

Blender Documentation

Last modified 25 jan 2002 bv

Florian Findeiss

Alex Heizer

Reevan McKay

Jason Oppel

Ton Roosendaal

Stefano Selleri

Bart Veldhuizen

Carsten Wartmann

Blender Documentation: Last modified 25 jan 2002 bv

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

I. Introduction.....	1
1. Introduction	1
About this manual	1
What is Blender?	2
Blender's History.....	2
About Free Software and the GPL	3
Getting support - the Blender community.....	4
2. Installation.....	5
Downloading and installing the binary distribution	5
OSX.....	5
Windows.....	5
Linux	6
(others).....	8
Building Blender from the sources.....	8
Introduction	8
Windows.....	8
Linux	8
(others).....	8
3. Understanding the interface	9
The window system	9
The File Select window	9
The Image Select window	9
The 3D window	10
The IPO window	10
The Text window.....	11
The Info Window	11
DisplayButtons	12
The Lamp Buttons.....	13
The Material Buttons	13
The Texture Buttons.....	14
The Animation Buttons	14
The World Buttons	15
The Realtime Buttons	15
The Edit Buttons.....	15
Mouse	16
Keyboard.....	16
4. Quickstart	17
(a tutorial here?).....	17
II. Linear 3D	19
5. Mesh Modeling.....	19
Basic objects	19
Editmode.....	20
Smoothing.....	21
Proportional editing tool	21
Extrude.....	21
Spin and SpinDup	21
Screw	21
Boolean operations	21
Noise.....	23
Warp Tool.....	23
Catmull-Clark Subdivision Surfaces.....	23
Metaballs.....	23
6. Curves and Surfaces	25
Curves	25
Surfaces	25
Text.....	25
Extruding curves along a path.....	25
Skinning	25
7. Special modelling techniques.....	27
Dupliverts	27
Dupliframes	27
Modelling with animation.....	27

Modelling with lattices	27
8. Materials and textures	29
Material settings.....	29
Textures	29
Texture plugins.....	29
Environment Maps.....	29
Introduction	29
The EnvMap buttons	30
UV editor and FaceSelect.....	31
Introduction	32
Assigning Images to faces.....	32
Selecting faces.....	33
Editing UV coordinates	33
Rendering and UV coordinates.....	35
9. Lighting.....	37
Lamp types	37
Shadow.....	37
Volumetric Lighting	37
Tweaking Lighting settings.....	37
10. Animation.....	39
Keyframe animation.....	39
Path animation	39
IPOs	39
Vertex Keys	39
11. Effects	41
Introduction.....	41
Build Effect	41
Particles Effect.....	42
Wave Effect	42
12. Rendering	45
Rendering animations.....	45
Antialiasing	45
Motion blur.....	45
Output formats	45
Cartoon shading.....	45
The Unified Renderer.....	45
Preparing your work for video.....	45
Color Saturation	45
Rendering to fields.....	45
Setting up the correct field order	45
13. The Sequence Editor	47
Editing video with the sequence editor.....	47
Using sequence editor plugins	47
Writing sequence editor plugins	47
14. Character Animation	49
General Tools.....	49
Armature Object	49
Skinning	53
Weight Painting.....	54
Posemode.....	55
Action Window	55
Action Actuator.....	57
Python	58
NLAWindow (Non Linear Animation)	60
Constraints.....	63
15. Python Scripting.....	67
Creating Python scripts	67
Python modules reference.....	67

III. Interactive 3d.....	69
16. Interactive 3d	69
Introduction.....	69
Designing for interactive environments.....	69
Physics.....	69
Logic Editing	69
Sensors.....	69
Always	69
Keyboard	69
Mouse.....	69
Touch.....	69
Collision.....	69
Near.....	69
Radar.....	69
Property	69
Random	70
Ray.....	70
Message	70
Controllers	70
And.....	70
Or.....	70
Expression.....	70
Python.....	70
Actuators.....	70
Motion.....	70
Constraint.....	70
IPO.....	70
Camera.....	71
Sound	71
Property	71
Edit Object.....	71
Scene	71
Random	71
Message	71
CD.....	71
Game	71
Visibility.....	71
Exporting to standalone applications.....	71
17. Usage of Blender 3D Plug-in	73
Introduction.....	73
Functionality.....	73
3D Plug-in installation	74
Creating content for the plug-ins	74
Jumping to another HTML page	77
Creating a custom loading animation.....	78
Embedding Blender 3D Plug-in in web pages.....	78
Embedding the ActiveX control in other applications.....	81
Embedding Blender content in PowerPoint.....	81
Blender 3D Plug-in FAQs	82
18. Python Scripting for interactive environments.....	85
(game engine specific Python subjects).....	85
IV. Appendices.....	87
A. Hotkey Overview	87
B. Blender Windows/Buttons	89
C. Command Line Arguments	91
D. Supported videocards	93
E. Documentation changelog.....	97
F. Blender changelog.....	99
G. Troubleshooting.....	101
H. About the Blender Documentation Project	103
About the Blender Documentation Project.....	103
Contributors	103
How to contribute.....	103

Getting your system ready for DocBook/XML	103
Learning DocBook/XML.....	103
The Blender Documentation Project styleguide.....	103
General Guidelines	103
Images Guideline	103
Tables	104
Code	104
Documentation Style in Practice	104
Images Examples.....	105
Tables	106
Code	106
Cross references.....	107
Keywords	107
Cross Reference Labelling Guidelines	108
I. The Documentation Licenses	109
Open Content License	109
Blender Artistic License	110
GNU General Public License	112
Glossary	119

Chapter 1. Introduction

About this manual

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 maillist², and drop us a line.

What has been done so far:

Ready

- the Section called *Boolean operations* in Chapter 5
- Chapter 14
- Chapter 17

Draft

- the Section called *What is Blender?*
- the Section called *Blender's History*
- the Section called *About this manual*
- the Section called *About Free Software and the GPL*
- the Section called *OSX* in Chapter 2
- the Section called *Linux* in Chapter 2
- the Section called *Windows* in Chapter 2
- the Section called *The window system* in Chapter 3
- the Section called *Environment Maps* in Chapter 8
- the Section called *UV editor and FaceSelect* in Chapter 8
- the Section called *Basic objects* in Chapter 5
- the Section called *Wave Effect* in Chapter 11
- the Section called *Build Effect* in Chapter 11
- Appendix D
- Appendix H
- Appendix I

In Progress

- Chapter 4
- the Section called *Editmode* in Chapter 5
- the Section called *Extrude* in Chapter 5
- the Section called *Proportional editing tool* in Chapter 5
- the Section called *Particles Effect* in Chapter 11
- the Section called *Dupliverts* in Chapter 7
- the Section called *Keyframe animation* in Chapter 10

- the Section called *Path animation* in Chapter 10
- Appendix A
- *Glossary*
- Coding Guidelines

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, postproduction 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 workflow
- 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 toolset 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 shut-down 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 Source³ 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 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 roadmap

- 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 site⁴. For reference, a copy of the GNU General Public License is included in Appendix I.

Getting support - the Blender community

(to be written)

Ideas:

1. Official Blender Foundation site @ www.blender.org
2. User community @ www.elysiun.com
3. Blender Knowledge Base @ vrotvrot.com/support
4. Chatbox #blenderchat @ irc.openprojects.net --> how to access

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>

Chapter 2. Installation

Downloading and installing the binary distribution

OSX

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.

Windows

Quick Install

Download the file `blender-publisher-2.25-windows.exe` 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

Blender Publisher is available from the Blender Web site (<http://www.blender.org/>) in source form, and as a binary for Microsoft Windows. 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.

Download the file `blender-publisher-2.25-windows.exe` 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 saved the file in and double-click it to start the installation.

After accepting the license, select a place to install the files to (the default should do well), and click "Next" to install Blender. 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 Publisher" has been created by the setup routine) or by double-clicking a Blender file (*.blend).

Linux

Quick Install

Download the file `blender-publisher-2.25-linux-glibc2.1.2.tar.gz` from the downloads section of the Blender Website. Unpack the archive to a location of your choice. This will create a directory named `blender-publisher-2.25-linux-glibc2.1.2-i386`, in which you will find the **blenderpublisher** and **blenderplayer** executables. To start blender just open a shell and execute **blenderpublisher**, when running X.

In-depth Instructions

Blender Publisher is available from the Blender Web site (<http://www.blender.org/>) in source form, and as a binary for Linux. 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.

Download the file `blender-publisher-2.25-linux-glibc2.1.2.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-publisher-2.25-linux-glibc2.1.2.tar.gz`). If you like, you can rename the resulting directory from `blender-publisher-2.25-linux-glibc2.1.2-i386` to something shorter, e.g. just `blender`.

Blender is now installed and can be started on the command line by entering `/path/to/blenderpublisher` followed by pressing the enter key. 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", "blenderpublisher" and "Blender 2.25" 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-2.25-linux-glibc2.1.2-i386/blenderpublisher`, the "Work Path" would be `/home/user/blender-publisher-2.25-linux-glibc2.1.2-i386/`.

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-2.25-linux-glibc2.1.2-i386/blenderpublisher`
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 con-

tains 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.

