MoviesBandit's Blender Tutorials

Table Of Contents:

Introduction:

This tutorial is aimed at beginners who want to learn Blender3D. It also focuses on The Movies for content creation but most of the steps can be used for making mods for other games. If you need to learn Blender then this is a good starting point. I will assume you know nothing about Blender and go over each step with lots of pictures.

The Movies is a 3D Movie Editor and Exporter created by Lionhead Studios LTD. It is also a game in its secondary function. Most users ignore the game portion and focus primarily on making movies.

It is possible to add more content to The Movies. The process requires adding properly named subfolders within the game path. These folders will contain all the files required to make a new item available to the game. In order to become familiar with those files we can export an existing item right from the game using a program called MED - **The Movies Editor.** Usually there will be some **(.INI)** files. They tell the game which new item is present or available for use and selection. Other files are in **(.DDS).** These are image files the game uses. They show up visually right on top of the 3D models found in the game. A wood looking texture will show up on a floor set. Other image files are thumbnails, or overlays or backdrops.
The last files of interest are the **(.MSH)** files, the 3D models the game uses in almost every instance. Creating 3D (.MSH) files is what this tutorial is dedicated to.

(.MSH) files, being 3D objects, will need a 3D program to make or edit. **Blender3D** is a free program, the one most of us use for The Movies. https://archive.blender.org/development/release-logs/blender-249/index.html There was a web page dedicated to making tools for modding The Movies called DCModding.com. DCModding released the **Python Scripts** for importing and exporting 3D models (.MSH) into The Movies.

For making mods for The Movies, you will need Blender3D **version 2.49b** (and not any other version. Blender 2.5 and up will not work with the scripts.) Blender requires Python Scripting Language to function properly. Python 2.6 https://www.python.org/download/releases/2.6/

You know Python is installed when the console window says… got it!

You will also need the import/export Python Scripts from DCModding. A program called MED - **The Movies Editor**. This program extracts the files of the game, but also injects new content into the game. MED and the Python scripts can be downloaded from TheMovies3D.com. You will also need a paint or image program that can edit (.DDS) image files. The one I use is free called **Paint.Net** GetPaint.Net.

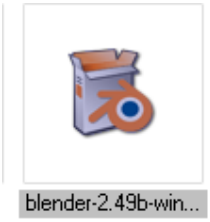
The process of learning Blender will follow a basic path in this tutorial. First installing Blender (and be sure to tell the installer where the **scripts** folder is located during the install). Setting up Blender to be a bit more user friendly, with the needed windows made available for use. Familiarizing yourself with Blender and it's functions. Some basic steps for importing a model. Also how to create a very simple mod for The Movies. How to use the import and export Movie Python Scripts to get your mod into the game. And will then explore some of the tools and procedures for using Blender.

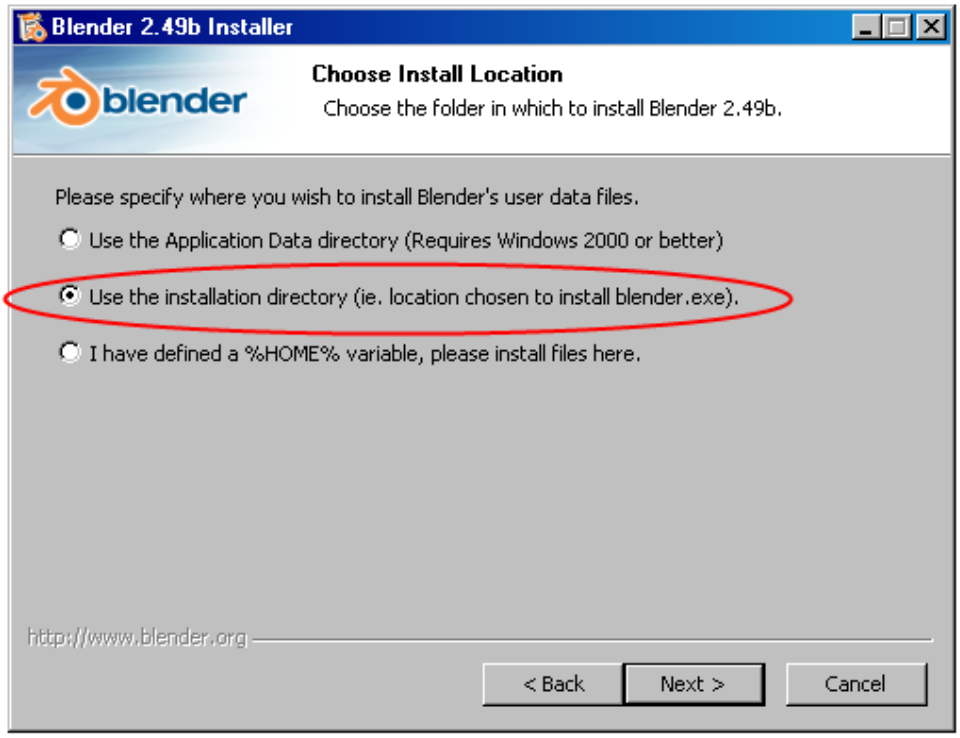
You will always want to save your project as you go along in case Blender crashes, which it does sometimes. The files it creates are **.blend** files. And they will open with Blender.

1. Setting up Blender 3D

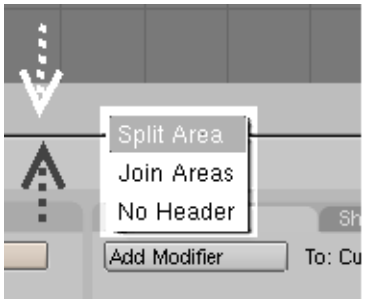
Important Note: Blender 2.49b's install file will ask you where you want to install the ".blender\scripts" folder. Don't ignore this question. By

default the location chosen is the hidden AppData folders. Instead choose the location that Blender is already installing itself.





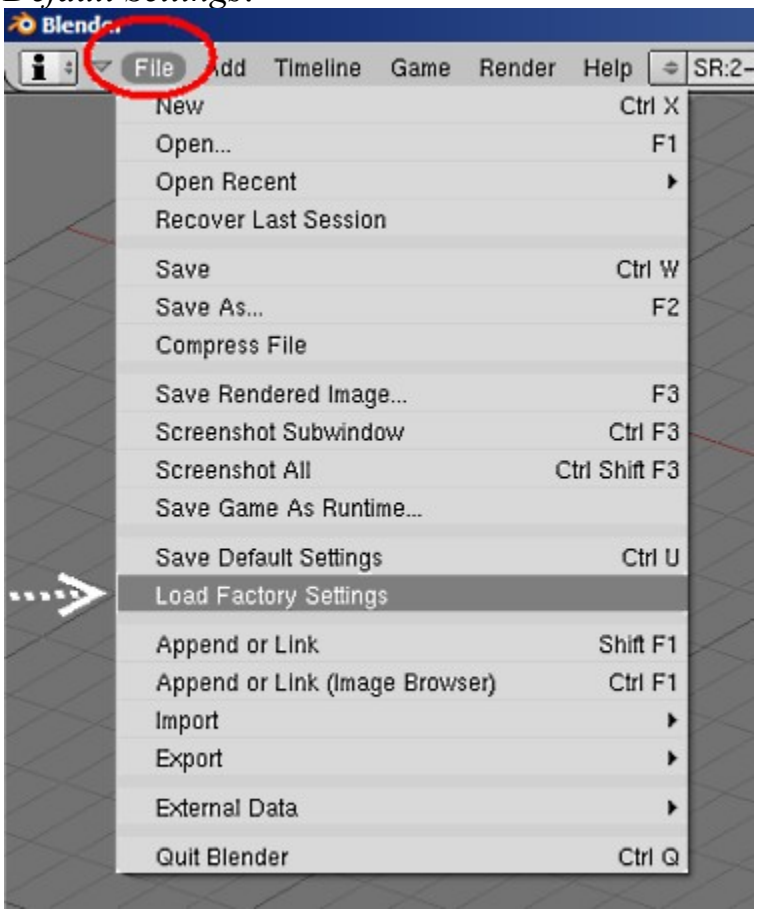
You will also need to place the *The Movies 2009 Import/Export Scripts* into Blender's **scripts** folder… blender-2.49b-windows\.blender\scripts. Blender has two windows going on. A black one (console window) and the editor window. You can switch between windows when you need to. If something goes wrong in Blender you can go to the black window to see some information about what is wrong (generally).



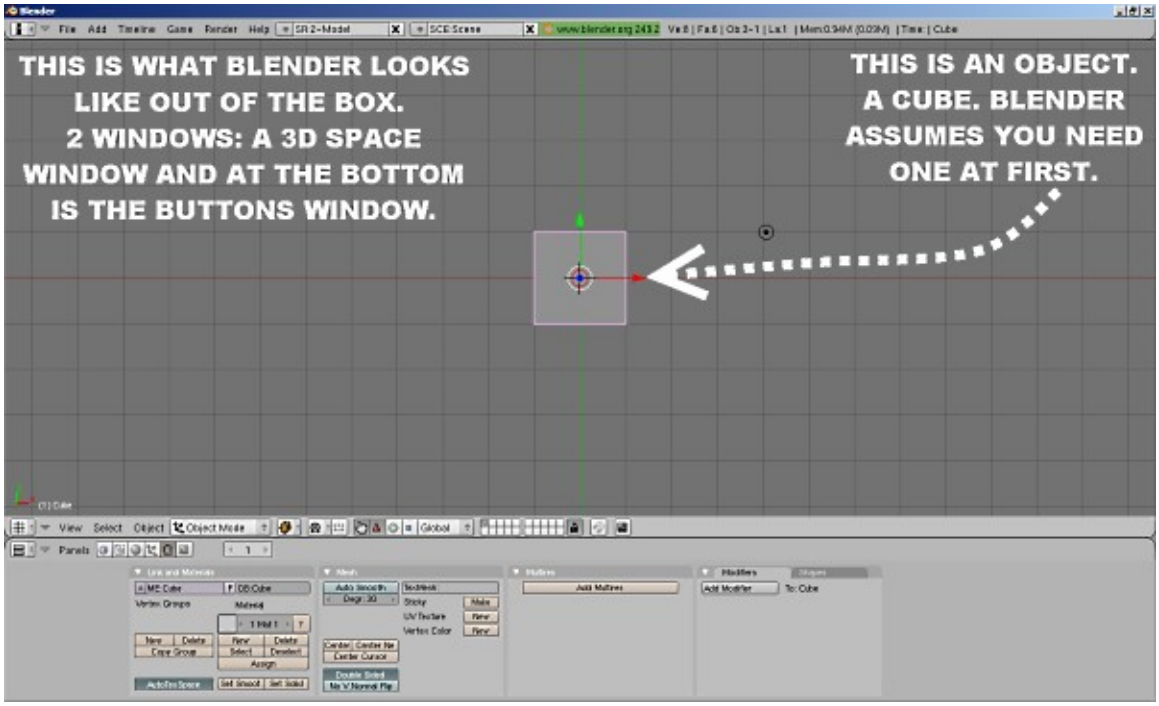
Windows can be changed or customized for your convenience. If you hover over a window's border, you can click hold on it and drag it larger or smaller. You can also create new windows.

Often you will change windows, Move menus around. Create more windows. Depending on your project, these could often change. DCModding

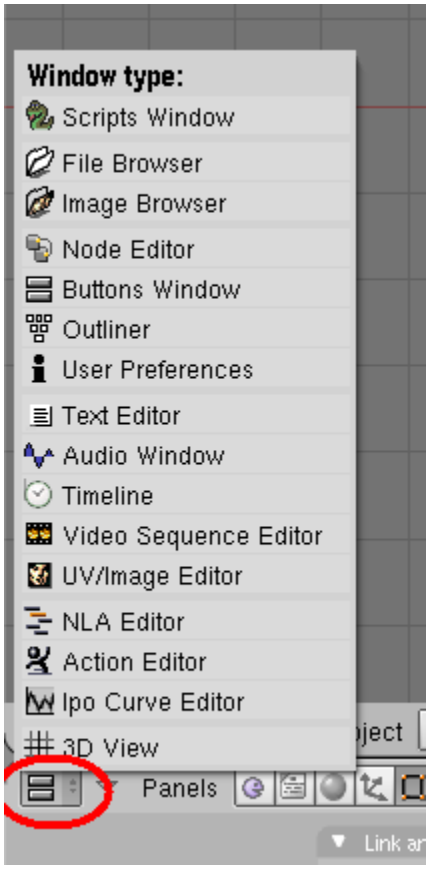
had some very smart suggestions for setting up Blender to mod the Movies Game Meshes. We need Blender's tools ready for the job.
You can always get Blender to an "Out Of Box" state by loading the *Factory Default Settings*.



In the picture below is what Blender looks like when you run the program for the first time. It is divided into 2 windows. Above is the *3D* window and below is the *Buttons* window.

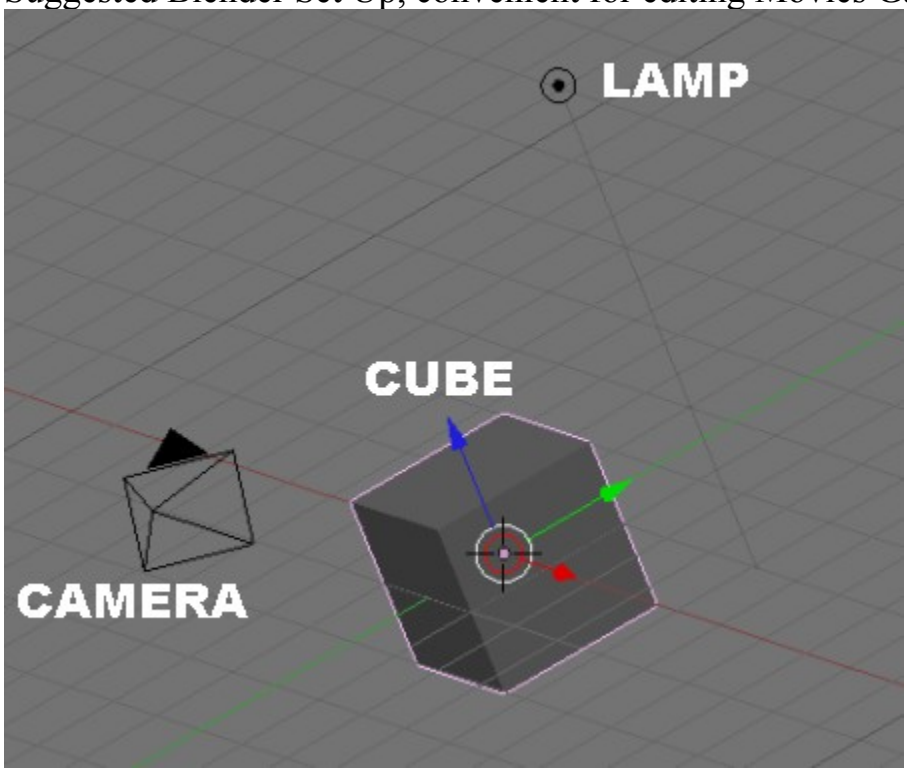


Each window can be changed into another window by clicking the tiny *window icon* found in the corner. More about that later.



There are 3 objects in the 3D window. A Cube, a Lamp and a Camera. The

cube already has a material. Since a camera is present, and the scene can be lit up by the lamp, you can hit the **F12** button to render what the camera sees. A cube in 3D space. Blender assumes you need these 3 things. You don't. But why not? A cube can be turned into almost anything we would like to create in a 3D program. So you see what's happening so far in Blender. These things will be changed. I am following DCModding's Suggested Blender Set Up, convenient for editing Movies Game content.



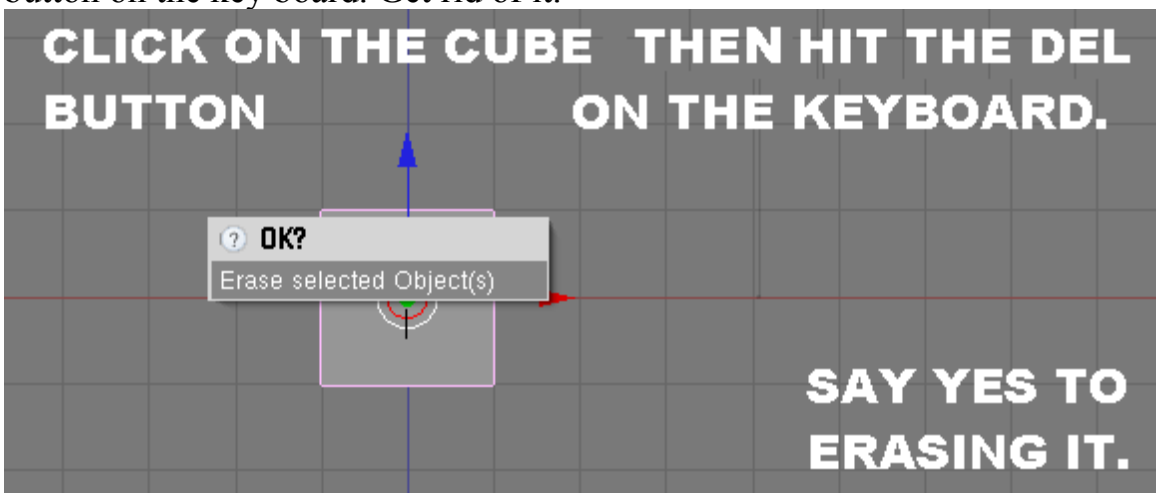
Most people coming to my sight want to know how to edit game models. Blender can do a lot of great stuff. It can film movies, it can create texture images, it can make video games. But for a beginner who wants to edit the Movies Game models, or other game models, Blender can be customized to do the job better.

At first **Left Clicking** doesn't select anything. The buttons are backwards. Perhaps the designer was left handed. In which case this is a fine choice of mouse use. Righties will want to change this. At the first run Blender selects stuff by right clicking, not left clicking.

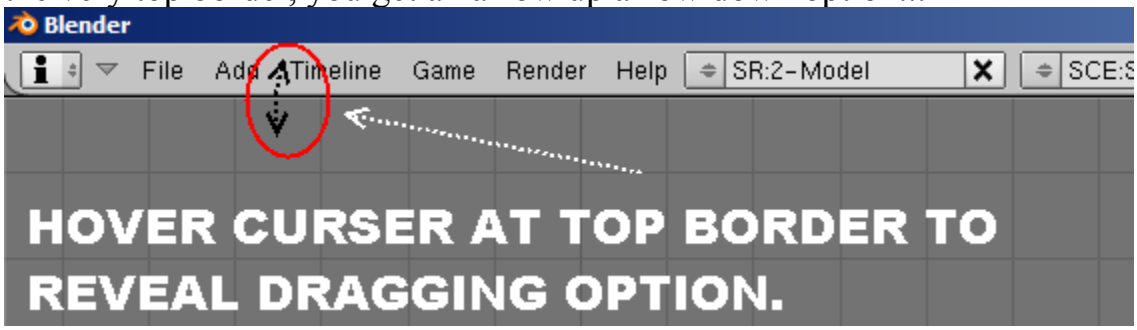
Blender is best used with a 3 button mouse. If yours has a wheel, often pressing down on the wheel as if it were a button acts as a third button. In Blender that is how you pivot the view. And it spins all around as you hold click the middle mouse button (Or mouse wheel). For me it's kind of disorienting. We will change that.

First suggestion was to get rid of the Cube object. We don't need it for

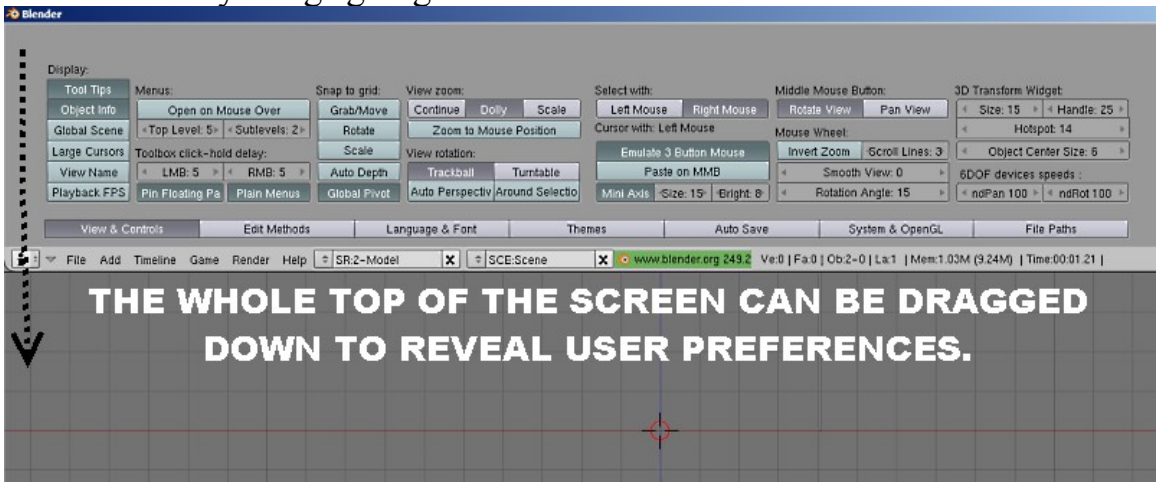
editing Movies Game content. Select the cube and then hit the **DEL** (delete) button on the key board. Get rid of it.



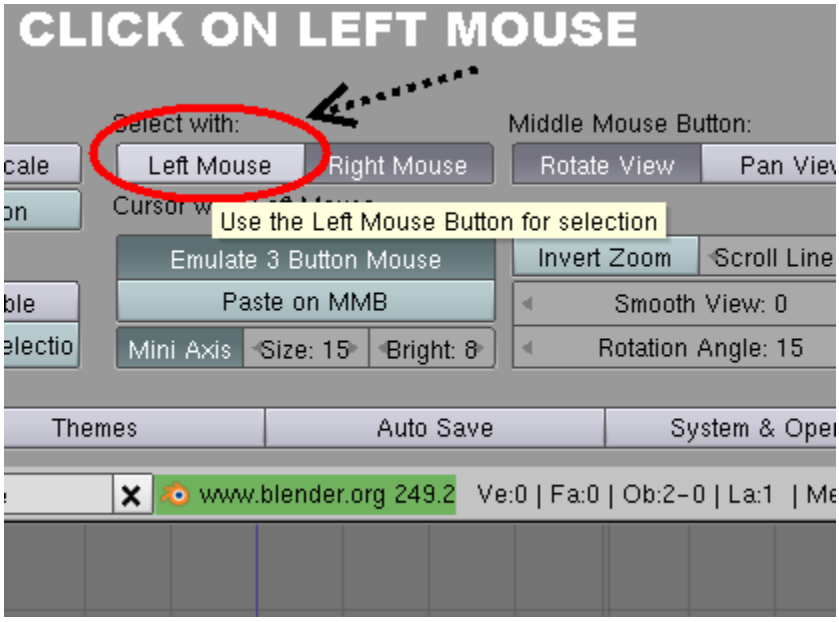
The next thing we want to do is modify the *User Settings*. Blender is also a magician's stage. Very well hidden tricks are happening. And you will be amazed! Ok maybe not but if you go up with the mouse curser and hover at the very top border, you get an arrow up arrow down option...



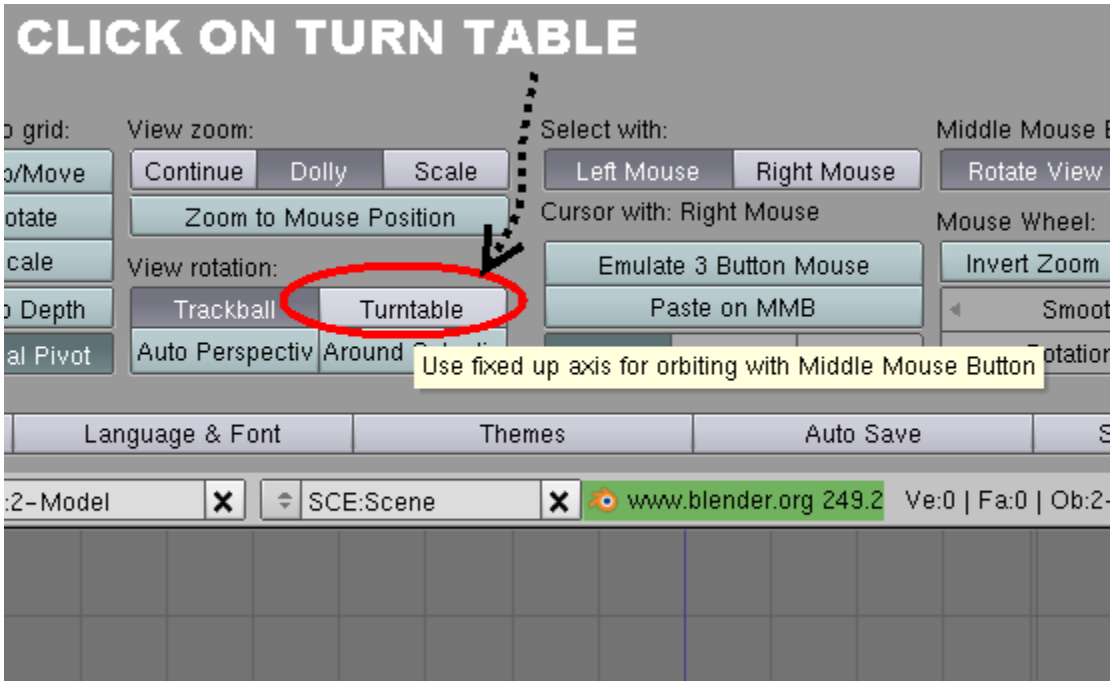
Hold click to drag down the very top border and... PRESTO... Another menu. Holy Moses! What is this? **User Preferences** and thank God. Blender had some crazy things going on but now we can force it to make sense.



First line of business, click on a button called **LEFT MOUSE.**



The next thing is to adjust how the view pivots around. To the left of the mouse setting is buttons under the View Rotation and here we will click the button called **TURN TABLE**.

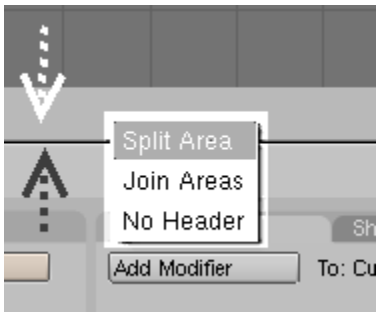


Thank Goodness! That'll make things easier for us. That's it here so we can close the user preferences window. Hover over the bottom border of it and drag it back up.

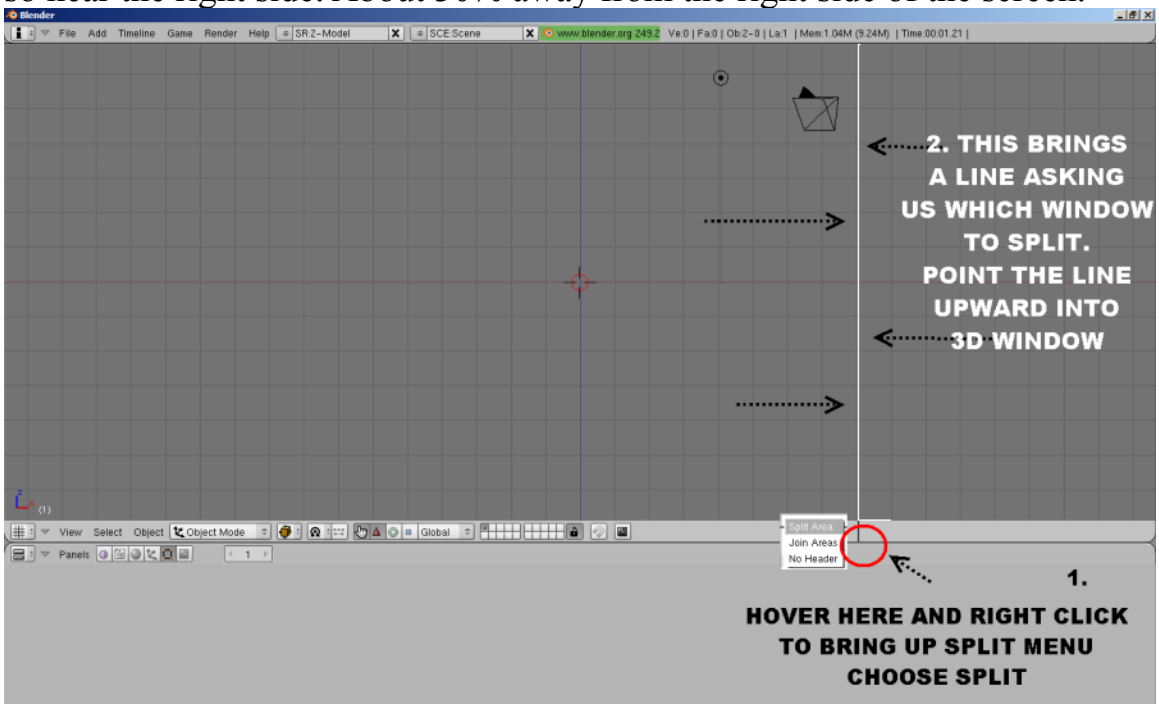
Next we want to create more windows we will need. Models have information we need to get at and fortunately there are windows that get at them. But we can't see any of these other windows. If one wanted to, they

could temporarily change an existing window into one you need, then change it back when you are done. That would happen by clicking an icon in the corners of these windows that will bring up a popup window. Then we can select which window we need for the moment. But the size and dimensions of these windows is not so handy at the moment. We need more of them.

What we will do is go to the border between 3D space and buttons window and hover over it. Then left click on the border to bring up a quick menu to choose from.

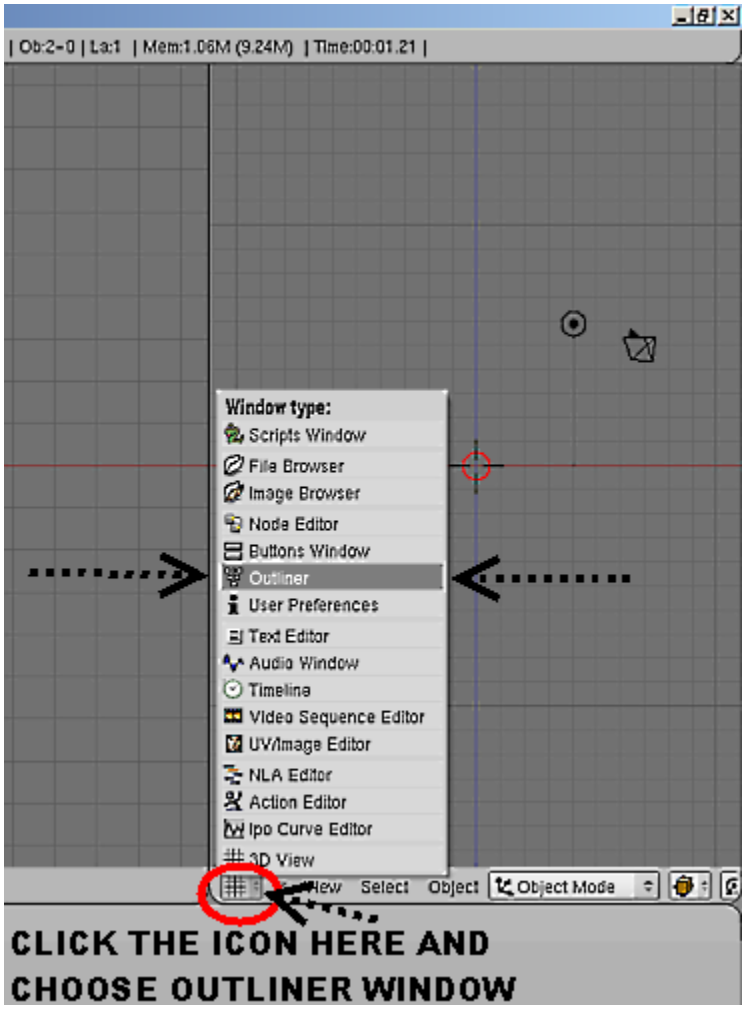


We want to **split** the window and this brings a flashy line. Aim this line where we want it to become split. Aim the line upward, not downward. Leave the *Buttons Window* alone. But Break the 3D window in two. And do so near the right side. About 30% away from the right side of the screen.

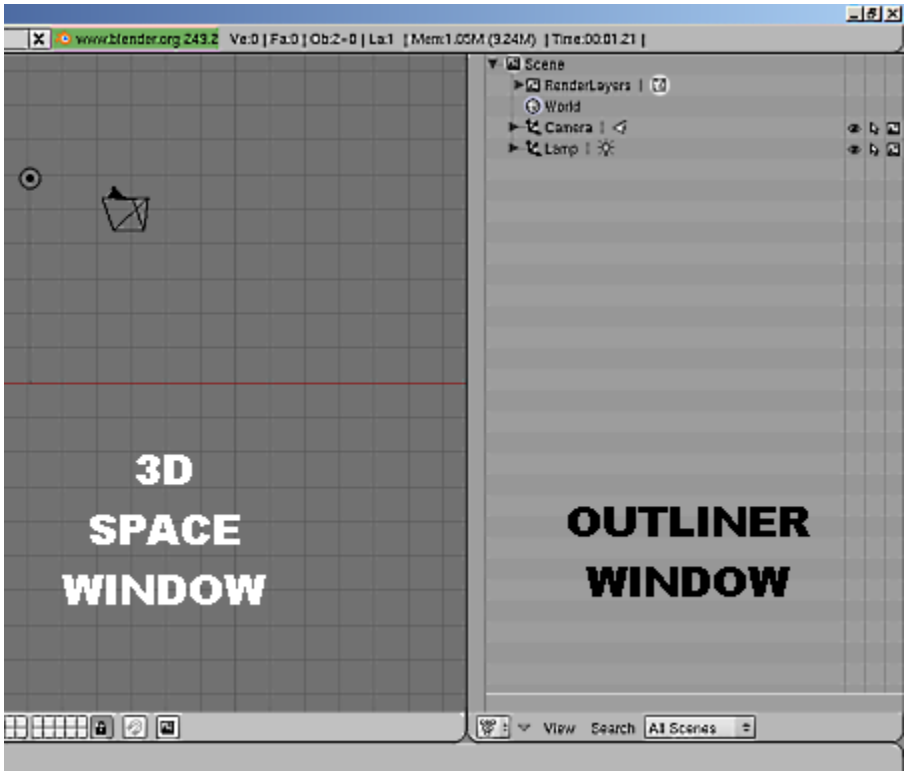


Now we have two 3D windows so we need to change the one on the right to one we need. Every window has an icon button that tells Blender what

window it is. Usually in a corner. Here we will click on it and this brings up a popup menu of windows to choose from. We will choose the *Outliner* window.

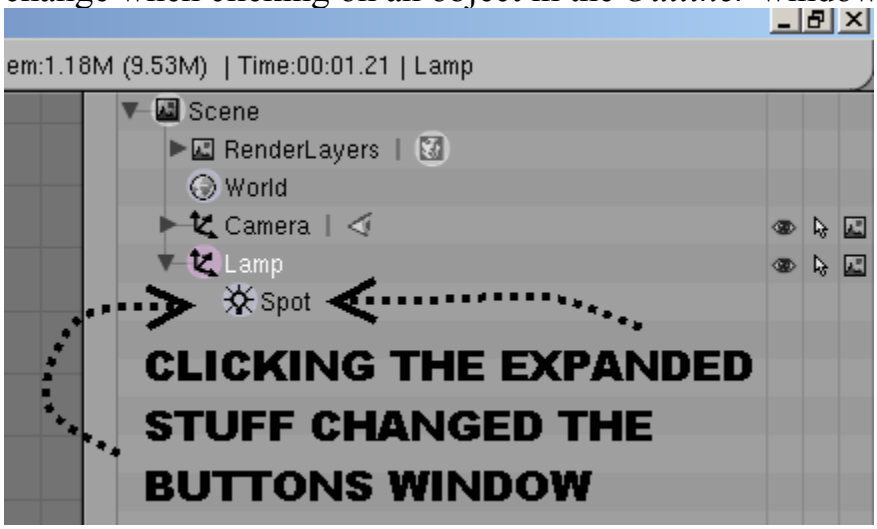


In the *Outliner* window we see the names of the objects found in 3D space. Since we deleted the cube, the name **Cube** is not present. But we can see **Lamp** and **Camera**. You can leave the lamp and camera objects. The export script will ignore these so who cares. I don't like them and I delete them. If I need them I add them on the fly. But it wasn't suggested to me to delete the camera or the lamp.



In the *Outliner* window we can select objects in 3D space by clicking on its name. These items get lit up when it is selected. Eventually you can see the *parent* and *child* order of objects when we get to those chapters.

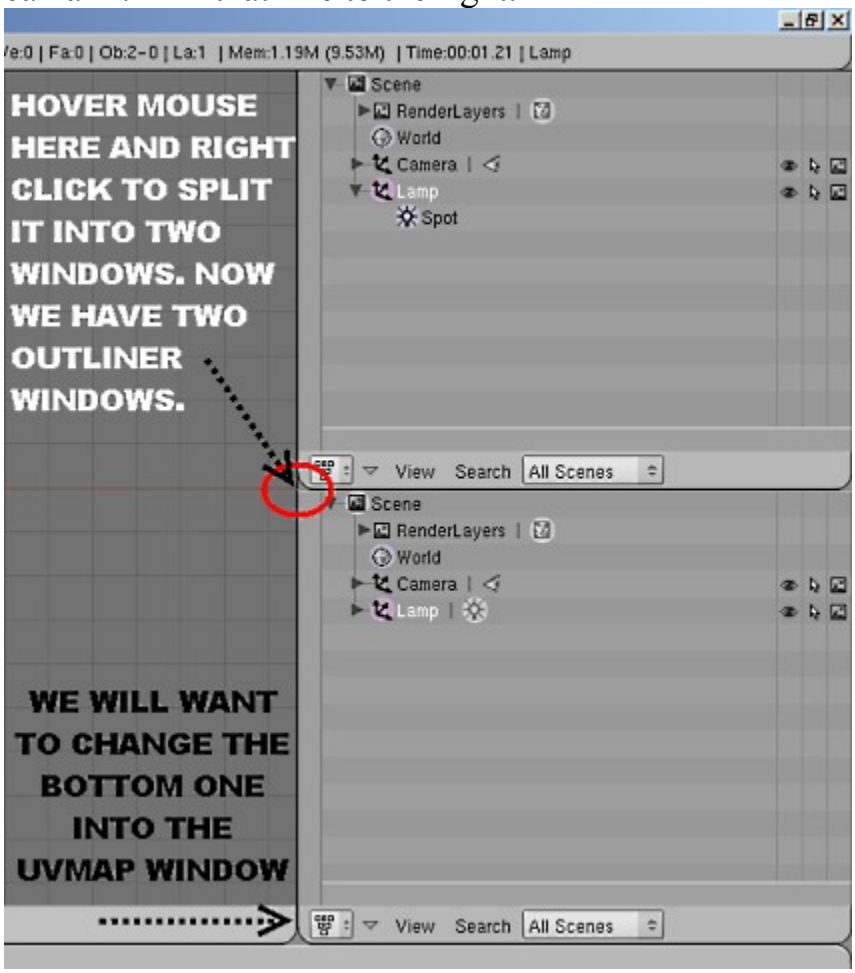
Clicking on the items in the *Outliner* will change the *Buttons Menu* window. The settings found down there are for additional modifications and will change when clicking on an object in the *Outliner* window.



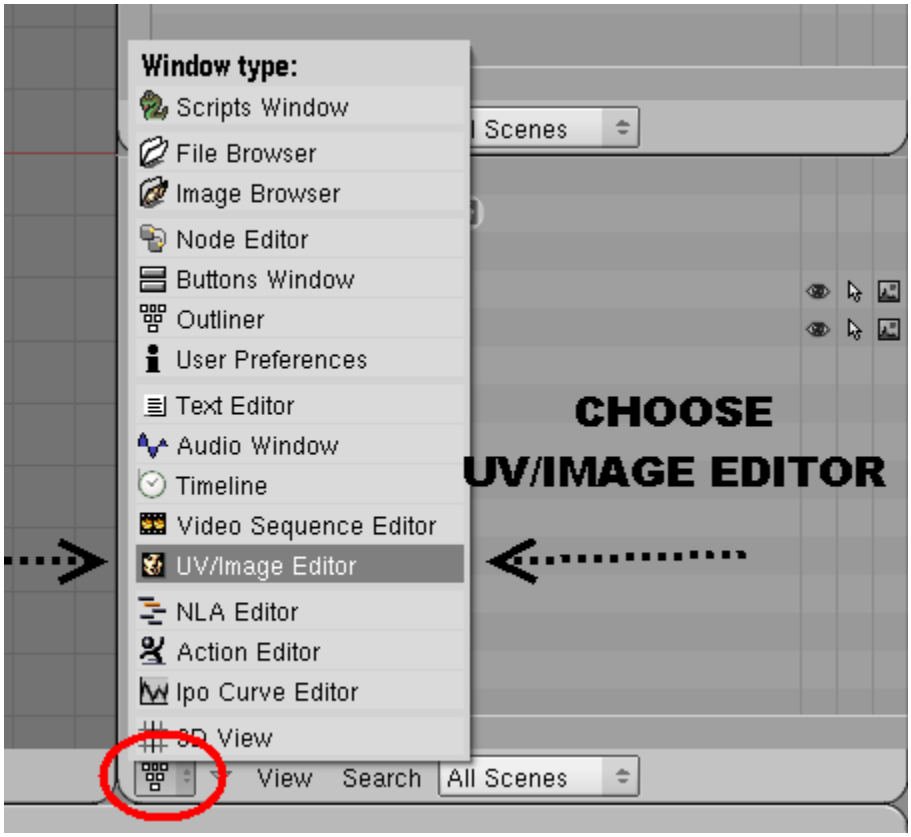
Another tip in the *Buttons* window, Holding down **CTRL** while using the mouse wheel will drag the menus around or even pan in and out. If you have a hard time seeing the menus you can make them bigger, and even drag the

line above upward to reveal more. And you may do this often during your projects.

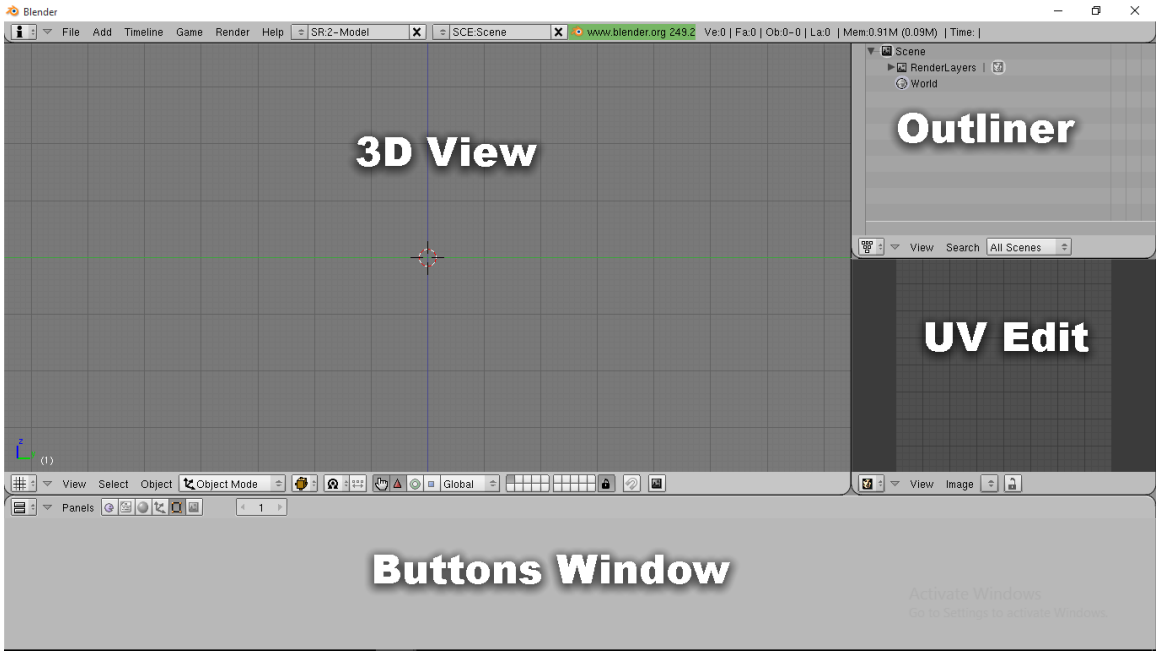
Another window we will need is called the *UV Edit* window. It is a very important window because most 3D models (from a game) have a **UVMap**. Lets split the *Outliner* window in two. Blender ask where we want a new window when we split them. Hover over the line between the Outliner and the 3D Space window. Then select **Split Area**, which will bring up a line we can aim. Aim that line to the right.



This will give us two *Outliner* windows. Every window we split will become two of the same. The bottom one we will turn into the *UV Edit* window. We would do this by clicking the window icon (as we did for the *Outliner* a bit ago) and this will bring up a menu of windows to choose from. Select the *UV/Image Editor***.**

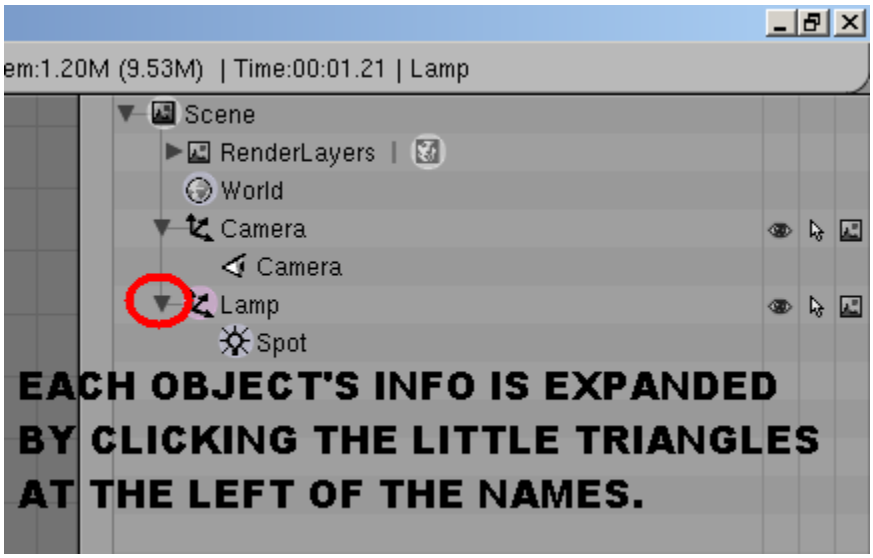


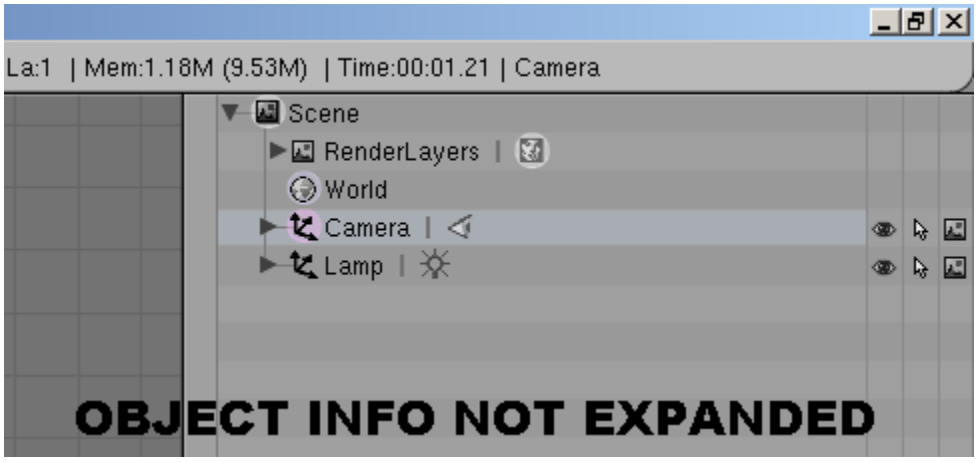
Now it has turned into the *UV Editor* window.



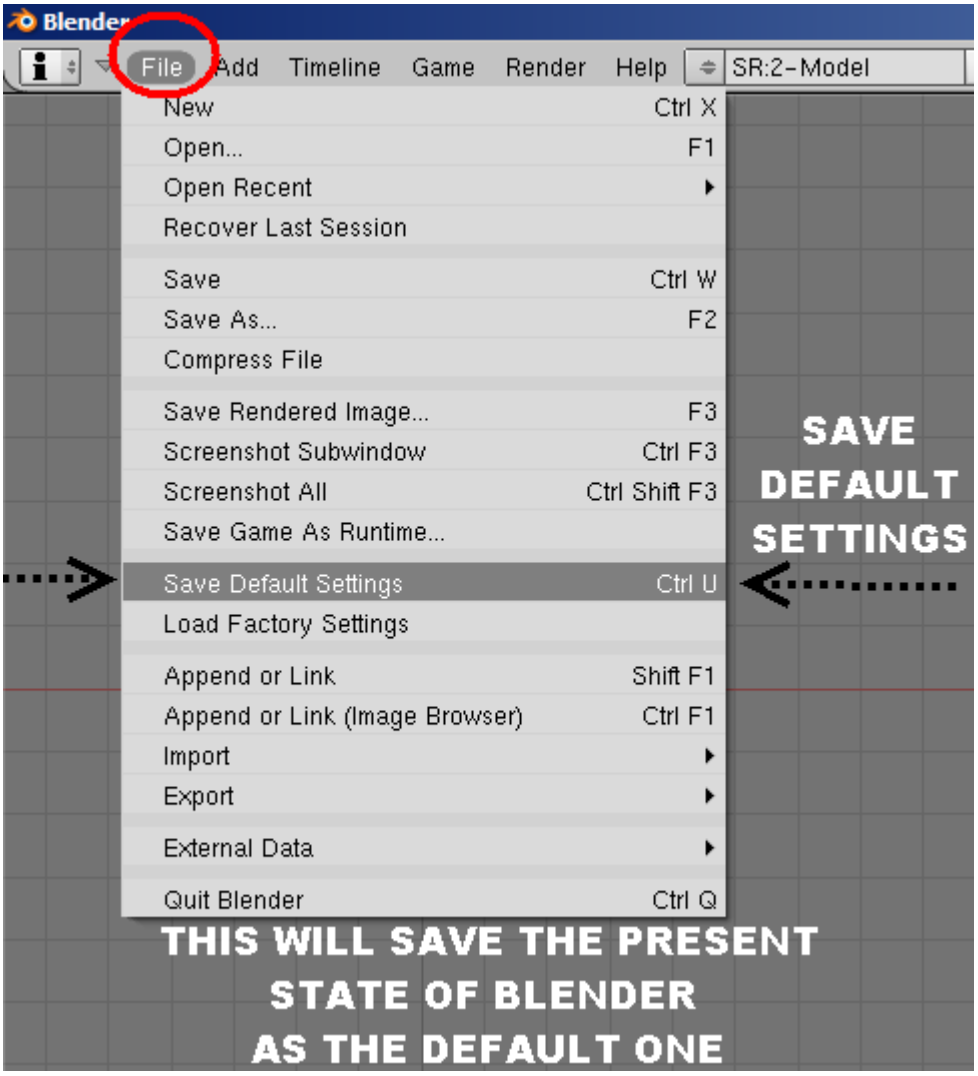
Now we want to save this setup as the *Default Settings*. This way whenever you start Blender, it will look like this. A more handy approach to modeling Movies Game Content (Or Game Models). Because we are saving the *Default Settings*, we want to make sure that everything is calm and peaceful

and ready for something new. It would be nice to not have anything selected in the *Outliner* or in *3D* space. Hotkey **A** is a handy and often used tool. Simply *Select All* or *Select None*. Press it as many times as you like during a project. A thousand times if you want, don't be scared of Blender. But in this case we want nothing to be selected. If nothing is already selected then hitting the **A** key will make everything at once become selected. The objects in 3D space will all light up when selected. Press **A** until nothing is selected. In the *Outliner* window, if you expanded the drop down info of, say the lamp or the camera, undo that. So the little triangle doesn't point down but points at the name.





Now for finalizing the deal. Let's save the present state of Blender as the *Default Settings*. Go up to **FILE** (is a button at top left), and from the drop down menu select **Save Default Settings**.



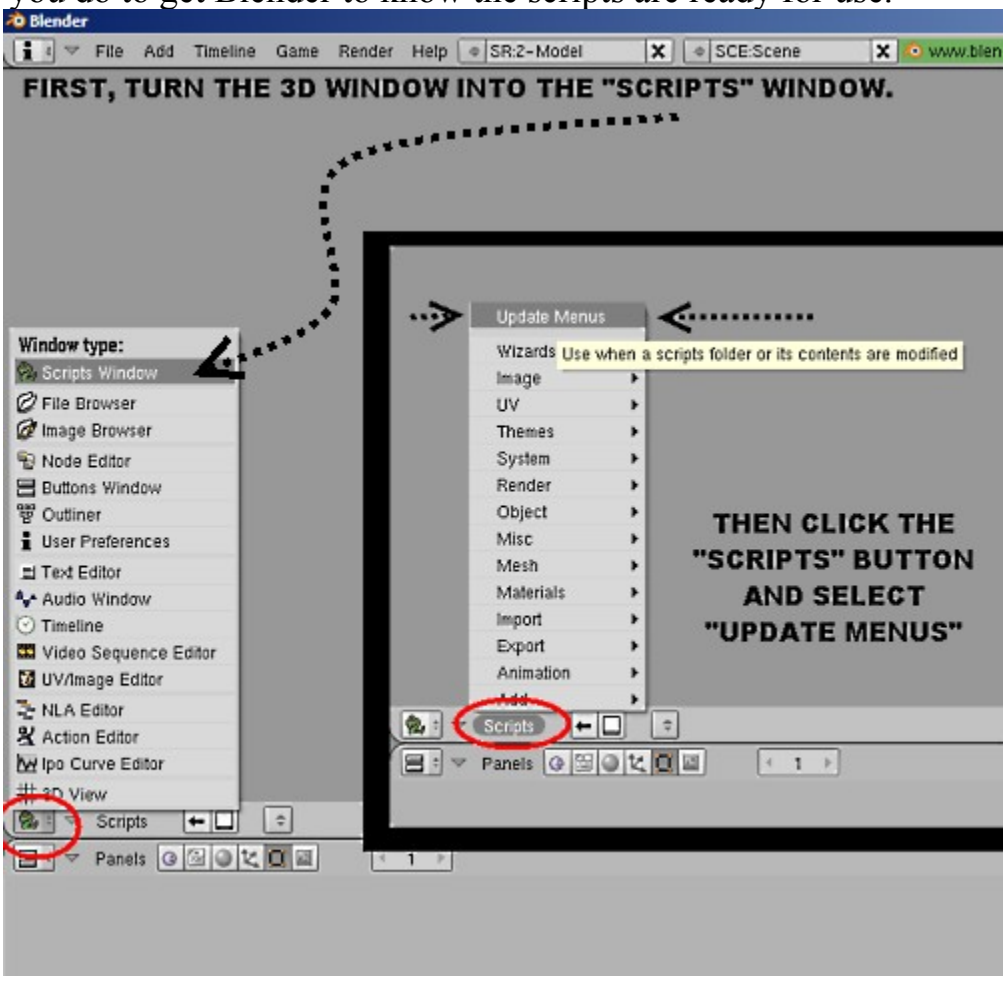
Next step is to install the *Import/Export Movies Game Scripts* for Blender. They were originally released by DcModding.com but the site has closed. Once you extract the contents you will need to add them to the proper location that Blender needs them to be in. This goes for any python scripts, or any Game you want to add models to. The scripts created for them will need to be where Blender can find them.

All scripts go into the
C:\Program Files\Blender Foundation\Blender\.blender\scripts

Now there may be an issue here. When you install Blender it will ask you were you want the folder called ***".blender"***. And there is a default location chosen. The default location is a bad one. It won't be at the address listed above. It's not bad but hidden. The default will be in a hidden folder you can't see unless you set the folder options to see it. It is in the user **AppData**

hidden folder. Google "Hidden AppData Folder" to find out more. If it was installed there then you need to find that location instead. In that case make sure the eventual location is in the **"...\.blender\scripts"** folder. If you are a normal person and get annoyed by hidden folders you can uninstall Blender and reinstall it, and make sure you tell the installer where you would rather see the *".blender"* folder. Which is where Blender actually installs itself. You would have to go through this setup tutorial again. But it is your choice.

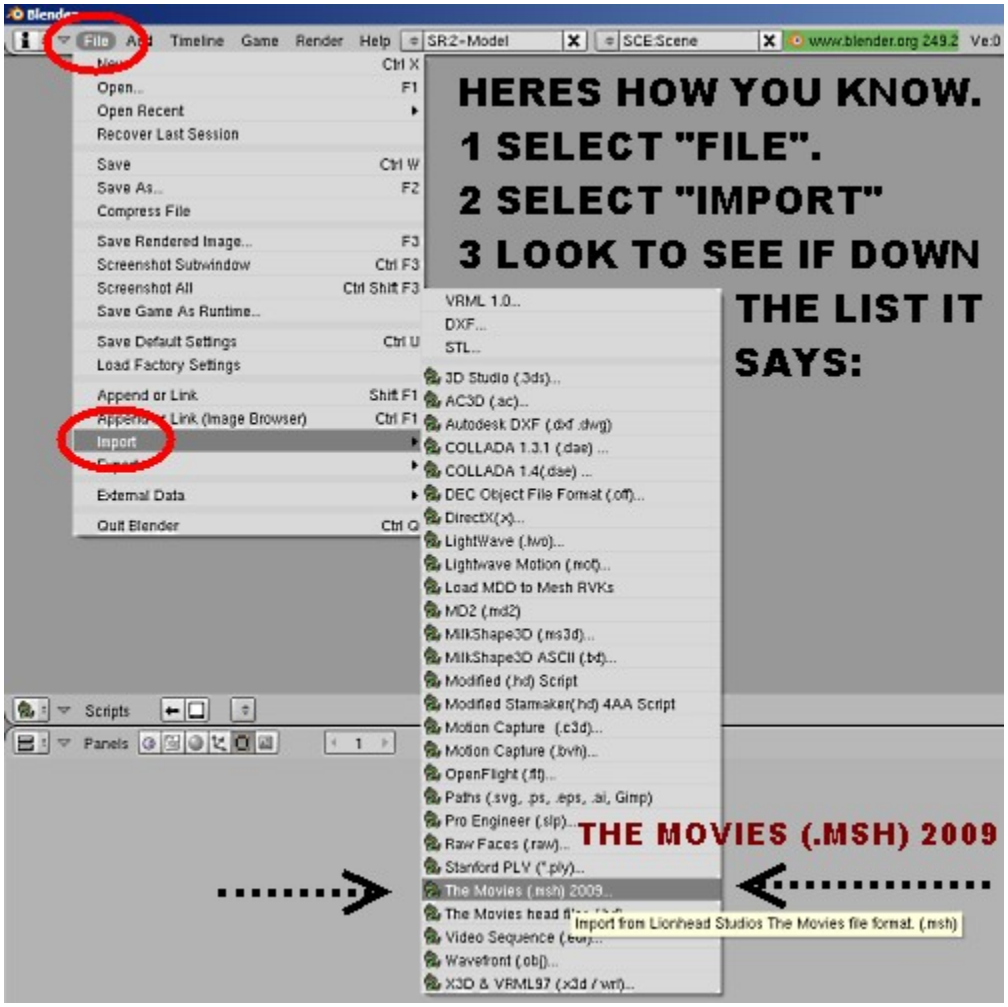
After you extract the Import/Export scripts and installed them, you can go back into Blender just to make sure the scripts have been updated. I've never needed to do this but did it anyways. And whenever I installed them in the right location, the scripts were already available to me. But it may be that they are not yet seen by Blender. If you are going through that then here is what you do to get Blender to know the scripts are ready for use.



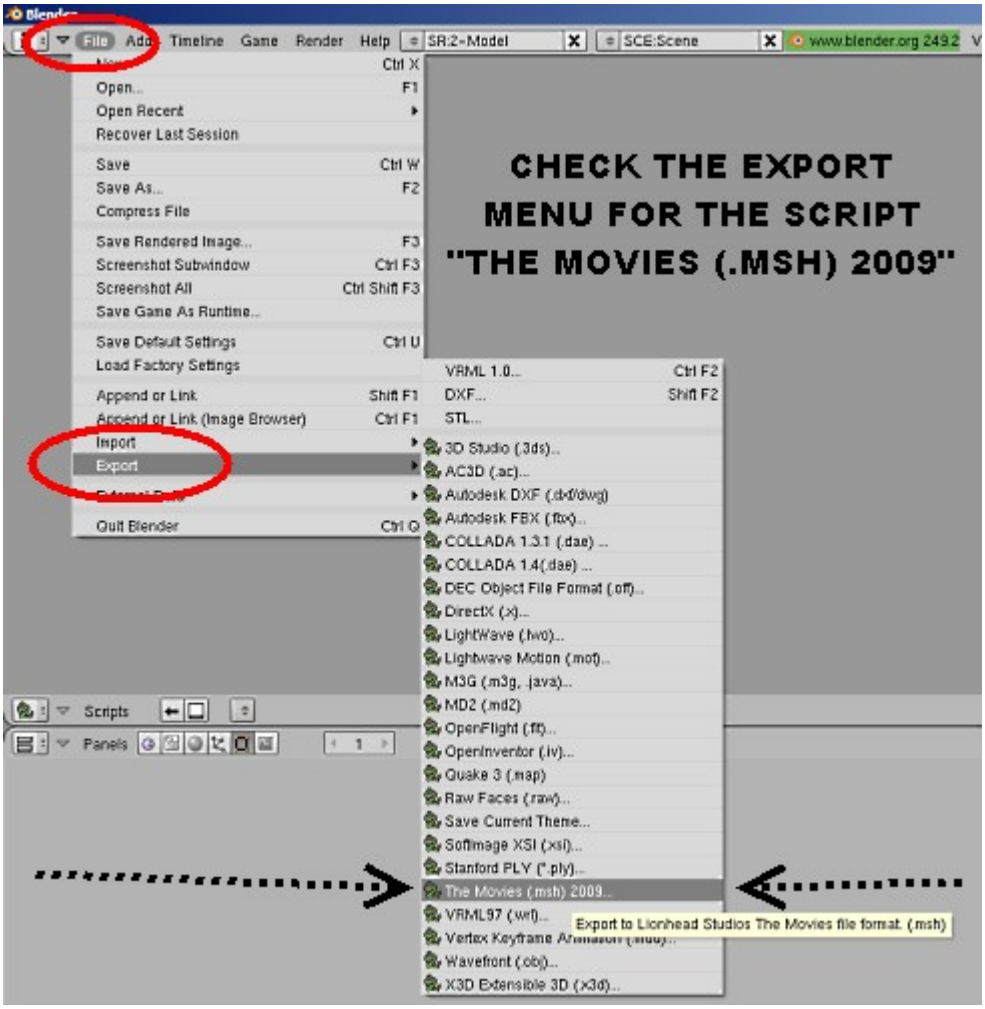
First turn the 3D window into the *Scripts Window*. Do this by clicking on the Icon that changes windows. Select the *Scripts* window. Next to the Icon is a **Scripts** button. Click this button and when the popup menu appears select

Update Menus. That will let Blender know the scripts have been added.

Now verify the scripts can be seen by Blender, go up to **FILE** and when the popup menu appears, select **IMPORT.** That will bring up a list of all the import scripts Blender can see and use. If you see **The Movies (.msh) 2009**, then the install was successful.

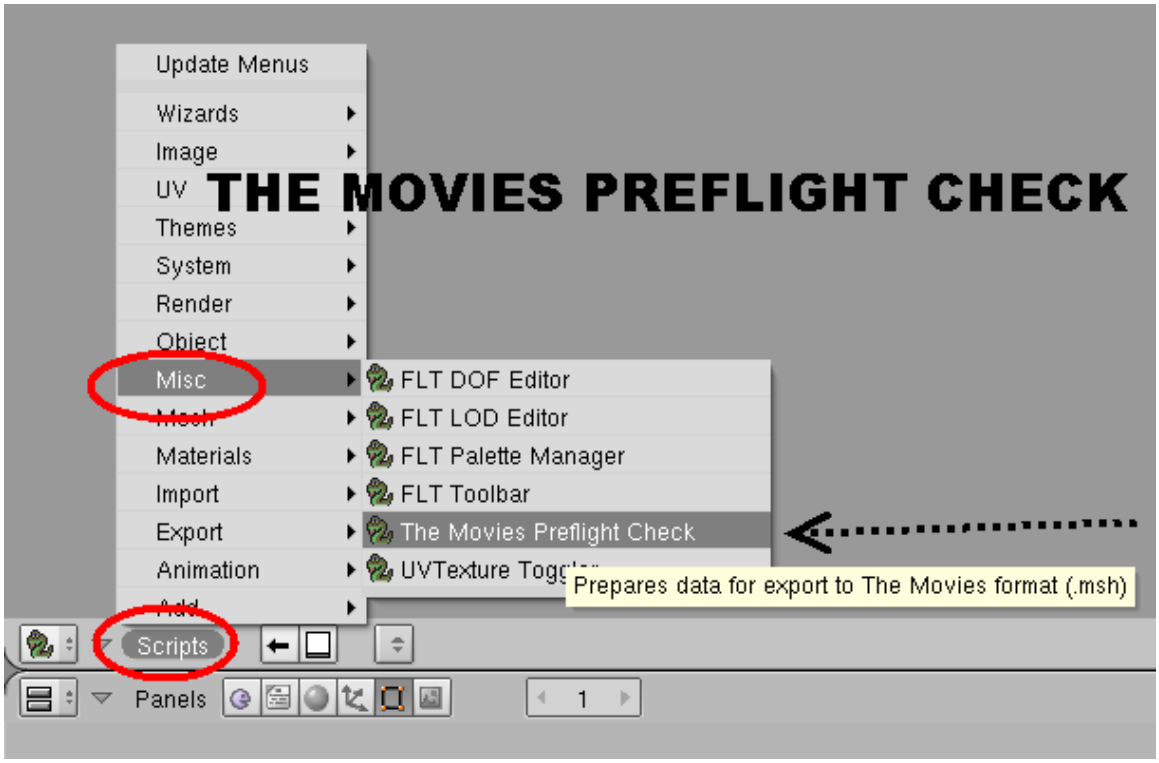


Also check your export menu for the Movies script.



One more script to check for is **The Movies Preflight Check**. This script will go over your project and see if there were any mistakes. And what needs more attention. You will know when the 3D model is ready for export when this Preflight comes up all clear. You can find it in the *Scripts* window, and click the **Scripts** button. From the popup menu select **Misc.** That will bring up another popup menu. From there look to see if the script is present.

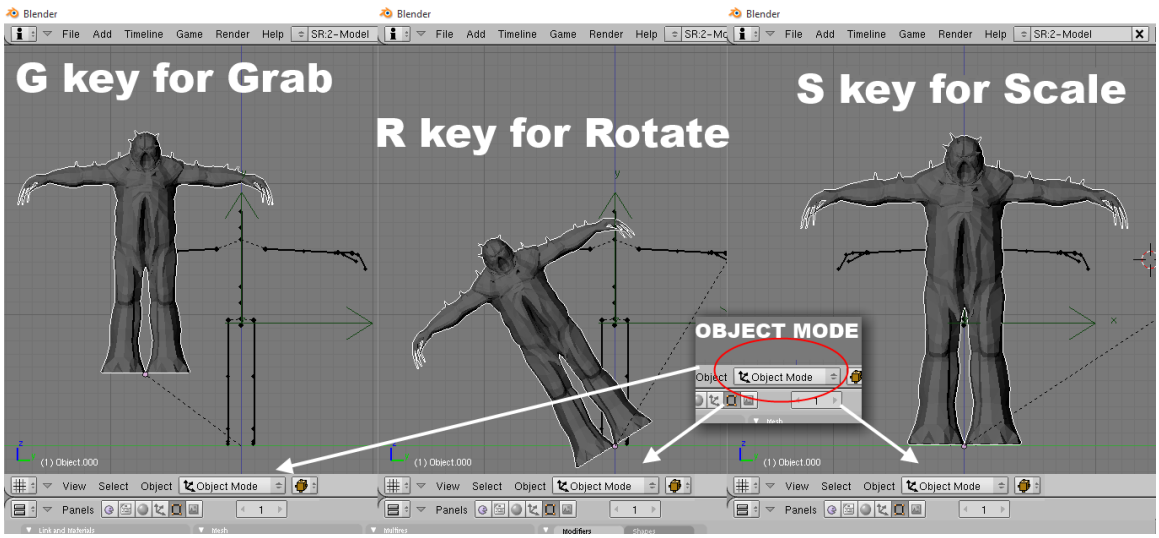
The Movies Preflight Check



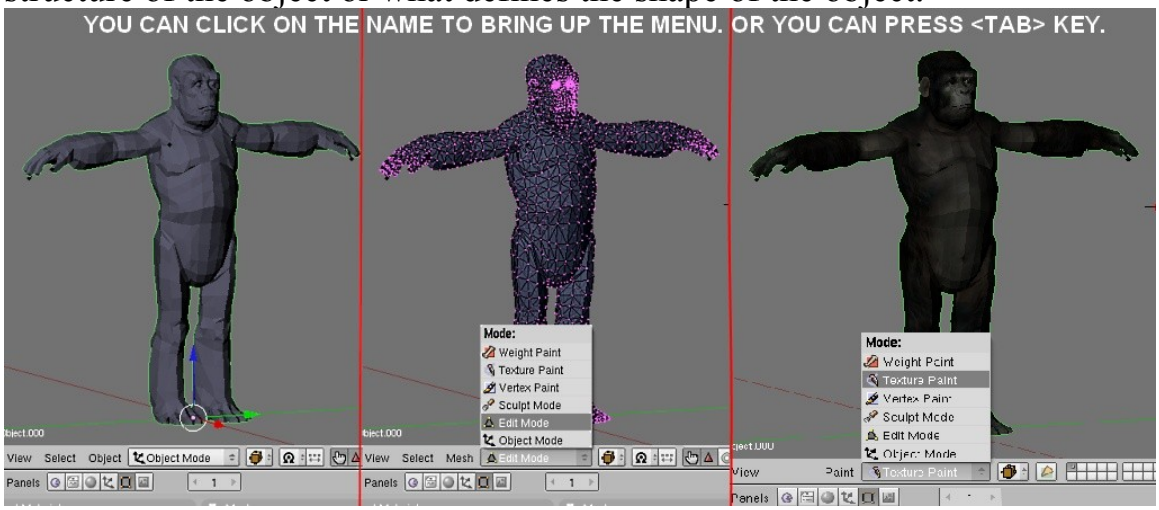
DCModding created this as a script that runs over your project when you are ready to export your mod into The Movies. Our mods in Blender need to be arranged properly. They need to have the correct names, proper materials, no hanging items. To know if everything is good or if something is still missing run this script and it will let you know by opening a browser window listing a report of your Blender project.

2. Getting Familiar With Blender

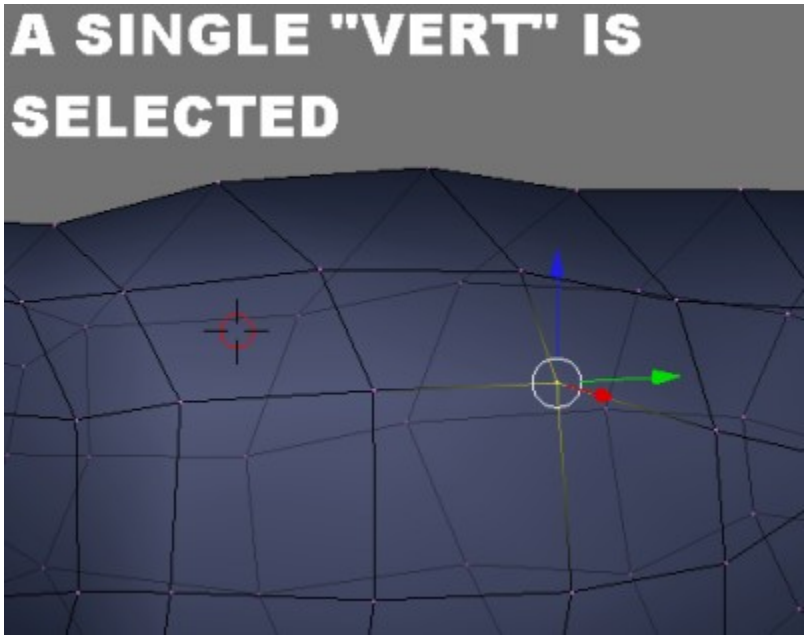
Importing a Movies Game costume will have things already set up in Blender for us. Objects will have *materials*. Costumes will have an *armature* (The Bones) which moves a costume around. A *blend group* for the *Empty* containing *child* objects.
Every 3D model has a structure we can edit in Blender. But to get at this structure is a matter of being in the right *Mode*. The object itself can be edited with Blender's basic tools. **S** key for *Scale* (to change the size of an object.) **G** to *Grab* or move the object. **R** to *Rotate* the object. These tools work in several modes. The default mode is **Object Mode**.



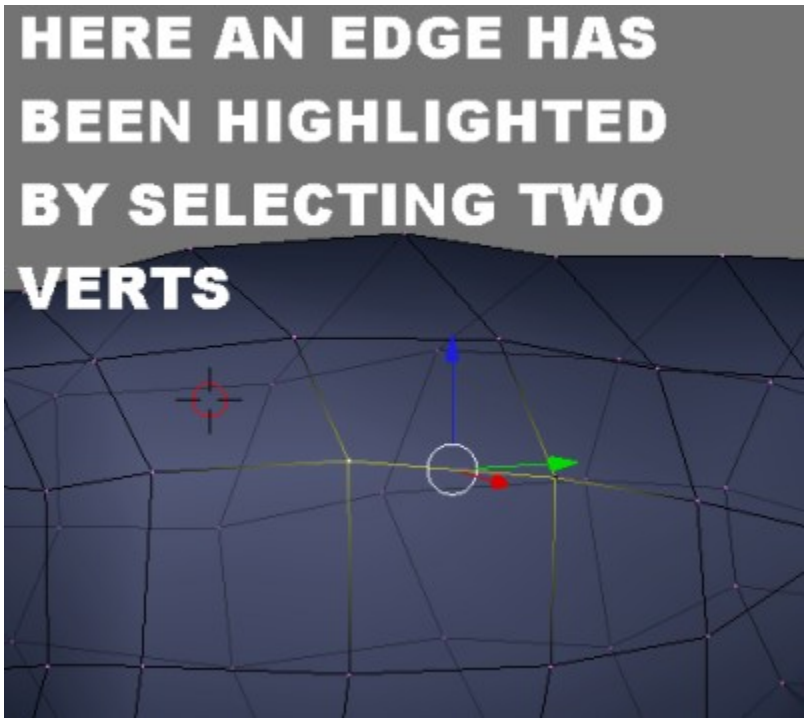
There is another mode called **Edit Mode**. In Edit Mode we see the true structure of the object or what defines the shape of the object.



Each shape is proportioned by points floating in 3D space. Each of these points is called a **Vert** or **vertice**. Commonly known as polygons.

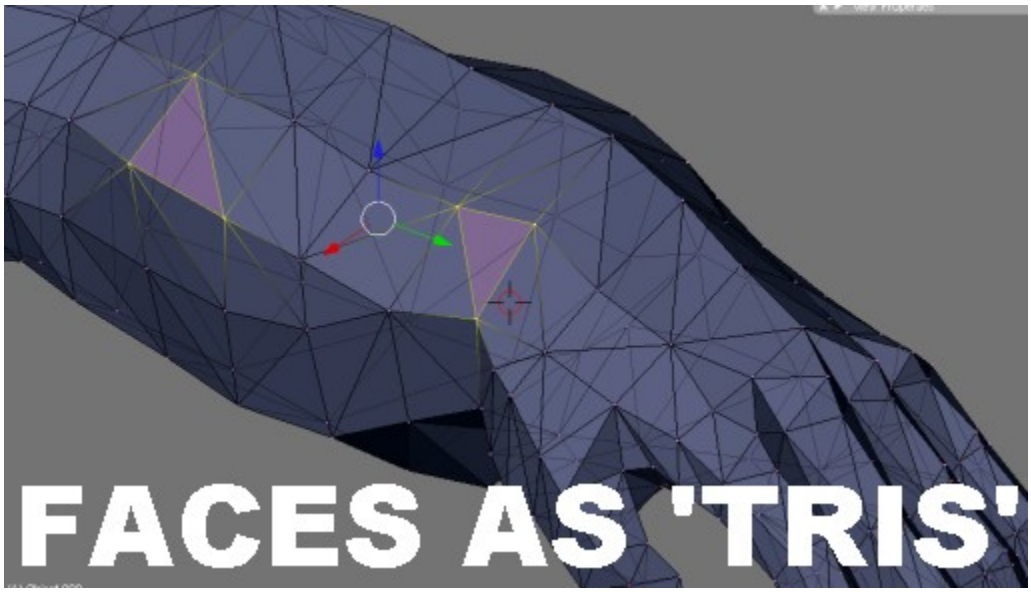


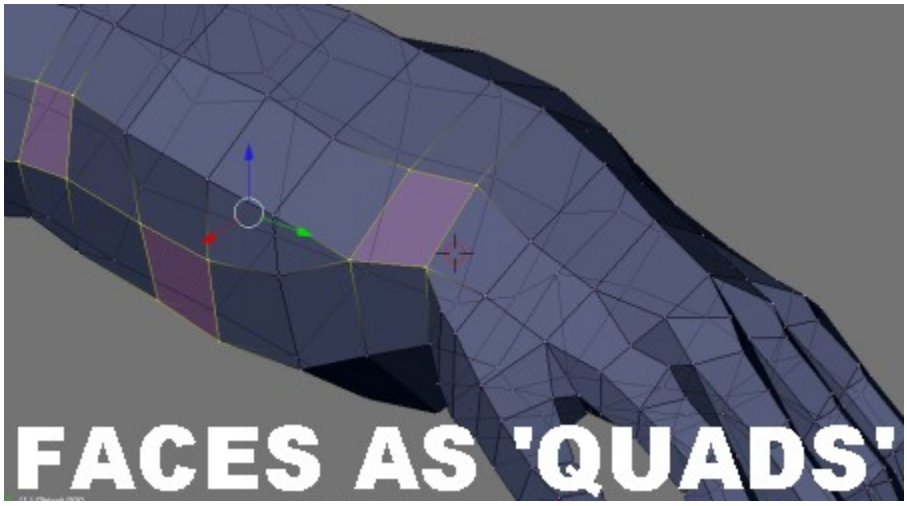
Each of the **verts** are connected to each other with an **Edge.**



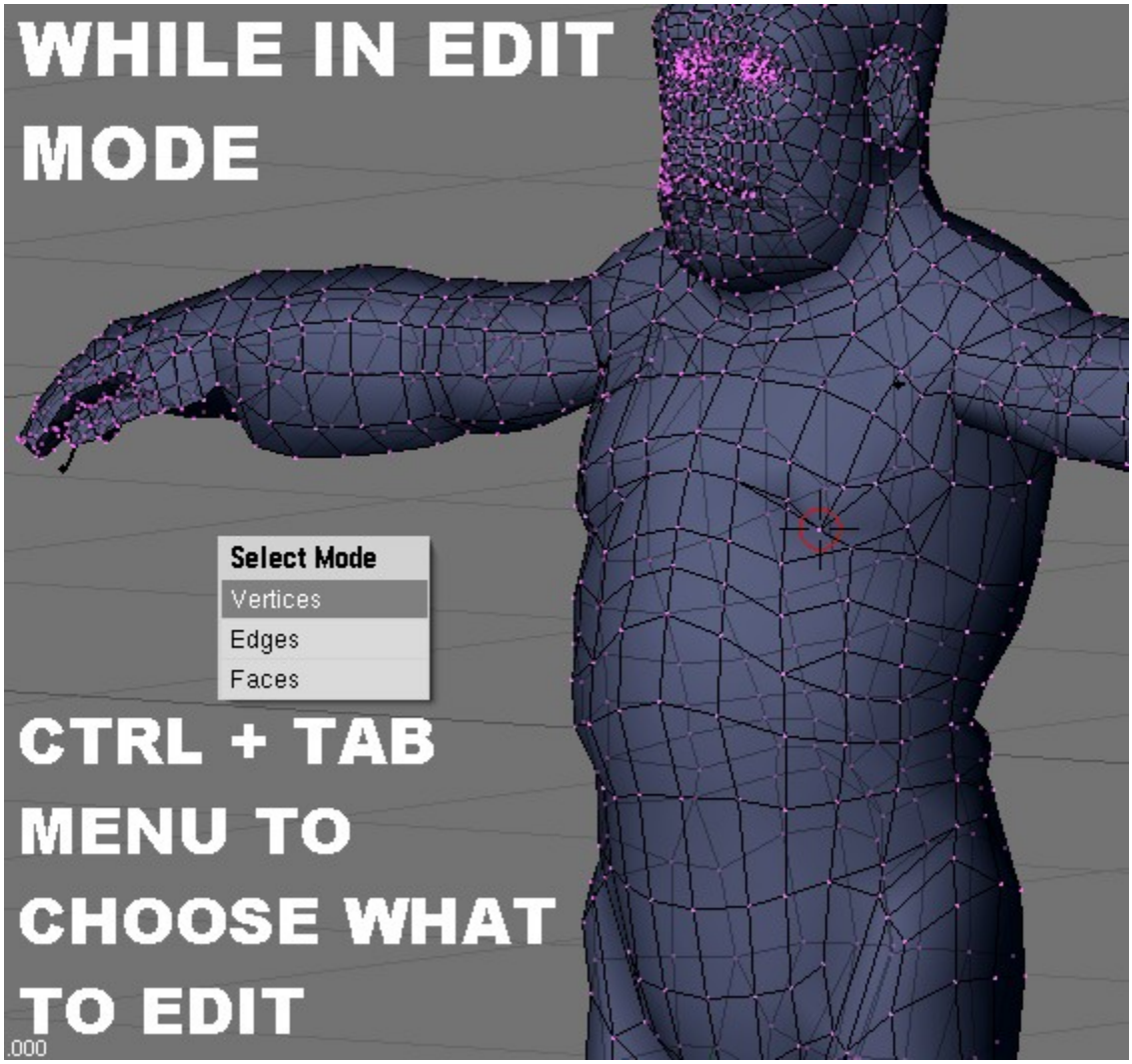
And 3 or 4 edges often form a **Face.** A face is either a *tri* or *quad* (triangle or square). The import script will break the model's faces into *tris*. On export the faces will be converted back into *quads*.

