get the Standard Version.

    b) make other distribution arrangements with the Copyright Holder.

6. You may charge a reasonable copying fee for any distribution of this
Item.  You may not charge a fee for this Item itself.  However,
you may distribute this Item in aggregate with other (possibly
commercial) programs as part of a larger (possibly commercial) software
distribution provided that you do not advertise this Item as a
product of your own.

6. If this Item is a Blender File the rendered output from it obtained
via Blender does not automatically fall under the copyright of this Item,
but belongs to whoever generated them, and may be sold commercially, and
may be aggregated with this Item.

7. The name of the Copyright Holder may not be used to endorse or promote
products derived from this Item without specific prior written permission.

8. THIS ITEM IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.
```

GNU General Public License

The Blender source code is licensed under the GNU General Public License (the GPL). This license allows you to use, copy, modify, and distribute the program and the source code. The GPL only applies to the program itself, not to this manual nor to any works created by people using the program.

GNU General Public License

Version 2, June 1991

Copyright 1998, 1999 Free Software Foundation, Inc., 59 Temple Place, Suite 330,Boston, MA 02111-1307 USA

> **Note:** Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

a. copyright the software, and

b. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU General Public License -- Terms and Conditions for Copying, Distribution and Modification

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

12.

NO WARRANTY

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS

PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type
'show w'. This is free software, and you are welcome to redistribute
it under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Notes

1. http://opencontent.org/opl.shtml

Glossary

A-Z

Active

Blender makes a distinction between *selected* and *active*. Only one Object or item can be *active* at any given time, for example to allow visualization of data in buttons.

An active object is one that is in EditMode, or is immediately switchable to EditMode (usually by **TAB**). No more than one object is active at any moment. Typically, the most recent selected object is active.

See Also: Selected.

Actuator

A LogicBrick that acts like a muscle of a lifeform. It can move the object, or also make a sound.

See Also: LogicBrick, Sensor, Controller.

Alpha

The alpha value in an image denotes opacity, used for blending and antialiasing.

Ambient light

Light that exists everywhere without any particular source. Ambient light does not cast shadows, but fills in the shadowed areas of a scene.

Anti-aliasing

An algorithm designed to reduce the stair-stepping artifacts that result from drawing graphic primitives on a raster grid.

AVI

"Audio Video Interleaved". A container format for video with synchronized audio. An AVI file can contain different compressed video and audio-streams.

Back-buffer

Blender uses two buffers in which it draws the interface. This double-buffering system allows one buffer to be displayed, while drawing occurs on the back-buffer. For some applications in Blender the back-buffer is used to store color-coded selection information.

Bevel

Beveling removes sharp edges from an extruded object by adding additional material around the surrounding faces. Bevels are particularly useful for flying logos, and animation in general, since they reflect additional light from the corners of an object as well as from the front and sides.

Bounding box

A six-sided box drawn on the screen that represents the maximum extent of an object.

Bump map

A grayscale image used to give a surface the illusion of ridges or bumps. In Blender bumpmaps are called Nor-maps.

Channel

Some DataBlocks can be linked to a series of other DataBlocks. For example, a Material has eight *channels* to link Textures to. Each IpoBlock has a fixed number of available *channels*. These have a name (LocX, SizeZ, enz.) which indicates how they can be applied. When you add an IpoCurve to a channel, animation starts up immediately.

Child

Objects can be linked to each other in hierarchical groups. The Parent Object in such groups passes its transformations through to the Child Objects.

See Also: Parent.

Clipping

The removal, before drawing occurs, of vertices and faces which are outside the field of view.

Controller

A LogicBrick that acts like the brain of a lifeform. It makes decisions to activate muscles (Actuators), either using simple logic or complex Python scripts.

See Also: LogicBrick, Sensor, Python, Actuator.

DataBlock (or "block")

The general name for an element in Blender's Object Oriented System.

Doppler effect

The Doppler effect is the change in pitch that occurs when a sound has a velocity relative to the listener. When a sound moves towards the listener the pitch will rise. when going away from the listener the pitch will drop. A well known example is the sound of an ambulance passing by.