(others)

(to be written)

Building Blender from the sources

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.stuffit.com/>
3. <http://www.blender.org/>
4. <http://www.blender.org/>

Chapter 3. Understanding the interface

(introduction to be written)

The window system

This tutorial gives a summary of the most important windows and ButtonsWindows in Blender. It does not try to be complete nor to explain each window in detail. *Source:www.blender.nl*

The File Select window

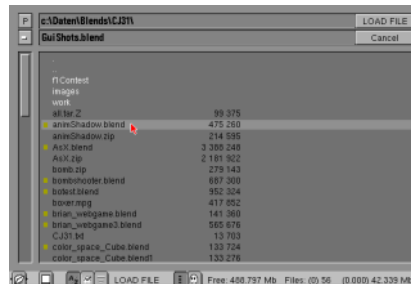


Figure 3-1. The File Select window

Hotkeys: **F1** for loading, **F2** for saving.

Left click to select a file and press **ENTER** to load it, or middle click to both select and load it. Alternatively you can use the "LOAD" (or "SAVE") buttons in the upper right corner of a FileWindow.

The FileWindow in Blender is general and will be called for all file operations.

To save a file, enter a name in the filename field (the second one from the top) and press **ENTER** to confirm the filename. Press **ENTER** again to save the file. If you have already entered a filename earlier, pressing **F2**, **ENTER** will save, too.

Entering a new directory name in the directory box (the one on the top) and pressing will **ENTER** create the new directory for you after asking confirmation.

The Image Select window



Figure 3-2. The Image Select window

This window type works exactly like the file load window, but all recognized image files are shown as thumbnails. An example of where this window is used is in the Texture Buttons window, for loading image texture maps.

The 3D window

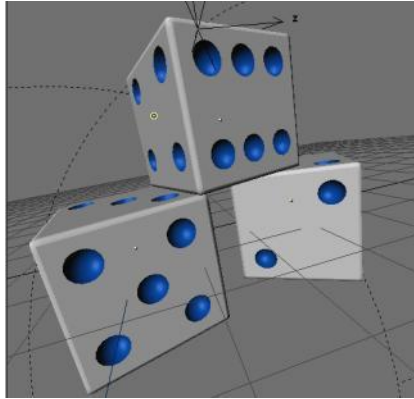


Figure 3-3. The 3D window

This window allows you to manipulate the objects in your scene directly by dragging (**GKEY**), rotating (**RKEY**) or scaling them (**SKEY**).

Change between top, front and right view with **numpad 7**, **numpad 1** and **numpad 3**. Toggle perspective mode with **numpad 5**, and change to camera view with **numpad 0**. Toggle between wire, solid or shaded drawing with **ZKEY** and **CTRL+ZKEY**

You can navigate through 3D space by dragging with the middle button; this will rotate your view. Dragging with middle button and holding the key will translate the view, and holding down will allow you to zoom.

The IPO window

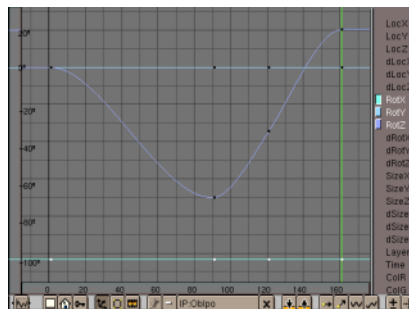


Figure 3-4. The IPO window

Hotkey: **SHIFT+F6**.

In this window you can edit the animation curves that are associated to different object properties, like their position and rotation, but also their color or layer. The different object properties or 'channels' that are available are shown on the right. You can select channels by left clicking on them - **SHIFT** will extend the selection.

Just like in the 3Dwindow, enter edit mode by pressing **TAB**. You can now edit the individual vertices of the curves or change their curvature.

The Text window

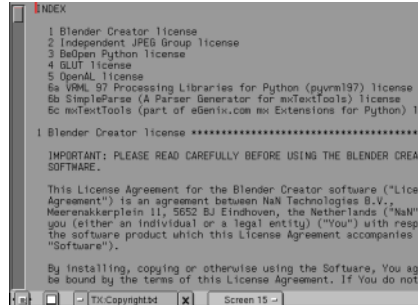


Figure 3-5. The Text window

Hotkey: **SHIFT+F11**.

The TextWindow is mainly used for writing Python scripts, but you can also use it for other things.. It is often a good idea to keep notes of your animation projects. Information that I always forget (especially after a couple of weeks!) are the maps that I have used, the scenes that I have defined, important frame numbers, well, you get the picture.

A big advantage of using the TextWindow is that the notes are stored inside the .blend files - this way you can never loose your notes again!

When creating a new TextWindow, you have the choice of creating a new text object, or to import an existing text file. In that case, a FileWindow will be opened and you can select a file as usual. Of course, you can have more than one text object in your Blender file.

The Info Window



Figure 3-6. The Info Window

The Info Window does not have a hotkey. Instead, it is 'hidden' at the top of your screen. To reveal it, drag down the top window border on your screen as indicated in the picture.



Figure 3-7. Drag the window edge to reveal the InfoWindow

The Info Window shows you the settings of Blender. Among other things, you can tell Blender where to store autosave files, when to save them (use the 'Time' value) and how many versions to keep on disk. Here, you can also activate

Blender's tooltips function. Tooltips will appear in the right corner of the header of the Info Window, right next to the Blender URL and version number.

You can make your settings permanent by pressing **CTRL+U** and confirming the requester that pops up. (Please note, that this manual assumes that you have not changed the default Blender settings file!)

Important note for owners of a two-button mouse: Activating the 2-mouse option will cause Blender to simulate a middle click when you hold down **ALT** and left click.

DisplayButtons

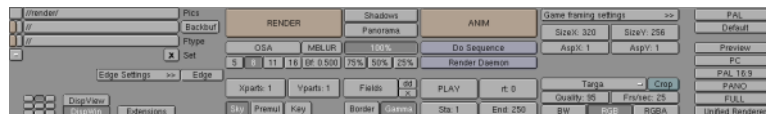


Figure 3-8. The DisplayButtons

Hotkey: **F10**.

Once you are ready to render your image or save it to disk, you need the DisplayButtons window. In this window you can control all the parameters that have to do with items as rendering quality, rendering size, animation length and animation filenames.



Figure 3-9. Image size and quality

To define the image size you can either select a preset value from the row of buttons on the far right or set an image size yourself by changing the SizeX and SizeY values. For preview rendering, you can also render the image at 25%, 50% or 75% of the final output size.

Turn anti-aliasing on or off with the OSA button. Control the image quality with the oversampling buttons 5, 8, 11 or 16. A higher value will result in a better image quality at the penalty of a longer rendering time.



Figure 3-10. Saving animations.

Start rendering by selecting the 'Render' button or by pressing **F12**. A separate window will pop up. **ESC** aborts rendering. **F11** will toggle the rendering result window.

Press 'Anim' to render an entire animation. When rendering an animation, enter the basename of the animation frames in the 'Pics' field. Each file will be auto-

matically named (basename)+framenummer. For example: frame0001, frame0002 etc. After rendering, press the ' Play' button to play back your animation.

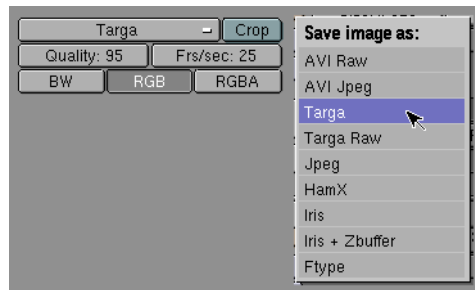


Figure 3-11. Saving animations.

Select a file format with the MenuButton labeled 'Targa' here. A pop-up menu lets you choose the desired format. For animations, you can also choose 'AVI raw' and 'AVI jpeg' and define the number of frames per second for these files.

Tip: The Windows Mediaplayer has problems playing back Blender's AVI-raw and jpeg files, but the Quicktime4 player plays them back correctly.



Figure 3-12. Animation length.

To determine the start- and endframe of your animation, change the Sta: and End: values. After you have rendered your animation press Play to play it back.

The Lamp Buttons



Figure 3-13. Lamp Buttons.

Hotkey: F4.

The Lamp Buttons will only show when a lamp has been selected. With these buttons, you can change all the parameters of lamps, like their color, energy, type (regular lamp, spotlight or sun). You can also control the quality of shadows by manipulating the clipping values and shadow buffer sizes.

The Material Buttons



Figure 3-14. Material Buttons.

Hotkey: F5.

MaterialButtons are only shown when an object with an Material has been selected. To create a new Material or browse in the scene-materials use the MenuButton in the MaterialButtons header. This window allows control over properties as object color, shininess, transparency, textures, texture types, texture projection methods.

The Texture Buttons

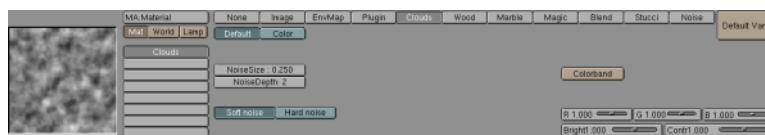


Figure 3-15. Texture Buttons.

Hotkey: F6.

In this window you can select several types of textures for use in a material, light or world setting. The available types are:

Image: You can load an image as a 'texture map'. This image will then be projected on an object in the way that you define (for example using planar or spherical projection). Projection method is a material property.