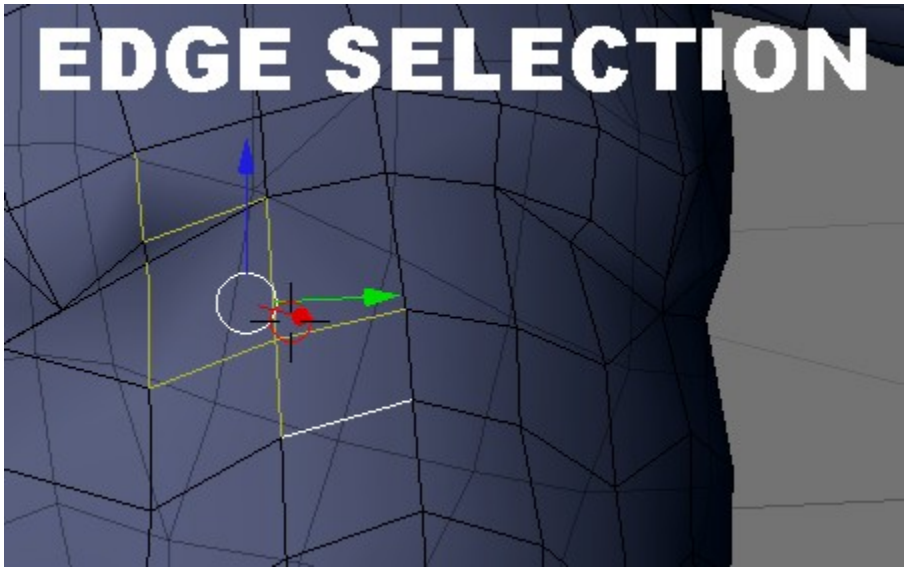

WHOLE FACE IS
LIT UP BY
SELECTING 4
VERTS

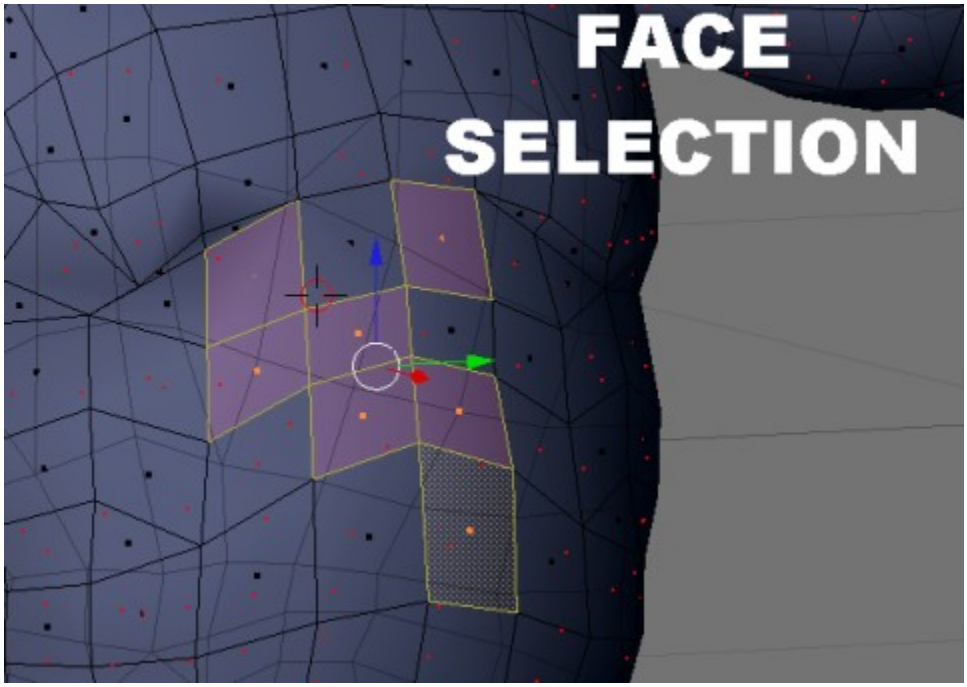

FACES AS 'TRIS'



Each of these are selectable either individually or several at once. You can select a single *vert*, or a single *edge* or a single *face* by clicking on it.
You can tell Blender which type you wish to select or edit by pressing **CTRL + TAB** keys. A menu will pop up and you can select from it either *vert, face* or *edge.*

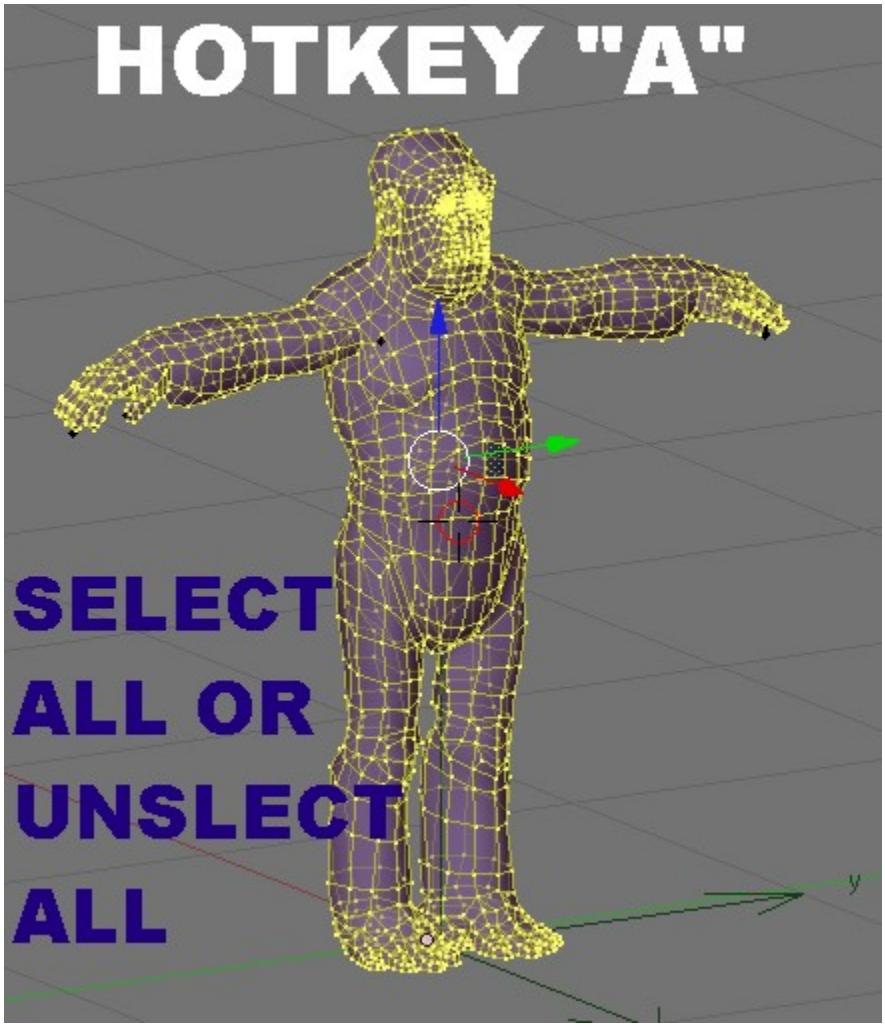

WHILE IN EDIT MODE
Select Mode
Vertices
Edges
Faces
CTRL + TAB MENU TO CHOOSE WHAT TO EDIT


EDGE SELECTION

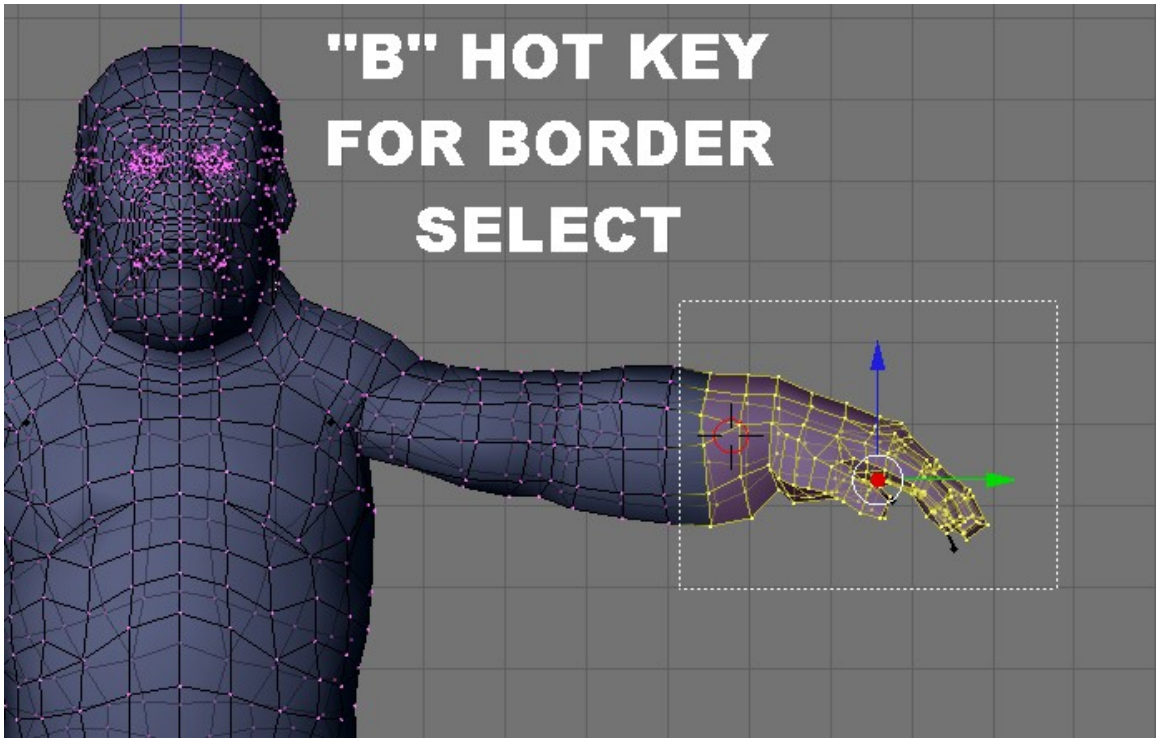


After you select a single *vert*, (or *edge* or *face*) you can select another one. Doing so will change which vert is selected at present. But if you hold down the **Shift** Key while clicking on another *vert*, the second one will also become selected without losing the first selected. Now two *verts* have been selected at once.

You could quickly deselect everything by pressing **A** key**. A** Key either selects All or deselects ALL.



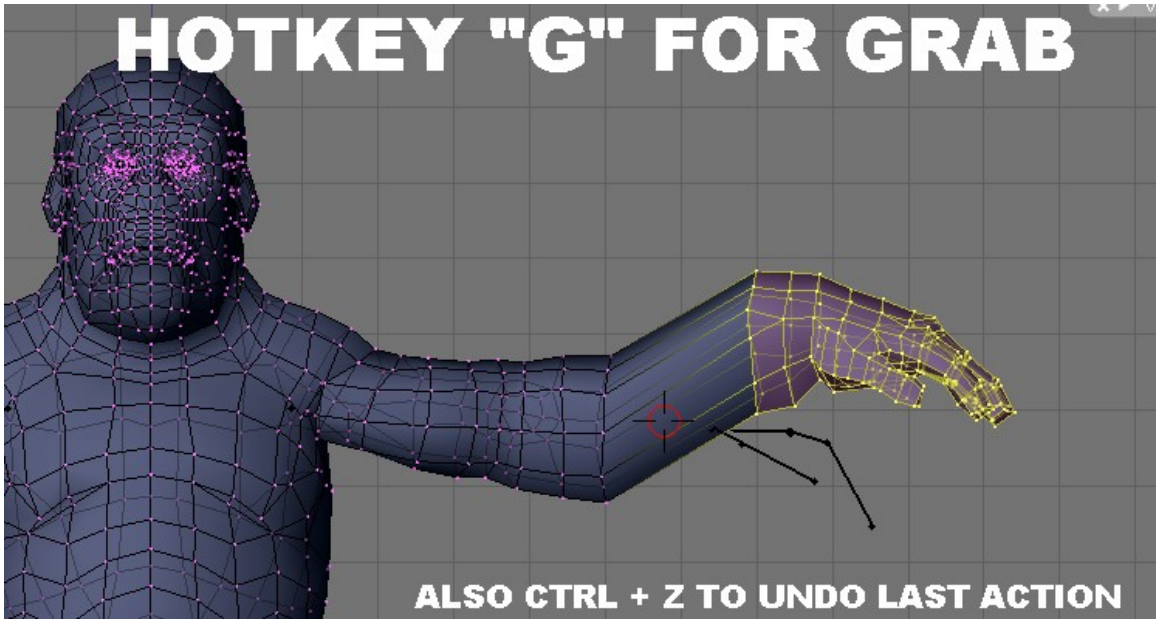
There is a tool for selecting. By pressing the **B** key a box will show up. Actually you have to make the box. Press **B** and click hold the left mouse button somewhere on the screen. Holding down left click, drag/draw a box over a select number of *verts.* When you let go of the mouse button whatever is in the box will suddenly become selected.



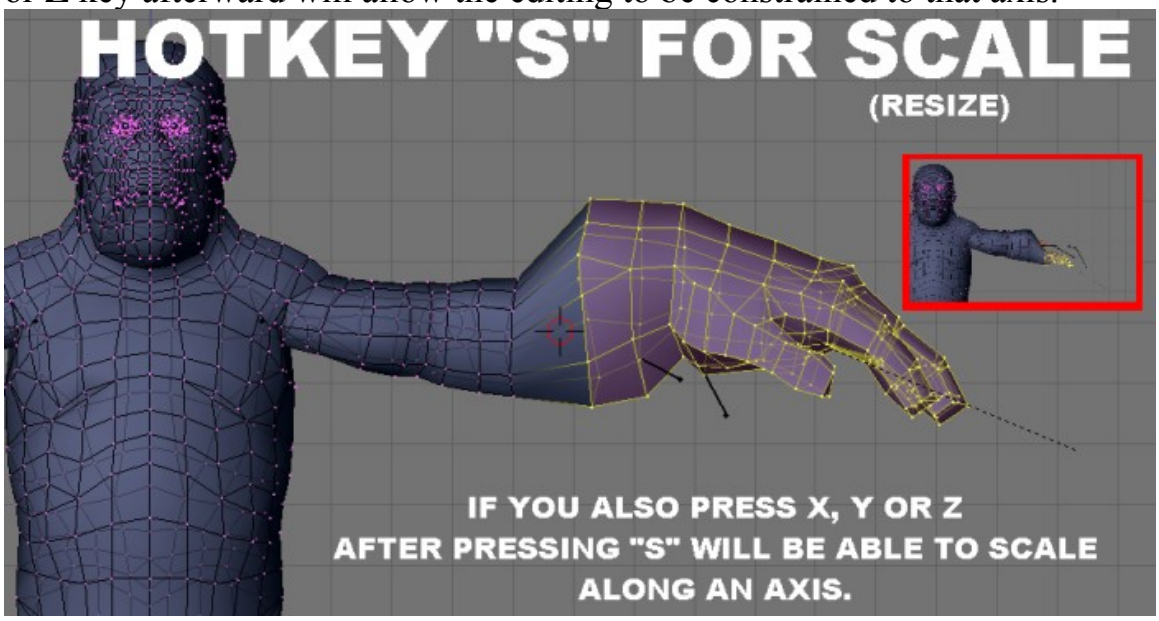
If you press **B** key twice in a row you get Blender's second selection tool. The circle. The circle stays active. From here you can move the circle over any particular area. Left click anywhere to make whatever is in the circle to become selected. Press right mouse button to end the circle select function. Or hit the **ESC** key to end it. Also, if you spin the mouse wheel the size of the circle increases or decreases.

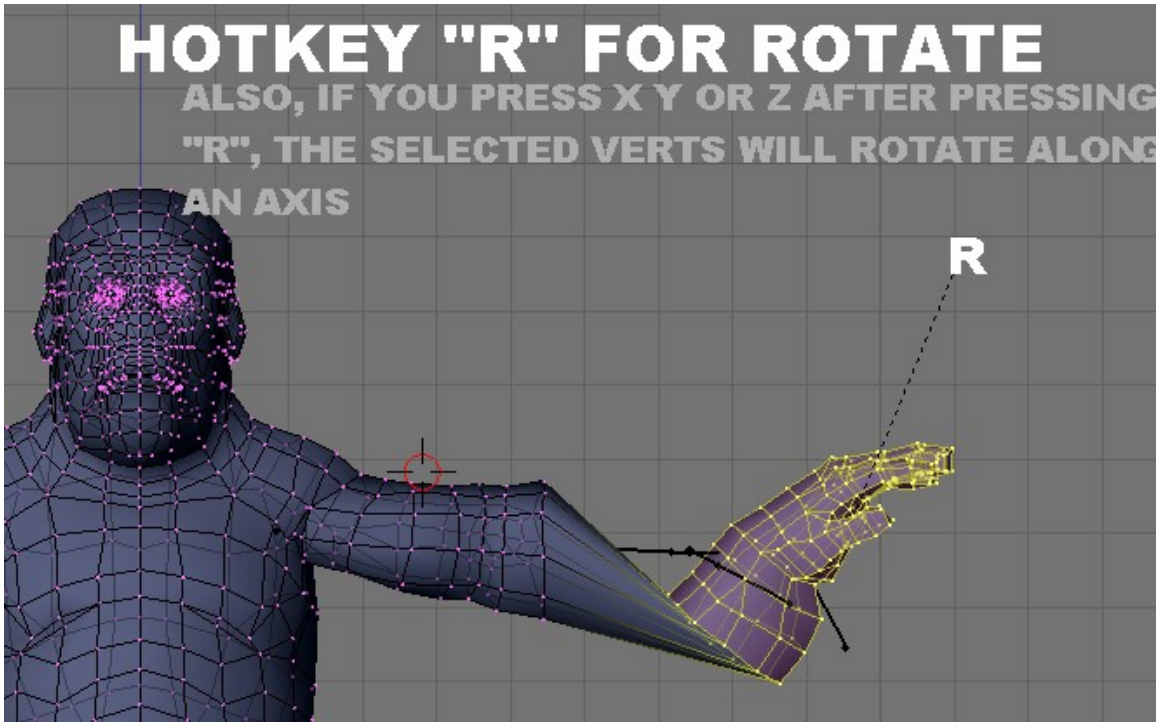
The same selection tools work in reverse. For the circle, middle mouse button deselects *verts*. And using the right mouse button to drag/draw the square will deselect stuff found in its borders.

Once the *verts* have been selected, you can move only those *verts* by hitting the **G** key for *grab*. The left mouse button will un-grab the *verts* and release them at a new location. (RMB to let go and return to previous state.) You can also move them along an axis. X Y or Z axis. If you want to move some *verts* in only an upward direction, you can, after hitting **G** for grab, press the **Z** key (which is for Z axis 'up and down'). Wherever the *verts* move from here will be an up and down motion.



You can move or edit selected items in other ways. You can change the size of an object with Hotkey **S** for *scale*. Or you could rotate the object or selected items with Hotkey **R** for *rotate*. And as with *Grab*, if hitting the **X Y** or **Z** key afterward will allow the editing to be constrained to that axis.





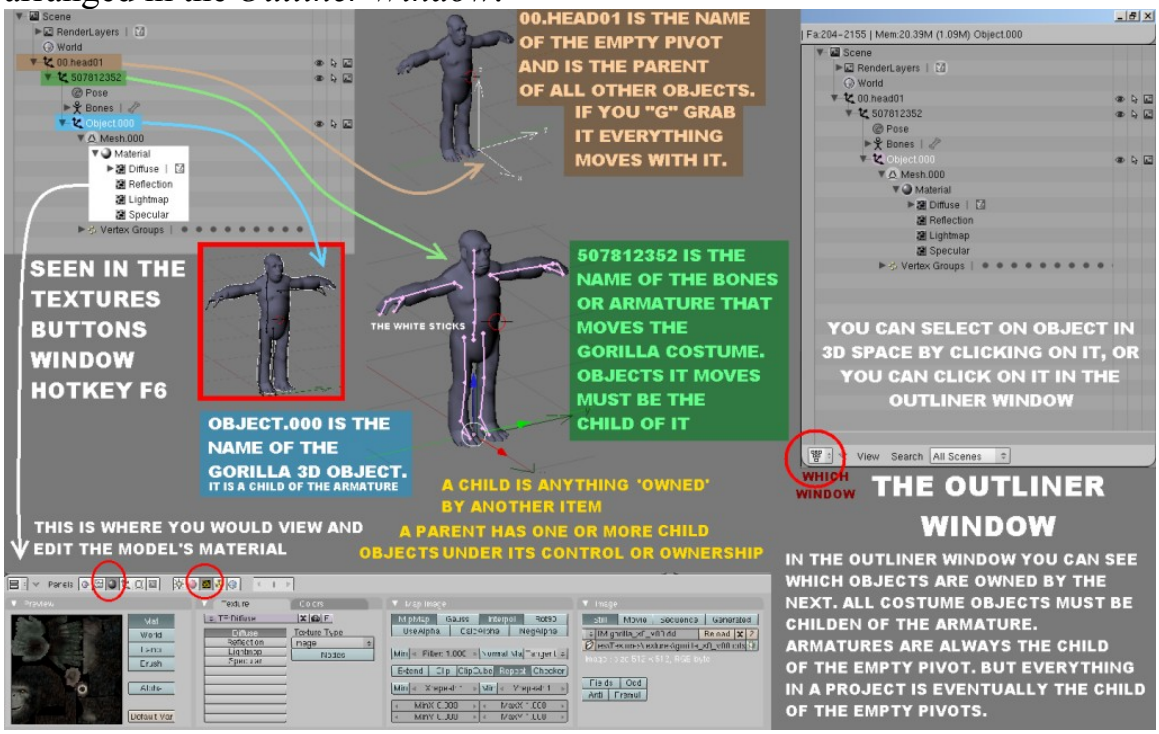
Now sometimes you have a *vert* selected but it seemed to be more then one located at exactly the same spot. When any Movies Game object is imported, each object is solid only by which part is sharing the *uvmap*. You can click one *vert* and then press **CTRL + L** key to have everything matching the *UVMap* to become selected at once. If you wanted each version of the vert to move at the same time, you can use the selection tool **B** key to have it all selected. Or you can remove all double cases of verts. There is a button called **Remove All Doubles**. You must be careful before pressing it. Right next to the button is a box with numbers in it. These numbers tell each Blender tool how strong it should be. Blender's default is at **0.001** (which is crazy) *Verts* near by will get joined together. This is to be avoided unless you need that specifically. Click on the numbers and enter **0.000**. Then you can hit the **remove doubles** button and every *vert* right on top of another will get merged into one *vert*.

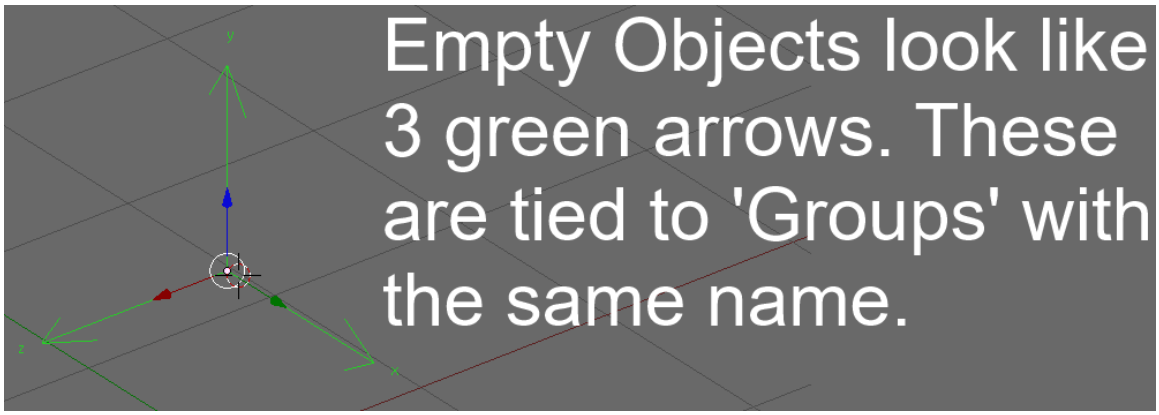
Put merging doubles off until you need to. It is handy to have the object broken up into selectable segments. However, you may find after *smoothing* verts, some have come apart. The doubles were not removed and that is why. Press **CTRL + Z** until the smoothing is undone. Then remove doubles, then smooth them again.

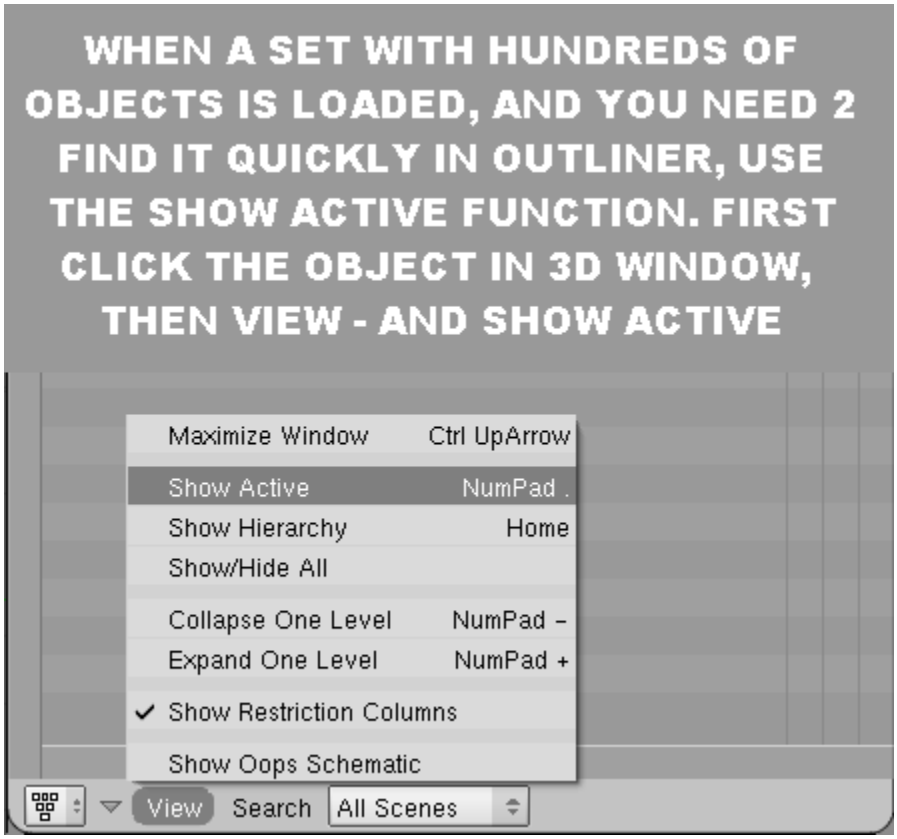
The selection tools work in both *Object* or *Edit modes*. In *Object* mode, you can select one object at a time, or have them selected at the same time by

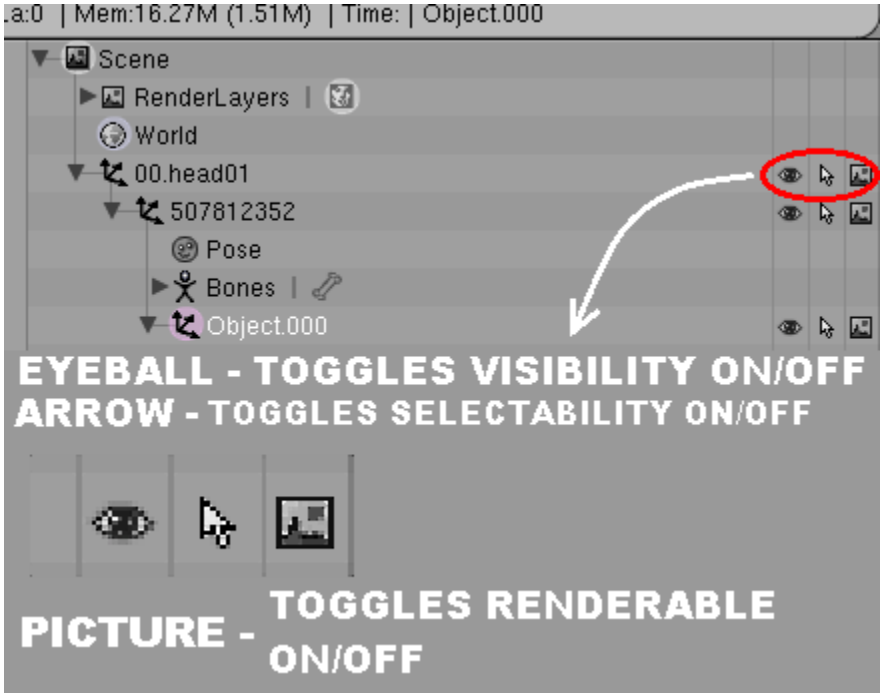
holding down **shift** key and then clicking on another object. Then two or more objects can be selected at the same time.

When making Movies Game mods in Blender, you will need to know how they are set up in the *Outliner* window. Much of the selecting and renaming of objects happens in the *Outliner* window. To rename an object in the *Outliner* hold down **CTRL** button and then click on the name. *Empties* and Objects will need numbers in the name. The best way to see what goes on is to import a Movies Game item into Blender and see for yourself the items arranged in the *Outliner Window*.









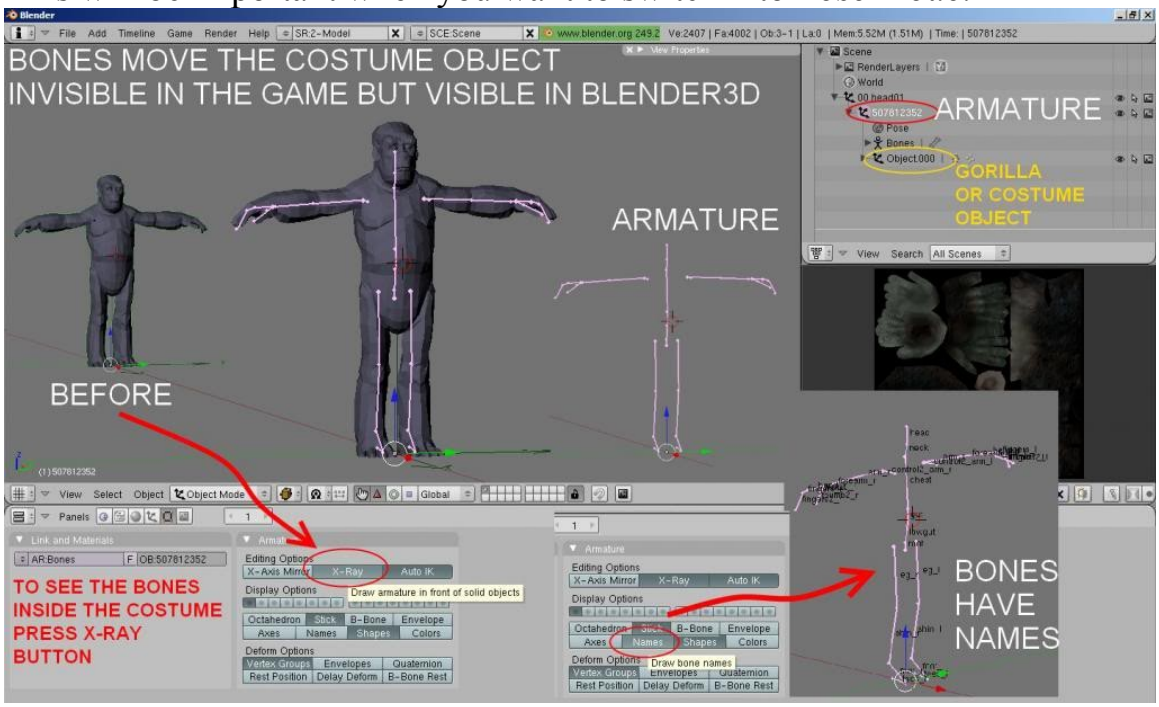
Icons at far right in the Outliner Window:

- **Eyeball Icon:** Turns On/Off visibility of the Objects (or Empties or armatures) in the 3D Window.

- **Arrow Icon:** Turns Off Selectable in 3D Window.
- **Picture icon:** Turns Off Rendering of Object in a scene or in *Baking*, perhaps from producing shadows also.

Visibility allows you to hide the object in the 3D window. Clicking on the eyeball icon will hide or reveal it. You can also turn off it's selectivity. In this way *border select* or Hotkey **A** will ignore the object in 3D space. The picture icon is a button that will tell *Cameras* to ignore the Object when rendering a scene.

Here is the *Armature* of the gorilla costume. It is found inside the object. Normally it is hard to see. Only a bit of the finger *bones* will poke out of the hands. We can see the armature inside the gorilla if we hit the **X-Ray** button. This will be important when you want to switch into *Pose Mode*.

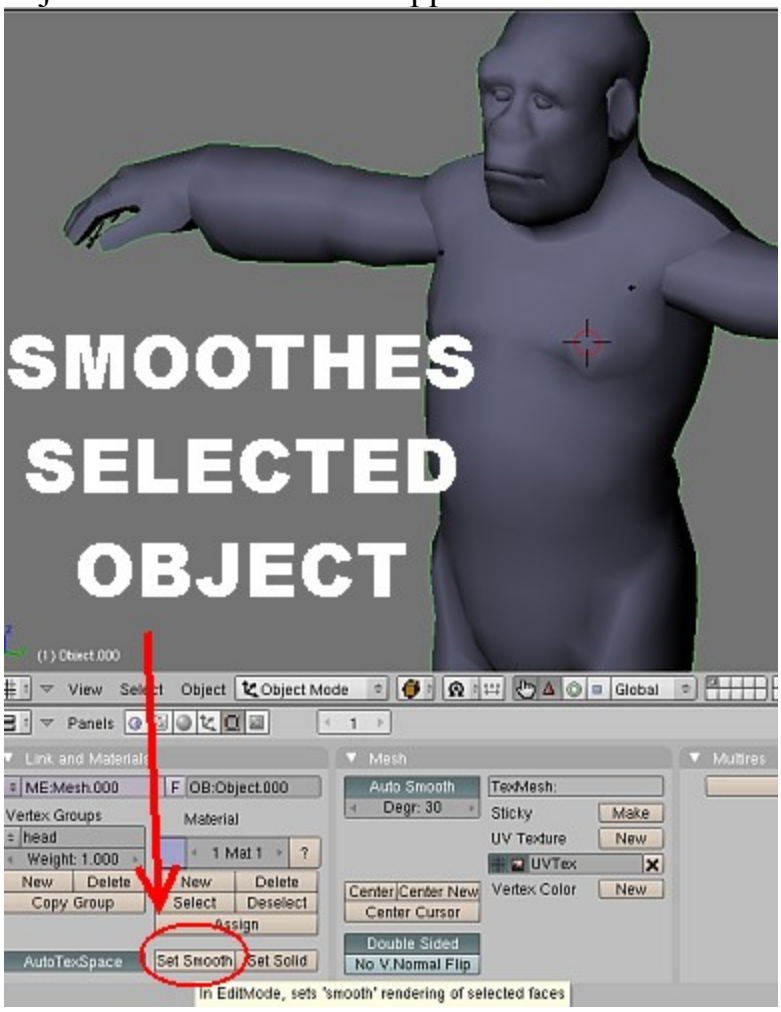


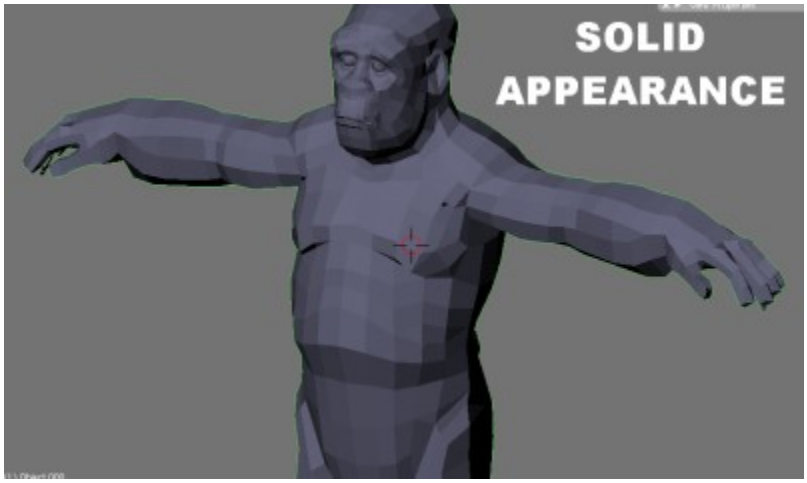
Each item can have it's visibility turned off so we can edit other smaller objects. Sometimes stuff is just in the way. The *Empties* (3 green arrows) sometimes gets in the way and I turn those off from visibility. When something isn't visible it can no longer become selected in the 3D window. Usually I turn visibility on or off by clicking on the eyeball icon next to the object in the Outliner window. On the other hand you can press the **H** Key for *hide*. You can press the **ALT + H** to make all hidden items visible again.

In *Edit Mode* we can make selected verts become hidden also by hitting the **H** Key. **Alt + H** restores the hidden verts. It is easy to accidentally hit the wrong key in Blender. If you hit the **H** key while in *Edit Mode*, the selected verts disappear. Then you are left wondering what happened. You can

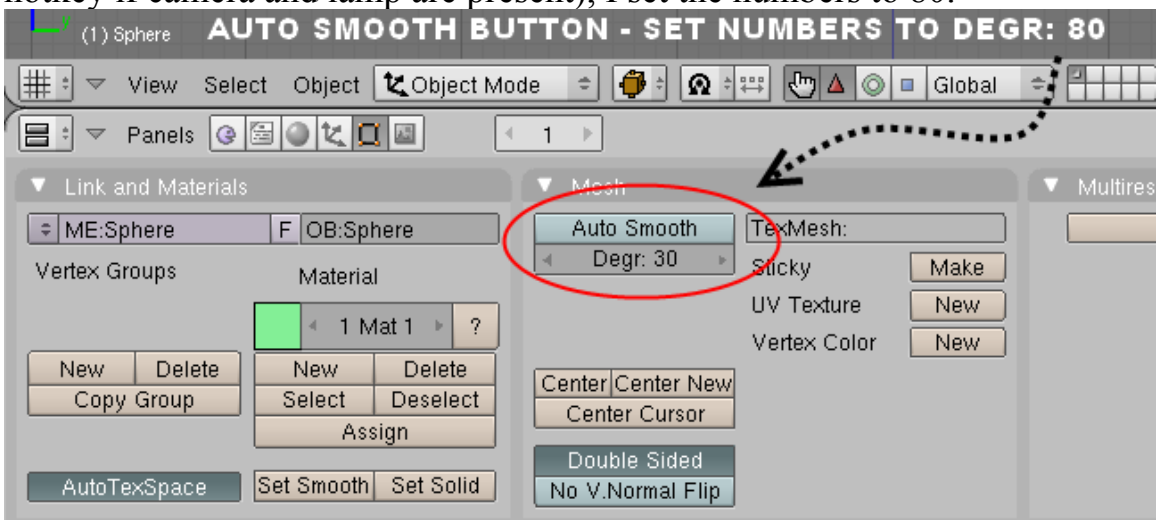
always hit **CTRL+Z** to undo the last action, and this would return the verts to visibility. Or just hit **ALT+H** to *un-hide* those verts. *Edit mode* is tricky. The *Outliner* will always show if an object still exist, even if it's visibility is turned off. But in edit mode, if you accidentally turned off visibility of some verts, the *Outliner* window won't tell you more verts are there. Going back into Object Mode (**<TAB>** Key) you can see the shape of the object still knows those verts are there. If in doubt, use **ALT+H** to make sure all verts are visible.

In the *3D* window objects can have more then one appearance. But it's appearance may be different then an actual rendered scene or *baked* texture. Other buttons can help with this. There are two buttons that will change an object from a hard or soft appearance. **Set Smooth** and **Set Solid**.





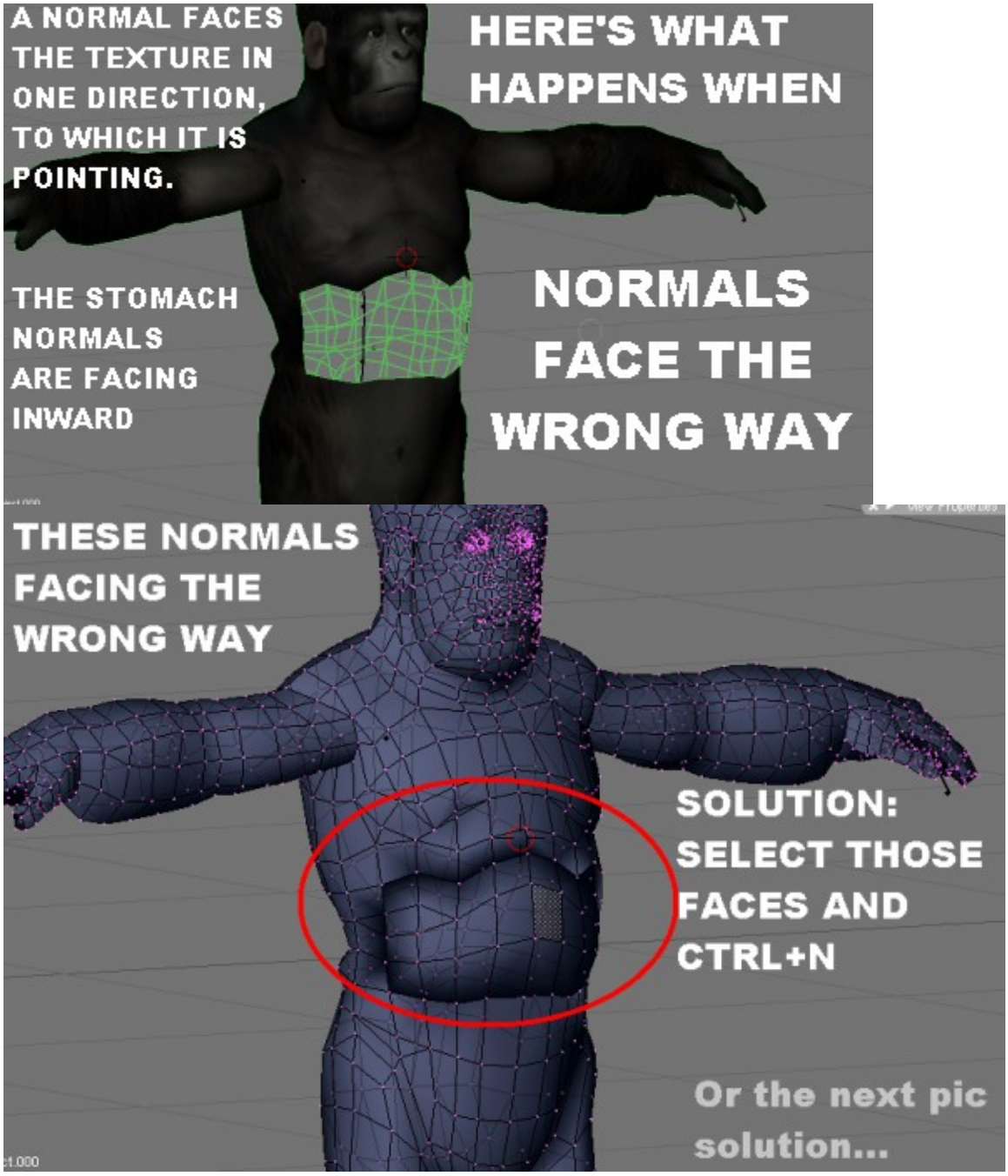
To get the *render* or the *Baking* to reflect smoothness the next panel over has a button called Auto-Smooth. I do not press it. However I do change some numbers. The numbers just below it decides the strength of smoothness. I believe the max is **80**. 0 for no smoothness. When I need a smoother appearance for a texture bake, or even when I render the scene (exp: **F12** hotkey if camera and lamp are present), I set the numbers to 80.

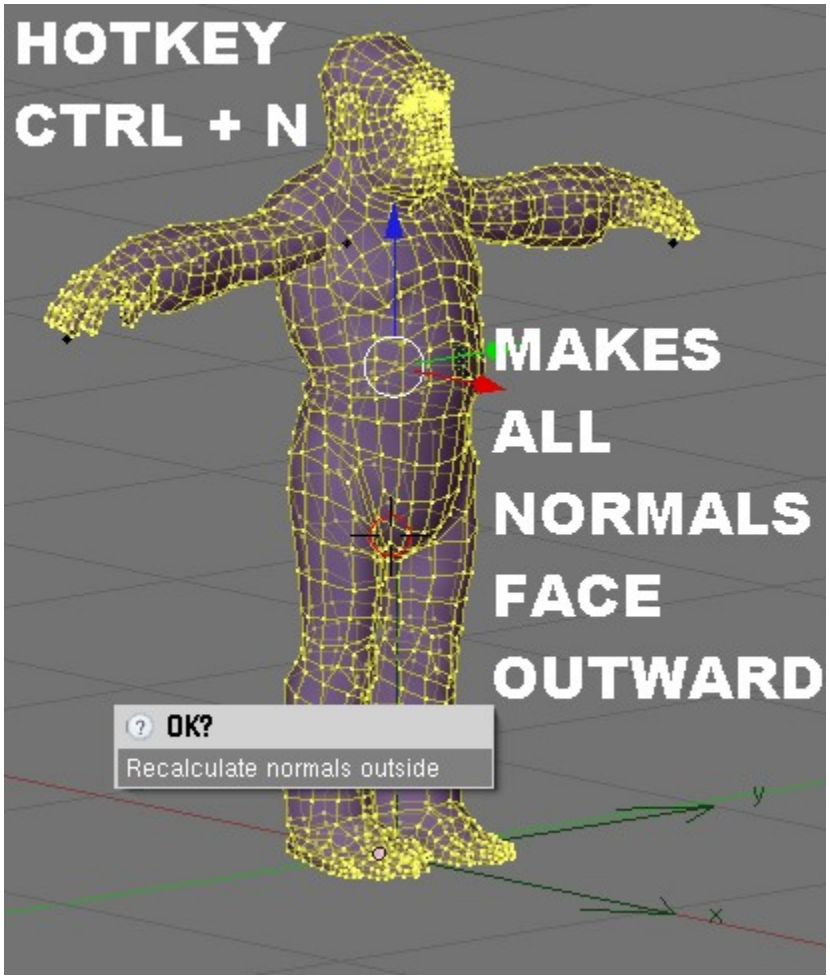


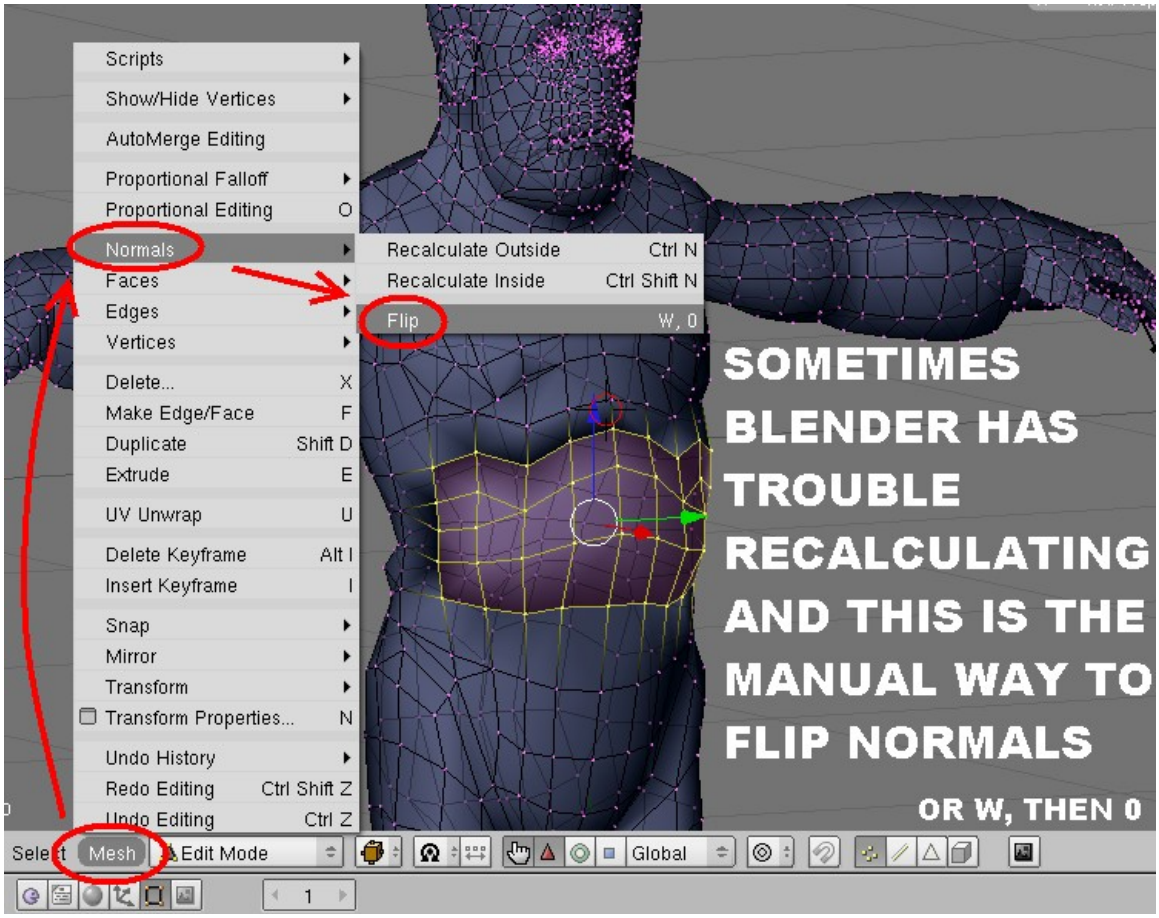
The Movies Game has its own Auto-Smooth during game play. This is to save memory, disk space and CPU draw. Everything does not need thousands of verts to achieve a smoothness. DAZ models have lots and lots of verts to give their models smoothness, but such models, although they work in The Movies, is an unnecessary draw on the computer's performance.

Sometimes a texture won't show up? Or the model has a darkening thing going on? **Normals** are in the wrong direction. **Normals** tell the program which way the *face* is pointed. A bad cheat is to have faces *double sided*, or to have a face show the image on both of its sides. But this is bad. Double siding a face should be used to make walls visible on both sides. But not as a

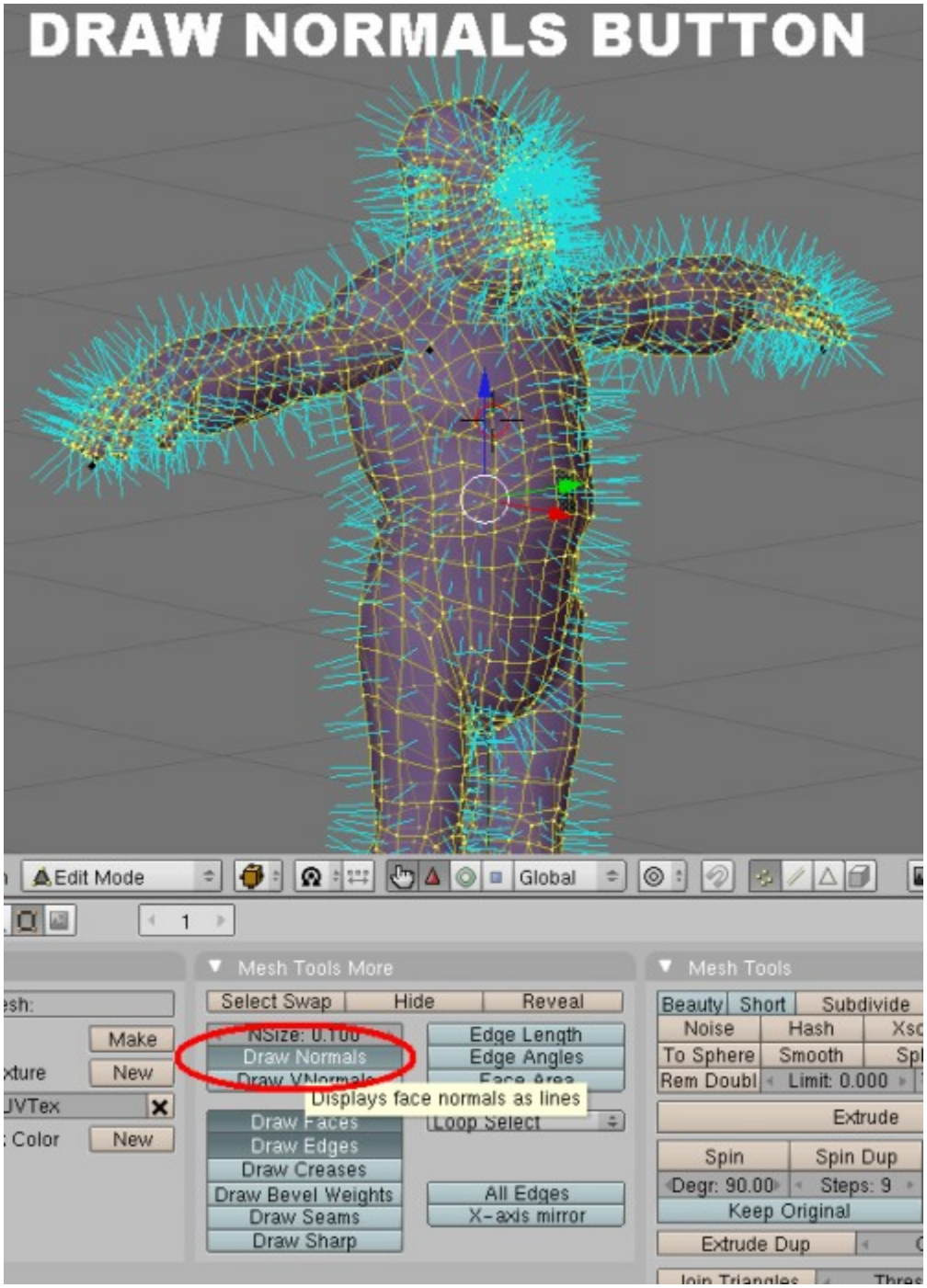
fix.


A NORMAL FACES THE TEXTURE IN ONE DIRECTION, TO WHICH IT IS POINTING.
HERE'S WHAT HAPPENS WHEN
THE STOMACH NORMALS ARE FACING INWARD
NORMALS FACE THE WRONG WAY
THESE NORMALS FACING THE WRONG WAY
SOLUTION: SELECT THOSE FACES AND CTRL+N
Or the next pic solution...


HOTKEY
CTRL + N
MAKES
ALL
NORMALS
FACE
OUTWARD
OK?
Recalculate normals outside
y
x

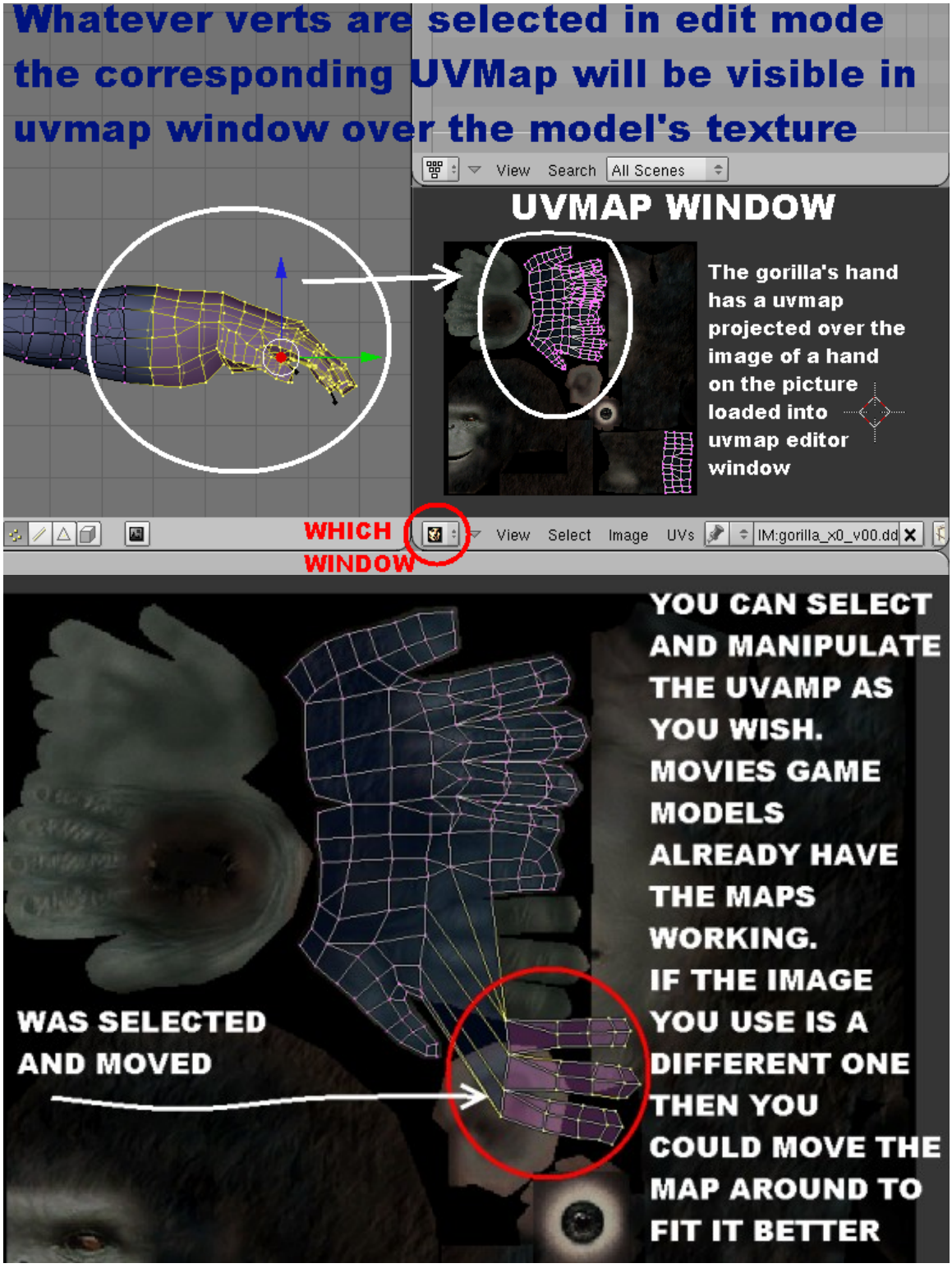


You can see *Normals* in *3D* window with a button. I rarely use it and it's not needed to manipulate or edit them. But you can see them with a click of a button.



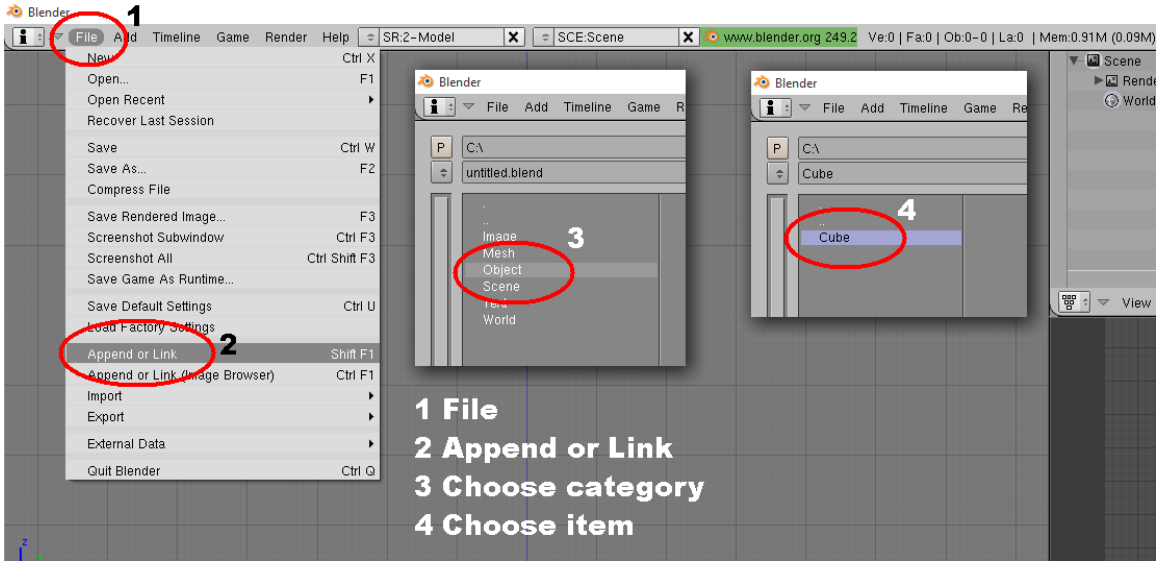
Your 3D models use images to give an outward appearance. These image files are in (.DDS) format. If a mesh is imported after being extracted by MED, it should also import the texture it uses. This depends on if the (.MSH) file was in the data\mashes folder, and the image was in the data\ textures folder. Moving the files will stop the image import. You will see the image in the UVMap window. Or with this window you can simply load in another image. Also, when you go into *Edit Mode*, and then select some of

the model's faces, the *uvmap* of that face will appear in the *UVMap* window. The map can be moved around. Or repositioned. Later I will describe how to make new uvmaps or more then one uvmap... and how to make one from scratch.



Much of what you need to know will be **what conditions must exist to allow a model to export into the Movies Game.** The export script will attempt to do the job regardless of how ready the project is. Fortunately we have the **Preflight Script** which will go over your project and find what is wrong or still "needs to be". The script is run from the *Scripts* window. Press the **scripts** button>**Misc**>**The Movies Preflight Check.** Wait a few seconds and a browser window opens showing a list of checks. Each shows green if it passes. If not it will show you an error. A **detail** link will reveal what item is in the wrong.

It is possible to retrieve any item from one blender project to another. This is called *Appending*. There are several categories to append from. You could append a material that already has the alpha setting. Or any object models. You could append entire groups.



This tutorial will cover much of the above tools with plenty of pictures. The hard part is learning what Blender can do and why. A good way to learn is read the list of Hotkey Functions in the Appendix. Once you become familiar the work becomes fast and fun.

3. Importing A Movies Game 3D Model Into Blender 3D

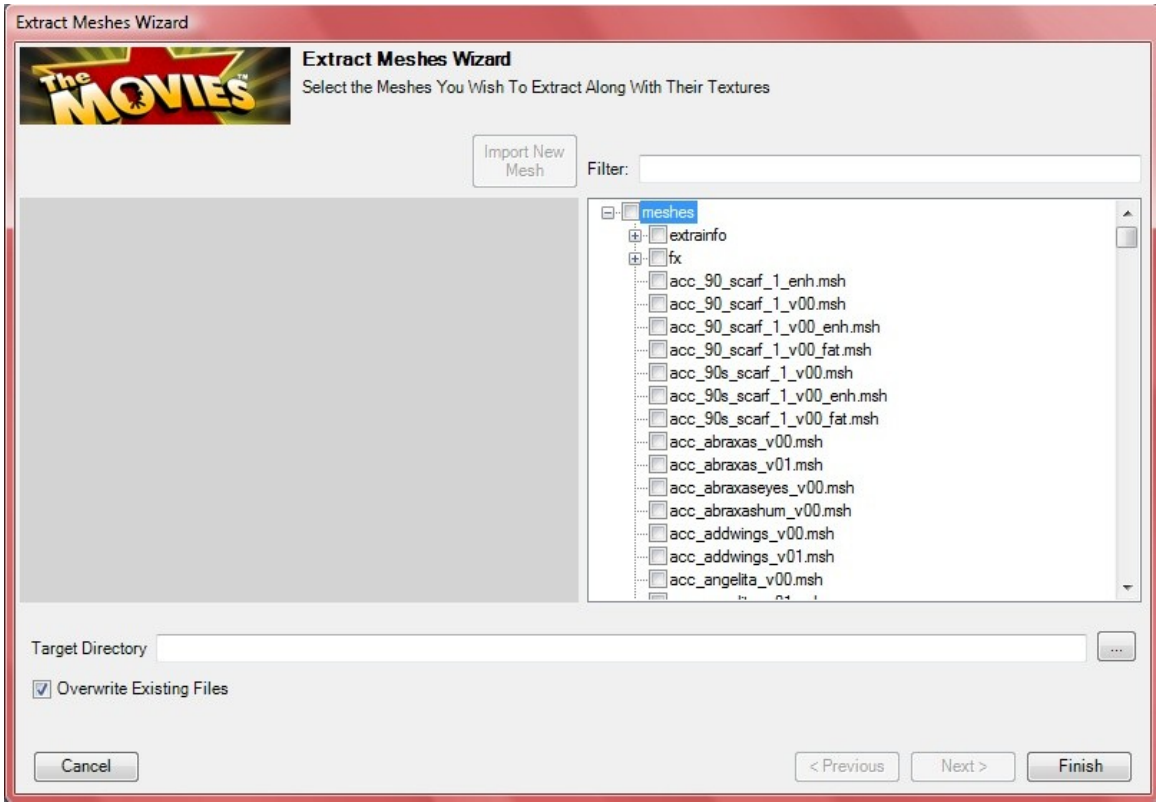
Once you install The Movies 2009 (.MSH) Scripts the next thing is to start using them. The following tutorial describes how to extract a game model using MED and import it into Blender. The tutorial is specific to The Movies, but it is the step that most modders for other games would use as well. Namely you would extract a game model from the game, then use an import script to import it into Blender for a conversion. Your other game would need an import script specifically designed for your game's models. Which won't be the one used in this tutorial. Fortunately for Movies game modders an import script was released by DCModding.
If you have models from a different game the best advice for finding a Blender import script is to first know your game's model file tag. For The Movies it's (.MSH). In this case I would Google "Blender import script .msh" or "Blender Import script The Movies". You may have to do some searching and asking in forums as well.

Step 1: Extract A Model

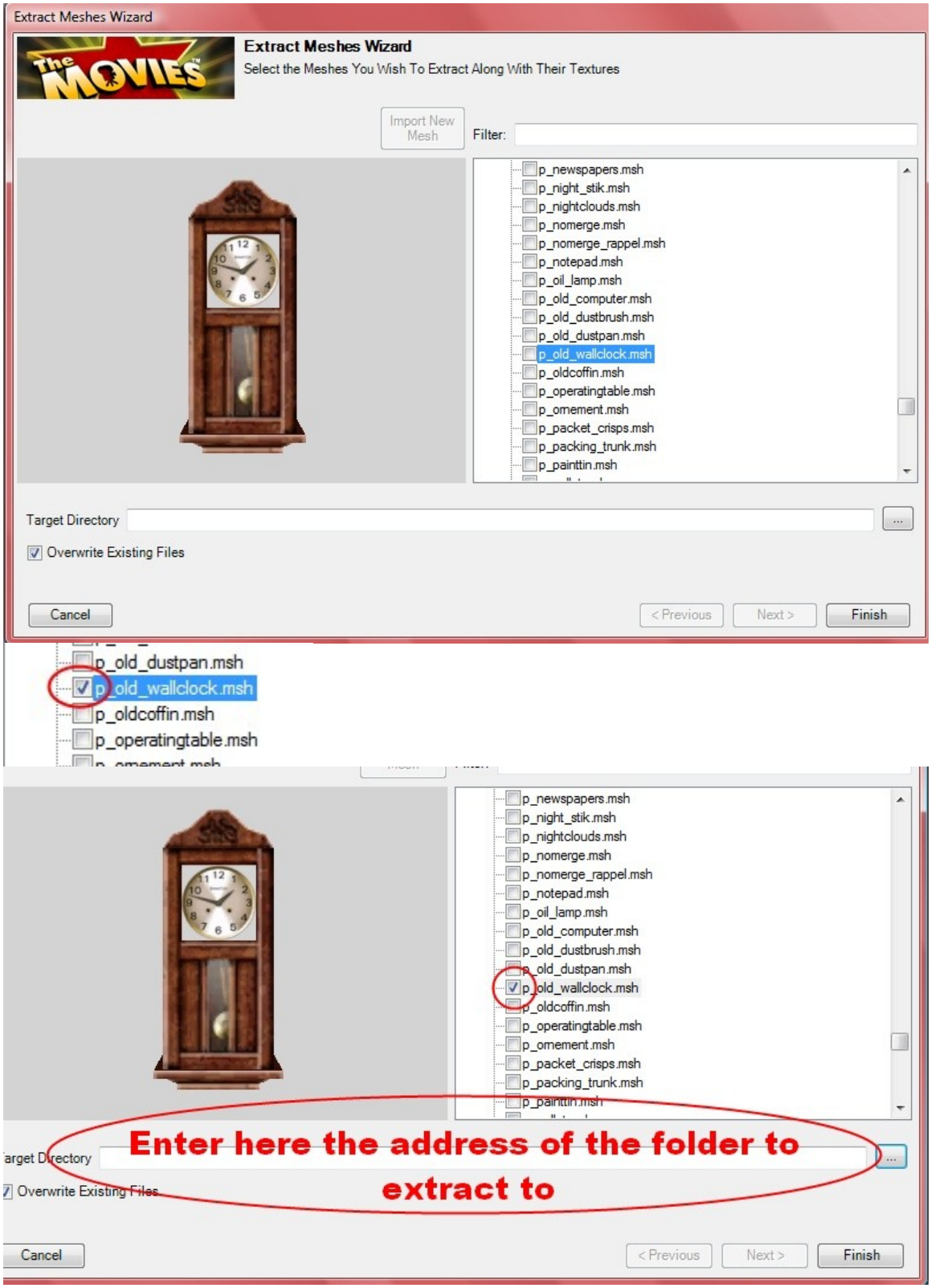
You would get a model by extracting it from the game's compressed files. The first tool was Reshoot. PakPoker was the next tool. Now we have the amazing **MED - The Movies Game Editor** which can extract any set, prop, or costume. Open MED and wait until it analyses your game. When the main window opens you will be at the *Action Wizard*. At the right side is a button called **Extract Mesh For External Editing**. Mesh files **(.MSH)** are the 3D models we import into Blender.

Note: Once you are making regular mods for the game, you may want to extract every game model and keep those (.MSH) files in a separate folder. And from that folder you can import any game model. For me it became handy to do so, making entire texture and mesh libraries neatly organized. This because I actually couldn't extract the download version of Stunts&Effects content with MED. Had to have a friend give me those libraries. However, when I finally did have a normal version of S&E, I would still extract the entire mesh and texture folders. A huge time saver and Window's search function is a lot more handy then MED's search function. I think Machinimods.com had this download before they closed.

Another window will pop up called **The Extract Meshes Wizard**. There is an empty 3D viewing window on the left. And a list of meshes on the right. You have to expand the list by clicking on the "+" symbol to the left of the name meshes. To the very right is a scroll bar. If you click on any file name, you will see it loaded in the 3D viewer.



There is a box next to each name. Click on the box for the prop **p_old_wallclock.msh.** Below is a bar to enter the address of the folder you wish to extract this wall clock to. A little button is on the right side of it. Sometimes clicking that button will not open the folder browser which is a bug. Instead I copy the location of the folder from the address bar and paste it in the *Target Directory* bar at the bottom of the window.



If you forget to click the little box next to the name, clicking the finish button will make the window disappear but without exporting the model.

If you did click the little box then a green progress bar will pop-up showing the extracting is taking place. After that bar goes away, the model is extracted. Navigate in a separate window to the location of the model before you close MED. Verify that an error didn't occur or that you forgot to click the button. You will see a **Data** folder there. If you see that folder then go ahead and close MED.

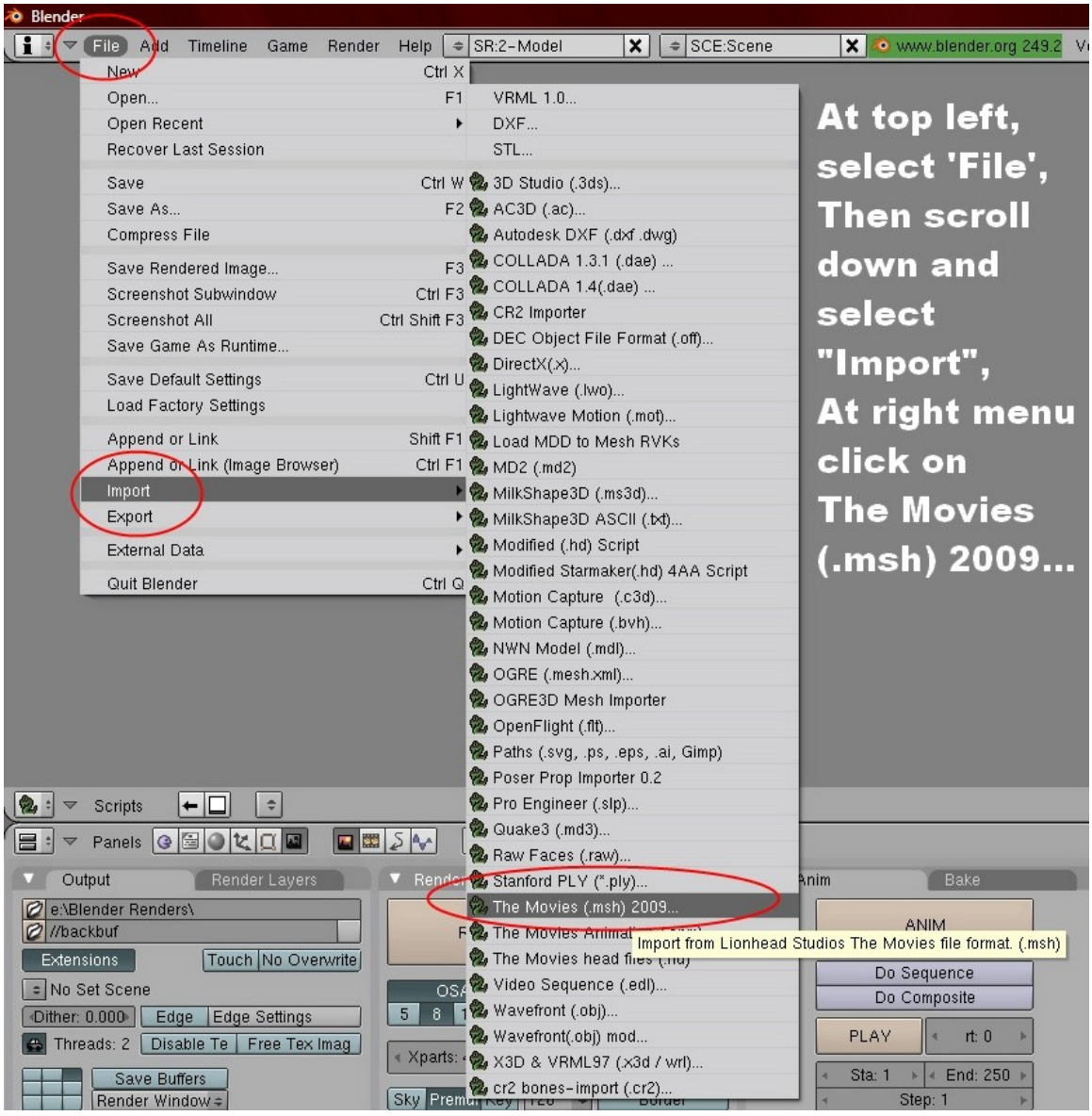
Within the **Data** folder is two more subfolders called **Meshes** and **Textures**. Since we extracted a prop, there is even another subfolder in the textures called **Props**. The **Meshes** folder contains the 3D object and the **Textures** contain the image that wrap around the 3d model.

So in all we have extracted two files.
The 3D model: **p_old_wallclock.msh**,
and the image file: **p_oldclock.dds**.
Both of these will get loaded into Blender using the import script. But make sure you do not move them from the sub-folders. Keep the texture image (.DDS) inside the data\textures\props folder and keep the (.MSH) file in the data\meshes folder. You can, however, move the Data folder that is carrying these files and subfolders. You can relocate the data folder to somewhere else before loading Blender if you choose to.

