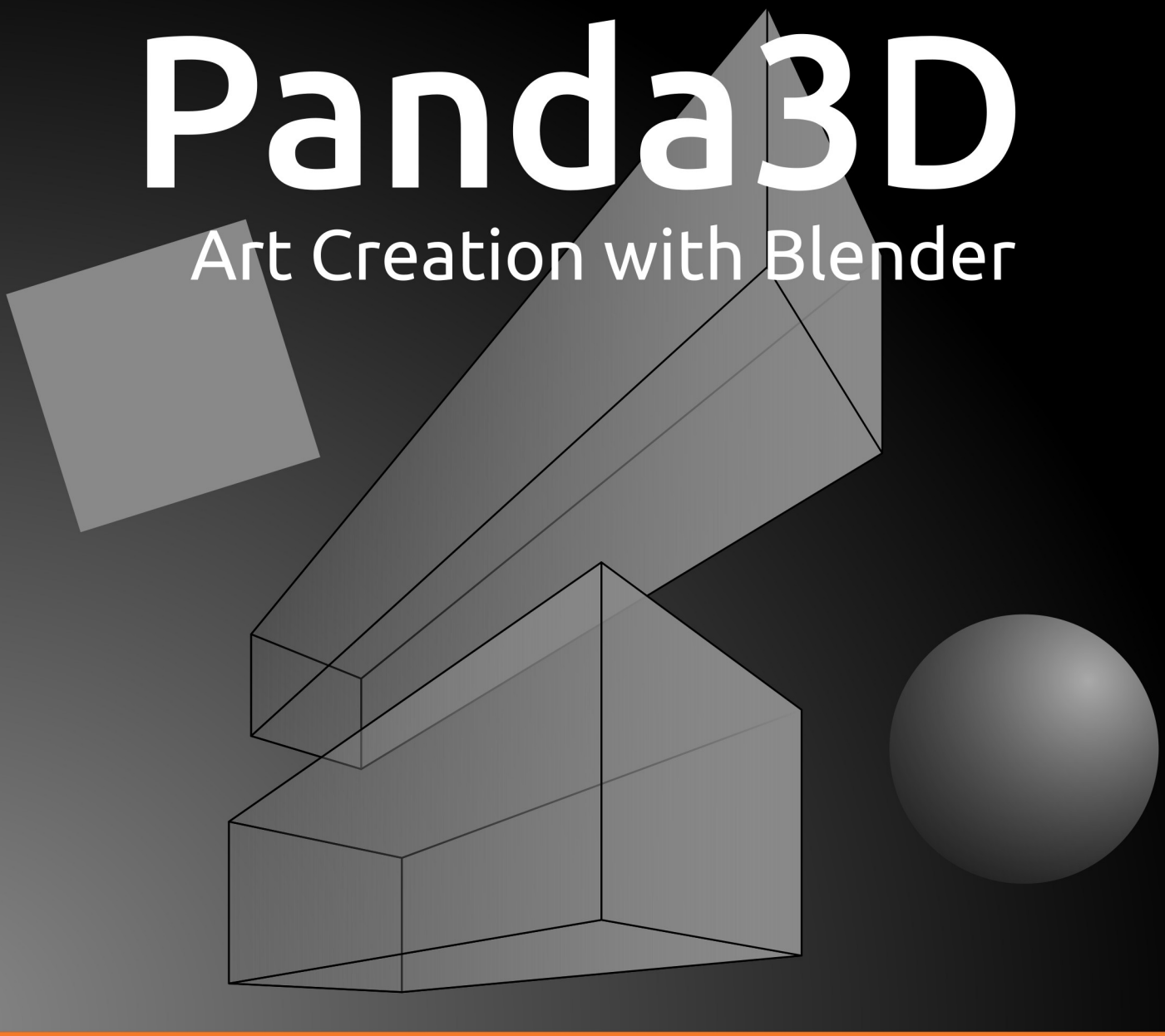
Panda3D

Art Creation with Blender



Game development for everyone

A complete Tutorial about creating
3D Art for Panda3D with Blender

Fireclaw the Fox

{GrimFang}
open source game studio

Content

Copyright

Version 16.04

Introduction

This is the second book about game development with Panda3D and other open source tools. It is tailored towards 3D Artists which like to create game ready assets with some information specifically useful for using them in Panda3D applications.

As just noted, this book is mainly for artists. If you are a Developer, you should take a look at the first Panda3D Tutorial book, which is about developing games with that engine.

Though, even if this book isn't intended for coders, it will have a few notes here and there that will give artists a little insight of Panda3D applications which is necessary to understand why some of the parts need to be done as described to work correctly and performant in the resulting games.

If you know blender, you may also know how rapidly it evolves. We may not be able to hold this pace for updating this tutorial book but we try to keep the descriptions here as general as possible and as detailed as necessary, so you should be able to follow this book with newer as well as older versions. If there are specific functions only available from more recent versions of Blender, we will add a note.

Also note, this book even though it has some beginners description, won't have to much in depth information about most of blenders features as blender as an application is just to big to fit into this one book and there are many other, better tutorials out there which we can't compete with, so this book tries to give you an overview of what needs to be done for creating game ready models as well as some guides on how to achieve the desired results. Also as Art creation is a really creative task and it always changes from project to project and artwork to artwork, we won't have many details on the modeling and texture creation itself as those can't really be generalized. Though we try to put as many helpful information in it as possible that may help you in your process but won't disturb your creative processes.

Now, if you already know how to model game ready assets and just want to read this to get into modeling for 3D games with Panda3D, you may want to skip the first few chapters and first take a look into the exporting section which also has some notes on what you can expect if you export your models.

Aside of that, you should make sure to have a look at http://www.panda3d.org/manual/index.php/Pipeline_Tips which gives some handy tips and notes on what you should take care of as a designer when it comes to building 3D assets which are used by software developers.

Chapter 1: Basics

In this first chapter, you will learn how to get Blender installed on your system as well as the basics to use the application.

Blender is quite a big tool and has many features in it. As we won't be able to tackle them all, we will try to give you a good overview of the most important but also most basic features in this chapter. Those include navigating within the 3D space, the Layout system of Blender as well as very simple interaction with the objects in the scene.

If you already know Blender and how to use it, you may skip this section and head right over to chapter 2 if you're interested in Character creation or chapter 3 if you're an environmental artist. For those who only glue together existing models and want to create astonishing worlds from those, head over to chapter 4. If you already are a professional in creating 3D assets and know how to do all the modeling and setup and may have previous experience with using Blender as modeling tool for Panda3D but have used another exporter and now want to know how to use YABEE, head right over to chapter 5.

Getting Blender

First of all, you need to download and install blender. For the most recent version of the application, head over to https://www.blender.org/download/ and select the download for the system you use.

Dependend on your download, you may or may not need to install the application. In case you downloaded the bz2 or zip just unpack it and run the blender executable within it, that's all. If you downloaded the msi installer, double click it and follow the instructions on screen.

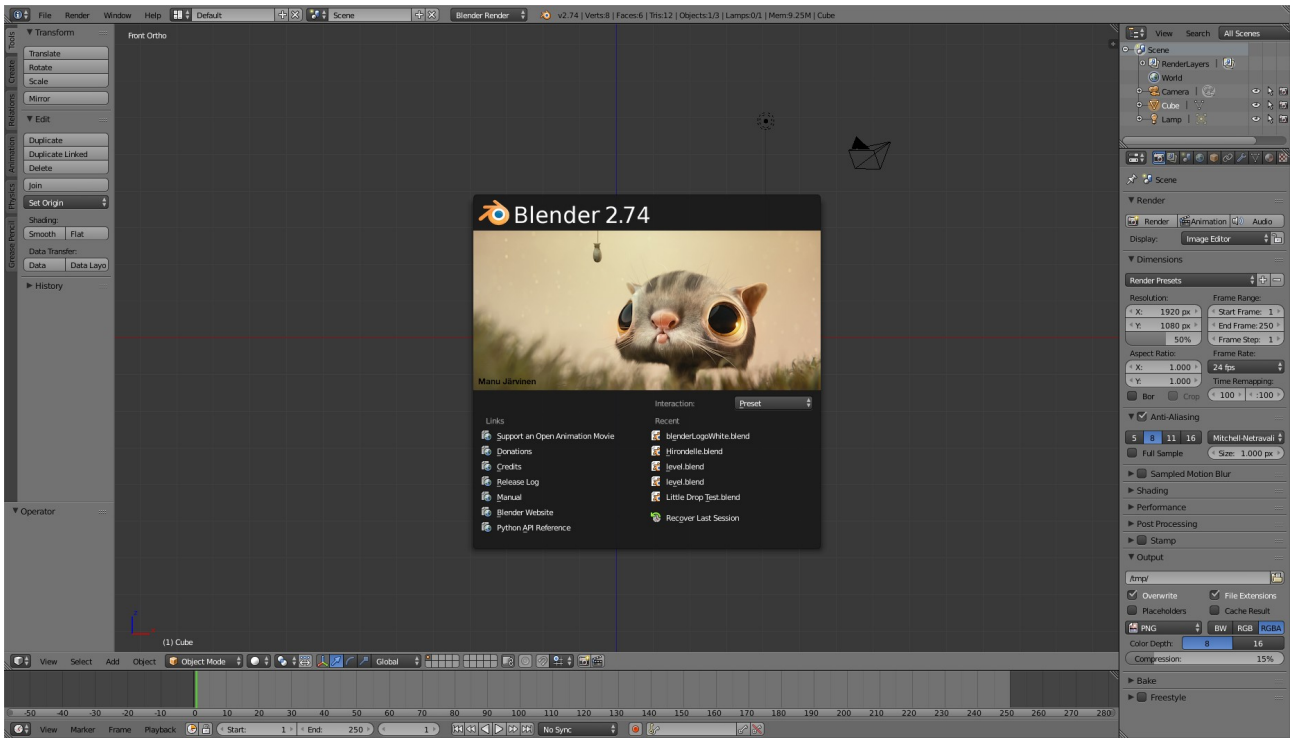
Most Linux systems also offer blender through their software repositories. Those may not always be most recent versions but should fit for this book and most of the tasks necessary for creating game ready models.

The version which was used for this book is 2.74 from the ubuntu software repository.

To start Blender, either click on it within the start menu if you have it installed, or double click the blender executable as found within the unziped directory.

As soon as you have blender ready and started, move on to the next section which will give you an overview of what you see after you started the application.

The Layout of Blender



This screenshot shows the splash screen of blender which will always come up first. The splash screen can be used to open recently edited blend files, gives you some helpful links at hand as well as change the interaction mode to for example simulate another modeling tool like Maya. We don't need anything of that yet so click anywhere around the splash screen to close it and continue to the main window.

Following we will describe, what you will see in the main window. On the top edge of it we have the Info bar.

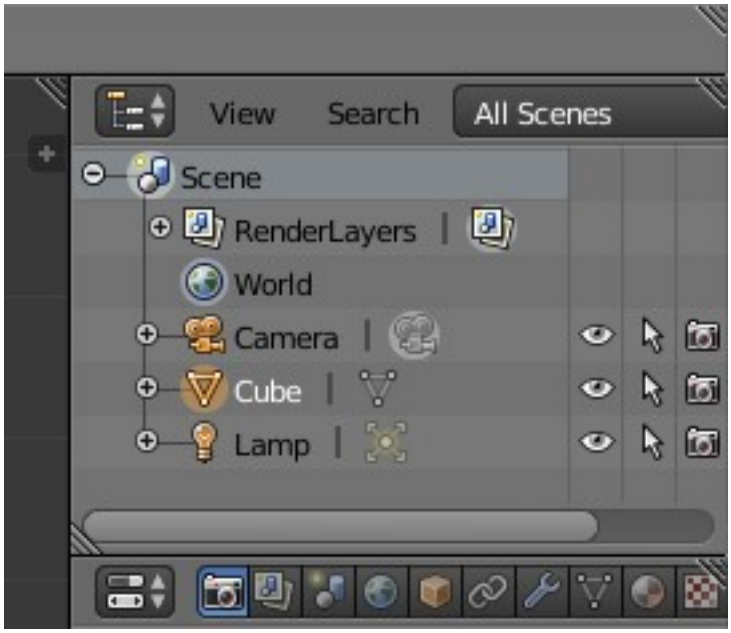
On the info bars left side we can reach default functions like file loading and saving as well as adding basic models, editing the screen layout and Blenders Help.

On the right side, we have information about the currently selected model and the active scene.

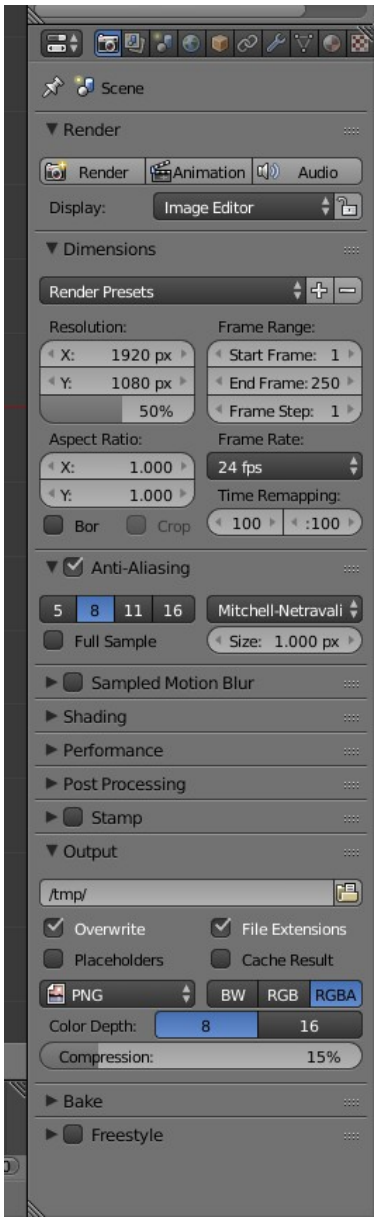
In the middle of the screen we have the 3D space, in which you will spend most of the time with editing your models. Attached to it you will find on the left side, the tools panel. In it you will find

useful functions for editing your model and context sensitive information for example if you add a new model.

To the top right of the 3D Space we have the scene graph browser which displays a list of all objects placed in your 3D Scene. ↓



Right below it we have the properties panel. In it you will find all the properties of the scene and the currently selected object. This are things like model modifiers, material and texture settings. →

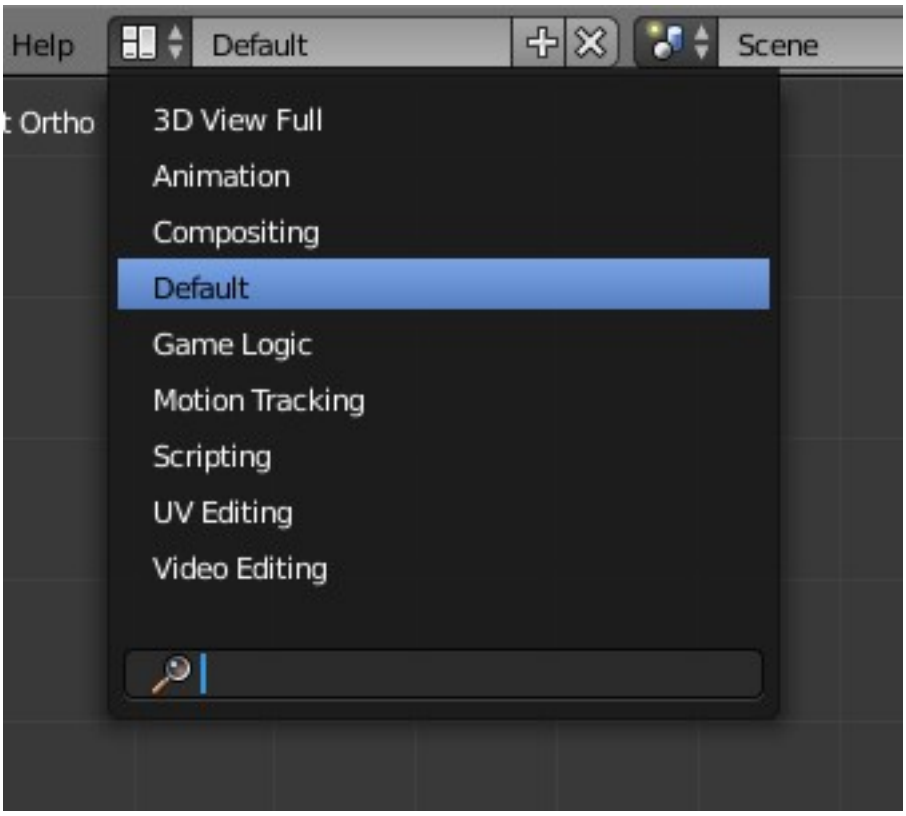


Screen Layouts

Now that we know what is where in the main screen layout of Blender, we will take a quick lock at other layouts for different purposes. I will describe you only the ones that will be used within this lectures and leave out the others.