Procedural: Clouds, Wood, Marble, Magic, Blend, Stucci and Noise. These textures are predefined and have a number of parameters that you can adjust. Procedural textures are also truly three dimensional. An example of this would be a block of wood: if you cut out a part of it, the wood texture needs to continue on the inside in a realistic manner.

Plugin: You can also program your own piece of code to use as a procedural texture. It is similar to writing a sequence editor plugin. More information about this can be found in the help section of this site.

Environment map: Environment maps are used to simulate reflections of the environment in an object. To achieve this, Blender calculates six images from the object's viewpoint. These images are then combined to compute a reflection.

The Animation Buttons



Figure 3-16. Animation Buttons.

In the left part of the Animation Buttons window you can set properties for things like curve following, automatic object duplication and object tracking.

The middle part of the window contains the interface to object plugins like particle systems and the wave effect. Pressing the 'New Effect' button and selecting an effect from the effects list on the right will attach a new effect to the currently selected object and display the list of options for this effect.

The standard effects are: particle system, wave and build.

The World Buttons



Figure 3-17. World Buttons.

In the WorldButtons you can define settings that are scene-global. These are for example the mist (fog) settings, rendering of stars, the color and textures of the horizon and zenith.

The Realtime Buttons



Figure 3-18. Realtime Buttons.

Hotkey: F8.

The RealtimeButtons are Blenders editor for defining interactive realtime 3D graphics. Here you can edit settings for the physics of realtime objects, and connect objects with game-logic. Check our numerous tutorials on creation of real-time content!

The Edit Buttons

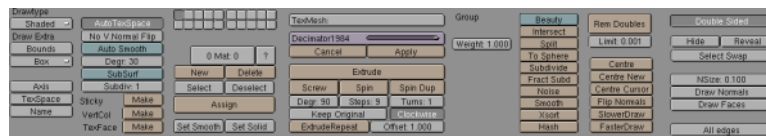


Figure 3-19. Edit Buttons.

The buttons that are shown in the EditButtons window depend on the kind of object that you have currently selected. Each type of Objects has its own specific set of Buttons and settings. Meshes for example have Buttons for manipulation of vertices and Curves have Buttons to edit the resolution and the order of the Curve.

Mouse

(to be written)

Keyboard

(to be written)

Chapter 4. Quickstart

(a tutorial here?)

Chapter 5. Mesh Modeling

Basic objects

To create a basic object press **SPACE** and select "ADD->Mesh". You also can access the 'add'-menu by pressing **SHIFT-AKEY**. Then you can select the basic object you like to create. Below every basic object you can create within blender is described. There is also a screenshot, where every basic object you can created it shown.

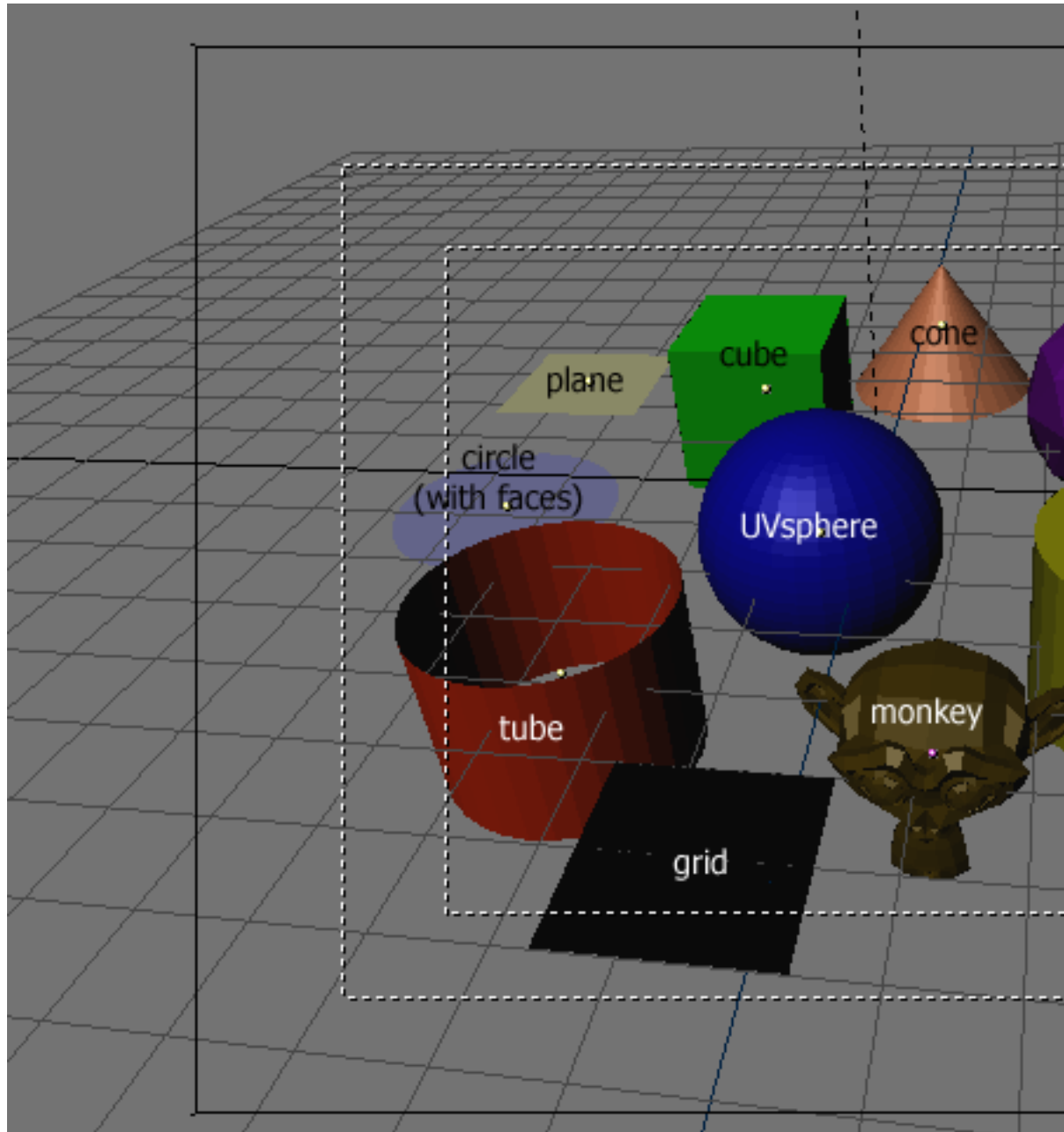


Figure 5-1. Basic Objects

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 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 shown 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 n rings. These levels of detail can be specified in the popup window shown when the UVsphere is created. The higher the number of segments and rings is, the smoother the surface of the result UVsphere becomes. Example objects that can be created out of UVspheres are balls, heads or pearls for a necklace.

Icosphere

A standard Icosphere is made out of triangles. The number of subdivisions can be specified in the popup window shown when the Icosphere is created. The higher the number of subdivisions is, the smoother the surface of the result Icosphere becomes. This object is normally used to achieve a more symmetrical and economical layout of vertices than the 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 shown when the object is created. The higher the number of vertices is, 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 shown when the object is created. The higher the number of vertices is, the smoother the hollow circular cross-section becomes. Example objects that can be created out of tubes are pipes or drinking glasses.

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 shown when the object is created. The higher the number of vertices is, 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 vertices. The resolution of the x-axis and y-axis can be specified in the popup window shown when the object is created. The higher the resolution is, 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 monkey's head after you have pressed the 'Oooh Oooh Oooh' button.

Editmode

(to be written)

Smoothing

(to be written)

Proportional editing tool

(to be written)

Extrude

(to be written)

Spin and SpinDup

(to be written)

Screw

(to be written)

Boolean operations

The boolean operations 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 and 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.

Boolean operations are invoked by selecting 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 5-2. Options for boolean operations

For the "Difference" operation the order of selection is important. The active object (light purple in wire-frame view) is subtracted by the selected object.

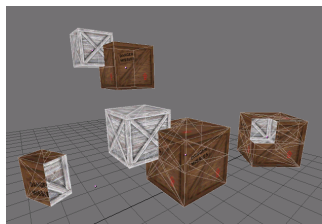


Figure 5-3. Resulting objects, original top, intersect, union, difference

This functionality is currently under heavy development. To make it possible for you to use this functions for your work we list here the current limitations:

Known problems of the Boolean Operations

- The number of polygons generated can be very large compared to the original meshes. This is especially true for complex concave objects
- 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 do not match up
- 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. To get rid of the doubled vertices use the "Remove Doubles" button in the EditButtons **F9**.

- 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.
- Operations are between two objects only, there is no way to perform operations on more than 2 operands, such as intersect all these selected objects with the active object.

Noise

(to be written)

Warp Tool

(to be written)

Catmull-Clark Subdivision Surfaces

(to be written)

Metaballs

(to be written)

Chapter 6. Curves and Surfaces

Curves

(to be written)

Surfaces

(to be written)

Text

(to be written)

Extruding curves along a path

(to be written)

Skinning

(to be written)

Chapter 7. Special modelling techniques

Dupliverts

(to be written)

Dupliframes

(to be written)

Modelling with animation

(to be written)

Modelling with lattices

(to be written)

Chapter 8. Materials and textures

Material settings

(to be written)

Textures

(to be written)

Texture plugins

(to be written)

Environment Maps



Figure 8-1. Environment Map Example

Introduction

This rendering technique uses texture mapping to mimic a mirroring surface. From a carefully chosen 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.