Double-buffer

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.

EditMode

The mode for making intra-object graphical changes. Blender has two modes for making changes graphically. EditMode allows intra-object changes (moving, scaling rotating, deleting, and other operations on selected vertices of the active object). By contrast, ObjectMode allows inter-object changes (operations on selected objects).

Switch between EditMode and ObjectMode with Hotkey: **TAB**.

See Also: ObjectMode, Vertex (pl. vertices).

Extend select

Adds new selected items to the current selection (**SHIFT-RMB**)

Extrusion

The creation of a three-dimensional object by pushing out a two-dimensional outline and giving it height, like a cookie-cutter. It is often used to create 3D text.

Face

The triangle and square polygons that form the basis for Meshes or for rendering.

Field

Frames from videos in NTSC or PAL format are composed of two interlaced *fields*.

FaceSelectMode

Mode to select faces on an object. Most important for texturing objects. Hotkey: **FKEY**

Flag

A programming term for a variable that indicates a certain status.

Flat shading

A fast rendering algorithm that simply gives each facet of an object a single color. It yields a solid representation of objects without taking a long time to render. Pressing **ZKEY** switches to flat shading in Blender.

Fps

Frames per second. All animations, video, and movies are played at a certain rate. Above ca. 15fps the human eye cannot see the single frames and is tricked into seeing a fluid motion. In games this is used as an indicator of how fast a game runs.

Frame

A single picture taken from an animation or video.

Gouraud shading

A rendering algorithm that provides more detail. It averages color information from adjacent faces to create colors. It is more realistic than flat shading, but less realistic than Phong shading or ray-tracing. The hotkey in Blender is **CTRL-Z**.

Graphical User Interface

The whole part of an interactive application which requests input from the user (keyboard, mouse etc.) and displays this information to the user. Blenders GUI is designed for an efficient modeling process in an animation company where time equals money. Blenders whole GUI is done in OpenGL.

See Also: OpenGL.

Hierarchy

Objects can be linked to each other in hierarchical groups. The Parent Object in such groups passes its transformations through to the Child Objects.

Ipo

The main animation curve system. Ipo blocks can be used by Objects for movement, and also by Materials for animated colors.

IpoCurve

The Ipo animation curve.

Item

The general name for a selectable element, e.g. Objects, vertices or curves.

Lathe

A lathe object is created by rotating a two-dimensional shape around a central axis. It is convenient for creating 3D objects like glasses, vases, and bowls. In Blender this is called "spinning".

Keyframe

A frame in a sequence that specifies all of the attributes of an object. The object can then be changed in any way and a second keyframe defined. Blender automatically creates a series of transition frames between the two keyframes, a process called "tweening."

Layer

A visibility flag for Objects, Scenes and 3DWindows. This is a very efficient method for testing Object visibility.

Link

The reference from one DataBlock to another. It is a "pointer" in programming terminology.

Local

Each Object in Blender defines a *local* 3D space, bound by its location, rotation and size. Objects themselves reside in the *global* 3D space.

A DataBlock is *local*, when it is read from the current Blender file. Non-local blocks (library blocks) are linked parts from other Blender files.

LogicBrick

A graphical representation of a functional unit in Blender's game logic. LogicBricks can be Sensors, Controllers or Actuators.

See Also: Sensor, Controller, Actuator.

Mapping

The relationship between a Material and a Texture is called the 'mapping'. This relationship is two-sided. First, the information that is passed on to the Texture must be specified. Then the effect of the Texture on the Material is specified.

Mipmap

Process to filter and speed up the display of textures.

MPEG-I

Video compression standard by the "Motion Pictures Expert Group". Due to its small size and platform independence, it is ideal for distributing video files over the internet.

ObData block

The first and most important DataBlock linked by an Object. This block defines the Object *type*, e.g. Mesh or Curve or Lamp.

Object

The basic 3D information block. It contains a position, rotation, size and transformation matrices. It can be linked to other Objects for hierarchies or deformation. Objects can be "empty" (just an axis) or have a link to ObData, the actual 3D information: Mesh, Curve, Lattice, Lamp, etc.

ObjectMode

The mode for making inter-object graphical changes. Blender has two modes for making changes graphically. ObjectMode allows inter-object changes (moving, scaling rotating, deleting and other operations on selected objects). By contrast, EditMode allows intra-object changes (operations on selected vertices in the active object).