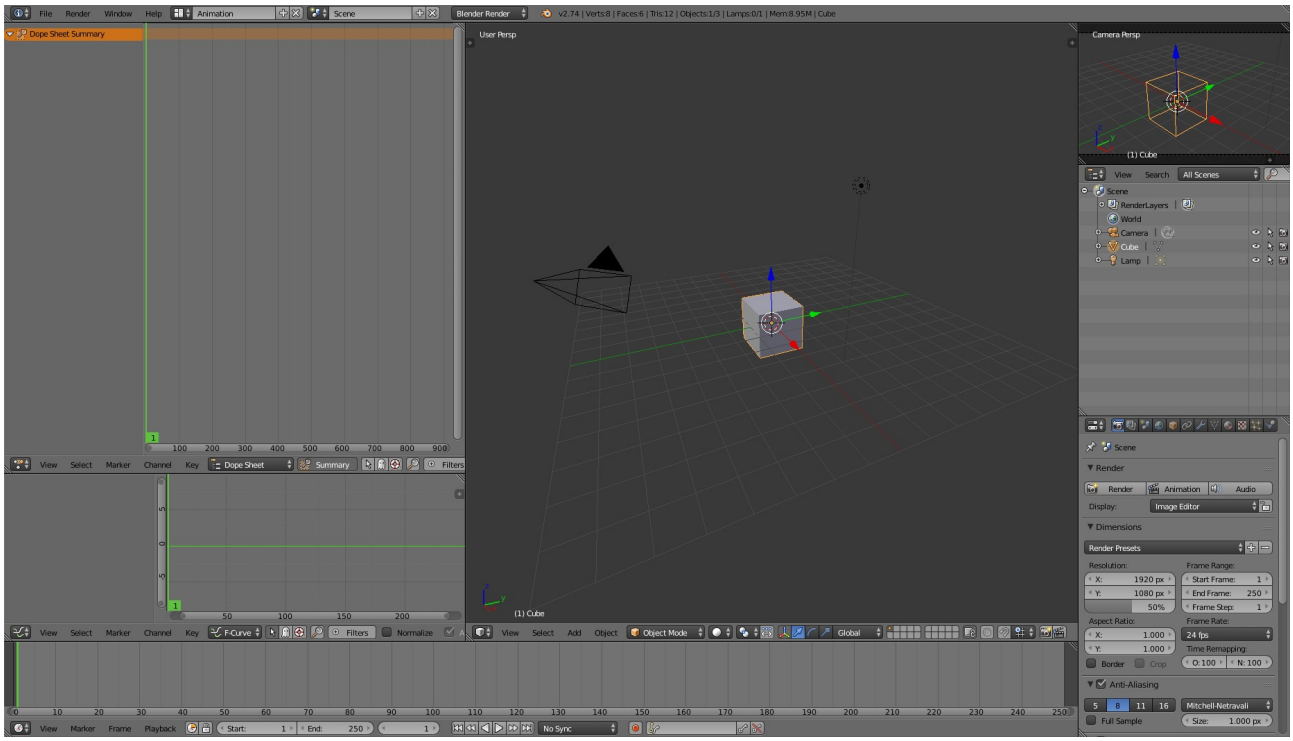
Up in the info bar, in the box which currently reads Default, we have some predefined layouts. We will now go through some of them which are useful for this book. The other layouts won't be used in this book but you can still use them as you see them fit in your creative process.

You can also edit existing layouts as well as create and save completely new layouts to enhance your productivity and workflow even more.



Animation Layout

Starting with the Animation layout.



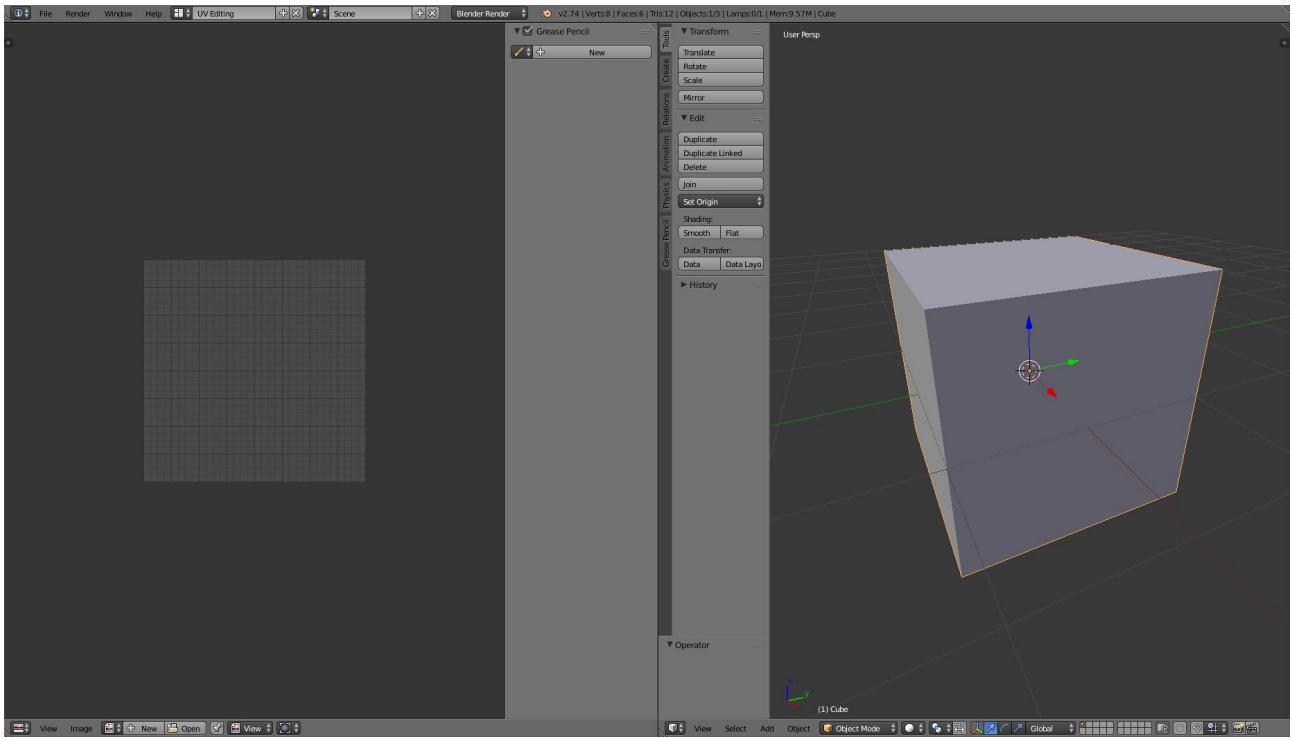
This layout is most useful for adding and editing Animations. Here we have the already known 3D space panel, a smaller second 3D space and scene graph and properties panel. Now let's see what's new here. On the top left side we see the dope sheet editor. It will show key frames of each part of your models animation. Currently it's empty but it will fill if we get to the rigging and animation chapter of this book.

Below the dope sheet we have the graph editor. It will show lines that represents the curves of animated parts of the model, whereby the curves for example represents the x, y and z position on the timeline.

Now the long panel below the graph editor and the 3D scene is the timeline, it displays the frames of your animation and can be used to quickly jump between frames as well as start an animation.

UV Editing Layout

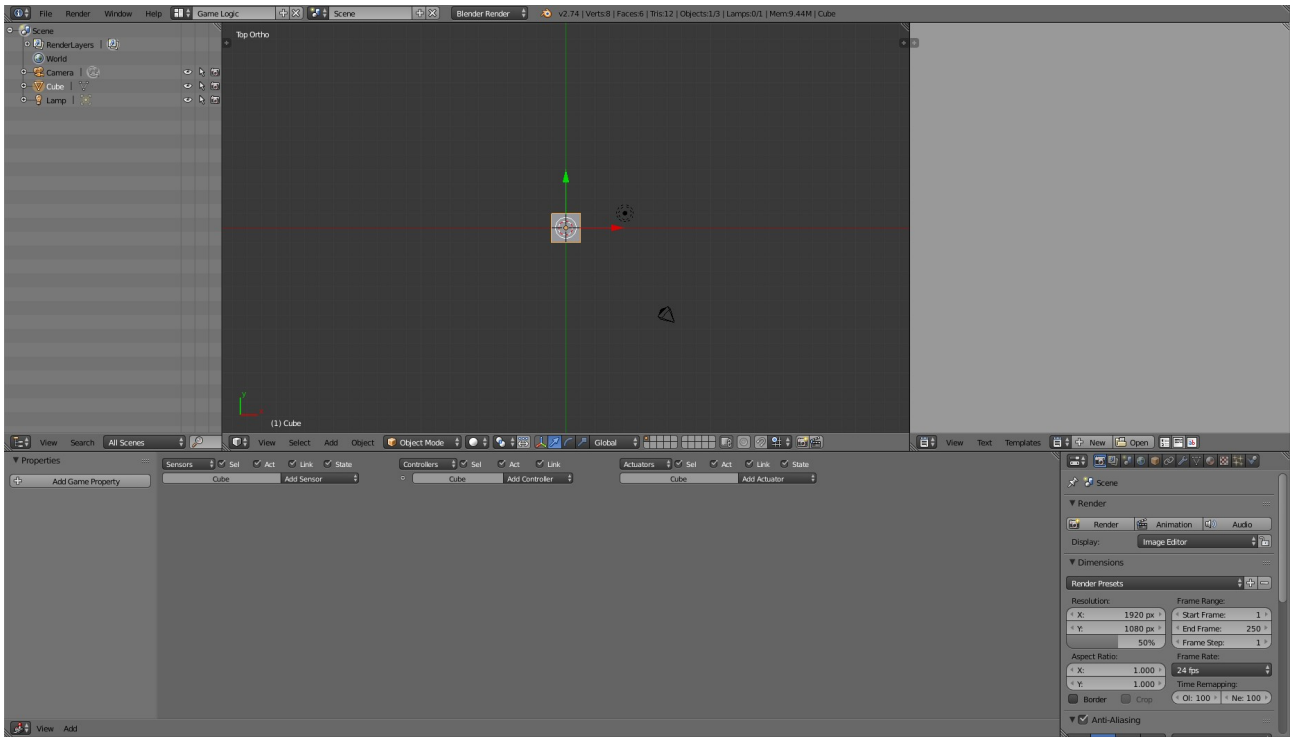
Now let's take a look at the next Layout, the UV Editing layout.



This Layout is a quite simple but also very important layout. In here you will fit texture images on your Models by editing the UV map. The left side represents the 2D view of the current active texture of the selected model or any texture you select from the UV/Image Editors info bar. You can also create completely new images in this editor. More on that layout and it's usage will be taught in the texturing chapter of the book.

Game Logic Layout

Now we move on to the last Layout we will use in our lectures, the Game Logic.



This Layout is mainly intended for usage with the Blender Game Engine, but we can also use a small part of it for our models, namely the Game Properties. Those Properties can be added in the left side of the Logic Editor panel. Those properties get exported in the models for our game projects. More about properties will be shown throughout the Level creation lectures.
That's it so far for the Layouts.

Navigating the 3D Space

Now back at the default layout we will take a look at the navigation in blenders 3D scene. At first you should know, that by default, the navigation system uses a pivot point navigation. Means you will always look at a specific point in the 3D space where you can rotate around, zoom towards and away from as well as move the point to another place to navigate through the scene. At the beginning your pivot point is located at the 0,0,0 Point in the 3d Space which is visualized with the orange dot you can see in the center of the 3D scene.

Now to rotate around it, press and hold the middle mouse button and move your mouse to the side and up and down.

Using the mouse wheel you can also zoom in by scrolling up and zoom out by scrolling down.

If you hit the shift key, press and hold the middle mouse button and move the mouse, you will shift the pivot point on the current view axis.

You can also move to predefined views with the number pad of your keyboard whereby 1 is front view, 3 side few and 7 is top view. 2, 4, 6 and 8 will rotate the view in 15° steps around the point in the respective direction. With the 5 on the number pad you can change between the perspective and orthogonal view, by default you start with a perspective view.

That should be enough about navigating the 3D space for now. Let's take a look at interaction with the 3D objects.

A simple right click will select the object below the cursor and shift right click will deselect an already selected object or lets you select multiple objects whereby the last selected object will be the active one.

The orange colored outlines signalize that an object is currently selected whereby a lighter colored orange marks the object that currently has the focus of all selected objects. To select or unselect all objects in the scene, hit the 'A' key.

To delete objects simply select them and hit the 'Delete' or 'X' key.

To add a model, hit the space bar. This will show up a search window which gives you the ability to search through all the possible functions available in Blender. Now, for example to add a cube simply type “add cube”. You will notice, that the object is placed in the center of your 3D scene. This is due to the 3D cursor being positioned in the center. If you left click anywhere in your 3D

scene, you will move the cursor around. If you now add another object, you notice it will be placed in the new location.

Now to change an objects position, select it and then either grab the little arrows that occurred on it or hit 'G' to “grab” the object and move it around. During the grab mode you can apply the position with the left mouse button or cancel it with the right mouse button.

Similar to the position, you can also change the rotation and scale of an object, therefor you can switch between the modes in the panels header. Or using the 'R' and 'S' key to rotate and scale respectively.

That's all for this Chapter, you now know the layouts we use and how to navigate in the 3D scene as well as doing simple editing tasks. In the next part we will take a look at our first real modeling tasks and what you need to consider for game ready models. In addition to those sections, you should also read/view/listen to some other basic blender modeling tutorials. At the end of this book I'll list a few links to tutorials which are quite helpful for general and game specific modeling.

Chapter 2: Character Creation

This chapter will teach how you can create a character step by step, starting from a simple idea to the finished 3D model, fully textured and animated.

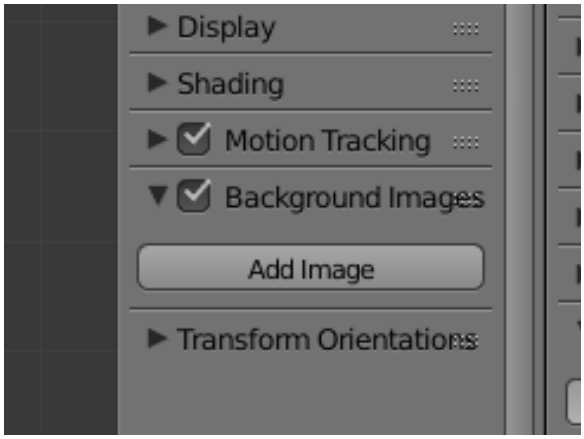
We won't go into to much detail about the creation process for designing a unique avatar. Instead we will go into more detail on the actual building of your designed characters and give you hints on how to create them in 3D.