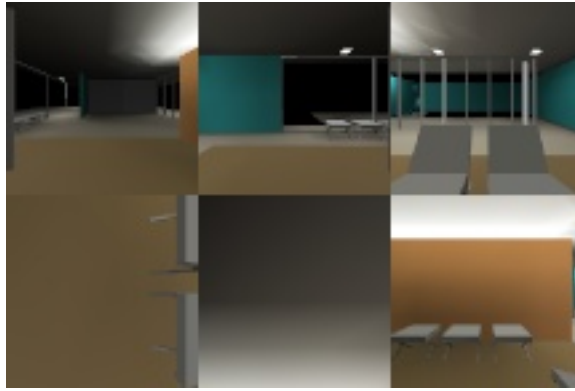


Figure 8-2. Environment Map 'Lookup table'

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 at 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 8-3. Environment Map 'Lookup table'

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 8-4. 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 8-5. 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 8-6. 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 8-7. 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.

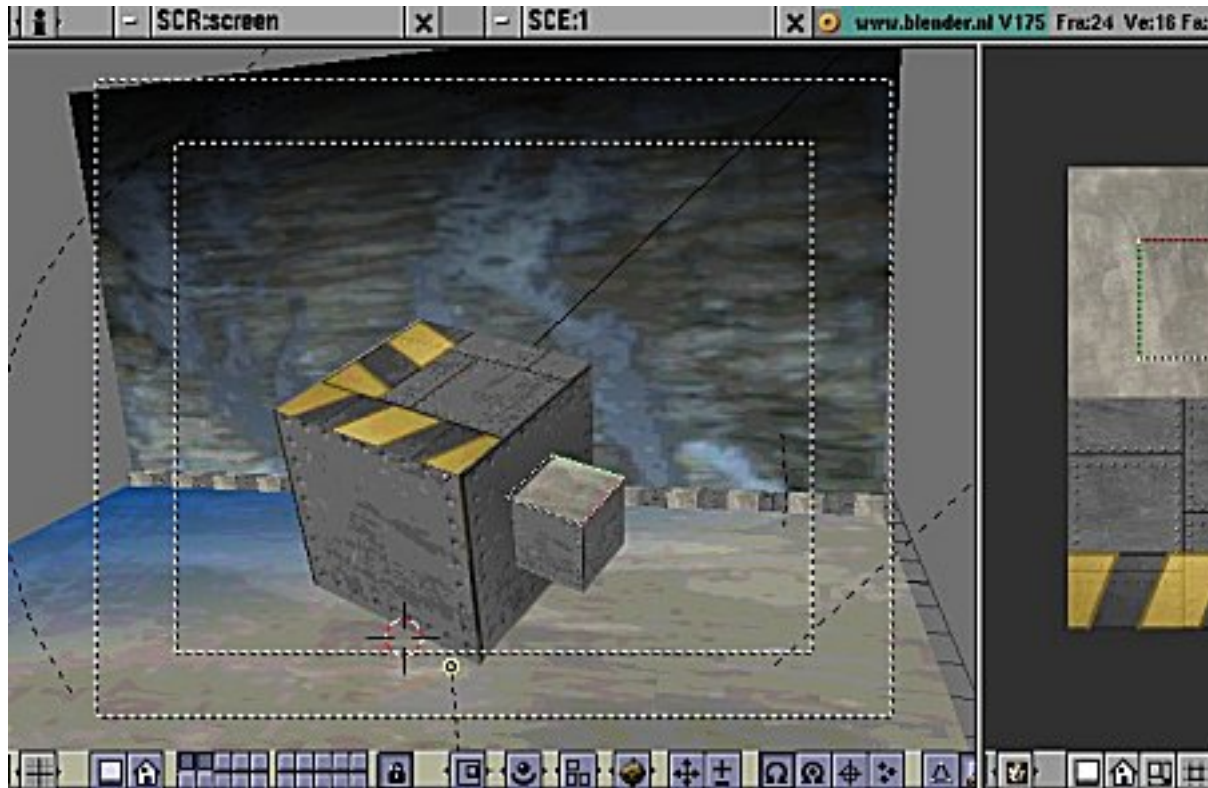


Figure 8-8. 3Dwindow and ImageWindow

Loading or browsing an Image in FaceSelect automatically assigns the Image to the selected faces. You can immediately see 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.

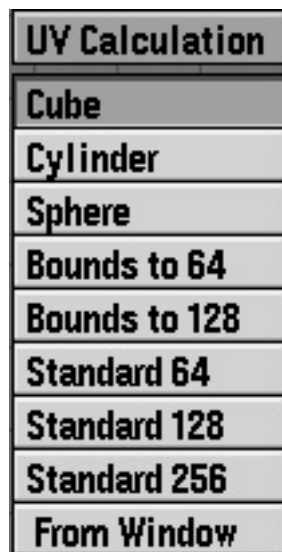


Figure 8-9. ..

- *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 boundingbox 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 8-10. ..

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 hard-disk. 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.

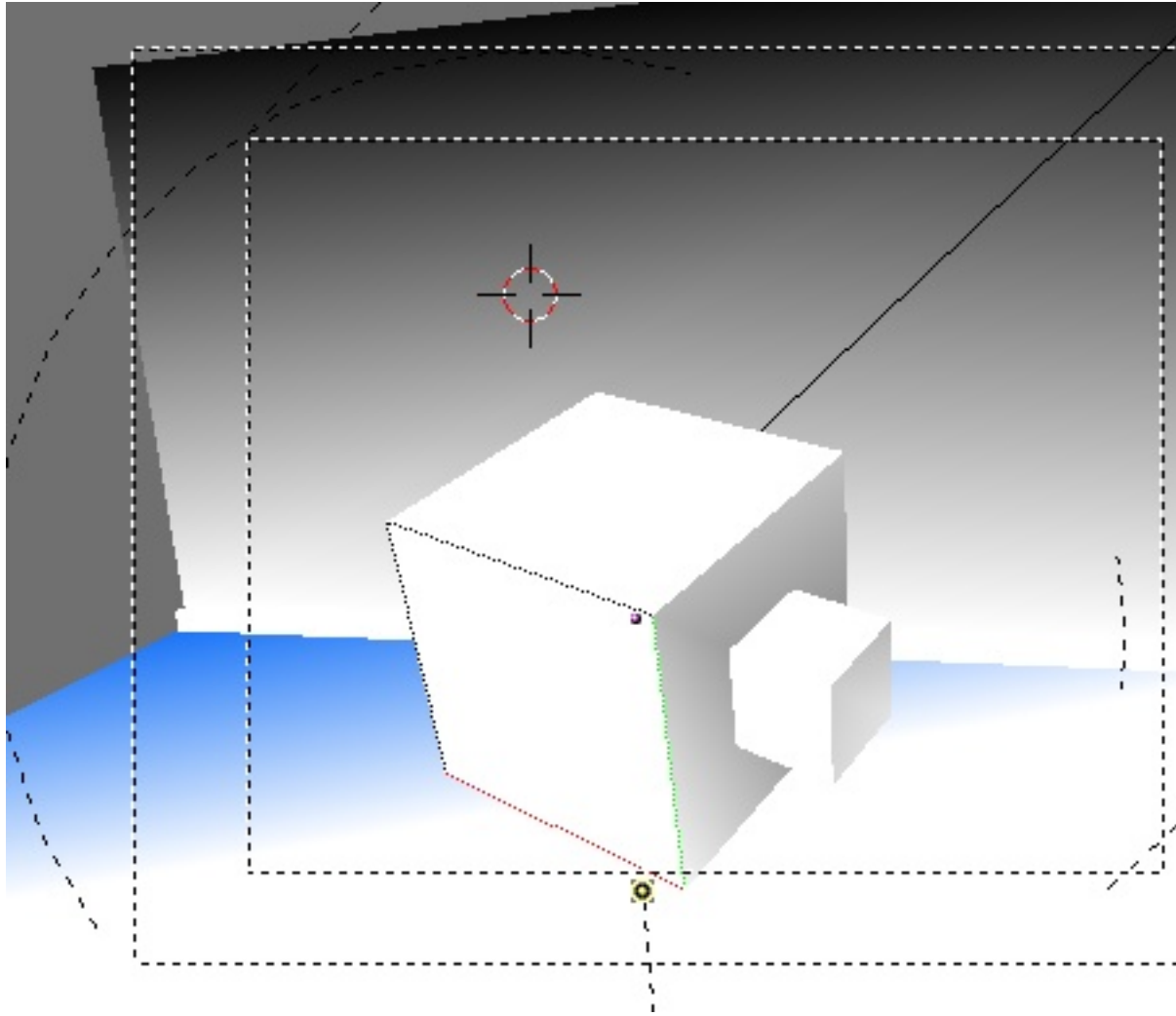


Figure 8-11. vertex colors modulate texture

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

Lamp types

(to be written)

Shadow

(to be written)

Volumetric Lighting

(to be written)

Tweaking Lighting settings

(to be written)

Chapter 10. Animation

Keyframe animation

(to be written)

Path animation

(to be written)

IPOs

(to be written)


Vertex Keys

(to be written)

Chapter 11. 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 Effects" button of Fig. Figure 11-1.