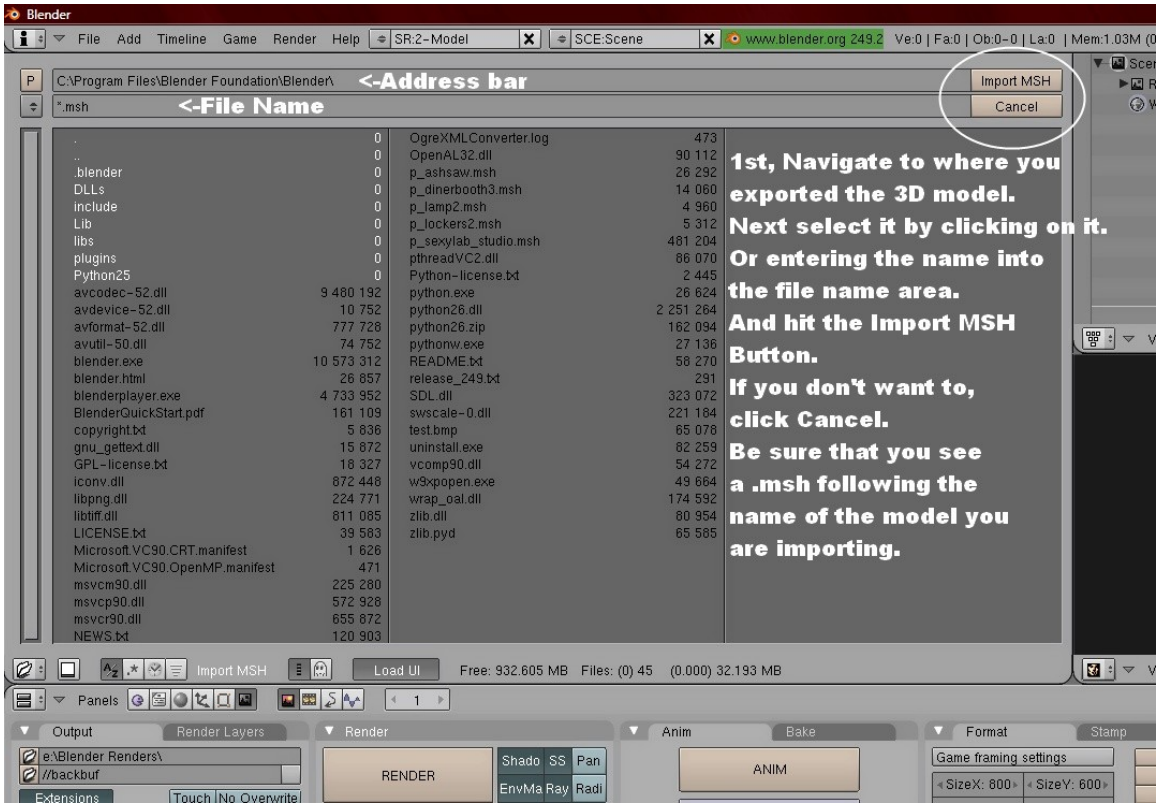
p_oldclock.dds

Step 2: Importing the Model Into Blender

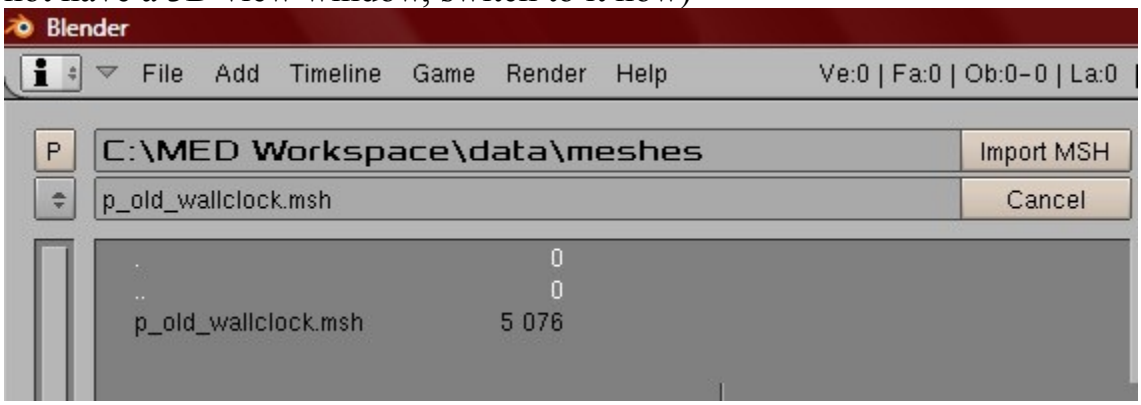
Open Blender3D 2.49b. Once Blender opens, go up to the top left and click on **File.** A drop down menu appears. Select **Import.** Another menu appears. From this menu select **The Movies (.msh) 2009...** Probably the name continues on to say import script but the name was cut short by...

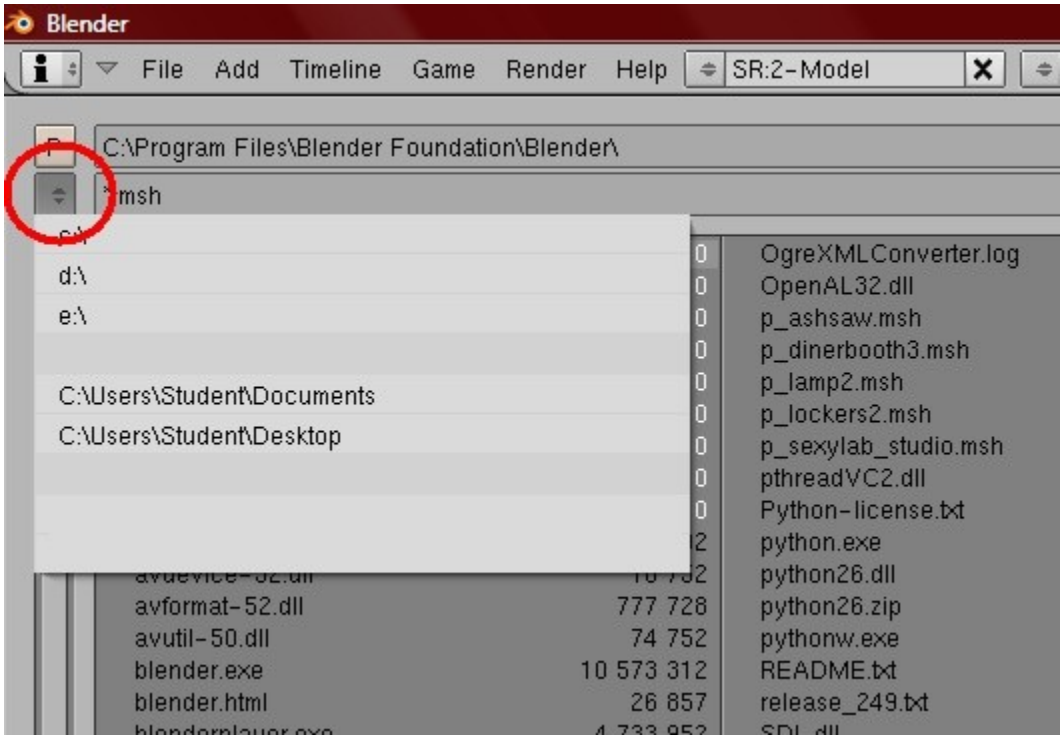


Now one of your Blender windows has turned into a browser window. Which window has always irked me. It was always random and sometimes a very small window turned into the browser window. You can expand any window to full screen by hovering the mouse within that window and using hotkey **CTRL + Up Arrow**. To minimize the window back to normal size use the same CTRL + Up Arrow Key. I have found a way to force the largest window to always become the browser window. For me the largest window is the "3D View". But if I change it real fast to *Scripts* window then use the import script, the large window becomes the browser window. When saving the *Default Settings* of Blender, I keep this window the *Scripts* window. And after importing the model switch back to 3D view.

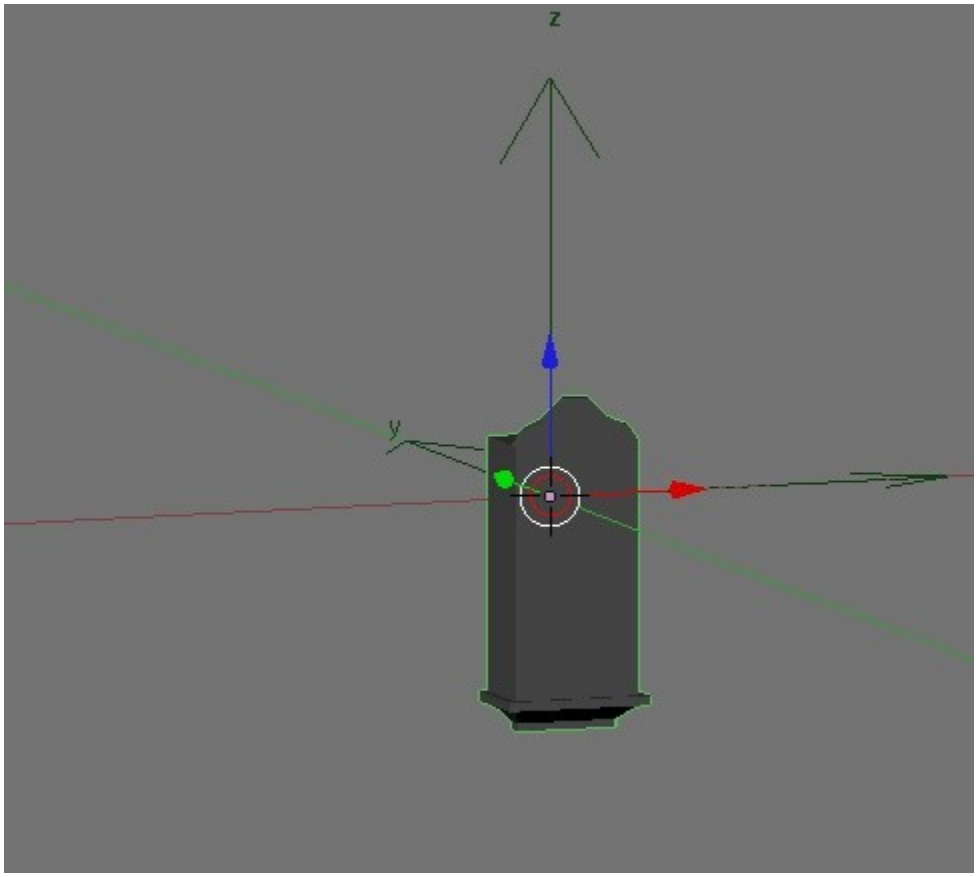


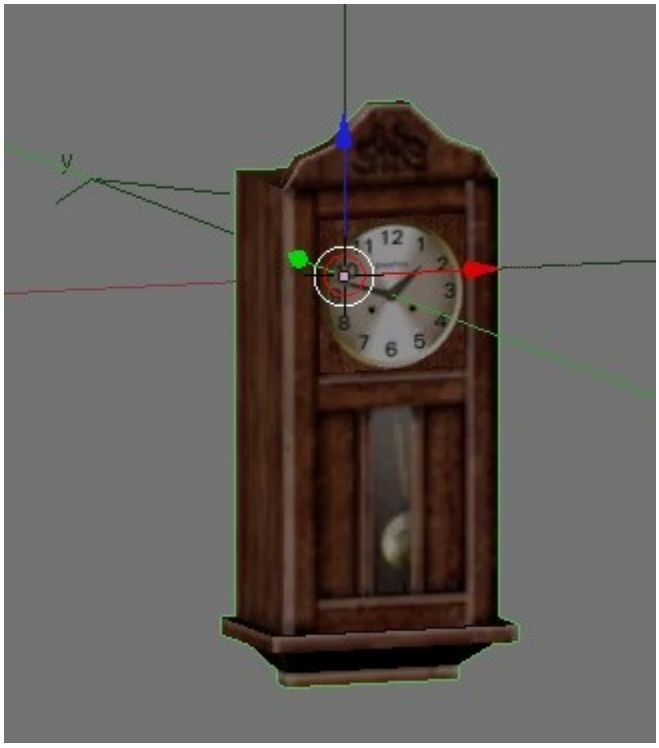
After selecting the import script notice one of the windows has become a browser window. At the very top of this window is an address bar. The next bar underneath is the filename of the 3D model we wish to extract. You can enter **p_old_wallclock.msh** into this field. Or you can first navigate to where you extracted the model and click on the file. After seeing the file name in the field it wants it in, click on the **Import MSH** button. (If you do not have a 3D view window, switch to it now)



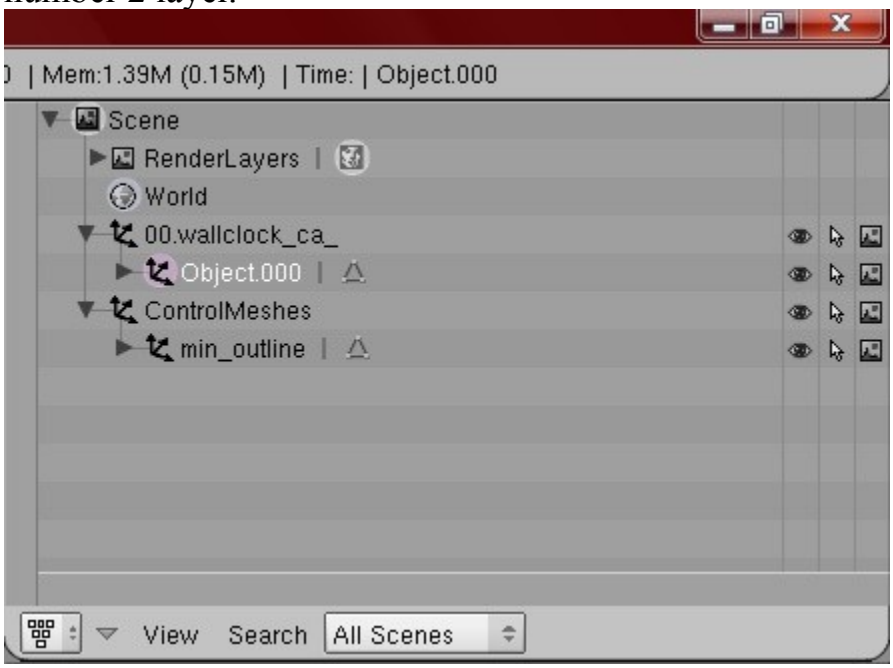


Note: There is no mouse copy and paste in Blender. But you can use **CTRL + V** to paste. So if you copied the address of where the 3D model is, you can click on the address bar in Blender's File Browser and CTRL+V the location. This is probably the fastest way. You can also click the little button before the file name field and get a drop down menu. This will allow you to click your way to where your file is located.

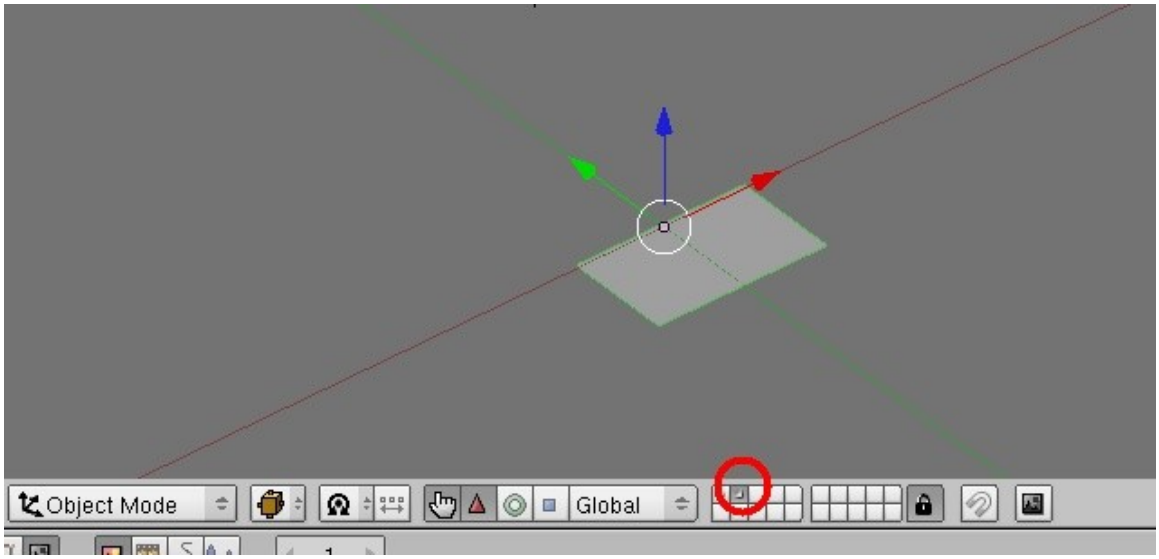

z
y


y
12
1
2
3
4
5
6
7
8
11

In the 3D view window you will see a grayish 3D model of the clock. Middle mouse wheel held down as a button will spin the view around. In the *Outliner* window you see two *Empty* names: **00.wallclock_ca_** and **ControlMeshes**. Under the 00.wallclock_ca_ is the 3d model's name: **Object.000**. This is a scene usable prop. Most props that can be selected in scenes has a **_ca_** in it's name. The **min_outline** is in another Blender layer. It is actually not needed if you wanted to make your own clock. If you wanted to you can right click on the **min_outline** and delete it. And also the **controlmeshes**. But lets say you just leave this alone. If you wanted to make a new clock, you could leave these two items alone and never touch them and it won't hurt the result of your model. The game uses that data, and we particularly, in the case of a wall clock, would not need it. If you want to look at it in the 3D view window you will have to switch to Blender's number 2 layer.



Wish I picked a different model for a beginner. These Blender layers aren't that important in the beginning. In this case it's not going to hurt anything. But for the sake of curiosity, lets go into another *Blender layer*. Look at the bottom of the 3D view window. There is a small grid of little tiny boxes. 2 of them are being used by the model. These are the buttons for entering Blender's other layers.



What is a layer? For modding, a *layer* is where you can hide stuff that does not need to be visible during the modeling process. So if you are designing a 3D set, you don't want to be working around objects that only the Movies Game needs and obstructs your view. So the import script puts these items into another layer. I have no idea why a small clock needs a miniature outline. But it is not important. Layers really come in handy if you are going to film a movie in Blender. You could place items in other layers and have them pop into view while the camera is rolling. For instance, a candle flame. Either a sophisticated particle or merely another bright object, can be hidden in layer number 2. And then have an actor light a candle. Then, lets say at frame 127, the candle flame object is told to switch from being in layer two to the first layer. And for the rest of the movie scene, the flame will now be visible to the camera. It was always there but the camera didn't see it because it was in layer number 2. Just as in Hollywood, filming movies is also about trickery and slight of hand. Cheating to achieve a desired appearance. Layers come in very handy. But don't worry about them now. If you want to take a look at some of the objects we can find in those other layers, try importing a set into Blender. Use those little buttons to go between the layers. Sets have 4 layers in use.

The Steps Taken:

- Extract a model using MED (The Movies Game Editor)
- Select the import script by clicking on File>Import>The Movies (.msh) 2009 Import Script
- Navigate to where you 3D model was extracted to
- Select the model so that it's name shows up in the file name field
- Click the **Import Meshes** button

4. Making & Exporting A Prop In Blender For The Movies

This tutorial shows how to create the easiest type of game model, as far as work load is concerned. A *Prop* has, or can have, only one **Object**. One **material** and one **Empty**. Those are very fast to add or create. You could create this model in less then a minute.
The immediate steps are:

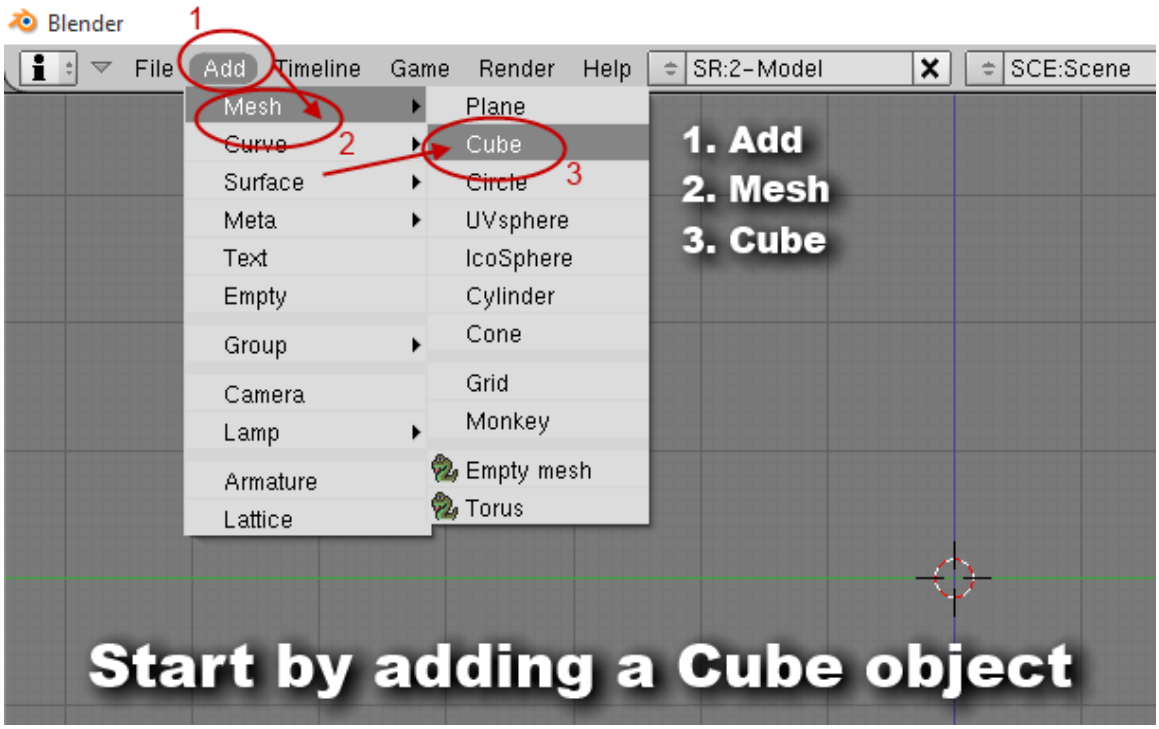
- Add a **Cube** object
- Rename it to something with a number. **11** would work, or **Object.000** would work.
- Unwrap a *UVmap* for the Cube
- Load an image into the *UVMap* window
- Project all of the UVmap over this image
- Give the Cube a *material*
- Give the material a *Diffuse* Block
- Give *Diffuse* Block an *Image* assignment
- Select the image for it
- Add an **EMPTY** (little green arrows)
- Rename the *Empty* to something with a number **00** will work
- Add a **Blend Group** and rename it to the same name as the *Empty* **00**
- Enter the *Empty* into the same Blend Group (which now has the same name **00**)
- Enter the Cube (**11**) into that *group* as well
- *Parent* the cube to the *Empty*
- Run *The Movies Preflight Check*, to see if there are any errors
- Export the model into the game
- Use MED to inject the Cube model, as a prop, into the game

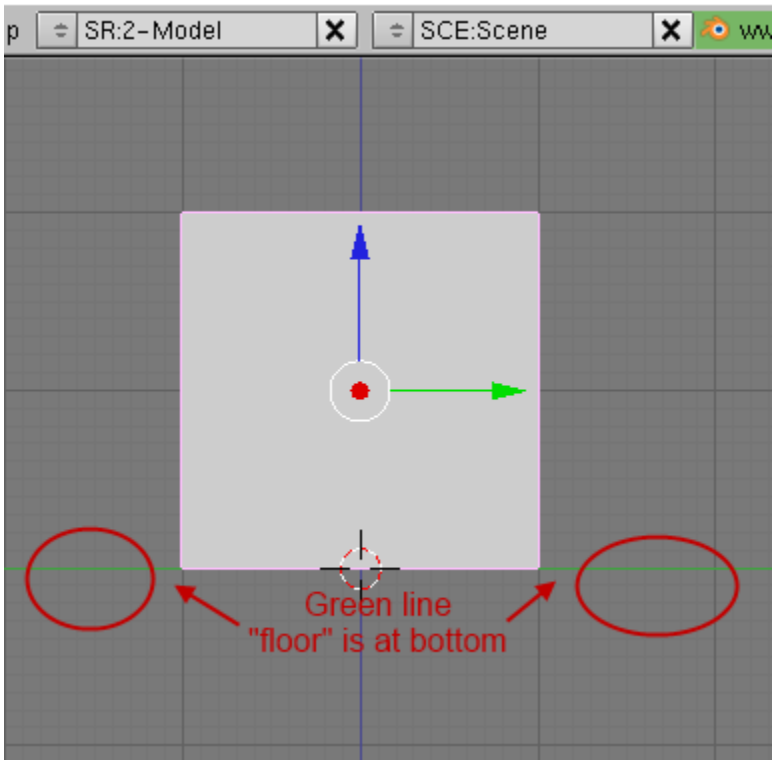
The tutorial shows the process of developing something in Blender and getting it into the game. Also shows that there are requirements for the project for it to be exported.

To follow this tutorial, you will need a texture image for the model you will be creating. In this example a 6 sided dice. You can use the image here. If this PDF allows, you can right click and save it. Then load it in a Paint

program, for example Paint.Net. Then just save it as p_6sided.dds. Make sure that the file format you save it as is (.DDS). And be sure to save it in your Movies data\textures\props folder. The image should be 512x512 pixels.
C:\Program Files\Lionhead Studios Ltd\The Movies\Data\Textures\Props

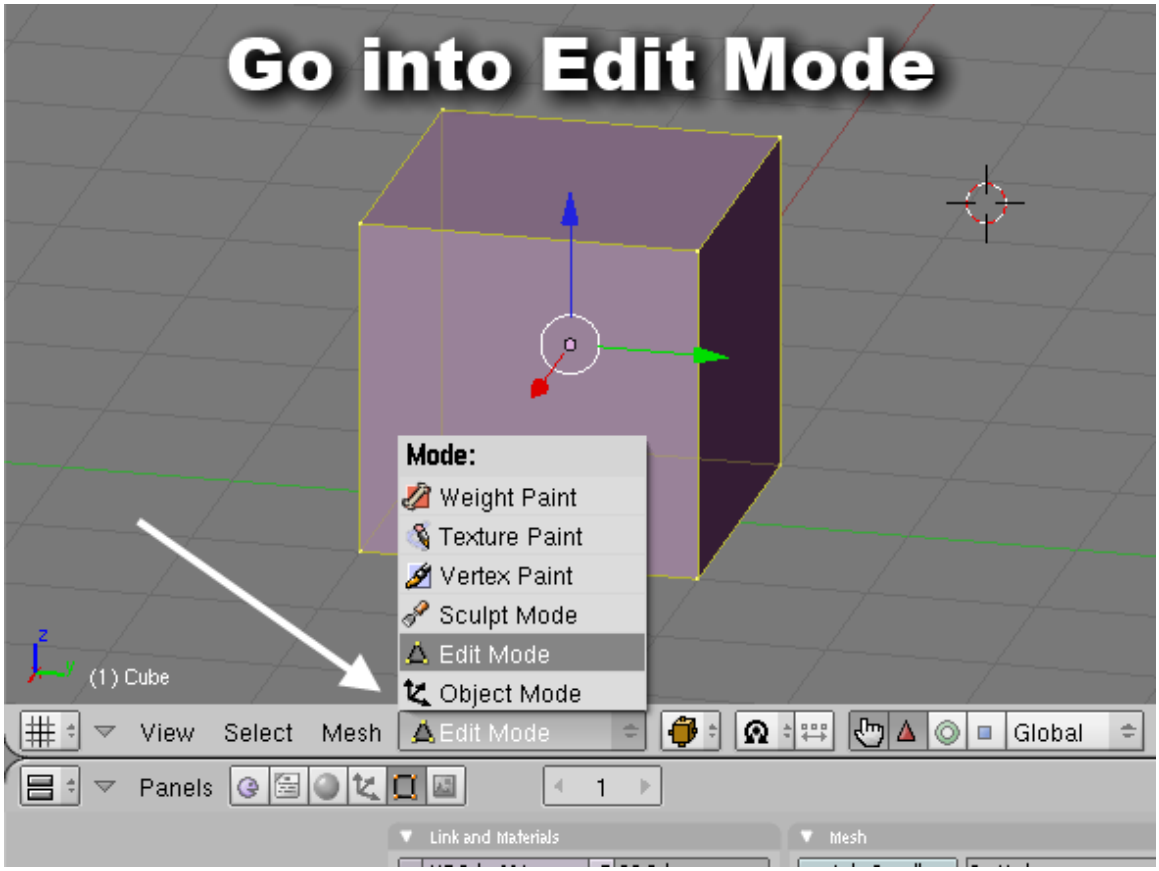
The first thing we need once you start up Blender is a Cube object. You can bring up a popup menu by hitting the **SPACE** bar.
Or go up to **ADD**, and select **Mesh**, then select **Cube**.



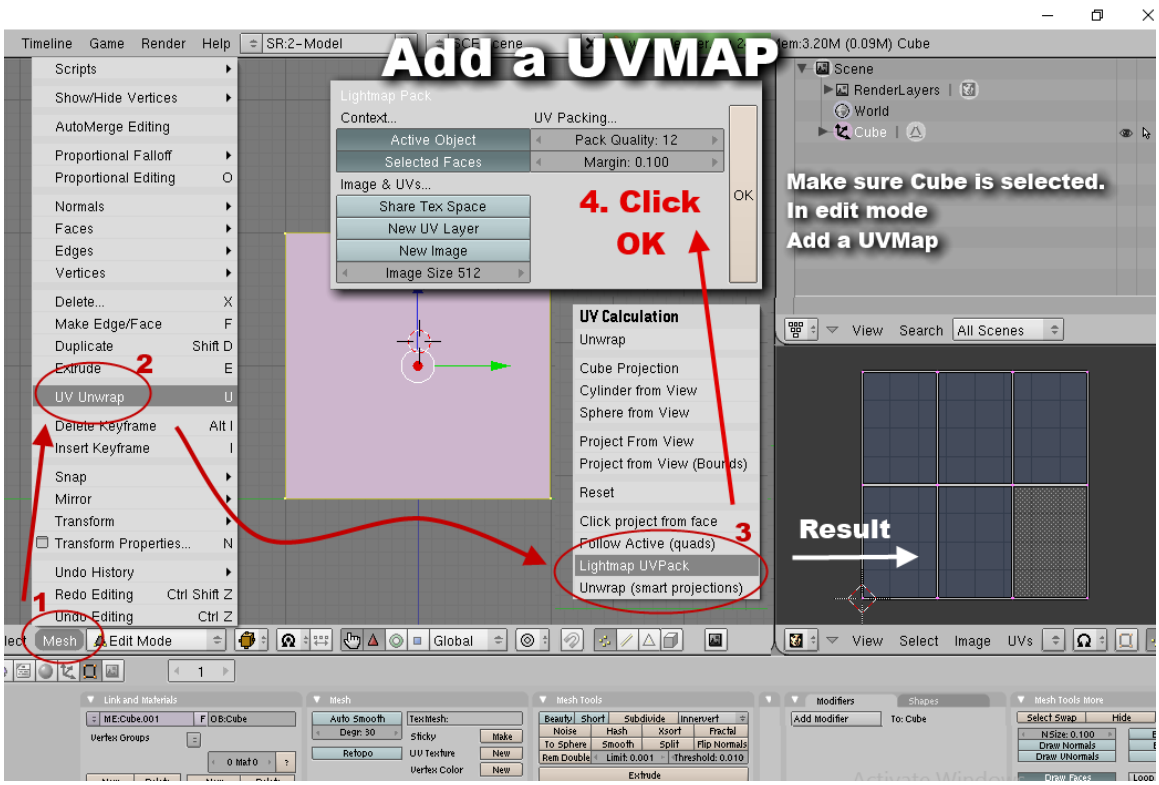


We need the bottom of the Cube to rest on the ‘floor’ of the set. When we added the Cube it’s center is at the floor level. In Blender the *Y axis* is the floor and is represented by a green line.

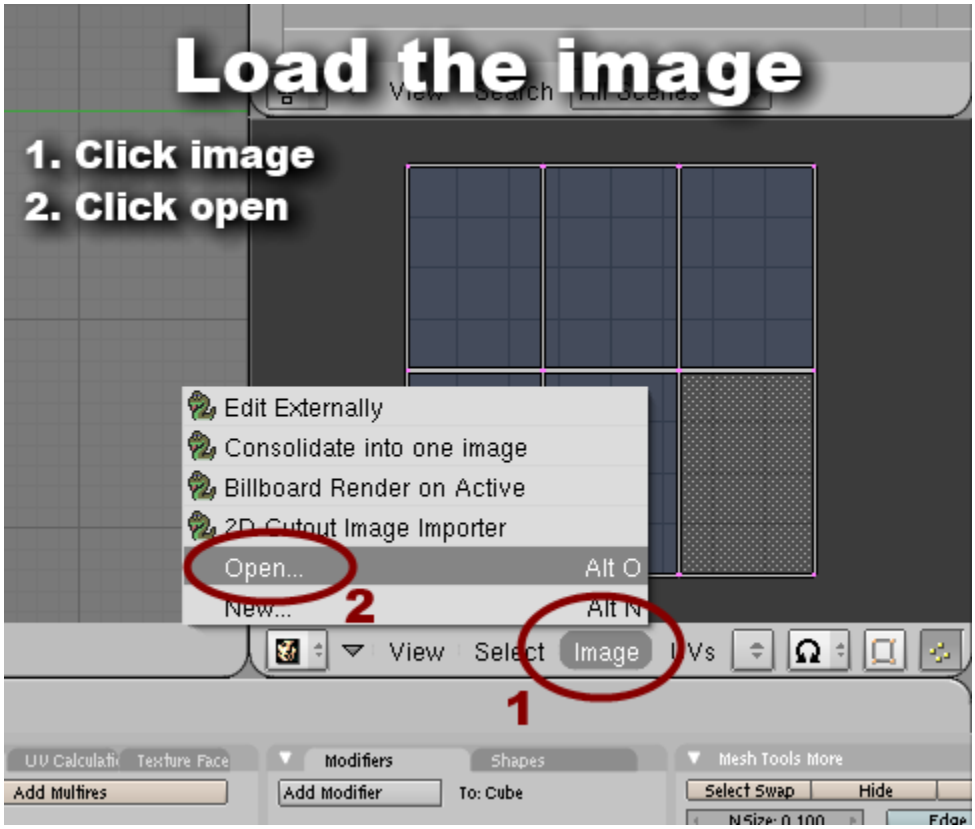
Select the Cube if it is not selected already. By clicking on it or by clicking on it's name in the *Outliner* window. Then press Hotkey **G** for *Grab*. After that click Hotkey **Z**. This will constrain the moving or grabbing to the Z axis, which is up and downwards. As you move the Cube hold down **CTRL** key. Holding down **CTRL** while moving an object will snap it along the grid lines. When we reach above the green line it will be snapped to it.
Once the bottom of the Cube rest on the floor or the green line, we want to give it a *UVMap*. This can be done in *Edit Mode*.



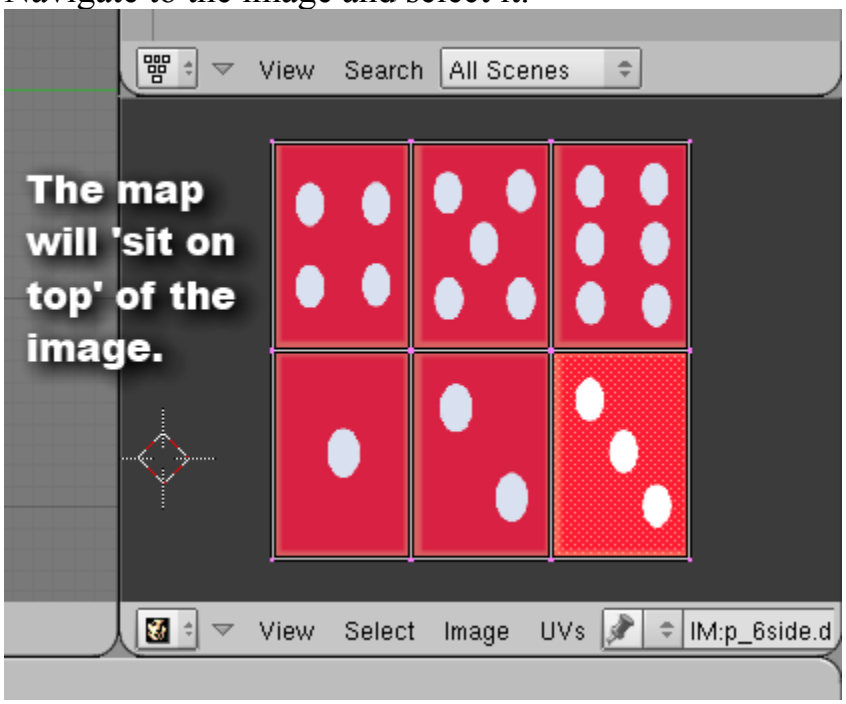
Go into *Edit Mode*. Make sure all verts are selected. Be default they should be. If not press Hotkey **A** for *All*. Until the object is lit up in 3D window. Every *face* that is selected in 3D window will get a UVMap when projecting it.



In *Edit Mode* click the **EDIT** button. From the menu select **UV Mapping**. Which brings up a popup menu. Select **Lightmap UVPack**. This choice will allow each face of the Cube to occupy it's own space in the *UV Window*. When the next popup menu shows, just click **OK**, At the *UV Window* to the right a UV map has been created. An image needs to also be shown in this window. The UV Map needs to appear to be 'sitting' on top of this image. Since the map is visible, loading a new image will automatically set it on top of the image. *A result of still being in Edit Mode*. Click the **Image** button, then select **Open**.

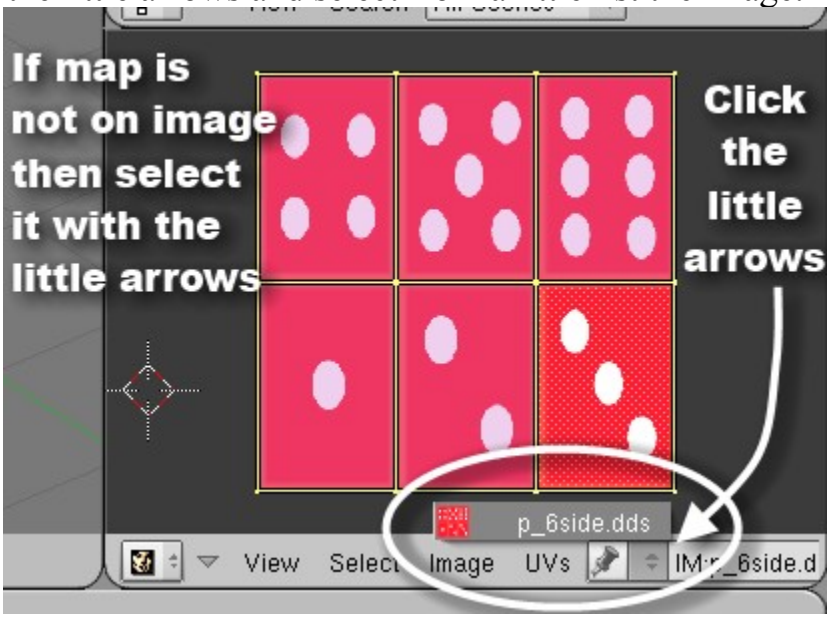


Navigate to the image and select it.

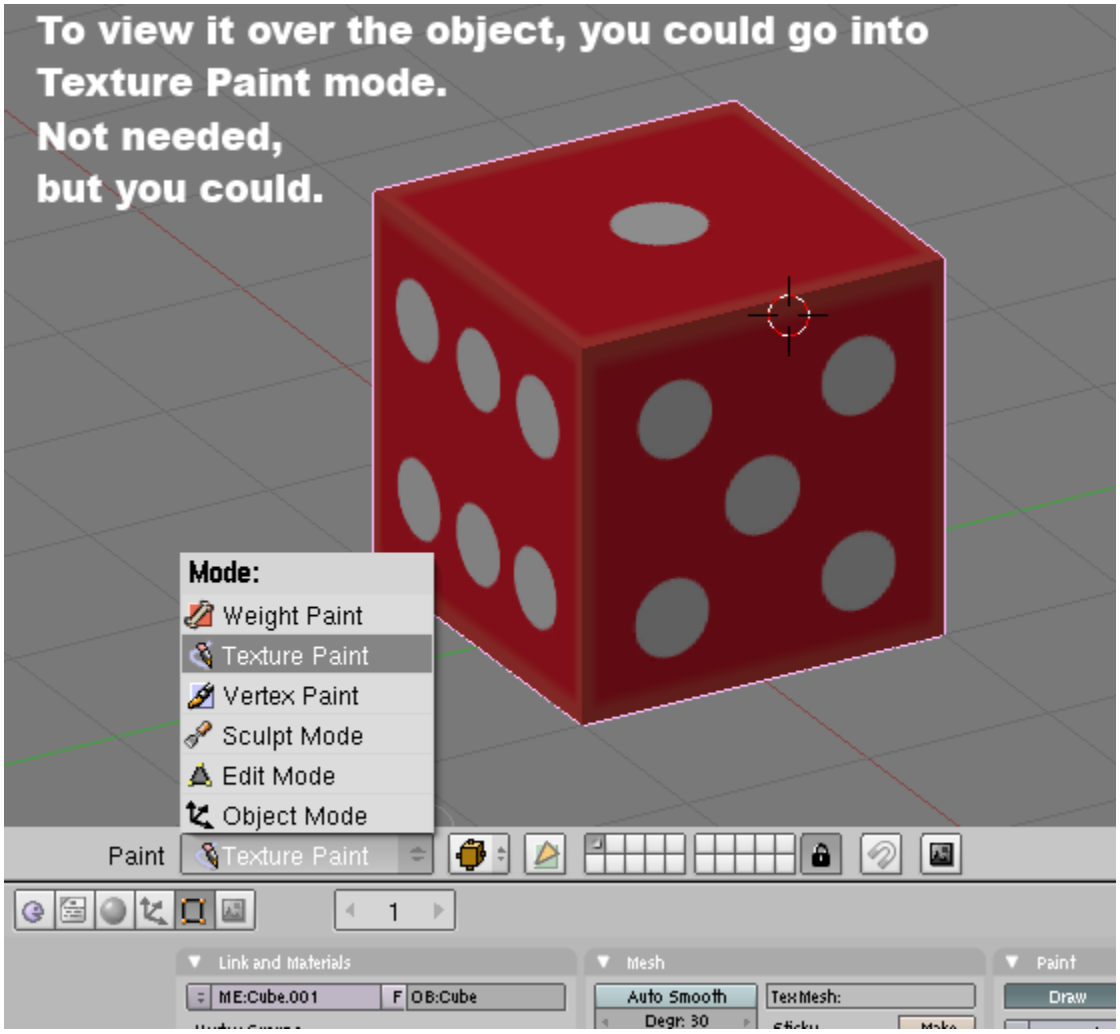


If you need to place a UV map over an image make sure that it is ALL selected. It should already be, provided you haven't done anything else. But for future reference, to place a UV map over an image or a different image, make sure the object is in *Edit Mode* and that all the verts are selected. Make

sure the entire map is also selected in the *UV* window. It will be yellowish. Then click the little arrows and select from a little list the image.

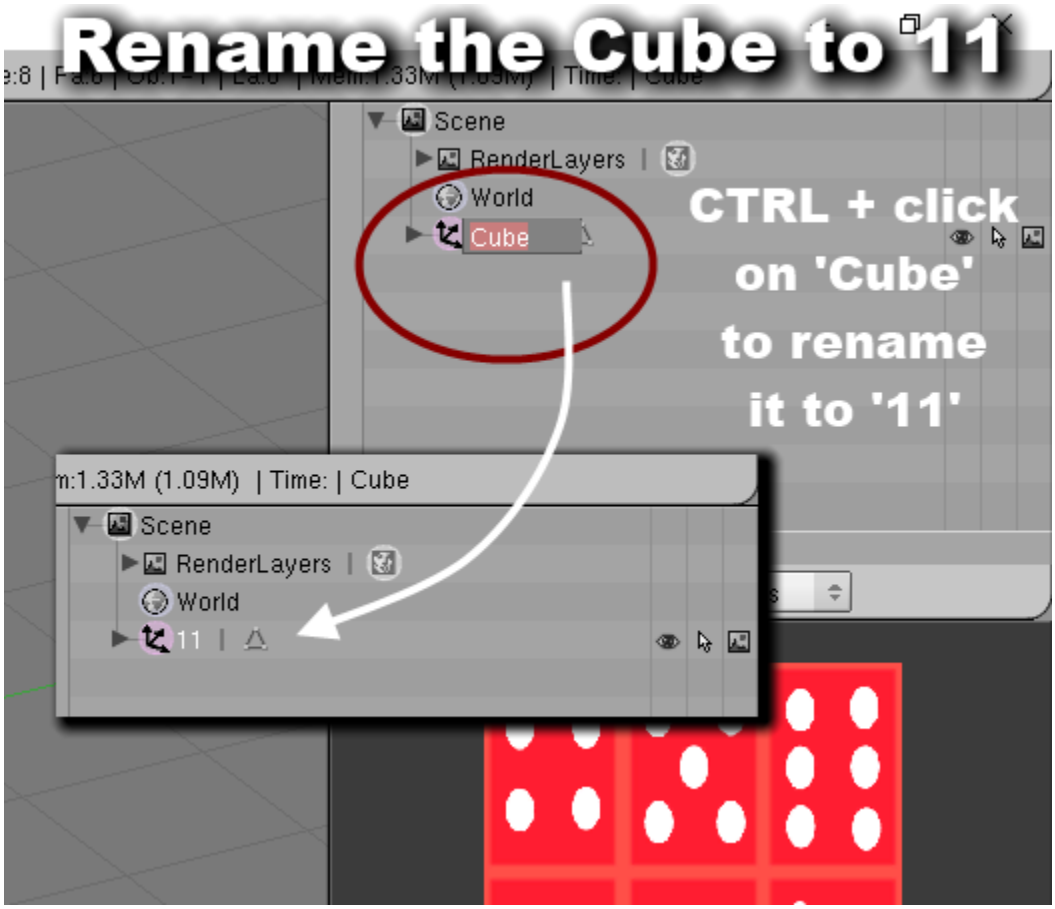


You may want to see how the image now looks over the model. Because that is what the *UV map* is for. You could go into *Texture Paint Mode* to see it on the model. You don't really need it in order to finish the tutorial, but why not? Just be sure to go back into *Object Mode* when done looking at it.



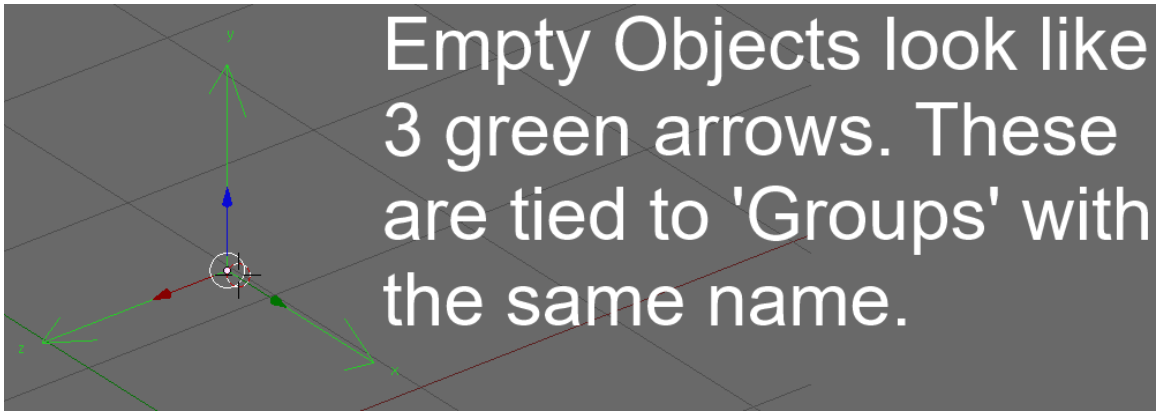
(Be sure to go back into *Object Mode.)*

The project needs all objects to have a number in its name. I opt to give the names only a number and no letters or words. The reason is objects have the number after the name Object. *Example:* **Object.000.** While *Empties* have the number before the name. *Example:* **00.floorset**. And this can get confusing for a beginner. But both the *Empty* and object can just have numbers as the name. (Just not the same name). Rename the 'Cube' to **11**.



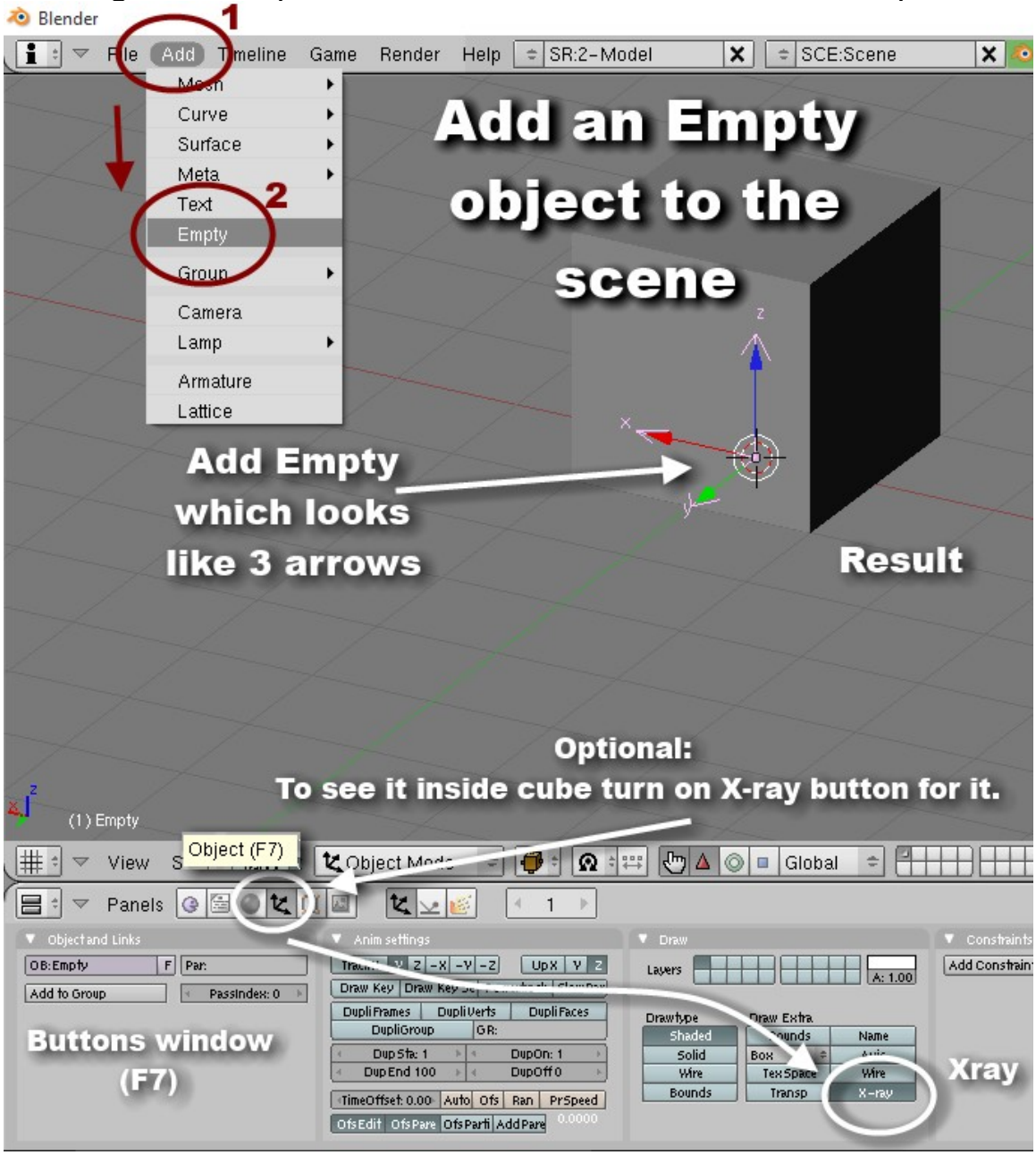
If you hold down the **CTRL** button at the same time clicking on an item in the *Outliner* window, you will be prompted to rename it.

EMPTY OBJECTS…



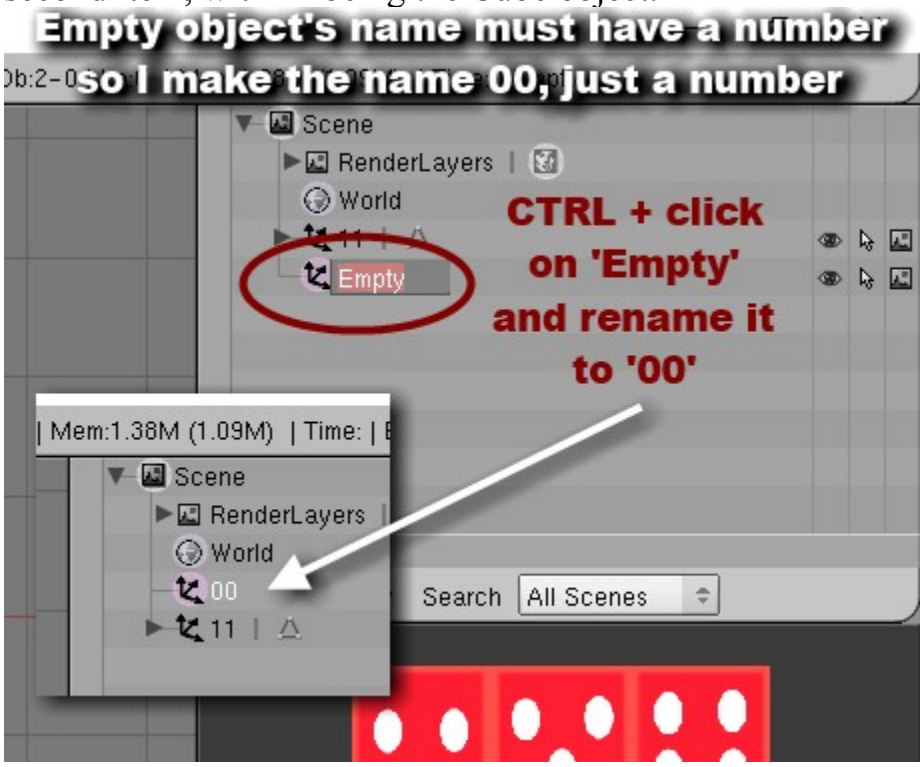
When an *Empty* is added to the scene they will simply be called 'Empty' in the *Outliner* window. But Movies game content has different names for them. The names have numbers in the beginning of them to keep them in order. Example would be *00.corridore* followed by *01.slidingdoor*. Then *02.elevator* and *03.floor,* or whatever. The Movies need these to have a

number in them. For a simple prop we create with no special functions, it does not matter what the name or number is. So to make it simple I make the name only a number. Like **00**.
The *Empty* we will add Blender will call **Empty** but we will rename it to just **00**.
Lets add the **Empty** object. This is done in the way we added the Cube. Selecting from a drop down menu of the **Add** button. Or with the space bar.



Because the Cube is larger then the 3 Arrows you might not be able to see it. You can make it visible inside any object by clicking the **X-ray** button. But you don't have to. Next rename the *Empty* to **00**. This will move it

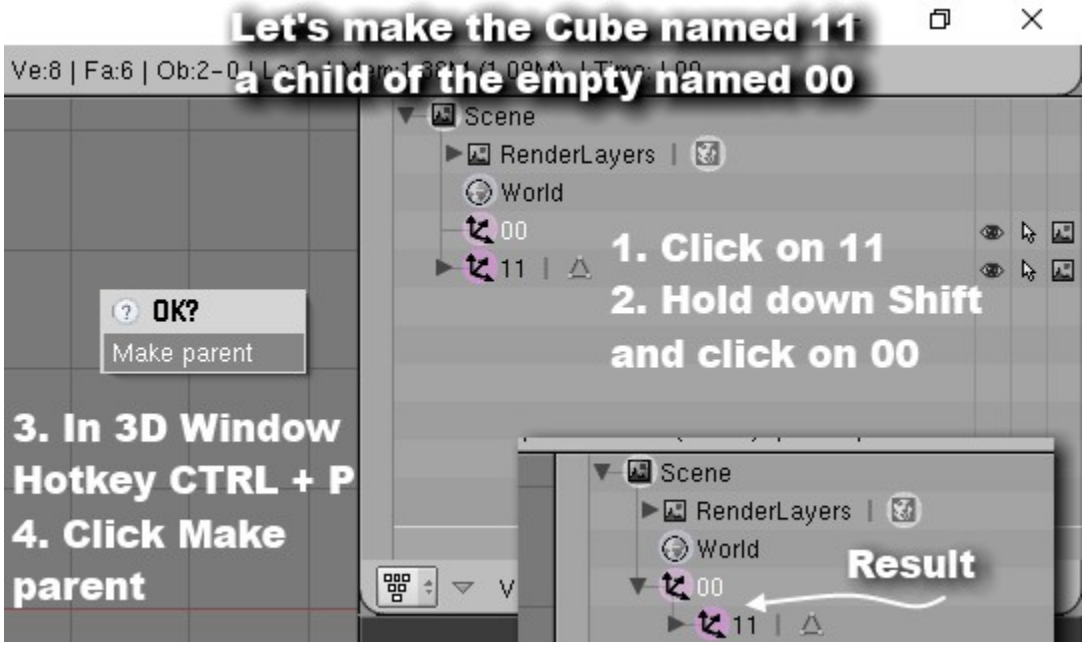
alphabetically to the first item in the *Outliner* window instead of being the second item, with **11** being the Cube object.



Parenting is making an object the child of another.
We want the *Empty* named **00** to be the *Parent* of the cube named **11**.
We want **11** to be the *child* of **00**.
What is a *Parent* in Blender? It is an assignment that can be applied to any object in the 3D window. A Parent will have under it's control any item that was made it's *child* object. We can already see the *Parent* and *Child* relationship with content we import from the Movies game. A simple prop will have a single *Empty* object and a single 3D object as it's *child*.
Some more advanced Movies models have more then one *Empty*. Like sets. They are dynamic and can even have moving parts. For this reason all of the static items are pooled into one group. And the moving parts of a set will have their own *groups* as well.

Start by clicking on **11** (the dice).
Hold down S**hift**.
Click on **00** (the *Empty*).
Then in the 3D window, hit Hotkey **CTRL + P**.
This activates *parenting*. And because the *Empty* was the last thing selected in the sequence, it will be made the *Parent*. And the cube will be made the

child. Remember the object we want to be the *Parent* must always be selected **last** in a sequence!



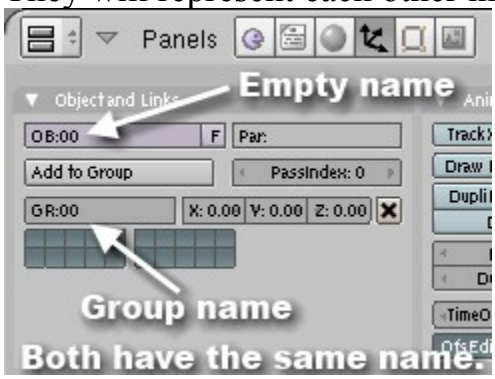
Clicking on the tiny black arrow next to the **00** will show the *child* objects under it. You will no longer see **11** in the list but it is still there. It is now the *child* of the *Empty*. Clicking the arrow next to a name in the *Outliner* window reveals all *child* objects belonging to it.

Groups…

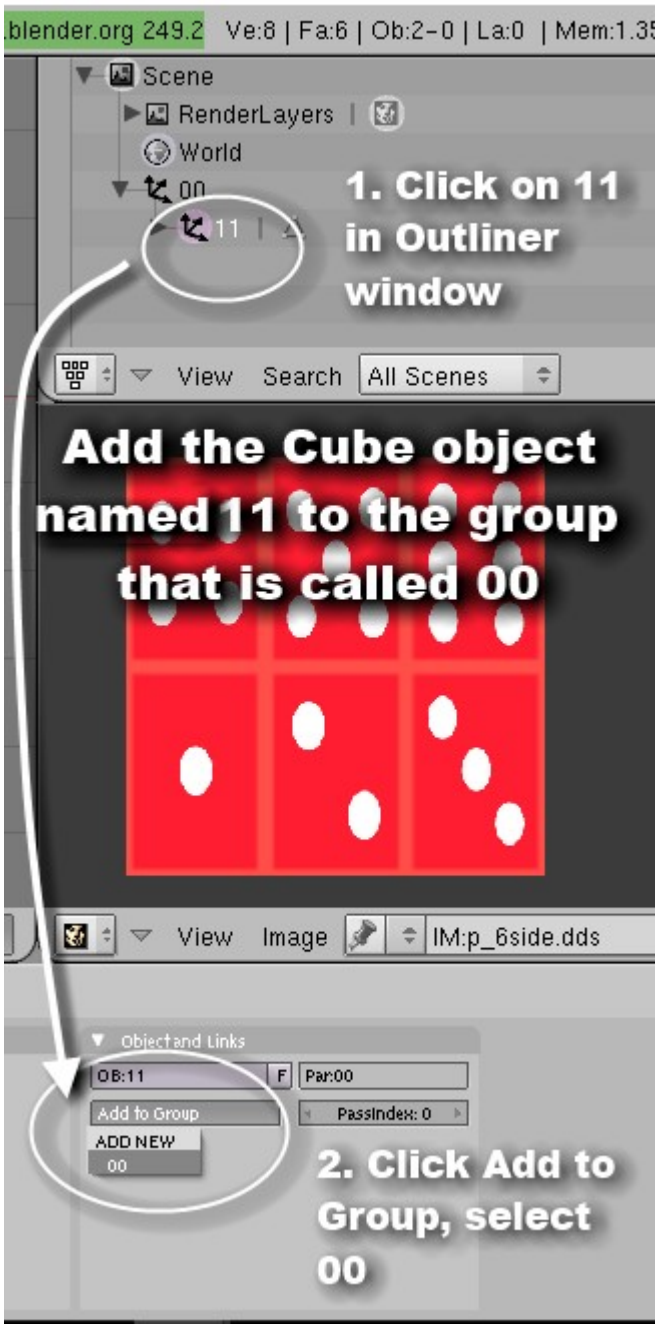
Empty objects have a special property the other objects do not. They also have a *Group* with the same name. We don't have a *Group* yet and we will make one soon. All *children* objects of the *Empty* (like the dice) will also be entered into this *Group*. But we have to make one first. We make a *Group* by adding an object to it in the *Buttons* window.

First click on the *Empty* **00** in the *Outliner*. Press **F7** to set the *Buttons* window to the appropriate menu.
Click the **Add to group** button.

Since there isn't one yet Blender will ask if you want to create one. **Add new**. A new *Group* has been created and is simply called **Group**.
Click on it to rename it.
Rename it to **00**, the same name as the *Empty*.
Now the *Empty* **00** belongs to a *Group* also titled **00**.
They will represent each other in the game.



The *Empty* titled **00** is the master of the whole project but the dice object is not yet entered into this new *Group*.
Let's enter it now.

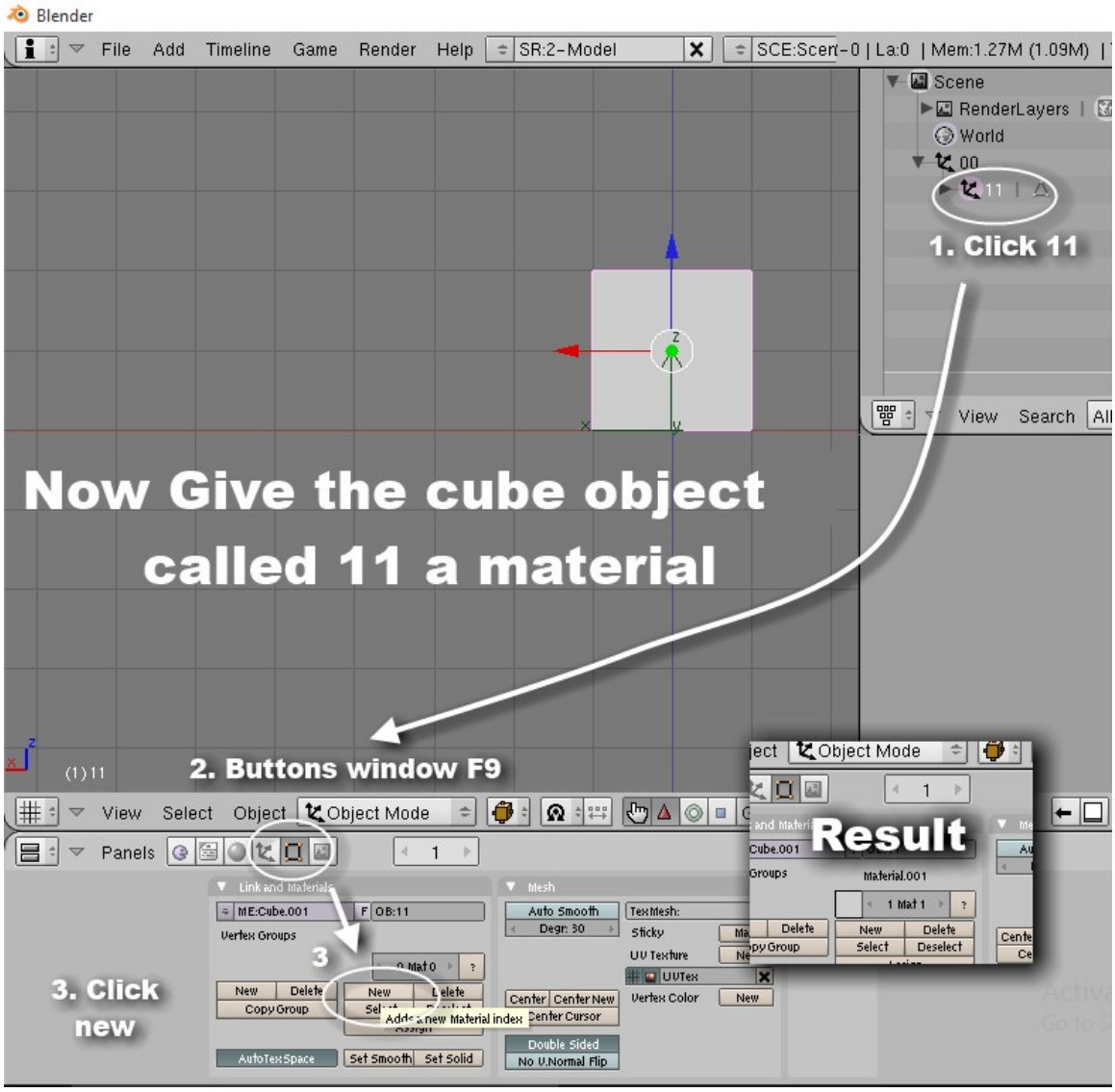


Click on the **11** in the *Outliner* window. (You could also just click on the dice in the 3D window but selecting in the *Outliner* window is what I prefer.)
In the *Buttons* window click **Add to group**. Now a *Group* is already available for selection since we just created one a second ago. Select **00**.

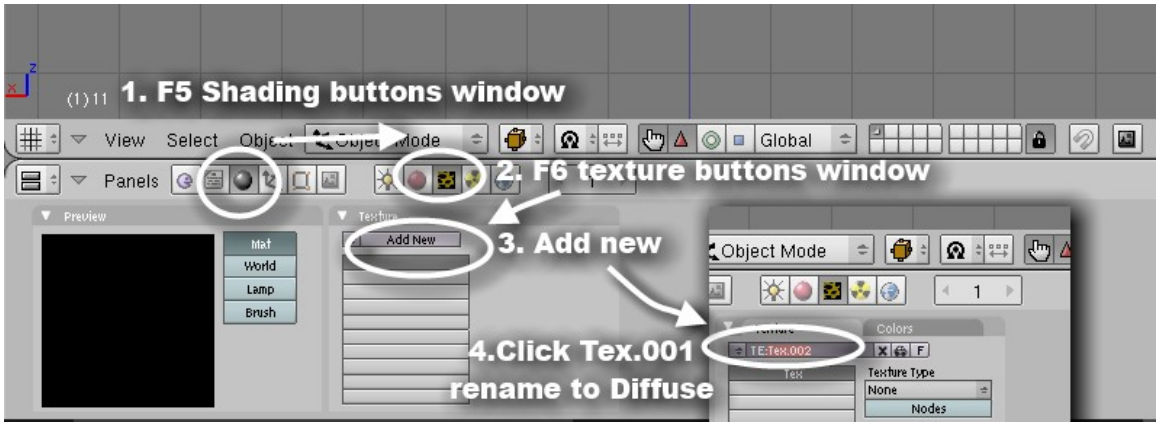
Adding a material…

The object has an image but needs a *material* so that the game knows how to make it visible. Windows are transparent because it's *material* is set to it.

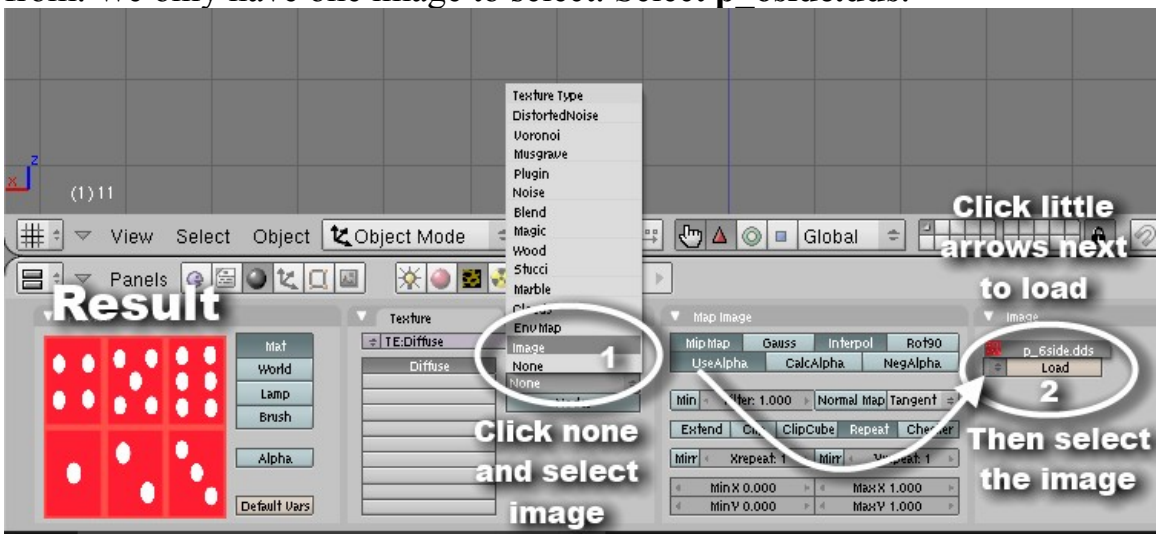
Car bodies are shiny because they have that setting in its *material*. The dice will have its own *material* too.
Usually I add a material in the **F9** buttons menu. First click on the dice which is **11** in the *Outliner.* Then hit **F9** and in the buttons menu and click the **New** button. Now the dice object has *material.*



This *material* needs more work. It needs the image, the red dice texture, to be assigned to it. We need to change to the *material* settings in the *Buttons* window, hotkey **F6**. You will see a horizontal line brick called **Add New**. Click it. The first horizontal line in the stack is now called **Tex**. Menu options to the right keep appearing as you progress. Click on the **Tex** and rename it **Diffuse**. The Movies uses four blocks. *Diffuse, Reflection, Lightmap* and *Specular*. All we need is *Diffuse* and you can leave the others blank.

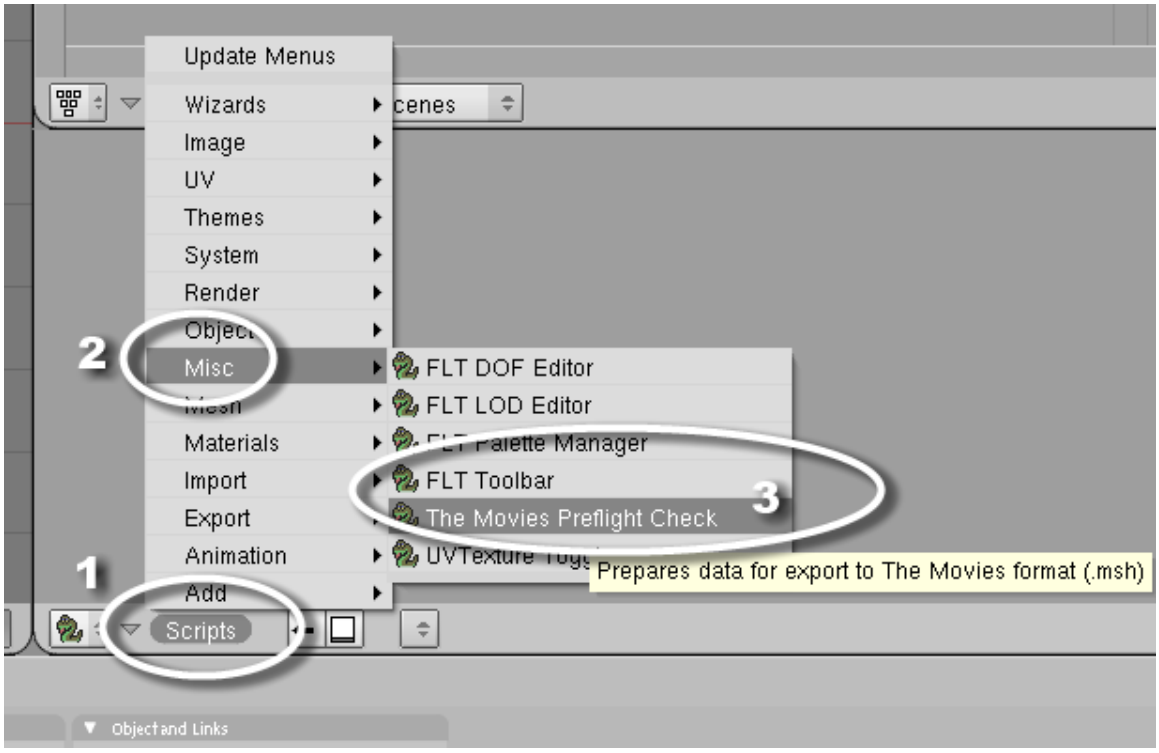


To the right is a popup menu which at the moment is called **None**. Click it and select **Image**. Another menu at the far right was added. You could load a new image to it. But since we have already loaded the red 6 sided dice texture into Blender, we will select it instead. Right before the button called 'Load' are **little arrows**. Click it to reveal a popup list of images to select from. We only have one image to select. Select **p_6side.dds**.



At the far left we now see the image and know that it is now the *Diffuse* texture assigned to the cube object.

The Movies Python scripts came with a 'preflight' script that will check this project to see if it is ready for export. Turn the *UV* window into a *Scripts* window. You will see a button called **Scripts**. Click it and hover over the **Misc.** From the next menu select **The Movies Preflight Check**.



A browser window will appear after a moment and show you a stack of green rows. You should see something like the following…

The Movies MSH Export Preflight Tests

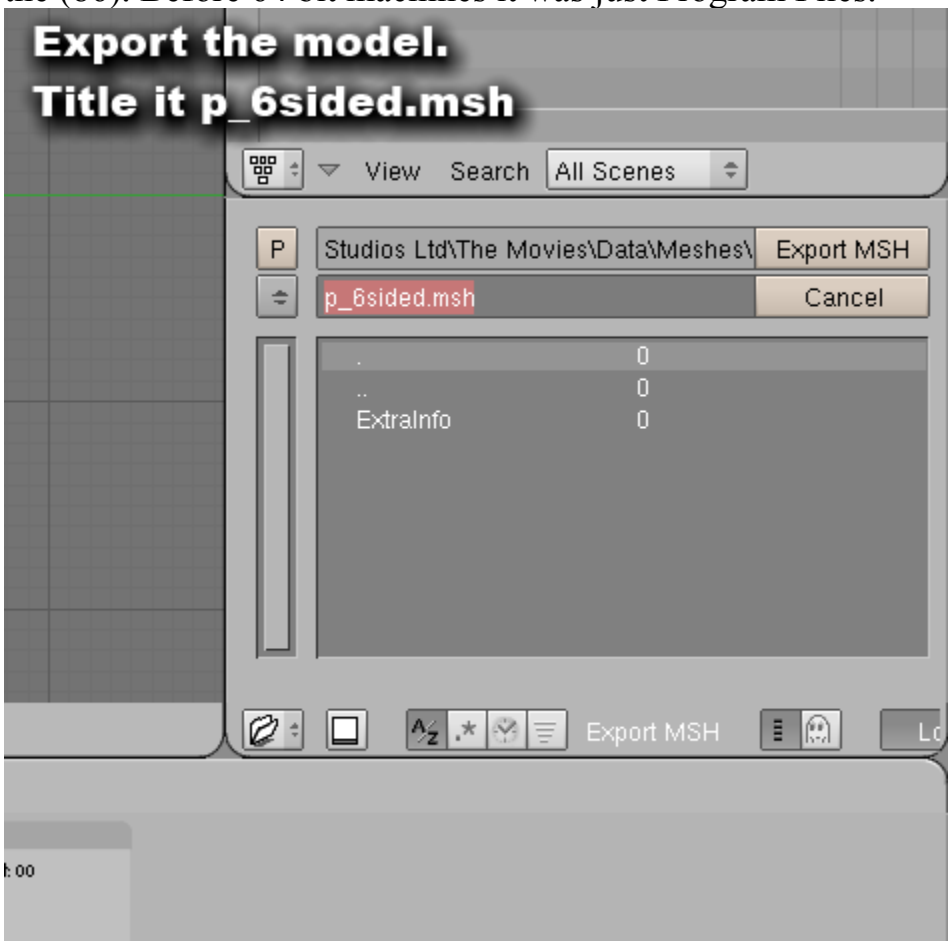
Start Time:
Duration: 0:00:01.856000
Status:

Results for checking validity and integrity of mesh to be exported to The Movies game format.
Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
BasicTests: Verifies very basic requirements.	4	4	0	0	Detail
ArmatureTests: Tests several aspects of any Armatures in this mesh.	4	4	0	0	Detail
InGroupTest: Each Blender Object is in a Blender Group.	11	11	0	0	Detail
MultiGroupTest: Each Blender Object is in only ONE Blender Group.	11	11	0	0	Detail
GroupHasEmptyTest: Each populated Blender Group has an Empty pivot object.	1	1	0	0	Detail
GroupHasOrderNumberTest: Each Group has a number in the name, determining export order.	1	1	0	0	Detail
Group-ParentName: The name of the Group and the parent Empty are the same.	1	1	0	0	Detail
Group-PropertyName: The name of the Group and the IDProperty 'grpName' value are the same.	1	1	0	0	Detail
GroupHeirarchyTest: Meshes must be children of an Empty or an Armature.	10	10	0	0	Detail
BoneVertexGroupName: Vertex Groups must match Bone names.	10	10	0	0	Detail
Parent-PropertyName: The IDProperty 'grpName' and the name of the Empty are the same.	1	1	0	0	Detail
MaterialZeroTest: Each Grouped Mesh has a material in the first slot.	10	10	0	0	Detail
MaterialTypeTest: Textures in Materials must be Images.	16	16	0	0	Detail
UnusedMaterialsTest: Checks for unused Materials.	10	10	0	0	Detail
UnusedTexturesTest: Checks for unused Textures.	40	40	0	0	Detail
UnusedImagesTest: Checks for unused Images.	11	11	0	0	Detail
UVTest: Meshes must have UV mapping.	10	10	0	0	Detail
UVLayerNameTest: UV Layers must be named 'UVTex' or 'LightMap'.	10	10	0	0	Detail
NonFaceVertsTest: Checks for vertexes that are not in any faces.	10	10	0	0	Detail
VertWeightTest: Checks for vertexes that are not weighted to any Vertex Groups.	3378	3378	0	0	Detail
Total	**3550**	**3550**	**0**	**0**	

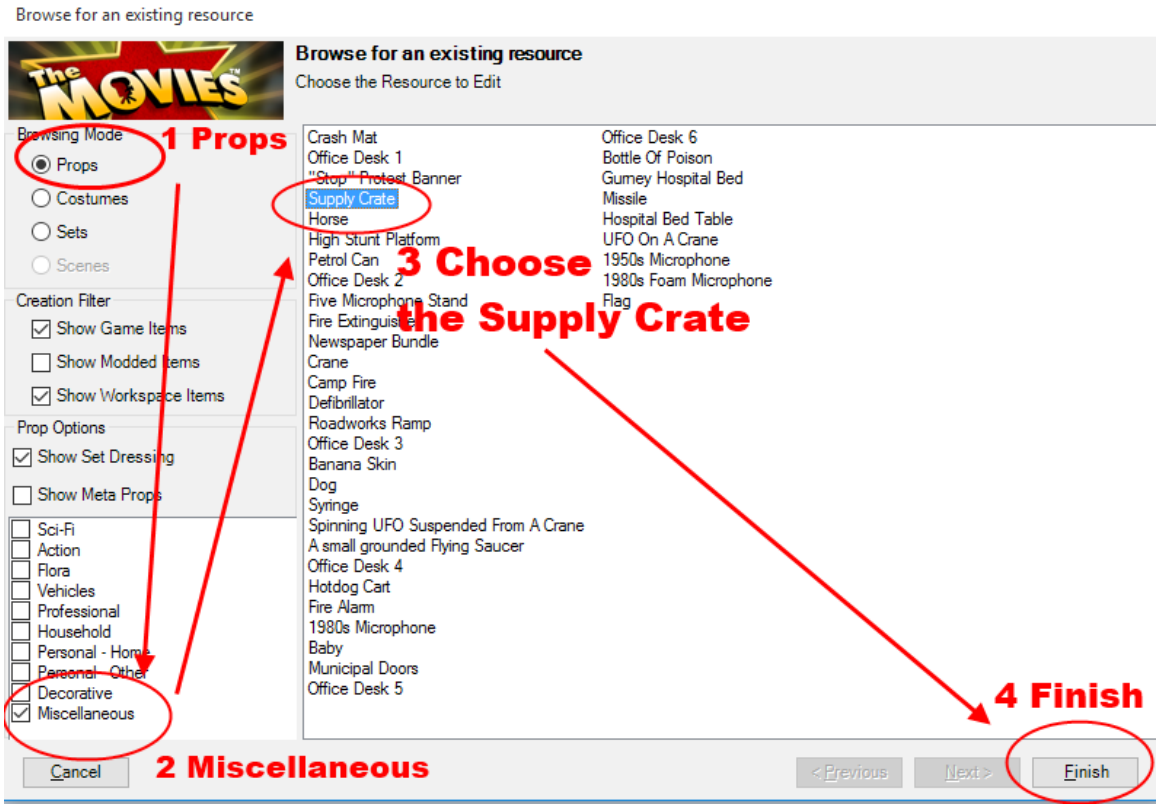
If all went well you got all green lights.
Now export the model into the Movies game.
You can export it to any location and have MED import it into the mod making process. However you should probably export the model to your game's own meshes folder.