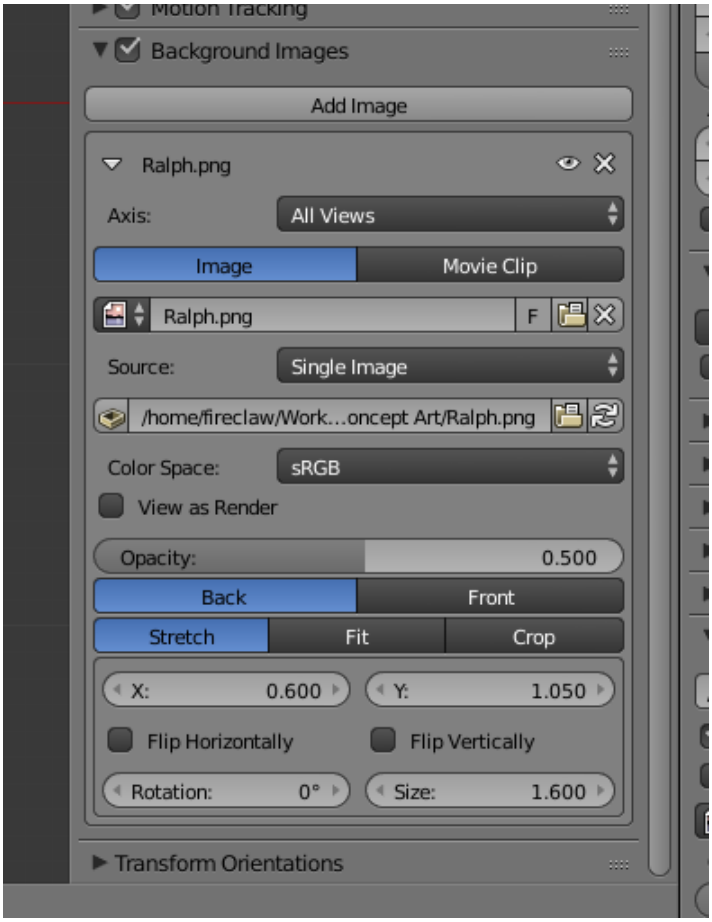
Getting Started

Before we start creating our first 3D model, we need to have a few things done in advance. First of all, we need a good idea and the general design of our character. Those can best be done with some traditional 2D drawings, so get to your sketchbook and lay down some ideas. As soon as you have a good visualization of your character with all it's little bits and pieces, you should draw a reference sheet showing the characters front, side, back and optionally top, bottom and any other angle views. Those can then be used within blender as background images to simple create your model along the outlines of those drawings.

To get the image of your character as background in blender, open the side panel on the right side of the 3D View by hitting the small + sign or press 'N'. Within this panel, scroll down until you see the Background Images section. Expand it if it isn't already expanded and click the Add Image button.

Next click the Open button and select the image you want as background. You can also load a movie with this technique which you can for example use for animation. Now with the opened image, you have



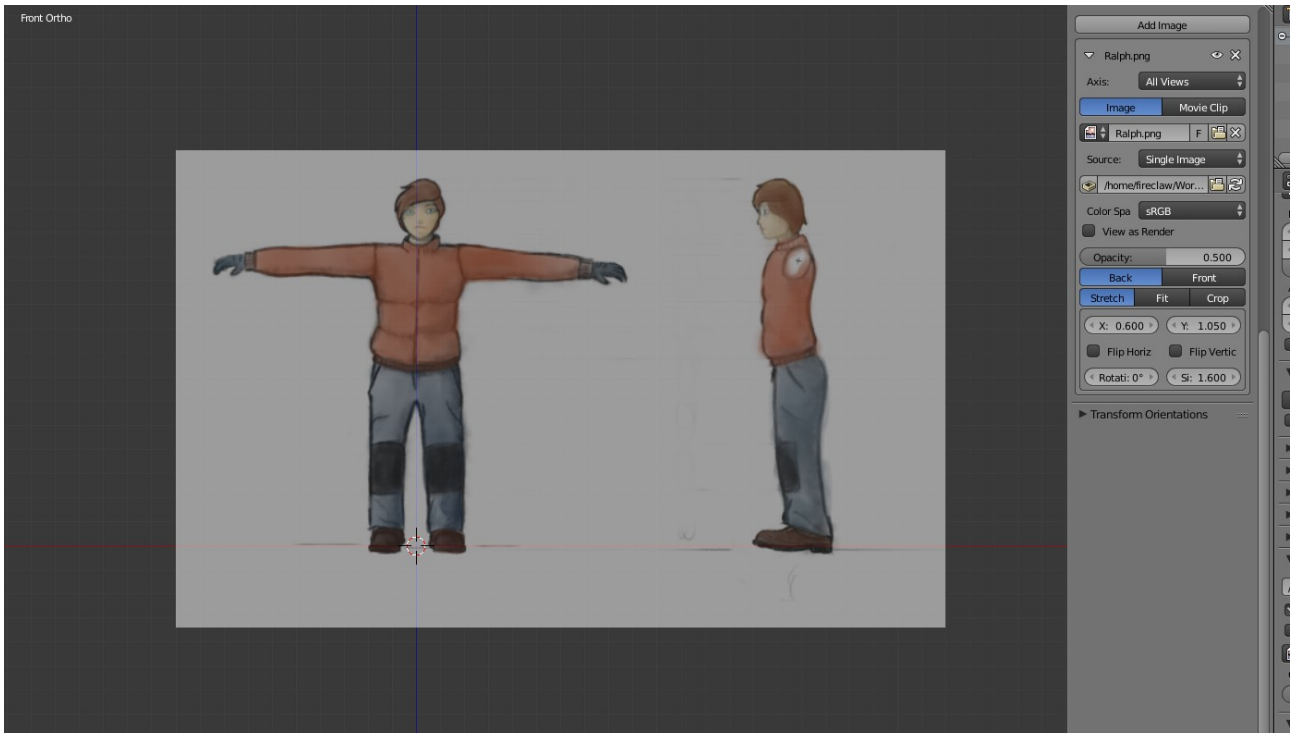


quite a few options to fit the image to your 3D scene. Most important are the X, Y and Size values which you can use to place and scale the image to fit the actual characters model. Also you can change the Axis option to see the image only in certain view directions like front or left/right side.

You can repeat this steps and add as many images as you need to simplify your modeling task.

As for the position of the images, it is recommended to place them in a way, that

the foot/ground line of the character is placed at the 0 point on the Z-Axis. This will simplify the process of adding them correctly to the game without needing to know each characters size. You could possibly also set the 0-Point to any other position of the character, but you should then make sure the developers know where it is so they can correctly set the position within the application.



Modeling

Now that we have that set, we can start the actual modeling. There are many ways to do a character modeling and as it's still a creative process, there are no right or wrong ways in creating an avatar as long as you get to a good result in the end.

One way for example is to model a lowpoly representation of the character either from scratch with the image you have in your head or, by using background images as we have set in the previous section. Afterward you can copy that lowpoly model and use sculpting tools to create a high detailed version of the model which will be used for creating things like normal maps and the like.

Another way would be to first sculpt a high detailed version of the model and then use that to create a lower detailed model on top of that. Blender even have some features to simplify that task for you for example with the Decimate Modifier and the snap during transform tool which enables you to snap vertices directly on the surface of your highpoly model. This technique is also called retopologize or retopo for short.

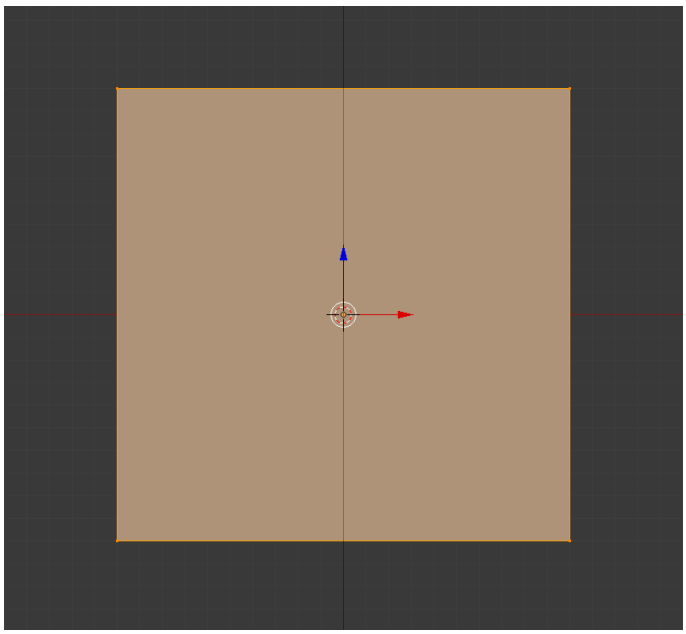
In the next parts we will go into more detail for the first way, creating a lowpoly representation and afterward a detailed version as it should be quite simple and if you know the basics of this technique, it is quite simple to change the workflow to a retopo one as it basically only reverses the workflow.

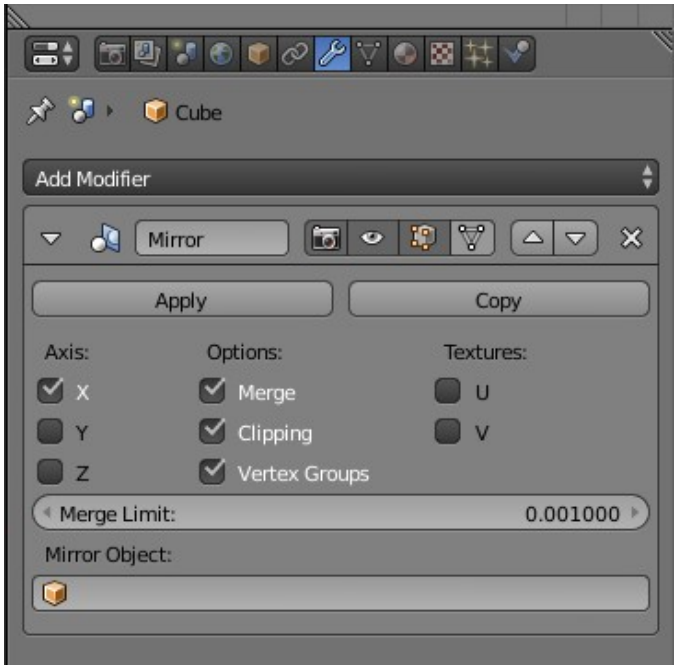
In the end it doesn't mater which way you take as long as you get a good looking lowpoly model for the application you need it for. In most cases it must be a lowpoly version of the model as otherwise with all other game elements, not only graphical but also mechanical like the AI and other game logic, the game would not run smooth on most machines. It's always good to take a look at current AAA games and try to figure out, how many polygons they used for their models and how detailed the elements are. As one can't really say how many polygons are to much as it always depends on the type of game and how much is going on in the scene, you always need to check back with the game designers and developers on how many you should give your characters and other game elements. Characters in current games tend to have a triangles count of 2-5K in RTS games with many characters on screen going over 20K for FPS console titles to about 100K for next generation AAA PC games. In general you should always know what target platforms you have to scale the polycount according to the given specifications. For example, most consoles won't be as powerful as a next-gen high end PC but if you target PCs, you also have to think about low spec PCs. So it's also a good idea to find some statistics on which hardware most of your customers have and take the average of all of them.

Modeling I – Lowpoly

By default, blender gives you a camera object, a light and a 2x2 units big cube. We will use the cube to start our modeling task and create a character with a low polygon count (hence lowpoly) which will be used in the game.

First we recommend you to change your current view to a one which is a bit simpler to use for modeling. By pressing 1 and then 5 on the num pad, you move into front view, orhtogonal mode. This is just a recommendation and of course you can use whichever view fits best for you. Now simply move the cube to a place where you'd like to start the modeling. If you work on a humanoid character, you could for example start with the characters chest. Next, split the cube in half by moving into edit mode, which can either be set in the drop down menu in the toolbar of the 3D View, which should read Object Mode at the beginning. Or by pressing the 'tab' key on your keyboard. In edit mode, hit 'ctrl' + 'R' and move your mouse cursor to the upper or lower line of the cube so that a pink line gets drawn vertically on the cube, then click on your left mouse button. Make sure to not move your mouse cursor or hit 'ctrl' to move the cut in incremental steps. To accept the position of the cut either hit the 'Enter' key or again the left mouse button.





With this cut, deselect all vertices by hit 'A' and use the box selection 'B' to select all the vertices of the left side of the cube and delete them 'X' or 'Delete'. Make sure to disable the selection limitation to visible to really select all the vertices on the left side, even the ones that may be hidden behind other vertices or faces. Now head over to the Modifiers tab within the object properties panel. There click on the Add Modifiers box and add “Mirror”. Check the Clipping checkbox and that's it, now we can start the real modeling.

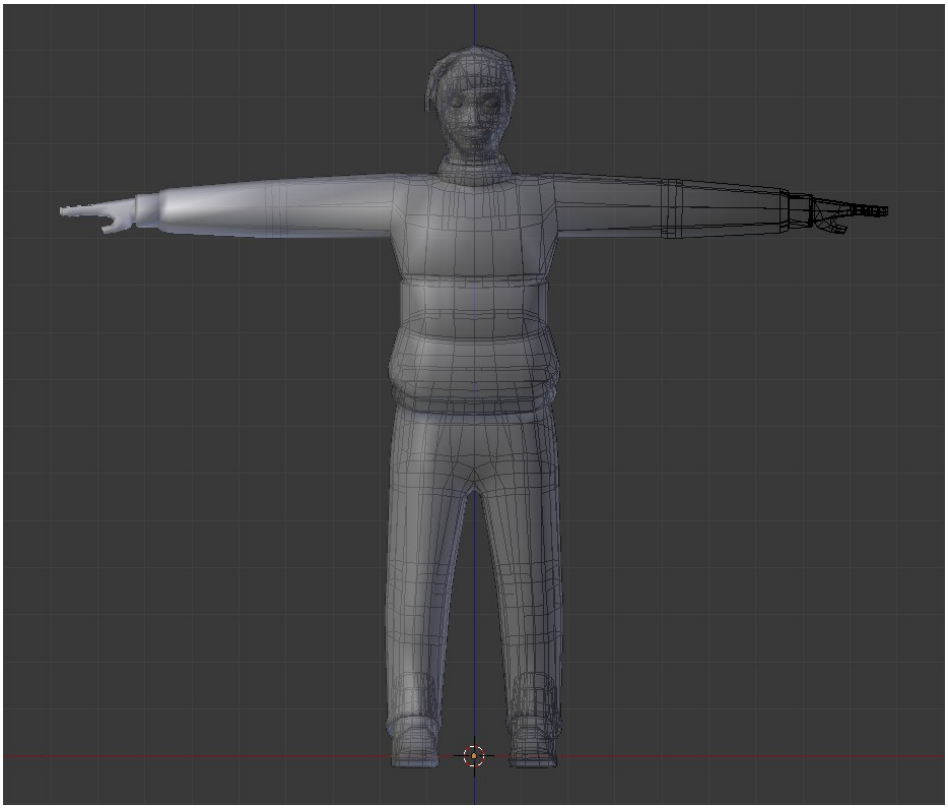
Usually you would now use the extrude 'E', split 'Ctrl' + 'R' and grab 'G' functions to mold your character from that cube. First you should model the outlines of your character as seen from the front, after that you can move into the side view and fix the depth of your character. Next you add little details and shape the character even more until it finally resembles the character you have in your head as good as possible.

Make sure to have a good topology of faces. For example, you should make sure all faces which consist of more than one triangle, like quads (faces with 4 vertices), are perfectly planar, otherwise better triangulate them 'Ctrl' + 'T' as otherwise the application has to predict if the face should be concave or convex which sometimes does and other times does not fit the expectations of the designer.