Switch between ObjectMode and EditMode with Hotkey: **TAB**.

See Also: EditMode.

OpenGL

OpenGL is a programming interface mainly for 3D applications. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Blenders whole interface and 3D output in the real-time and interactive 3D graphic is done by OpenGL.

Oversampling

See: Anti-aliasing

Overscan

Video images generally exceed the size of the physical screen. The edge of the picture may or may not be displayed, to allow variations in television sets. The extra area is called the overscan area. Video productions are planned so critical action only occurs in the center safe title area. Professional monitors are capable of displaying the entire video image including the overscan area.

Parent

An object that is linked to another object, the parent is linked to a child in a parent-child relationship. A parent object's coordinates become the center of the world for any of its child objects.

See Also: Child.

Perspective view

In a perspective view, the further an object is from the viewer, the smaller it appears. See orthographic view.

Pivot

A point that normally lies at an object's geometric center. An object's position and rotation are calculated in relation to its pivot-point. However, an object can be moved off its center point, allowing it to rotate around a point that lies outside the object.

Pixel

A single dot of light on the computer screen; the smallest unit of a computer graphic. Short for "picture element."

Plug-In

A piece of (C-)code loadable during runtime. This way it is possible to extend the functionality of Blender without a need for recompiling. The Blender plugin for showing 3D content in other applications is such a piece of code.

Python

The scripting language integrated into Blender. Python[1] is an interpreted, interactive, object-oriented programming language.

Quaternions

Instead of using a three-component Euler angle, quaternions use a four-component vector. It is generally difficult to describe the relationships of these quaternion channels to the resulting orientation, but it is often not necessary. It is best to generate quaternion keyframes by manipulating the bones directly, only editing the specific curves to adjust lead-in and lead-out transitions.

Render

To create a two-dimensional representation of an object based on its shape and surface properties (i.e. a picture for print or to display on the monitor).

Rigid Body

Option for dynamic objects in Blender which causes the game engine to take the shape of the body into account. This can be used to create rolling spheres for example.

Selected

Blender makes a distinction between *selected* and *active* objects. Any number of objects can be *selected* at once. Almost all key commands have an effect on *selected* objects. Selecting is done with the right mouse button.

See Also: Active, Extend select.

Sensor

A LogicBrick that acts like a sense of a lifeform. It reacts to touch, vision, collision etc.

See Also: LogicBrick, Controller, Actuator.

Single User

DataBlocks with only one user.

Smoothing

A rendering procedure that performs vertex-normal interpolation across a face before lighting calculations begin. The individual facets are then no longer visible.

Transform

Change a location, rotation, or size. Usually applied to Objects or vertices.

Transparency

A surface property that determines how much light passes through an object without being altered.

See Also: Alpha.

User

When one DataBlock references another DataBlock, it has a user.

Vertex (pl. vertices)

The general name for a 3D or 2D point. Besides an X,Y,Z coordinate, a vertex can have color, a normal vector and a selection flag. Also used as controlling points or handles on curves.

Vertex array

A special and fast way to display 3D on the screen using the hardware graphic acceleration. However, some OpenGL drivers or hardware doesn't support this, so it can be switched off in the InfoWindow.

Wireframe

A representation of a three-dimensional object that only shows the lines of its contours, hence the name "wireframe."

X, Y, Z axes

The three axes of the world's three-dimensional coordinate system. In the FrontView, the X axis is an imaginary horizontal line running from left to right; the Z axis is a vertical line; and Y axis is a line that comes out of the screen toward you. In general, any movement parallel to one of these axes is said to be movement along that axis.

X, Y, and Z coordinates

The X coordinate of an object is measured by drawing a line that is perpendicular to the X axis, through its centerpoint. The distance from where that line intersects the X axis to the zero point of the X axis is the object's X coordinate. The Y and Z coordinates are measured in a similar manner.

Z-buffer

For a Z-buffer image, each pixel is associated with a Z-value, derived from the distance in 'eye space' from the Camera. Before each pixel of a polygon is drawn,

the existing Z-buffer value is compared to the Z-value of the polygon at that point. It is a common and fast visible-surface algorithm.

Notes

1. http://www.python.org/