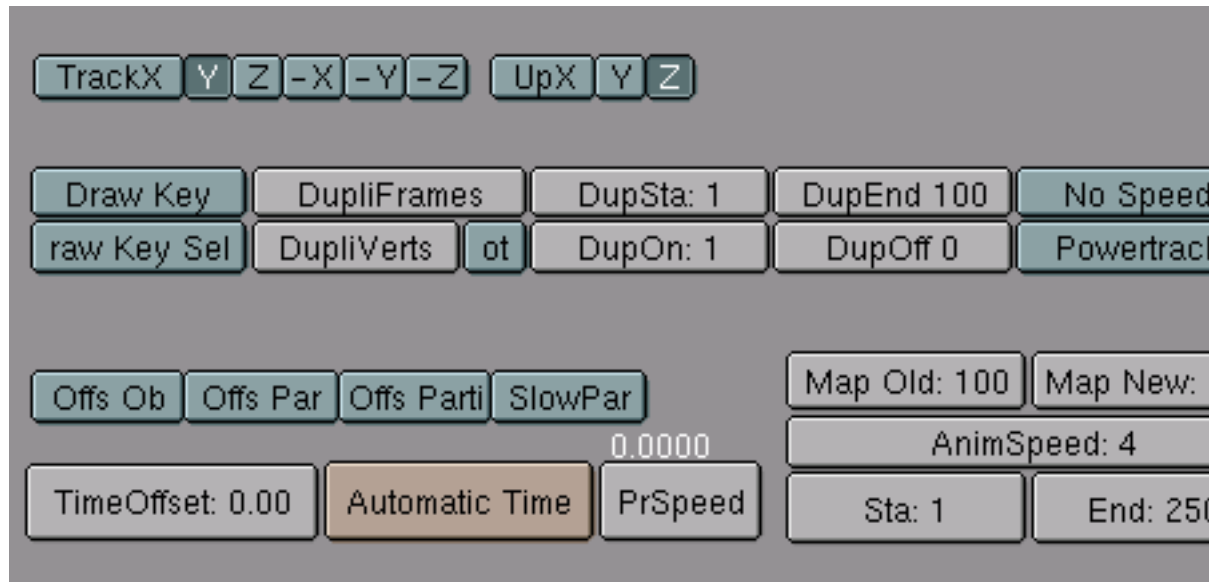


Figure 11-1. Animation Buttons Window

The "Delete" button removes the effect, while the drop down list which appears on the left once an effect is added (Fig. Figure 11-2) select the type of effect.

More effects can be linked to a single mesh, a row of small buttons, one for each effect, is created beneath the "New Effect" button, allowing 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 in detail each of them.

Build Effect

The Build effect works on Meshes and causes the faces of the Object to appear, one after the other, during 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, appears one after the other in time.

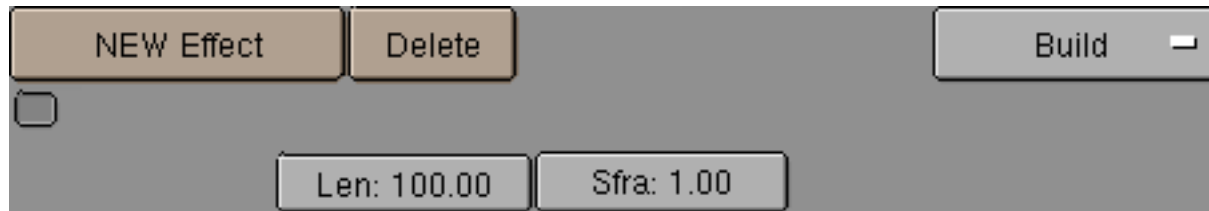


Figure 11-2. Build Effect

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 edit mode. This causes faces to be re-sorted in function of their quote (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 by 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 difference in the quote of the faces, and be able to reorder them.

The Build effect exhibits only two NumBut controls (Fig. Figure 11-2):

Len - Defining how many frames the build will take.

Sfra - Defining the start frame of the building process.

Particles Effect

This is not for me...

Wave Effect

The Wave effect adds a motion to the Z co-ordinate of the Object Mesh.

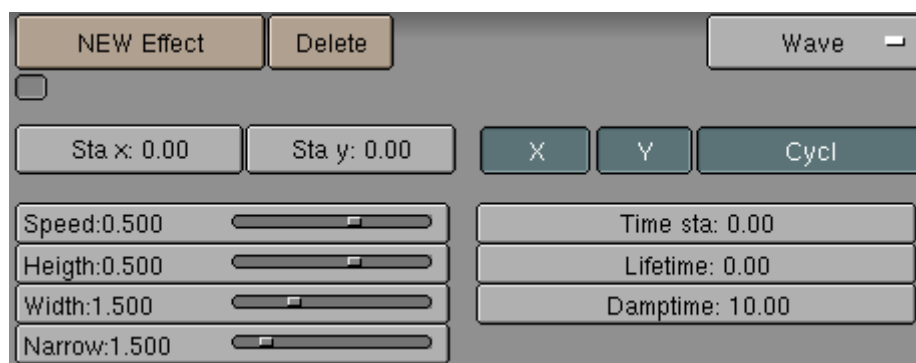


Figure 11-3. Wave Control Panel

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 (Fig. Figure 11-4).

**Figure 11-4. Wave Origin**

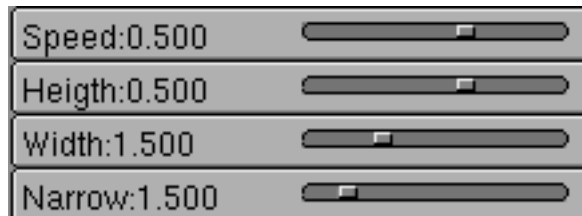
The Wave effect deformation originates from the given starting point and propagates along the Mesh with circular wavefronts, or with rectilinear wavefronts, parallel to the X or Y axis. This is controlled by two "X" and "Y" toggle buttons. If just one button is pressed fronts are linear, if both are pressed fronts are circular (Fig. Figure 11-5) .

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 11-5. 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" (Fig. Figure 11-6).

**Figure 11-6. 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 (Fig.).

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 (Fig. Figure 11-7).

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 Fig. Figure 11-7 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.

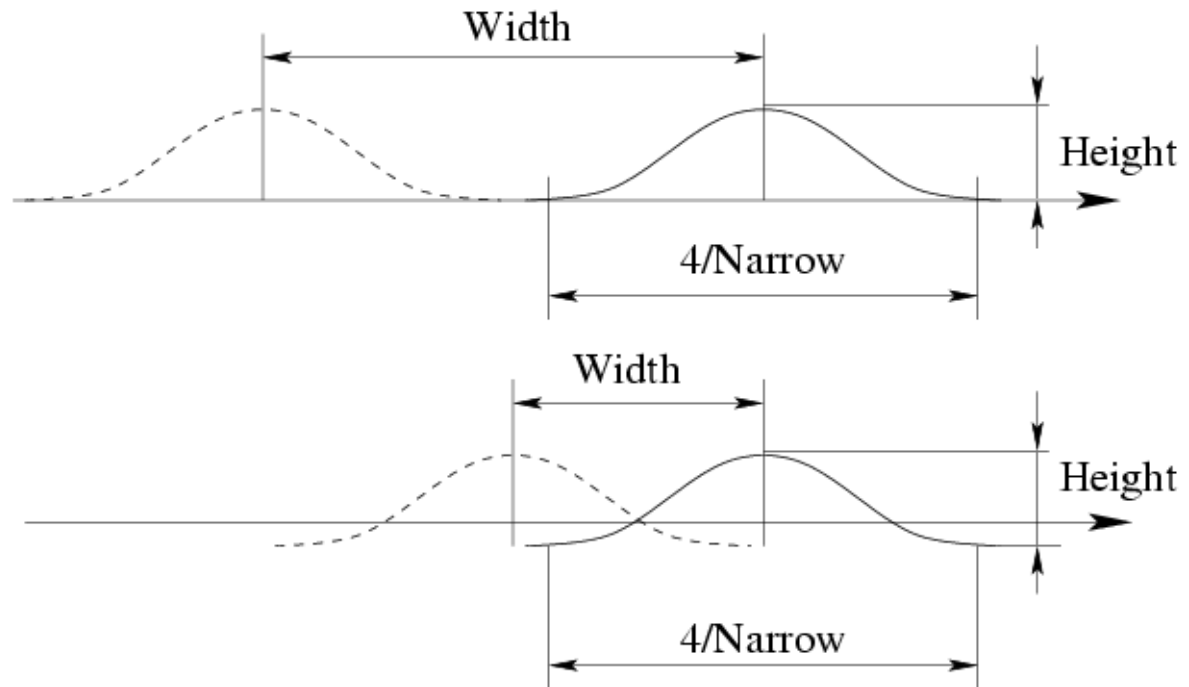


Figure 11-7. Wave front characteristics

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 NumBut defines:

Time sta the Frame at which the Wave begins;

Lifetime the number of frame 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 the on the first frame after the "Lifetime" is over. Ripples disappears di "Damptime" frames.