Also often move around your character in any direction to quickly spot pieces that need to be reshaped to look naturally and pleasant to the viewers eye.

You should also think about how the character should be able to move. The joints between two movable parts for example should get a few more polygons as they will stretch as soon as the two parts get moved. This could for example be the knees or shoulders. Generally it's a good idea to know the anatomy of your character to know how to model things best.

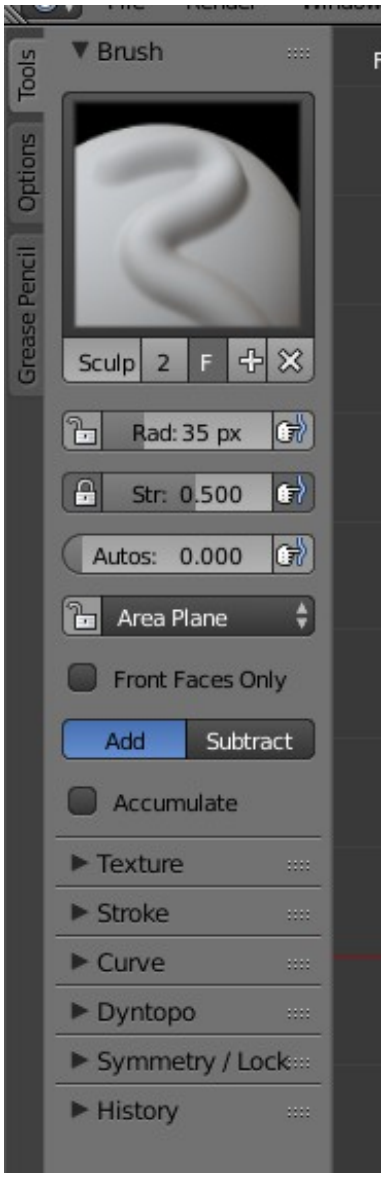
As a designer, it is beneficial to know which parts will be seen more closely and more often. Those parts should get higher details. For example the face of a character should mostly have quite a highpolygon count whereas the legs and feet may not need to be as detailed.



Modeling II – Highpoly [sculpting]

Having a lowpoly representation of your character is good and for some games it might even be enough, thinking about some toon characters or flat shaded ones, those might not even need a highpoly model. After all, the highpoly model won't be seen directly in the game, instead it is used later in the texture generation stage to bake texture images for Normal Maps, precalculated Ambient occlusion, shadows and the like.

Note, for working on highpoly sculpts, it is recommended to have special hardware for it like a haptic device that simulates a pen in 3D space or at least a graphics tablet.



Now, copy your lowpoly model and switch from “Object Mode” to “Sculpt Mode”.

← On the left side you will notice some new tools which can be used to change the brush for your sculpting. The default sculpt brush is SculptDraw and will kinda inflate the parts/vertices that you draw on. If you use the 'Ctr' key while drawing this will go in the opposite direction and deflates the parts under your pen. The third method that's available for your brush is smoothing which is activated as long as you hold the 'Shift' key.

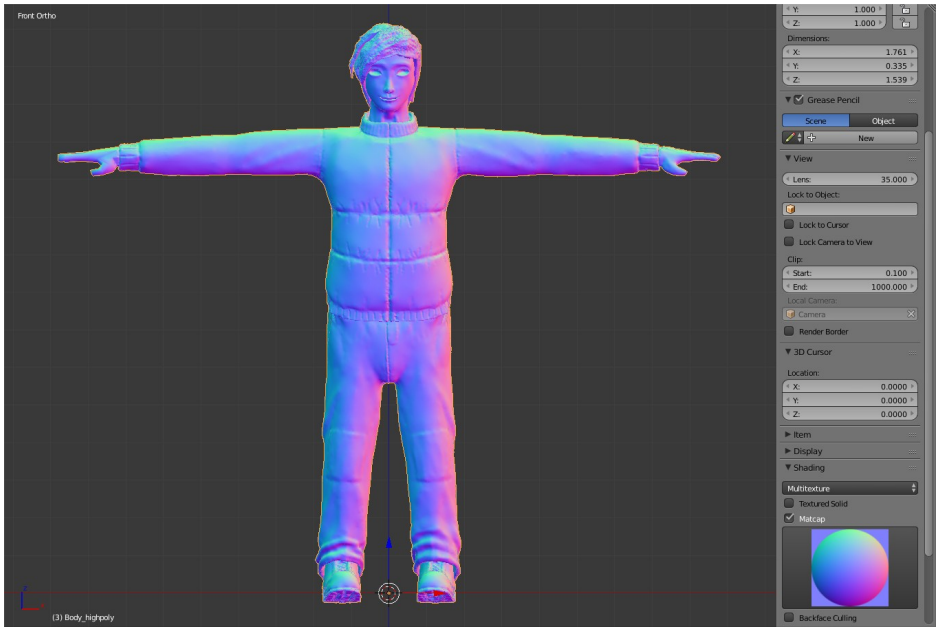
Over at the tools panel, you can also change the pens size, shape and strength as well as a bunch of other settings. If you not always want to move over to the tools panel, you may also want to remember the shortcuts for resizing the pen 'F' and rotating the pen 'Ctrl' + 'F', which is only useful for pens that aren't round for example if it has a texture applied.

As a little tip, you should enable the Matcap option from the Shading tab of the right panel of the 3D View. This will render the model with a given texture and lighting which greatly enhance the visual representation for sculpting. →

Next, as we have a lowpoly model but want a highpoly model in the end, enable the dynamic topology or Dyntopo for short from the sculpting tool panel. This will add and remove vertices as you sculpt. Note, by default, the closer you are to the model, the more vertices will be added and the finer the strokes will be that you make. If you are further away, the topology will become more rough.

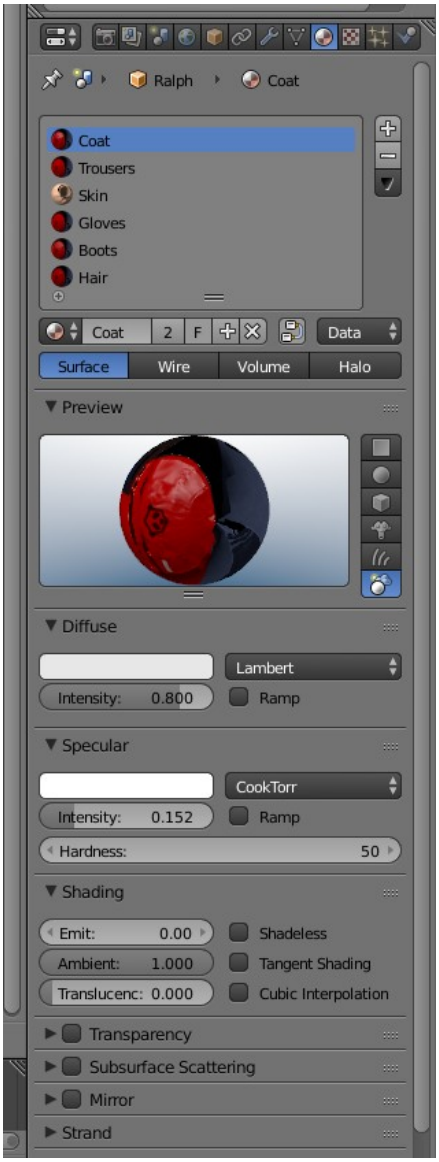
While sculpting, it sometimes is helpful to go into edit mode, select a bunch of vertices and hit 'H' to hide them. This will enable you to work more precisely on a specific part of the model. To show the vertices again hit 'Alt' + 'H'.

From here on it's up to you and your artistic skills to decide how detailed each part of the character will be, but the more finer details you have, the better will look the result on the final lowpoly model. After all, this model is only limited to the capabilities of your hardware as a sculpted model can have hundreds of thousand to millions of triangles.



Material

So far you should have a finished model for your character. It should already look quite good but if you export it, it will only be in gray scale, so let's get some color in there to make our character look even more unique and stand out.



Before we go over to texturing our character, we need to add materials and set them on the model. Materials will change the style of how the models surface will be rendered. For example materials will describe if a model is rough or smooth, shiny or dull and even if it receives shadows or not. Using materials, you can also set the base color of your model, so for example if you want to have a solid red or green model, you might not even need a single texture on it, though if you want to have fine details and not being forced to vertex colors, you definitely need textures.

So head over to the Materials tab (the small checkered ball) in the properties panel in blender. If not already existing, create a new material and give it a meaningful name. Then you can set the Diffuse color, Specular information as well as shading and transparency. Note, for transparency you better change to the blender game renderer over in the middle of the info bar in that small drop down box that should read “Blender Render” by default. Tough, we will talk about transparency in more detail in the asset creation section of this book.

Most of the properties you see in the Materials tab will be exported, but not all of them. For example changing the Shader Model won't have any effects as panda has it's own shader Model and won't distinguish between the selected ones. After all if specific Shaders are used in the application, the type of shader that gets used will most definitely be defined within the game Logic as it may change during the application. Therefor it's only important to correctly set the configurable values respective to those sections. Aside of that at present only the following things will be exported.

Diffuse Color, Diffuse Intensity, Specular Color, Specular Intensity and Hardness, Shadless and Transparency values.

You can use as many materials as you need for a model. Therefor you should assign the various materials to vertex groups, which can be done within the “Edit Mode”. Switch to that mode, select the vertices you want to have grouped and in the “Object Data” tab (the small triangle) in the properties panel, you can assign them to a new group by hitting the '+' in the “Vertex Group” section and then the “Assign” button that will pop up below the list, which will assign all selected vertices to that newly created group. Give that group a meaningful name and continue grouping vertices. Now you can select a group by selecting it in the list and hit the “Select” button below. This will add the vertices of that group to your current selection. Next, with the vertices selected, head back to the Materials tab and click the “Assign” button from the Material you want to be attached to that vertices. The Select button in the Materials tab will select all vertices that have the given Material assigned to them.

Having all that done, we now are ready to start adding textures to the model to add even more details to it's look.

Texturing

In this part we will describe how you can apply textures to your models to make them look in a style you want for your game. For example adding photos as textures will make models look like they would in real life whereas hand painted textures can make an object look like in a cartoon or other surreal worlds. They can also be used for very artistic looking models without needing to add a bunch of geometry.

Some special textures can also add the illusion of having more vertices then the model actually has. Those are for example normal maps. On the other hand there are textures for ambient occlusion and shadow maps which reduce the cost of calculating the shadows in the application, though the drawback on those is, that you will loose the ability of dynamic shadows as you already have static shadows applied to the model.

For the technical side, texture images in general should be images in a loose less format like png, tga or the like. They should have a size that resembles a power of two number like 16, 128, 256 or 1024 pixels. Though, you normally would not get over the size of 2048px except you target high end extremely detailed models. You can also mix and match different aspect ratios, means you can for example create textures which have sizes like 2048px width and just 32px high. Of course you can also use other non power of two sizes for your textures but then they will probably get resized to the next closest power of two value. How and if they are scaled depends on the applications settings as well as the graphic card that is in the machine the application is run on.

If you need very many details, you can also split up the textures in multiple files. For example you can have a Face.png a Body.png, Feet.png and Hands.png which then get mapped to the relevant body parts of your character. This way you won't need one large texture but have many small ones which get loaded much quicker and can more simply be edited by the texture artists.

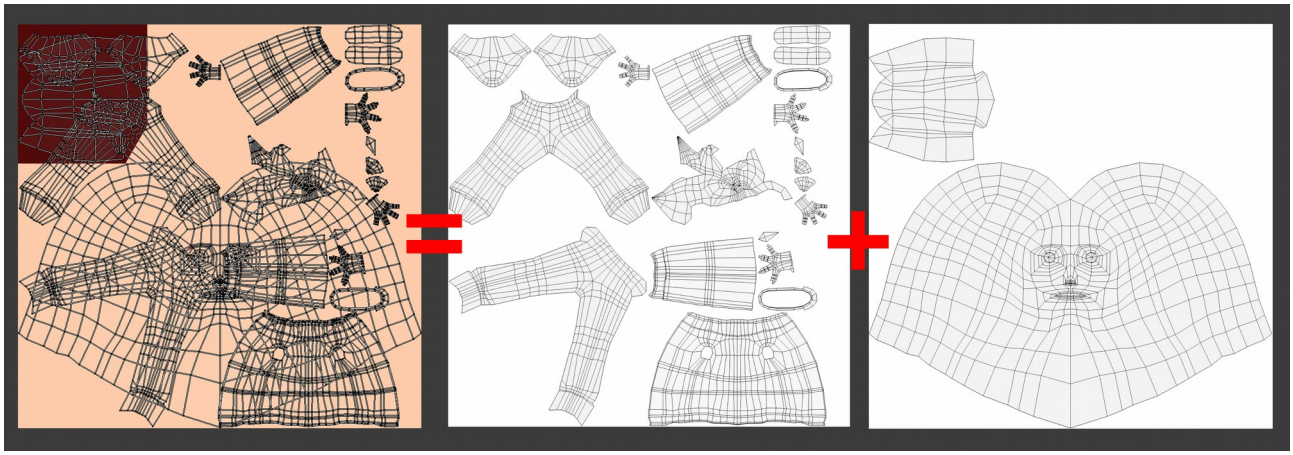
UV Map

Before you can correctly apply any textures to the model, you need to create a UV Map. This map tells the model which part of the 2D image and how they get mapped onto the 3D models surface. In the end it is like the sewing pattern for clothes, plush dolls and the like, which is also a good way to learn how to place seams that tells blender how to split up the UV map. For example if we have a character model, you may split it up in face, head, torso, arms, hands, legs and feet. This should give you UV maps that look like the sewing patterns for helmets, jackets, gloves, trousers and shoes.

To add and edit seams, move into “Edit Mode” and best use the “Edge select” mode. Now select a line of vertices that should be used as seam and hit 'Ctrl' + 'E' which will pop up the Edges menu. From this menu select “Mark Seam” which will add a red line next to the edge which is the visual representation of the seam. Continue to work on the seams until you have all seams placed.

Now select all vertices, or the vertices of a specific sub part of your model to create a UV map. You can create a as many UV maps as you like, though normally you would only have one map per model which may have various parts of the model mapped differently so that UVs may overlap tough with different textures assigned to the different parts which the UVs belong to, the end result still looks correct. To create the UV map, hit 'U' which pops up the “UV Mapping” menu. From that, select Unwrap and move over to the UV Editing Layout. There you should see the newly created UV map on the left side of the screen. If it looks good, you can use the “UVs” → “Export UV Layout” from the UVs Header to get a png file containing the UV layout. Make sure the size fits the texture size you target for that part of the model while exporting the UV layout.

If you're not happy with the UV layout, either tweak the UV directly or change the seams until they create a UV map that you like. Also take a look at the tools section to the left of the 3D View which gives you some tweaking tools for the Unwrapping. This for example contains the margin setting which will add some spacing between the Islands of the UV map, which may be very useful for the Texture artist and helps avoiding color bleeding within the UV Isles.



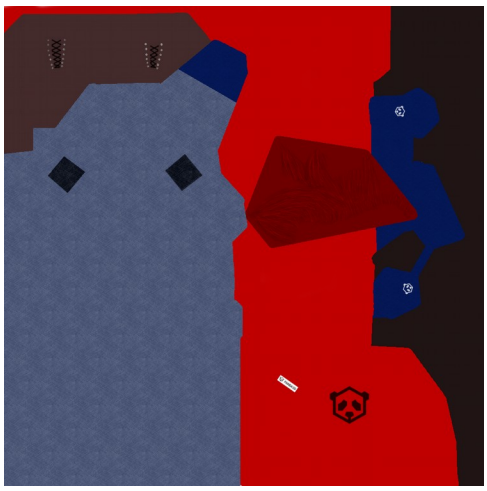
As soon as you have a good looking UV map, continue to the next part.