C:\Program Files (x86)\Lionhead Studios Ltd\The Movies\Data\Meshes
Note that in 64 bit machines the location is the secondary program files with the (86). Before 64 bit machines it was just Program Files.



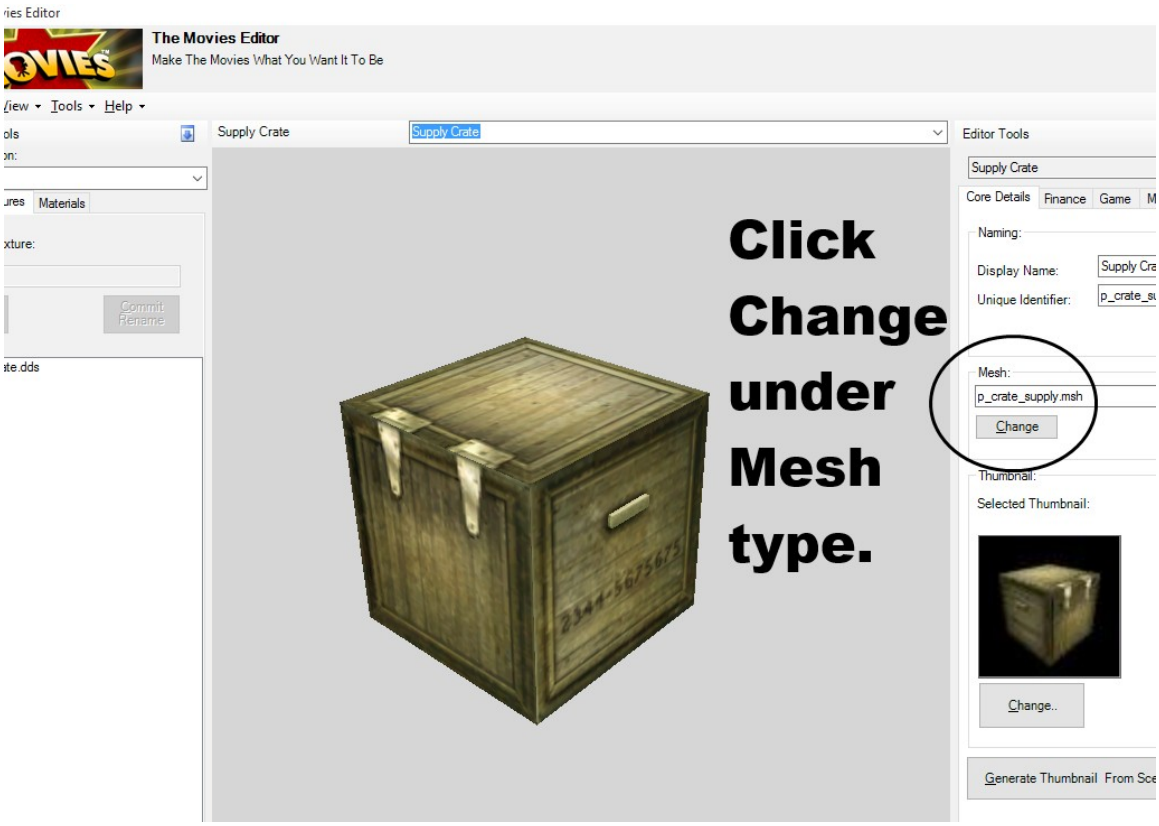
Also be sure to put the texture called p_6side.dds in the game's C:\Program Files (x86)\Lionhead Studios Ltd\The Movies\Data\Textures\Props
MED does a scan at start up so if those two files are already in your game's folders MED will see them. Otherwise, when you get to that step just import it. Fire up MED The Movies Editor…

When you see Action Wizard select **Modify (Create From Existing Resource).** MED will inject new mods into the game by barrowing from an existing item. Once you load an item into MED the next thing to do is replace the 3D model with your new model. Rename the item. And generate a new thumbnail. I will walk you through it. First select props. Then unselect *Sci-Fi,* and select *Miscellaneous*. And finally let's select the **Supply Crate** from a list on the right side.

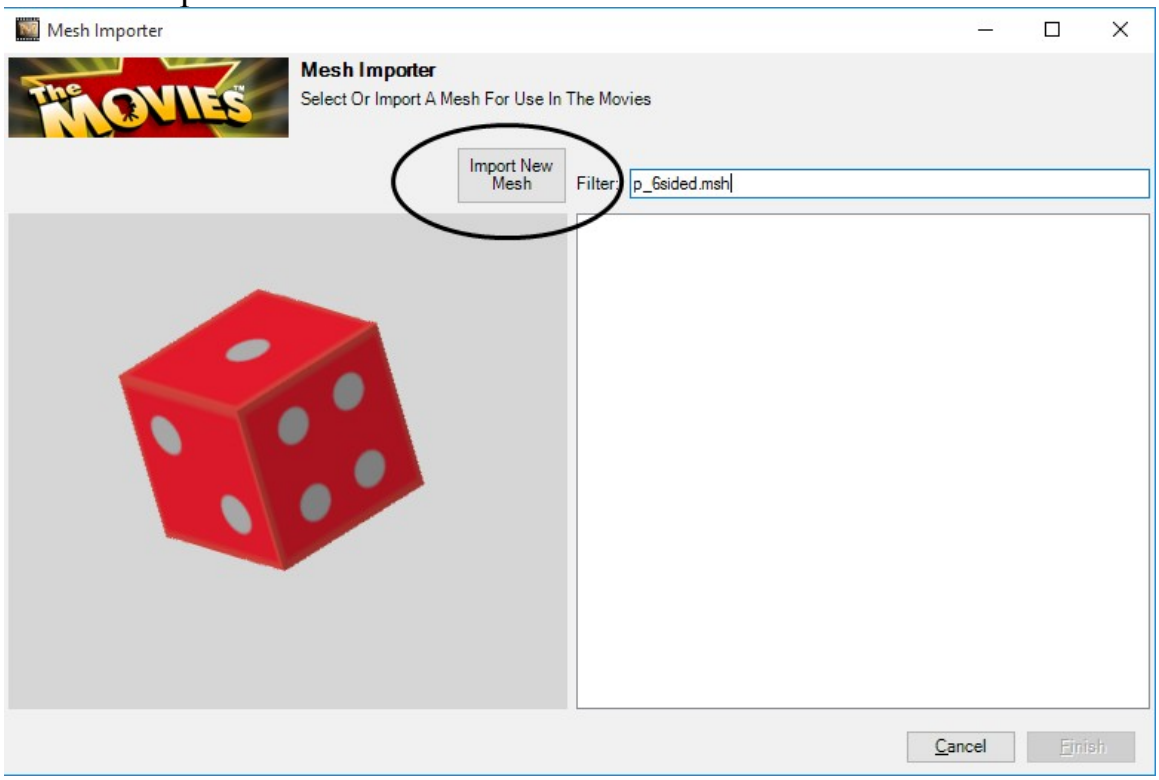


Then click **Finish**.

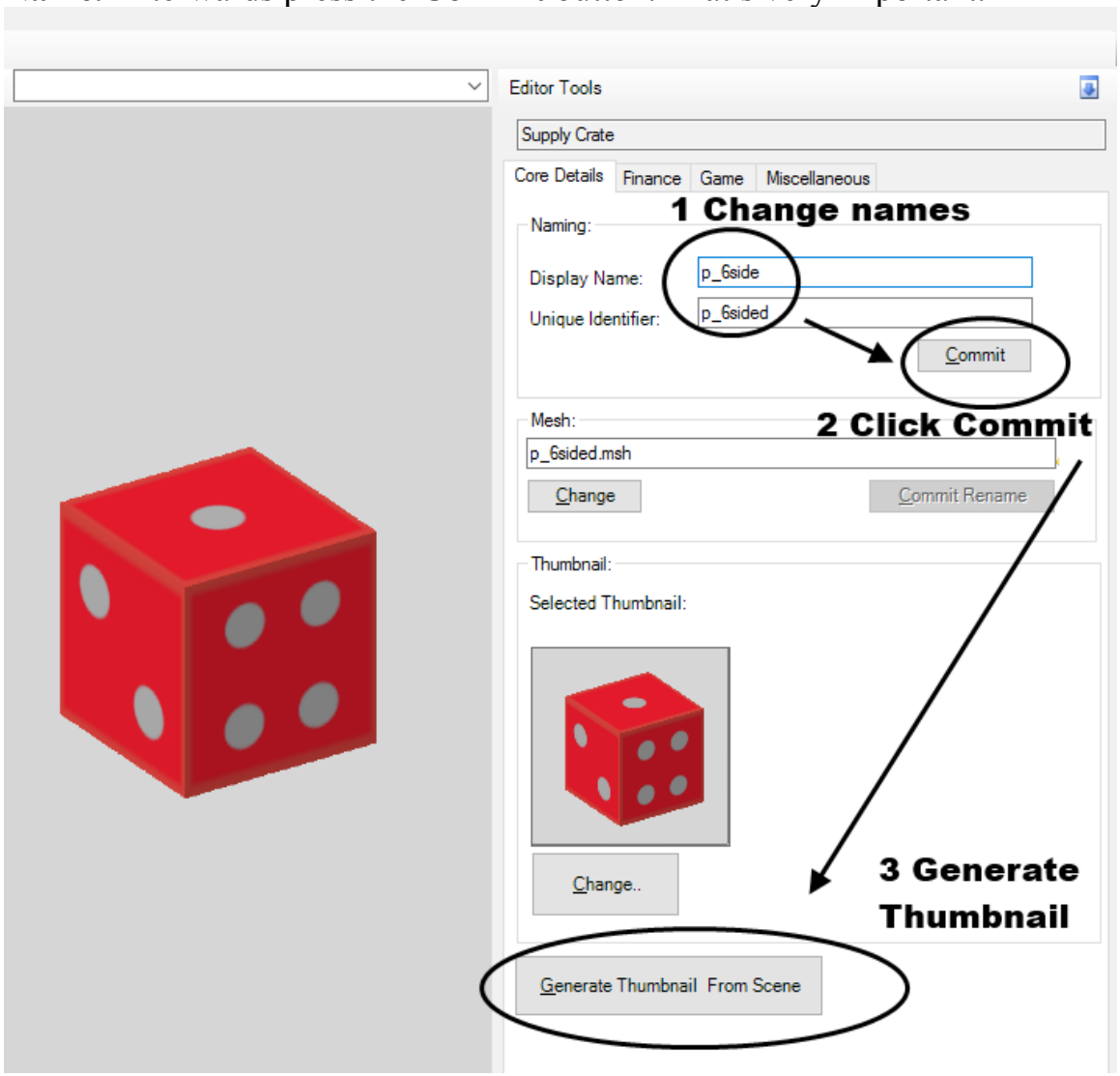
Now change the 3D model with the Dice model. Click the **Change** button on the right under *Mesh*.



When the popup window appears you can scroll down the list and find the dice. If it is not there then it isn't yet in your game's folders. In that case click the import mesh button.

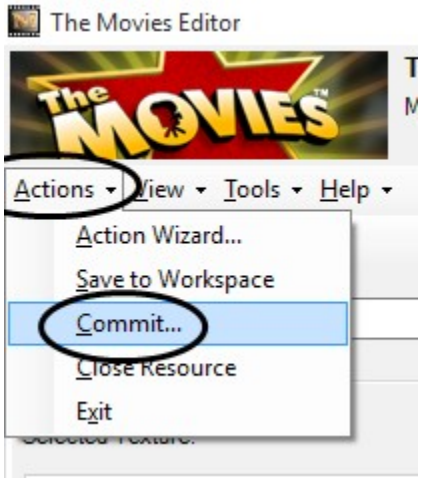


Next we want to change it's name. Both the **Unique Identifier** and **Display Name.** Afterwards press the **Commit** button. That's very important.



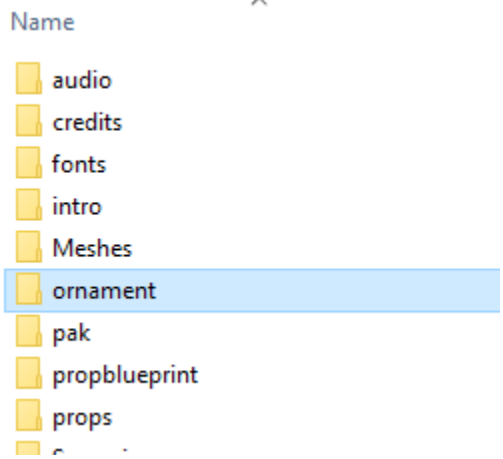
Then make a new thumbnail by clicking **Generate Thumbnail From Scene.**

It's ready now so lets put it in the game. At the top click **Actions.** From the drop down select **Commit.**



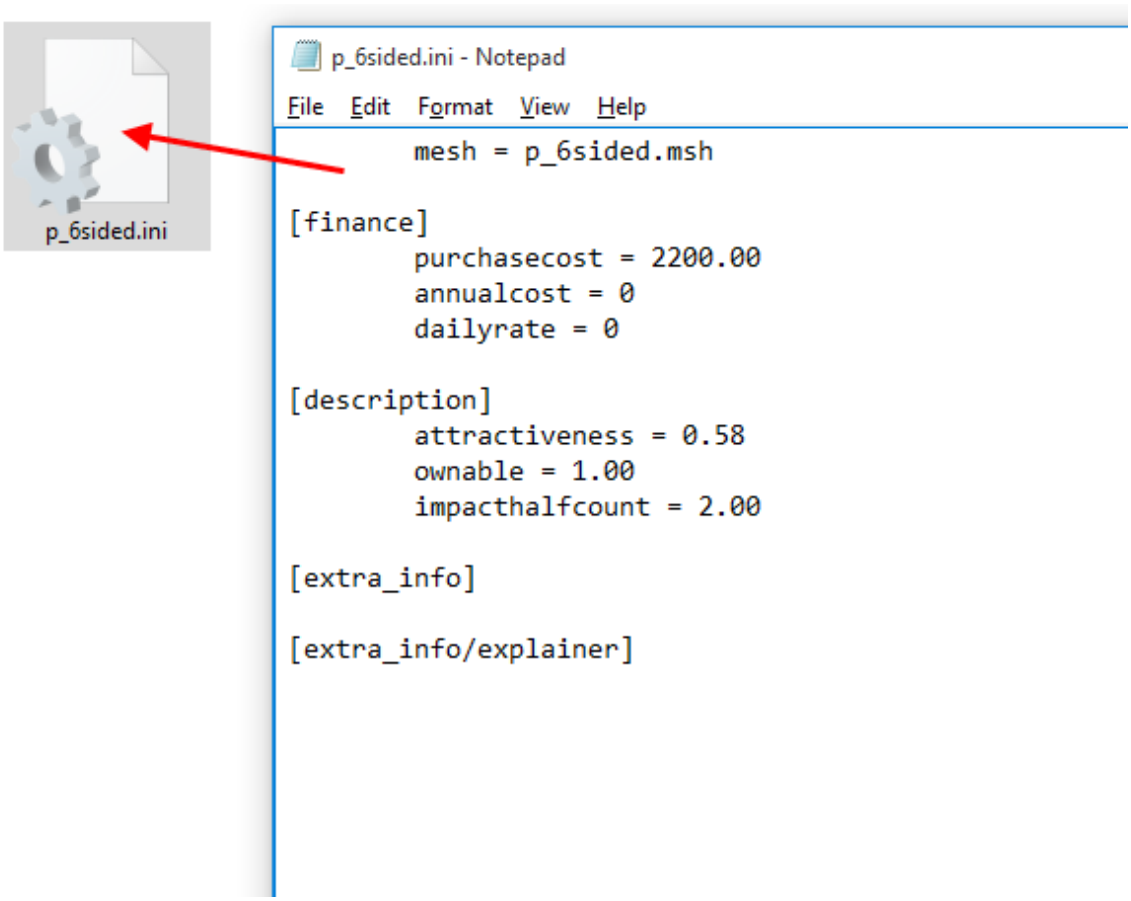
That will put it in the game. Fire up the game and place it on a set. If you want to see it on the studio lot then you can go another route. Instead of the Supply Crate, select the *Lionhead Fountain*. Replace the lionhead fountain 3D object with the dice and do everything else. Like renaming, thumbnail and committing. We picked the lionhead fountain because it can be found in the lot ornaments for selection. That is decided by the PropBluePrint folder's files. If you manually make the file, just extract the propblueprint file for the lionhead fountain and rename it to p_6sided.ini

After that you need to make a new folder in your game called **Ornament.**



Navigate into that folder and right click, select NEW.

Select Txt file, name it p_6sided.ini. It will normally create a (.TXT) file but you should rename it to end with (.INI) instead. In Windows10 you can turn on 'see file extensions' and in that way be able to rename file types.



Copy the following text and paste it into Notepad. Then save it.

```
        mesh = p_6sided.msh

[finance]
        purchasecost = 2200.00
        annualcost = 0
        dailyrate = 0

[description]
        attractiveness = 0.58
        ownable = 1.00
        impacthalfcount = 2.00

[extra_info]

[extra_info/explainer]
```

5. Materials

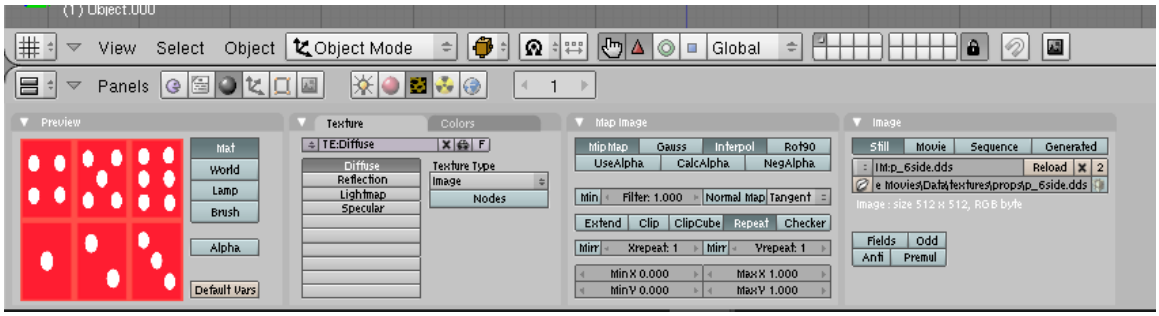
Most of the 3D objects found in a Movies game (.MSH) files has a *material*. *Materials* tell the game how an object is to be rendered. How it will appear to the player on the screen. *Materials* will have several flags. Whether it has alpha (transparency), or if the object will remain bright when the light setting is shifted to darkness on a set. *Materials* also list how many textures the object uses, like a lightmap, or the shiny environment texture. When creating or converting a 3D object for the Movies game, each will require a material.

Some larger files like the sets will have several objects sharing the same *material.* In such cases these objects will also share the same texture image assigned to it. There are some tricks you can learn. Some car textures have the car's body in the (.DDS) image, but also a small part of the image has some alpha glass appearance for the windshield. In that case the car can have more then one *material,* but still share the same (.DDS) texture image file.

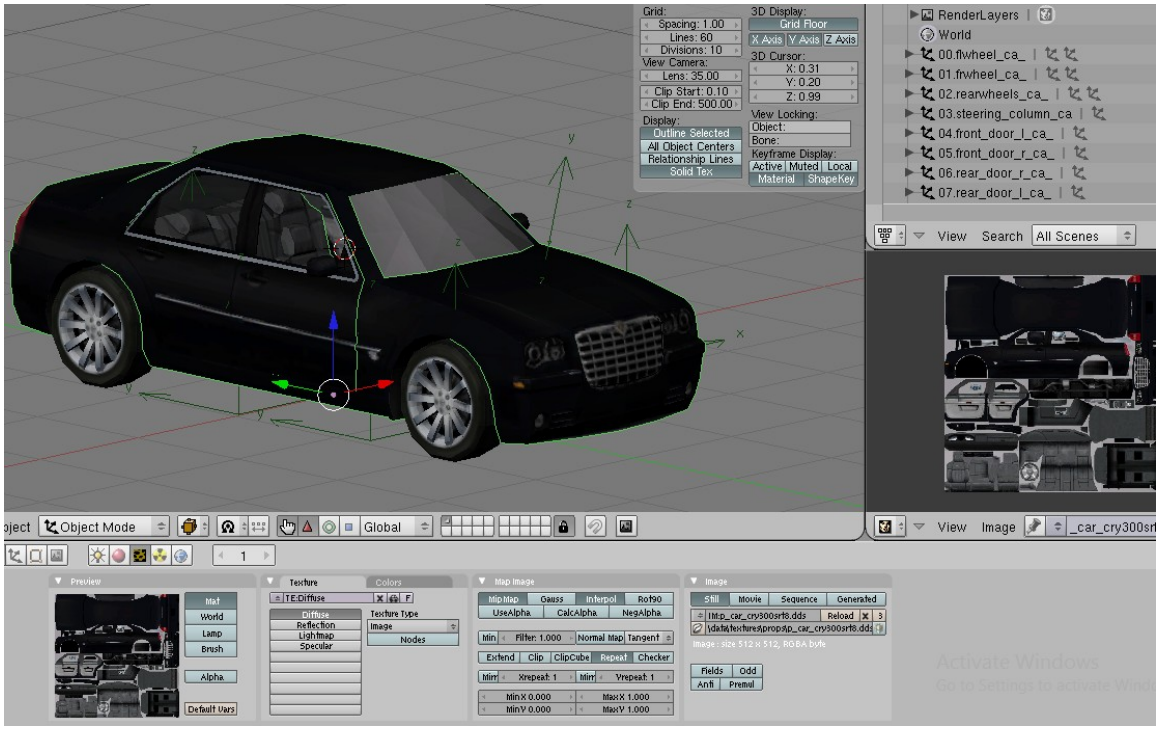
Some thoughts about deciding how many *materials* and images your project needs... During the time The Movies came out game developers, as a rule, always sought to reduce the size of the game. This meant reusing as many textures as possible. You may noticed that The Movies uses the same wooden texture for a desk as does a clock. In doing so there was no need to make another texture. Today it is different. With today's high end consoles and larger disk space and memory, you may find lots of repeat models and textures in the compressed folders. Loading times for The Movies on today's computers is much faster then it was on our old XP machines.

The basic idea of the *Material* is making sure your model will appear correctly during game play. Since you made a *UVMap* for the model, and can see this map resting over the (.DDS) image, the next step would be to give the object a *material.*
In the prop tutorial we created a very simple material for the 6 sided dice. We titled a texture block to Diffuse. The diffuse block is what the base (.DDS) image file was assigned to.

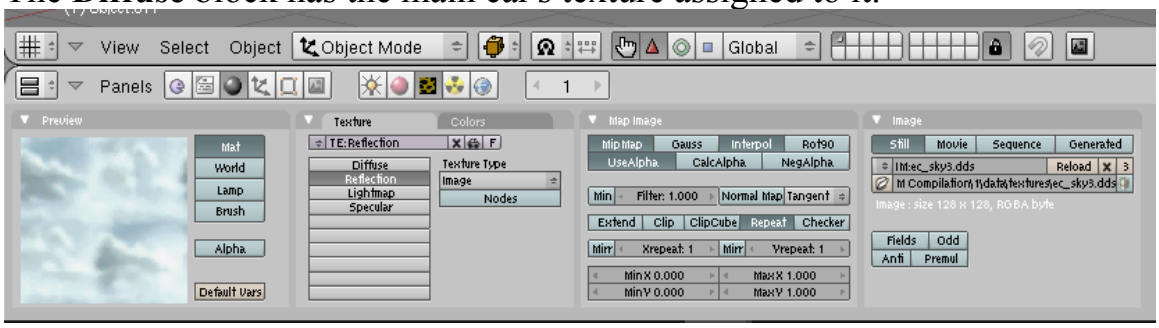
We could have created the three other texture blocks that Movies objects have. Or we could have left them blank because they were not needed for the project. The Movies export script doesn't care if they exist so long as the Diffuse block exist.



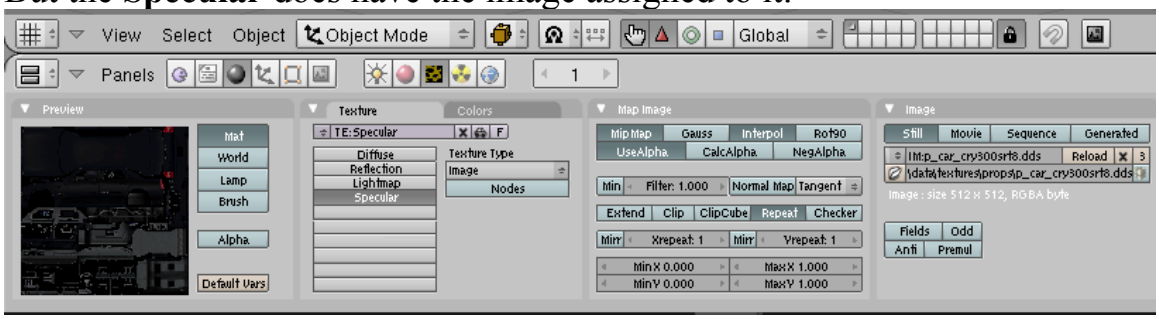
Every Movies object we import will have those four blocks listed. And in many cases there is no texture image assigned to each name in the stack. While some will use three or even all four. Import a car from The Movies and we will these settings.



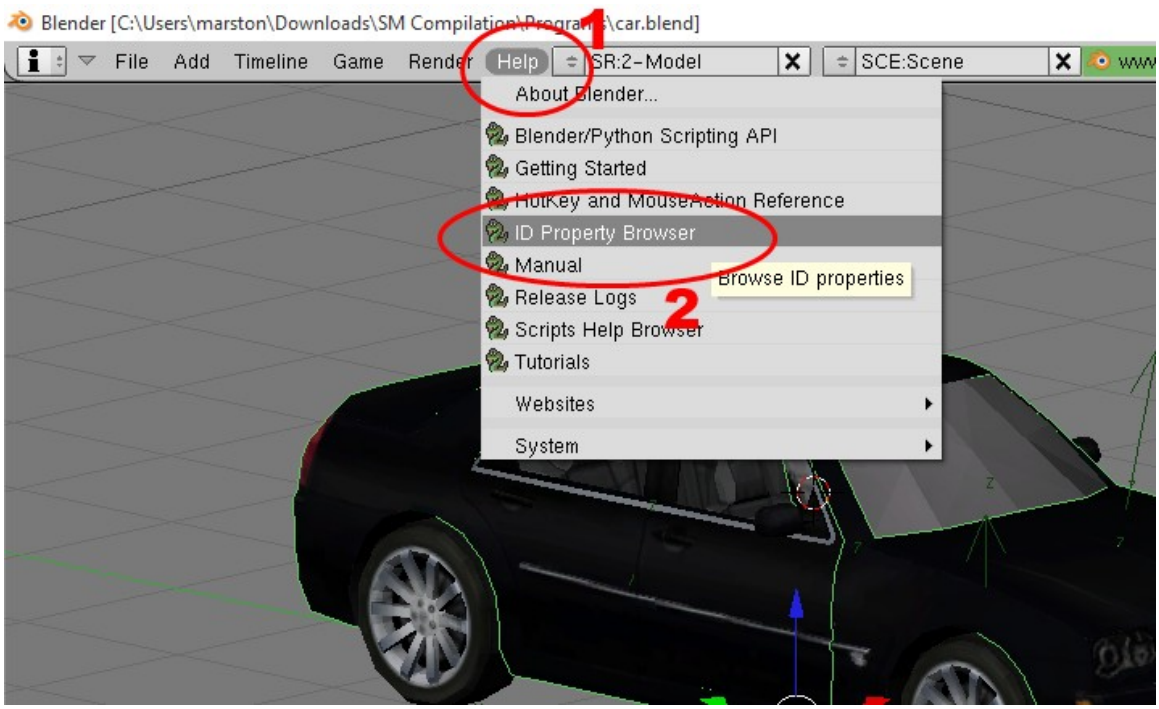
The **Diffuse** block has the main car's texture assigned to it.

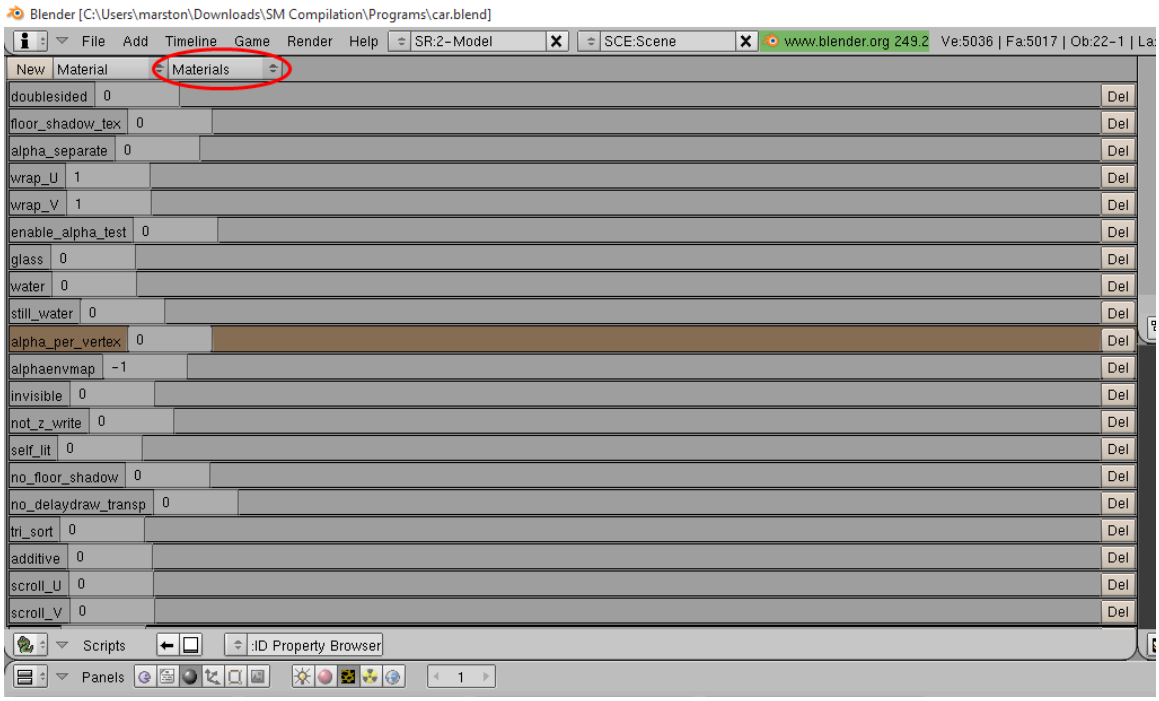


The **Reflection** block has the car's shine image.
The **Lightmap** is left blank on most cars.
But the **Specular** does have the image assigned to it.



The Movies import scripts generates a flag menu in the Blender's help menu. This menu gives you lots of options over your (.MSH), including some material flags.



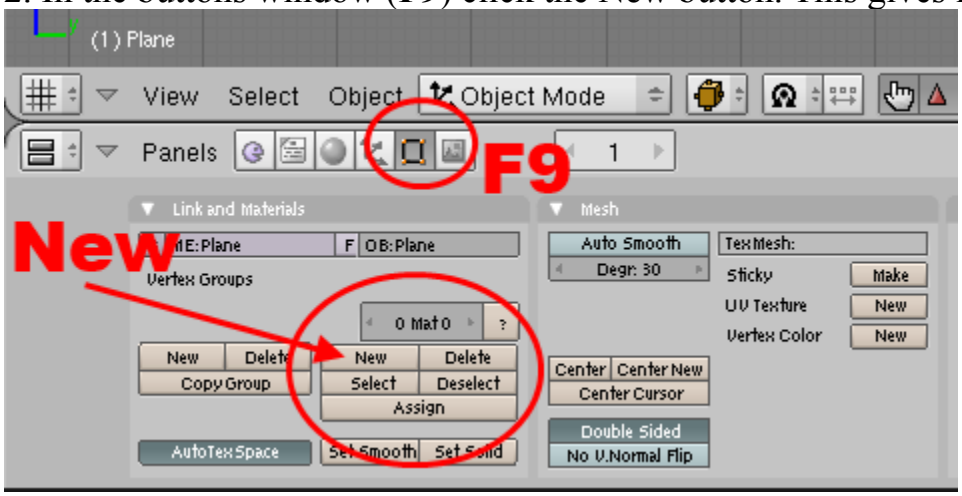


However this menu can only be created after importing a (.MSH) file. So the only way to have these options is to export your model in a sub-state (.MSH) providing that you are finished with everything else but setting flags. Then re-import it back into Blender just to gain access to the options from this menu. Once those flags are set you can go on to export the final version. Which is a lot of extra stuff that might as easily be accomplished in MED

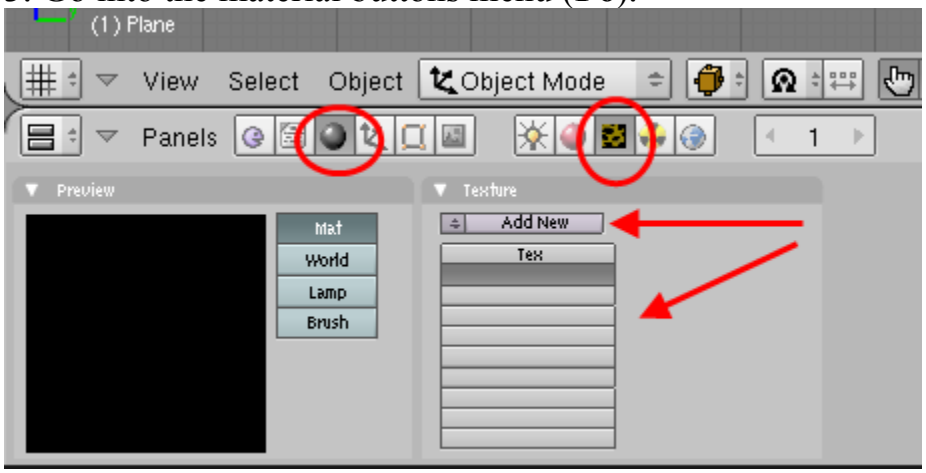
(The Movies Editor) or Meshmanip. And I have yet to learn how to successfully set alpha from this menu. I also suspect that it isn't set here because in the *Buttons* menu alpha is revealed to be active. Does the export script look to the *Buttons* window to know if alpha is present? It was never explained.
Also there is still so many unknowns in a (.MSH) file. Many more flags that we suspect are there but do not know their function or reason yet.

The steps for giving an object a Movies game material is:

1. Make sure the object is selected.
2. In the buttons window (**F9**) click the **New** button. This gives it a material.

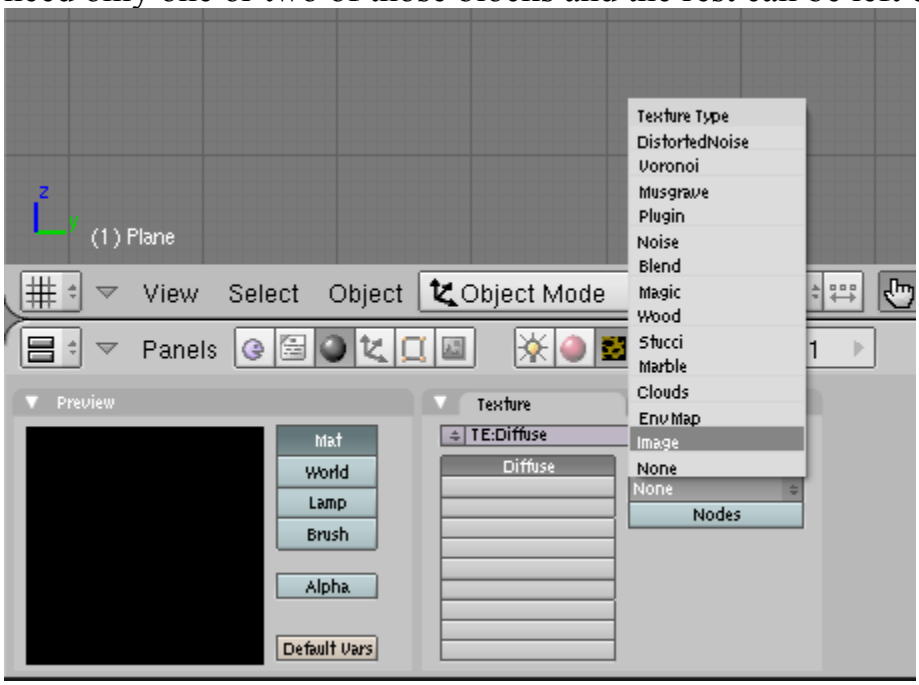


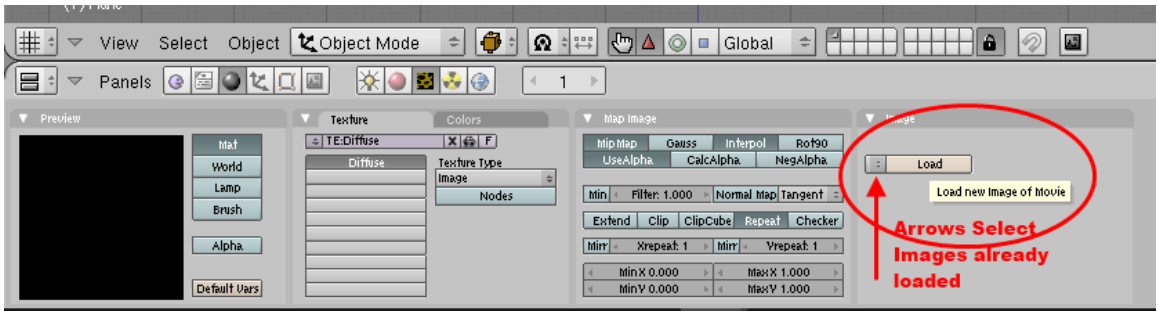
3. Go into the material buttons menu (**F6**).



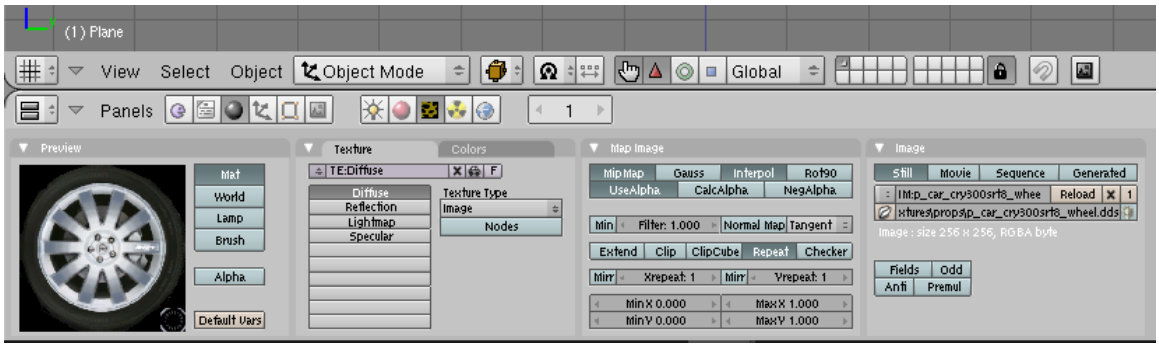
4. Create four Tex blocks by hitting the **Add New**, also clicking on the bars in the stack.
5. Rename these four Tex blocks to **Diffuse, Reflection, Lightmap** and **Specular.**

6. Assign the appropriate images to each of those blocks. Note: you may need only one or two of those blocks and the rest can be left empty.



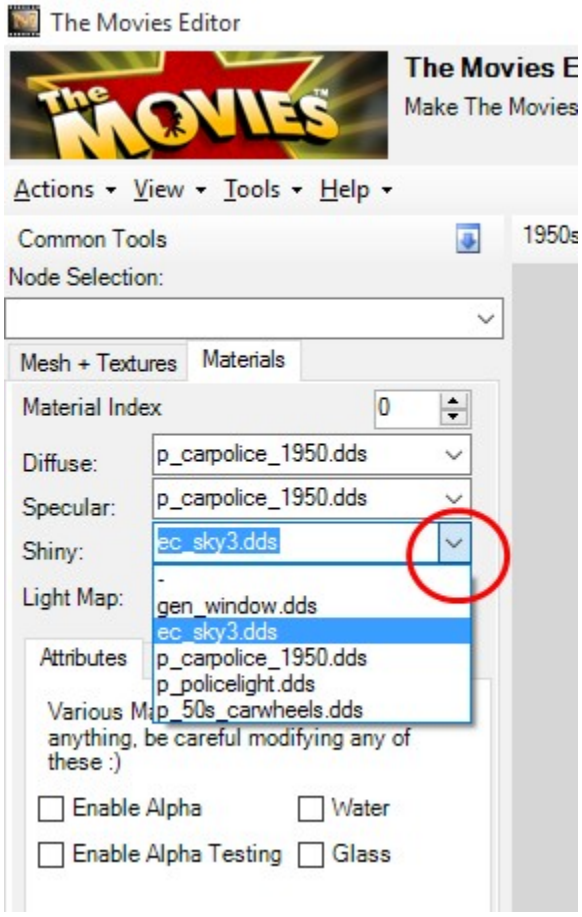


When images are loaded or selected into those stacks, they will show up on the left.



There have been times that the materials were not written to the model correctly after export. This usually happens with large set files. The good

news is that the materials were still written anyways, and the file still knew the images were called by the model. I would learn this by importing the finished model into MED. Also sometimes what was assigned to the **Diffuse** ended up in the **shiny** or **lightmap**. (shiny is *reflection* in MED) If you look back to your .blend file you will see everything looks normal. Fortunately in MED you can easily change which image gets assigned to each of those four blocks. This is surely not the export script's fault but Blender's memory which is a tad buggy for it's strength.



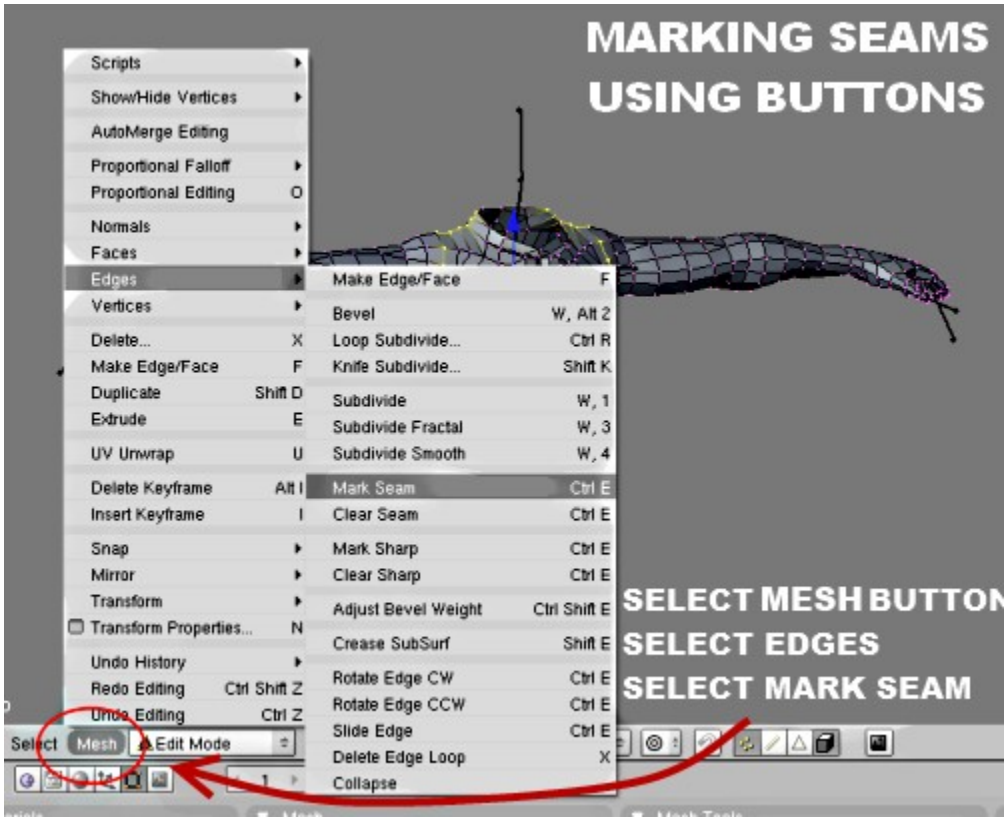
Most of the fine tuning for materials can be done in MED or Meshmanip. Those programs make it much easier. But the one thing MED cannot do is create a *material* from scratch. And it can have a new texture loaded into it for your model, but only to replace an already existing one. Meshmanip has the power to not only create *materials* from scratch but also replace images and load new images on the fly. It's downside is that is does not have a very good 3D render and you won't be able to see your *material* settings in the 3D preview. Something MED does well enough.

6. UVMaps and Baking

We've already made a UVMap for the Dice model in chapter 4. Here are some more notes about making a UVMap and *Baking* a new texture image. The process is about how you want the object to utilize the texture image. UVmaps reveal what verts are where on the image. New objects do not have them yet and need to be made. Imported game models often have them (The Movies Game Models will have them). And if they have them you can either move it in the *UVMap* window. Or in the following case, we will just make a new one, and forget the old one.

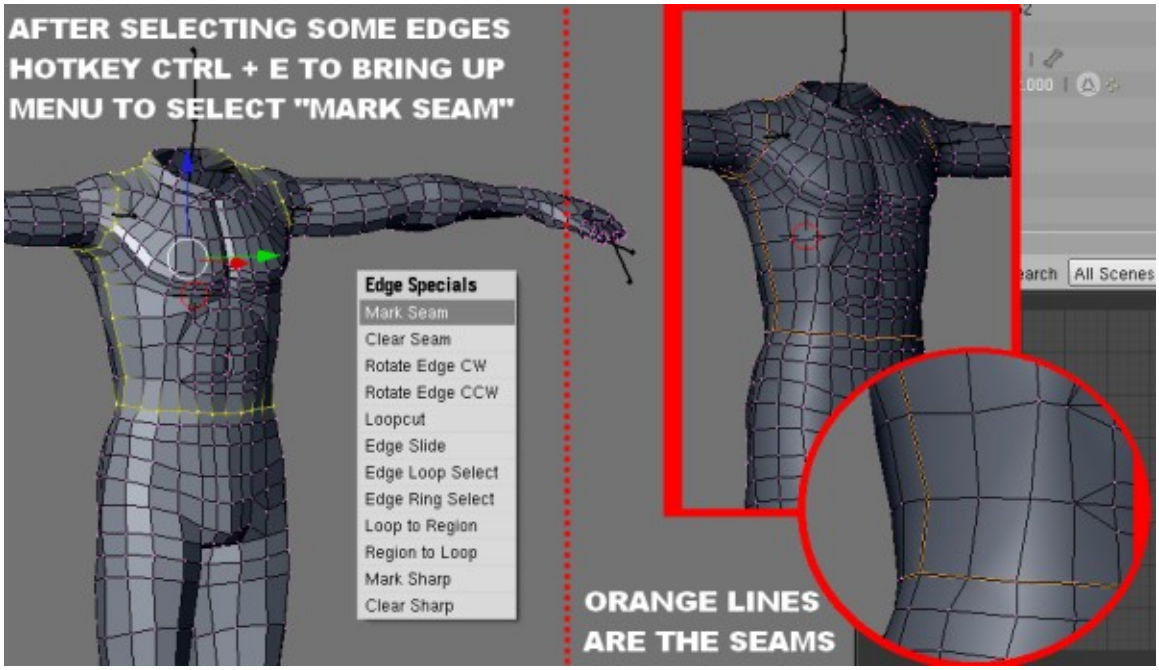
1. I will take some picks of edges being selected and give those edges a **seam** assignment. Those edges will turn orange in the 3D window.
2. Have all of the model's verts selected.
3. Click a button called **UVMap Unwrap**.
4. Arrange it in the *UVMap* window to how I want it.
5. Add some **lamps** all around the costume object.
6. Add *materials* to the object and change the color of the *materials.* The colors will *bake* into the image.
7. In edit mode select only the verts I want to have a specific color, and by clicking a button called **assign**.
8. Click on the object so it is selected, and then go into the *Buttons* window and click on a button called **Bake.**
9. Save the image as a new texture. Blender does not save images in (.DDS) format that the Movies Game requires. So it will have to be loaded into a Paint Program and then saved in (.DDS) format.

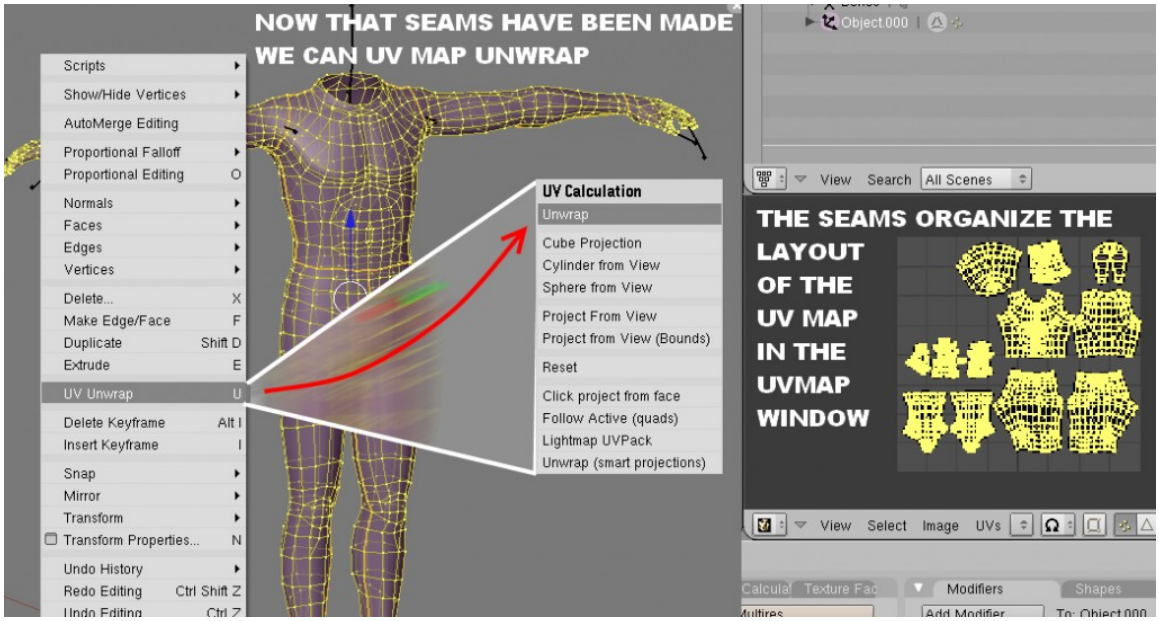
Seams are the borders of the *UVMaps* we create. Select an edge, and most likely a whole bunch of edges, then select **MESH** button (next to mode box), from the popup menu, select **Edge**. Select **Mark Seam**. Or the easy way is to use a hotkey popup window. **CTRL + E** and the top should be **Mark Seam.** You can also clear a *seam* that was misplaced. Or if you no longer need anymore *seams*, select all verts and select **Clear Seam**.

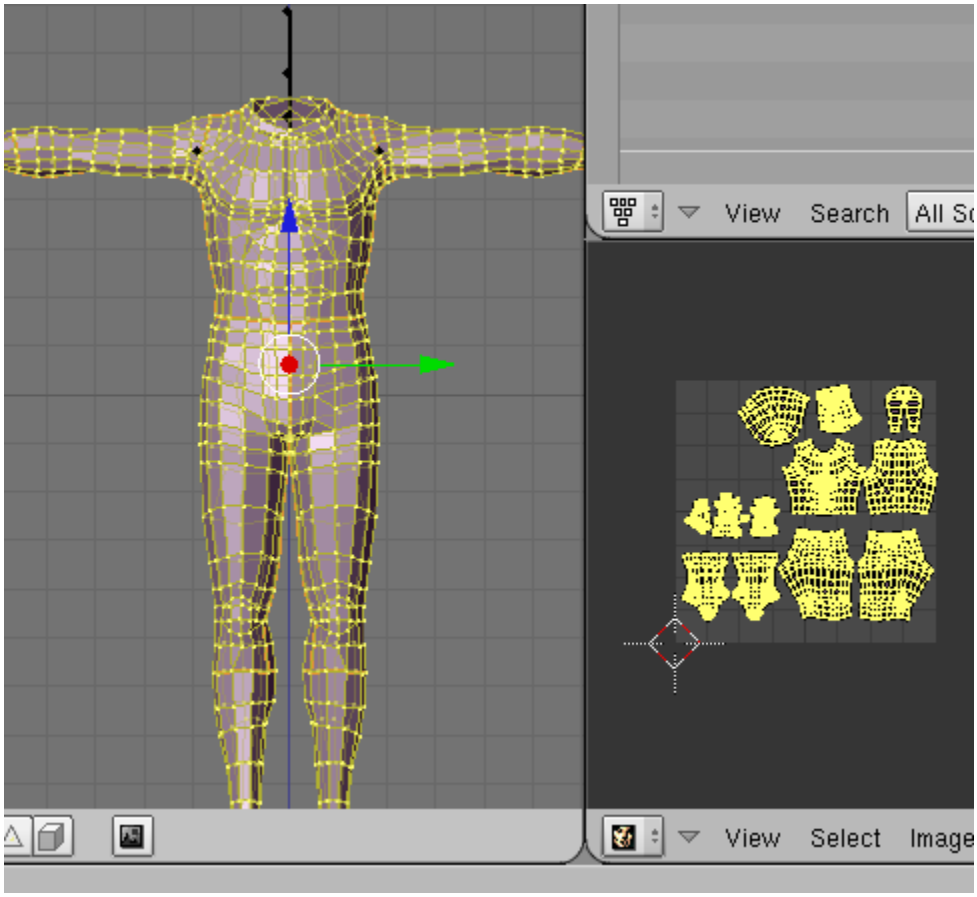


Once the *UVMap* has been unwrapped in the *UVMap* window it may be any which way. I will find the two portions of the legs and put them side by side. I will get the chest and the back and put them side by side. Same with the hands and feet. Looks better that way but not real important. It is easier to add further details in a paint program that way. Like a superheroes symbol on the chest.

There are several ways to *Unwrap* the map depending on your need. Some will not require a **seam** to be made. *Project From View* is good for walls. Once the view in 3D window is facing head on (for a wall object) you could select **Project From View**. Then it will appear over say a brick image in *UVMap* window. Now the object is a brick wall. In this example I will just use the regular unwrap choice.


AFTER SELECTING SOME EDGES
HOTKEY CTRL + E TO BRING UP
MENU TO SELECT "MARK SEAM"
Edge Specials
Mark Seam
Clear Seam
Rotate Edge CW
Rotate Edge CCW
Loopcut
Edge Slide
Edge Loop Select
Edge Ring Select
Loop to Region
Region to Loop
Mark Sharp
Clear Sharp
ORANGE LINES
ARE THE SEAMS
All Scenes


NOW THAT SEAMS HAVE BEEN MADE
WE CAN UV MAP UNWRAP
Scripts
Show/Hide Vertices
AutoMerge Editing
Proportional Falloff
Proportional Editing O
Normals
Faces
Edges
Vertices
Delete... X
Make Edge/Face F
Duplicate Shift D
Extrude E
UV Unwrap U
Delete Keyframe Alt I
Insert Keyframe I
Snap
Mirror
Transform
Transform Properties... N
Undo History
Redo Editing Ctrl Shift Z
UV Calculation
Unwrap
Cube Projection
Cylinder from View
Sphere from View
Project From View
Project from View (Bounds)
Reset
Click project from face
Follow Active (quads)
Lightmap UVPack
Unwrap (smart projections)
Object.000
View Search All Scenes
THE SEAMS ORGANIZE THE
LAYOUT
OF THE
UV MAP
IN THE
UVMAP
WINDOW
View Select Image UVs
Modifiers
Shapes



You can see the UVMap in the UVWindow and I have arranged it in the order I like. It will need an image to bake on. Blender will make one with the click of a couple buttons.

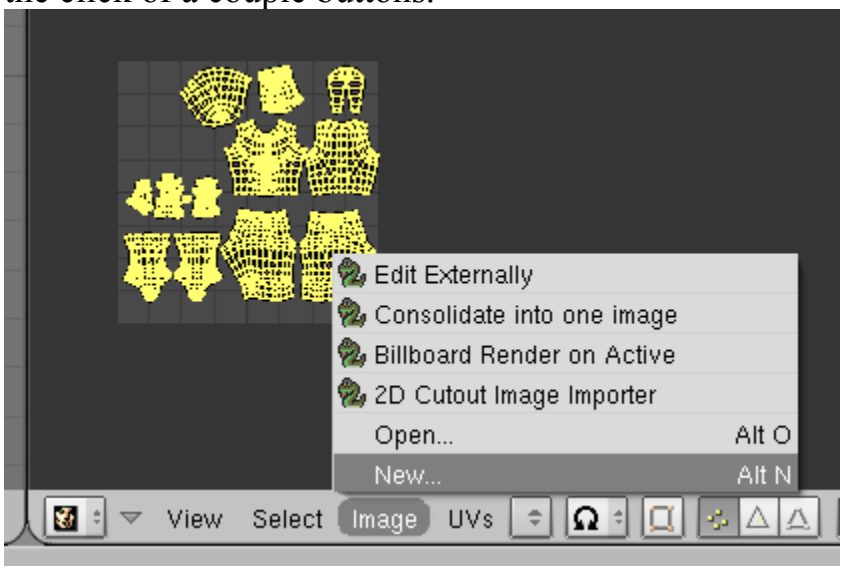
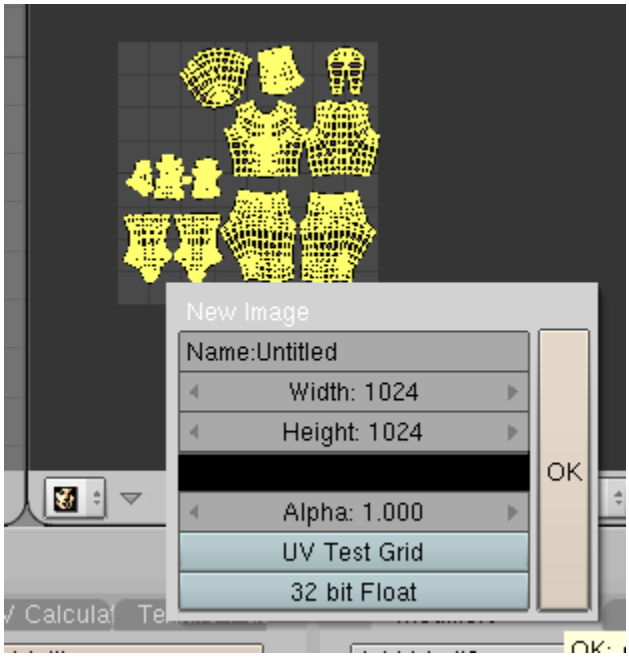
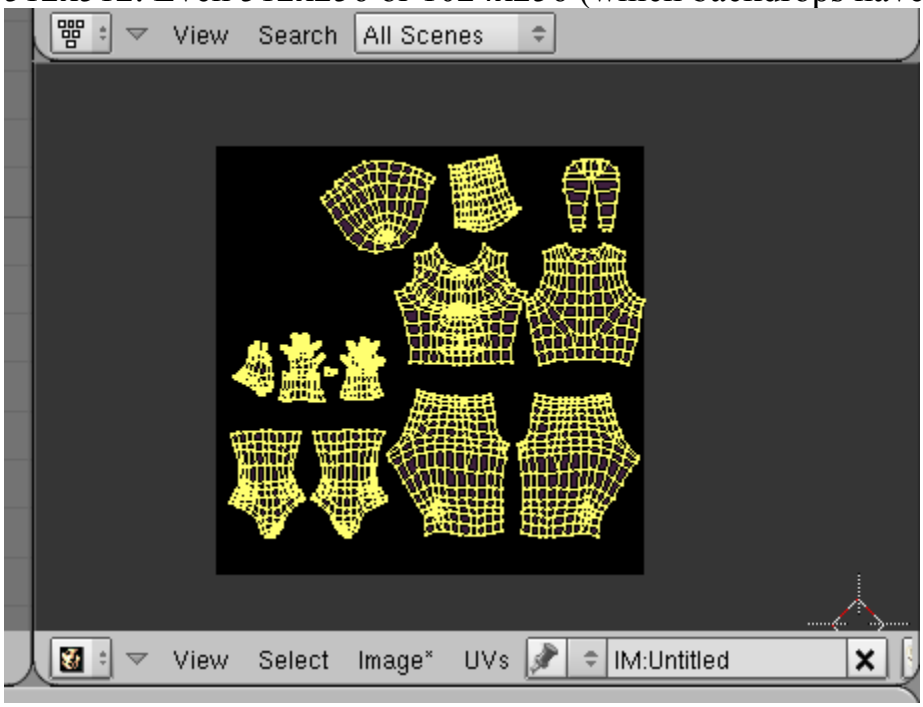


Image. **New** and select the resolution.

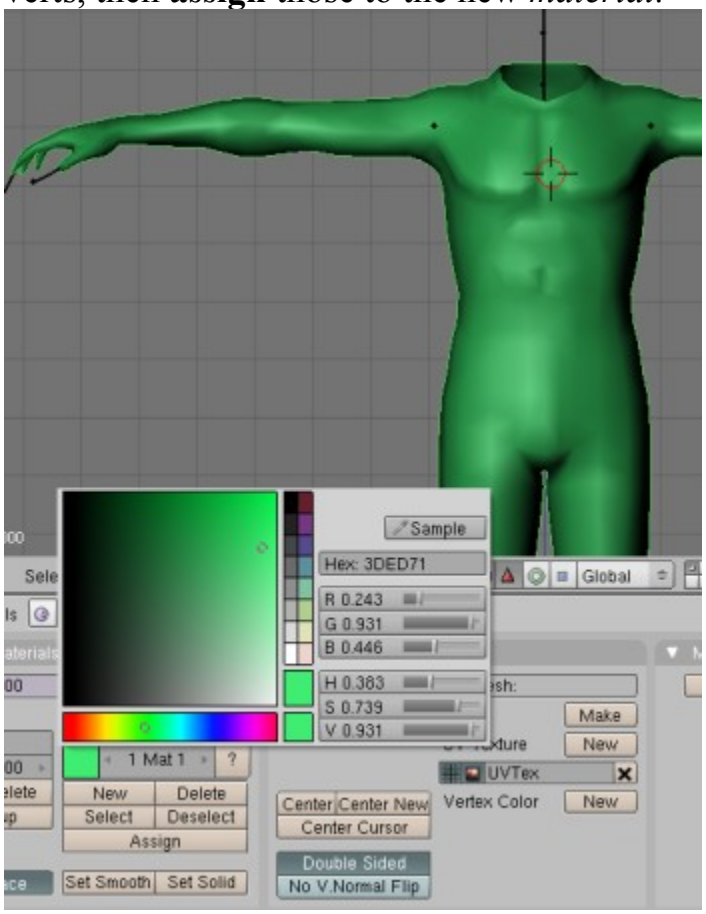


The size is important. 1024x1024 is the best and is chosen by default. You could also choose 512x512 (Standard Movies Game costume size). Most important is the 4x4 multiplier. 32x32 or 64x64 or 128x128, or 256x256, or 512x512. Even 512x256 or 1024x256 (which backdrops have).

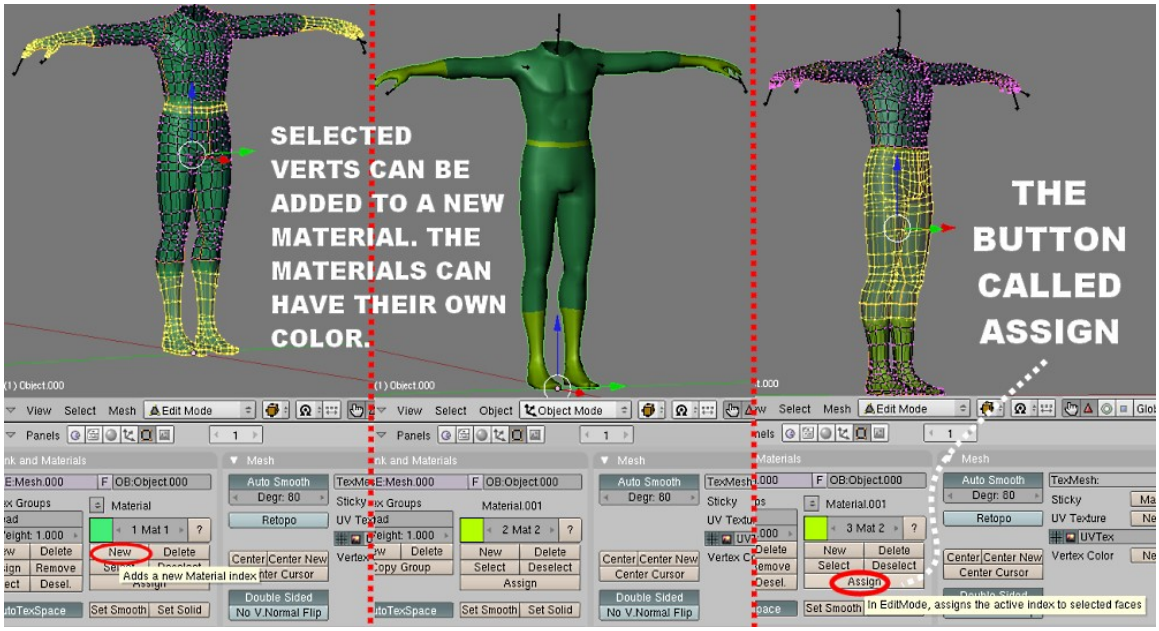


The black square is a blank image. A button in UVMap window would show it's *alpha* if you needed it to. But its not important right now.
(Being lazy and this is for newbies).
The **New** button in the *Buttons* window **(F9)** will add a new *material* to the

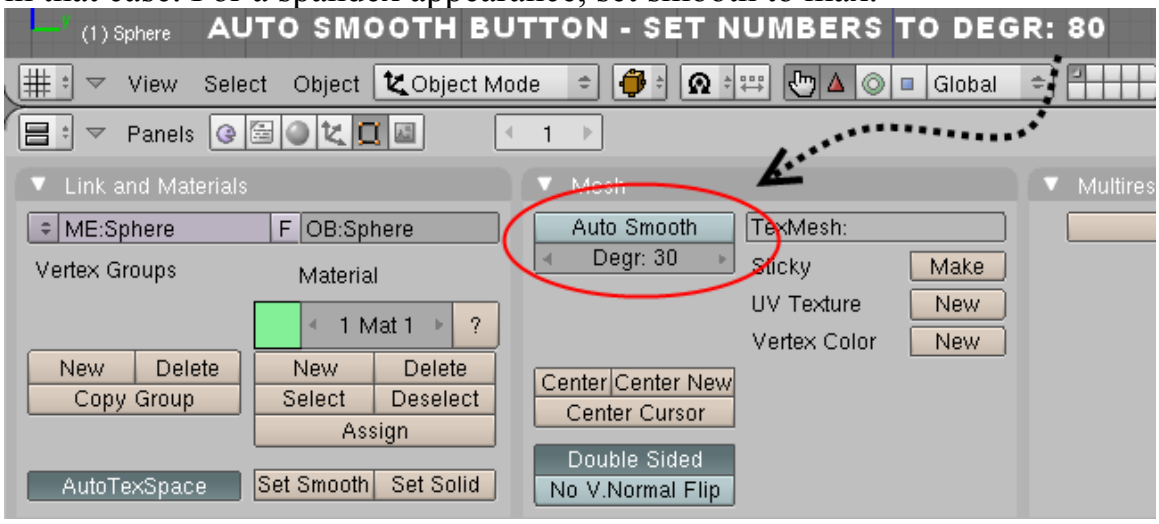
object. Make sure the object is selected first. You can choose the color of the *material* by clicking on the color box and a popup window appears. These colors will be *baked* into the new image. In *Edit Mode* you can select several verts, then down in buttons window select the **Assign** button. That part of the object, where the verts were selected, will now have that color appearance. For another color, add a new *material*, then select different verts, then **assign** those to the new *material*.



Click the color box for a popup color picker



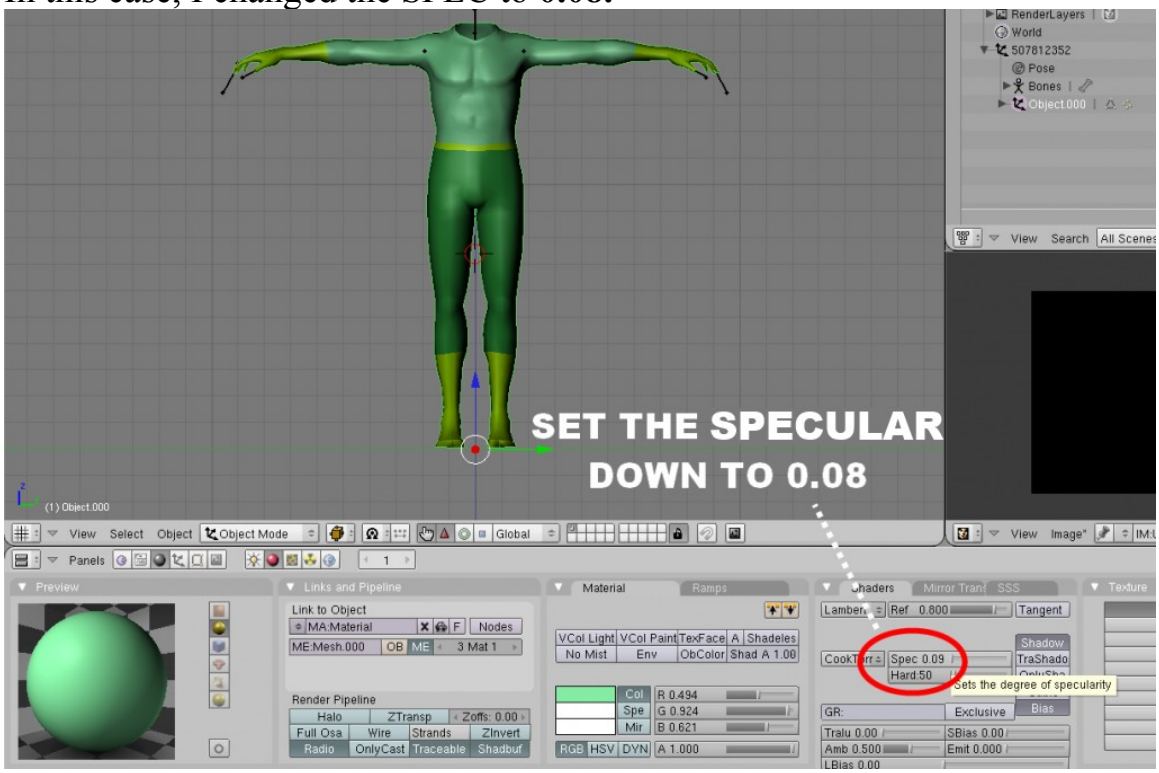
As mentioned earlier, the *bake* will need to a have a smooth factor. Just because we clicked the **Set Smooth** button does not insure the bake will be smooth. Use the numbers below the Auto-Smooth button. I do not press the button. I just set the numbers below it to a higher setting. Set the numbers below to 80.0 to make sure the *bake* is smooth. If you are baking hard surfaces, like a robot body, then you won't need the smooth. Leave it alone in that case. For a spandex appearance, set smooth to max.



Here is an important note. Blender doesn't like *materials* that were created during the import. And if the object already had a uvmap with an image, we can not use the image. It also could be a black square for the image. That happens when the (.MSH) file was imported while not being in a location the scripts wanted. Or the texture was missing in the data\textures folder. It's not needed, and you might not have planned on using them. Blender created

a dummy image in such a case. This topic requires that we are editing the *materials.* In the example pictures I used the *cos_m_50s_und.msh* file. First thing I did was delete the material. And the second thing I did was rid the image file. (Actually I have the object saved in a .Blend file with no materials or images saved to it. And here I just appended it for the example. Which is what I needed.) If you are using an object already found in the game, make sure you delete the materials before adding new ones. And make sure the *bake* is over a newly created image (and not an image that was imported).

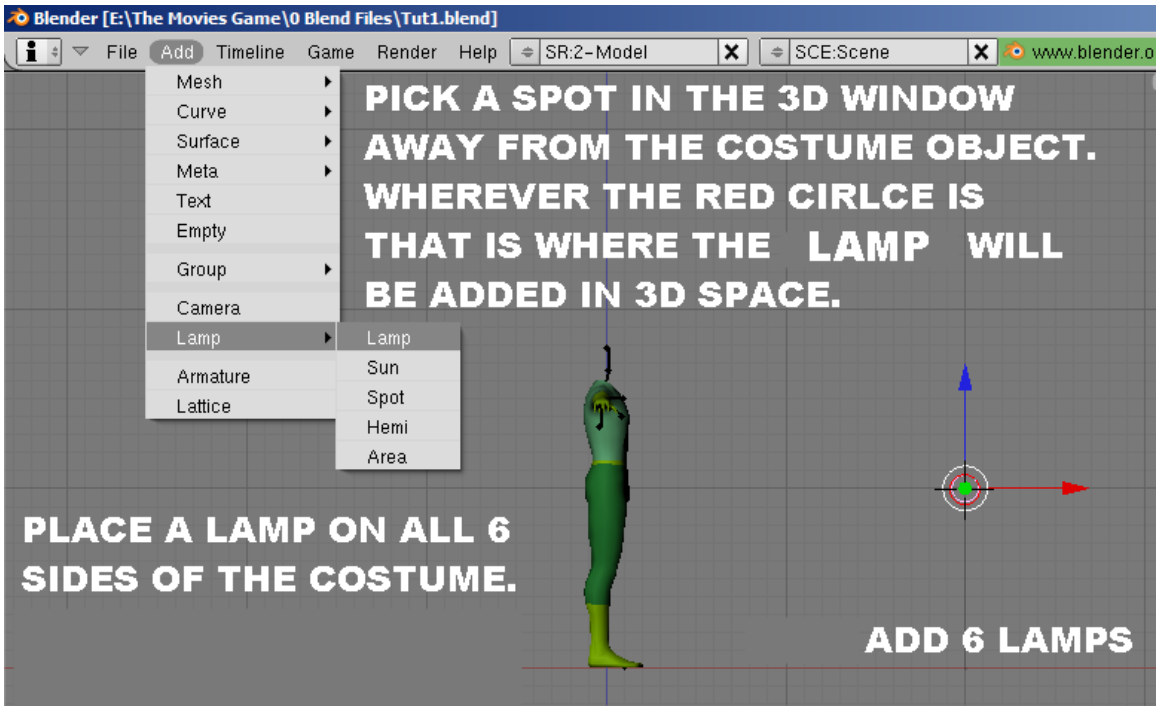
Light shines brightly on a new *material.* So we need to turn down the **specular** of the material. Here is a pic of where to edit the numbers. In *Buttons* window everything is a slider, but if you aim the mouse skillfully over the name, it will become a number you can change instead of a slider. In this case, I changed the SPEC to **0.08.**

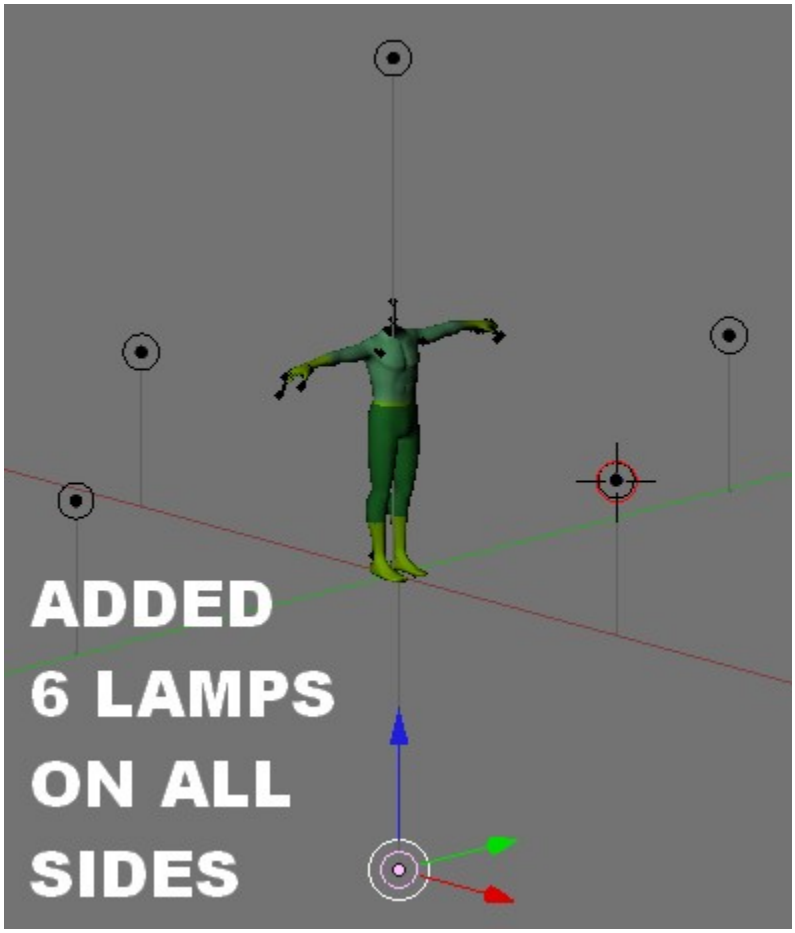


We need **Lamps** to b*ake* the colors into the images. Lamps illuminate the colors on the model and the *baking* will imprint on a blank image how illuminated the color is. Lamps provide where this illumination will occur. We need several lamps on all sides of the model to ensure that the colors are well illuminated.