Time sta: 0.00
Lifetime: 0.00
Damptime: 10.00

Figure 11-8. Wave time controls

Chapter 12. Rendering

Rendering animations

(to be written)

Antialiasing

(to be written)

Motion blur

(to be written)

Output formats

(to be written)

Cartoon shading

(to be written)

The Unified Renderer

(to be written)

Preparing your work for video

Color Saturation

(to be written)

Rendering to fields

(to be written)

Setting up the correct field order

(to be written)

Chapter 13. The Sequence Editor

Editing video with the sequence editor

(to be written)

Using sequence editor plugins

(to be written)

Writing sequence editor plugins

(to be written)

Chapter 14. Character Animation

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 independent of one another.



Figure 14-1. 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.

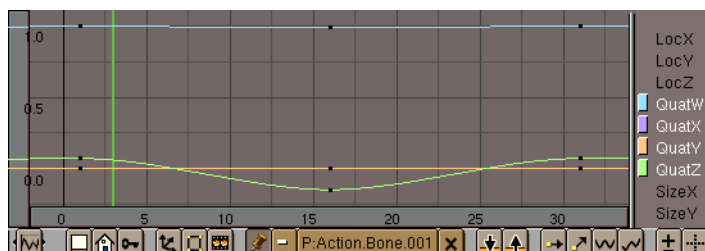


Figure 14-2. Pinned Action IpoWindow

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.

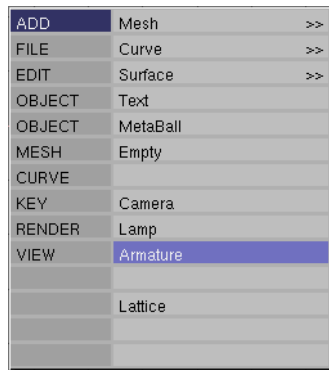


Figure 14-3. Adding an Armature

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 bone 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.

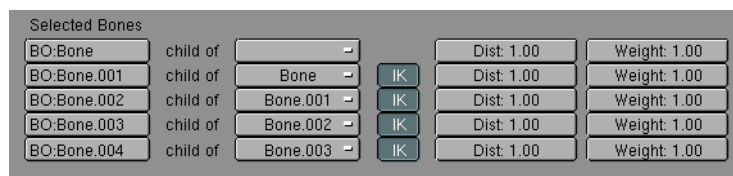


Figure 14-4. EditButtons for an Armature

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).

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 match 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, that 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 (DEPRECATED)

The Weight and Dist settings are only used by the automatic skinning which is a deprecated 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 14-5. 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 (DEPRECATED)

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 14-6. 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

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**.

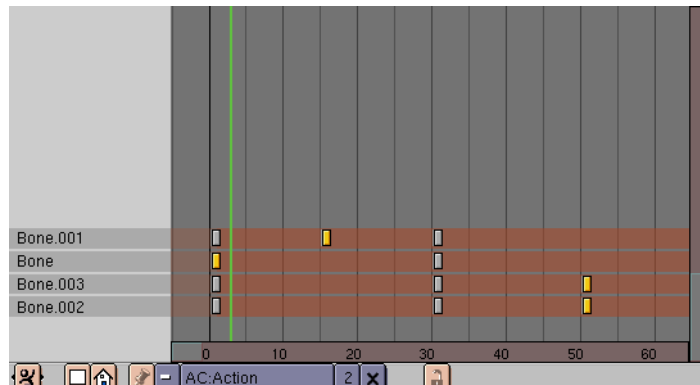


Figure 14-7. ActionWindow

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, there 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.

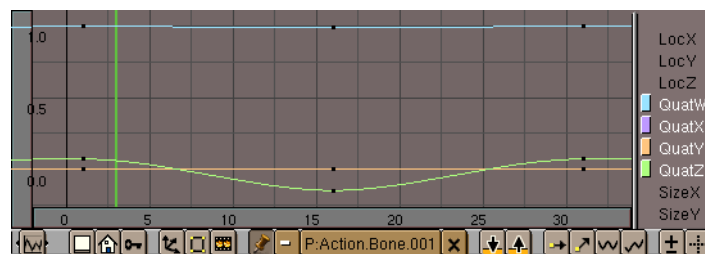


Figure 14-8. ActionIpo

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 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.

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.

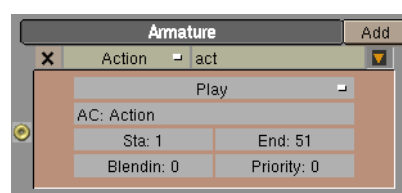


Figure 14-9. Action Actuator

Play Modes

Play

Once triggered, the action will play all the way to the end, regardless of other signals it 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 “Blending” 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 Blending 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.

getBlending()

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)

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**.

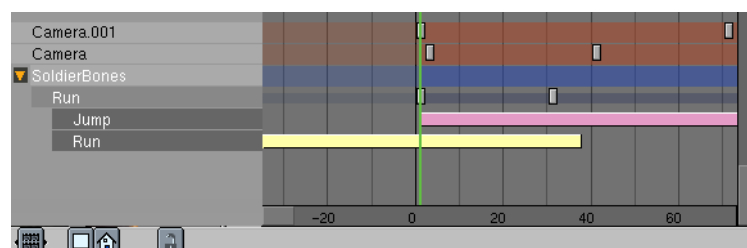


Figure 14-10. NLAWindow

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:

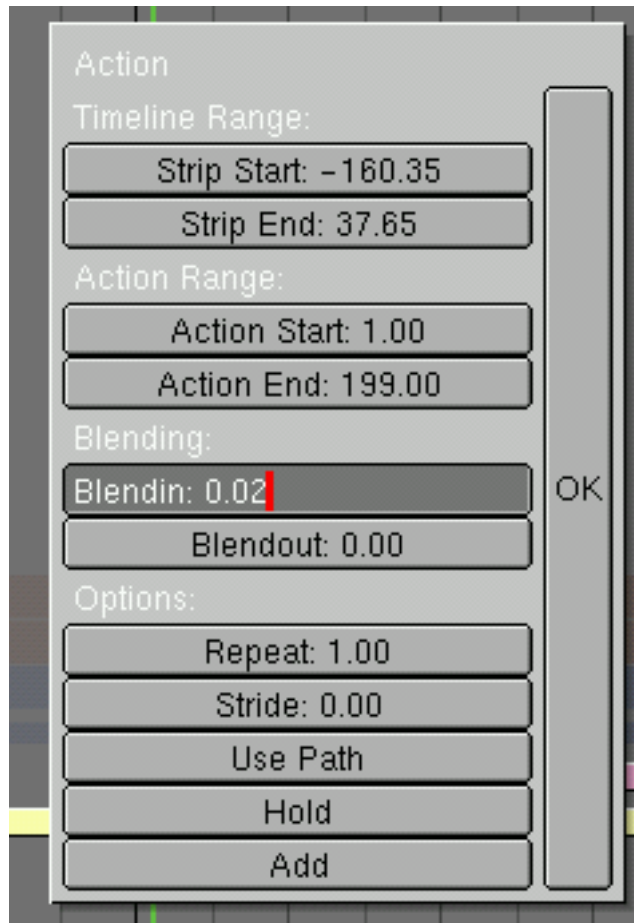


Figure 14-11. Action Strip Options

StripStart/StripEnd

The first and last frame 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.

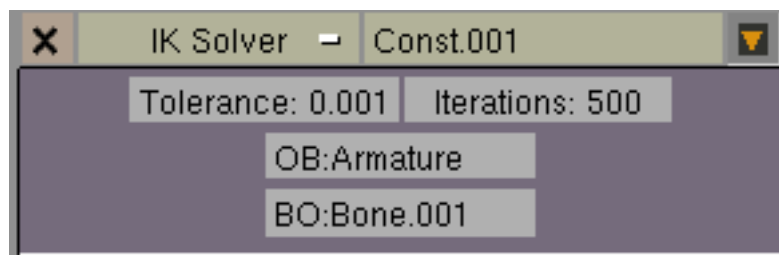
For most constraints, a target must be specified in the appropriate field. In this field you must type in the name of the desired target object. If the desired target is a bone, first type in the name of the bone's armature. Another box will appear allowing you to specify the name of the bone.

Adding Constraints to Bones

To add a constraint to a bone, you must be in pose mode and have the bone selected.

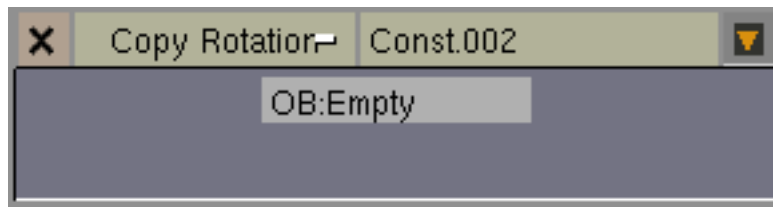
Constraint Types

IK Solver



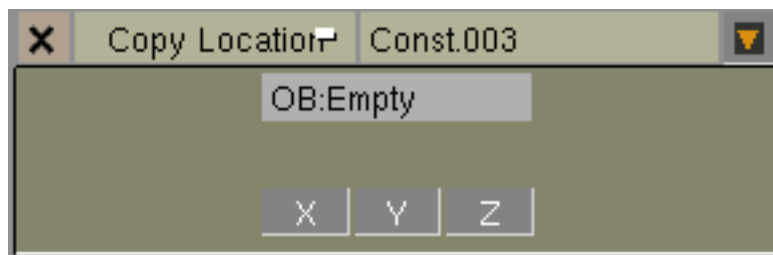
To simplify animation of multi-segmented limbs (such as arms and legs) you can add an IK solver constraint. IK constraints can only be added to bones. Once a target is specified, the solver will attempt to move the ROOT of the constraint-owning bone to the target, by re-orienting the bone's parents (but 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



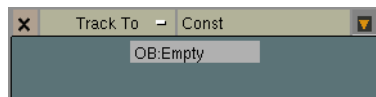
This constraint copies the global transformation of the target and applies it to the constraint owner.