Diffuse Texture

A diffuse texture contains the colors of the character. This is the main texture that all models normally have.

As we should already have a UV map now, all we need to do is create a new image. This can be done from within the UV/Image Editor by hitting the “New” button on the header bar. Give the image a name and set it's width and height and if you like also the “Generated Type” to for example, UV or Color Grid which can help creating the textures later by making the isles more distinguishable. Finally accept by click the OK button. After that save the image by clicking “Image” → “Save As Image” or simply 'F3'.

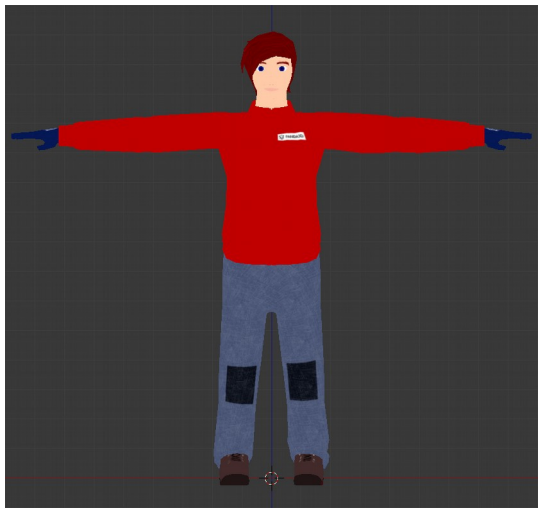
With the saved image, you can load it and the UV Map into your favorite graphics editing application to edit the texture. We would recommend to use one of the free software tools like the Gimp, Krita or MyPaint to edit your textures to fit the style of your game, no mater if it should be a photo realistic texture, flat cartoon colors or something stylized. Of course you can also use blender to edit your texture. A nice way to edit your texture in blender for example is the “Texture Paint” mode which can be selected from the Object interaction mode selection.



Using the texture paint mode you will be able to edit the texture with various brushes you can set from the tools panel. Some tools from the texture paint are quite similar to the ones from the sculpt mode.

If you edited the Texture externally and saved it, hit 'Alt' + 'R' to reload the image from disk.

Next head over to the “Texture“ tab in the properties panel. The starting cube already has a basic texture object attached to it, so either use that or remove it and create a new one with the '+' button below the list next to the textbox. Give it a good name and set its type to “Image or Movie”. A little further down, you can load or create a new texture image. As we already have one, hit the small icon left to the “New” button and select your diffuse color texture. Now head down to Mapping and set Coordinate to “UV” and enter the uv map in the upcoming field below the drop down box. That's it, now your texture should be properly exportable. These settings should be mostly the same as a base for any other of the following textures, so try to keep them in your mind.

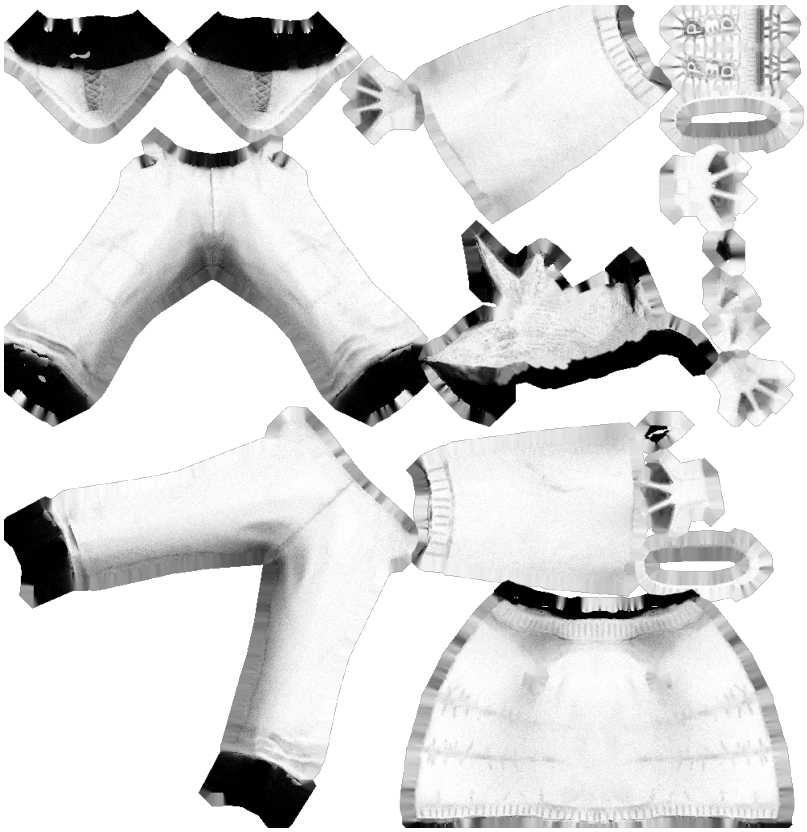


Ambient Occlusion

Ambient occlusion (AO for short) textures are used for precalculated shadows which are different to simple drop shadows. AO represents the darkness that occurs in edges only lit by ambient light and no direct light source. For a more detailed description on ambient occlusion read https://en.wikipedia.org/wiki/Ambient_occlusion which has also a more technical overview of that technique.

Basically you can create precalculated AO very simply within Blender. Therefor move to the Render tab (the small camera) of the properties panel and head down to the bottom where you will find the Bake section. Open it and select “Ambient Occlusion” from the Bake Mode. Now make sure you have your model, or if you have an additional high poligon model, both your low or if applicable your lowpoly chache model and highpolygon model selected. Whereby the low polygon one that has the UVMap needs to be the active one. Next go into edit mode (should show you the vertices of your lowpoly model as it is the active one) then head to the UV Editor, create a new image for the to be created AO and give it a meaningful name. Leave edit mode to get back to “Object Mode” and look at the properties panel set the remaining configurations for the bake, like Distance, Margin and Bias to get the best results for your model. The Normalized checkbox should be kept active except the application needs a non normalized version of the AO and most of the time Automatic split should give the best results but that again depends on your model. If you have a highpoly model selected, make sure the “Selected to Active“ checkbox is checked. Most of the time, the default values should be sufficiently set but you should play around with them until you find the best settings for your model.

Now finally hit the “Bake” button and watch your new AO texture getting created in the UV Image Editor.



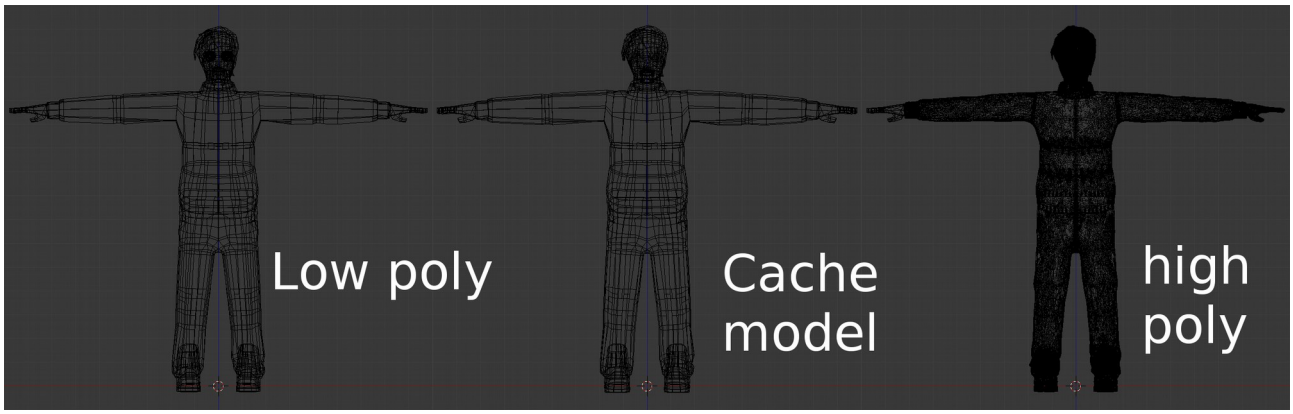
After the successful bake you should have the just created texture image filled with gray scale color representing the ambient occlusion. Save the image and continue.

As blenders AO rendering sometimes tend to be a bit noisy and you may get some black or to dark spots here and there, you may want to edit the image manually with an external image editing application like Gimp or Krita. A

Gaussian blur filter and some manual editing of to dark areas with a clone tool should help most of the time. If to much of the image is badly rendered and you haven't already used, you should definitely try baking with a model cache. Therefor copy the lowpoly mesh and fatten it 'Alt' + 'S' so it occupies most of or all of the highpoly model. Then select this cache model as the one to be baked to and do follow the steps as given above. Cache models should help removing to dark spots as well as most other baking errors.

Next over in the texture panel, add a new texture and set the newly created image to it. Scroll down to influence and uncheck the diffuse>color checkbox and tick the Shading>Ambient one. After you tweaked it's values correctly to your needs you're done and can continue to the next section.

Note, make sure the diffuse color texture is used for the model to have it correctly rendered in blender again, otherwise you might only see a gray scale model with the applied ambient occlusion.

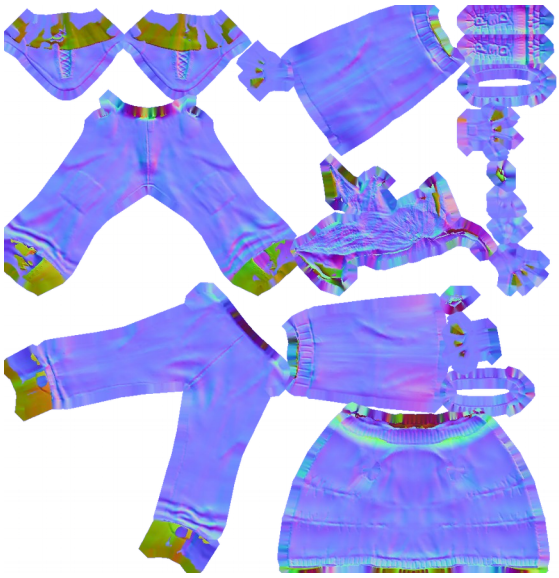


Normal Map

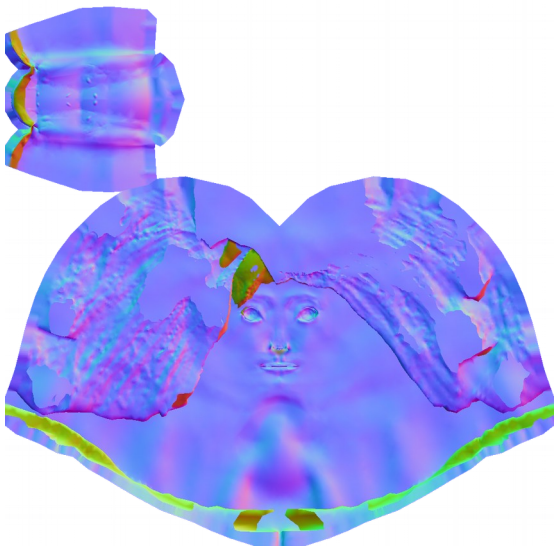
Normal maps are a bluish representation of heights and depths on your model. Using such textures will tell the renderer how the light gets reflected from a single triangle and hence how to color the various pixels. Those special textures are for example often used for stone walls, where you simply create a few big polygons with just a few details and have a realistic looking stone wall in the end with all it's little cracks and cavities not looking flat but just as where they real modeled.

Note, even though this may look good from the front and at some angles, you need to make sure to somehow cover any open edges as otherwise the player will notice that the surface is just a simple plane as this technique does not add actual geometry. For a real displacement of geometry you need to take a look at shaders which are capable of adding such geometry on the fly and are less expensive than modeling the actual geometry first hand.

To Bake the ambient occlusion just create a new image in the UV Editor just as described in the Ambient Occlusion section above and select Normals from the Bake Mode within the Bake section of the Render tab in the Properties panel. Also same as before with AO, set the values as necessary and hit “Selected to Active”, if you have the highpoly model selected in addition to the lowpoly one. Bake them, save the image and proceed to the textures tab.



Now back in the Texture tab in the properties panel, add a new texture, set the normal map image and uncheck the Diffuse>Color checkbox in Influence. For normal maps, you will check Geometry>Normal and set the strength of the normal map. Additional to that you need to check “Normal Map” within the “Image Sampling” section.



That's it, now your model has a normal map texture applied to it.

Note, if you don't have a high detailed model, there are also other tools, which can create normal maps for you. One for example is the normalmap plugin for gimp, which can either be installed from most linux repositories or by downloading it from its project page over at the gimp plugin registry: http://registry.gimp.org/node/69

Possible Baking Errors and how to solve them

Circular reference in texture stack

This error occurs if you have already attached the image you want to bake the normal map to on the material of the model. To solve this issue untick the checkbox next to the texture that has the normal map image set. After the bake was successful make sure to check the checkbox again. To prevent this error, use a cache model with a separate material and texture image (could be one that is generated by blender) applied that does not have the to be baked to image set as texture. This way you can also leave the to be baked to image active on the other models and hence you should always be able to bake a new version.

No objects or images found to bake to

This means that you have not set any texture within the UV Image editor to the model you want to bake to. To solve this issue, head to the UV editor and make sure that it contains an image when you are in the edit mode of the object you want to bake to.

Image has badly baked parts (dark spots, wrong normal representation, etc)

Those parts can most of the time be fixed by using a cache model created by copying and fatten the lowpoly model. If that doesn't fix these spots, either move the vertices around these spots or head over your most favorite image processing application and edit those parts by hand.

Rigging

Rigging describes the task of adding a skeleton to a model which then can be animated. During the task of rigging your model, you will create an armature object which is another word for the skeleton we create. A single part of an armature will be a bone.

Armature and Bones

At first we need to add the skeleton which is also called the armature. Make sure you are in the object mode then hit 'Space' and search for 'Add Armature'. This will add an armature with a first bone. Make sure your cursor is placed at the root of the character, which should be X, Y, Z = 0, 0, 0 while adding the armature.

You might notice that an armature only has three object interaction modes. The usual Object Mode, then the Edit Mode and the Pose Mode. In this section we will talk about the edit mode, the pose mode will be discussed in detail from the Weight Painting section on.

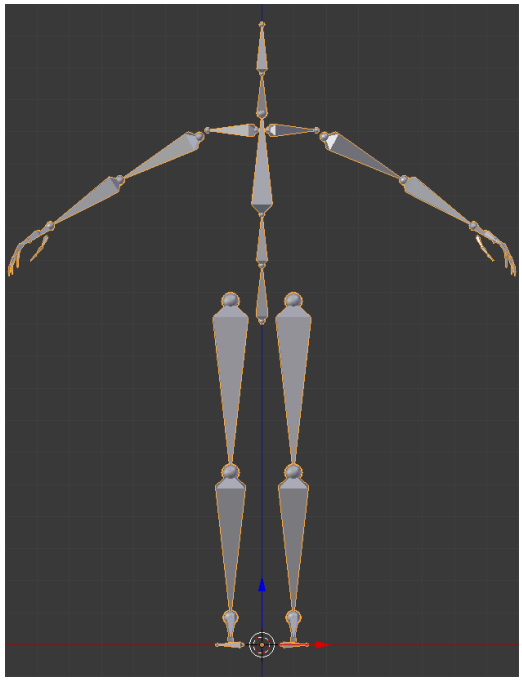
Let's start with editing the Skeleton. First move into edit mode. Now you can select the parts of a bone separately which are the head (the ball at the smaller end), the tail (the ball at the bigger end) and the main body part in between those balls. Now you can edit the bones just like you can edit a model, you can extrude 'E' and subdivide 'W' it as well as rotate 'R' and move 'G' it and do many other things with it.