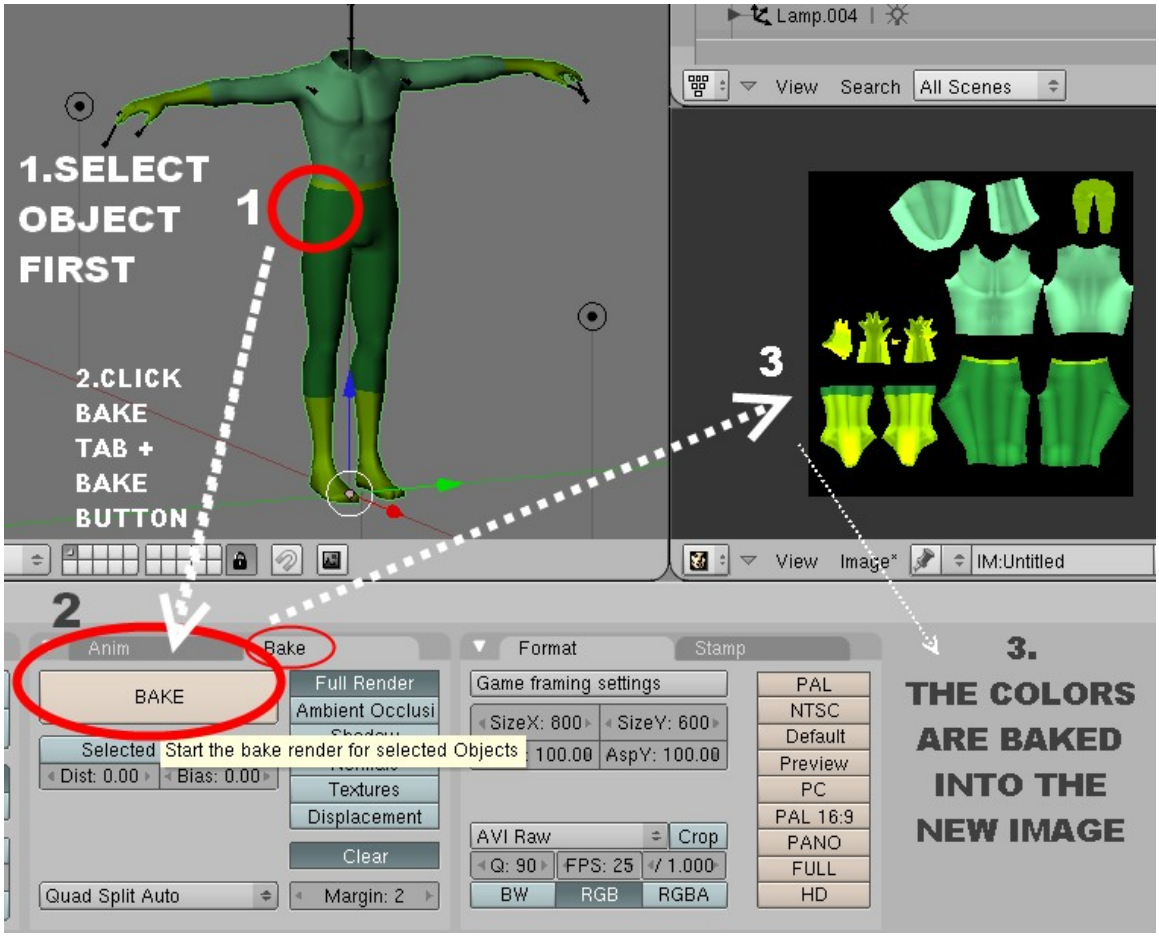
There are several types of lamps. They each have settings that can be adjusted. But the default settings will be fine for this task. As long as the

lamps are not too far away or too close to the costume body object. The idea is to have light hit every side of the costume from all angles. 6 lamps will do. Above, below, in front, behind, the left and right. The 3D cursor is where a Lamp will appear when you add one to the scene. You will also see in the outliner window the Lamp object name. Lamp's visibility can be turned on or off as well. One trick is to use the "Make A Double" function with the lamps. Once you add a lamp, you can hit **SHIFT + D** key to make a double and then move it to the other side of the costume.



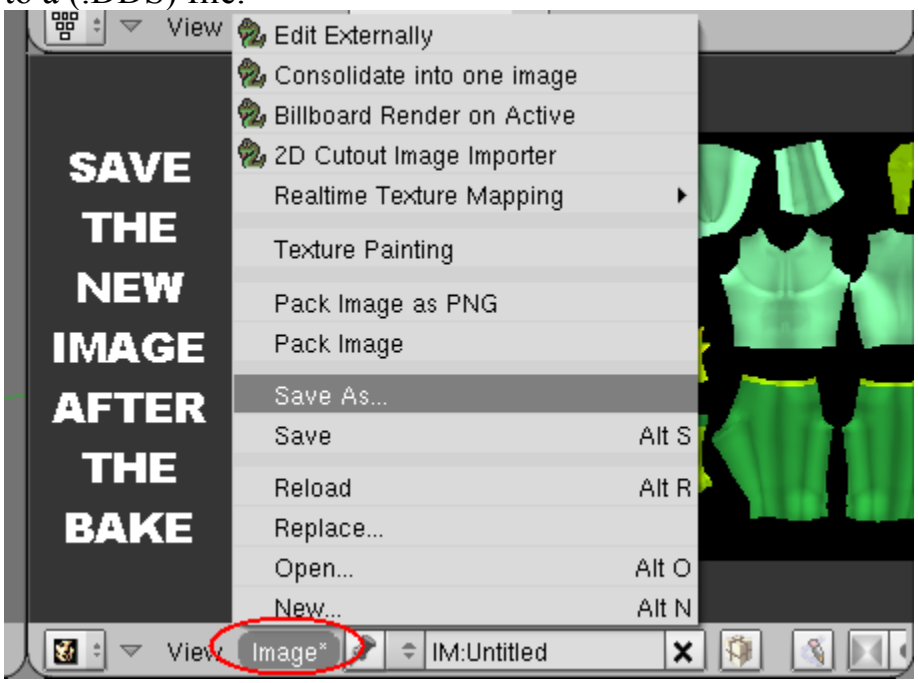


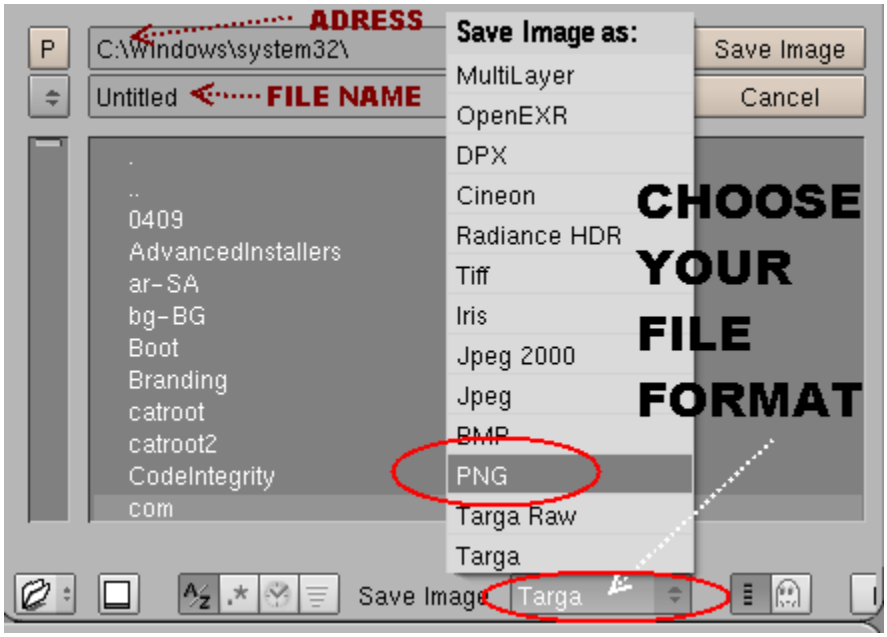
Whatever object that is having a texture *baked* must first be selected. Then in the *Buttons* window you may not see the Bake button at first. There is a tab shared with **Bake** which is called **Format**. Instead of format select the **Bake.** Now a big BAKE button is shown. Next to this button is a stack of smaller buttons. These fine tune the baking process. We want **Full Render.** At the bottom is a button called **clear**. It may be you are baking a texture several times over, in which case you would leave **Clear** unclicked. But leave it on for a one time bake, or if you do not like the outcome you can do it again, (as many times as you want.) Hit **Clear** button and every time you do a bake, it will overwrite the last one. Perhaps you find the lamps are too close or too far or you need to change the color of the light the lamps are giving off, or need to add more lamps or less lamps. In such cases you can do another bake.



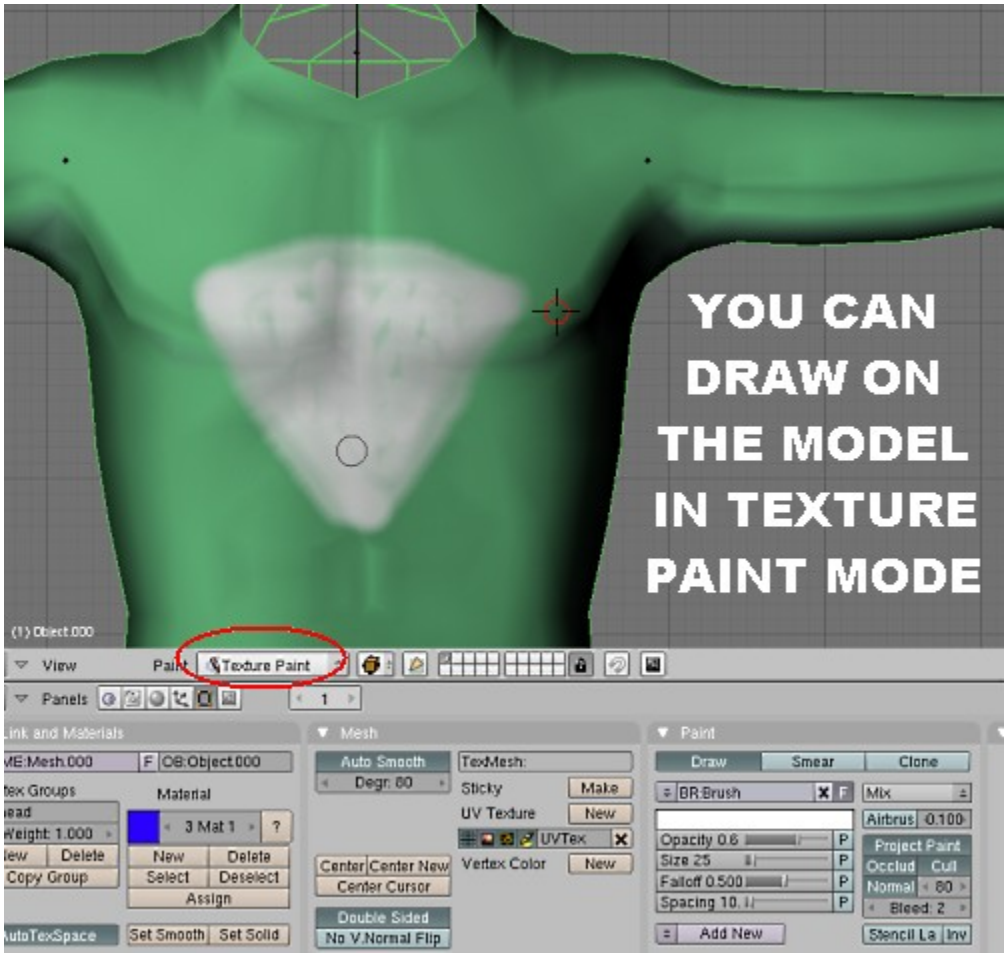
You will see the results in the UVMAP Window. You can also see the new texture over the costume model by selecting the **Texture Paint** mode. Actually there are a great many ways to see the texture over the model. If you had a camera facing the scene you could hit the **F12** key to render a frame. Or you can use a small floating popup menu called the **View Properties Menu**. Either way you can see the result. Since, in this case, I still have the colored materials over the model, the result doesn't look that much different then the model's assigned *material* colors.

The next step would be to save the new image. Chose a file type and a location to save it at. Unfortunately Blender does not save images as (.DDS) format. It will save it as a (.PNG) and in a paint program you can convert it to a (.DDS) file.





You can draw over the model and have the brush strokes saved on the image. Every time you draw on the model, you will see the texture image also is drawn on. If you do not like the last brush stroke then you can press the image-reload button (in the uvmap window). If you like the new appearance then you can save the image over the last save. Be careful and make a backup if you need one.



As with any project make sure you save your work often. Blender will make a backup of your last save every time you update the save. It is in the location of your .Blend file. But it will have a number in it. (.1blend). You can tell Blender that you want several backups. As many as you want I think.

In this example it was as if a new costume was being made. And there would be a few more steps to make this costume one you can export. Since a new UVMap was made, the texture you baked would only work with this model (this costume), which means you need this model. Movies Game costumes do not have more then one material. You would then delete all the colored materials you made, or just delete all but one. Then use the last one as the one the game will need. Or if you deleted them all, make a new official material the game will need. The last or new material will need to have texture names before you export it.

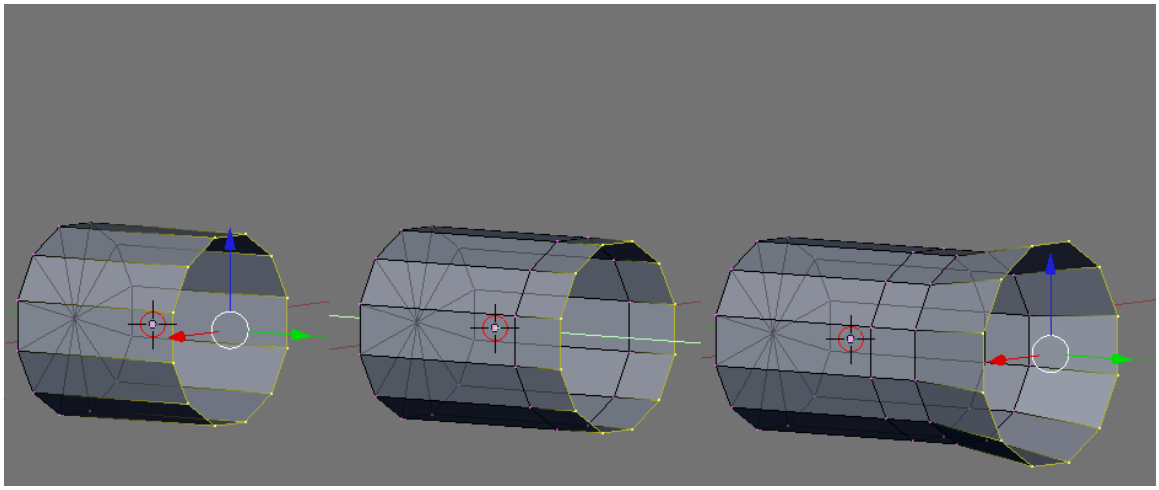
On the other hand perhaps you just want to make new textures for existing Movies Game models and don't need the costume object. An example would be a Body Paint, or a skin color for a latex head. Or another color option for

any of the game's costumes. In that case you would not make a new uvmap but use the one the model already has. But that presents a problem. Movies Game costume textures share space on the images. The arms share the same part of the image. The hands and feet also share the same part of the image. And when you set up your cameras and material colors, then hit the **Bake** button, the hands feet and arms bake twice, one on top of each other and often this will mess up the image. If you were doing that, I would delete one of the hands, one of the feet and one of the arms. Then that portion of the image will only get baked once. Or you could set the new image size to 1024x1024, Then *scale* the uvmap back down to 50% the size. Move the UVMAP to the top left of the image. Then move the portion of the uvmap for the second hand, second foot and second arm off to the side. when you save the image, in a paint program, you can crop the top left side of the image (25% of the image). As long as no uvmap overlaps another part of the uvmap.

Tip: When *scaling* an object, or a UVMAP, if you hold down **CTRL** button you will see some numbers at the bottom that will tell you by how much you are growing or shrinking the item. **CTRL** button will scale in increments. When you get to 50% then you can stop easily. If you needed a very slow increase or decrease, you could hold down the **shift** key during scaling. Also when moving the uvmap, you can hit **G** (Grab) button, then hit the **Y** button to confine the movement to up and down. With **CTRL** button pressed it will move up or down in increments. **X** for left and right increments.

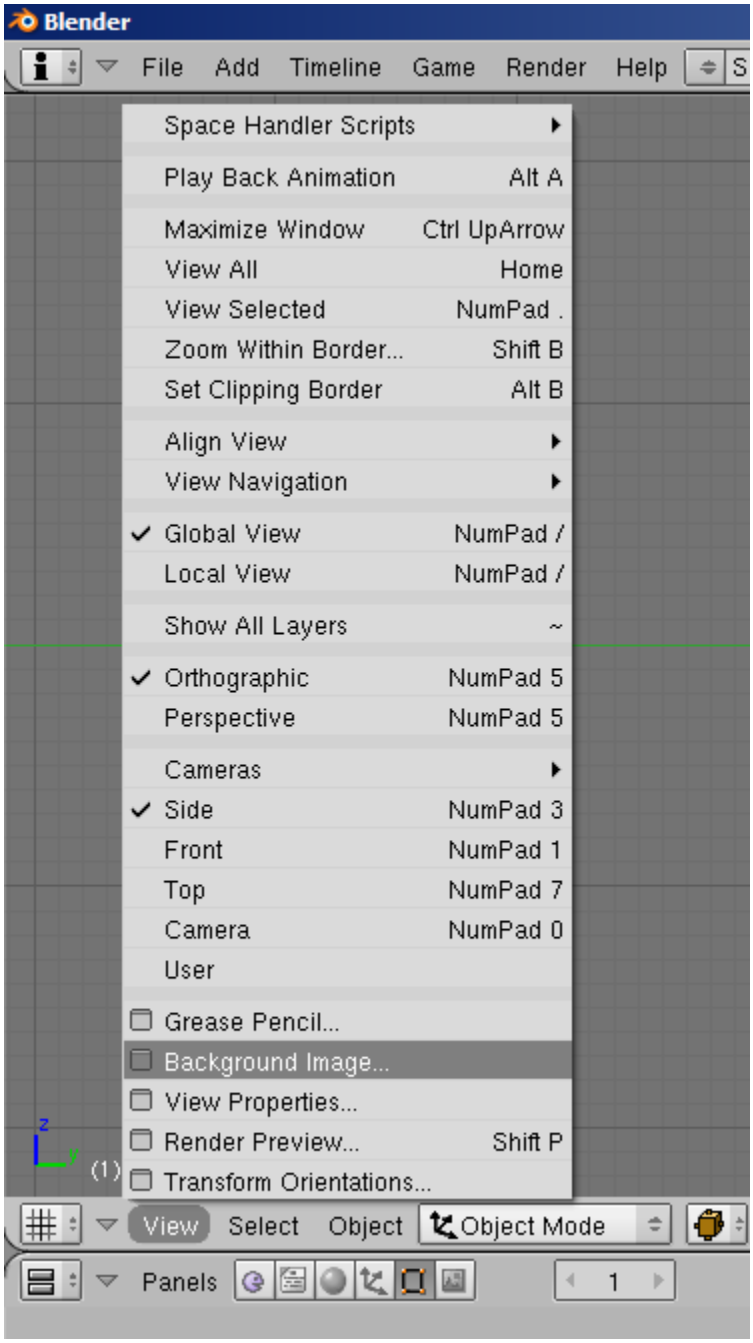
7. The Extrude Function

Hotkey E - for extrude. Any selected verts will get doubled and you can move them to a new location in 3D space. These verts will be connected to the originally selected verts, with faces even.
In this picture I added a cylinder object. Went into *Edit Mode*. Deleted the vert that made the wall end. Then selected the open edge. Now press **E** for **extrude**. (You don't have to delete the vert, all faces will continue to remain using *Extrude*. Try it both ways. Just an open edge and also a closed object.)

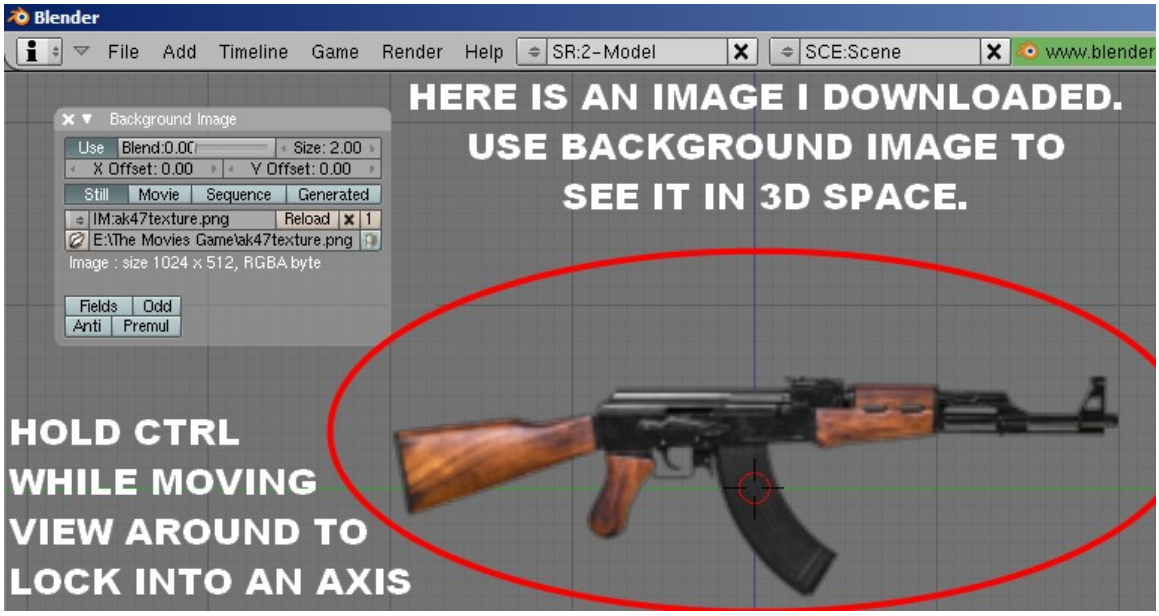


Erase everything. Fastest way is up to File button, then select **New**. Or Hotkey **CTRL + X.**

First I found an image of a weapon online. One photographed from the side view. Then I loaded it into a paint program and made sure the canvas size was 1024x512. A multiple of 4s which Blender will need or it will grind to a halt trying to render the sucker. If Blender freezes then make sure the image you use is the proper size. Blender will eventually (if it doesn't crash that is) catch up. But stop this mess immediately. Better to close the program, fix the image and start again with the proper sized image. If Blender won't even close use Task Manager to close the program. The newbie mistakes will never happen if you know what to avoid.
Next I loaded the background image I downloaded online. This is done in the *3D* window using the **View** button at the bottom. Selected **Back Ground Image.**



A popup menu. Click **Use**, then the **Load** button. Navigate to the image (adjusted in a paint program). Select it. The image shows up but may be too large to use. At least for me, I use the size setting to shrink it down a bit. Also it may be see through. The **Blend** slider tells how visible it is.

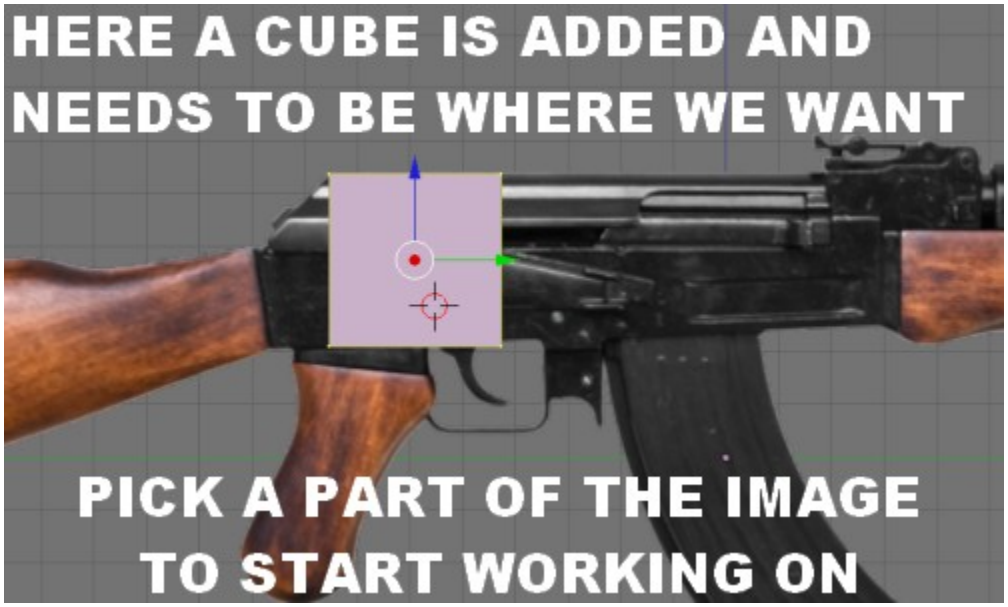


Holding down **CTRL** key while using middle mouse wheel as a button, will spin and snap the view into a locked axis view. This is very important and handy. Many adjustments must be made while the view is looking head on along an axis. Either top down, left side or right side. Front side or back side.

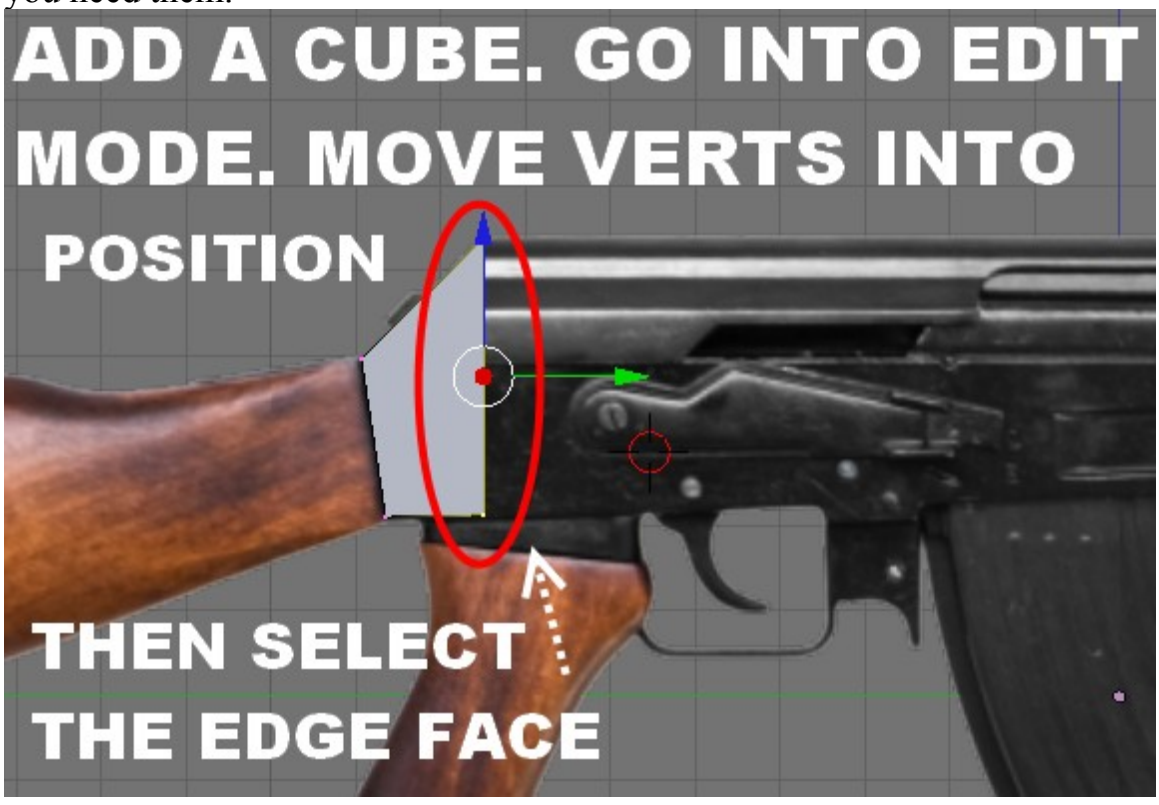
When using a background image, it will only become visible when you snap the view along an axis. This means, in this case, the grid floor is completely flat at the center on the screen. Also make sure you are not at a 45 degree angle (meaning at least one of the axis lines faces you and looks like just a dot and not a line).

If you move the view, which you will do from time to time, snap back into locked position along an axis using the **CTRL + Mouse Wheel Button** maneuver.

I added a *Cube* object. At top next to File is a button called **ADD**. From the drop down menu I selected **MESH**, then selected **CUBE**. Any object you add will be added wherever the 3D curser happens to be.

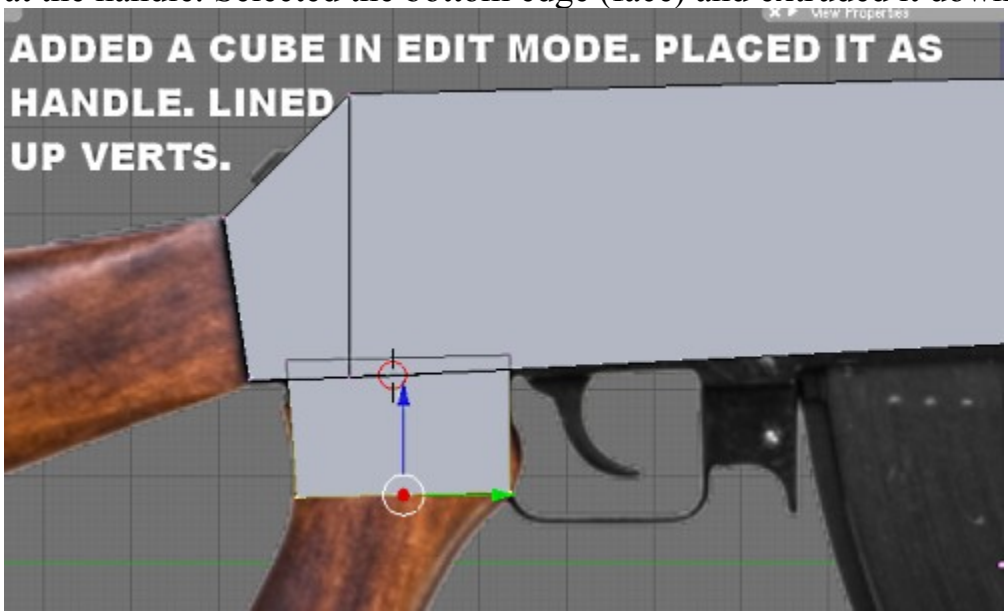


In edit mode move the cube to some starting position. Place the verts where you need them.

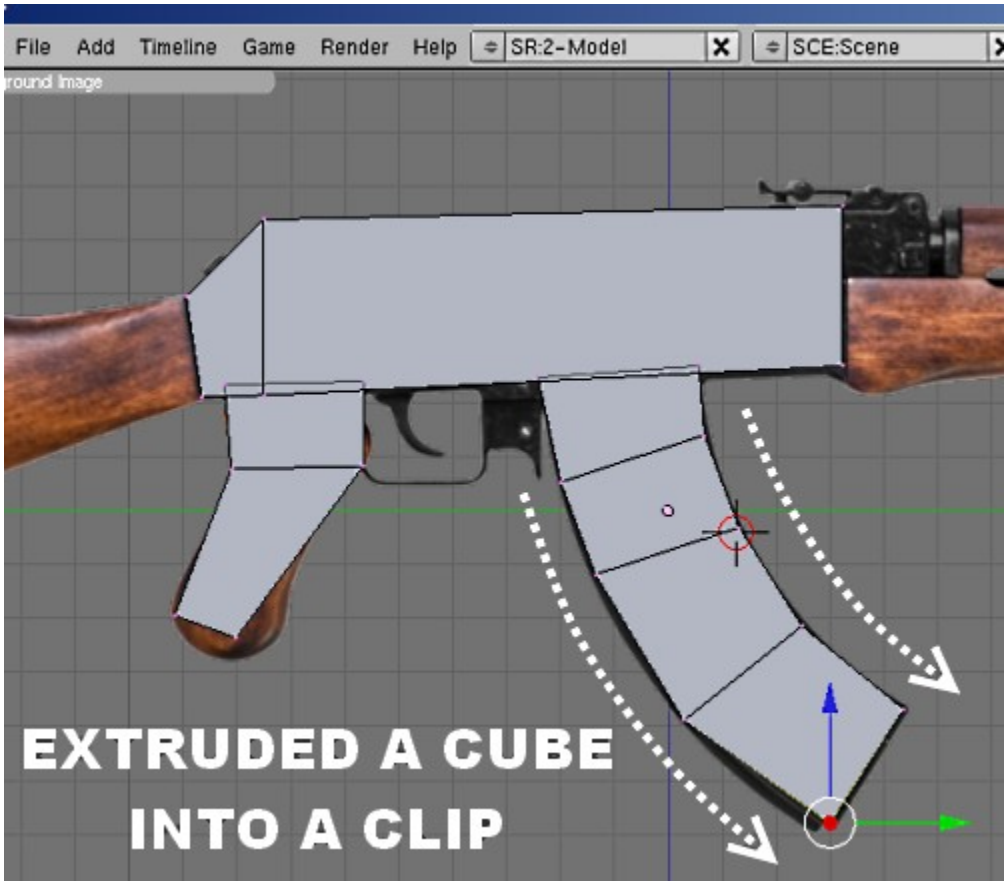


After selecting the edge I want to move outward I hit **Hotkey E** for *Extrude*. Then I hit the **Y** key to force its movement to move out along the Y axis. When I get to the end I further move the verts into better position.

Next I added another cube in edit mode. When adding another object and being in edit mode, the new object will be in edit mode too. Position it to be at the handle. Selected the bottom edge (face) and extruded it downward.

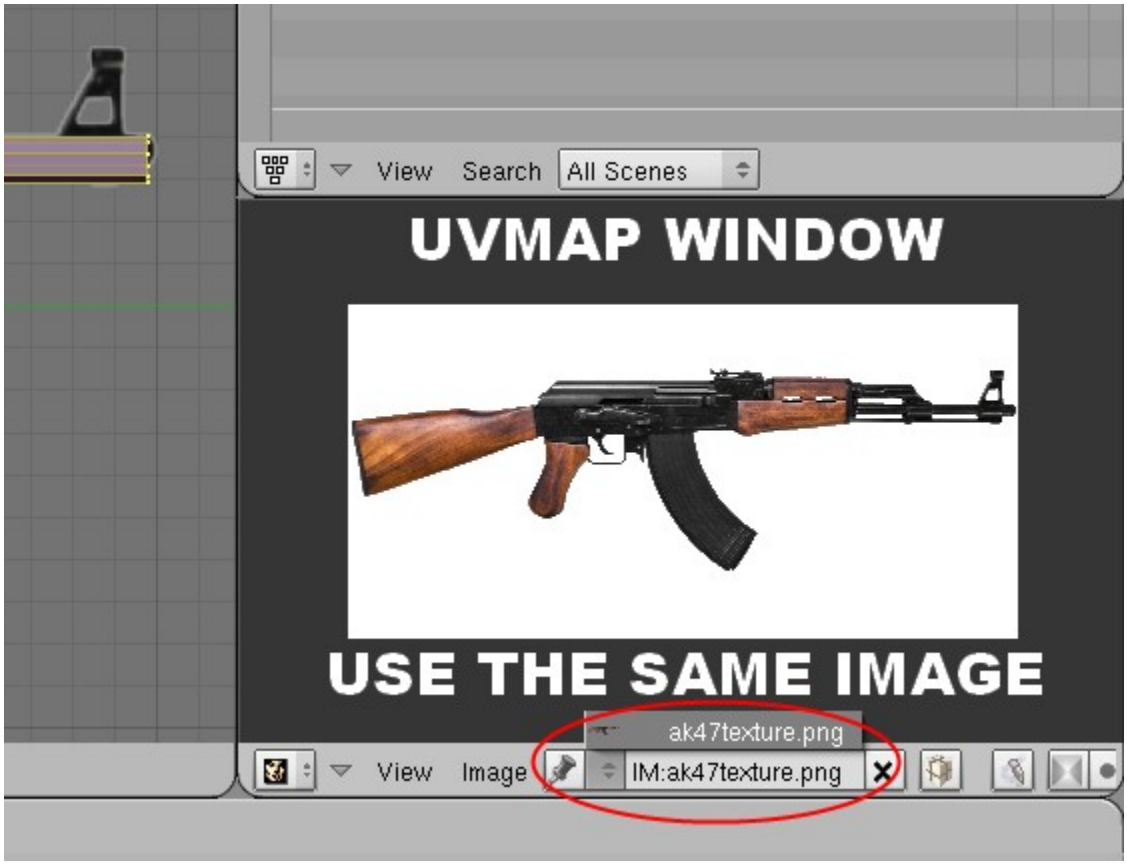


Now I choose the clip to be made. Add another cube object in edit mode, place it at the area of the clip. Pick the bottom face and extrude it several times to produce a curved effect.

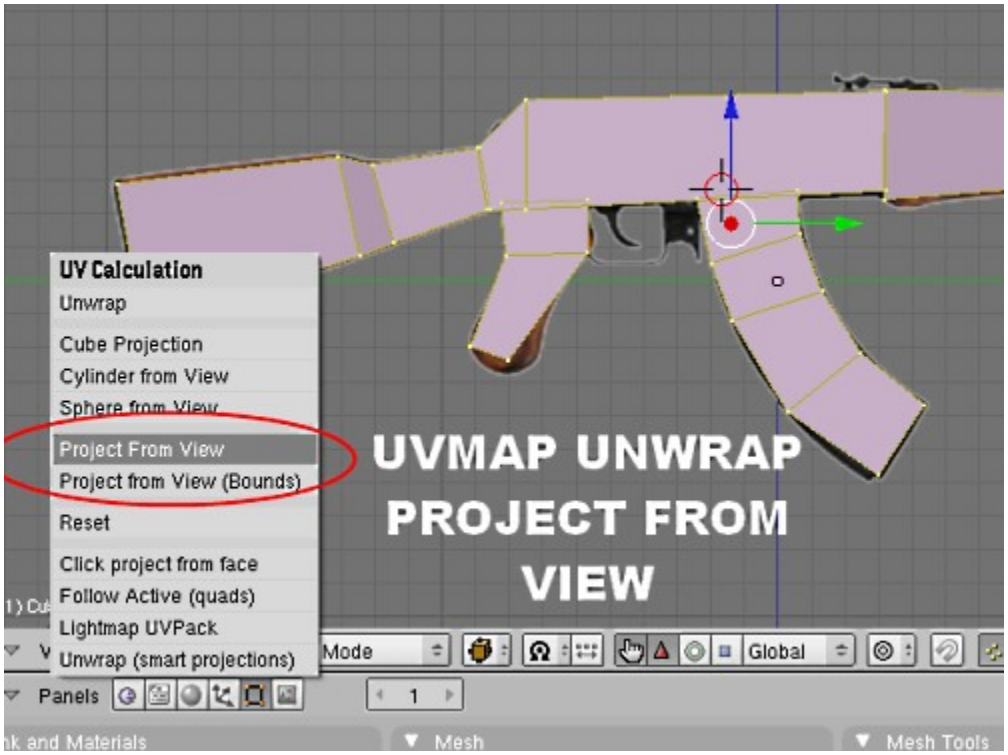


The game already has an AK47 and I don't want to marry this thing. It was fine for showing how to extrude objects into place.

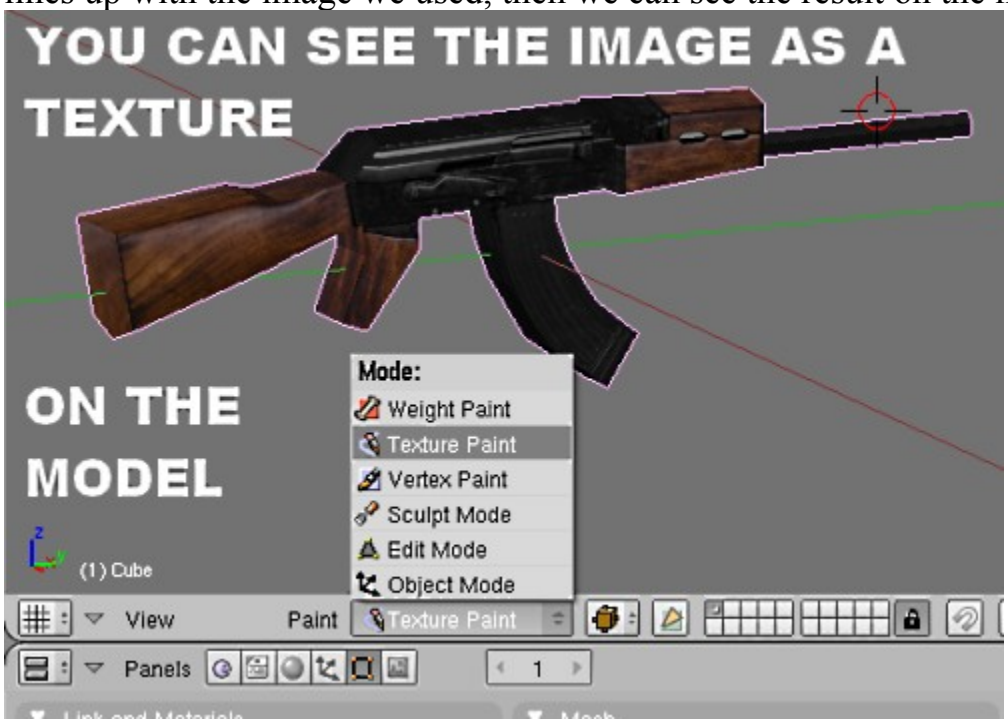
The texture making can go in different directions here. Once you more proficient at *Baking*, you can just make your own out of the 3D model, and with materials. They can even have a metal shine. But you could also use the same image we sculpted around. In the *UV* window the image can be selected. Just to show it is possible to use it, I will use a **Project From View** *uv unwrap*.



Here is another way you can create a UVMap. *Project From View* in 3D space. Position the view to be sideways as in the picture. Then select **Mesh** and choose **UV Unwrap**. Then Select **Project From View.**



In the *UVmap* window, you can scale the map to fit the image. Once the map lines up with the image we used, then we can see the result on the model.



8. Armatures, Weight Groups, And The Parent/Child Relationship

Armatures consist of a collection of **Bones** that are invisible in the game, but not in Blender. These bones move objects that are parented to them. The child objects being moved know to move with the bones because they each have **weight groups**. The *weight groups* consist of a select number of verts assigned to a given weight group. Weight groups can be edited in Blender's *Edit Mode*. And the bones can be moved in Blender's **Pose Mode**.

In the tutorial you will learn how to create a new Barbell prop, which has 3 bones. Barbells have 3 scenes in the Movies Game. They are not selectable in scenes, so you can't change one into another barbell. You will have to make new scenes to use new barbells. At the end of the tutorial is another tutorial describing how to use **FLM Reader Zero** to make a new scene for a new barbell. And it is very easy.

You do not have to actually export a new mod or barbell, and you do not have to make a new scene in order to participate with the tutorial. Its main purpose is to show you what is going on with 3D models possessing an armature.

• • •

The final part of this tutorial is optional. How to make a new barbell prop. Barbell props are not selectable in scenes. They are only called by the scene file so if you make another one you will have to make a new scene for it. But this is easier then making a new prop. It is simply making just one change in a scene file and that’s it. You will need **FLM Reader Zero** to make this new scene. You can download it for free at TheMovies3D.com. You do not have to actually make a scene to follow this tutorial as it will help you understand **Bones** and **Weight Groups**.

This installment will show you how costumes, or horses, or dogs, or some of the props move around on a set. Often these items also move around on the studio lot. Just as actors walk from facility to facility, or from one set to another while a movie is being filmed.

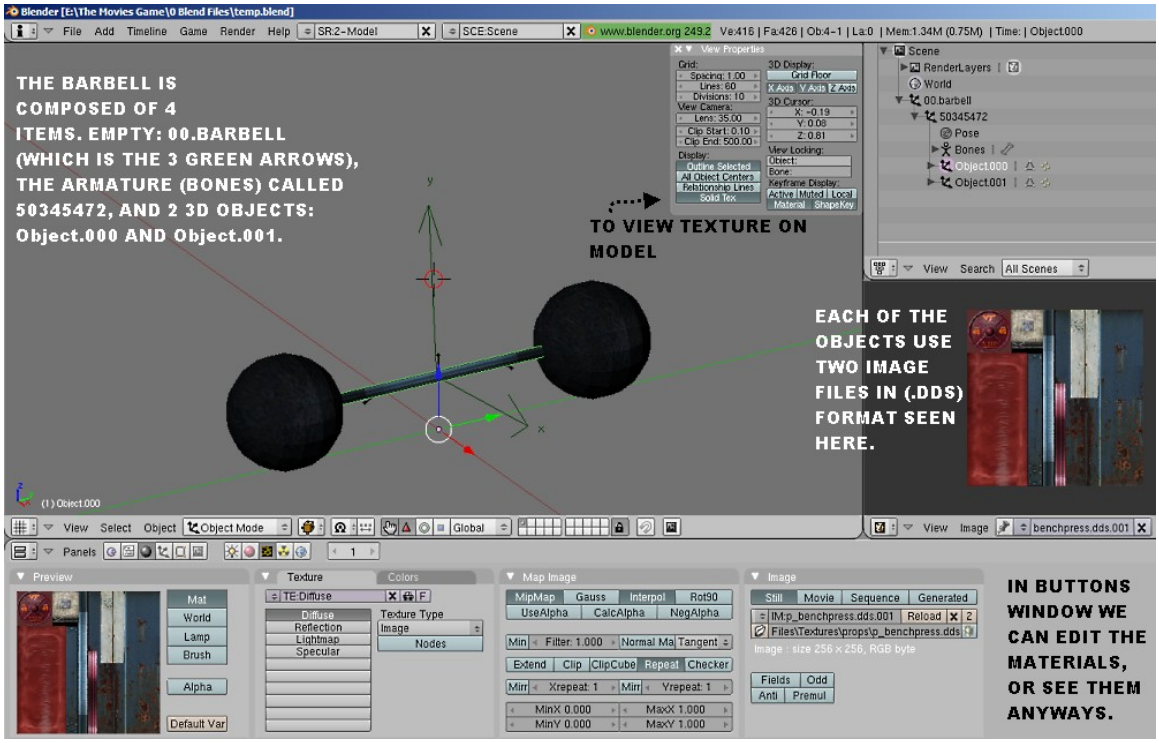
This movement happens when two items come together. The first item is an

animation file. We won't look at one but we will talk about it. The game has a library of animations it loads from to animate 3D models. Contained within those files is data targeting the 3D model. It has data for every single frame. If an animation takes 25 frames to complete, there is 25 different commands for the 3D model. Each of these commands has even more detailed information, but generally it is the "Shape" and "Location" of the model during animation. There has to be something in the 3D model listening for these commands, and it is called the **Armature**, the second item**.** Or sometimes it is called **Bones.**

What is an armature? It is a skeleton of sorts. It is a collection of invisible *Bones* that move 3D models. In the game there are 2 kinds of 3D models. Ones with an armature, and ones without an armature. That means there are 2 kinds of sets, sets with an armature and sets without one. There are props with and without armatures. There are facilities with or without an armature. All costumes have armatures. Anything with "Bones" found in the game also has animation files designed for it.

Armatures will not animate, either in Blender or in The Movies game unless an animation file is fired at it. Without the animation, the *Bones* do not move. Scene files will tell your model to move around as it calls the animation files during game play. Believe it or not, that is what is also happening on the studio lot. On the lot is a wonderfully orchestrated collection of scene files designed just for the studio lot.

Another tutorial will discuss animating in Blender. For now we will go over some basics about Movies game models with *Armatures*. Use MED to extract the 3D model called **p_barbell.msh** It has an armature of 3 bones which moves the object in a scene. Import p_barbell.msh into Blender.

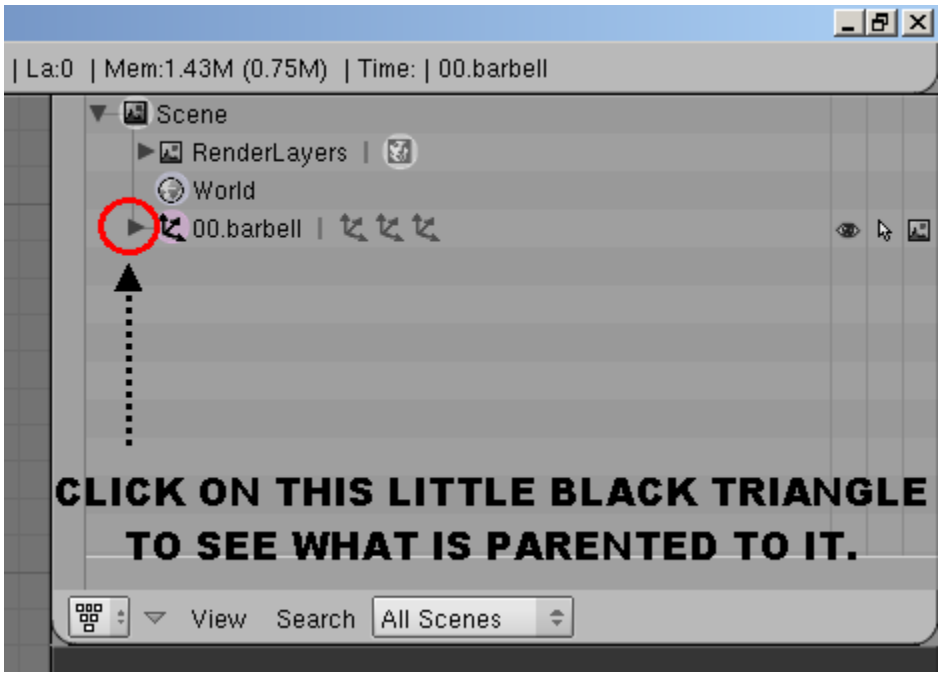


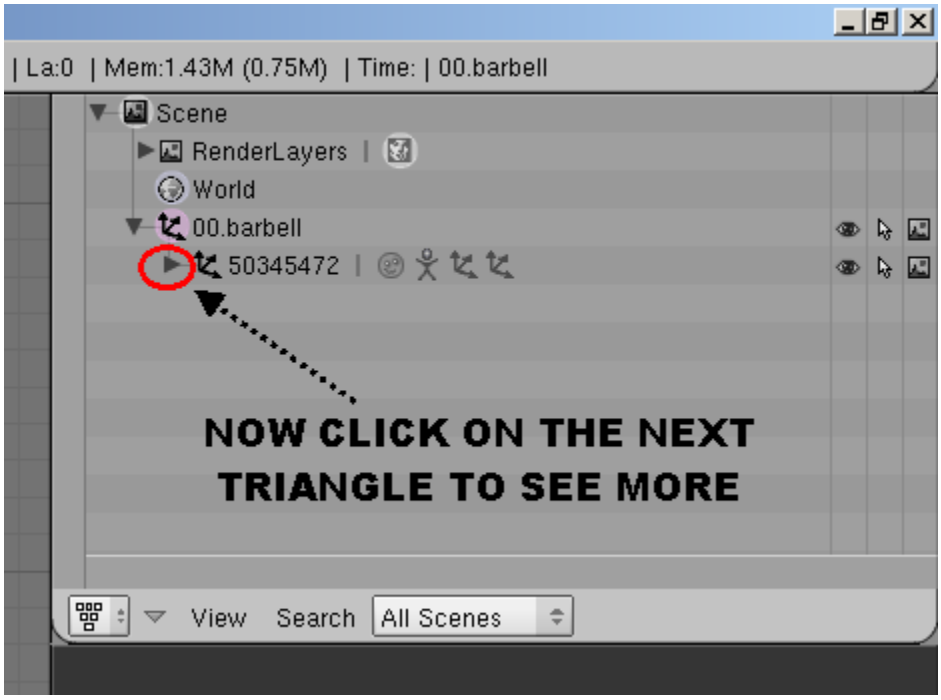
Let's talk about the *child* and *parent* relationship in Blender. A *parent* has children, right? In Blender **Parent** and **Child** is the status of a given object or item. In the *Outliner* window, expanding the information will reveal what is a *parent* and what is a *child*. In the *Outliner* click on the little triangles until we see what is parented to each item. The first parent is the *Empty* object (which looks like 3 green arrows in 3D space) and is called **00.barbell**. Under this *parent* is the *armature* (the Bones) called **50345472**, and under the *armature* is **Object.000** and **Object.001**, which are the 3D models giving the appearance of a barbell.

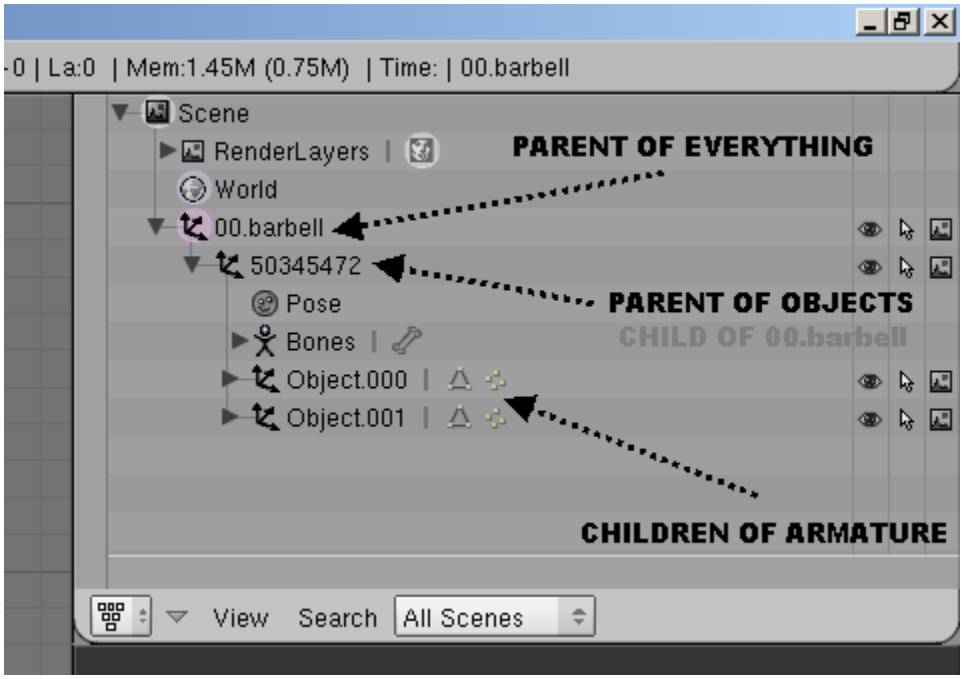
What is the reason for the Parent/Child relationship? Slavery! LOL, well not really. The objects are under the control of the *armature*, and Blender, or the game, will need to know this. You can think of it as the objects being attached to the *armature.* In this way they follow the orders of the *armature*. The export script sees the *armature* and then sees the objects, and will ask if the objects are *children* of it. If the export script doesn't see the objects as being *children* of the *armature* then it will give you an error. The **Preflight Script** will also look over your project and see if this is the case. If it finds the same error, it will tell you so. Let's see this in action.

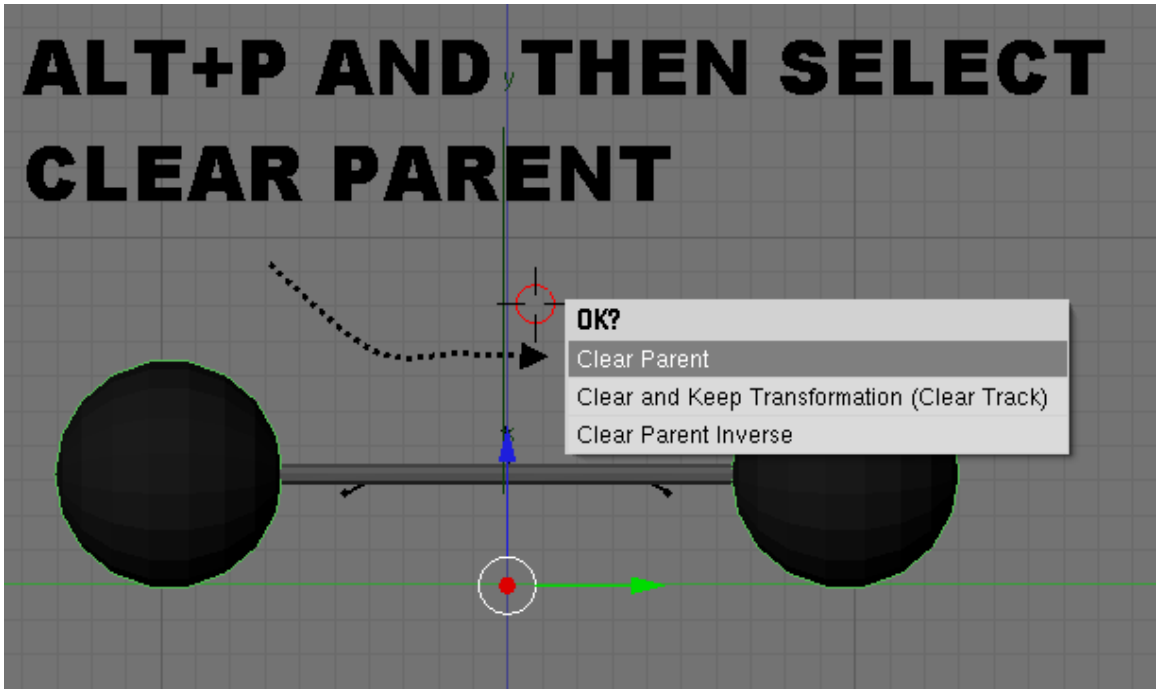
To do this you will need to extract the *p_barbell.msh* prop from the game using MED (The Movies Game Editor). Then you will need to import it into Blender. And do nothing else but what I am about to tell you to do.

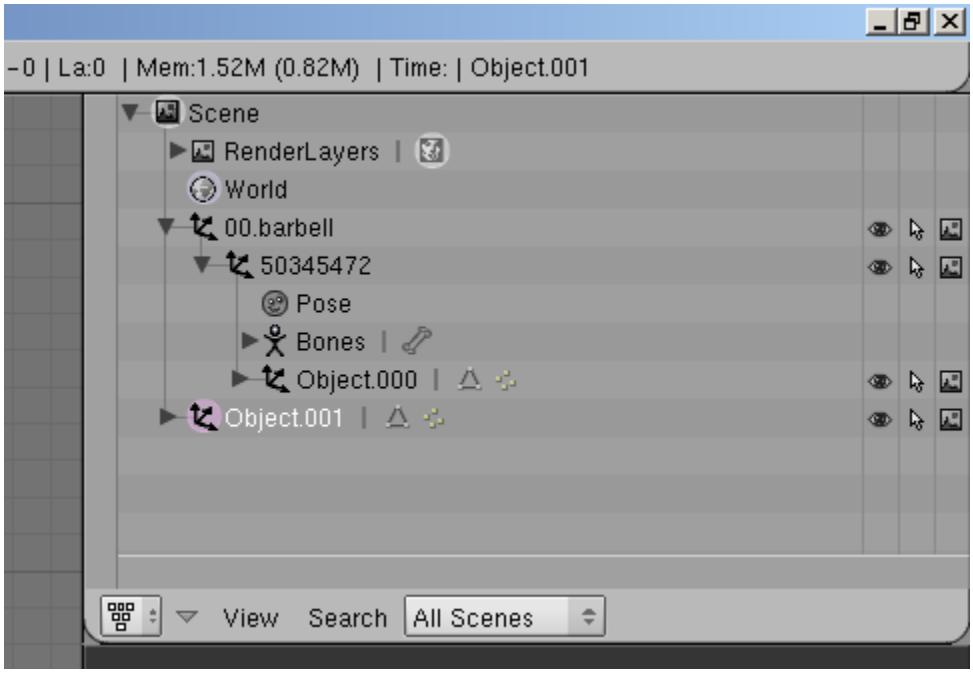
- Expand the little triangles in the Outliner Window. This will reveal the objects *parented* to each item
- In the *Outliner* window click on **Object.001**. The object gets lit up in *3D* space when it is selected
- Now hover the mouse in *3D* space
- Hit Hotkey **ALT + P**. This will Unparent a selected item. In the *Outliner* window, the item is by itself
- In 3D space, select green arrows and press **G** key for grab. And just move the mouse around. You can see the black balls not moving with the whole

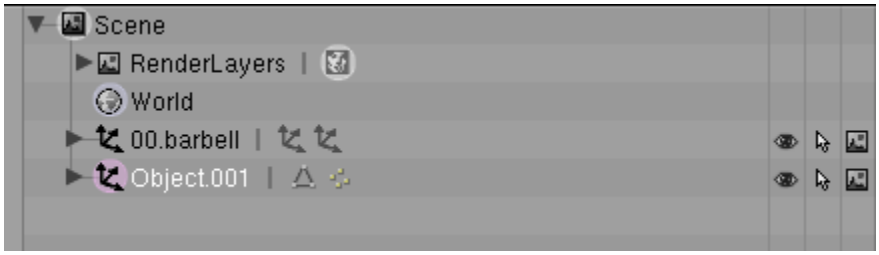


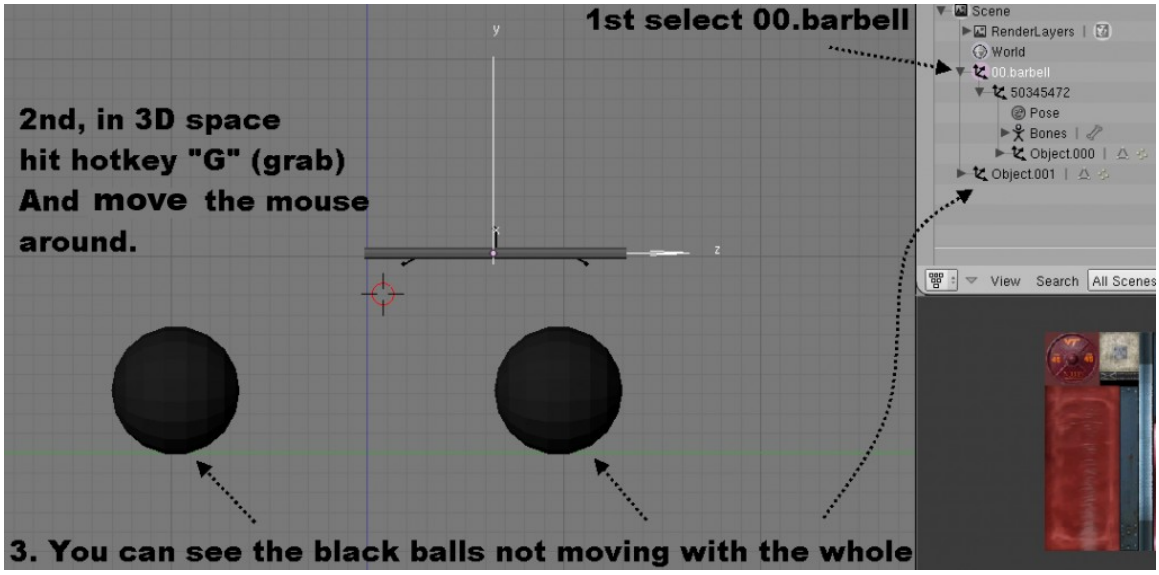

| La:0 | Mem:1.43M (0.75M) | Time: | 00.barbell
Scene
RenderLayers
World
00.barbell
50345472
NOW CLICK ON THE NEXT
TRIANGLE TO SEE MORE
View
Search
All Scenes


0 | La:0 | Mem:1.45M (0.75M) | Time: | 00.barbell
Scene
RenderLayers
PARENT OF EVERYTHING
World
00.barbell
50345472
PARENT OF OBJECTS
Pose
CHILD OF 00.barbell
Bones
Object.000
Object.001
CHILDREN OF ARMATURE
View
Search
All Scenes


ALT+P AND THEN SELECT
CLEAR PARENT
OK?
Clear Parent
Clear and Keep Transformation (Clear Track)
Clear Parent Inverse

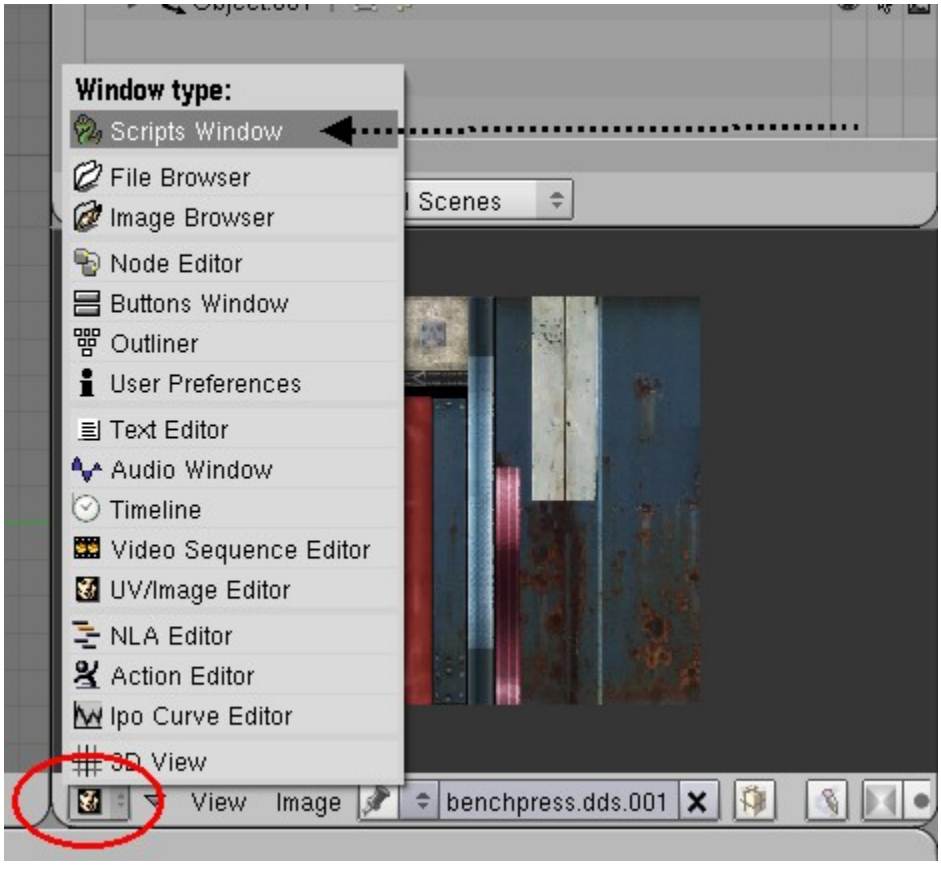

-0 | La:0 | Mem:1.52M (0.82M) | Time: | Object.001
Scene
RenderLayers
World
00.barbell
50345472
Pose
Bones
Object.000
Object.001
View
Search
All Scenes


Scene
RenderLayers
World
00.barbell
Object.001

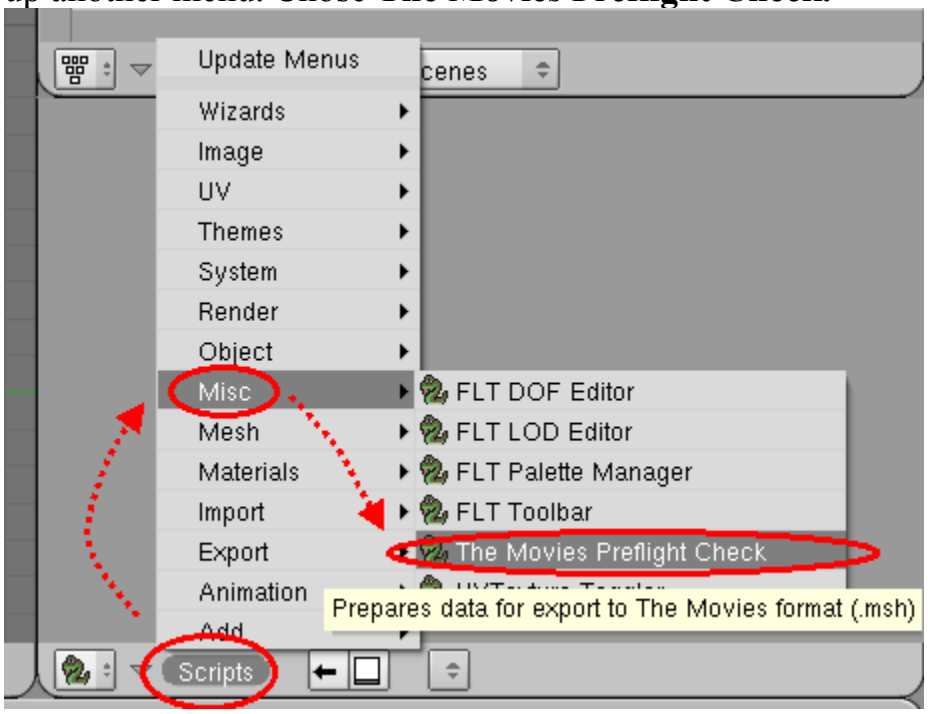


The black balls do not move around because they are no longer the children of the armature.

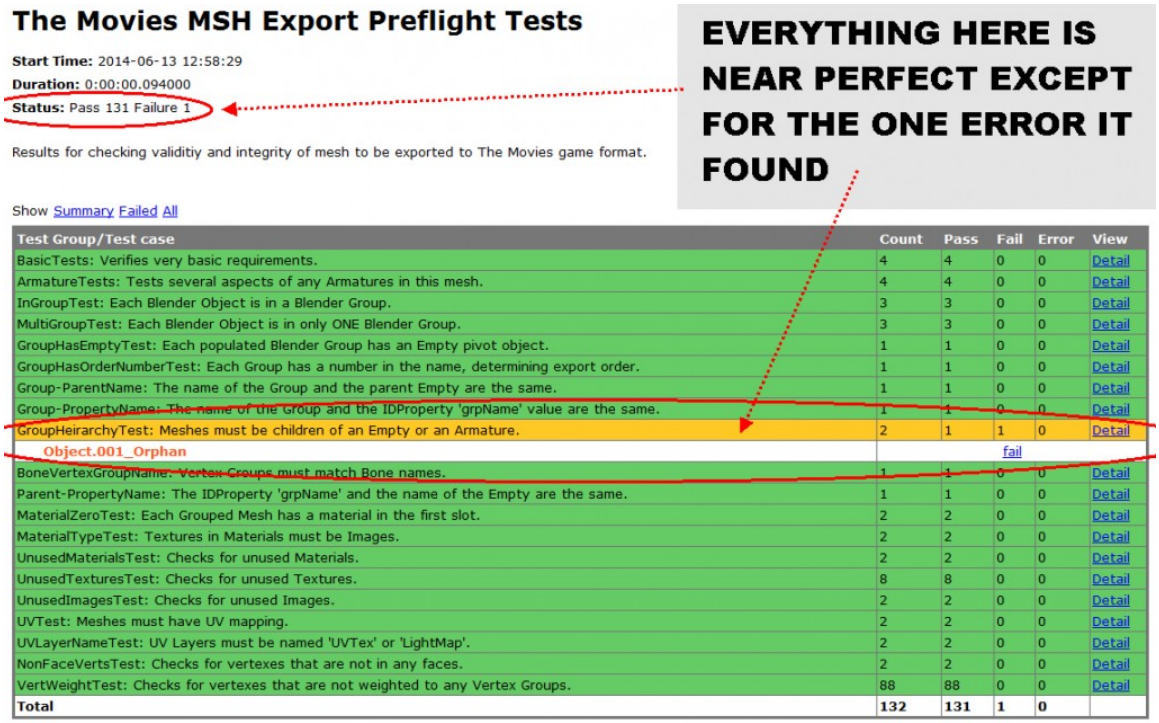
Before we make the Parent/Child relationship again, lets run the *Preflight Script* to get a clue as to what it will find. For a brief moment we will need to turn one of the windows into a *Scripts* window. Any window will do it, but let's use the *UV* window...



Once the window has changed into a *Scripts* window, let's run the Preflight Script. At the bottom of the window is a little button next to the icon called **Scripts**, it brings up a popup menu to chose from. Chose **Misc.** This brings up another menu. Chose **The Movies Preflight Check.**



Now we wait a second or two. The project is now being analyzed for errors. We would run this script when we think we are done and ready to export back into the game. No need to export yet, but let's see what the script would find since we un-parented Object.001 from the armature.



The Movies MSH Export Preflight Tests

Start Time: 2014-06-13 12:58:29
Duration: 0:00:00.094000
Status: Pass 131 Failure 1

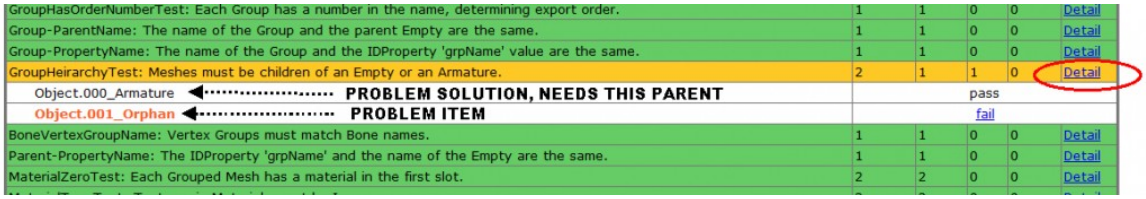
Results for checking validitiy and integrity of mesh to be exported to The Movies game format.

EVERYTHING HERE IS NEAR PERFECT EXCEPT FOR THE ONE ERROR IT FOUND

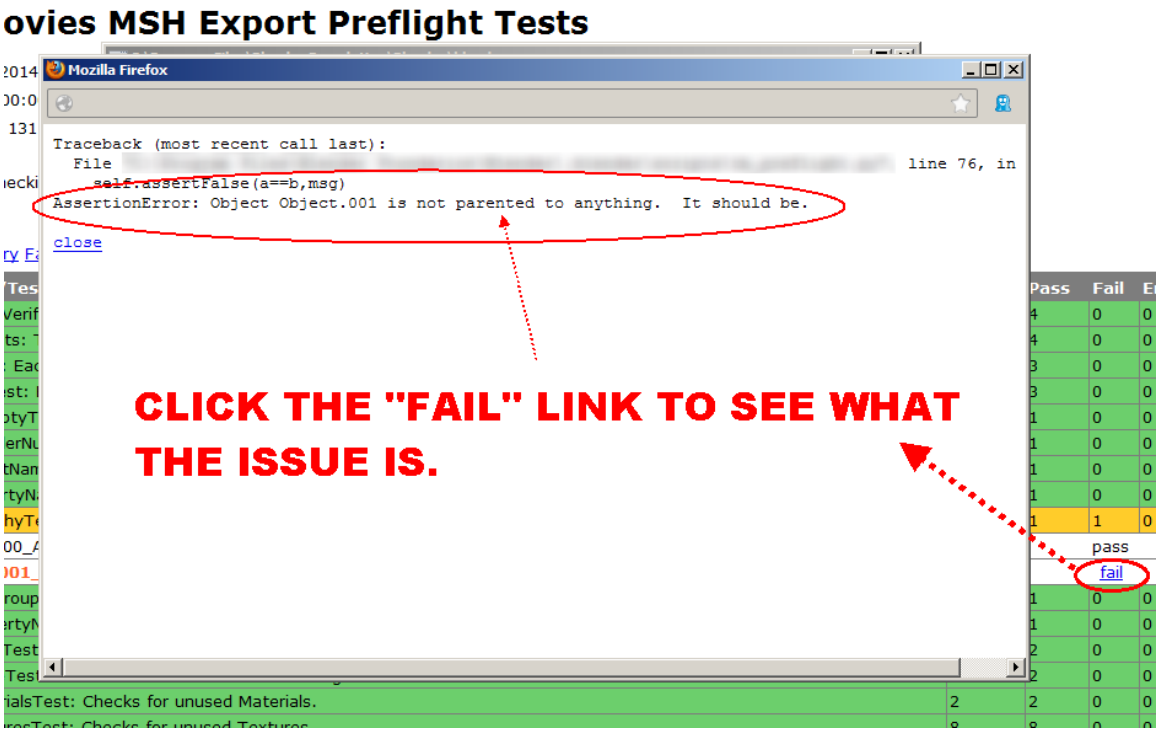
Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
BasicTests: Verifies very basic requirements.	4	4	0	0	Detail
ArmatureTests: Tests several aspects of any Armatures in this mesh.	4	4	0	0	Detail
InGroupTest: Each Blender Object is in a Blender Group.	3	3	0	0	Detail
MultiGroupTest: Each Blender Object is in only ONE Blender Group.	3	3	0	0	Detail
GroupHasEmptyTest: Each populated Blender Group has an Empty pivot object.	1	1	0	0	Detail
GroupHasOrderNumberTest: Each Group has a number in the name, determining export order.	1	1	0	0	Detail
Group-ParentName: The name of the Group and the parent Empty are the same.	1	1	0	0	Detail
Group-PropertyName: The name of the Group and the IDProperty 'grpName' value are the same.	1	1	0	0	Detail
GroupHeirarchyTest: Meshes must be children of an Empty or an Armature.	2	1	1	0	Detail
Object.001_Orphan			fail		
BoneVertexGroupName: Vertex Groups must match Bone names.	1	1	0	0	Detail
Parent-PropertyName: The IDProperty 'grpName' and the name of the Empty are the same.	1	1	0	0	Detail
MaterialZeroTest: Each Grouped Mesh has a material in the first slot.	2	2	0	0	Detail
MaterialTypeTest: Textures in Materials must be Images.	2	2	0	0	Detail
UnusedMaterialsTest: Checks for unused Materials.	2	2	0	0	Detail
UnusedTexturesTest: Checks for unused Textures.	8	8	0	0	Detail
UnusedImagesTest: Checks for unused Images.	2	2	0	0	Detail
UVTest: Meshes must have UV mapping.	2	2	0	0	Detail
UVLayerNameTest: UV Layers must be named 'UVTex' or 'LightMap'.	2	2	0	0	Detail
NonFaceVertsTest: Checks for vertexes that are not in any faces.	2	2	0	0	Detail
VertWeightTest: Checks for vertexes that are not weighted to any Vertex Groups.	88	88	0	0	Detail
Total	**132**	**131**	**1**	**0**	

A browser window pops up. If everything came up green then you would be good to go, (You could export the model and have it working fine in the game). If something comes up yellow it is saying, "Slow down, there's still an error." And this error would need our attention before we try to export it. Means there is more work to do.



GroupHasOrderNumberTest: Each Group has a number in the name, determining export order.	1	1	0	0	Detail
Group-ParentName: The name of the Group and the parent Empty are the same.	1	1	0	0	Detail
Group-PropertyName: The name of the Group and the IDProperty 'grpName' value are the same.	1	1	0	0	Detail
GroupHeirarchyTest: Meshes must be children of an Empty or an Armature.	2	1	1	0	Detail
Object.000_Armature ◀ PROBLEM SOLUTION, NEEDS THIS PARENT			pass		
Object.001_Orphan ◀ PROBLEM ITEM			fail		
BoneVertexGroupName: Vertex Groups must match Bone names.	1	1	0	0	Detail
Parent-PropertyName: The IDProperty 'grpName' and the name of the Empty are the same.	1	1	0	0	Detail
MaterialZeroTest: Each Grouped Mesh has a material in the first slot.	2	2	0	0	Detail



Clicking on the links in the browser will reveal more data about what is wrong. Fail brings up why it failed. Detail tells you what object or item went wrong.

We knew this because we un-parented Object.001 from the *armature* and now we see how useful the Preflight Check is. The Preflight will see this and tell us it needs to be *Parented* to the *armature*. So let's make the Parent/Child relationship again. Go ahead and exit the browser windows. We will correct this.

To make a *Parent* we need two objects selected at once. This is important and we have to learn how to select two objects at once. Using the selection tool won't work, meaning we can't use hotkey **B** for *border select*. You can select objects by clicking on them in *3D* window, or by name in the *Outliner* window. They have to be selected in order.

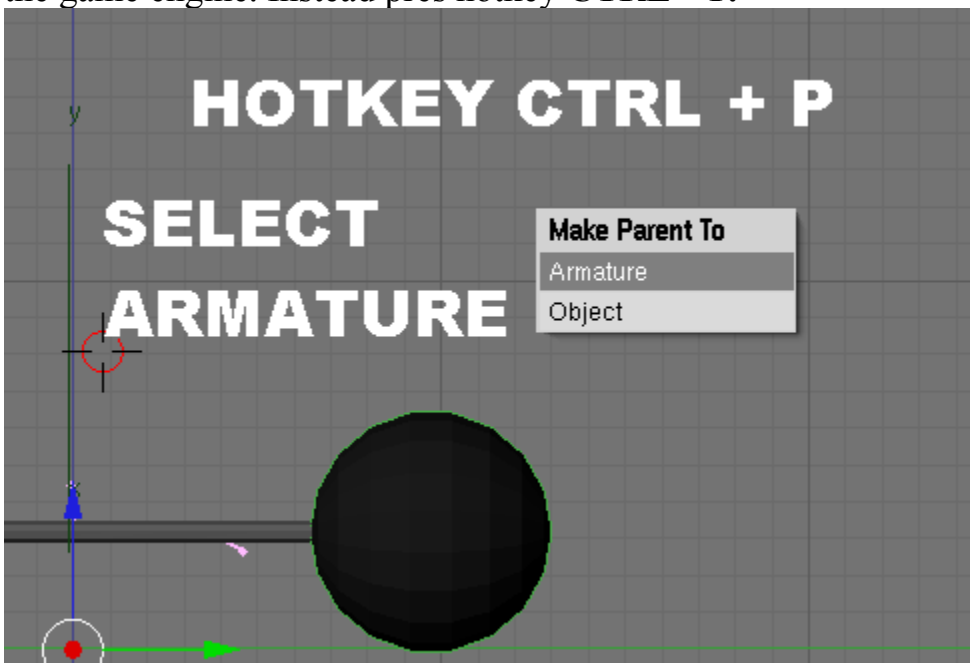
There are many functions in Blender that require more then one object to be selected. You could transfer the weights from one object to the next, or you could *parent* one object to another. But BLENDER NEEDS TO KNOW WHICH ONE. Blender will know which object is the primary recipient by the order it is selected in. That goes for making a *Parent*.