Copy Location



The constraint copies one or more axes of location from the target to the constraint owner.

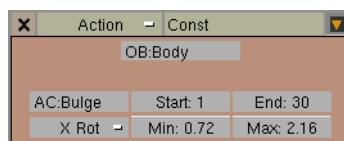
Track To



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.



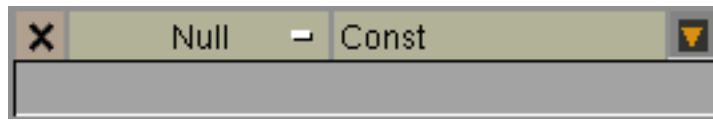
The AC field contains the name of the action that contains the flexing animation. The only channel that is required in this action is the one that contains the bulge animation for the bone that owns this constraint

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.

Chapter 15. Python Scripting

Creating Python scripts

(to be written)

Ideas: how to edit a Python script within Blender; how to attach a script to an object; example scripts

Python modules reference

(to be written)

Chapter 16. 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 17. 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 a 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

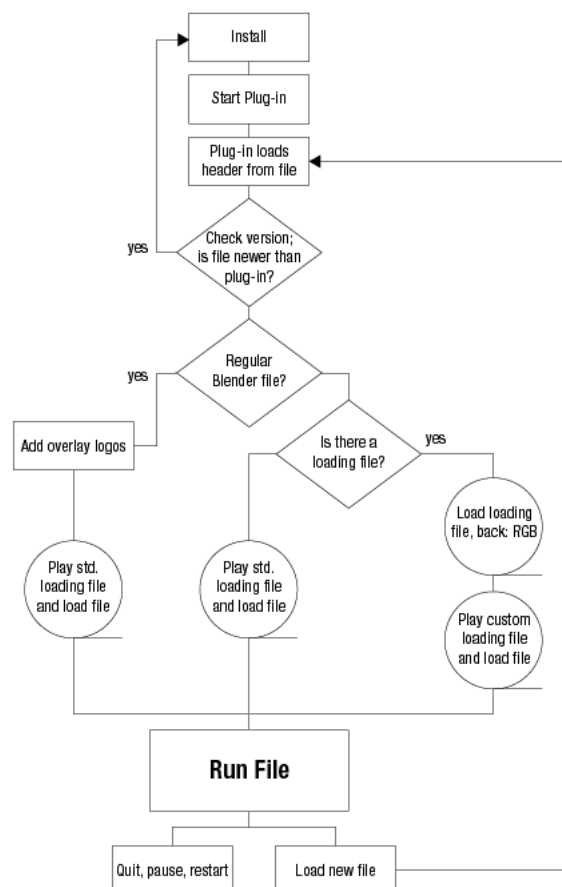


Figure 17-1. Blender 3D Plug-in functionality diagram

Figure 17-1 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 17-2, 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 17-3 has been set to the same values.

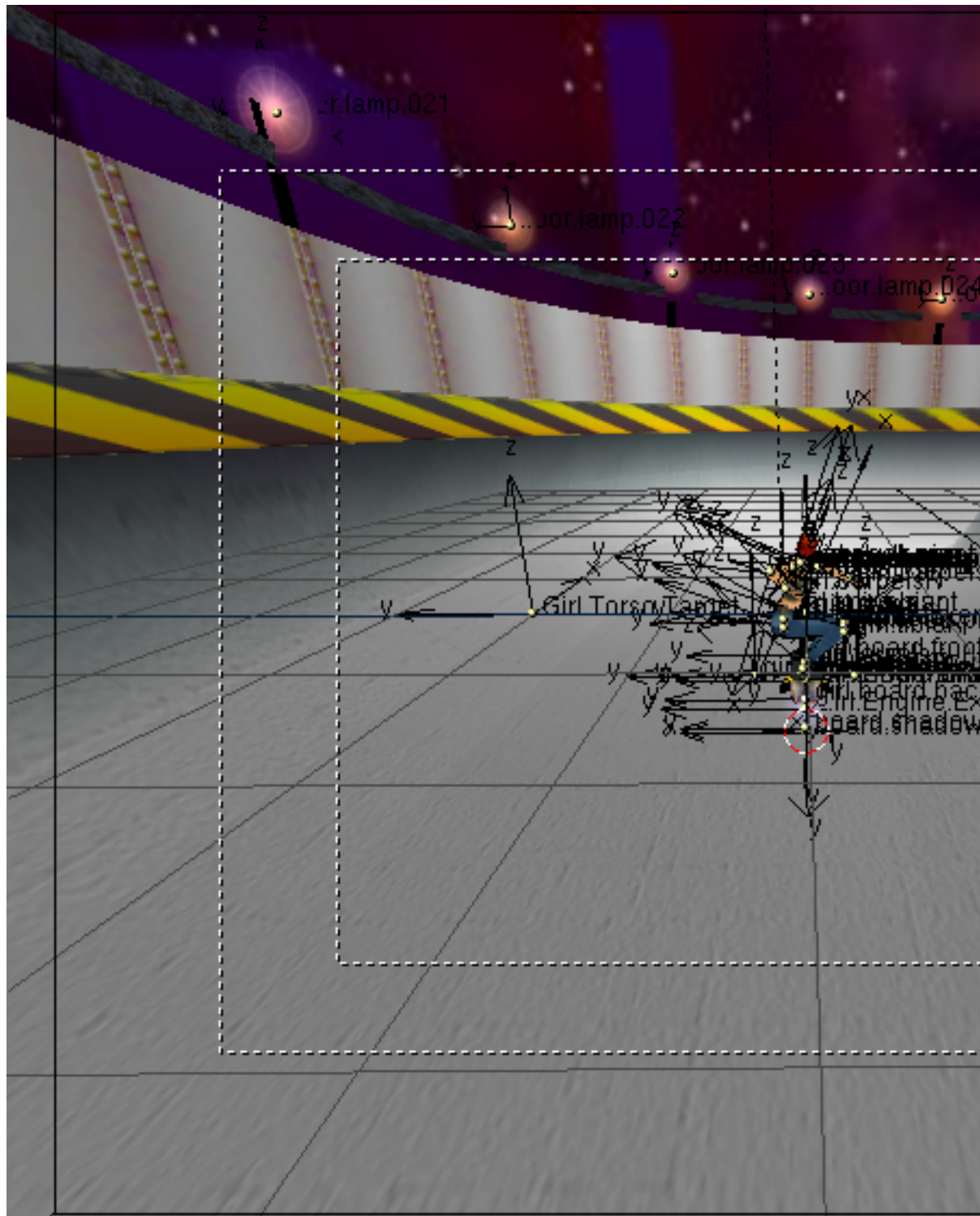


Figure 17-2. File in 3D view



Figure 17-3. Perfect match in the plug-in

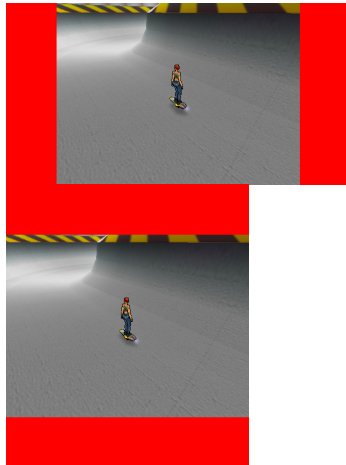


Figure 17-4. Framing of content

In the images in Figure 17-4, 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 17-5 you'll get results similar to the pictures in Figure 17-4.

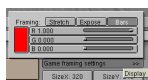


Figure 17-5. GUI for Framing in the DisplayButtons (F10)

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.

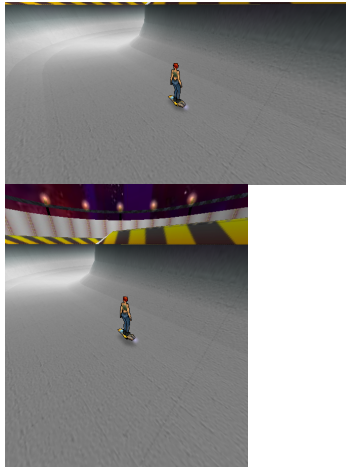


Figure 17-6. Extended framing

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 17-7) but you'll never see bars or more from the 3D world as defined in Blender.



Figure 17-7. 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.

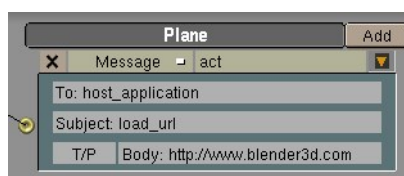


Figure 17-8. Browsing HTML pages from within the Plug-in

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 (2.25) 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"
  PLUGINSOURCE="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:

- PLUGINSOURCE (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 the 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>&nbsp;</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 pages³.