If you're going to create a character you may also find the x-Axis Mirror functionality useful which can be activated from the Armatures options on the right panel, the tab below the tools tab. Now if you want to have a mirrored bone, press and hold 'Shift' and then hit 'E' to extrude a bone, now move it on the x-axis and see how you get two bones move simultaneously. Note, you only need to press and hold the shift key if you extrude from center bones, other already mirrored bones will automatically add bones on both sides of the armatures x-axis.

As soon as you're done with building the skeleton, you need to parent the object to the armature. Make sure your model gets parented to the armature not the other way round. To do this first select your model, then shift select the armature and press 'Ctrl' + 'P' which will pop up the “Set Parent To” menu. From there select “Armature Deform” or any option below it to get the model being transformed by the armatures bones. If you used one of the options under Armature Deform you might get some instant results when moving the bones in Pose Mode, but most of the time you want to get them tweaked a little. But more on that later in the Weight Painting section.

Note: If you don't want the bones to deform your meshes volume, you can hit the “Preserve Volume” checkbox in the Armature modifier set on your object that gets deformed by that armature.

Rigify



The task of creating a humanoid skeleton can be quit a huge and repetitive one, therefor some people have created an add-on named rigify. Rigify can create a completely set up humanoid like skeleton and add it to your scene. From that on all you have to do is parent your objects to the armature, shift and add extra bones to fit your model and fix the bone weight where necessary with the weight paint tool or by vertex selection.

To enable this add on move to File → User Preferences… → Add-ons and search for rigify. Check it's checkbox and close the preferences window. Now if you hit space, start typing Add Human then you should see “Add Human (metarig)”. When you select it, it will create a new armature at the cursor position named metarig as well as all bones correctly named to what they represent in a human body.

There also is an extended human rig called “Add Pitchipoy human (metarig)”. This rig has more bones, especially in the face region.

Weight Painting and vertex groups for Bone influences

After you have build up your skeleton and parented it to your model object, you can fix which vertices get affected by which bones.

Vertex Groups

A rather simple (if done right) and very accurate but also sometimes time consuming way to tell bones which vertices they affect is by using vertex groups. Therefor you need to create or edit (if parented using automatic weight or with empty groups) the vertex groups of your model. Hereby each group represent a single bone and hence must have the same name to be mapped correctly. With these groups, you can assign and remove vertices which then get affected by the bone with the same name. You can find the tools to add/remove vertices to groups as well as the groups themselves in the properies panel in the Data tab (the small wireframe triangle) under the Vertex Group section. To edit the groups, simply head into edit mode, select a group and then select the vertices that you want to add to or remove from that group. Now below the list of the groups you will find a few buttons and a slider for Weight. The buttons should be quite self explanatory and the wight value defines how much weight the vertices will get applied to. The weight can be seen within the “Weight Paint” mode, which is another and maybe more simpler and at times faster way to edit weights.

Weight Painting

To set the bone weight of vertices, first select the armature and move into pose mode. Then select the the model you parented to the armature and switch to “Weight Paint” mode.

Now if you select a bone of the armature with a right mouse click, you will see the affected vertices colored in red. The more red they are the more they get affected by that selected bone. Basically this mode works just like the texture paint and sculpt mode, you have your brush and various settings to change its size and shape. The values given to that brush like strength affect how much weight gets added or removed from the vertices you paint on. Aside of that, you have some specific Weight Tools that can for example normalize and mirror the weight.

You can also assign wight on vertices for more than one bone, this way both bones will affect the vertex, for example if bone one affects it by 25% and the other by 75% the vertex will move more towards the second bone. If you have it set to 100% on both bones, the vertex will be in the middle of both bones.

Most of the weight should already be set OK if you used “With Automatic Weights” when parenting your model to the armature, hence you should check by moving the pones around and see if they move the model correctly, otherwise fix the weight until it fits to what you expect.

Bone Constraints

Now you should have a basic simple rig attached to the character which will also be able to move it, but if you're heading for animation and need to do a bunch of different animations for your character you soon will find it cumbersome to edit every bone by hand. To help you with that and let a few bones move together in relation to other bones, blender gives you the bone constraints tool. In it you can set things like inverse kinematics, copy locations and transformations as well as limit those and many other things that help you to build up a rig that can simply be controlled by only a few master bones.

To add bone constraints, you need to be in “Pose Mode” of the armature. With that, you can see a little bone and chain symbol in the properties panel. That is the tab for Bone Constraints. Bone constraints do work just like other object modifiers, you select one from the drop down box to add it to the selected bone and set it's values. As there are quite a few constraints you can set and a huge amount of possibilities to chain bones together, we will just give a quick overview of some important constraints.

Copy and Limit Constraints

The copy and limit bone constraints do what you might already predict by their names, they copy the targets position, rotation, scale and transformation or limit the bone to those values of the selected other target. Some of them might not be able to have a target set, like the Limit Scale, those constraints can have the values set directly. As the copy and limiters are rather simple to use we won't go into much more detail with them, it's also rather simple and intuitive to toy around with

them, so try and find what suits best to your rigs and animators needs as each rig can be set up differently depending on the desired animatability and the taste of your animators. For a sample of how rig constraints can be used and grouped together, take a look at the Ralph blend file that comes with this book. It's rig was build mostly by following the following tutorial videos for riging a humanoid body: http://www.blenderguru.com/tutorials/introduction-to-rigging/

Track To Constraints

Similar to the copy and limit is the Track To constraints. This constraint lets a bone always look at its target and are useful for many things. One for example is if you want to have a keyframe animated camera node that you can use in the application to parent the camera to, to simplify a level fly through animation or a cinematic camera setup in some sequences of the game that should be rendered directly in the game. Another example would be, and this way it's also used in the Ralph example, as bones for the eyes which tracks another bone setup that is used to control both eyes at the same time.

Inverse Kinematics

Aside of the copy and limit constraints, another very important and often used constraint is the Inverse Kinematic or IK for short. Take a look at https://en.wikipedia.org/wiki/Inverse_kinematics for a detailed description of what Inverse Kinematics mean and do. Basically speaking, IKs are used to tell a bone how to behave when moved, for example an IK can be used for the shin bone that the knee don't bend in the wrong direction. IKs are also used in the Ralph model at the shin bone, the upper arm and forearm. You may also see extra bones that have IK in their names. Those are control bones for bones that have IK and other constraints set which needs a target bone and which best have a separate bone to work properly and not one of the “main” bones of the armature which are used to move the geometry, so those IK bones only move other bones but no vertices by themselves.

Custom Bone Shapes

Most of the time, if you create the bones of an armature, they will get hidden behind the actual model in the end unless you activate the x-ray option of the armature. Though, for animation tasks, seeing all the bones all the time won't be the best solution and will hinder the animator in it's work. To not need all the bones be visible all the time, you can create models which can be used as custom shapes for models. Those shapes can be rendered as wireframe and made big enough with some padding so they actually are bigger than the to be animated model and hence will be seen but don't get in the way of the animator, better than that, if set up correctly, they will help him on its task to animate by exposing the most important bones and visually represent what they will control.

To set custom shapes, you first need to create simply shaped objects. Those can be rings, spheres and other objects for example but also shapes that resemble other parts like a curved egg shaped ring to be used for the head controlling bone or a flat footprint/sole shaped model placed below the characters feet to control its feet and leg with it. Note, all these shapes should be placed in the

center of the 3D space to get correctly placed on the bones.

If you have your shapes done, you can simply assign them by selecting a bone in Pose Mode and move over to the Bone tab in the Properties panel. There within the Display section you can now set a “Custom Shape”. If you set your desired object there, your bone should be transformed to look like the modeled object.

Now, if you use custom shapes, you can also set the armatures displaying mode to Wire so the remaining bones won't get in your way when you going to animate the model. This can be done in the Display section of the Armature Tab of the Properties panel. There you can also toggle if the bones should be displayed with shapes or without.

As soon as you finished adding custom bone shapes to your rig, you should be finished by now and ready to animate.

Animation

Now that we have a fully rigged character, we can start with animating it. The simplest way to animate is by putting all animations in one timeline by simply adding keyframes where the character has a special pose. Aside of having just one timeline for all animations, you can also use the action editor to have a timeline for each animation.

We will start with the basics of Keyframe animations first and then describe how to use the action editor to separate all your animations in their own timelines.

Keyframe Animation

A Keyframe describes a specific point in time of an animation. Normally at those keyframes the animated object is in the most extreme position of that given movement. For example in a walk cycle this would be the legs all spread out widely in the first keyframe and in the second one foot on the ground in the center and the other one angled moving forward.

To add and edit keyframes, you need to do the following.

At first you need to select the armature you want to animate and change to “Pose Mode”. Make sure that the green line in the timeline (or the number right to the “Start:” and “End:” boxes) is set to the correct frame where you want to add or edit a keyframe. Before you now start moving the bones of your armature, you should know that there are two ways for adding key frames. One is adding the frames manually by hitting th 'I' key and select the things you want to store in that frame and the other is to set the frames automatically as soon as you move around the bones. To enable automatic keyframing, press the red dot button in the info bar of the timeline. Using automatic keyframing will speed up creating animations, but you also need to take care of what you do to not accidentally destroy a previously posed frame or add keyframes at frame numbers where you didn't actually wanted any keyframes.

Let's take a look at timing your animation. This is a really important but also rather mathematical – thankfully rather simple math – point as you need to calculate how fast an object is moving over time for each keyframe you add. This is especially important for things like walkcycles where the character needs to walk as fast as he gets moved forward. If the timing is not correct in such animations, the character may seems to slide over the ground, so always make sure to talk to the designers and developers to ensure your animation is at the same speed as the speed of the object in the application in the end.

To calculate the speed, you need to know a few things.

1. How fast should the object move in the desired direction (in units per second whereby the units can be anything, for example they could be metric as well as blenders default measurement which

should map 1:1 to Panda3Ds measurement)

2. How fast is your animation in frames per second (FPS)

With those two values you can calculate how far the object needs to be moved for each keyframe to get the desired animation that fits the application it will be used in.

Now this is the formula you can use to calculate:

$$\frac{movementspeed/second}{Frames/second} = \frac{movementspeed}{Frames} = move\,distance\,per\,Frame$$

And here is an example filled with numbers that could be used in a walk cycle for the placement of the feet. If we assume, the character walks 1m in 1 second and 1 second has 24 frames, the following can be applied:

$$\frac{1\,m/1}{24/1} = \frac{1\,m}{24} = 4{,}2\,cm/Frame$$

Now we know that the character needs to move 4.2cm each frame. If we want to use this to place keyframes, we can simply multiply that value by the number of frames that are between the keyframes and hence know how much we must move the feet away or towards each other. Say at the furthest distance when the legs are spread out, the distance between them is 1m and you want to have 5 keyframes. You can use 4.2 x 5 to get the distance of 21cm. If you move the characters feet by that amount, you will make him walking at the correct speed.

Aside of this formula, there are a few tools in blender that can help you with animating your model. The first one which is to simply add a background video which you just need to fit to your animations frames and then trace it by going through the video search for a few good key points and add your keyframes there. After that go through your animation from start to end and check if it fits to the movement on the video as well as fit to the other needs for that animation, which could for example be that it's loopable or respects the just described speed calculation. Videos can be added just like background images which we used in the modeling chapter before, just change from Image to Movie, uncheck the “Camera Clip” option and then instead of loading an image, simply load a video clip.

Action Editor

Timeline and Keyframe editing is simple and does work nicely for one or just a few animations in one timeline but if you have more than a few animations or like to keep your timeline clean and organized, you definitely like to use the “Action Editor”.

To use the Action Editor, best move to the “Animation” Layout and in the “Dope Sheet” (Top left window) change the “Editing Context” from “Dope Sheet” to “Action Editor”. Now make sure you have your armature selected and preferably are in Pose Mode, then hit the new button next to the

magnifying glass in the header of the Action Editor. This will add a new animation for that Armature. Give the animation a name and start your keyframing. You will see the keyframes as yellow dots in the Action Editor whereby each line there represents a bone of your armature. If you have set a keyframe but don't see a dot on some of the lines, that means there is no data saved to remember for the Action Editor at that specific keyframe. In the window below the Dope Sheet, you can see the Graph Editor. There you can see a curved representation of your bones movements each line represents another bone and each color a separate axis of that bone. You can always unfold elements on the left of both of those Editors to see finer details of specific bones.

Next, as soon as you're done with editing the keyframes of your current selected animation, hit the button that reads 'F', which is located next to the name of your animation name in the Action Editor. This will ensure that the Animation has at least one fake user even if it is not applied to the armature. This is important as otherwise the action won't have any users and will get deleted the next time blender is started. As a side note, each object (may it be textures, actions, materials or anything like that) that has a 0 next to his name will get deleted when blender gets closed. So make sure all of these which you like to preserve have a real or at least fake (that 'F' button) user which they are applied to.

With your finished action, simply hit the '+' button next to the action name to create a new one. Using the '+' will also copy all the keyframes of that current action. You can either use them as base for your new animation or simply hit 'A' in the action editor to select all the keyframes (they should be marked yellow) and then hit 'Del' to remove them, which should give you a fresh looking Action Editor and timeline with no Keyframes set. You may need to reset your bones transformations which can be done with 'Alt' + 'G' to reset the bones positions, 'Alt' + 'R' to reset the bones rotation and 'Alt' + 'S' to reset the bones scale. Those shortcuts also work for any other objects in your scene, not just bones.

After creating a few animations, you can now jump between them by selecting the desired action via the “Browse action to be linked” button left to the actions name field. Just make sure you have your armature selected as otherwise the action will get applied to the current selected object and hence get a new user. If you accidentally attached an action to anything else than the armature and want to get rid of it again but can't find the object you attached it to, head over to the “Outliner” panel and enter the actions name in the search field. This will list all objects that have that specific action applied to. This way you can also simply select the object and hit the 'X' button next to the action name in the Action Editor panel.

Additional Notes

In here, you will find a few notes and hints on how to achieve a variety of things for your character.

Attaching objects like weapons and clothes at runtime

To make your character able to hold weapons or wear different clothes, you need to add an empty node or bone at the position where object should be attached to. In Panda3D this works quite similar to the parenting in blender, so you should be able to test your nodes placement and rotation quite simple.

For example, if you want your character to be able to hold an object in his right hand, you best add an empty node or bone within the palm of his right hand (Bones sure will be simpler to animate and control). Then name give that node or bone a name and if you chose it to be a node, make sure it gets moved so it follows the rest of the characters animation.

Now, as these nodes or bones need to be accessed by the developers, you should maintain a list of those and hand it over to the developers so they know what and how to access to attach other objects.

Also as designer, make sure the objects that will be able to get attached to the character fit in the spots where they should be attached and make sure if for example the character holds it in his hand, that the fingers are closed around the object so it really does look the character is holding the object.

Chapter 3: Asset creation

This section will be a pre section to the level creation section and talk about creating little parts that can be used within a level blend file. Creating assets is in most parts quite similar to character creation, just with the little difference that most assets will be static and don't have any animations applied to them even though they could have some. So basically speaking, even characters are assets but in this special case, we'll use the word asset for models that we want to reuse in our to be created level.

Modeling

Modeling assets is really the same as creating a character. You'll model a high or low counting polygon object that represents the object and then create the respective other one using the first created. Then apply some textures to it and bake the special textures like normal maps, ambient occlusion and whatever else you might need for your object to look good.

Now as the modeling is the same, here are a few tips and considerations you could take into account when creating assets.

1. Assets normally don't need to have a polygon count as high as the main character in most games as most users won't look that long and closely at the assets thrown into a level, they still must look good though and you still need to decide by your own which models need to get a higher and which a lower detail level.