First click on **Object.001** in the *Outliner* window.

Next hold down the **Shift** key.
Keep the **Shift** key pressed and click on **50345472,** which is the *armature.*
Now two things are selected at once. Let go of **Shift.**

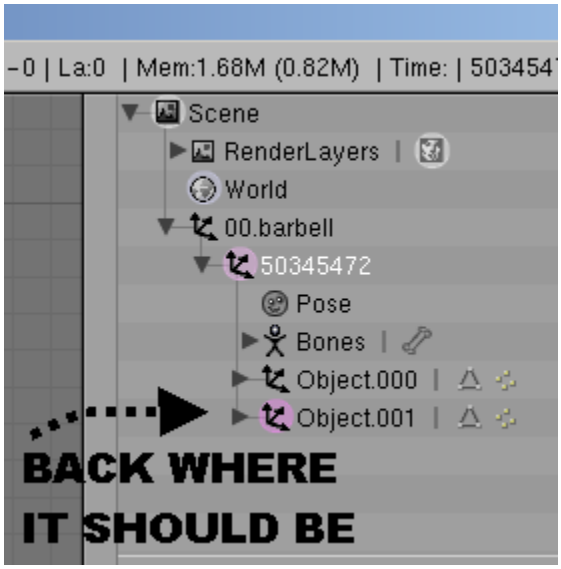
This means that holding down **SHIFT** key will allow you to keep selecting stuff without losing the selection of the previous object. You will do this a lot in your projects.
The very important thing here is the Armature **50345472** was chosen last and not first. Object.001 was chosen first, and 50345472 was chosen second. This is how Blender will know that we want the *armature* to be the *parent* and Object.001 to be the *child.* Using the S**hift** key and the order of things being selected will be your skill.

Now **Object.001** was chosen *first*, and the **50345472** was chosen *second*, lets make a *child* and a *parent*. Hover mouse in 3D window and press Hotkey **CTRL + P** (make *Parent*). Note: If **CRTL** wasn't held down while hitting **P** key then Blender might freeze. If it freezes it is because **P** without **CTRL** start's Blender's game engine. If that happens press **ESC** key to exit the game engine. Instead pres hotkey **CTRL + P.**



A popup menu will ask you which item. Click **Armature**.
Another important menu pops up. This one is because we were *parenting* to an *armature*. These groups the menu is referring to will be discuss in a bit. They are the *weight groups* of the black balls, and Object.001 already has them. Because it already has *weight groups* we will select **Don't Create Groups**. This will keep the groups it already has.

In the *Outliner* window we can see the original position of Object.001.



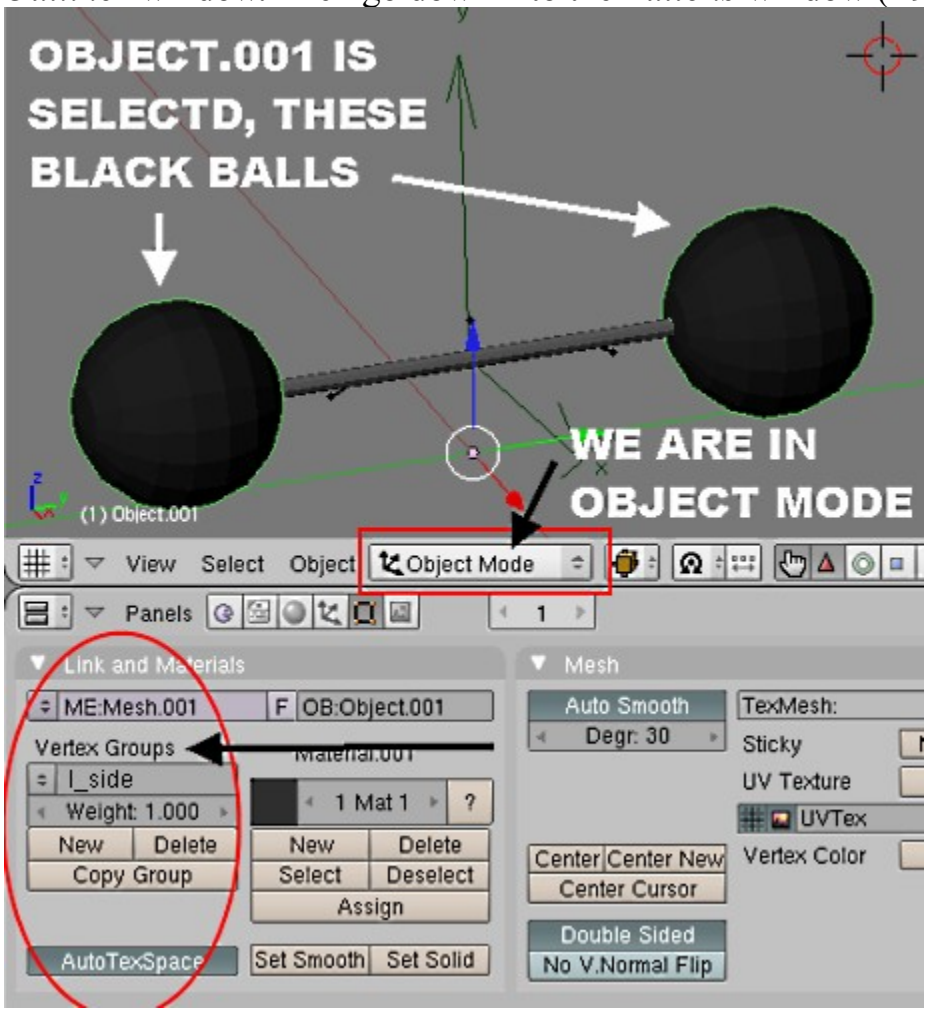
Hotkey **CTRL + P** will make an object the *child* of another object.
ALT + P will clear the *Parent* from a *child* object.

The *armature* is the magic wand of the Barbell prop. It is what moves the barbell around during a scene. I think the very first movie the game makes uses the prop in a scene on the stage set. Baggage Boy or something. We will move it ourselves in Blender. A question here is, how does the objects move or bend to the *armature*?

Each *child* object of the *armature* contains weight groups. *Weight groups* are useless unless the project has an *armature,* and an *armature* is useless unless the *child* object has *weight groups*. Blender is an amazing animation tool. It

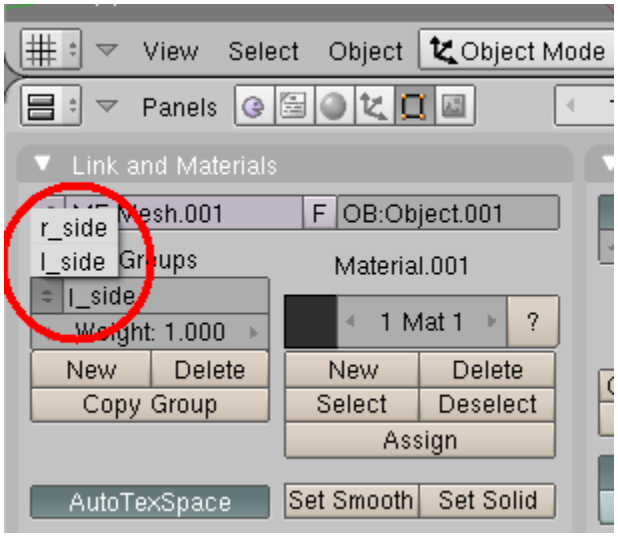
has loads of tools designed to view and edit the *weight groups*. In this case we will use the buttons window to edit the *weight groups.*

Let's take a look at the *weight groups*. First click on **Object.001**, in the *Outliner* window. Then go down into the *Buttons* window (**F9**).

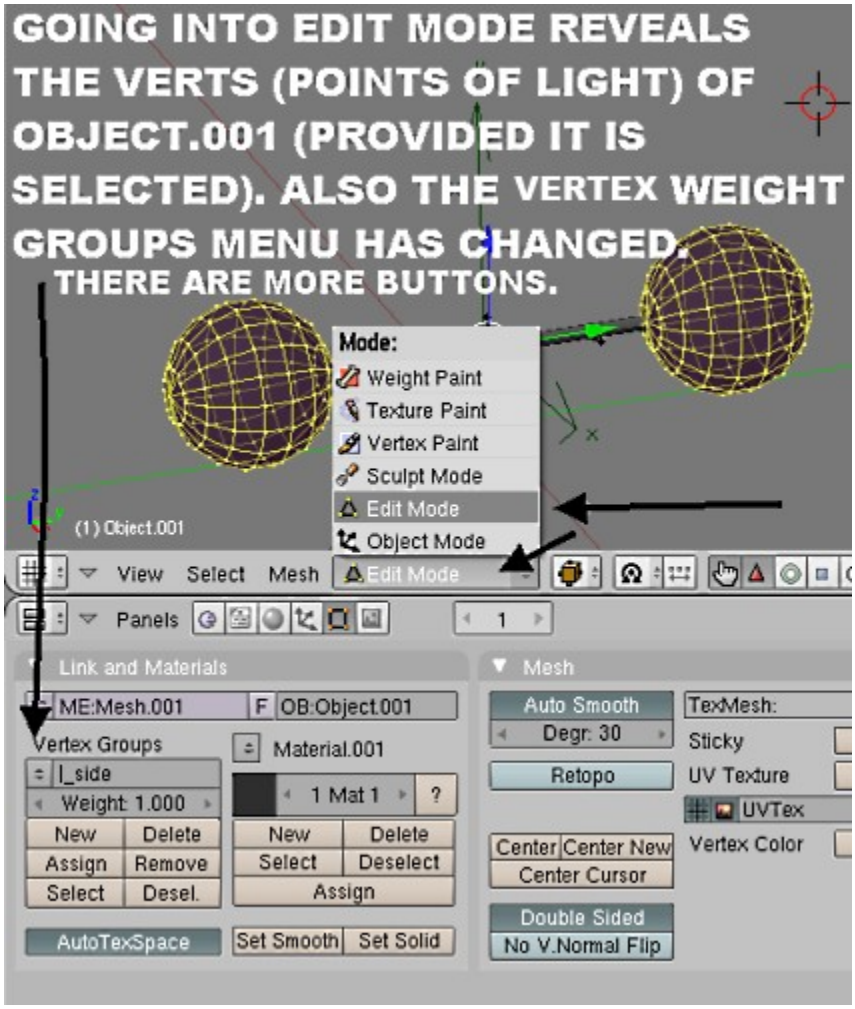


We are in Object Mode which gives us only a couple buttons for the vertex groups. The **Vertex groups** are at bottom left of the screen.

Only one *weight group* is selected at present. It isn't doing nothing yet being selected, but it is. The one selected in the pic is **l_side.** To the left of the name is a little arrow up and down icon. Clicking it brings up a menu for all of the *weight groups* assigned to Object.001.

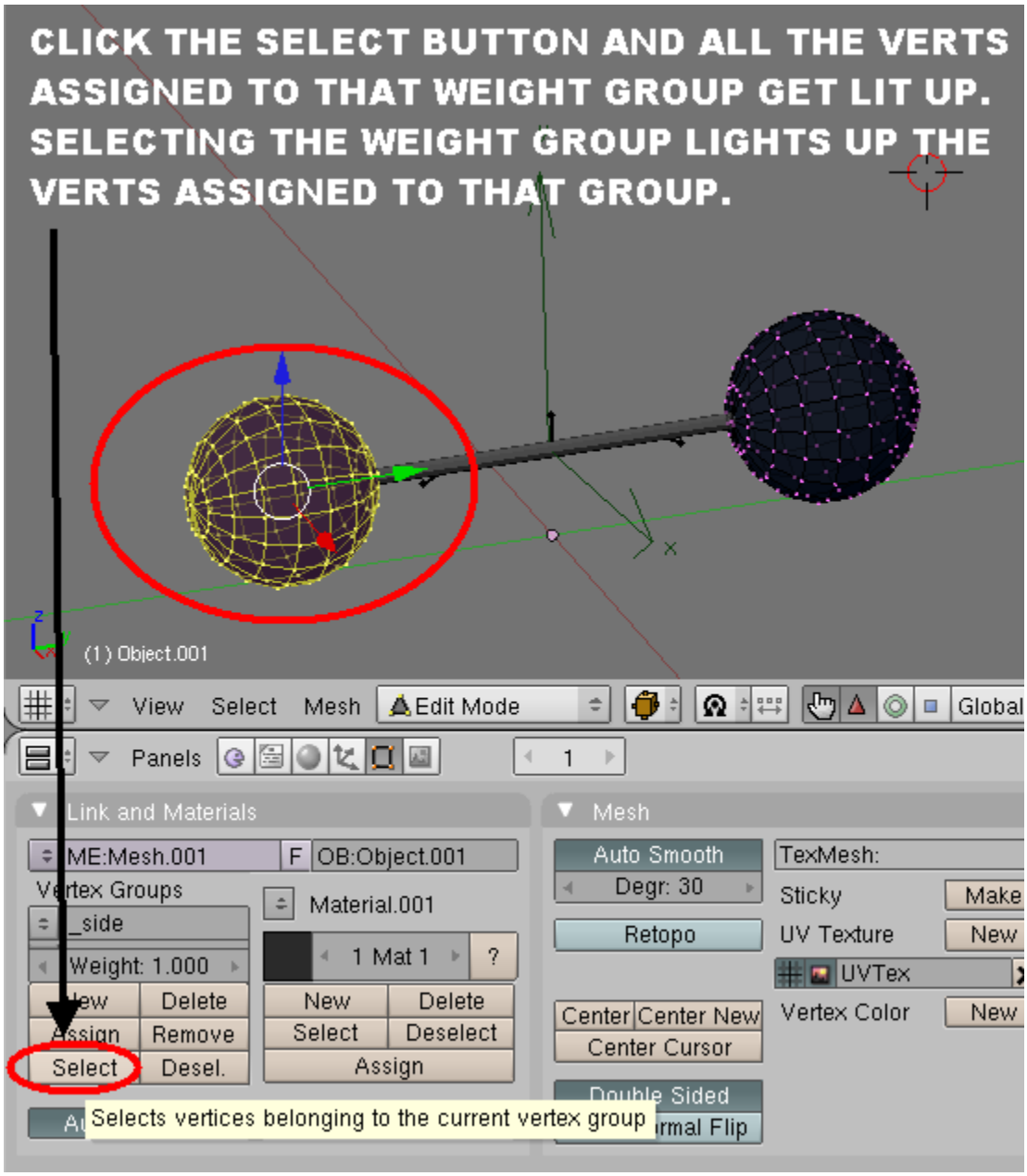


Only 2 groups, **l_side** and **r_side**. What exactly are these names for? They are the names of *bones* found in the parent armature item. More about that in a bit. Let's go into **Edit Mode** (Press **TAB** key) or click the mode box and select *Edit Mode*.



In *Edit Mode* Object.001's verts are all lit up. First thing make sure we deselect everything so that it is not lit up. Lets turn the lights off for the vertices. (these are commonly called polygons in other 3D programs). Press Hotkey **A** which either selects all or deselects all.

When all the lights are off, we know every vert is no longer selected. This will be important because the *weight groups* consist of a select number of these verts. So lets take a look at the second piece of evidence that a weight group exist on Object.001. Down in the *Button* window, click on the button **Select.**



What did this do? It lit up some of the verts in the 3D window. The verts became selected when we hit the **Select** button. The verts that became selected are the very same verts belonging to the *weight group*. Belonging to the weight group called **l_side**. (Or maybe you had **r_side** group ready for selection and in that case the verts that lit up belong to that group instead.)

For now we won't select any of the other buttons.
Deselect will deselect the verts that belong to that weight group. If all verts were selected at once and you hit deselect, then the verts assigned to **r_side** will get deselected and the remaining verts will still be lit up and selected.
Remove button would take away the verts that were selected from the weight group and in this case no more verts would be in that group. Don't do this.
Assign would mean that the verts that are presently selected would be entered into the weight group. You see the numbers just above saying 1.000? That is the strength of the weight group's assignment. 1.000 is max. If you wanted to assign verts to a group but have little strength you could enter 0.500 for half strength or 0.1 for just barely a little strength.
New button would create a brand new *weight group*. We don't need to do this. If the object had no groups yet, this is exactly the button we would need. You would create groups so the *armature* would move the object around in a scene. The barbell has them already.
Delete gets rid of the *weight group*. And sometimes we no longer want an object to have one or more groups.

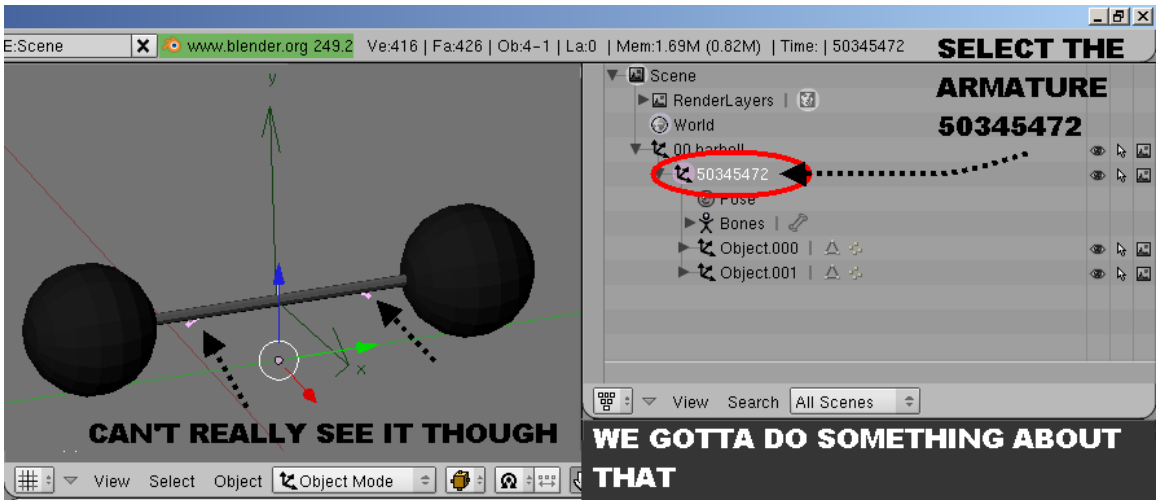
There is a mesh/script that will get rid of an object's unused weight groups. Mesh>Scripts>Clean Mesh>**V Group Clean**. Just so you know and we don't need it now.

Deselect every vert with hotkey **A**. Now change the *weight group* selected by clicking on the little up down arrows icon next to **l_side** and select **r_side**. Then click the **Select** button and the verts on the other side of the barbell will light up.

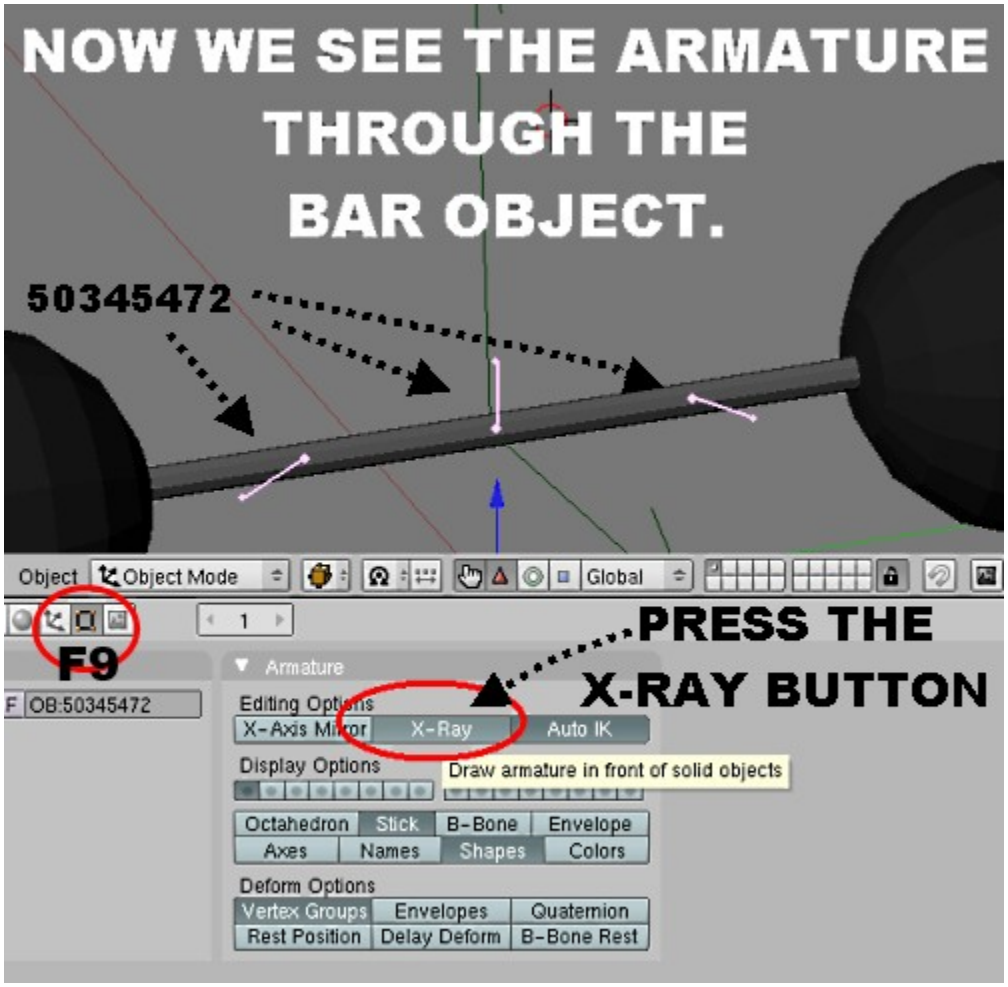
What are *Weight Groups*? Weight groups are like gravity. How much sway the armature's bones have over the *child* objects. The armature is a collection of bones. Each bone has a name. The weight groups uses those very same names. That way the child objects move to the bone because some of the verts assigned to that group shares the same name. The verts assigned to the **l_side** *weight group* moves when the bone that is called **l_side** moves.

The names are important because they have to match the names of the *Bones* belonging to an *armature*. Let's get a better look at the armature (50345472) itself.

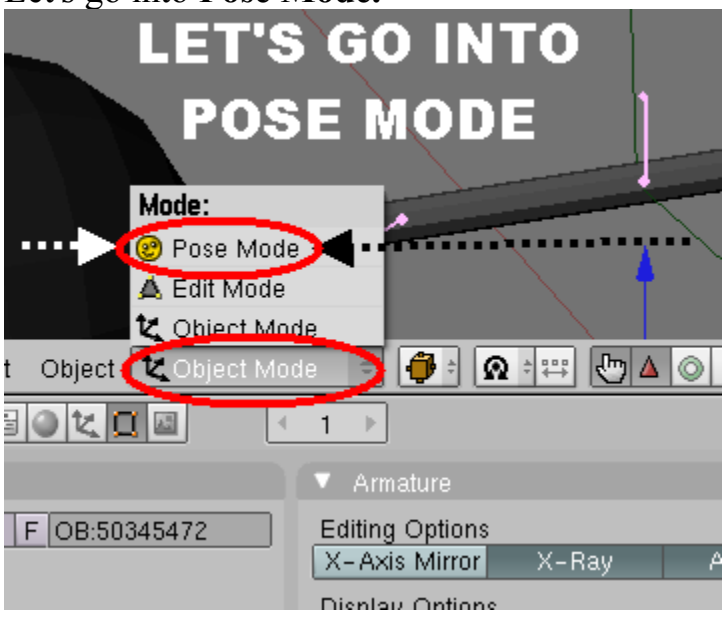
Go back into *Object Mode*. **Tab** key or click the box with then name *Edit Mode* and from the popup menu select **Object Mode**.
Now in the *Outliner* window click on **50345472** which is the armature.

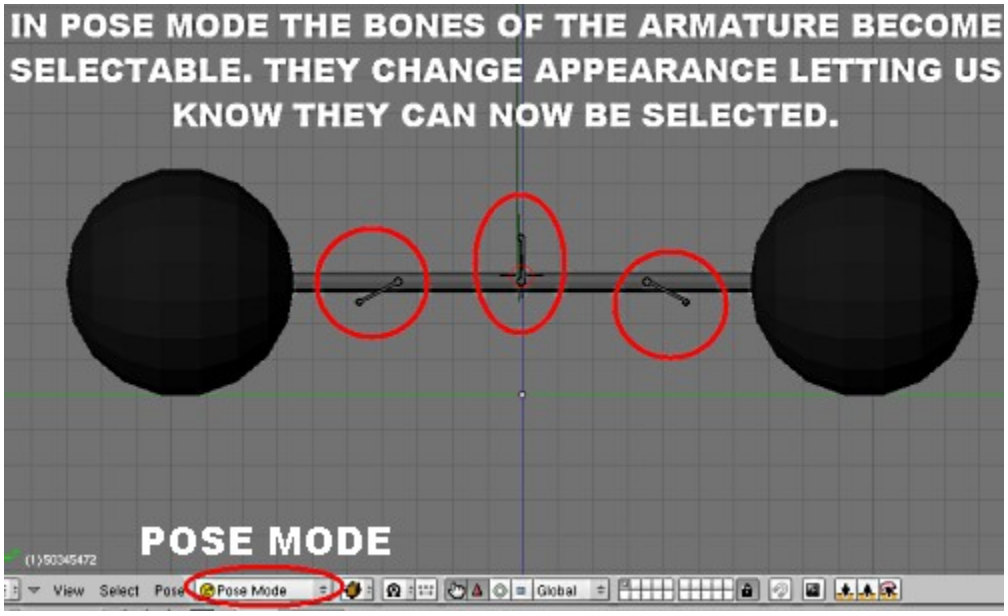


We can't see it too well in the *3D* window because it is inside the barbell. Fortunately there is an **X-Ray** button that will allow us to see it at all times. Press the **X-Ray** button in the *Buttons* window (**F9**).

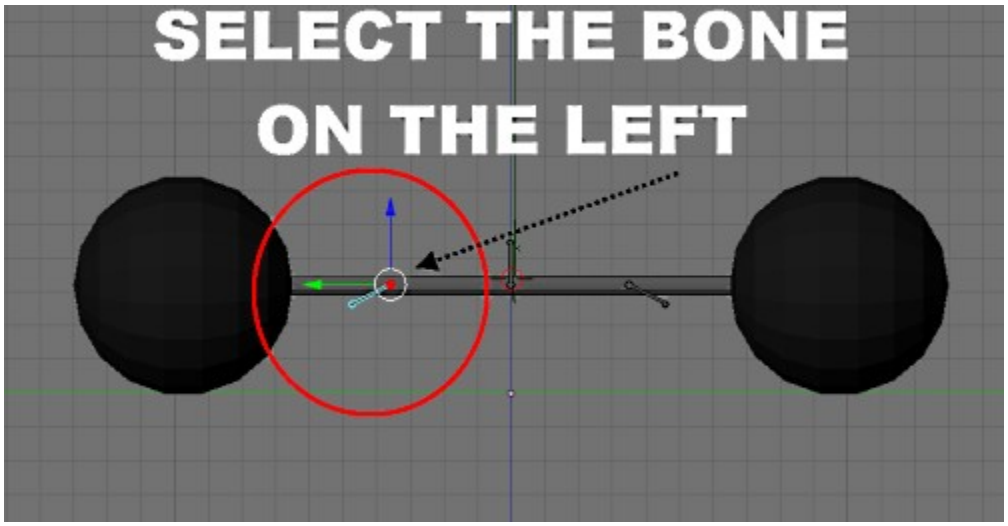


Now that we can see it lets see what the *armature* can do.
Let's go into **Pose Mode.**



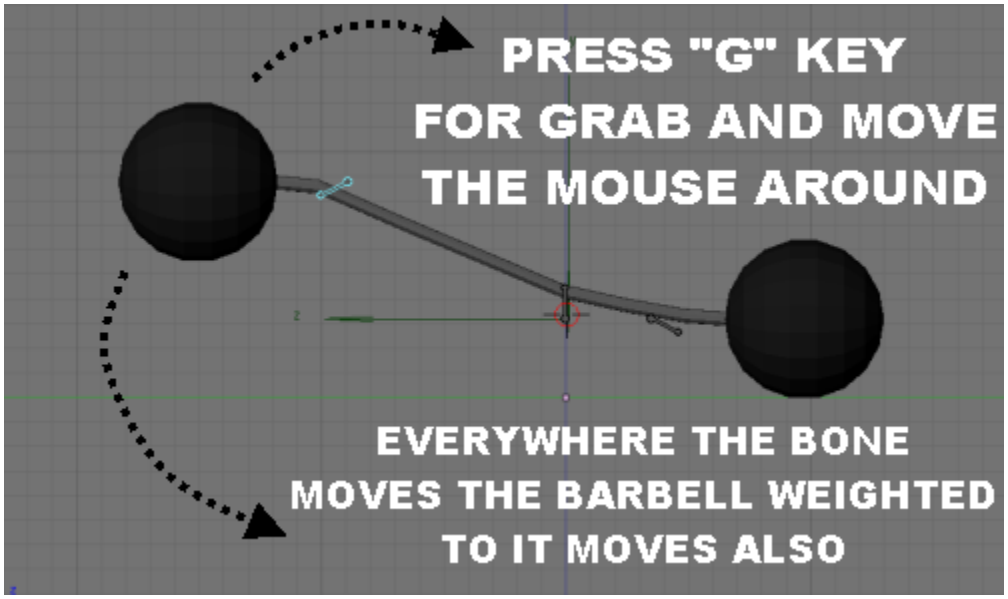


Select the bone on the left and it becomes blue.
That blueness is how we know it is selected.

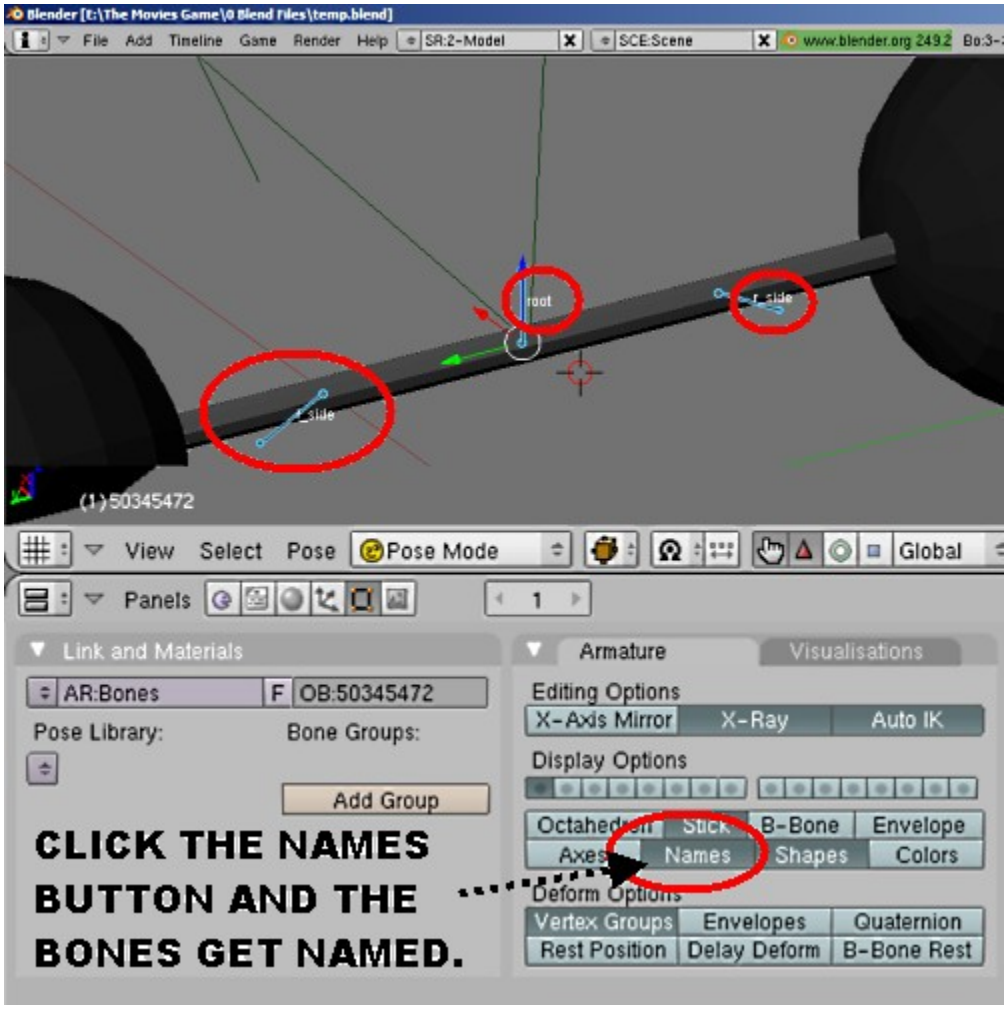


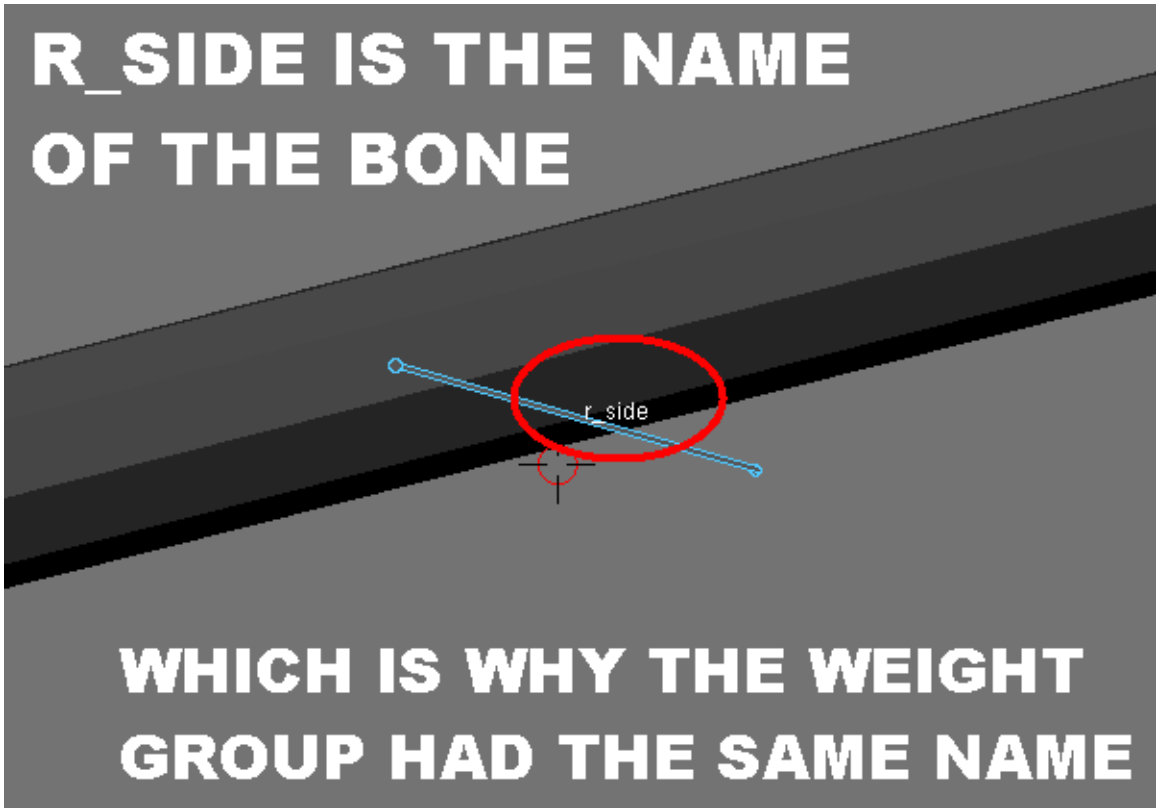
Press hotkey **G** to *grab* the bone. Now we move the mouse around and see that the bone can be moved. Also the child object, which has the weight group for it also moves. **l_side** and **r_side** are the groups allowing the barbell black balls to move when we move the bones around. If we had removed those *weight groups* from the object, then only the bone would move and nothing else. This is why Object.001 needs the *weight groups,* because without them, there would be no movement.

When moving a bone around you can set a new place for it. But if you right click, then the bone will snap back to the original rest position. If you moved the bone to a new place you can press CTRL + Z to undo the placement.



Now let's take a look at why this works. What does the names of the weight groups have to do with the armature? We need to see the names of the bones. There appears to be three of them. In the *Buttons* window is a button called **NAMES**. Click that button. In the 3D window we now see each bone will have a name next to it.


Blender [I:\The Movies Game\0 Blend Files\temp.blend]
File Add Timeline Game Render Help SR:2-Model SCE:Scene www.blender.org 249.2
root
r_side
l_side
(1) 50345472
View Select Pose Pose Mode Global
Panels 1
Link and Materials
AR:Bones F OB:50345472
Pose Library:
Bone Groups:
Add Group
CLICK THE NAMES BUTTON AND THE BONES GET NAMED.
Armature Visualisations
Editing Options
X-Axis Mirror X-Ray Auto IK
Display Options
Octahedron Stick B-Bone Envelope
Axes Names Shapes Colors
Deform Options
Vertex Groups Envelopes Quaternion
Rest Position Delay Deform B-Bone Rest

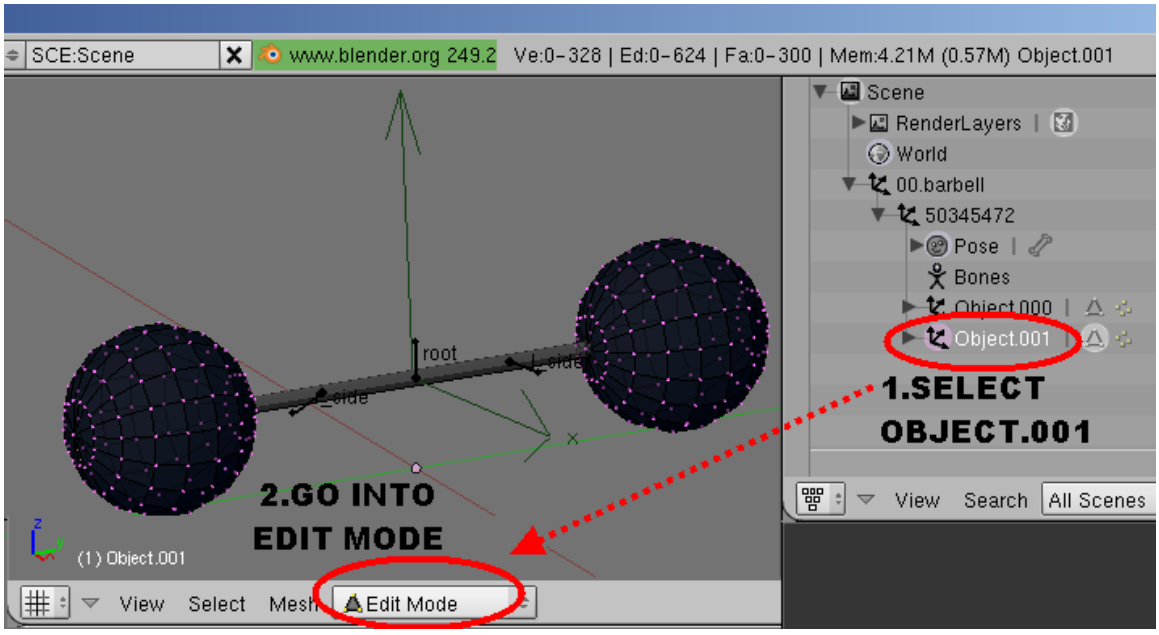


• • •

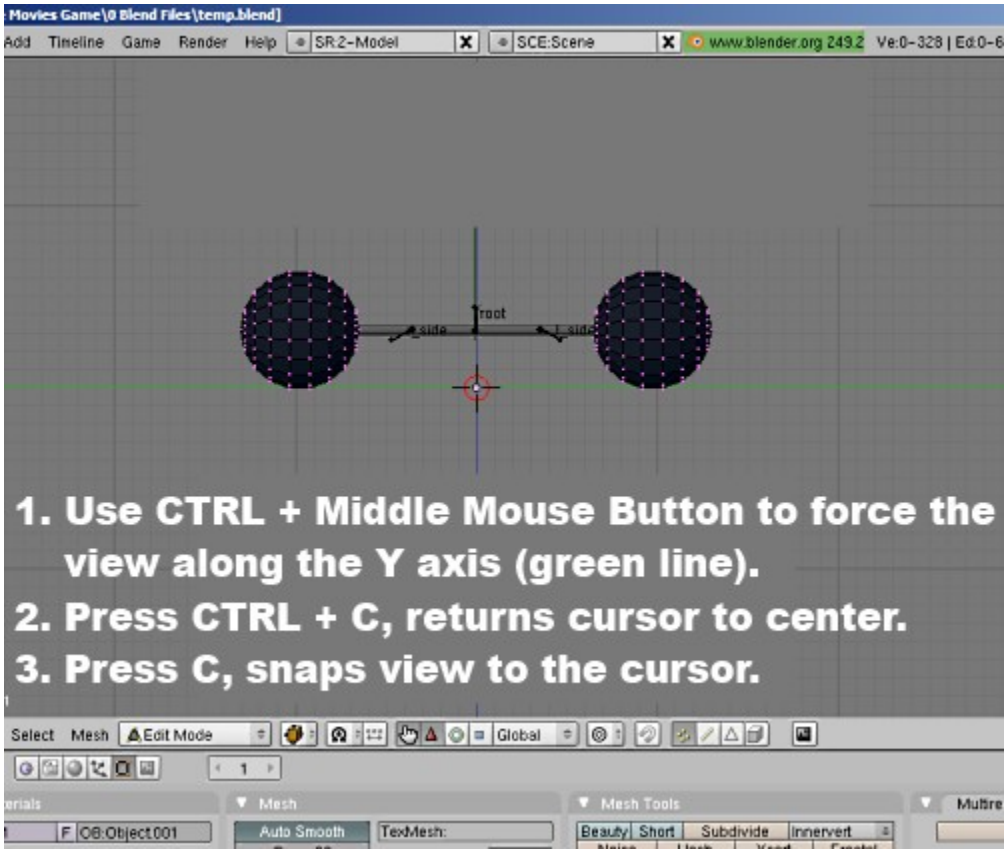
Now it is clear what the parent child relationship is. And it is also clear what the armature is for, which is to move the child objects. We also know what weight groups are, they match the names of the bones belonging to the armatures, and these tell the child objects to move with the bones. We know weight groups can be changed added deleted or edited. We know that Bones can be moved.

Let's do something radical. Let's make a new barbell, but instead of spheres or black balls, lets make 2 cubes. I will post some pics showing how to do this properly. I'm going to take the funny rout. I'm going to add objects to the blackballs (Object.001) while in *Edit Mode*.

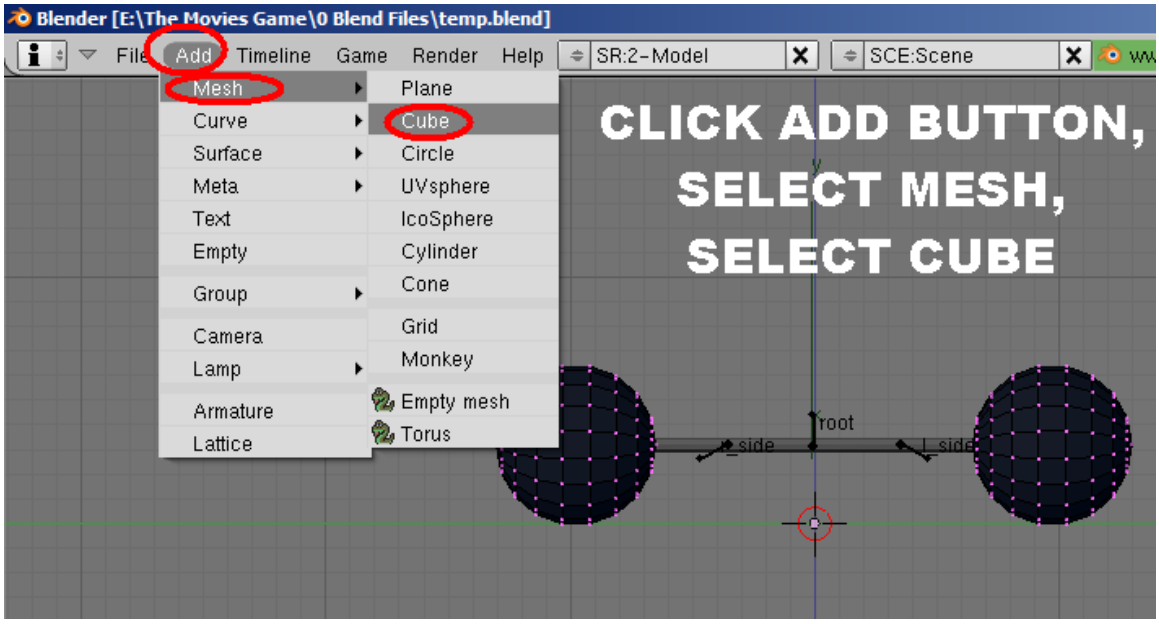
Let's do this. Select **Object.001** in the *Outliner* window and go into **Edit Mode.**

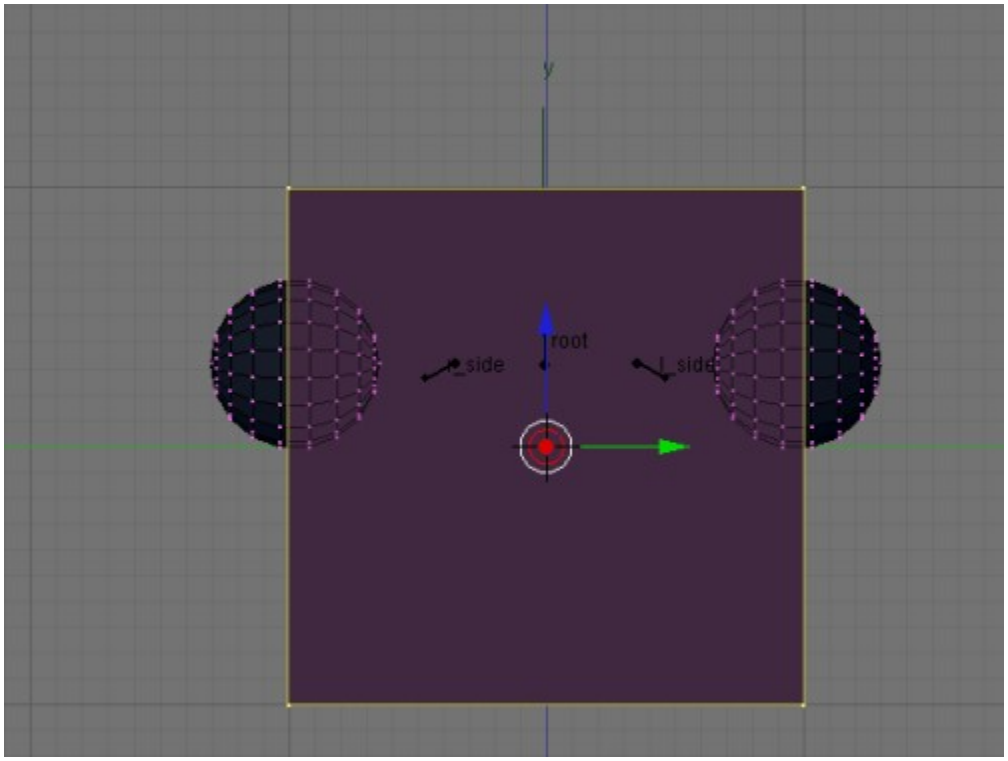


Now we want to center our view exactly by placing the 3D red circle cursor at dead center. Lets line up the 3D view to face the barbell dead center.
Hold down the **CTRL** button and using middle mouse button (Or the mouse wheel pressed down as a button), pivot the view around until it snaps into perfect center alignment.
Press hotkey **SHIFT + C** to have the red circle cursor go to the center.
Then press **C** by itself to have the view snap to it.
In other words get your screen to look like the following picture…

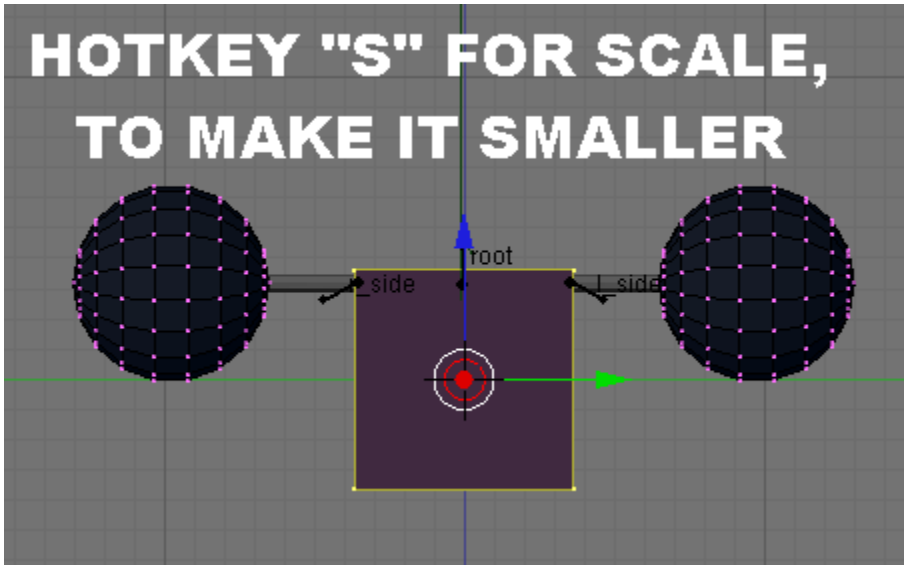


The view being aligned to an axis is perfect for the next step. **Object.001** is selected, (go ahead and select it if not). Then go into *Edit Mode*, **TAB** key. Then deselect every vert so that nothing is selected. Use Hotkey **A** to select all or select none (Select None). Nothing is lit up. Lets add an object to this collection of unselected verts. Go up to the top left of the screen and find a button called **Add**. This will bring a drop down menu. Select **Mesh**, and another menu select **Cube.**

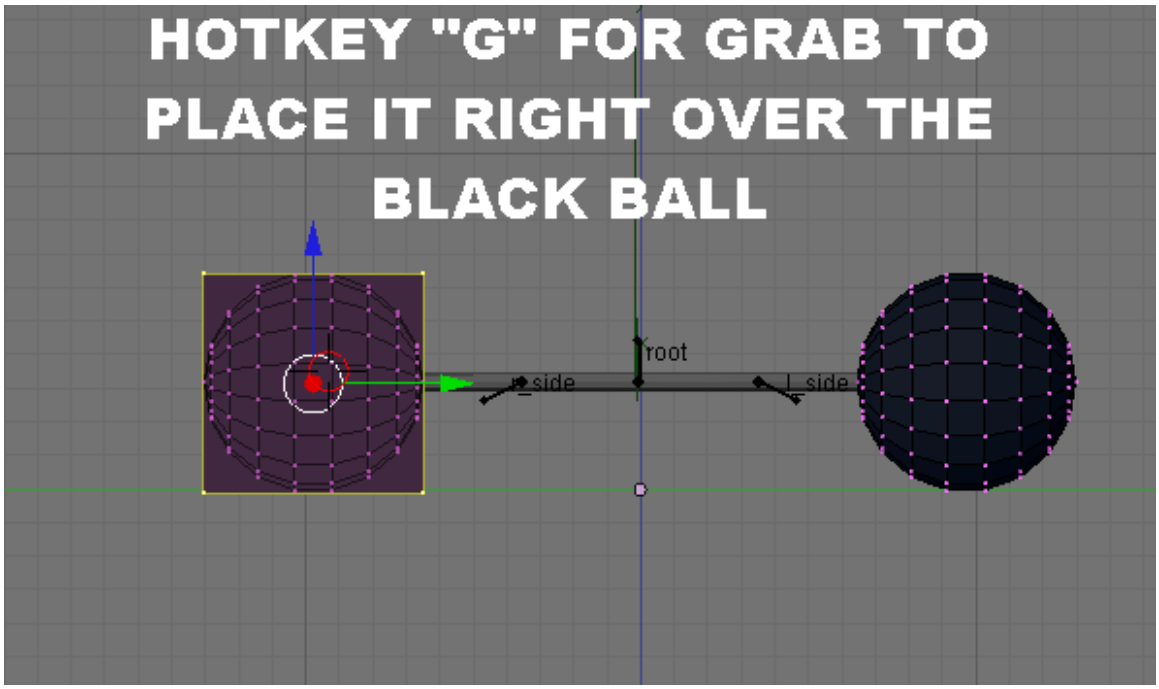




The cube is way too big. We will *scale* it down to size. Press hotkey **S** for *scale*. And move the mouse inward to make it smaller.



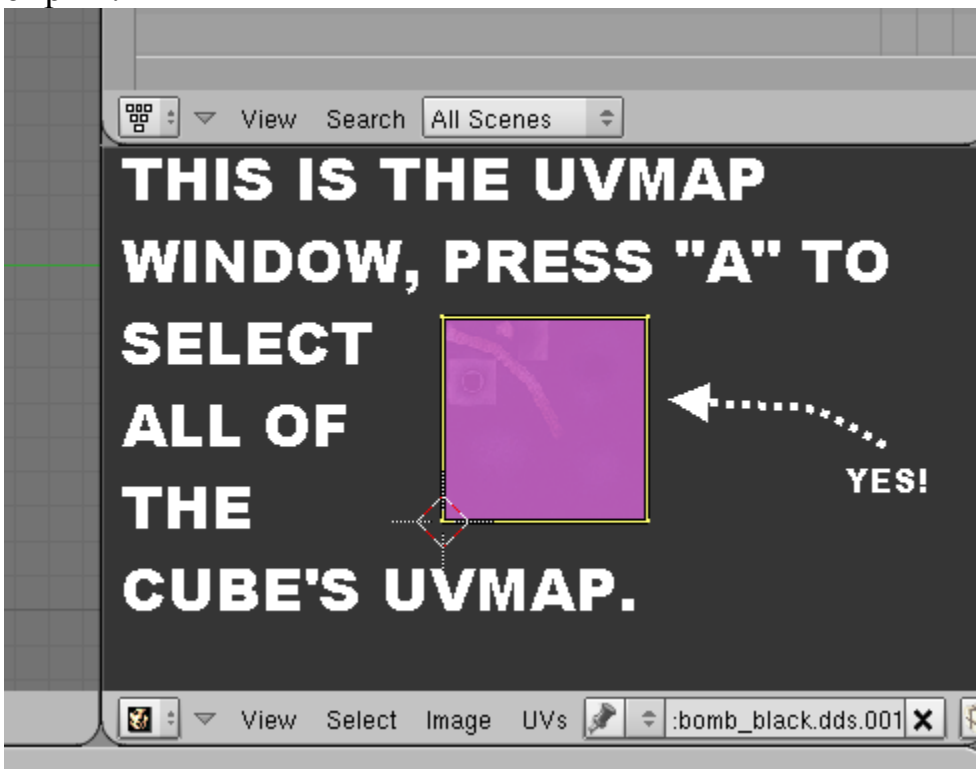
Now that it is smaller, lets move it to one of the black balls. Press hotkey **G** for *grab* and move the mouse until the cube is on top of the black ball. If you want to you can continue to *scale* and *grab* until it is exactly the same size and location.



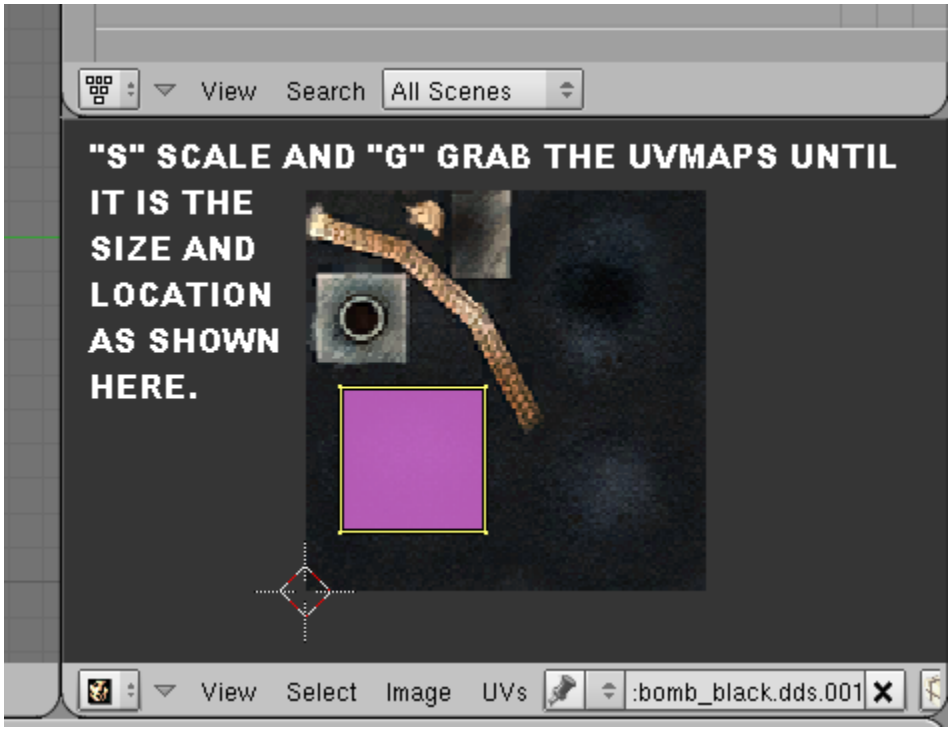
Now that it is there lets move & scale it's UVMAP. It already has one because it was added to an object that already had one. It was added to Object.001 while being in *Edit Mode*. So the new UVMAP was created automatically. But crudely at the same time. Just several squares in the *UVMAP* window, all on top of each other. (So it looks like one square.) But this is good. We don't need to get real fancy. Head over to the uvmap window and hovering the cursor in that window, press the **A** key. This

selects all, but will select all in the UVMAP window. Be sure to do it while the cursor is inside the *UVMAP* window. If you hit **A** key in 3D window it will deselect all the verts there. If done so by accident, then click on one of the verts of only the Cube and then press **CTRL** + **L** and all of the Cube will get lit up. We don't want any of the black balls selected for this. Just the Cube.

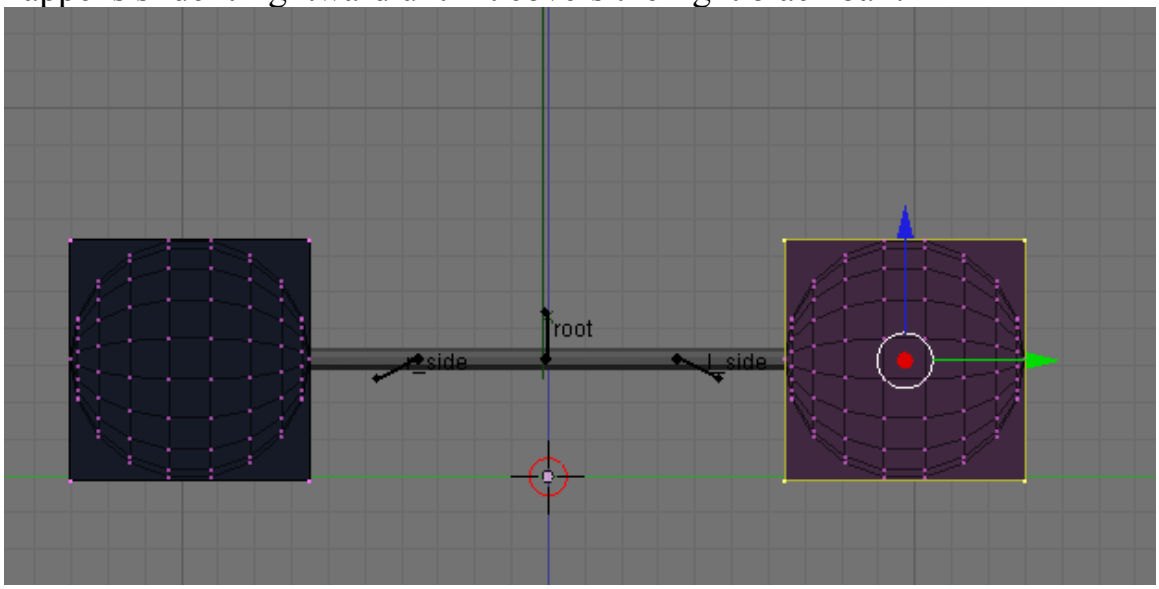
You know all of the Cube's UVMap has been selected when it turns purple or pink.



Now we want to *scale* the uvmaps down to a smaller size. Press hotkey **S**, again while cursor is in *UVMAP* window and not in the 3D window. Move mouse inward to shrink the squares down to a smaller size. Use **G** for grab and position it where it looks like in the following picture.



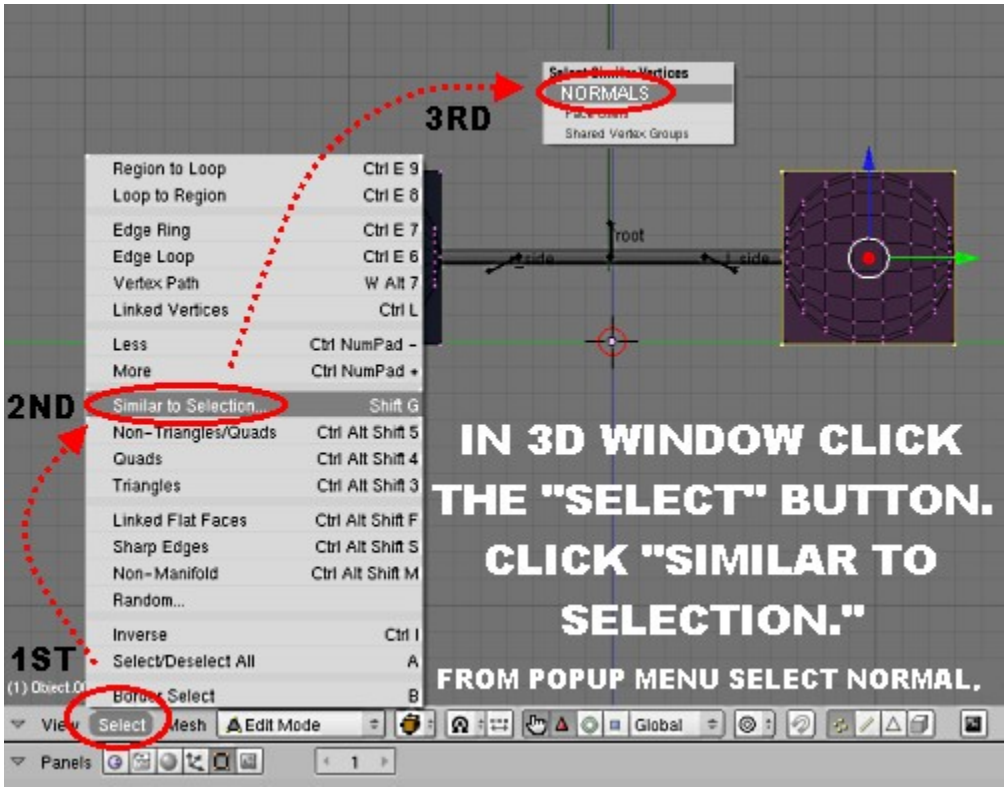
Next we want to double the Cube's verts in the 3D window. Still in **Edit Mode**, still having only the verts of the Cube selected, press **CTRL + D** (for *create double*). After the double is created, you already are in a mode to place the double. You can move the mouse anywhere and the double will follow it. But here we will confine it's movement along only the Y axis by pressing **Y** key. A white line to the left and to the right will show up. The new cube will slide leftward or rightward along this white line. When this happens slide it rightward until it covers the right black ball.



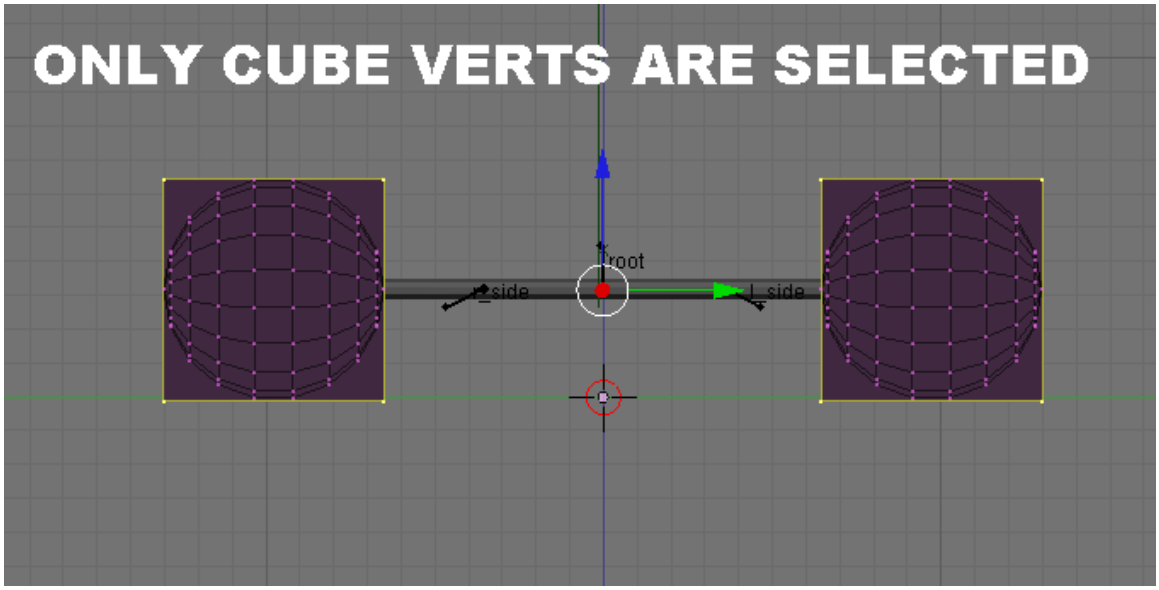
A fantastic thing happened here. The second cube also has the same UVMap

of the first cube. We will not have to scale or position it like we did the first time because the double also copied it's uvmap.

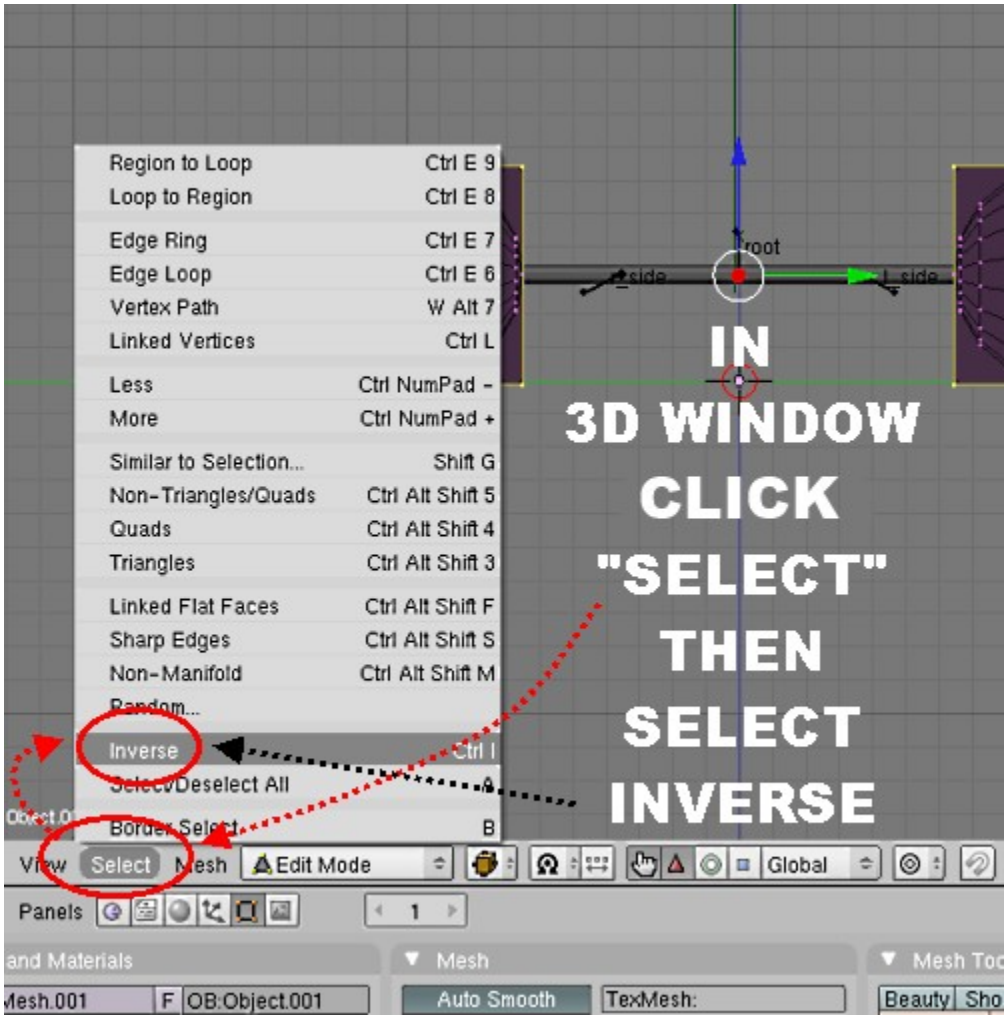
Now we want to delete the vertices belonging to the Black Balls and KEEP only the verts of the 2 Cubes. There are a hundred ways to do this. But we want the easiest. Here's what we will do. In the 3D window is a button called **Select**, find it at the bottom of the window. It brings up a popup menu. Select **Similar To Selection**. This will bring up yet another popup menu. Select **Normals.**



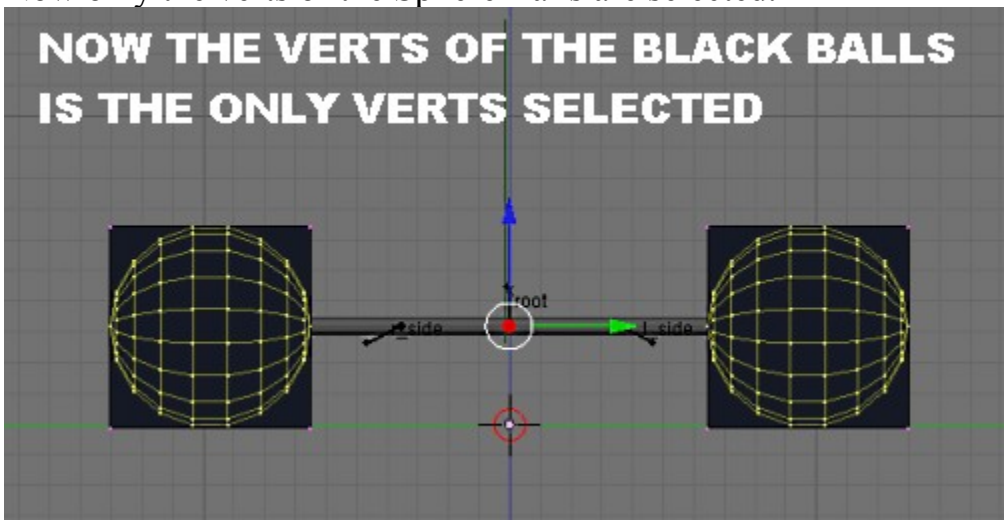
Now the other cube verts have become selected. Leaving the black balls verts unselected.



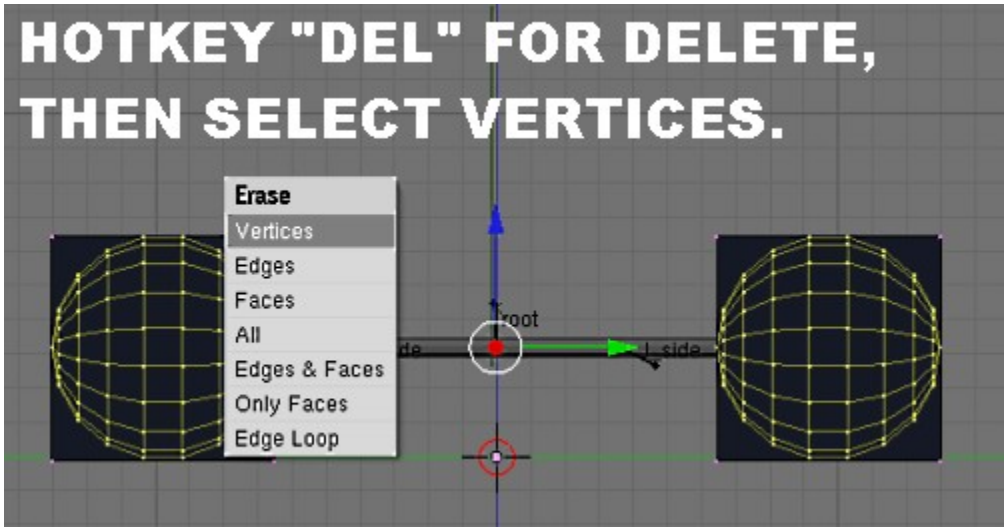
You may (or may not) be asking why would we want to select the verts we want to keep? The same **Select** button has another option called **Inverse.** The Cube's verts we want are selected. (This meaning we have none of the verts we don't want selected.) Now we will do an inverse selection. We will 'flip' the selection of the verts.

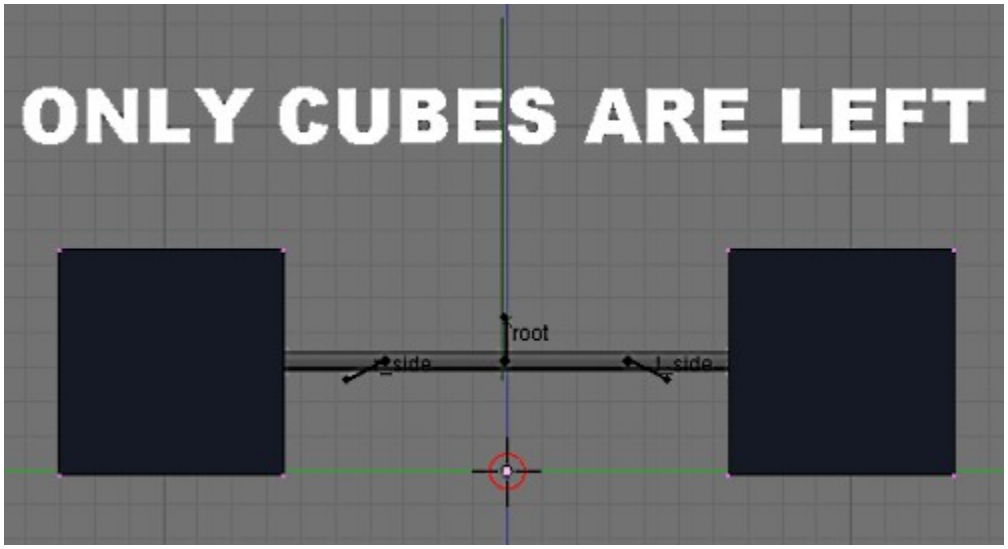


Now only the verts of the Sphere Balls are selected.

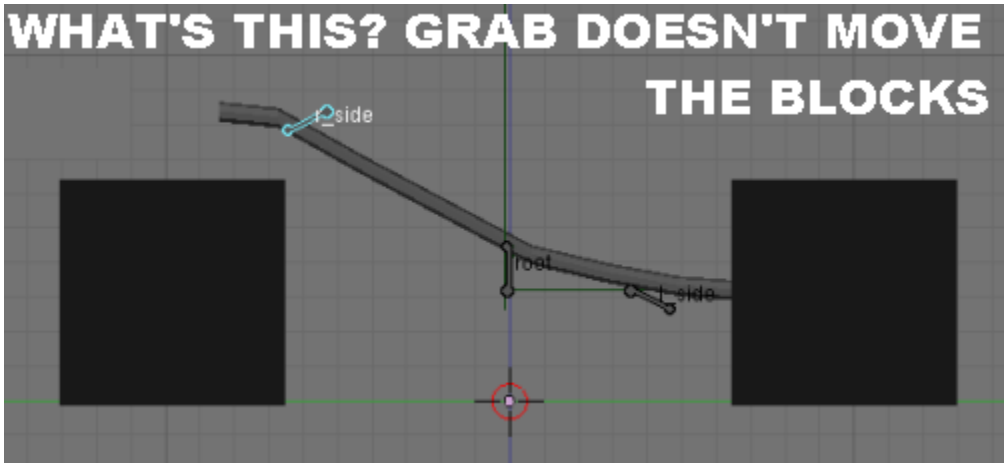


From here we will now *delete* the verts that are selected. Only the balls verts. To do this press hotkey **DEL** button and when it asks what you want to erase click on Vertices.



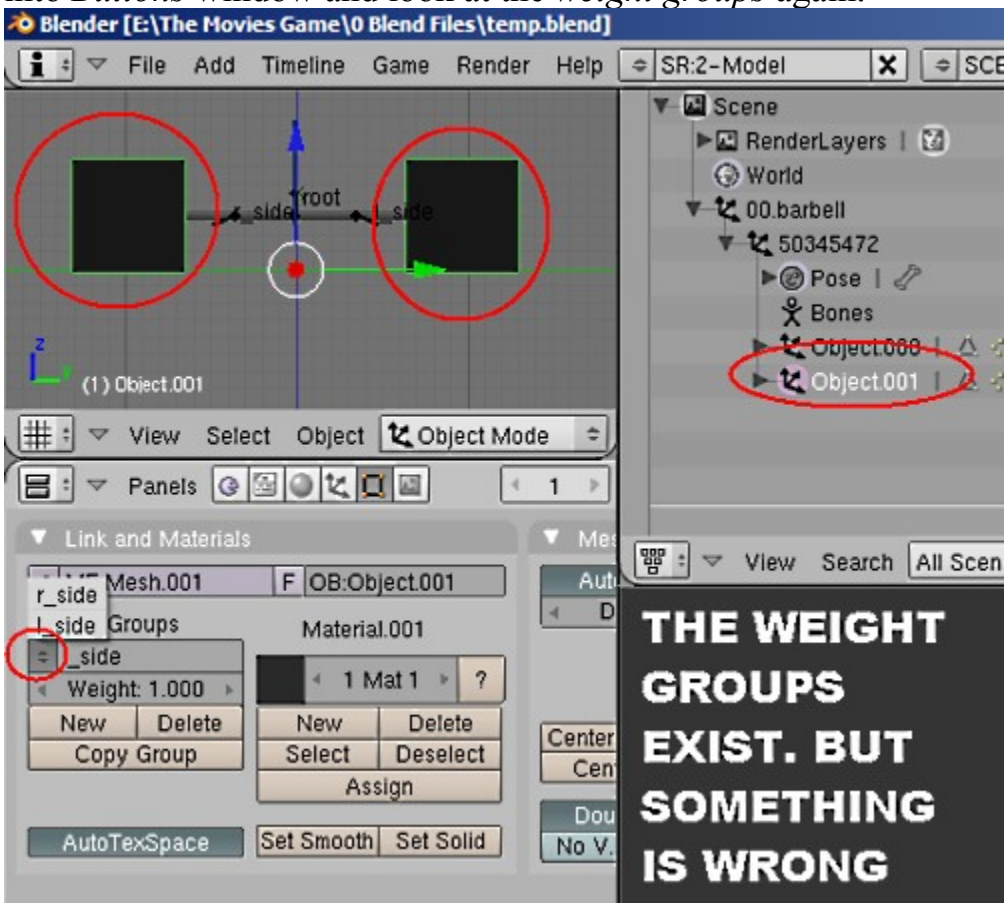


Now we are left with only the Cubes. This may look silly for the Barbell scene, but so what. It gives us a chance to explore weight groups and the armature. What will we do next? Let's try moving the bones again to see what happens. As we did before, in the **Outliner window**, click on the Armature object called **50345472**. Then go into **Pose Mode** in the 3D window. For me it was already in *Pose Mode* because we left the armature that way when we went on to other stuff. If not go into *Pose Mode*. (You should know by now how to do it, and if you don't go back up in the tutorial for it... it's a little box button at the bottom of the 3D window.)