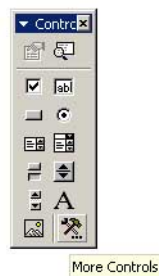
When a control is needed for a certain type of playback, it is automatically downloaded from a secure location, the control installs itself, and then becomes available in Windows. The advantage of Active X controls is that although they are generally downloaded and used in Internet Explorer, once they 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.

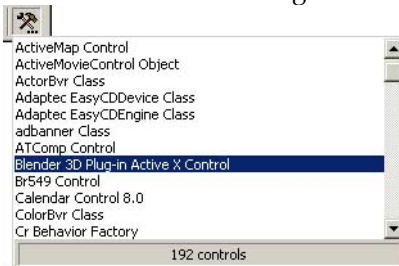
Steps to embed Blender content in PowerPoint 2000:

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.

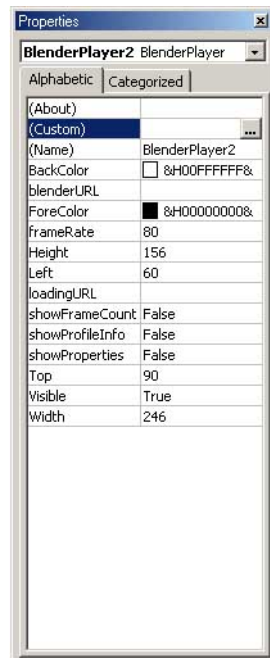


Click on the button to have a list of Active X Controls appear that are installed on the machine.

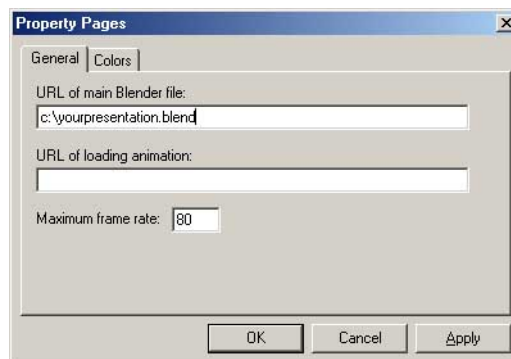
4. Choose "Blender 3D Plug-in Active X Control" from the list:



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 it 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 18. Python Scripting for interactive environments

(game engine specific Python subjects)

(to be written)

Appendix A. Hotkey Overview

Appendix B. Blender Windows/Buttons

Appendix C. Command Line Arguments



(to be written)

Appendix D. Supported videocards

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 may 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 Kryo 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.

Determining your graphics chipset

The easiest way of finding out what graphics chipset is used by your card is to consult the documentation (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 have, 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.

Display Properties

The display properties dialog has many useful 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 acceleration. 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.voodoo.com
3DLabs	Permedia	http://www.3dlabs.com
ATI	Rage/Radeon	http://mirror.ati.com
Intel	i740/i810/i815	http://developer.intel.com
Matrox	G200/G400/G450	http://www.matrox.com
NVidia	Vanta/Riva 128/Riva/TNT/GeForce	
PowerVR	KYRO/KYRO II	http://www.powervr.com
Rentition	Verite	http://www.micron.com
S3 Graphics	Savage	http://www.s3graphics.com
SiS	300/305/315/6326	http://www.sis.com
Trident Microsystems	Blade/CyberBlade	http://www.trident.com

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
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-

Appendix D. Supported videocards

Chipset Manufacturer	Chipset Model	Windows 98	Windows 2000
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

Appendix G. Troubleshooting

Appendix H. About the Blender Documentation Project

About the Blender Documentation Project

(to be written)

Contributors

(to be written)

How to contribute

(to be written)

Getting your system ready for DocBook/XML

(to be written) DocBook environment, XML editors, ...

Learning DocBook/XML

(to be written)

The Blender Documentation Project styleguide

To maintain consistency throughout all Blender Documentation, perspective authors are kindly requested to abide to 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 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 Guideline

The utilization 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/size guarantees that printed images won't be larger than 17cm and hence fills in a A4 printed page.

The usage of images larger than 800 pixel is anyway strongly discouraged since they are too wide for a comfortable reading on a web browser.

Among Blender's Interface most prominent feature is that it's fully OpenGL and scalable. This is great but will sadly result in much disuniformities if many different 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 pixel 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 noone 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. (humor is another matter, of course you can be humoristic!)

Tables

Tables are an appropriate way to display lots of data in a clear structured way. They can be a valid alternate 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

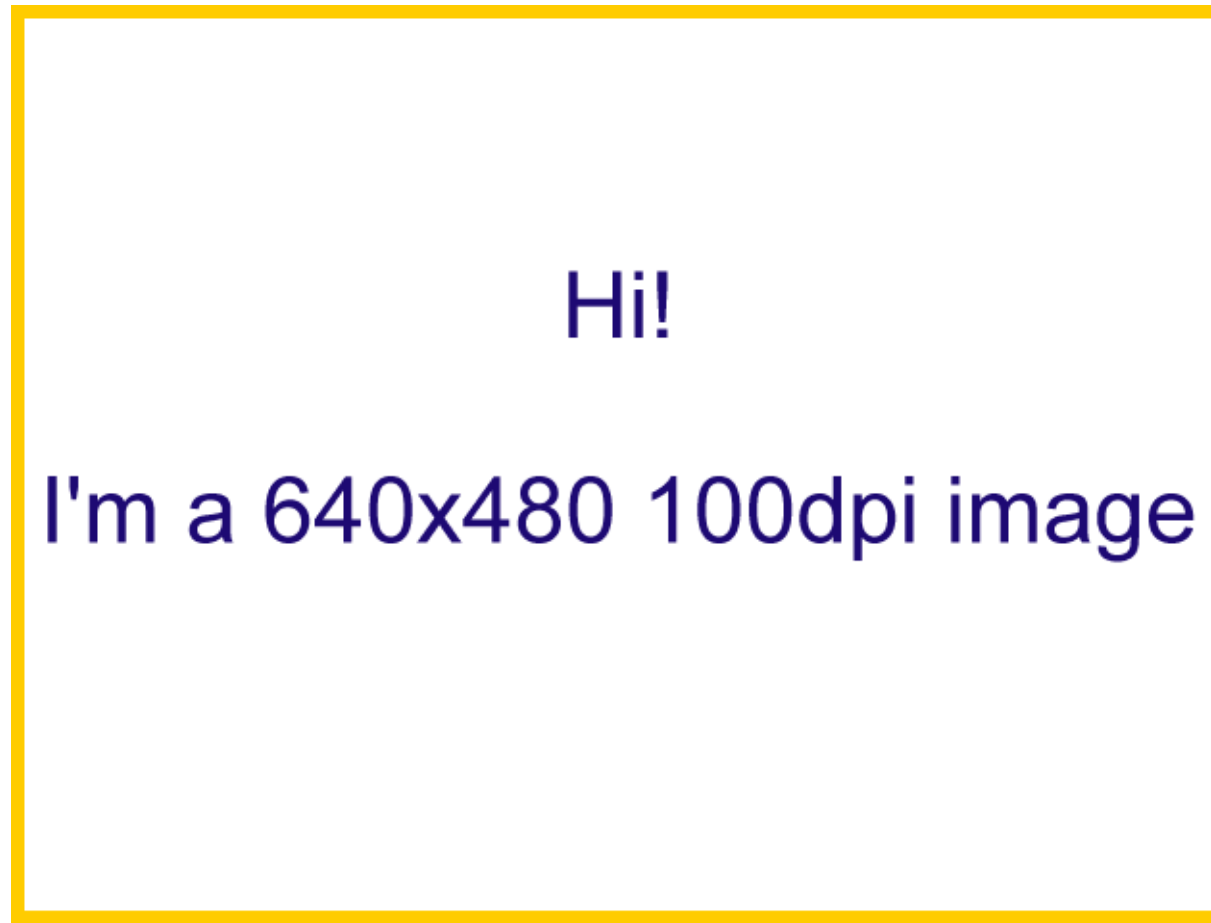



Figure H-1. A demo float image, with its appropriate caption

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 cap-
tion</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"></
</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 in-
clude 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 listing you have already understood half of how cross-reference 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 adviceable 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
- Lightning
 - 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 references 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 up to the author to define;
- TID is a three letter tutorialidentifier;

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.

Appendix I. The Documentation Licenses

Open Content License

This is the Licence 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 by 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 the Section called *Blender Artistic License*.

The source of the Open Content License is <http://opencontent.org/opl.shtml>. It is reported 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 Licence designed for Tutorials, .blend example file still images and animations. It is more restrictive than the Blender Documentation License and is thought to protect more the intellectual rights of the artis 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 adviceable to prepare zipped package containing, besides the Tutorial etc. files, a file calle LICENSE containing this license.

If you are distributing a .blend file alone it is still adviceable to add a LICENSE file, otherwise you can add the license in a text buffer of Blender and make sure it shows up wen opening the file.

If you are releasing binary files like printable tutorials and you don't want to include the whole licens (why you would not?) 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 package 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 Version 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 MERCHANTABILITY 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.

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

Mode to select and transform vertices of an object. This way you change the shape of the object itself. Hotkey: **TAB**.

See Also: 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.

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. Blender's whole interface and 3D output in the real-time and interactive 3D graphic is done by OpenGL.

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. Python¹ is an interpreted, interactive, object-oriented programming language.

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, Selected, 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/>