2. The further away an object will be, the simpler an object can be modeled. For example if we take some far away sky scrapers, they can be modeled by a simple block and some nice textures. If you have skyscrapers very close to you, you need to model finer details like doors and window frames and whatever else won't look good as just a close up texture.

3. Pack as many meshes in one object as possible this will enhance performance more than if you have many objects and meshes separated.

4. Always give your objects a meaningful name.

5. Group meshes and objects that will be used as one object in another blend file. Groups will be used to put objects in other blend files like levels. More on that later in the next Chapter, section “Linking”

Collisions

Talking about creating collision solids as well as using object types to simplify the collision creation and make them more solid.

Self created collision solids

The first and most accurate (regarding the models shape) way but also maybe not the best way for solid collisions (regarding collision detection accuracy) is by creating a mesh shape that resembles a very low polygon version of the object. The less polygon the collision mesh has, the better it will be for the games performance. Of course, if your object already has a very low polygon count, you could also use this mesh directly for collision detection.

Whichever way you will take, to set an object to be used as collision solid, you need to add a game property to it so that the exporter later on will know that this object needs to be set as collision mesh.

So with the collision object selected, head over to the “Game Logic” layout and “Logic Editor” add a new property. Set it's type from Float to String and add the following as the properties name:

```
Collision
```

For the value, use one of the following lines:

```
Polyset descend
```

which is for objects that only represent collision solids but should not be shown within the game.

```
Polyset keep descend
```

See the “keep”, this will tell the engine to keep the actual geometry visible but also set it as a collision object.

There also are some more options and types that you can use for your collision solid like Plane, Sphere for it's shape or level to let all the collisions normals face upward or intangible to detect collisions but not prevent things from passing through it which is for example useful for implementing triggers which will also lead us to the next section.

For a complete list of these options, take a look at
http://www.panda3d.org/manual/index.php/Egg_Syntax#Collide

Pre-built Object-Types

Aside of creating specific polygons for collision solids and set them to a specific behavior, you can also use some predefined object types. There are for example types like barrier which will set up a simple collision setup just like the collide : Polyset descend setup as well as trigger which incorporates the intangible option and are basically just a nice way to shorten and simplify the type work you have to do when tagging objects. These object types are defined within the Confauto.prc file which comes shipped with the Panda3D SDK and can also be enhanced in there with your own setups. Just make sure to populate your changes to other team members so that these types will be recognized by everyone using them in the engine.

For a small list of available object types and those who have the SDK not installed, take a look at http://www.panda3d.org/manual/index.php/Egg_Syntax#ObjectType which lists a few but not all by default available types.

As you might have seen, in the Game Logic panel you can add properties which get exported to the resulting egg file. There are quite a few properties that you can set in an egg file which will have an immediate result. For a list of possible properties take a look at the egg file documentation over at http://www.panda3d.org/manual/index.php/Egg_Syntax or for the complete description https://raw.githubusercontent.com/panda3d/panda3d/master/panda/src/doc/eggSyntax.txt

You can also add custom properties which get added to the egg file as a simple tag with the name and value set like <Tag> property_name { property_value } and can then be read from within the application the egg file is used in.

Texturing

Texturing the object, is the same as for character creation so we will just talk about some texture types that we haven't talked about before in the character section.

Creating Tiling Textures

Most of the time, if you build big models which needs rather detailed textures, you want to use textures that can be put next to each other without creating seams.

This way you can use your texture at a lower resolution and filling up more space than it would actually do at it's current size. To create tillable textures here are some tips which describes how to transform any kind of texture image (even photographs) into a tiling image with just a bit effort. These tips describe the necessary tools from gimp but most modern graphic editing programs should have similar working functions so it shouldn't be hard to map this tips to your favorite editing application.

The most simple and fastest way would be to use gimp's own tilling filter, though this might not always get you the best looking results.

For a better result, follow these simple steps.

1. Paint your texture or load your picture into your editing application. If you paint you should only paint the background layer and keep your details for a later step.
2. Use the Offset function to shift your active layer (in the gimp this can be found at Layer → Transform → Offset)
3. Click the “Offset by x/2, y/2” button or enter half the image size for the X/Y coordinates. For a 1024x1024 image for example this would be 512x512.
4. Make sure the Edge behavior is set to wrap around and hit OK.
5. Now you have the seams from the edges like a + cutting the center of your. From there on you can use the clone tool and the like to remove those seams.
6. After that, repeat the layer offset and check if the texture is already tiling, otherwise repeat with removing the seams until all seams are gone and the image looks good to you.

7. If you use a hand painted texture, you should now add your details but make sure they do not touch the edges of the image. Then again use the offset tool and continue on your details.

There's also a way to simply create hand painted tiling textures within blender and it's image editor. Take a look at the tutorials section at the end of this book where we have linked a video to one showing how to do that.

Emit/Glow Texture

An emit or as in Panda3D named a glow texture is used to describe which areas and how much those areas of an object will be glowing. Note to see the effect, the application that uses the glow map needs to have a special glow shader, which also comes with Panda3D. As an example of such textures, you can take a look at the glow-filter example that ships with the Panda3D SDK.

A glow map contains a gray scale representation of the glowing areas, where lighter more white pixels will glow stronger and darker pixels will glow less.

Specular Map

A specular map controls the specular intensity and specular color of a model. Normally if you have a model, the specularity set on the materials tab is used over the complete model, with this map now you can for example set a specific area of that model to have no specularity at all. This can be useful for things like medieval armory where the iron plates will have a high specularity but the leather belts holding it together only have a very low if any specularity.

A specular map contains a gray scale or colored representation of the shiny areas, where lighter pixels will have a brighter, shinier specular and darker pixels will shine less. Note, the look of the brightest specular will still be defined via the materials settings.

Specular maps can be created using an graphic editing application. One way of creating those would be the following:

Gloss Map

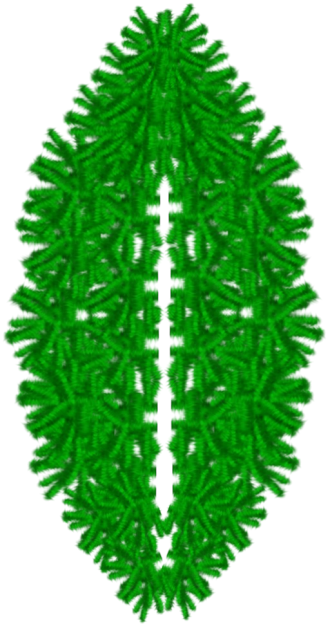
The last texture map that we will going to show is the gloss map. This map is used to determine the sharpness of a models specular and is quite similar to the specular map. The difference to a specular map is, that the gloss map does not define colors.

To create a simple gloss map, you could use the following steps in your favorite graphic editing application:

1. Select all glossy sections of your diffuse texture
2. Invert your selection and fill the selected areas with black
3. Unselect everything
4. Use a desaturate filter to change your colored image into a gray scale one
5. Optionally play around with brightness and contrast to get a better result

That's it, now you have a very simple gloss map.

Transparent Objects



To make object transparent, like glass, light rays or water, you'll need to have a texture with transparent parts and tweak a few settings in Texture and Materials tab of the objects properties panel. To enable Alpha on a texture, you need to activate the Alpha checkbox in the textures influence section or if the complete material should be transparent also in the Materials tab, check “Transparency” and make sure it's “Z Transparency”.

We also need to set a few properties in the Game Logic panel so the alpha will be correctly handled by the engines renderer.

Here is how to do it in a way that should work for most things like godrays, windows and the like.

1. Set the Transparent Texture on your model.
2. Enable Alpha in the Textures Influence section
3. In the game logic editor, properties are set for:
   blend → ADD
   blendop-a → INCOMING-ALPHA
   blendop-b → ONE
   bin → fixed

Of course you can also change the properties values to something different if you need another behavior. To get a list of possible options, take a look at the egg syntax documentation at:

https://raw.githubusercontent.com/panda3d/panda3d/master/panda/src/doc/eggSyntax.txt

In there search for the properties name and take a look at it's description which should list the possible values.

Advanced 1: Object types 'n' tags and egg files

We already discussed object types earlier, but here's a little more information about them which might be useful to know. As you should know now, the object types will be added as properties in the Game Logic panel and get exported to the resulting egg file.

Now, for a list of possible properties, more exhausting than the manual, take a look at the egg file documentation over at:
https://raw.githubusercontent.com/panda3d/panda3d/master/panda/src/doc/eggSyntax.txt

This documentation also describes what an egg file is possible to hold and what not, so it may be useful to know and as reference if you going to search if a specific setup in blender could possibly be exported and if it's the fault of the exporter or of the egg file if it's not exported correctly.

As a side not, you can also add custom properties which get added to the egg file as a simple tag with the name and value set looking like

```
<Tag> property_name { property_value }
```

These can then simply be read from within the application the egg file is used in and can hence be used for anything you want, even controlling the game logic if correctly implemented in the application.

Chapter 4: Level Creation

Now it's time to get all the assets together into one big level.

This Chapter will describe how to create landscapes and indoor scenes and how to get your previously created assets in there as well as what you need to take care of when creating levels regarding collision solids and some hints for texturing and general level design.

Modeling

This section will talk about modeling the two main level variations, outdoor/landscape levels and indoor scenes.

You will learn the basics of how to get good results quite quickly and what to take care of to get them into your game.

Landscapes

Modeling simple outdoor landscapes is quite an easy task, which can be done using a subdivided plane and the sculpt mode, which will work like a real time interactive height map painting. Just make sure you lock the X and Y axis when using sculpt to make unwrapping the landscape very simple later on.

Alternatively to that you can also use a real heightmap image or create a random one directly within blender. To use a heightmap image, create a plane that is big enough to represent the X-Y bounding of your level. Then you need to get the heightmap image to deform your ground, so you first need to load it as texture. Next we need to enhance the vertex count of that plane which we can either do by subdividing it in the Edit Mode 'W' → 'Subdivide' → set the Number of Cuts in the Subdivision Tools or add the Subdivision Survace modifier. Note, the higher the subdivision, the finer and more accurate will be the details in the end. Now to make this image deform the plane, head over to the modifiers tab if you're not already there and add Displace. In the Displace modifier options load the texture containing the heightmap. Now you can play with the Strength and Midlevel values until you're happy with the look of the ground.

Note as you will use a plane model in both cases and don't deform vertices in their X and Y axis the level will look like a grid plane from the top in wireframe mode. This is also desirable as it will greatly simplify the unwrapping and texturing of the level as this plane will map perfectly to the textures we apply to it later.

LOD

LOD is short for Level Of Detail and stands for having multiple resolutions of a single model which can be changed within an application dependent on the view distance of the player, means objects far away will get rendered with lower resolution, less polygons than objects that are closer to the player.

This technique can not only be used for level files, but for any model in your game which means it can also be used for the assets in your level as well as for characters and whatever else you might have in your application.

To get multiple resolutions of for example your ground model, you can simply use some modifiers

of blender that reduces the poly count, which can be for example the Decimate or the Remesh modifier. You just need to make sure the resulting model still looks good from a far distance, it doesn't need to look good close to the camera. You should make sure which distance will or can be set in the application so you know which distances you should check your models against and how many models you might need to create.

As for the model, dependent on the setup of the application, you can put any LOD object in one model file or split them apart. This is all up to you and the developers team what they prefer.

Indoor scenes

When working on indoor levels, you may work in a different style than when working on outdoor, mostly height map like levels. For indoor levels, you will first load a few images that may look like a blueprint with rooms, floors and other building elements layered out. Next you will work from that blueprint and build up the walls and floors, either by using pre-built wall and ground elements or by simply extruding the default cube that is there when opening blender. If you're using the cube, just make sure to have the normals set correct as you most definitely want to have them facing inwards if you want the cube already represent a first room.

Dependent on your gameplay, you may or may not add a roof to the level. Some games also simply had the walls fade to black on the top or bottom, basically wherever the level had no actual walls and then just set the background color within the application to black or which ever color you used to fade the walls.

Level general

This section contains some generally useful hints and notes that you could use to incorporate in your levels.

Starting Points and other positions

Most levels need to have one or more points set, so that the application knows where to place the characters at the start. This Points can be simple empty nodes, just make sure to name them correct. Such nodes can also be used in the application to know where to place collectibles or any other things that need to be dynamically added in the levels at runtime.

Collisions

As you should already know quite a lot about collision solids if you read the previous sections, we will just give you some more details about collision solids for bigger levels and outdoor scenes.

Here we will start with indoor levels as they may be simpler to set up. Basically you will build an indoor levels collision mesh just like you build your levels visible geometry, with the only difference, that the collision geometry will be much simpler that the visible one. If you have used linked objects like walls and grounds to build up levels and those linked objects already have a correct collision setup, you may not need to do anything at all for getting your levels collision done.

Same as with the LOD, you could also split up your levels collision solid into separate objects. This way the application can load only the solids that are close to the player model and unload all other solids which are further away which will, if there are many (100+) collision solids, greatly enhance the performance. For that, best split up your level into equal sized squares and double check that the seams of those squares line up perfectly as otherwise it may result in strange collision errors when moving between those level squares.

The floor and dupefloor object types

Panda3D has a special type that you can set for your level collision solids, the so called floor and dupefloor object type.

floor: This type will create invisible collision polygons with the polygons normals always facing up and hence preventing the player from sliding down the levels ground no mater how step it is. Tough, keep in mind, this may also let the player climb walls which he should not be able to climb if the player isn't set up correctly or just in a very simple way.

dupefloor: Using this type is the same as the floor type, just with the difference that it will duplicate the mesh for a visual representation. This should only be used if the mesh's topology is really simple or otherwise prepared for collision handling to get a good performance in the game.

Note, if you're going to use the Collide tag, you can also use the “Level” option to achieve the same effect.

Texturing

Texturing a landscape is kinda the same as texturing characters or assets or anything else you might have, you need to create a UV map, create Diffuse images as well as ambient occlusion, shadow maps and whatever else you might like to have on your level. So as you should already know how to do all that from the other sections above, we will skip right to something that may help you creating nice ground textures.

As if you have a model created with a heightmap, unwrapping it should be really simple, normally a simple use of unwrap from the Unwrap menu 'U' should be sufficient to get a good UV map.

Texture Splatting

Texture splatting is a function that let's you draw with a texture on top of another texture. This way for example, you can create a single texture image that contains various terrains very simple even with textures that have a lower resolution than the actual out coming image. As an example for this technique, you can draw a road through a grassy landscape.

Actually it use the “Texture Paint” mode, so the difference is that you previously load some textures and then use them to paint on your model. You don't need to attach these textures to the model, just the final texture image, the one that you paint on needs to be set on it. So these are the steps that you need to do.

1. Unwrap the model
2. Create a new image
3. Set the Image as Texture on the model
4. Change into Texture Paint modeling
    1. In the Brush's Tools, open the Texture tab and select the texture you want to use as brush

Note, you can also use multiple texture layers which you can later bake into one single image, that way you will have greater control of you individual texture parts like setting blend types but you always need to keep in mind to switch between the layers which you paint on which sure won't be a problem if you're used to it from other graphic editing applications. Just make sure each layer is transparent by default so you will see the layer below it if the blend type is set to Add or Mix.

Now you're ready to draw your levels texture. You can use the Brush Mapping to change how the texture gets painted on your model. You should switch between them and see what fits best for which of your situations. Tiled and 3D may be the most useful ones for most of the time though. Some of those mappings will have extra values that can be set, like Angle which will rotate the brush by the given degree.

After you finished drawing the texture, head over to the UV Image Editor and save the created image in a file of your choice so it won't get lost when you quit blender.

Stencil Maps

Similar to the approach above is the use of stencils. Stencil images hereby are gray scale images which tells the application where other textures should “shine through”. So you actually load a bunch of textures and stack them over each other and with the stencils you tell where one texture and where the other texture should be shown by drawing on the stencil map. This way you can also use tiling textures for the model but still have a unique not tiling look on the whole model.