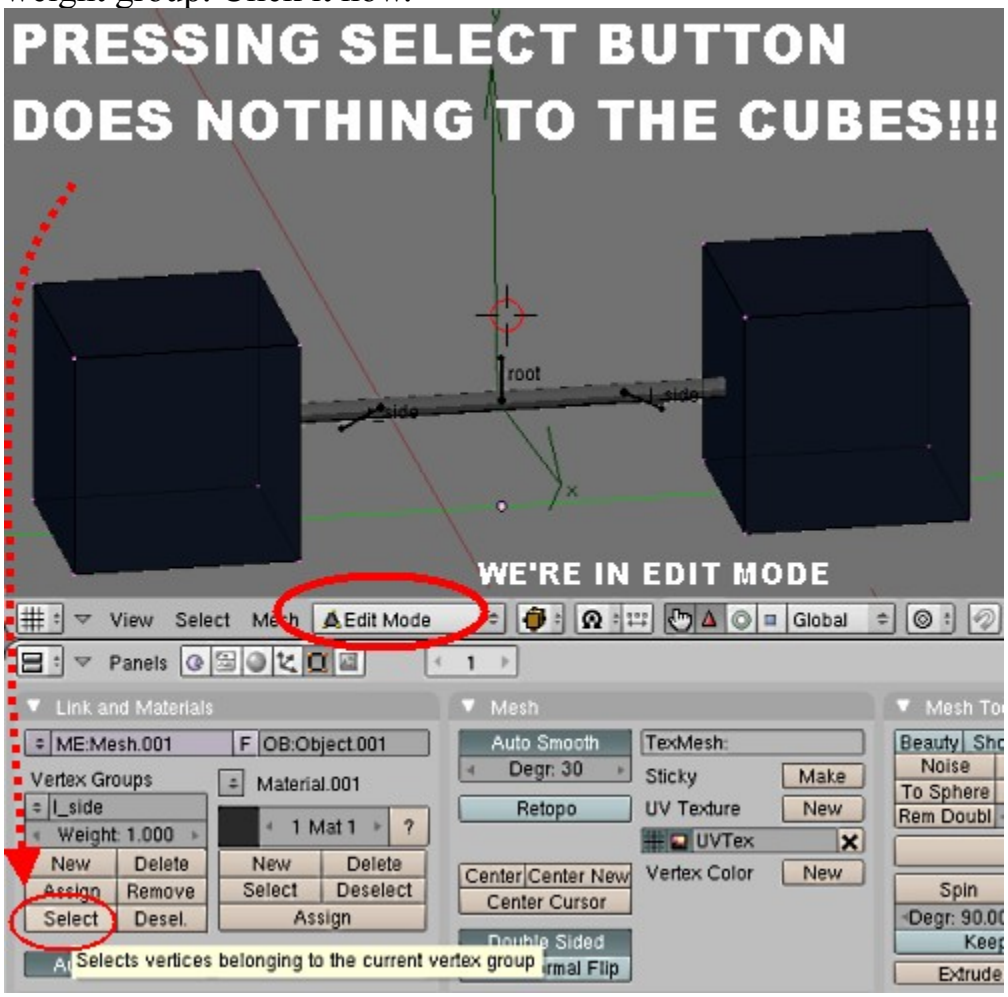
Now what is this? Select a side bone and then press **G** for grab. Move the mouse around and... The box isn't moving!!! And why is that? They were added to the spheres!? They still have *weight groups* we can see in the *Buttons* window!? What's going on?

Go to the *Outliner* window and click on the Object.001. Now press **F9** to go into *Buttons* window and look at the *weight groups* again.

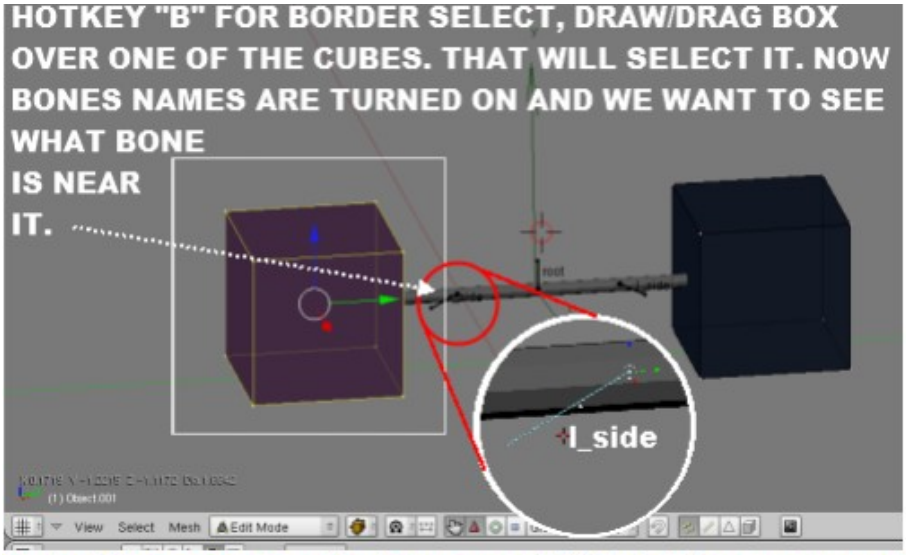


Let's go into Edit Mode to get a better idea. Now make sure no verts are

selected. Use **A** key until all verts are NOT lit up or yellow. They should be quiet. Now in the buttons window **F9** at the far left, is a little button called **Select**. We used this button before to see which sphere was assigned to that weight group. Click it now.

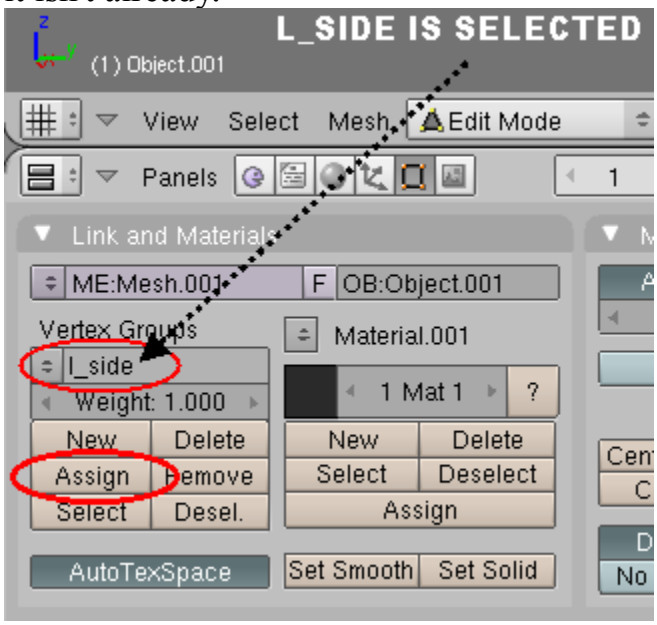


Wow. Nothing is happening in the 3D window. This is because when we added the cubes, only new verts were added to Object.001 (which used to be the black balls). Remember only the verts of the black balls were assigned to the weight groups. Adding new verts does not automatically let them use the weight groups. We have to assign the verts of the Cubes to them. To do this we need to select only one cube with the border select button. Hotkey **B.**



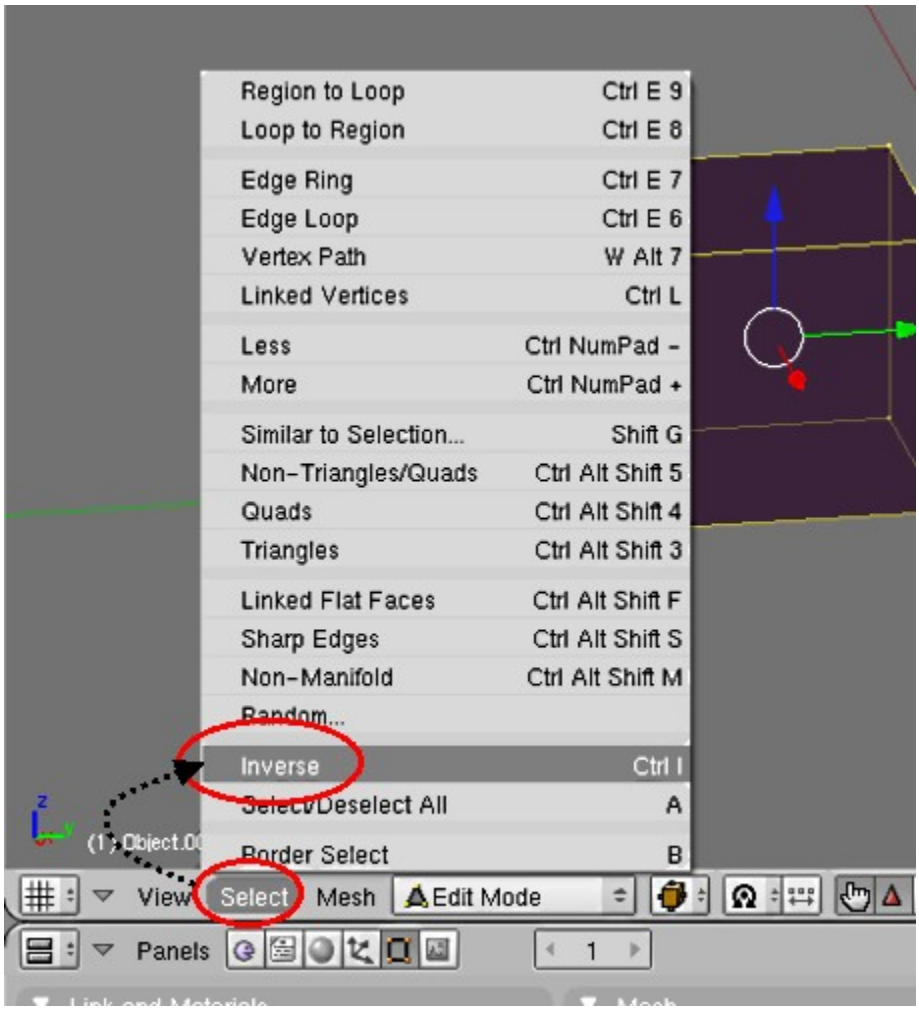
Note: Remembering that Bones and Weight Groups share the same name...

Now that it is selected, we need to see the bone that is closest to it. In this case it is **l_side**. If yours says **r_side** then you are viewing it at the opposite side, the back side. You can pivot the view around to get the right side or just use that bone. Either way when you see which is the bone nearest to the cube you have lit up and selected, go into the *Buttons* window and make sure that *weight group's* name is in the little box. That will make it selected. Click the little arrows at the left side of the box to bring up a menu and select **l_side** if it isn't already.



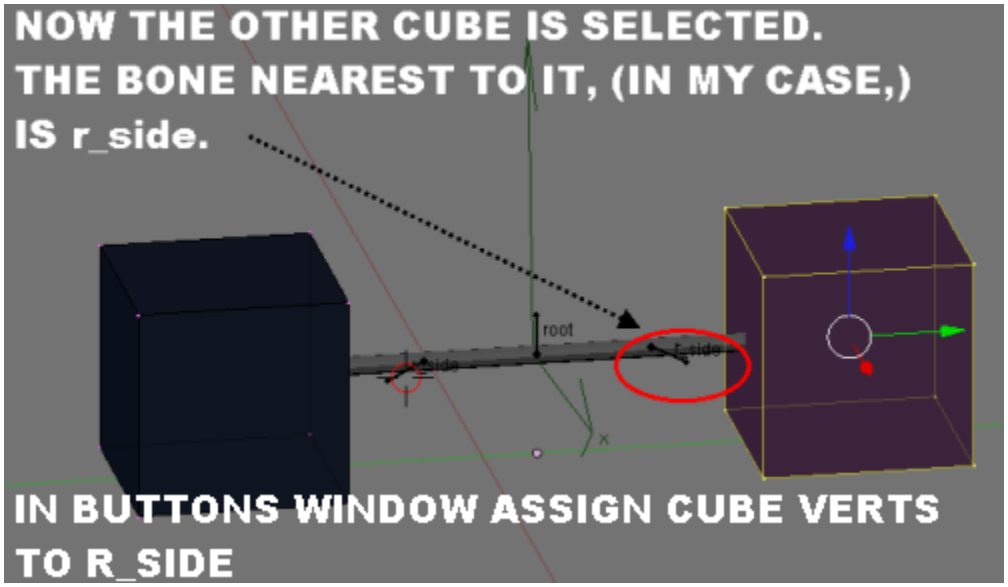
Now hit the button below it called **ASSIGN**. Now the selected verts of the nearest cube have been assigned to that *weight group*. Let's do the reverse. Remember when we used Inverse select to reverse what verts were selected, let's do that again. In the 3D window, at the bottom, is a button called **Select**.

Clicking it brings up a menu, select from it **Inverse.** Now the other cube's verts are selected.



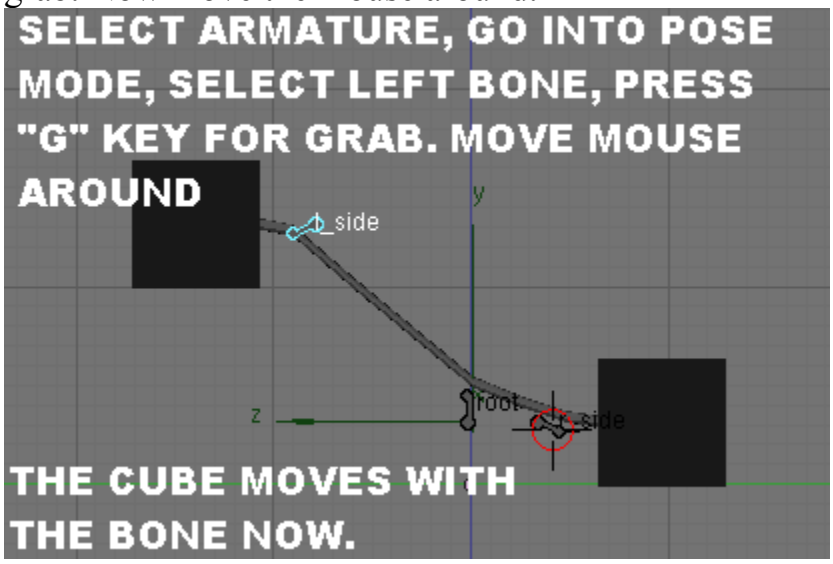
The bone nearest to it, in my case, is the **r_side** bone. You know by zooming in (and because the armature's names button is on) we can see the bone's name. Now I go to the buttons window again, just as before, and make sure the **r_side** name is selected in the little box for *weight groups*. Once I made sure of that, I hit the **Assign** button so the other cube's selected verts gets assigned to it.

Let's be clear here. It may be your view is on the other side of the barbell, in which case the names might be reversed for the bones. Zoom into the names to see them. If it is hard to see the black names then in the Outliner click on the armature. Maybe go into Pose Mode and select the bones individually and the name will light up. Make certain that the name you see nearest to the cube you are working with is the same name of the *weight group* being assigned to it. If you assign the wrong group then you will get funny results.



The verts of the Cube nearest to bone l_side must be assigned to the weight group called l_side.
The verts of the Cube nearest to bone r_side must be assigned to the weight group called r_side.

The weight groups have now been assigned to the cubes, in Object.001. We will now get better results when we animate, or move the bones around. Lets select the armature, then go into Pose Mode (if you are not in Pose Mode already). Now select the left bone by clicking on it. And press **G** key for grab. Now move the mouse around.



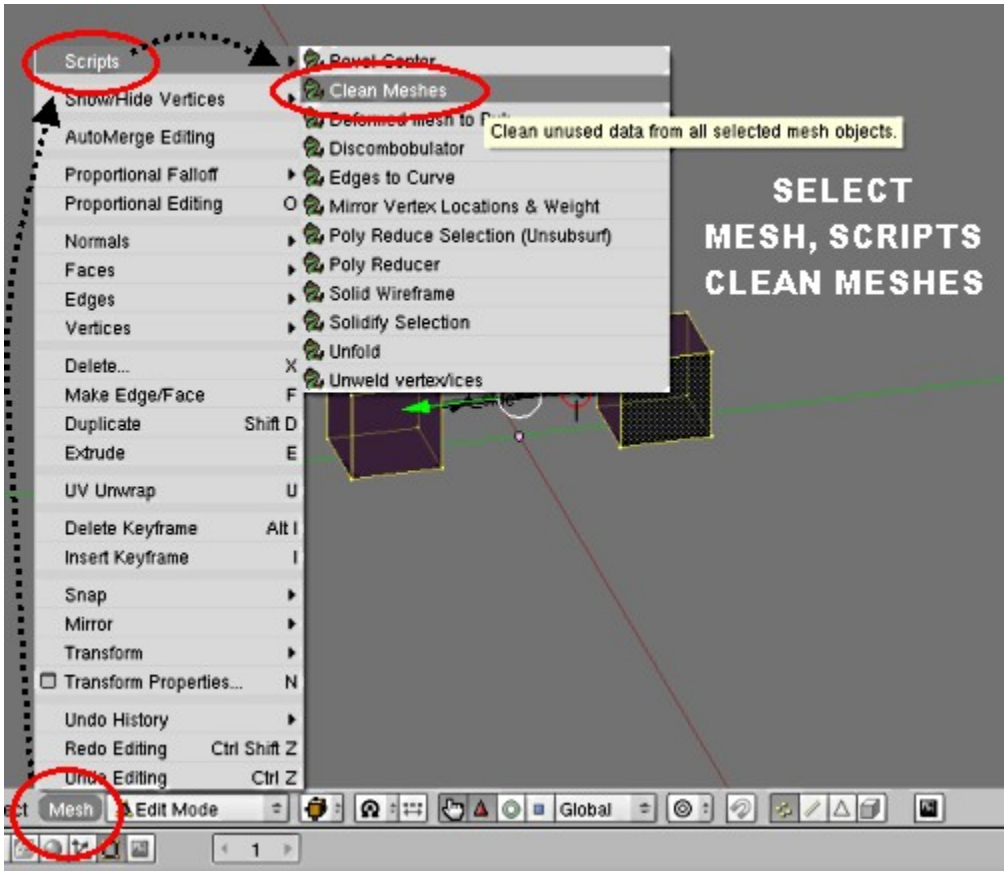
That's better. So what next? Let's export the new barbell and make it a new mod. So what that it is cubed? The only real evidence that any of this will work is to see it work in the game.

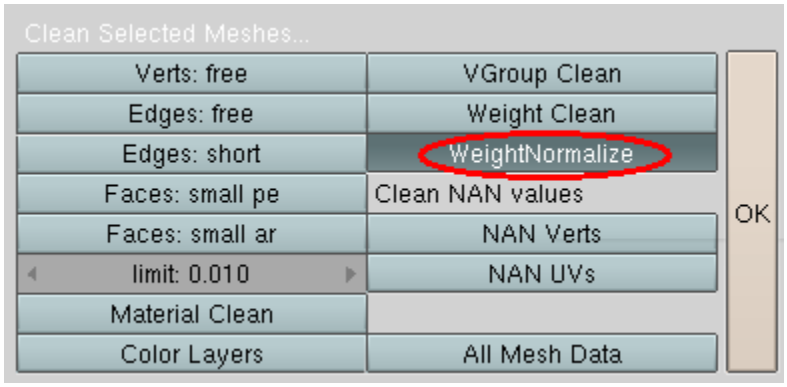
There are a couple things I do to READY a project for export. Some of you may have followed my mad science work. If so then you also know that right now this isn't merely a barbell but a giant barbell. I'm not kidding. The steps we just took, in Blender are just ordinary steps. But the way we just assigned those weights to the cubes, once it is in the game, will make them double in size during animation. Why? Because some very strange situation takes place when you assign weights right from the Vertice groups window. And it only gets noticed in The Movies Game!!! But I don't need a giant barbell. Not yet anyways. It would work with my giant costumes, but not my normal costumes. If I exported this project right as it is, in the game a giant version of the barbell will be floating up above the actor/actress on the stage set.

If you need to know more about what I am saying you will have to visit my tutorial for making giant costumes. There is a fix for this problem. A script that will even out the weights. It seems to neutralize the giant effect. Most people "Paint" the bone weights onto models. Painting the weights does not produce the giant bug. I rarely paint weights. Almost never. I always use the buttons window and that is why I found out about the giant trick. When new weights have been assigned and I always do a customary *Normalize the Weights* before export.

(Update: Lefty2000 has made a python script for creating proper giants and what I am talking about is the old way we used to make giants.)

Make sure **Object.001** is selected. Go into *Edit Mode*. Select all of the verts (Both of the cubes) by using hotkey **A** (all). Until all verts light up and are selected. At the bottom of the *3D* window is a button called **MESH.** Clicking this button brings up a popup menu. Go to the top and select **Scripts.** This bring up another menu, select **Clean Meshes.** Now a fancy menu comes up. This has all kinds of handy tools but we only want one of them right now. **Weight Normalize.** Perhaps the **Verts: Free** button is already selected. Deselect it. Now select **Weight Normalize**. Press **OK** button to finish it.





Now the barbell won't be a giant. You don't have to understand the giant thing, just take my word for it. It will mess up the whole thing. If you want to make something a giant pick something cool. But not this, at least not now. That being done let's run **The Movies Preflight Check**. Turn the UVMAP window into the **Scripts window**. We did this earlier. It's a little icon at the bottom left of the window. Clicking it brings up a menu of windows to choose from. Click on the **Scripts** window. Now we have next to the icon a button called **Scripts**. Clicking it brings up another menu. From here select **MISC.** From the next menu select **The Movies Preflight Check.**

The Movies MSH Export Preflight Tests

Start Time: 2014-06-13 23:27:02
Duration: 0:00:00.093000
Status: Pass 149

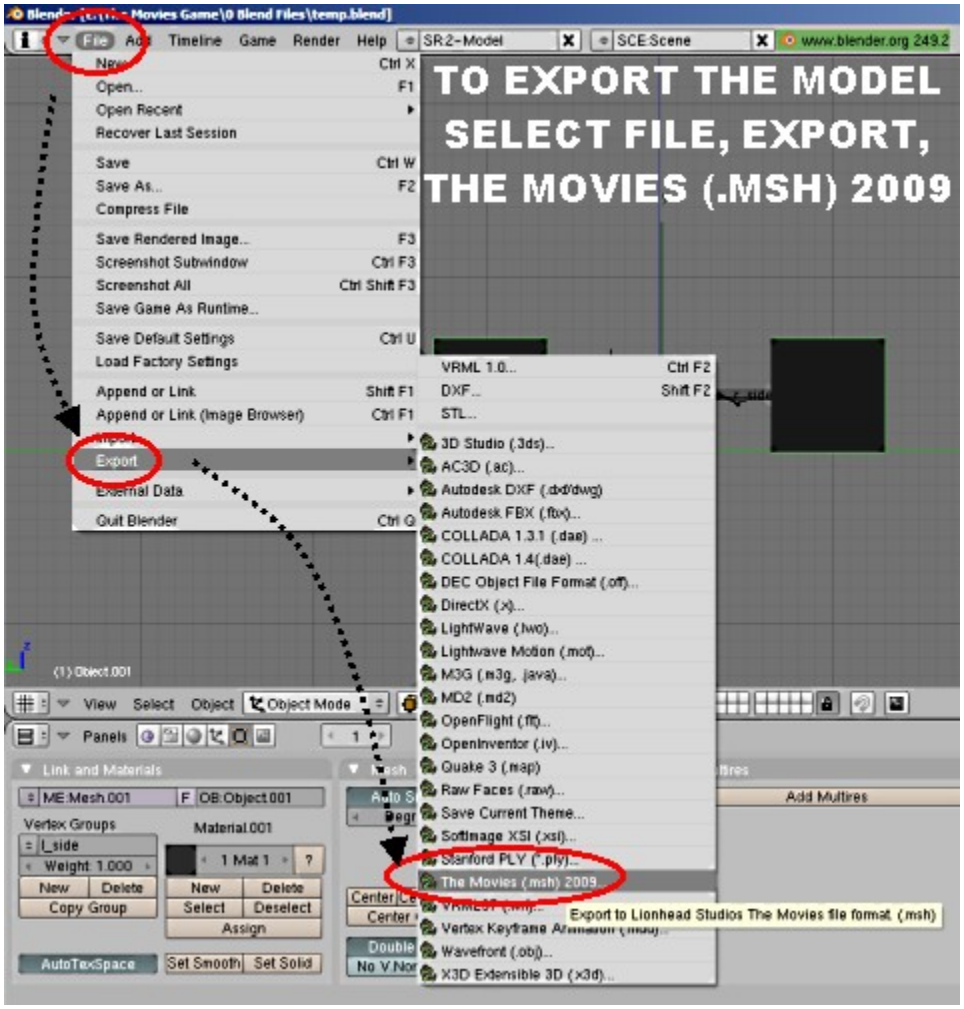
Results for checking validity and integrity of mesh to be exported to The Movies game format.

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
BasicTests: Verifies very basic requirements.	4	4	0	0	Detail
ArmatureTests: Tests several aspects of any Armatures in this mesh.	4	4	0	0	Detail
InGroupTest: Each Blender Object is in a	3	3	0	0	Detail
MultiGroupTest: Each Blender Object is	3	3	0	0	Detail
GroupHasEmptyTest: Each populated B	1	1	0	0	Detail
GroupHasOrderNumberTest: Each Group	1	1	0	0	Detail
Group-ParentName: The name of the Grou	1	1	0	0	Detail
Group-PropertyName: The name of the Group and the IDProperty 'grpName' value are the same.	1	1	0	0	Detail
GroupHeirarchyTest: Meshes must be children of an Empty or an Armature.	2	2	0	0	Detail
BoneVertexGroupName: Vertex Groups must match Bone names.	2	2	0	0	Detail
Parent-PropertyName: The IDProperty 'grpName' and the name of the Empty are the same.	1	1	0	0	Detail
MaterialZeroTest: Each Grouped Mesh has a material in the	2	2	0	0	Detail
MaterialTypeTest: Textures in Mate	2	2	0	0	Detail
UnusedMaterialsTest: Checks for unused	2	2	0	0	Detail
UnusedTexturesTest: Checks for unused	8	8	0	0	Detail
UnusedImagesTest: Checks for unused I	2	2	0	0	Detail
UVTest: Meshes must have UV mapping.	2	2	0	0	Detail
UVLayerNameTest: UV Layers must be named 'UVTex' or 'LightMap'.	2	2	0	0	Detail
NonFaceVertsTest: Checks for vertexes that are not in any faces.	2	2	0	0	Detail
VertWeightTest: Checks for vertexes that are not weighted to any Vertex Groups.	104	104	0	0	Detail
Total	**149**	**149**	**0**	**0**	

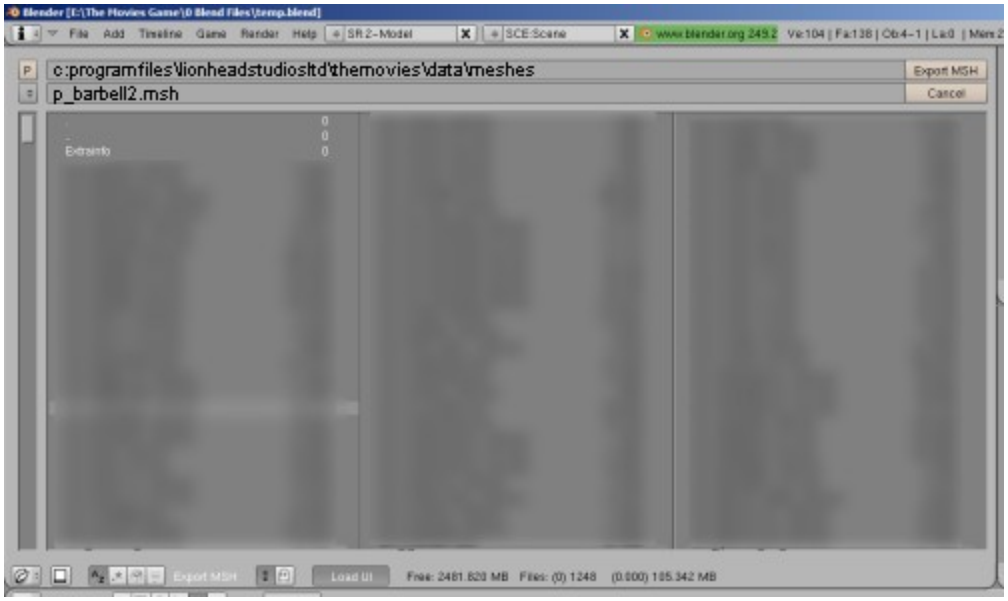
Nice, all green lights. That means there were no errors and we can export the new barbell to The Movies Game. If you followed what I said then you will have green lights too. If you do have errors, then start over. Keep in mind, this was a lot to read but could be done in less then 2 minutes. Right now you are learning what to do so it may seem a lot to digest. But really this is child's play compared to the mods that you will be making soon.

To export the model go to the top right of the screen and find a button called **File**. From the menu select **Export**, and from the next menu select **The Movies (.msh) 2009**. This will bring up a browser window. Navigate to where The Movies models need to be exported to.



This is usually at
C:\Program Files\Lionhead Studios ltd\The Movies\Data\Meshes

Once there you need a name for the model. I am titling mine **p_barbell2.msh**. Be sure to have the .msh tag follow the name at the end.



After it is exported it is ok to close Blender. This prop is unlike other scene usable props. The Barbells can not be selected from a list, as you could for pistols, or horses and such. There are 3 scenes that were made exclusively for the barbell prop. And to get a new barbell prop to show up you will have to make new scenes for it. You will have to use **FLM READER ZERO** to make them. This is actually easier to do then making a new prop using MED (The Movies Game Editor.)

If you think about it *Scene* files (.FLM files) treats the barbell just like an actor. So what we just did was actually making another actor in the form of a barbell. Silly.

Hopefully you have FLM Reader Zero to do this. If you don't then you could HEX Edit the new mod into a new scene. But I won't go into it now. You can download FLMRZ from TheMovies3D.com.

The first thing you will need is to extract two files using MED (The Movies Game Editor). Use file extractor to extract two files.

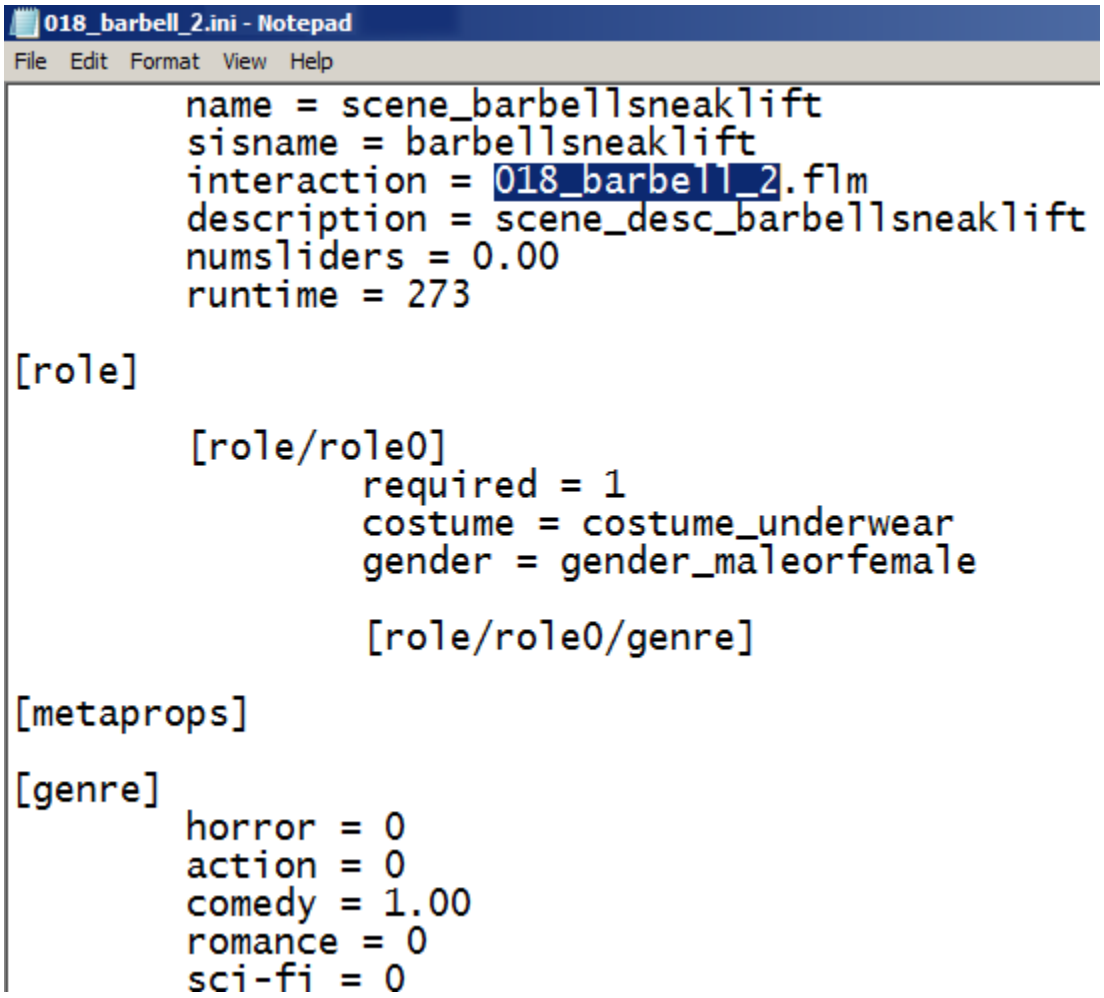
018_barbell_sneak_lift.flm and **018_barbell_sneak_lift.ini** without quotes. In your movies game data\scene folder you will need to have more folders made. If you extracted these files here then you won't have to. If you extracted both of the files to a workspace then copy the data folder and navigate to where your movies game was installed. C:\Program Files\ Lionhead Studios ltd\The Movies and paste it there. Next we have to rename these files so that they do not overwrite the original files.

I will rename them to **018_barbell_2.ini** and **018_barbell_2.flm**.

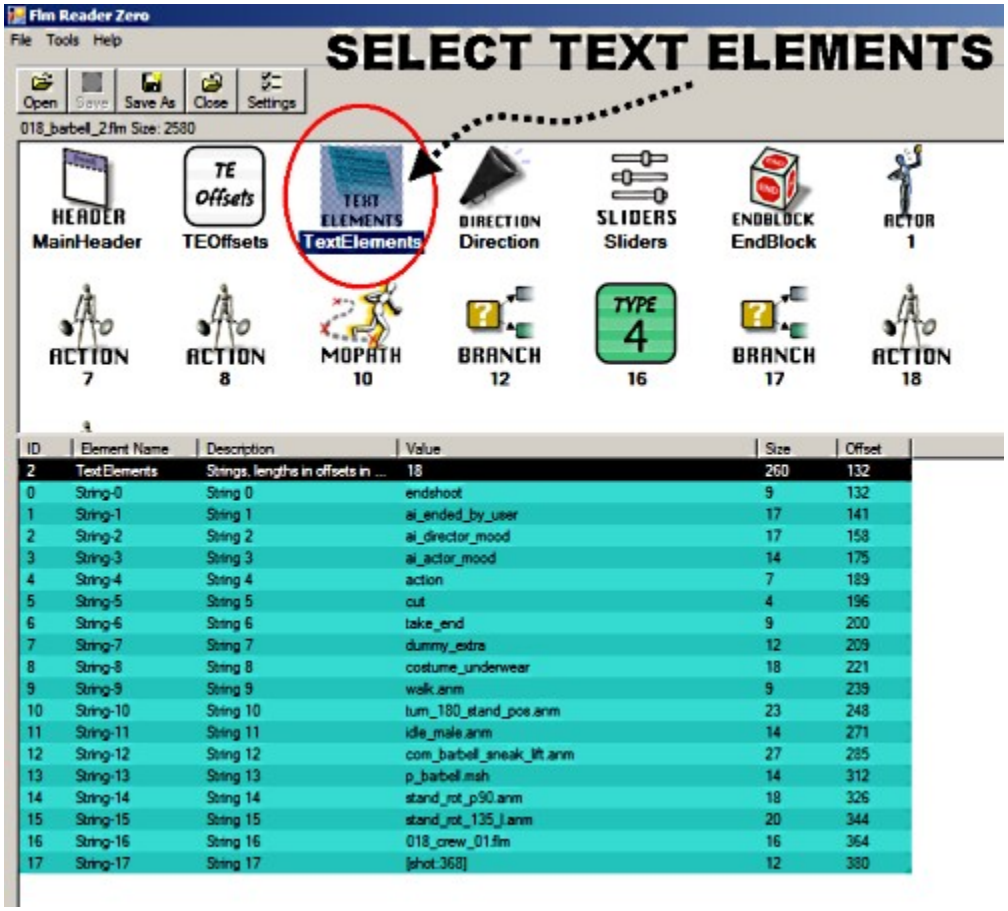
The (.INI) file goes into the data/scene/018 folder, and the (.FLM) file goes into the data/scene/interactions/018 folder.

When the files are renamed and in the right locations, you need to open

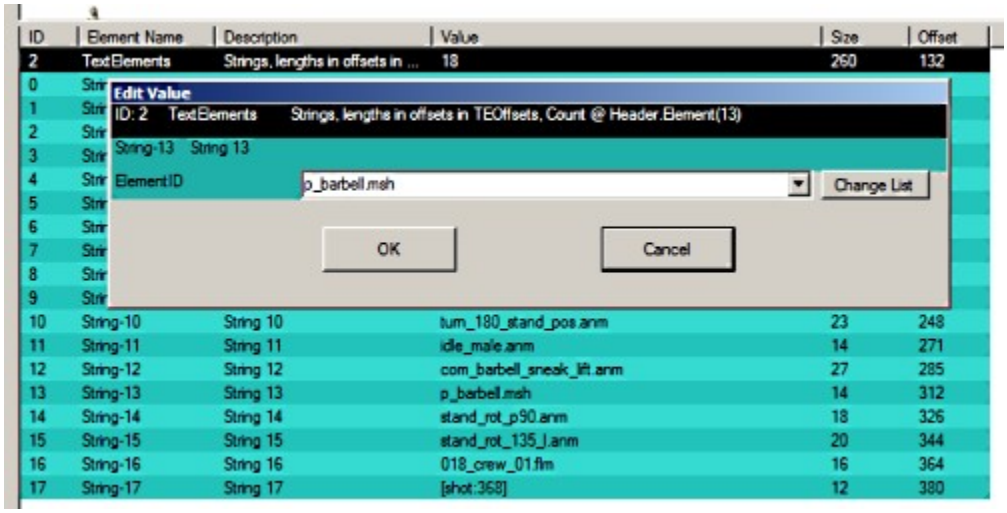
018_barbell_2.ini with Notepad and find a reference to the (.FML) file it calls in the game. The third line.
interaction = 018_barbell_sneak_lift.flm.
We need this file to see a new scene we will create. So overwrite sneak_lift and replace it with the number 2. So that now it says:
interaction = 018_barbell_2.flm.
Save the file.



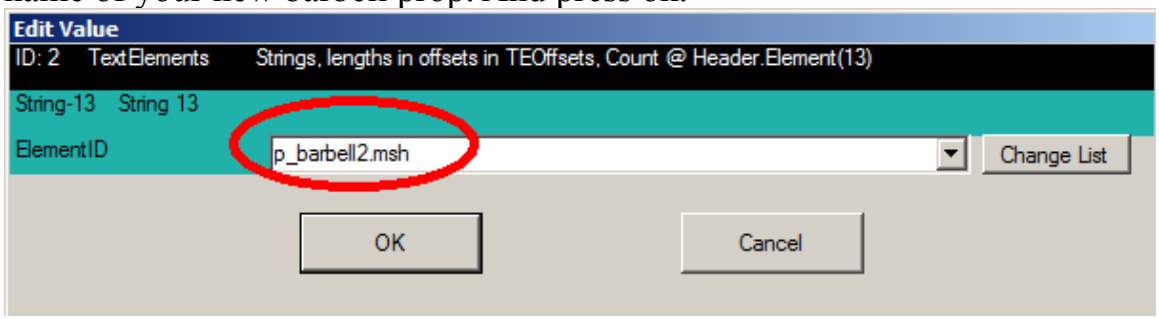
Next step is to open **FLM Reader Zero**. At the top left is a button called **Open**, navigate to the data\scene\interactions\018 folder and select **018_barbell_2.flm**. The top screen has a bunch of blocks to choose from. Select **Text Elements.** The bottom screen will change and have a bunch of lines. In this list is the name of the barbell 3D model. Right now it is called: **p_barbell.msh**. That is the old one. And this new scene is still using the old one. Instead let us replace that name with the new one we made. Double click on the mesh name and a popup window appears.


Flm Reader Zero
File Tools Help
SELECT TEXT ELEMENTS
Open Save Save As Close Settings
018_barbell_2.flm Size: 2580
HEADER MainHeader
TE Offsets TEOffsets
TEXT ELEMENTS TextElements
DIRECTION Direction
SLIDERS Sliders
ENDBLOCK EndBlock
ACTOR 1
ACTION 7
ACTION 8
MOPATH 10
BRANCH 12
TYPE 4 16
BRANCH 17
ACTION 18
ID | Element Name | Description | Value | Size | Offset
2 | TextElements | Strings, lengths in offsets in ... | 18 | 260 | 132
0 | String-0 | String 0 | endshoot | 9 | 132
1 | String-1 | String 1 | ai_ended_by_user | 17 | 141
2 | String-2 | String 2 | ai_director_mood | 17 | 158
3 | String-3 | String 3 | ai_actor_mood | 14 | 175
4 | String-4 | String 4 | action | 7 | 189
5 | String-5 | String 5 | cut | 4 | 196
6 | String-6 | String 6 | take_end | 9 | 200
7 | String-7 | String 7 | dummy_extra | 12 | 209
8 | String-8 | String 8 | costume_underwear | 18 | 221
9 | String-9 | String 9 | walk.anm | 9 | 239
10 | String-10 | String 10 | turn_180_stand_pos.anm | 23 | 248
11 | String-11 | String 11 | idle_male.anm | 14 | 271
12 | String-12 | String 12 | com_barbell_sneak_lift.anm | 27 | 285
13 | String-13 | String 13 | p_barbell.msh | 14 | 312
14 | String-14 | String 14 | stand_rot_p90.anm | 18 | 326
15 | String-15 | String 15 | stand_rot_135_l.anm | 20 | 344
16 | String-16 | String 16 | 018_crew_01.flm | 16 | 364
17 | String-17 | String 17 | [shot:368] | 12 | 380


ACTION 7
ACTION 8
MOPATH 10
BRANCH 12
16
BRANCH 17
ACTION 18
ID | Element Name | Description | Value | Size | Offset
2 | TextElements | Strings, lengths in offsets in ... | 18 | 260 | 132
0 | String-0 | String 0 | endshoot | 9 | 132
1 | String-1 | String 1 | ai_ended_by_user | 17 | 141
2 | String-2 | String 2 | ai_director_mood | 17 | 158
3 | String-3 | String 3 | ai_actor_mood | 14 | 175
4 | String-4 | String 4 | action | 7 | 189
5 | String-5 | String 5 | cut | 4 | 196
6 | String-6 | String 6 | take_end | 9 | 200
7 | String-7 | String 7 | dummy_extra | 12 | 209
8 | String-8 | String 8 | costume_underwear | 18 | 221
9 | String-9 | String 9 | walk.anm | 9 | 239
10 | String-10 | String 10 | turn_180_stand_pos.anm | 23 | 248
11 | String-11 | String 11 | idle_male.anm | 14 | 271
12 | String-12 | String 12 | com_barbell_sneak_lift.anm | 27 | 285
13 | String-13 | String 13 | p_barbell.msh | 14 | 312
14 | String-14 | String 14 | stand_rot_p90.anm | 18 | 326
15 | String-15 | String 15 | stand_rot_135_l.anm | 20 | 344
16 | String-16 | String 16 | 018_crew_01.flm | 16 | 364
17 | String-17 | String 17 | [shot:368] | 12 | 380
p_barbell.msh



The window has the name of the old barbell in it. All we have to do is replace it with the name of the new barbell (One with Cubes), the one I titled: **p_barbell2.msh**. So just click on the name and replace it with the name of your new barbell prop. And press ok.



That's it. All we have to do is save it. At the top left of the screen is a button called **SAVE.** Click on it. It told me the file already existed and asked if I want to overwrite it. Since I renamed this file before I opened it with FLM Read Zero, I said **yes**. Now I close FLM Reader Zero.

Now it is time to run the game and see if everything worked out ok. Open The Movies Game. Go into Custom Script Writing House. Select Stage Set and navigate to the new scene.


1920
Game Paused
$4,697,875


The Baggage Boy

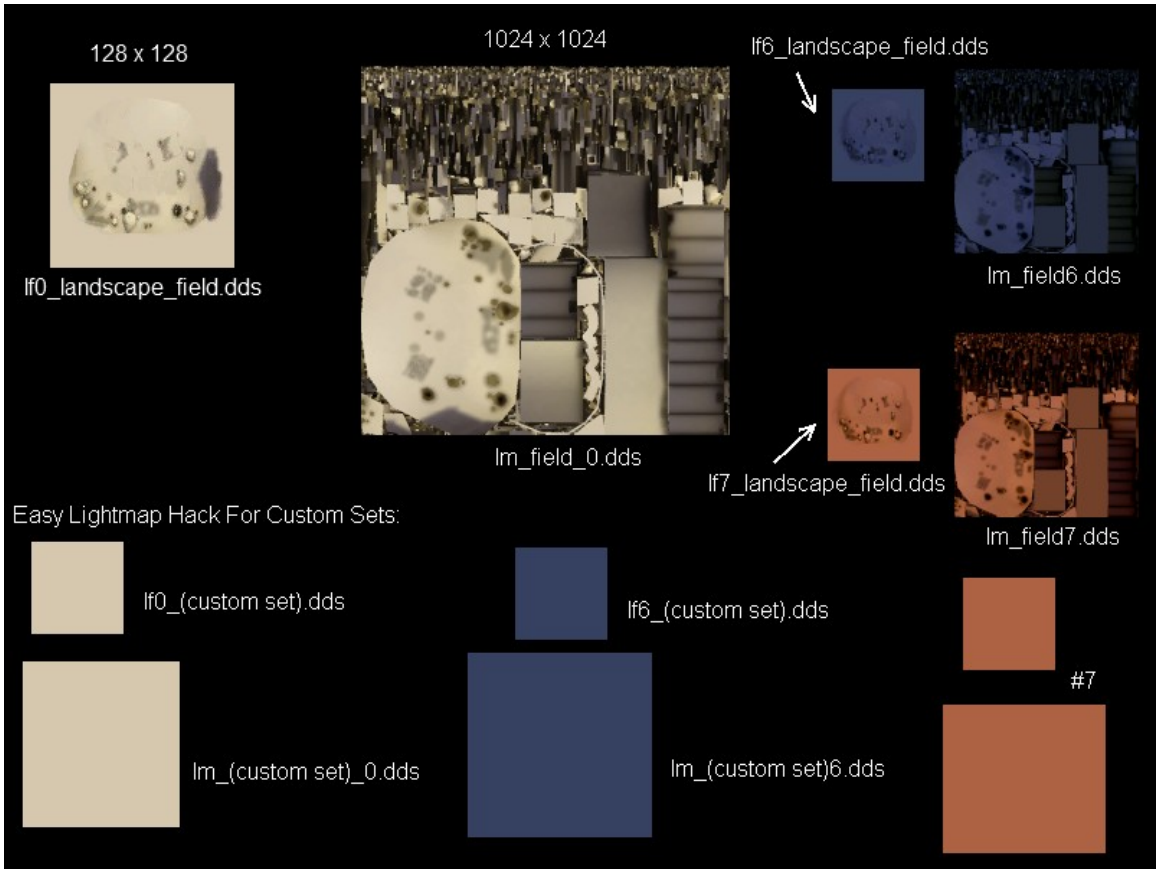
9. Baking Lightmaps For The Movies Game Sets

Lightmaps are image files. Like other Movies image files they are (.DDS) files. They overlay the appearance of a set. An artificial way to produce darkness on a set. They are numbered in name because they are selectable in the custom scripts house.

Movies game sets come with 2 different types of lightmaps that work together. One is used by the set itself. The .MSH file will have the object materials assigned with these. And they change during filming because of the way they are named. This image file is called lm_(name of set)_0.dds to lm_(name of set)_7.dds.

The other lightmap is called lf0_(name of set).dds to lf7_(name of set).dds. For a custom set they could be a solid color, so long as they match the sister version beginning with lm_. Blue for the nighttime, reddish for the dawn or dusk ones.

The first 4 (0 through 3) are for the studio lot. Sets can be placed facing four different directions on the lot. The next 4 or 3 are for filming movies (4 through 7). For custom sets 0 through 3 can be copies of this 5th one. lf4_(name of set).dds.



The easiest way is a quick hack that does not require baking or Blender. If your set is to have more detailed lighting effects or shadows then move on to the tutorial. If you just want a quick time of day, you could just use solid colors.

At the top of the image is the actual lightmaps of the landscape field set. The 128x128 lf_ image is light that is cast over actors or set placed props. Next to it, the 1024x1024 is a part of the model, (which the following tutorial will teach you how to make.) 0 through 4 is a day time lightmap. 5 is another night time map, and so is number 6. The bluish color gives the appearance of night. Number 7 is the reddish ones on the right.

Notice the lf#_ and lm_ numbered maps have the same number for the same color. With so many sets available on 8eyedbaby with no lightmaps, after you download one, you could make these solid color images very quick and give the set a lightmap for your movie. Do so by using a paint program that can save images in .DDS format. Make the first 5 a daytime solid color. (zero through four). And the next 2 or 3 colors a nighttime and a reddish one. Save them in the data\textures\lightmap folder. And use MED (Movies Editor) to assign the lightmap to the set. Or if Med can't import images then use MeshManip. Or Blender if you know how.

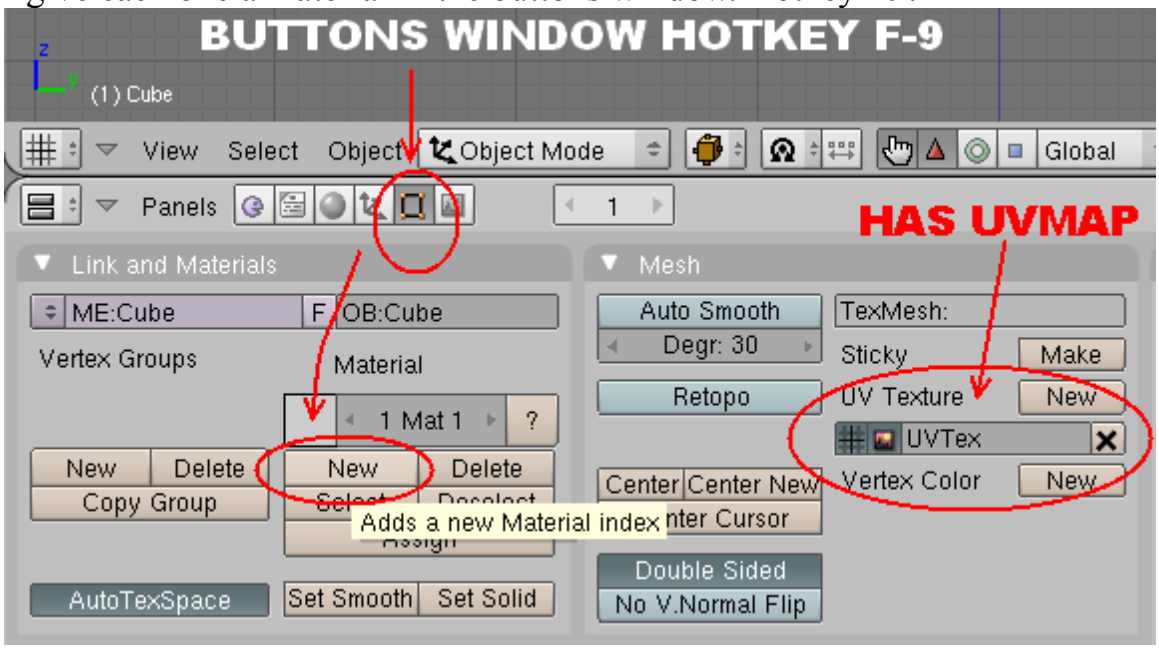
This tutorial assumes you know how to add and delete objects. Are familiar where Blender's UI. Have used lamps before. Have made materials for The Movies game before.

With Blender there is a million different ways to do anything. The method given here is just one suggestion and not an official method. Just a fast way to make a light map.

1. Import or make your set.

This will mean that your set has uvmaps already. And the uvmap for each object is projected over it's own texture images. Like wall objects are projected over the wall image. After all objects have one uvmap projected over the texture image, I gave each object a material.

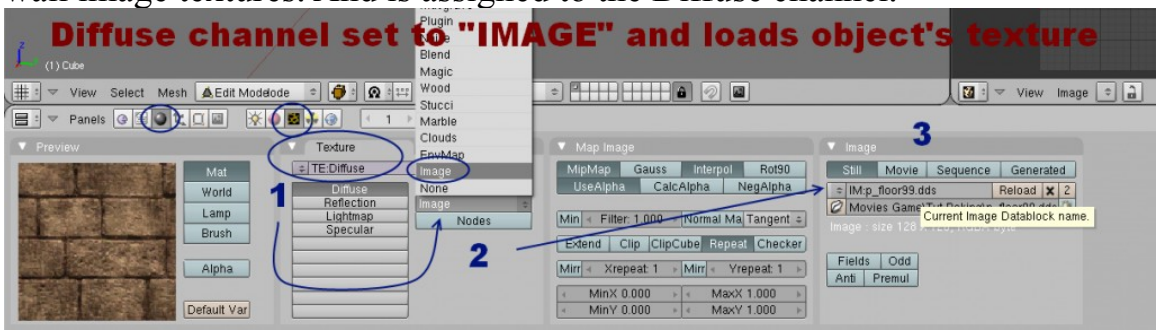
I give each one a material in the buttons window. Hotkey **F9**.



Then went into Materials buttons window, Hotkey **F6** and entered 4 names in the stack of channels.

Diffuse
Reflection
Lightmap
Specular
Keep that order. You enter the names by clicking on the line square above the stacked channels.

The *Diffuse* channel is the one using the image texture. The wall objects use wall image textures. And is assigned to the Diffuse channel.

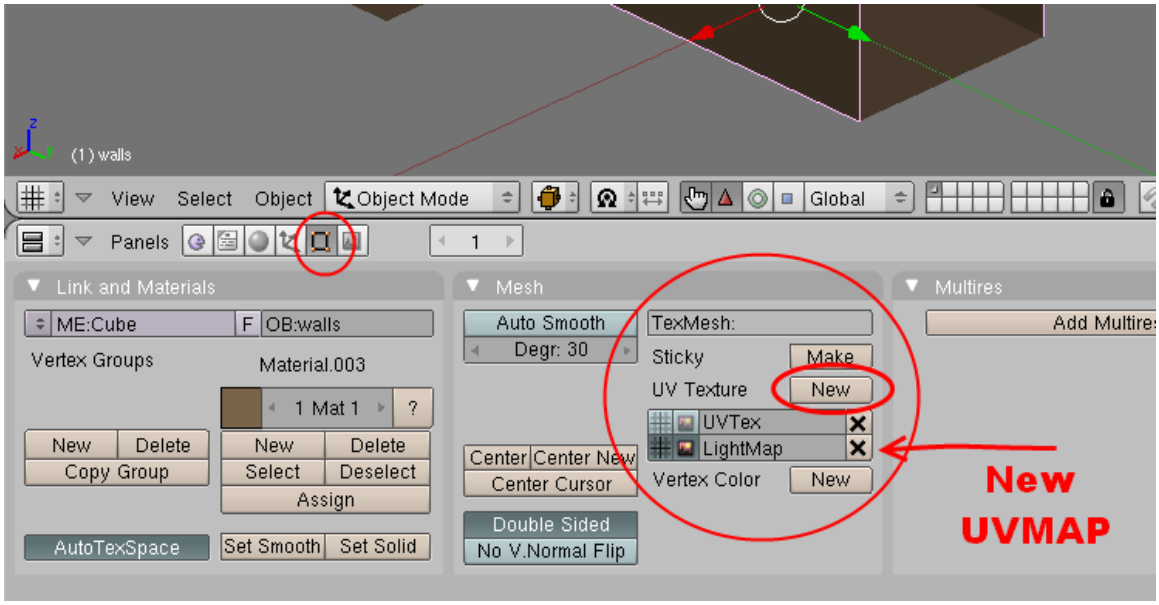


Besides materials, uvmaps and textures, sets also need extra layers to be completely functional in the Movies game. **zHeight** is an object found in another layer. Actors will stay on it when walking across the set. If you have stairs on the set, the extra layer zheight object will also have higher ground the actor is actually standing on. zHeight and doors and all the studio lot functionality requires a set's sister file that ends with the **.INF** extension. This extra file controls in some way the other lightmap image a set uses. If you want to see this, pick a set and replace it's .inf file with another one. This secondary lightmap will change lighting appearances on the studio lot.

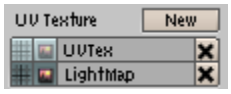
Important note: Your custom set should use the layout of an existing Movies game set. All scenes are tailor made for the set they appear on. Also sets come with lots of goodies that you can incorporate into your own set. Sliding doors, elevators, water objects, all of which require the extra info file (.INF) to function. You can make sets that do not include the extra info file, and you also can download many of these from 8eyedbaby. But a proper set needs one. Also you will be able to move or remove the set from the studio lot.

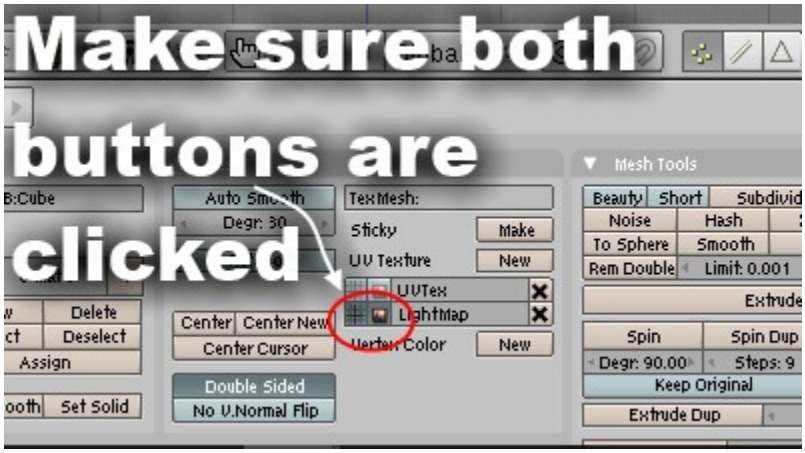
2. Give each object another **uvmap**.

Create another uvmaps with the buttons window. Hotkey **F9.**
Button is called **NEW.**



Title the new uvmap **LightMap**. Capital L and M. No spaces. Click the name to change it. The buttons to the left of the names activate the use of the uvmap. For both rendering and to view in 3D space. Make sure those buttons are clicked.

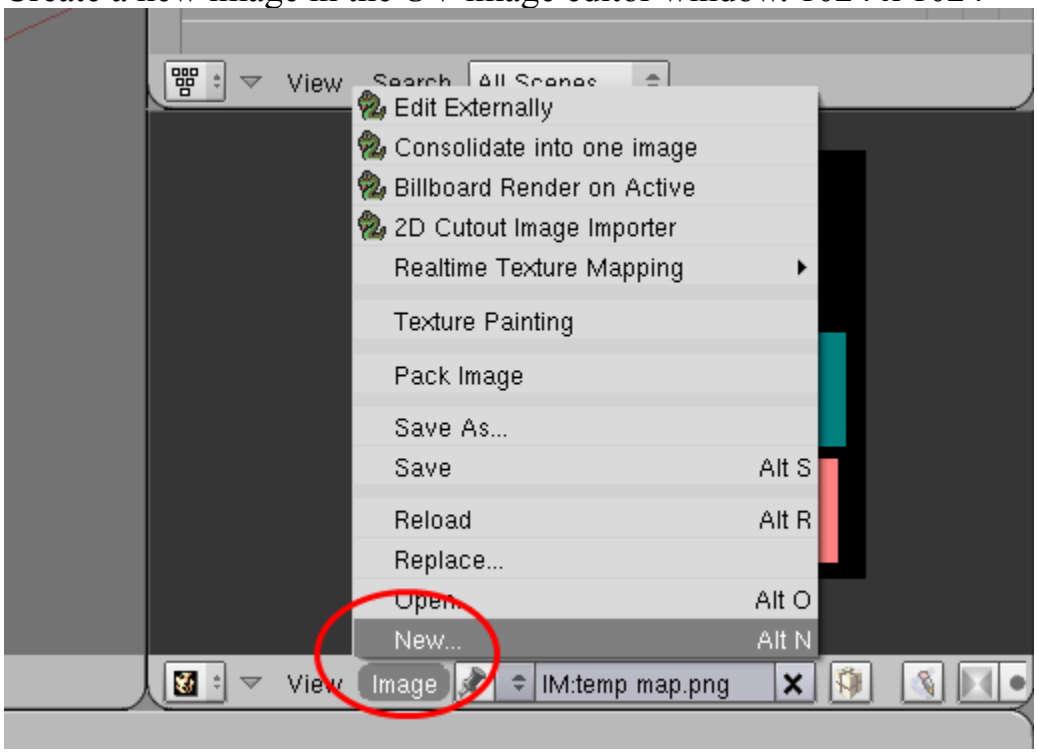


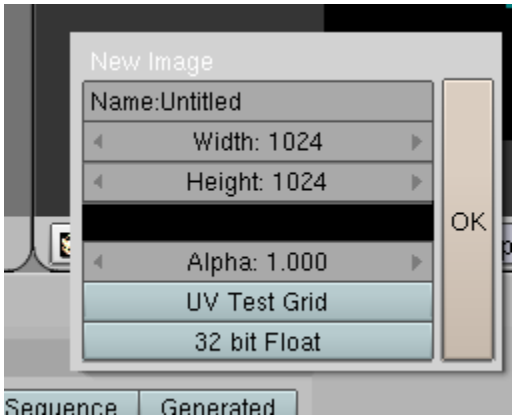


3. Create a new image and place new ‘LightMap’ uvmap over it in UV Edit window.

Since every object is going to share the same image we need a way to keep track of where all objects new uvmap is on the new image. We will bake every object one at a time onto the new image. Each time the image will have it’s appearance change. Each object adds it’s own position to the new

image. We need a way to keep track of each object's placement on the image so I bake it as I go along.
New uvmaps are copies of the first one until you change them. Meaning they will be the same lines projected over the same image. But soon all object's second uvmaps, the ones titled "LightMap", will all share the same image.
Create a new image in the UV image editor window. 1024 x 1024





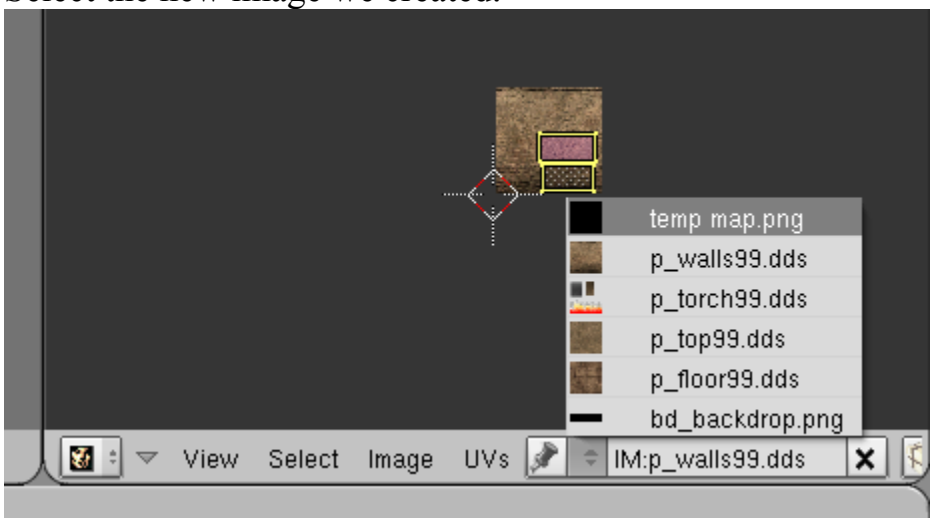
Keep in mind that since every object must share this image it will need it's own space. So this image should be large enough to fit every object in. 1024x1024 should do it. Once the blank image has been created go ahead and save it as a (.PNG). Something like **Temp map of UVs.png**.
Next we will place each object's secondary uvmap, the one titled **LightMap** over the new image. But as we do so we will bake it so we can keep track of

where. After each bake, save the image over the old one.

Side Note: In Blender 2.7 it is possible to have every object project a secondary uvmap all at once. And each face will have its own place on the image. Nice trick that saves a ton of time and can avoid most of the following steps. What I would do in that case is save the project as a separate .blend file called **Set_Uvproject_inBlender2.7.blend**. In this .blend file I would make sure that every object already has two uvmaps. Also would make sure that the maps are active. Then load the file in Blender2.7. Use the space bar menu to select the light map projection tool. I forget the name. It's been awhile since I've used Blender so you may have to research this yourself. It's certainly doable and a time saver. After I would save it as another .blend file, something like **Set_After2.7uvprojection.blend**. And be sure to click the **Legacy** flag when saving or it won't load back into Blender2.49.

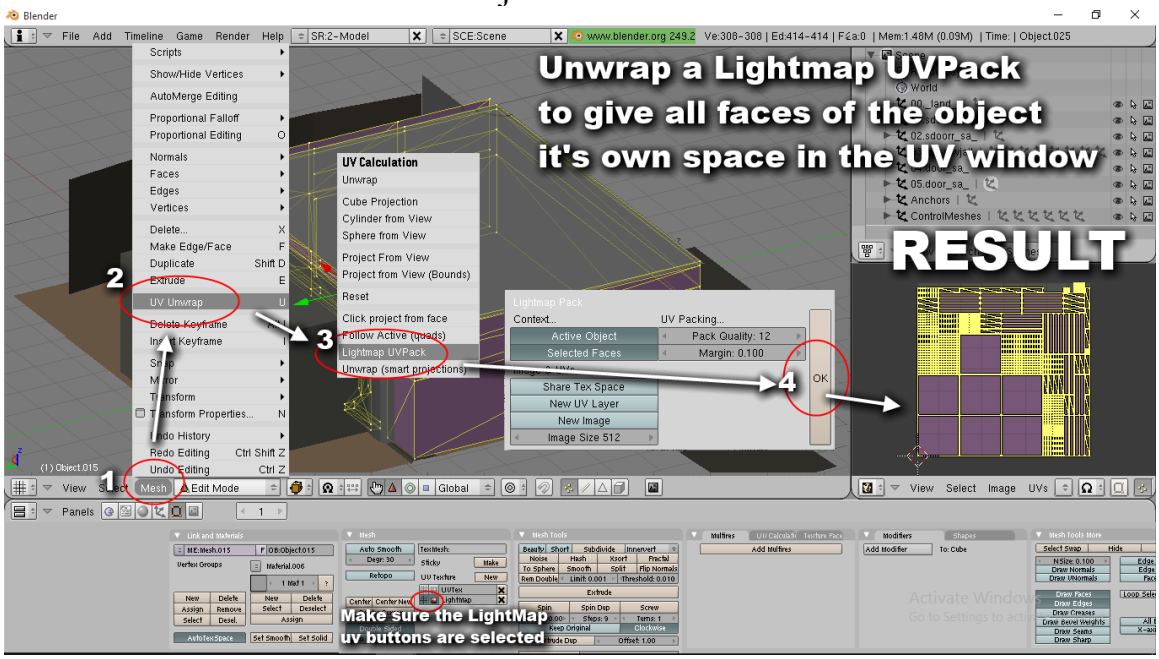
The first object, once selected, needs it's second uvmap projected over this newly created image. We need to change which image it sits on in the UV window and what the new Uvmap should look like.

When changing which picture the 'LightMap' uvmap sits on make sure all of the verts of that object are selected in *Edit Mode*. Next make sure all of the map is selected in the UV window. Now change the picture file by clicking on the little arrows next to the image name. This brings a list to choose from. Select the new image we created.



All verts must be selected at once when assigning a new image or the new assignment won't take.

Now we must remake the ‘LightMap’ uvmap. Each quad or tri should use it's own space on the image. You can custom project them however you wish. The following pic is an example of making sure each face of the UVMap has it’s on space. You don’t have to do it this way but it gets the job done. It is called UV Lightmap Unwrap. Designed specifically for this task. It would be ideal if it did this for all of the objects at once but it does not.



After I use this type of unwrapping, I then scale all of the map for that object down to just a corner of the image. Because each object will need it’s own space on the image, while this unwrapping used the whole image.

4. Bake the **Normals** onto the new image.

There must be a way to keep track of where every new uvmap is projected over this new image. There is. We can bake each object's Normals into it as we go along. To make a general map that keeps track of where each object's uvmap is projected, I will bake each on the same dummy image, one at a time.

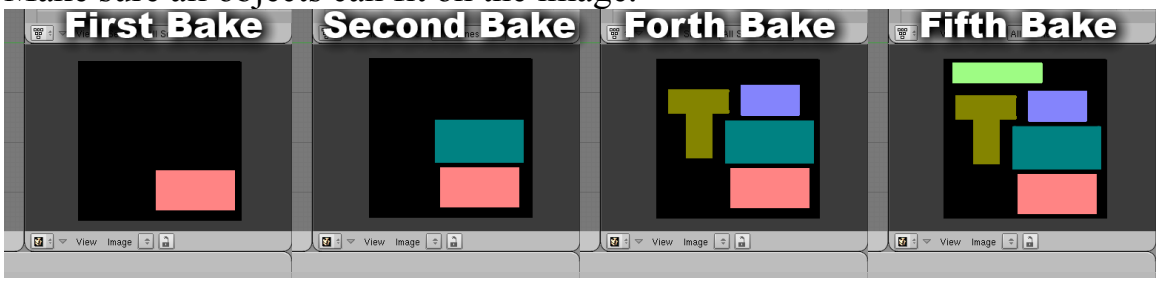
During this baking I will unclick or deactivate the **CLEAR** button in *Baking* buttons window. Hotkey **F10**.

Then make sure the **NORMALS** button is selected.

Once the **Clear** button is deactivated and the **Normals** button is selected, click on one of the objects, either in 3D space or in the Outliner and hit the **BAKE** button.

And after each bake I will save the image. Each time updating the image. I save it because Blender will only keep each bake for a certain amount of time. Blender may reset or you may take a break and finish the project later. So you need the image to be saved. The image will become blank if you do not save it.

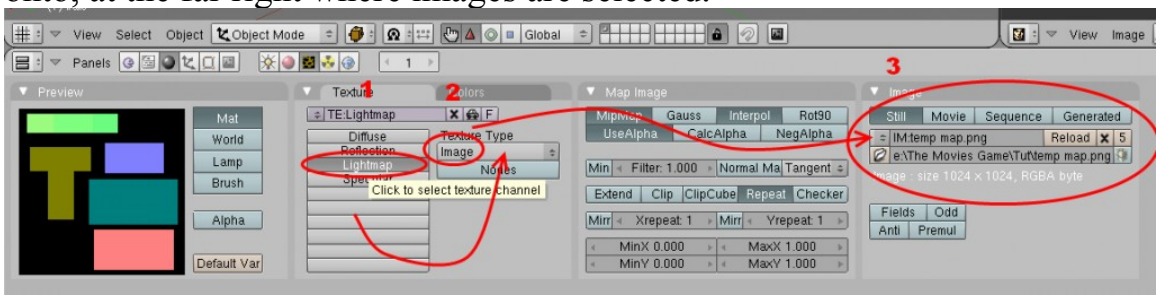
The **Clear** button would clear the whole image and only bake the object presently selected. Since this is a map of all objects we need to keep the last bake and the button un-clicked.
Make sure all objects can fit on the image.



In the uvmap image editor window you will see the normals of the present object baked onto the image. I just go down the list in the outliner window and bake each object one at a time.

Keep in mind… Make sure it has another UVMAP titled "LightMap".
Use the buttons next to it to make sure that map is the active one.
Re-project the new uvmap over the new dummy image we are baking on.
Customize the new uvmap so that it is a map of light and not for the texture image.
Shrink the uvmap in the image editor window, moving it to it's own space on the image. Also making sure you leave room for the rest of the objects.
Bake the *normals* of the object onto the image.
Save the image over the old save to update the new baking.
Then turn off the object's visibility just to show that it is done.
Repeat these steps on all the objects of your set.

Once every object has it's second uvmap (LightMap), baked onto the new dummy image, your set is nearly done. All that is left is making sure each of the objects in materials buttons window, **Hotkey F6**, has the lightmap channel set to **IMAGE**. Then select the dummy image we baked each object onto, at the far right where images are selected.



Why set it to the dummy image?

We don't have to but the set will need to have this channel set to an image on export. In MED (The Movies Game Editor) it can be exchanged for the actual image of the lightmap.

This dummy image can be replaced with the actual light map image later in MED, or in another Blender session. If you can not use MED, then use another Blender session to assign each object's material Lightmap channel to the actual lightmap image we will bake shortly. Otherwise assign the .png and let MED change it later.

Assuming that every other portion of your set is finished, you could go ahead and export it to the game. The following steps will be exclusively for making the light map image. You can export the set now because every object has another UVMap and was projected over the new image (albeit a dummy image). And each material has the *lightmap* image assignment. Now that these steps are finished save your project as a .blend file… or update it now. But we are going to make another .blend file soon. But be sure to make a backup of your set now while it is in a finished state.

5. Save project as another .blend file.

Now we are going to do something radical. We need another copy of your set. We need a copy of the set that will not be exported and used in the game. This copy we can do anything to and have no worries about the result. Go ahead and save another **.blend** file and title it something like: "Baking set

temp.blend". Since each object has a new uvmap projected over a new image and the materials are updated to know there is a new map, that part of the set is done. The only thing we need to do now is make the lightmap image. We only made a dummy image before. For the actual image baking we need this new .blend file.

All baking from here on is NOT done with your actual set's .blend file!!! It will be done with this new one. "Baking set temp.blend".

Now that we are working with a new .blend file, lets make some radical changes. I want to bake the entire set's lightmap at once. We could use your original set's .blend file but we would have to bake each object one at a time. A set with a hundred or more objects would take a long time to bake. And we don't have to. It can all be done at once.

6. Join all the objects into one Object.

To bake the entire thing at once, all objects must be joined together as one object. But wait. In this example, I made a corridor that has a torch hanging on a wall. The object for the flame will never need a lightmap because it will always stay on and never get darker. Also a fire object can be set to "Self Lit" meaning no lightmap will effect it. So I'll not join the torch object to the whole.

To join every object each object must be selected at once. But one object needs to be selected last so that Blender will know which object is the one the rest is being joined to. That's how Blender does it. Instead of using Hotkey **A** for selecting all. Select each object one at a time while holding down **SHIFT** key. Holding down **shift** key will allow more then one object to become selected at the same time. Once they are all selected hit Hotkey **CTRL + J** for *Join* in the 3D window.
Note: If you turn off *selectable*  of one object, then press hotkey **A**, all objects but one will become selected. Then turn on *selectable* of the first object and use **Hold Shift + Click** on that object, then Blender will know that all objects will join to it. A time saver when there are a hundred objects or more on your set.
All these objects are now just one object. It also inherited every object's *materials*. So delete all of the materials. Now give it just one new *material.*
Note: Blender has some memory issues. You could sometimes keep the last

material and not assign a new one. Yet often sets have trouble baking or rendering in Blender when the materials are imported or even just old. Sometimes deleting a material and adding a new one will fix a render issue. I know, kooky but what are you going to do?

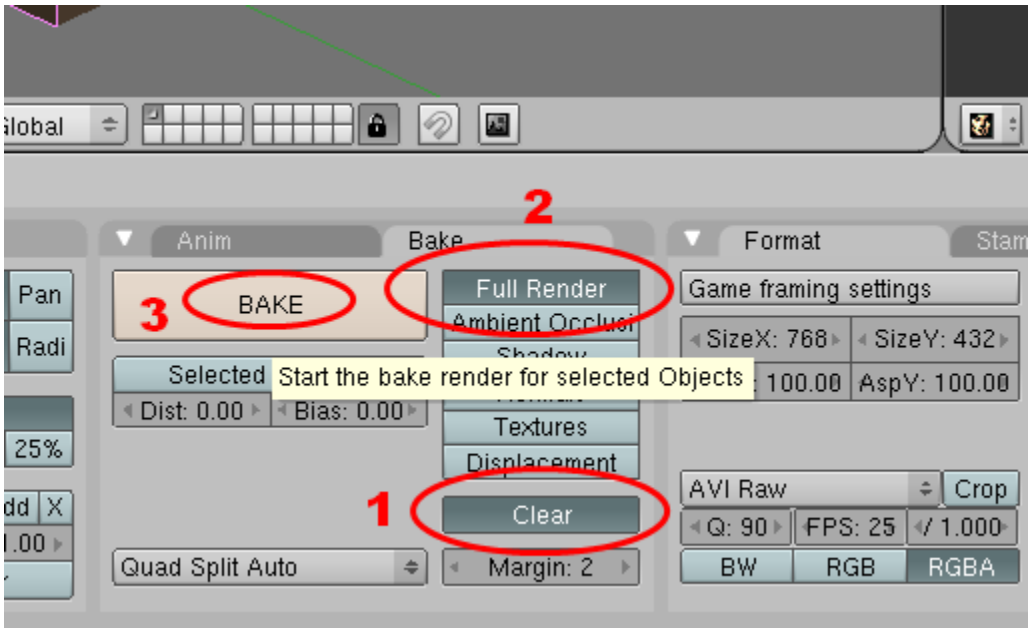
7. Add a lamp.

Since my example has a torch object, I will assume the set's light originates from it. So the lightmap I will bake will reflect this… that the torch is the source of light in our movie scene. To do this I will place a Blender *lamp* right where the torch fire is. I will select the fire object. And place the red cursor circle right where that fire is. Wherever we place the red cursor circle in *3D* window is where new objects will be added.