To achieve this in blender create 3 textures. The first texture will be the main texture that will be shown on the black parts of the stencil map. The second texture is the stencil map. This can for example be a 1024 or 512 square image (or any other size you need), dependent on the detail level you want on your object. The third image will be the one that gets shown on the white parts of the stencil map.

For the first and third texture just set them up as usual diffuse textures with an image of any other texture you want. As a side note, if you want your texture to be tiling and hence more detailed, increase the X and Y Size values in the textures Mapping section, whereby 10 may be a good starting value for textures with a size of 1024 or 2048 and objects that you come rather close to.

Now for the stencil map, the second texture in our stack, we need to set a few things.

1. Set UV as mapping coordinates
2. Load or create a new image with the size that fits your needs. If newly created make sure to save it
3. Uncheck the Color influence
4. Also in the “Influence” section check “Stencil” and “RGB to Intensity”

Thats it, you may won't see the stencil image itself on the model, but you should see the texture that is set to the black color of the stencil map, if you now head to Texture Paint mode and draw with a white or black brush on it, you should see the one texture disappear and instead the second texture appear.

Note, if you want to export models using this technique, you need to make sure the application, where the model should be used in, knows how to use and stack those textures as the exporters for egg files we use do not directly support this technique yet. So as the application needs to re-set the textures by using Panda3Ds texturing functions or some specific shaders, you should tell the developers how to combine those textures to get the correct result.

Shadow Maps

Shadow maps will represent the drop shadows within your game. They work just like ambient occlusion maps, black and gray areas will be drawn darker, are affected by shadow and white areas will not be affected by shadows.

To bake shadows to a texture in blender, you need to have at least one shadow creating light source. For outdoor levels you'd best use a sun lamp which will automatically create shadows for all shadow casting objects. Now, select your model that should be the shadow receiver and set the shadow texture within the UV image editor. Head over to the Bake panel and select Shadow as the Bake Mode. There aren't many settings you can set here and the few ones should be known from the previous baking sections of this book, so simply hit the Bake button and watch your shadows being created.

Now to correctly display the shadows in blender, you need to set the texture Blend in the Influence tab to “Multiply” which is by the way also the default method used in the egg file when exporting with YABEE later on. You may only see the multiple textures playing together in the Viewport Shading mode “Material”.

Note, by default, all baked shadows are pitch black, hence you won't see the texture below those shadows in the resulting egg file. To solve this, open up the shadows image in your favorite image processing application and light up the shadow parts which are to dark.

As an example for how to set the shadows to make the shaded areas 50% darker, this I how it can be done within the gimp.

1. Open up your Image in the gimp.
2. Choose Color from the menu
3. Click Levels…
4. Set the left box (the one for black) of the Output Levels color band to 128, which will be 50% black
5. Hit OK
6. Save Image

To set other percentages, just change the number in the box of the color band or drag the arrows around. If you want to have colored shadows, you may also use the Colorize or Curves tool.

Linking

Linking is a very helpful feature of blender when you going to create levels with existing assets that are stored in separate blender files. This feature will “link” a given object into your current scene where you can drag it as well as do a few other things with it, whilst it still can be edited from it's original source file. Hence you can spread the creation of assets to all the other people in your team. It also reduces the file size of your level, especially the exported ones, quite a bit.

If you link objects from other files, you should make sure the models you want to link are added to a group so they can actually be moved within your actual level file, else non grouped, directly linked objects can't be moved, scaled or otherwise edited.

To link an object to your current scene, go to “File” → “Link” and search for your model file. Now if you click the model file, you will get inside it and can search for the things you want to link. Basically you can link any object that is inside a blend file, but for Level editing we'll be most interested in groups, so open the groups folder and search for the group you want to link.

Now with the linked object in your scene, you can place it in your level, duplicate it and scale it to fit it inside your level.

For Linked objects though, it's very important to add a game property as otherwise the object itself won't get exported but only creates an empty node within the resulting egg file which we will export later. So move over to the “Game Logic” Layout and add a property by hitting the “Add Game Property” in the “Logic Editor” panel. Change it's type, which should be Float by default, to String and name it “File” (on the left side). As value (textbox on the right) you must add the path inclusive filename to the exported model, should be a relative path, that will later be placed at the otherwise empty node in the egg file. Note, you should not add the .egg extension to the filename as at release time egg files may be converted to Panda3Ds other file type with the .bam extension.

Having the game property set you can copy the linked object over and over in your world and they will all be placed in your resulting exported level. Note, if you forget to add that property and copy the object, you need to go through any copied object and add the property which can be a very time consuming task if you have to go through hundreds of objects fixing missing properties, so make sure you have it right at the beginning.

Advanced 1: Tips and tricks for level designs

In this part of the chapter we'll give you some hints on good level designs and how to make them interesting to play.

1. Don't make a level like a simple straight line. Levels where you can see through from the beginning to the end tend to be rather boring as the player can already see everything and predict what will happen, better add some curves like s-curves into it and don't let the player look to far into the level except you have some intended spots like a lookout. But even then it's a good idea to not show anything to the player.
2. Raise the difficulty of a level over time but also give the players some less difficult resting parts after a stressful part of the level. For example if you have a passage where the player has to quickly time jumps and running and successfully went through the passage, give him a few easier jumping places to slowly calm down but don't come to a sudden complete stop. This can also be done after fighting boss enemies or whatever else makes your game challenging.
3. Avoid empty rooms. Your game should not contain any useless rooms filled with nothing but a few assets. Try to even give the tiniest room a meaning. This can range from having checkpoints in the room over using a room as hideout or even having a story relevant room with very important things in it that the player needs to collect.

Advanced 2: Animated assets in levels

Animated assets need to be made clear with developers how to load them into the level if they should be linked but without a file tag or have a specific tag set to mark them as animateable as there are different ways to implement those in the games level loading code and may need to be accessed, started and stopped separately.

After all, an animated asset or if it contains animations, the main level file itself are just like a character, an actor that has a static model file and separate animation files, if not exported into one single egg file.

Those files then could be loaded in the application like a character, search for the position of the place where they would have been if linked and tagged and simply be controlled within the application. Though it's also possible to access specific objects within a loaded egg file and access its animation controls from that statically loaded models.

Hence it makes a difference if the objects are linked and tagged or just linked but not tagged with the File tag.

Chapter 5: Export

This is the last real chapter of this book and will teach you how to get your model out of the editor, into a format that can be used by Panda3D applications. There are a few ways to export your art to get it into an application but we will describe how you can use the exporter called YABEE which stands for yet another blender egg exporter. This exporter will be able to fulfill most if not all our needs for exporting game ready models into Panda3Ds own model format called EGG. Of course there are other ways for example exporting to formats like .x which can in addition to Pandas own format be directly loaded or exporting to .obj files which can be converted with tools like obj2egg which ship with the Panda3D SDK. There are also a few more *2egg tools that ship with Panda3D, but as they all have their drawbacks and YABEE seems to be the most supported and active tool to be used when its about Blender and EGG export, we'll stick with that and not talk further about the other tools. If you're interested, you can still take a look at the tools help description when called from the terminal or take a look at the given pages in the Panda3D manual over at http://www.panda3d.org/manual/index.php/Model_Export and its following pages.

Get and Enable YABEE

YABEE is hosted as a project initially created by the Panda3D community member ninth and is now actively developed by the community. The project is hosted on github, which you can find here: https://github.com/09th/YABEE

To get it, either use the Download ZIP function of github to get the latest master branch or use git to pull the sourcecode and simply update it via git methods. We won't talk about git here so whichever way you take, as long as you get a zip file of the project, you're good to go.

That's for downloading the project, now we need to get it as add on inside blender. So head over to blender and browse to File → User Preferences… or hit 'Ctrl' + 'Alt' + 'U'. This will open up a new window where you should click on the Add-ons button in the top row. Within this tab, hit the "Install from File…" button at the bottom and brows to the zip file you just downloaded. After the add-on got successfully installed, search for it with the search box on the top left by entering Panda3D. Finally check the checkbox to the right of "Import-Export: Panda3D EGG format" and you're ready to export.

Note, you need to check the checkbox for enabling the add on each time you restart blender unless you save the user preferences. Saving the user preferences will also save all other made changes like scene and layout changes, so if you like to keep the usual fresh look with only the add on enabled, start a new blender session and head over to the add ons section enable the add on and then hit "Save User Settings".

Note, for some of the features of YABEE to work, it's important to have the Panda3D SDK installed which you can find over at http://www.panda3d.org/download.php?sdk

Static Models

Exporting non animated meshes is quite a simple task. Simply select all the objects you want to have exported, then navigate to File → Export → Panda3D (.egg). This will show up the export file layout. In there, browse to the folder where you want to store your new egg file and give it a name.

Now you can set a few parameters for the export of the models. Skipping the checkboxes for actors and Animations for now, you can see options for texture processing, applying modifiers, Pview and Export PBS.

As all those items are already well described within the YABEE Manual I only give a short description of what they do.

TBS stands for Tangent Binormal Space and will generate tangent space vectors for shaders.

The Tex. processing block gives you some possibilities of how to export your models textures. They can be Baked, Simple or Raw. Whereas Raw would mostly be the best way which simply copies over the textures as they are. Simple also copies over most of the textures and materials as-is but also have the possibility to extra export Ambient Occlusion (AO) and Shadows in varying sizes.

Copy texture files gives you the possibility to set a specific relative path for the models textures. If you just enter a '.' the textures will land in the same folder as the exported egg file.

Apply Modifiers does what the name indicates, it applies all modifiers set on the objects before the egg file gets created.

If you checked the Pview checkbox, the exported model will be loaded with th pview application that ships with the Panda3D SDK.

Export PBS is the last point. It is about physically based shading exports which are used for the Physically based shading panel from the Panda3D BAM Exporter.

As soon as you have set all variables to your liking, hit the “Export to Panda3D EGG” Button in the top right edge.

Animated Models

Now that you know how to export still models, we will take a look at exporting animated models.

For animated models, make sure to not only select the model, but also the armature, or if you only want the animation, you can also only select the armature.

Let's take a look at the options you have when exporting animations. Therefore we have the following settings.

All actions as animations. This will take the animations as they are stored in the action editor. For most bigger projects, this should be the way to go.

An empty list. The list following the “All actions as animations” can be used to add key frame animations. Simply click the '+' button and set the start and end frames. Just make sure the animation is at least one frame long so you can't set something like start and end frame to 1. You can give each key frame animation a name and set it's animation speed in frames per second.

The Animation only should be quite self explanatory. Ticking this checkbox will let the exporter only export the animations of the actor but not the model file.

If Merge Actor is checked, an actor which is split into multiple parts will be merged into one.

That's all for the animation options. Now if you hit the Export button, you should get a model file and one to many animation filenames all in the style of [modelName]_[animationName].egg

To preview animations you can call pview [modelFilename].egg [animationFilename].egg and control the animation in pview using the 'A' key.

Notes before exporting

In this part, we'll give a short list of things you need to take care of before exporting to make sure your model gets exported correctly. Basically this is just a recap and short list of all the notes given throughout this book. You can also see this list as a checklist which you can go through before exporting any models.

1. Are all the models you want to export selected?
2. Are the material and texture settings correctly set?
    1. Are the influences of the textures set to the correct values?
        1. **Diffuse Texture** has Color value active and set?
        2. **Ambient Occlusion** has Ambient value active and set?
        3. **Normal Map** has Normal value active and set as well as Normal Map below Image Sampling checked and the TBS generation in the export menu is set to anything but no?
        4. **Glow Map** has Emit value active and set?
        5. **Gloss Map** has Intensity value under Specular active and set?
    2. Is the coordinate of the texture mapping set to UV and has it the correct map name given if there are more then one UV Map?
    3. If the material/texture contains transparency, make sure the following things are given?
        1. If the texture has alpha values, make sure Alpha under Diffuse in the Influence section is active and set?
        2. In the game logic editor, properties are set for:
           blend → ADD
           blendop-a → INCOMING-ALPHA
           blendop-b → ONE
           bin → fixed
           (You may need to tweak those to your needs)
3. Do all actions have a fake user and no action is actually applied to the armature? Otherwise this action will get exported twice or more times (it will get exported as many times as the number of users the action has)?
4. Are all objects correctly tagged?
    1. Are your linked objects tagged with the File tag and the paths correctly set?

2. Are collision solids correctly tagged with game properties?
    1. Normal invisible barriers can be tagged with
       object-type → barrier
    2. Level ground should be tagged with either
       object-type → floor
       object-tpye → dupefloor
    3. Spheres, tubes and other types of geometry should be tagged with specific object-types if possible. See confauto.prc which is shipped with Panda3D for available types.
    4. optionally you can also set the collide tag directly as game property.
    5. Collision objects may also have bitmask tags, make sure the mask values are set correct (you might need to check with the developers to know which masks should be set)
5. Are the exporting options correctly set?
    1. Make sure if you have animations in the action editor, that All actions as animations is checked.
    2. If you use the timeline for all your animations, make sure to have all start and stop keyframes are correctly set in the list, are at least one frame long (From 1 To 2 = 1 frame) as well as have the correct frame rate set.
    3. For textured objects, make sure the Texture processing is correctly set (most of the time this might be Raw).
6. After exporting make sure the model got correctly exported by checking it with pview!

That's it, now you should know how to create and export game ready models. Now head over to your developers, share them with others via online platforms like blendswap.com or read our other book about Panda3D game development to get your models into an awesome game.

Chapter 6: More information

In this section we just list a few tutorials and websites that we viewed and learned from ourself and thought would be good to share if you want to know more about specific areas of the various tasks that are necessary for 3D modeling and animation. You should also always make sure to not only learn from just one book/tutor/website/whatever but search and look around to find as many information, workflows and inspiration as possible to create your own unique style in doing things. This will be what makes you special and standing out from others and lead you to success.

Useful Add-Ons

In this part, we will list a few links to blender add-ons which may help you in your day to day work in blender to create models for Panda3D.

BlenderPanda

https://github.com/Moguri/BlenderPanda

An add-on that adds a render mode to blender which renders the model as it would be within a Panda3D application.

Panda3D-Bam-Exporter

https://github.com/tobspr/Panda3D-Bam-Exporter

Another modern and feature rich exporter which will export your models directly in the bam file format which is Pandas binary model format.

Notes for bam files: Models in this format will be loaded faster than egg as well as being smaller in size. Though, bam models are version bound and may only be used with the Panda3D version they were created for. If a game gets released through Panda3Ds packaging tools, egg files will be converted to bam files automatically anyway, so in the end it doesn't matter much what you use to work with.

More Tutorials

Here we will list a bunch of tutorials where we learned from.

One of the first resources you might want to look to know the ins and outs of blender is of course it's user manual which can be found at:

https://www.blender.org/manual/

The CG Masters website also has quite a lot of free blender tutorials with many useful for general creation of game ready models as well as other areas of 3D modeling.

http://www.cgmasters.net/category/free-tutorials/

Blender Guru is a page not just for tutorials, but it sure has a bunch of useful information.

http://www.blenderguru.com/

From that page there also are the videos, which the rigging techniques we showed in this book are mostly based on:

http://www.blenderguru.com/tutorials/introduction-to-rigging/

A bit outdated, but still with a few nice techniques in it which can still easily be adapted to newer blender version, are these Blender game engine videos made from the Yo Frankie game:

https://vimeo.com/album/49699/page:1/sort:preset/format:thumbnail

Another nice source for general purpose blender usage is this website:

https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro

Also a good site for learning blender for game ready content creation:

http://www.katsbits.com/tutorials/

A nice tutorial video on how to draw hand painted tiling textures within blender:

https://www.youtube.com/watch?v=bp-hESP55js