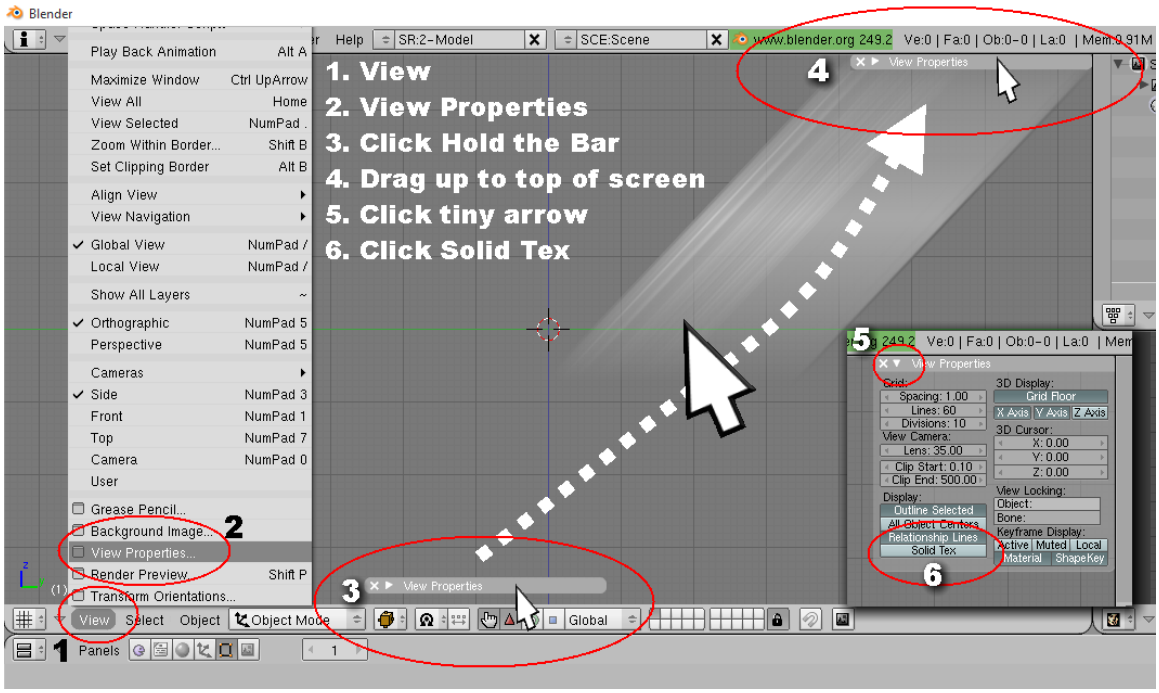
Go to the top and click the **ADD** button. Now select **LAMP**. And choose a regular lamp. One lamp gets added to the scene.

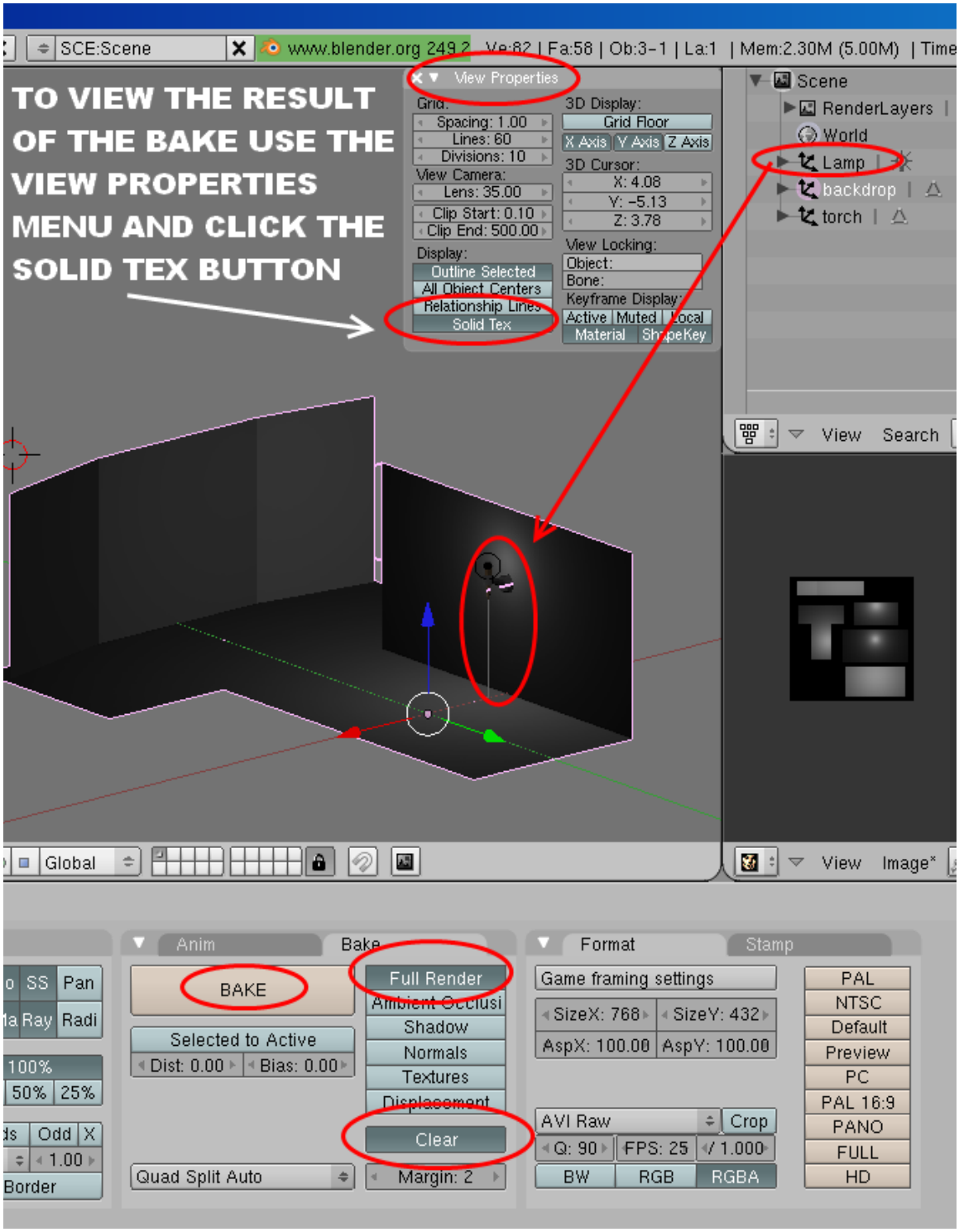
8. Bake the Light Map.

Now click on the set object. In the Baking buttons window below, Hotkey **F10**, activate the **Clear** button. Now click from the top of the list, **Full Render**. And hit the **BAKE** button.



You can see the result of the bake by using the *View Properties* window.


Blender
Play Back Animation Alt A
Maximize Window Ctrl UpArrow
View All Home
View Selected NumPad .
Zoom Within Border... Shift B
Set Clipping Border Alt B
Align View
View Navigation
Global View NumPad /
Local View NumPad /
Show All Layers ~
Orthographic NumPad 5
Perspective NumPad 5
Cameras
Side NumPad 3
Front NumPad 1
Top NumPad 7
Camera NumPad 0
User
Grease Pencil...
Background Image...
View Properties...
Render Preview... Shift P
Transform Orientations...
1. View
2. View Properties
3. Click Hold the Bar
4. Drag up to top of screen
5. Click tiny arrow
6. Click Solid Tex
1
2
3
4
5
6
View Properties
Spacing: 1.00
Lines: 60
Divisions: 10
Lens: 35.00
Clip Start: 0.10
Clip End: 500.00
Outline Selected
All Object Centers
Relationship Lines
Solid Tex
Grid Floor
X Axis
Y Axis
Z Axis
Object:
Bone:
Active
Muted
Local
Material
ShapeKey
View
Select
Object
Object Mode
Global
Panels



You can adjust the lamp's settings if you wish. It all depends what your set's needs are.

Once the image is baked, save the image as a new name, something like "Bake1.png"

Blender does not save .DDS image formats which is what lightmap images need to be. The images we bake will need to be converted to .DDS in an image editor like Photoshop. Or Paint.Net.

You can move lamps around, add more or change the color of the lamps as you wish.

And then make a new bake. As many as you want.

The game allows each set to have up to 8. The first four are for which direction the sun is to your set when it is placed on the studio lot. The next 4 are used when filming movies.

A night-time lightmap can be accomplished in Blender by setting the lamp to a bluish color. You could also decrease the power of the lamp. Or for the dawn texture, make it reddish. You can also do these same changes or colors with a paint program.

Most sets have a day time lightmap. A night time light map. And a dawn or sun set light map. Some have an additional. And you can make all of these in Blender.

Once you have several images saved you're done with Blender.

9. Convert the saved Baked images to .DDS format with a paint program

Open these images with a paint program and re-save them into the Movies game folders with proper names and in .DDS format.

Important! The Movies has two kinds of lightmap images. One used with the 3D model of your set, and the other to be cast upon objects and characters placed on your set during a movie.

I'm not going to make additional lightmaps for this throw away corridor set. If, like in this example, you only make or use one, then make 5 versions of it.
Title them lm_(name of set)_0.dds to lm_(name of set)_4.dds

Zero being the first map or four being the 5th map.
lm_(name of set)_0.dds
lm_(name of set)_1.dds
lm_(name of set)_2.dds
lm_(name of set)_3.dds
lm_(name of set)_4.dds

Now the secondary lightmaps are titled lf0_(name of set).dds to lf7_(name of set).dds. The same number as the first lightmap images. Since my example will use five, I will also make 5 of these other lightmap images. I generally will use a solid color on these. And make them 32x32.

lf0_(custom set).dds

lf0_(name of set).dds
lf1_(name of set).dds
lf2_(name of set).dds
lf3_(name of set).dds
lf4_(name of set).dds

Naming…

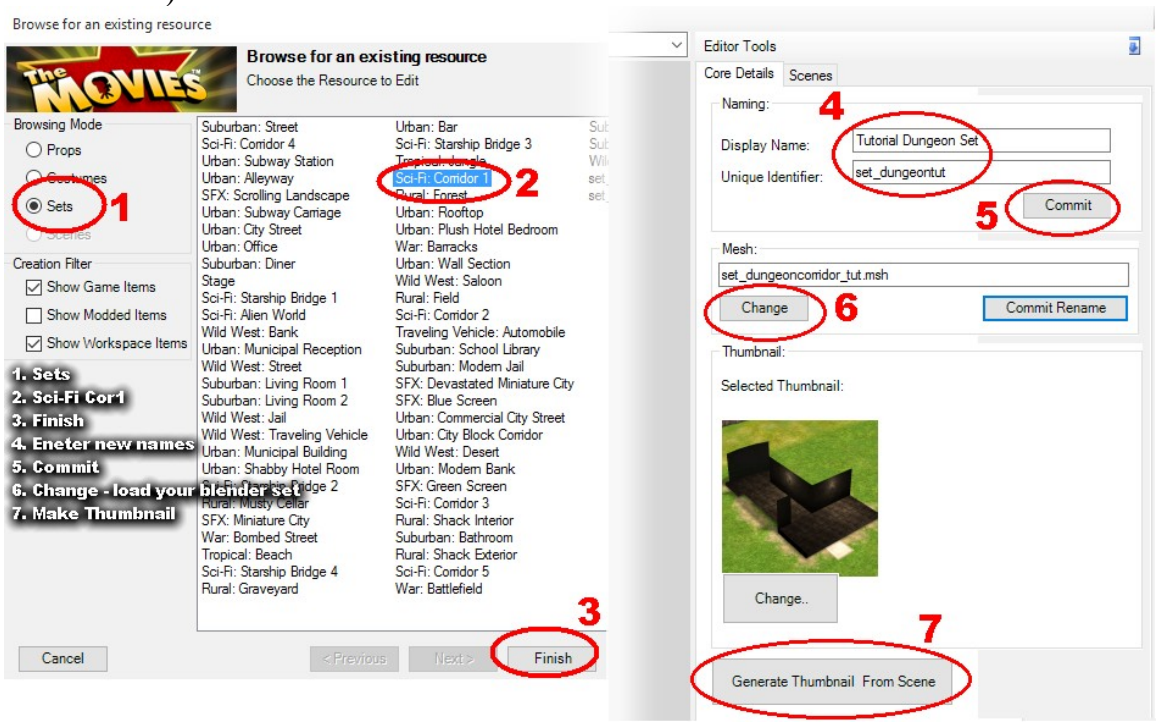
I seek to keep all the names uniform with sets.
If my set is called **set_dungeontut.msh,**
Then the first lightmap image will be titled **lm_dungeontut_0.dds**
The second lightmap image will be titled **lf0_dungeontut.dds**

All of these files go into your games data\textures\lightmap folder.
C:\Program Files\Lionhead Studios Ltd\The Movies\Data\Textures\lightmap
After you have your lightmaps ready and after your set is exported from Blender, you can now have the proper lightmap image assigned to the set.

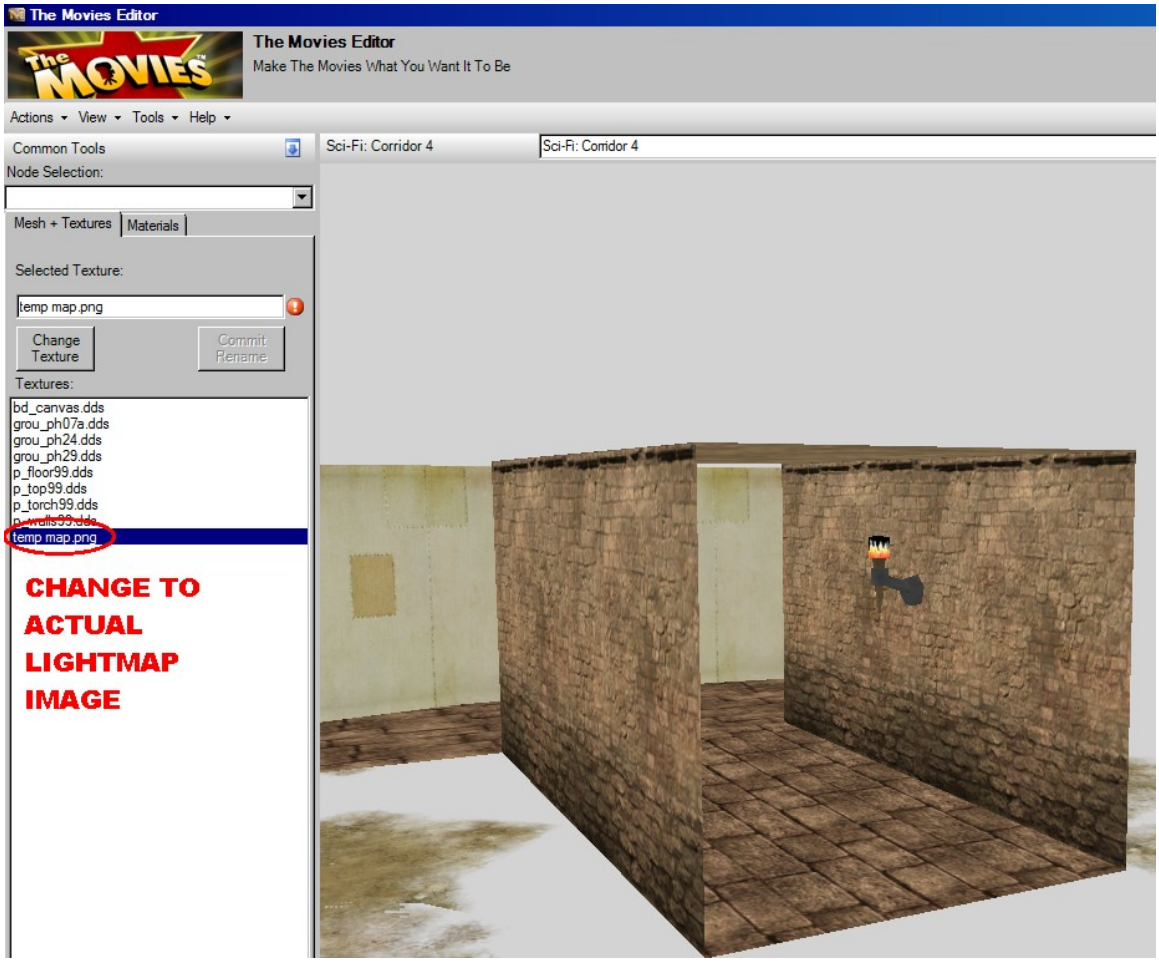
Use MED (The Movies Game Editor). Once Med is ready for orders, select "Modify / Create From Existing Content".

Select "Sets" at top left. The list of the game's sets is in the large window. Select one that matches your new set the most. (Always be sure to match your set to the layout of existing Movies game sets because the scenes are tailored to it. A set is almost worthless without the .INF file and following an existing Movies game set is the best way to make your custom set

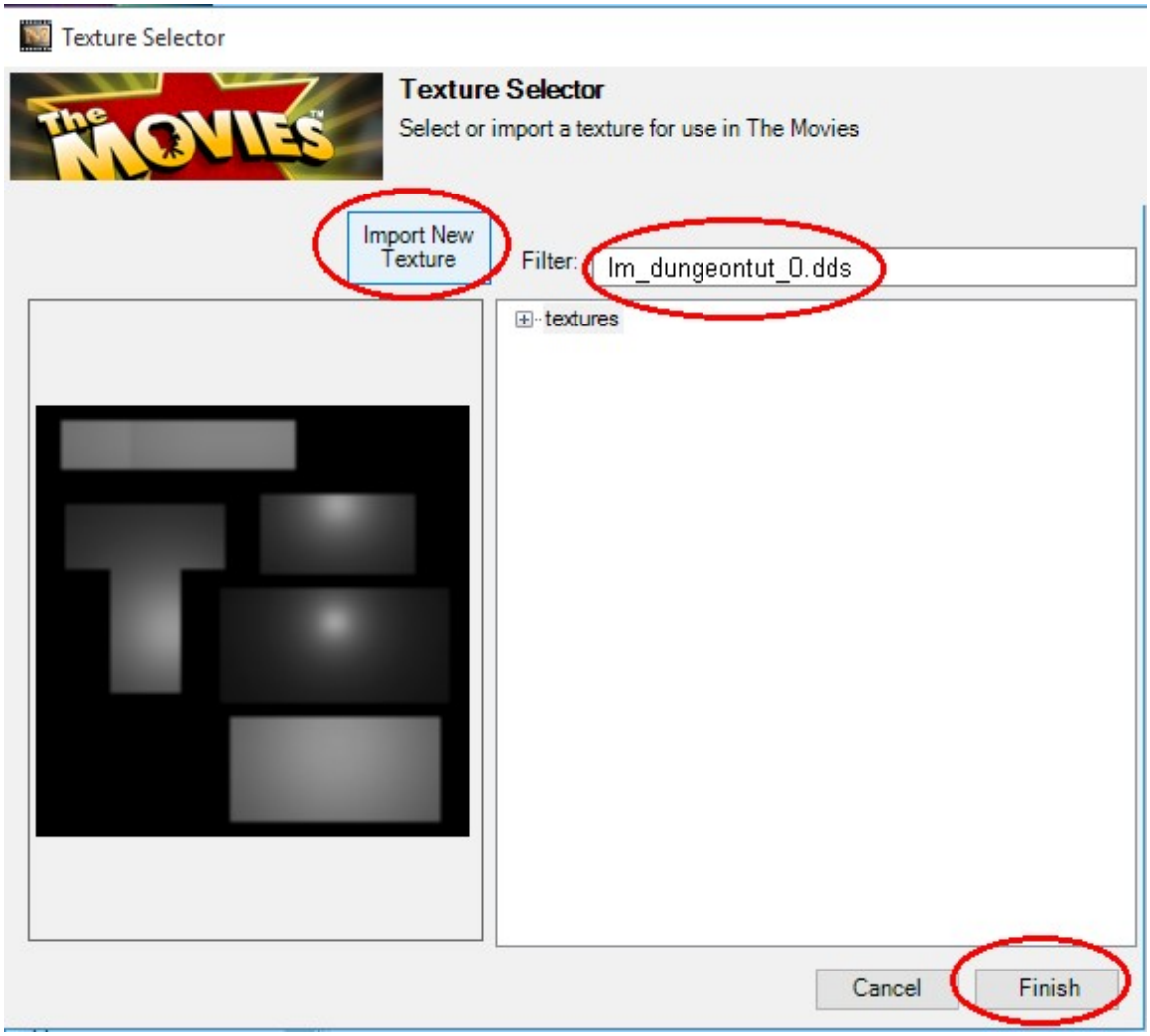
functional.)



In my example I will select the Sci-Fi corridor. Select a game set. Now it gets loaded into MED. At the right is a button for changing the .MSH object for the set. Click the **Change** button. From the browser navigate to your new set and select it. Make a Thumbnail.

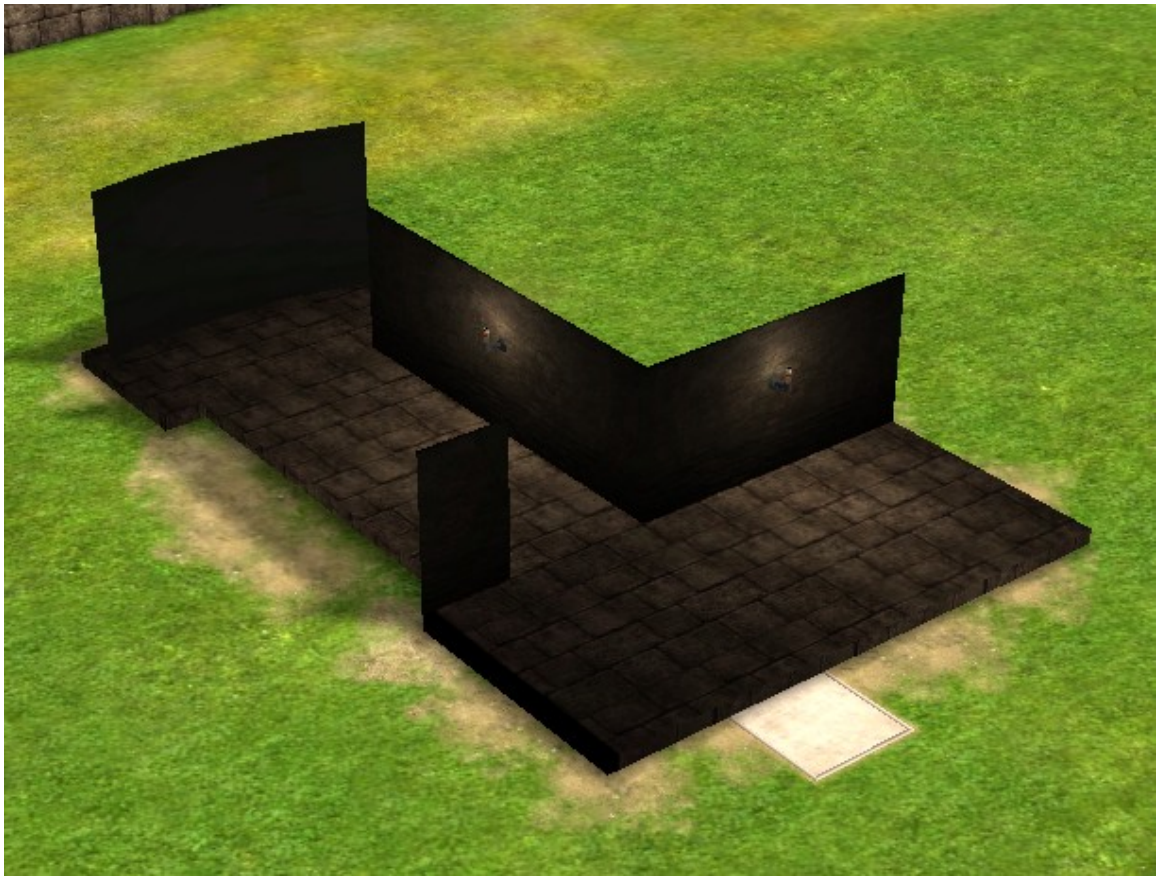


The dummy .PNG image was loaded with it. For mine it is still in the mesh file so I will change it to the new lightmap image I made out of my bakes.



Next change the set identity name from the sci-fi corridor to your new one. For me it will be: set_dungeontut

And here is how it looks.

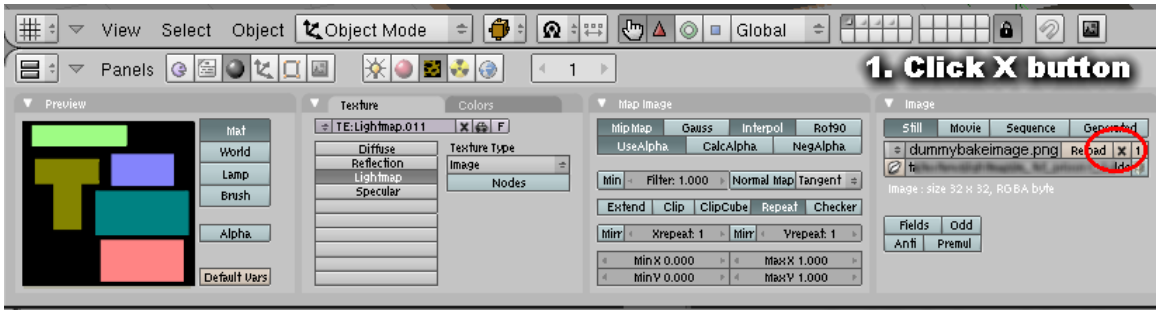


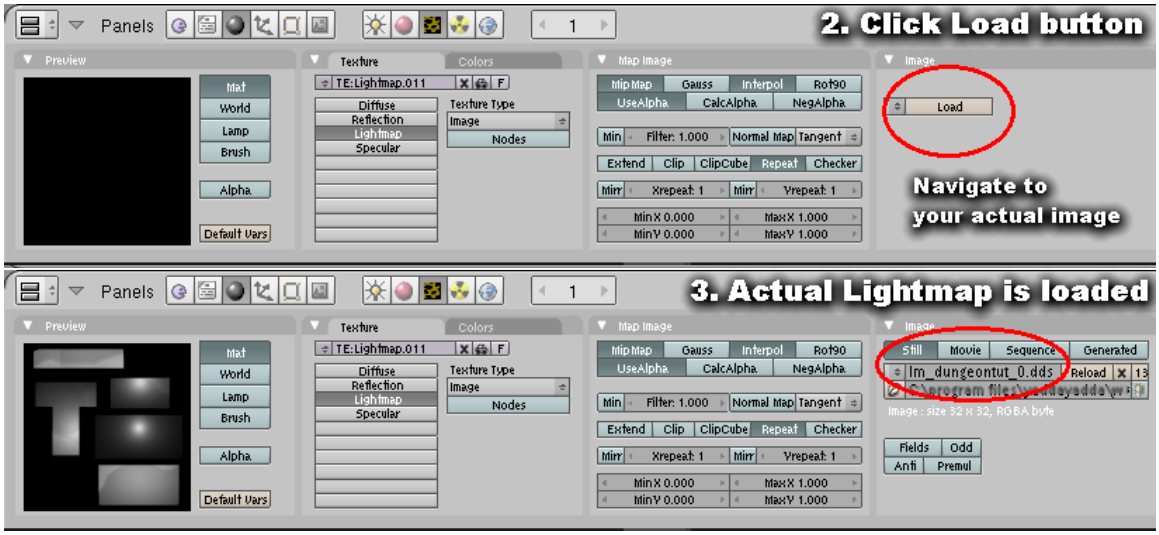
Done. But you're not done if you can't use MED.

The Long Way…

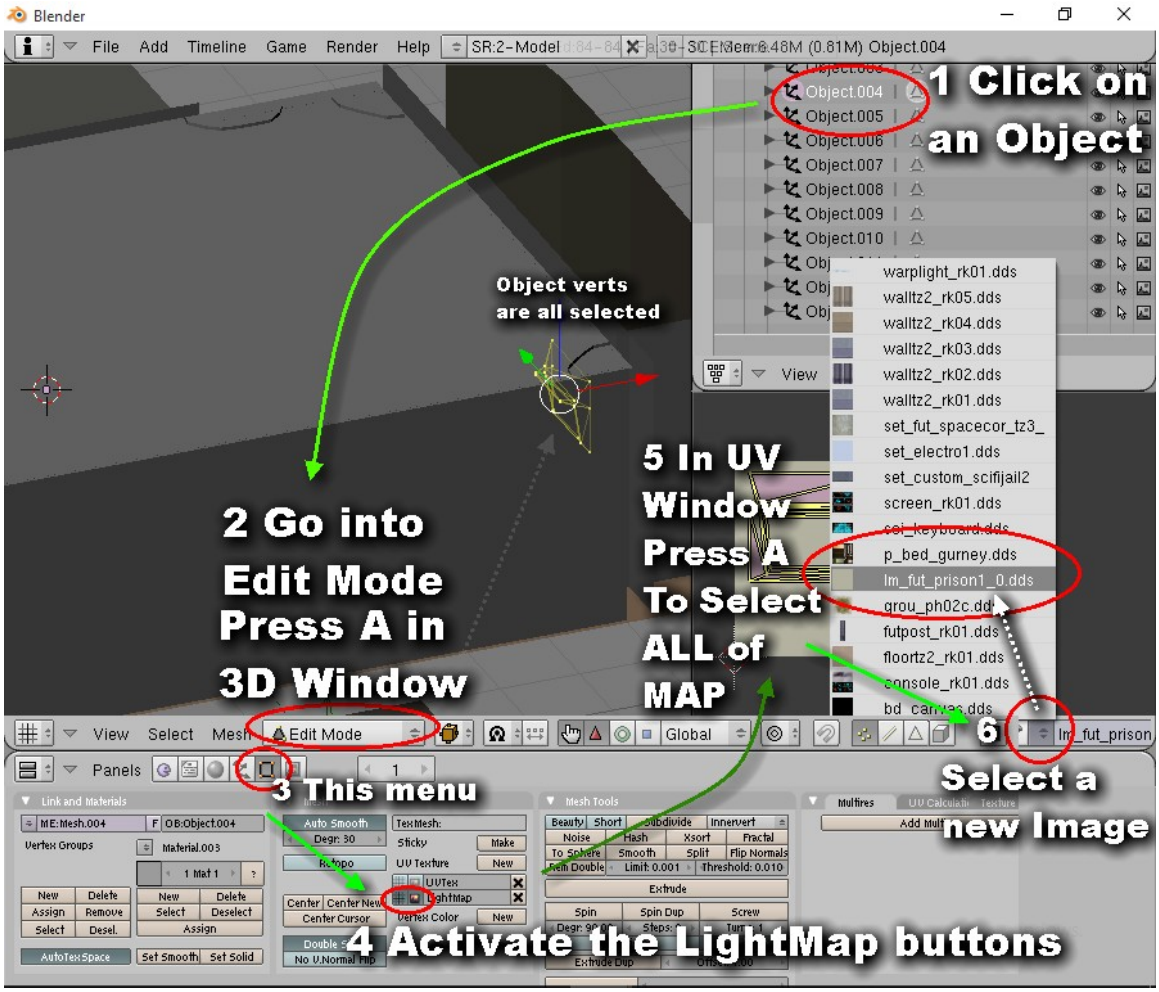
If you can't use MED for this step then use Blender.
Open your original .blend file, the one you used to export your finished set. And one at a time click on each object and change it's light map block to the new image. Also change the image in the UV window for the secondary uvmap.

1. Open the original .blend file you used to export your set model.
2. Click on first set object in *Outliner* window.
3. Go down to texture *buttons* **F6** window





4. Change which Image appears under the LightMap UV map in *UV* window.



5. Repeat the steps for every set object that needs the lightmap image changed.

6. Save/update the .blend file and export your set .MSH.

The third option is to use Reacher’s MeshManip.

Sometimes your set will have more then one light source, like two lamps. Or maybe their is a window and the outside may have a different luminance then inside. When this happens you have to join the objects differently. For more then one light source, just place more then one lamp. And adjust one or the other lamp's power if need be. But for two different states of light, you will have to join the objects differently.

For example let's take a house with a window set. Direct noon sunlight for outside will be brighter then the day light inside the house. Also for the night time map, outside may be dark, but inside will have lights on. And then you would want to make more then one nighttime map, one with lights on and one with lights off. Or one with a porch light on and one with a porch light off. Once I made a fireplace lightmap. For such a case baking may be more intricate. However one could still make it easy by joining the objects in separate instances. You could carve up the joined object into two different objects. One object is outside the house, including the yard, the walls on the outside, even the window shutters. And bake that part of the lightmap separately then all of the objects found inside the house.

You could also set up an intricate Blender environment that will bake every detail you could think of. Sunrays cast through stained-glass windows or the glow from a nearby stove. There is even a script specifically designed for baking lightmaps but I've never used it. It's suppose to bake everything without having to save the project as a separate .blend file like I did. And without joining all of the objects. I think it was written for Blender 2.49 but forgot where it was or how to get it.

10. Using The Append Function

During a Blender session we can get items from a different **.blend** file to our current project. A very handy tool for making sets or auto-animated props. You could eventually have whole libraries of .blend files to **append** from. So I made a two part tutorial showing how handy it could be.

Part 1 is also a tutorial for making an auto animated prop. (Ones that move on the set). It isn't 'the' way to make them, because there are several, but it is 'one' way to make them. It will show how to use a human costume, give it a

head, and turn the whole thing into a prop.

Part 2 will discuss how to transfer items from a Movies set to a custom set.

Most mods are forced to utilize stuff already found in the game. This is true with a lot of games. And each game needs to know how to find stuff. Usually this is accomplished by having the correct files in the right folder path, and also by how the files are named. Giving our mods the proper names allows the game to call into existence the items we make. But found within the 3D models themselves are things that also must have the right names. But not always. An example would be an Auto Animated Prop. But for sets the names and the numbers found in the names must remain the same for them to function correctly. Sets not only have the content numbered in a call order, but also there is extra stuff found in other Blender layers that is too time consuming to make from scratch. And we don't have to. We can just append the extra layers to our new set. Or vice-verse, delete the contents of a Movies game set we don't need and append the custom set we made in another .blend file.

This tutorial will give two examples for appending. What is appending? It is loading into your present Blender session items we created in another .blend file. The tutorial will focus on two instances where it is important to keep the naming and numbering, and one that does not need those specifics.

Another reason to use the append function is that sometimes if we keep importing new Movies game stuff to our project, we may eventually get the dreaded python error. And the last import may be buggy, or might upset Blender's memory. Also a full import of any Movies game content also imports materials and groups, which may not be needed for something like an animated prop. We can always have them removed after import but there is a cleaner way to do this. By using the append function.

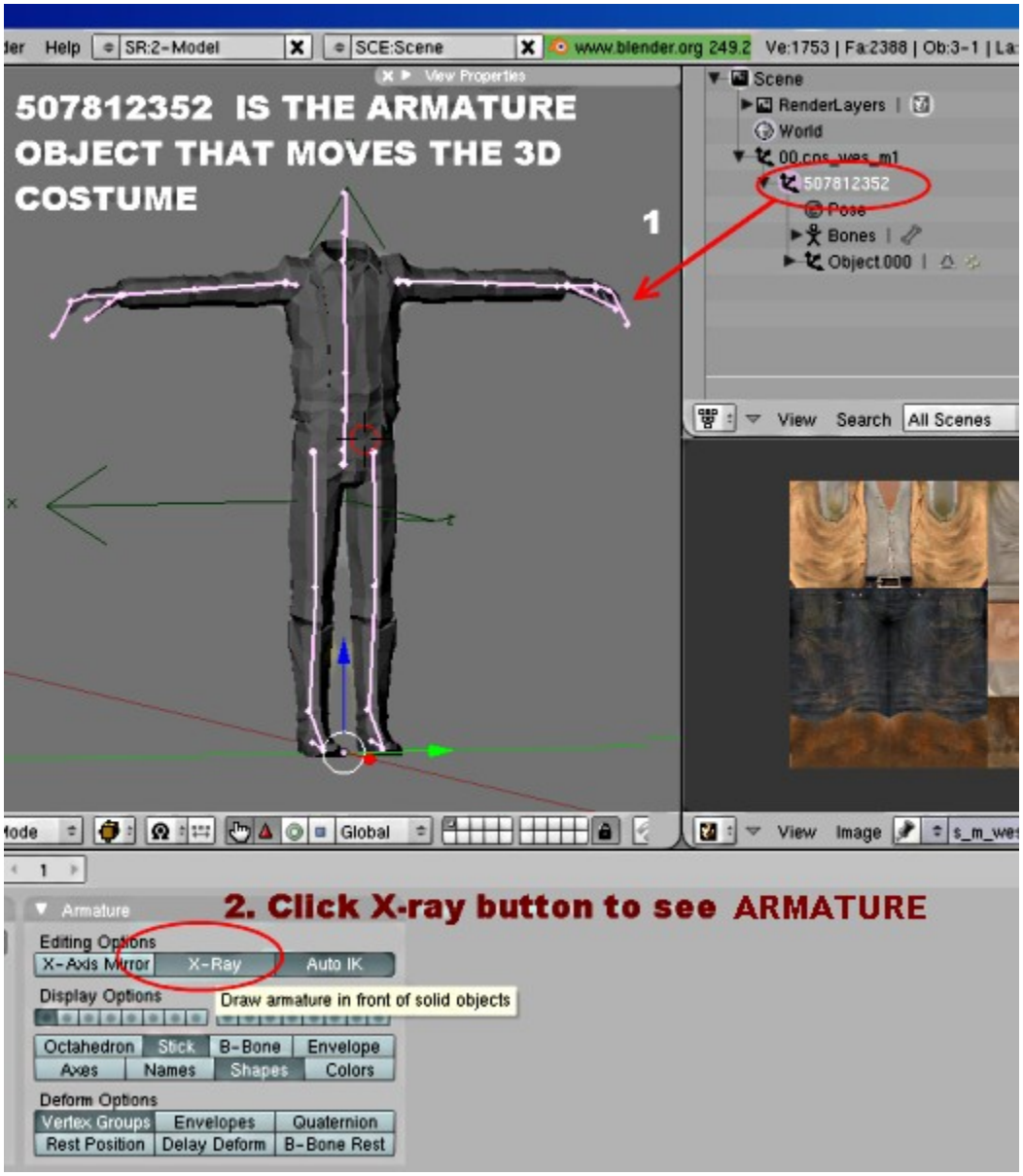
The first example will be creating an auto animated prop. Must props do not need specific namings within the model or within your Blender session. These are static props or auto-animated props. So long as the final result is exported to the game and saved as **p_(name of prop).msh**. But we can add lots of details to animated props using Blender.

There are few **auto-animated props** originally found in the game. The saddleless horse, the dog, the waving flag pole and a spinning UFO to name

a few. They can be placed on any set or sometimes on the studio lot. They will animate and move around just like actors do. The Movies game was designed to be open ended so that more content could be added, indefinitely if need be. Since TMO closed only we can create new auto-animated props.

What can be an auto-animated prop? Anything with an armature. Or rather anything requiring animation files. Scenes are a collection of animation files. Animation files move the armature or bones found within some models. If the model has no bones then there are no animation files for them. Ants have bones but these were not developed further by Lionhead. If they do have animation files no one has found any yet.

Let's look at a model that has bones or what we call an armature.



This is the first male cowboy costume: **cos_m_wes_1.msh**. The bones or armature is inside the object and is hard to see unless you click an X-Ray button found in the *Buttons* window, Hotkey **F9**. First in the *Outliner* window click on the name of the armature. The name is a number: **507812352**. Then press **F9** to get to the *Buttons* window. Then find the **X-ray** button and click it. The white lines are seen in 3D space. These move the costume around.

For human costumes, there are a lot of animations in the game. Use MED (The Movies Game Editor) to extract them. They would be found in the data\animations\high folder.

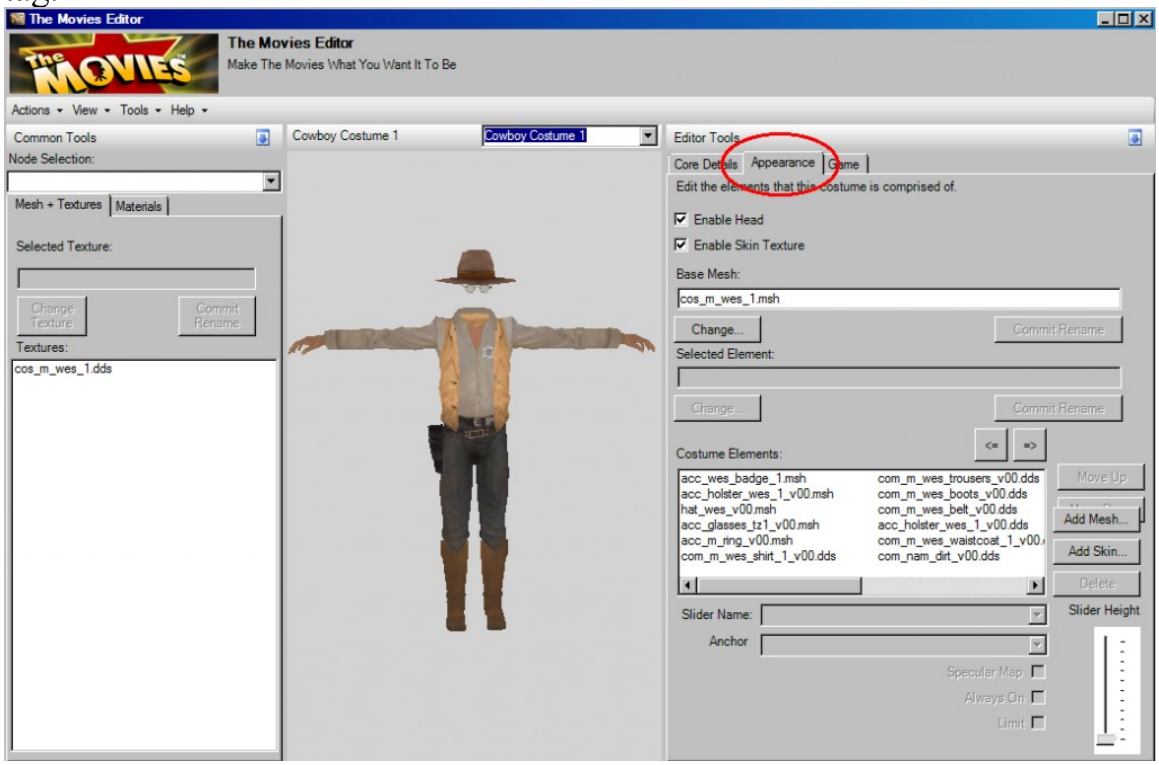
For this tutorial we would need to extract several .msh files to make an animated human costume. The game does come with a few heads that are

suitable for this tutorial, without facial weights. If you want to use a Starmaker head, you would have to remove the face weights from it. The game does come with heads that already do not have face weights. The head I will use, **generic_head.msh**, has no face weights but I am not sure if it only comes with the Addon Pack Stunts And Effects. I will provide a download link for all of these items if you do not want to use MED or can't.

(Link to Files pending)

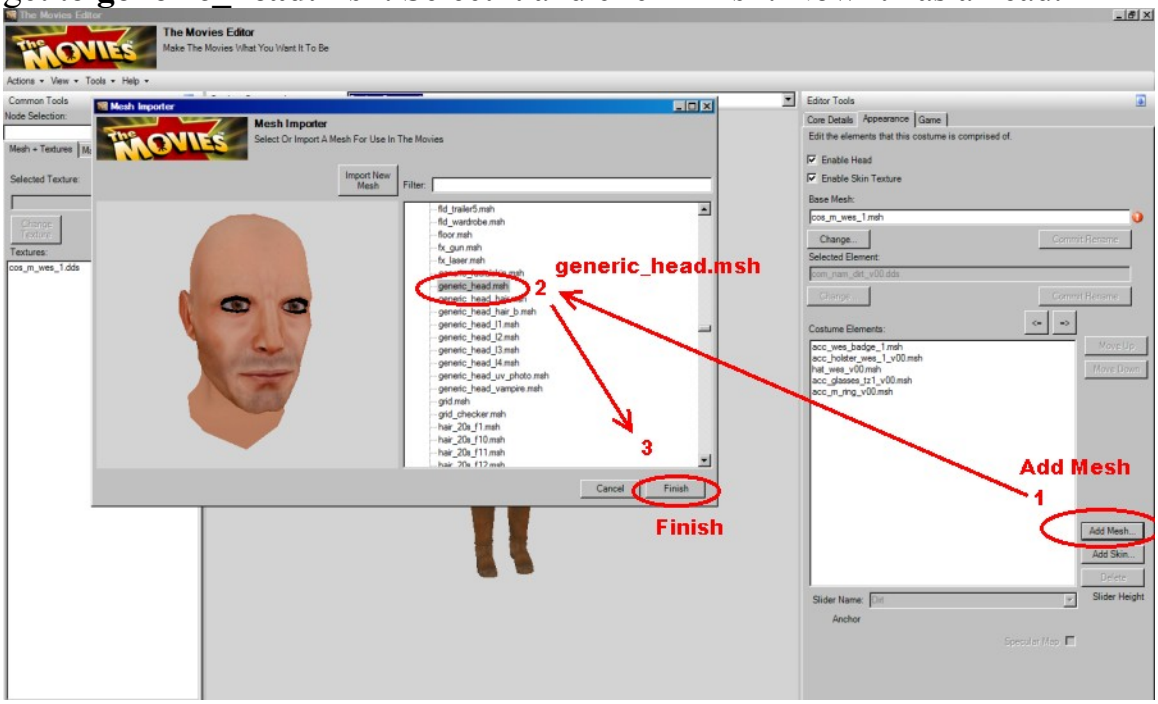
1. Extract the files we need with MED (The Movies Editor).

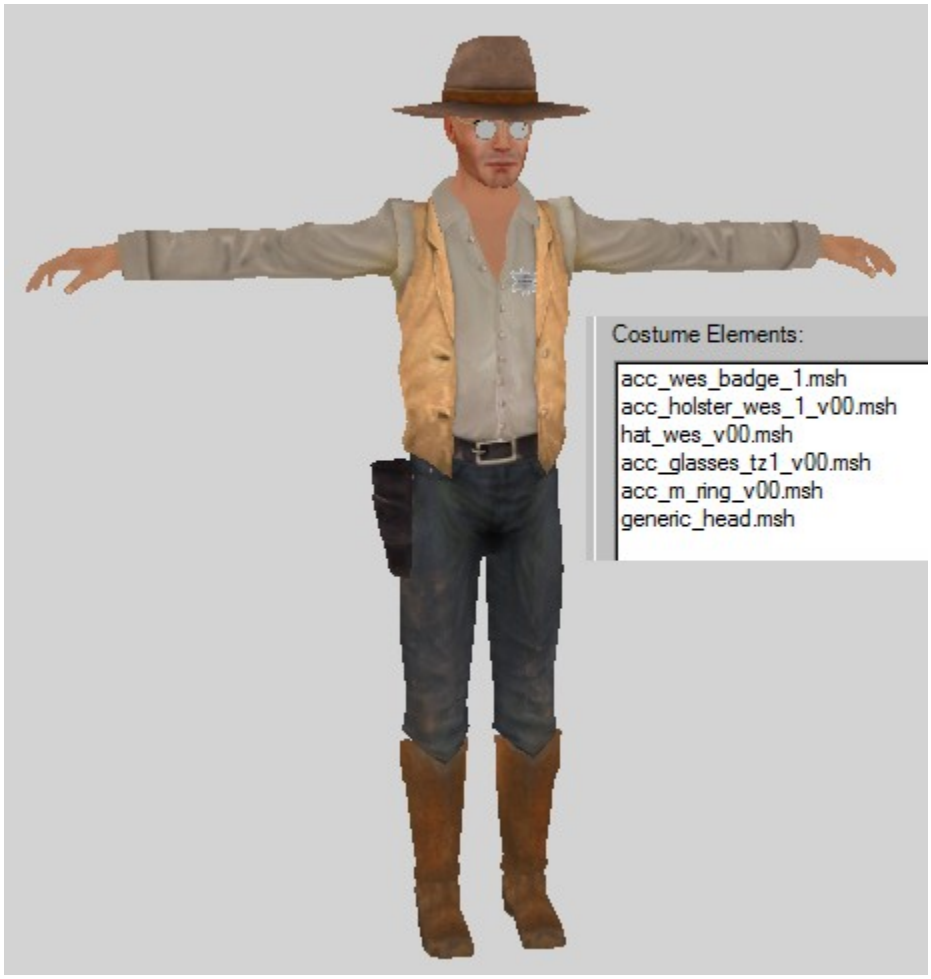
Open MED and select: **Modify (Create From) Existing Content**. Select **Costumes** at the left. Then below that select the **Western Category**. In the middle is the western costumes. Select **Cowboy costume 1**, then **ok.** We can see the model and it's accessories which we will use for the Auto-Animated props. Select the tab at the right called **Appearance.** In the box at the right is a list of the Image textures (color options) and the 3D mesh files that go with the costume. This is not a texture tutorial (though you could make your own custom images) so go ahead and delete all of the files ending with a .DDS tag.



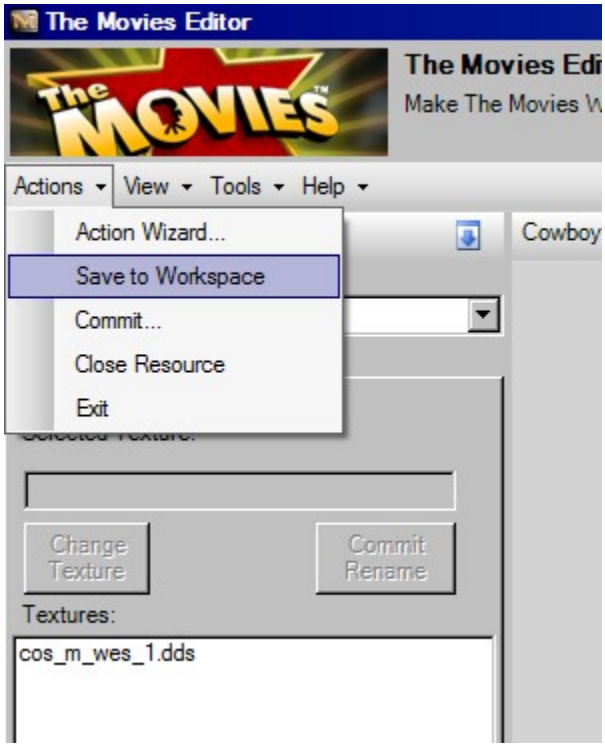
When only those files ending with .MSH are left then we're ready. Those are the 3D models. Some of these may not have weights so we will have to give them weights in Blender. Weights are added to 3D models so that the above Armature can move the object. Without weights those objects won't move.

This costume has no head so we will have to give it one that has weights. And the head I want to add to this may (Or may not) only come with the addon Stunts and Effects. Click the **Add Mesh** button. Scroll down until you get to **generic_head.msh**. Select it and click **finish**. Now it has a head.





Now save it to your workspace. This will export everything here to where MED places your projects files. It will be a collection of **.MSH** files and **.DDS** image files. Do not move any of these from the folders. Keep them there and we'll edit these using Blender. There is another folder we do not need. The data\costumes folder. This makes these items available as a costume and we are making an auto-animated prop. Delete the costume folder. Technically we do not need the textures folder. But let's keep it for now as they will get imported into Blender while we are working on them. They already are found in the game so when we are done, we will not put them into the game's folders. If you made your own custom textures then you would use them in place of these (more work then this tutorial will cover).



Close MED.

2. Import one object and save it as it's own .blend file.

Now that we have our 3D models we have to convert them to .blend files. Do this one at a time. Import the first object and save it as a .blend file. After saving it start a new Blender session and import the next MSH, save and start a new session, and so on until all msh files have been converted to .blend files. Title them by what kind of object they are. Example: **Badge.blend** or **Hat.blend** or **Costume.blend**. All separately. Once they are in a .blend file state, they can be appended from.

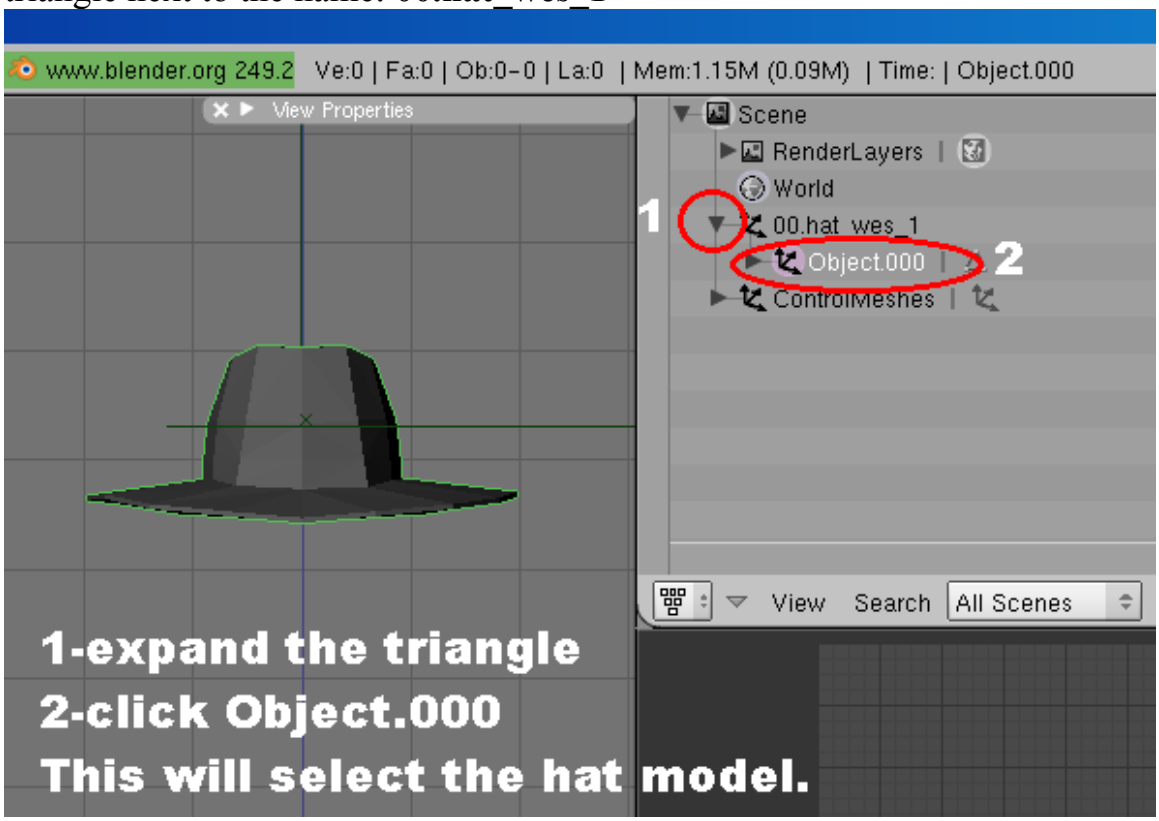
I would save them as follows:

glasses.blend
holster.blend
ring.blend
badge.blend
costume.blend
head.blend

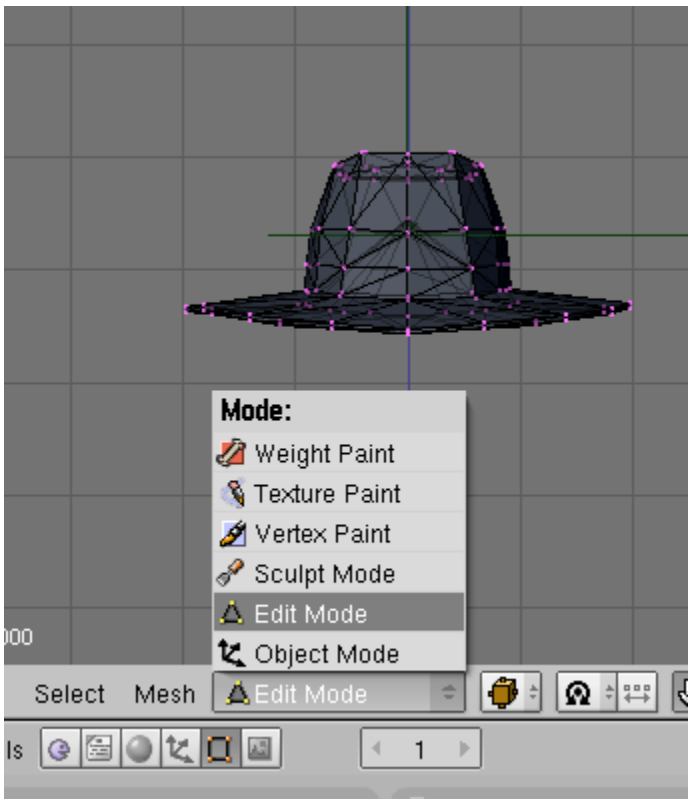
hat.blend

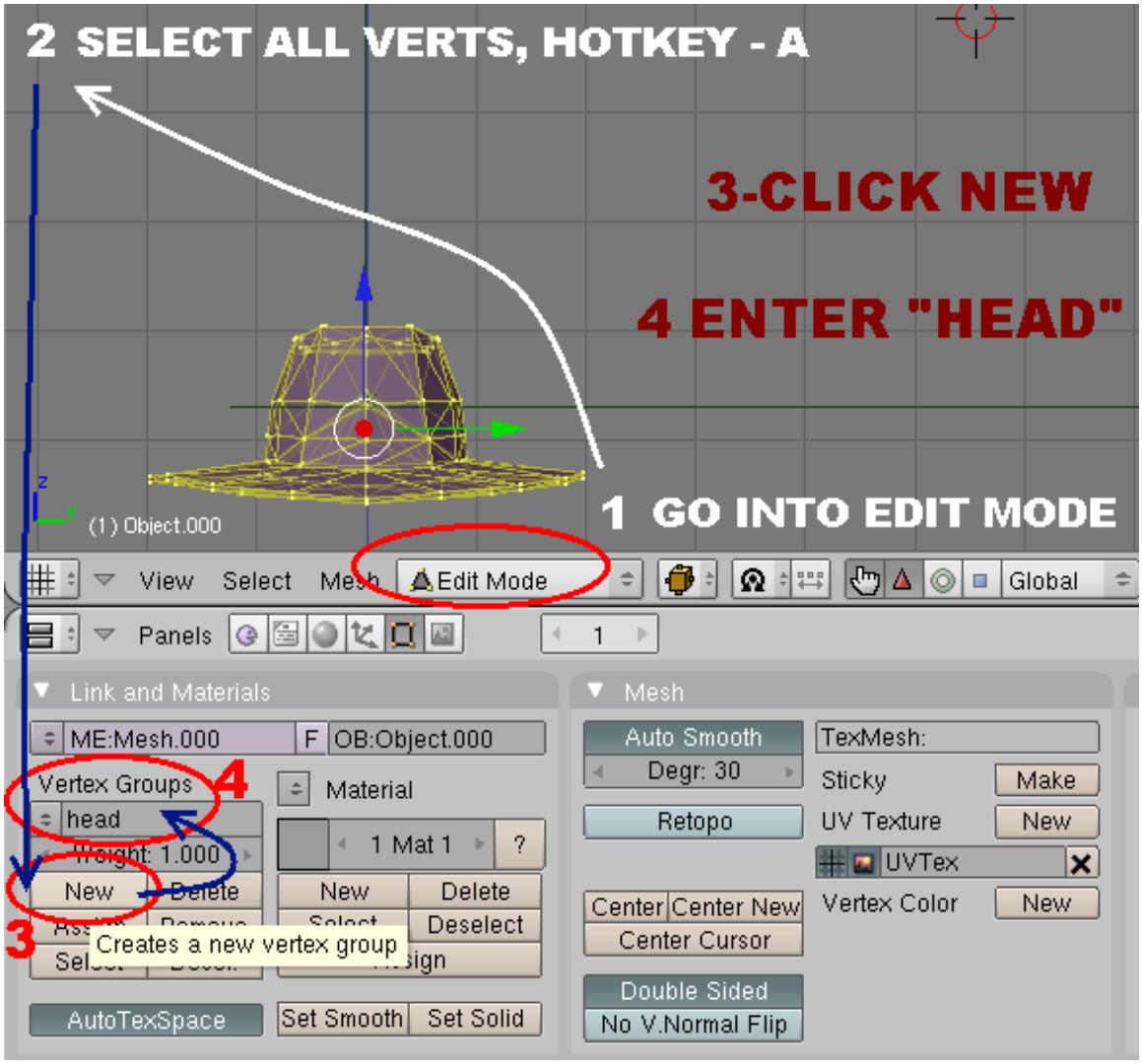
The hat does not have an armature (bones). This means it has no weight groups. Which means it will not work if we add it to the auto-animated prop. Good news is before we save the hat.blend, we can give it a *weight group*.

After the hat is imported, look to the *Outliner* window and expand the triangle next to the name: **00.hat_wes_1**

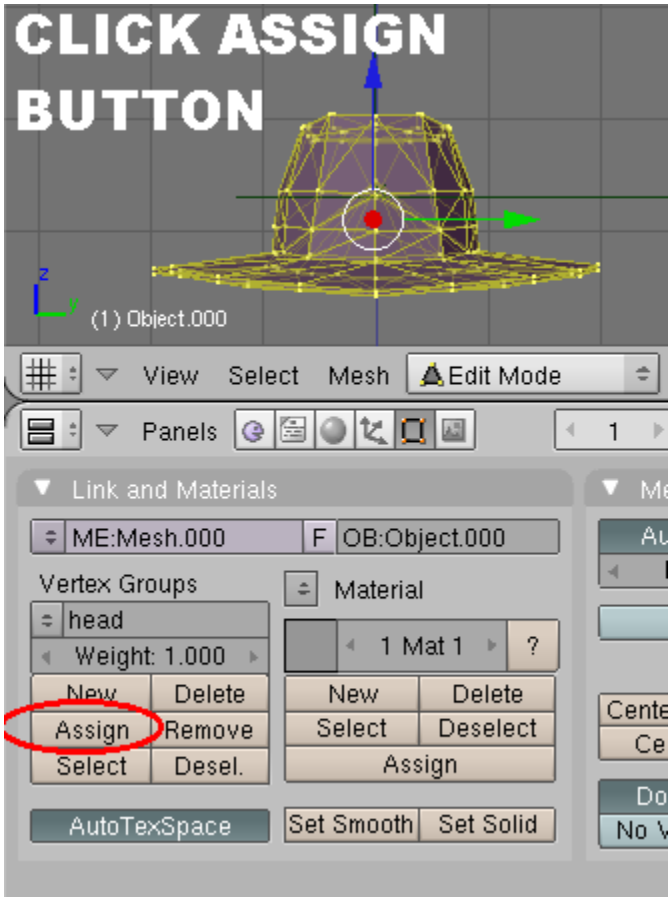


Then select the hat object model called Object.000. In 3D space the hat becomes selected. Next go into *Edit Mode*. Select all of the hat verts by pressing Hotkey **A**. All of the verts will turn yellow or light up. (like turning on the Christmas lights).


Mode:
Weight Paint
Texture Paint
Vertex Paint
Sculpt Mode
Edit Mode
Object Mode
Select
Mesh
Edit Mode
1

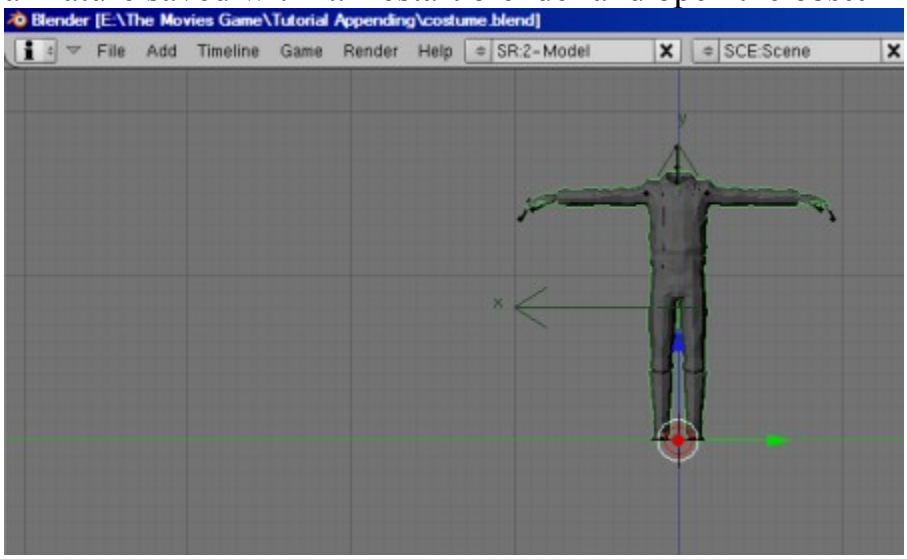


In the *Buttons* window **F9** at the far left is a place called **vertex groups.** Below are some buttons. We need the hat to move with the head bone so click **NEW** to create a new *weight group*. A new group in a small name box above that button was created called **Group**. Click on group to rename it. Title it **head**. In the armature is a bone also called head and so now the hat will move with it. With all verts of the hat being selected, go ahead and click the **Assign** button. Those verts we selected with Hotkey **A** are now assigned to the head *weight group*. Go ahead and save this as **hat.blend.**

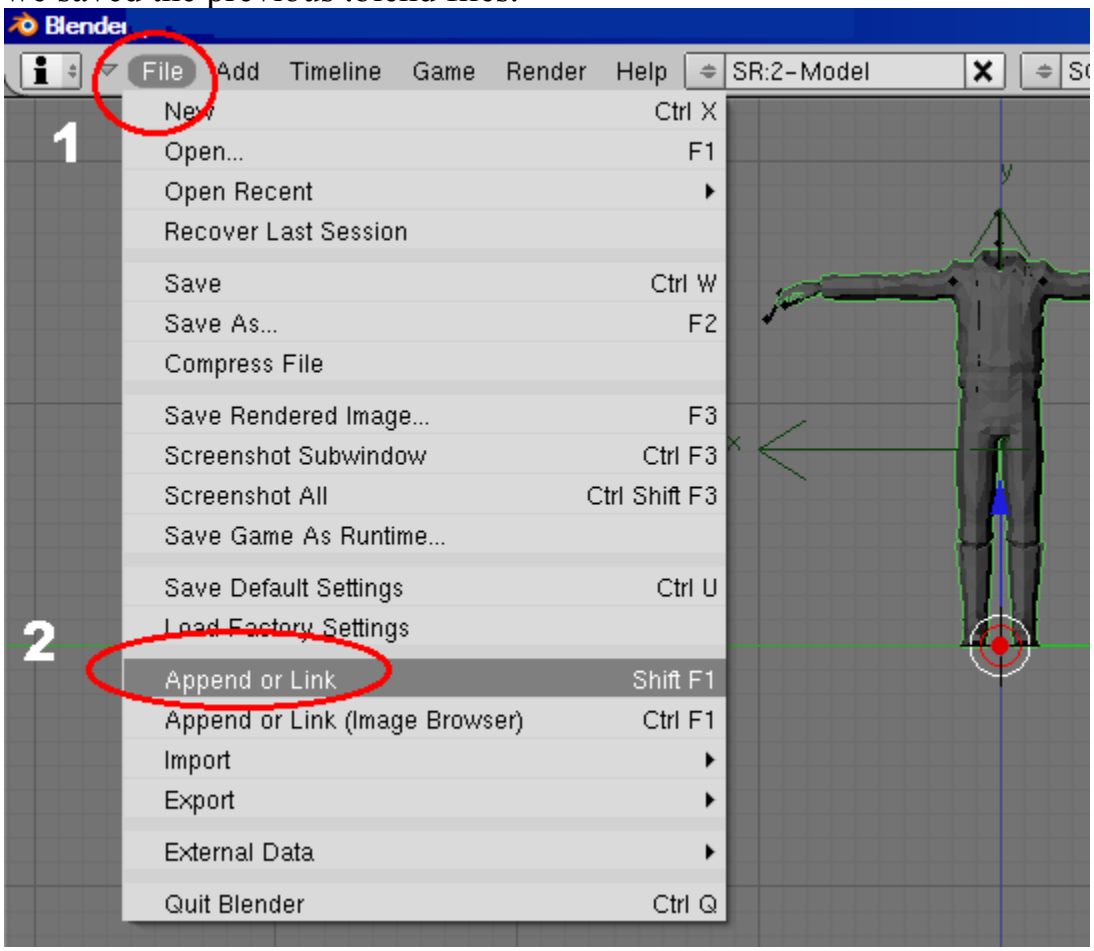


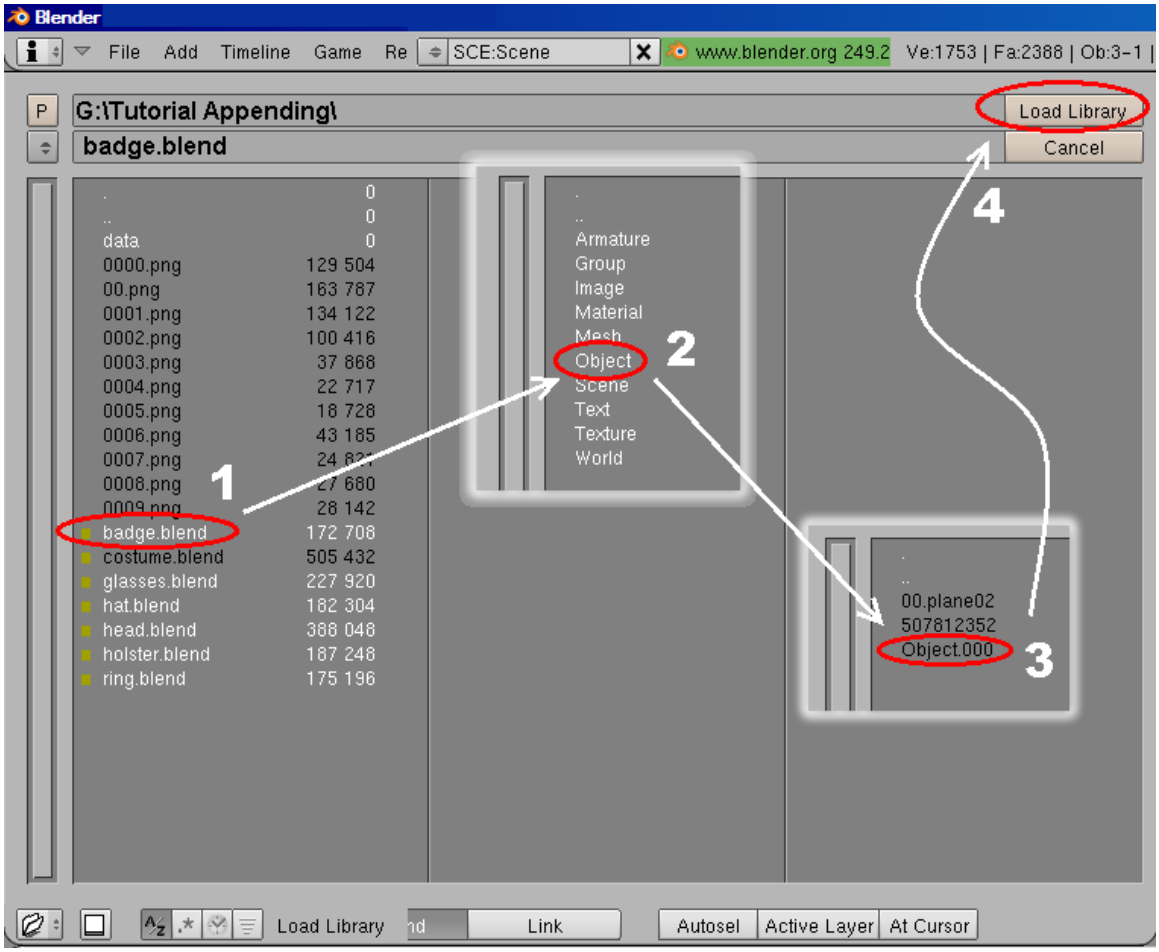
3. Let's Start Appending

All but one has a set of bones or an armature. We only need one. We need to pick one of these .blend files as the main one. I'll pick one that has an armature saved with it. Restart blender and open the **costume.blend** file.

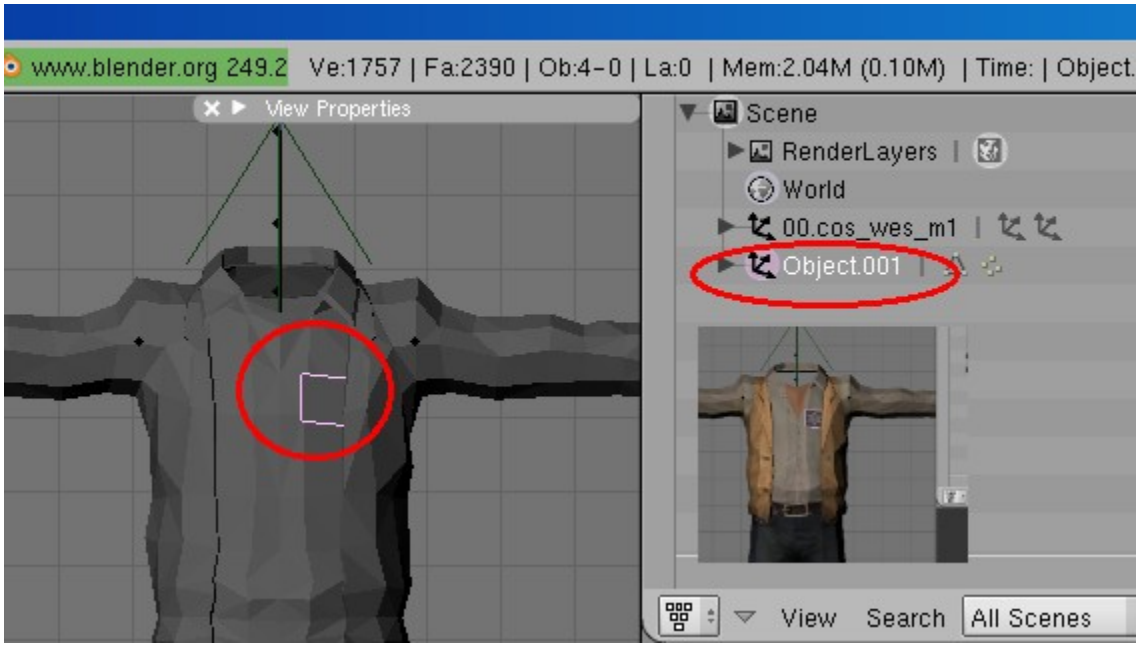


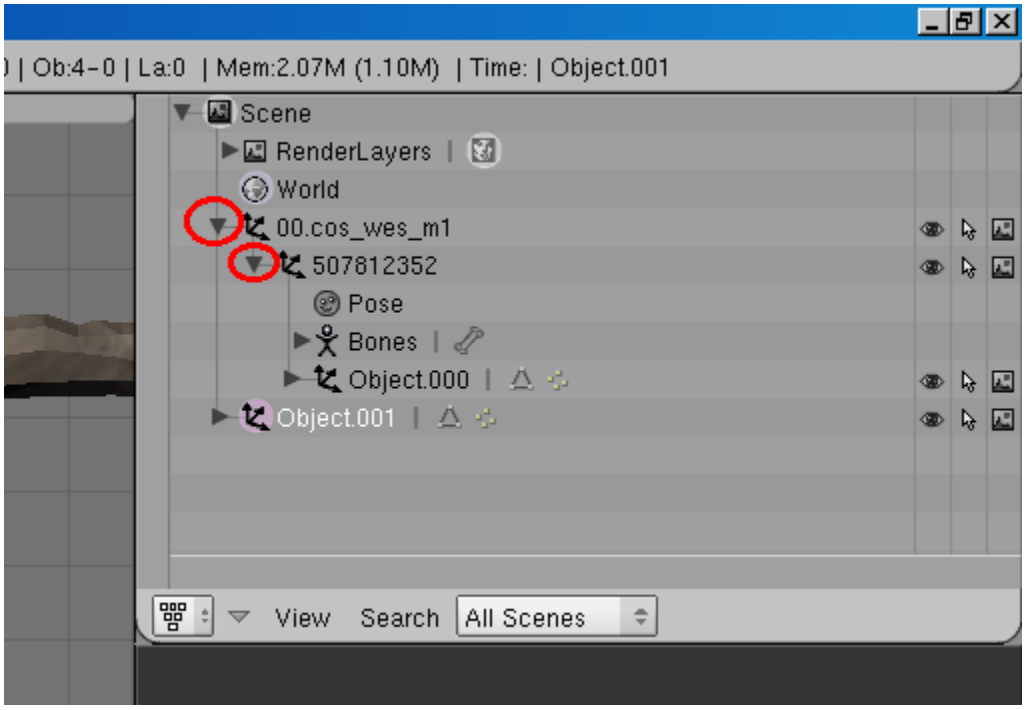
All other items will be appended to this Blender session. At the top left of the screen is a button/tab called **File**. Click it and from the drop down menu select: **Append or Link**. There is another append option below with image in it. Don't use this one. Just the first Append or Link. This opens the browser for selecting which **.blend** file to append from. Navigate to where we saved the previous .blend files.





Click on one of the .blend files we saved. Let's do them in order. Click on the **badge.blend**. This brings up another menu. We want **Object**. Next select which object. The Badge 3D model is called **Object.000**. Select it and then go up to the **Load Library** button and click it. The badge object has now been added to the costume in 3D space.

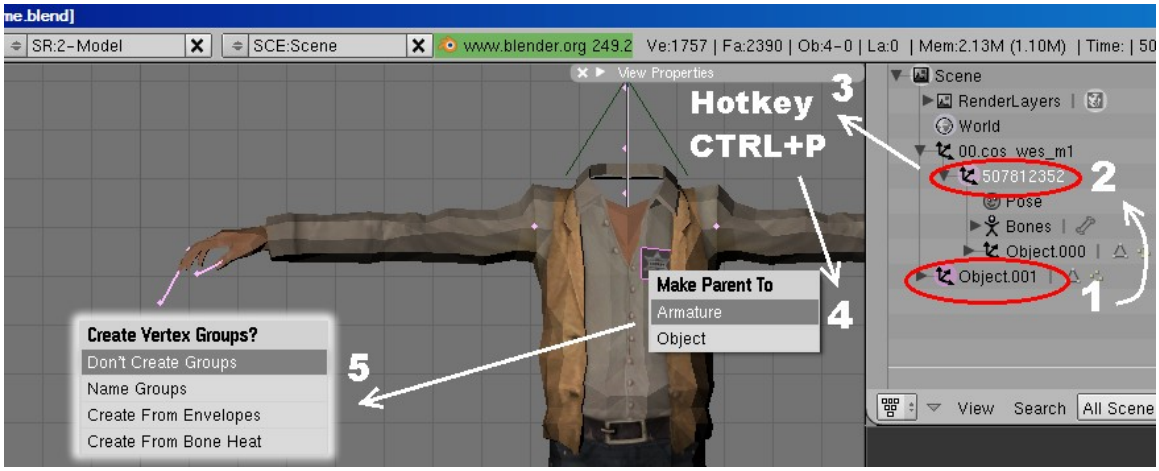


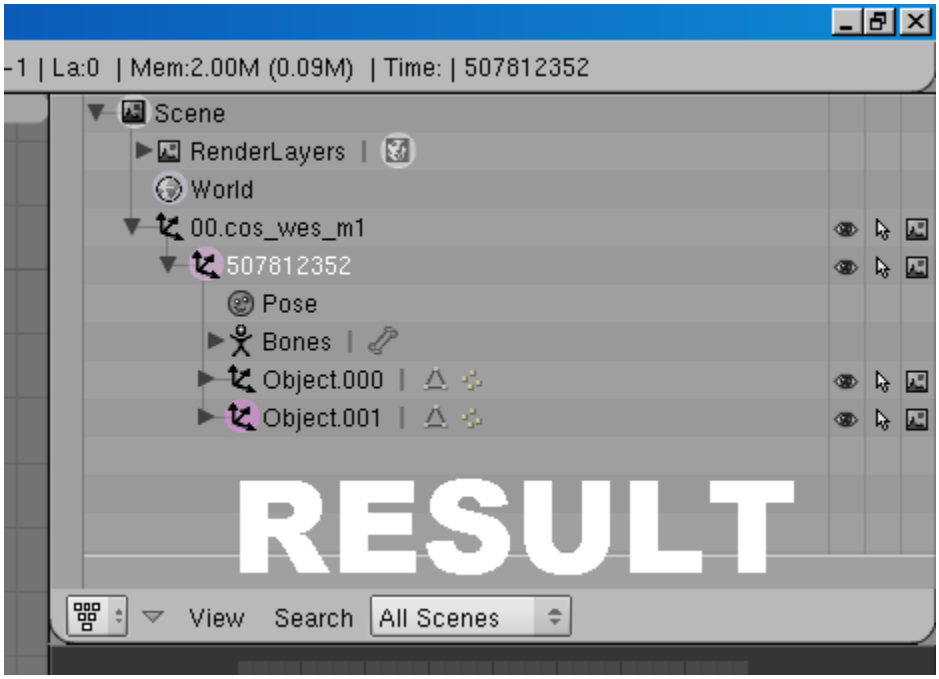


4. Parent the appended object to the armature and add it to the blend group.

We have added one object, so let's put it were it needs to be in the Outliner. In the *Outliner* window expand the triangle next to **00.cos_wes_m1**.

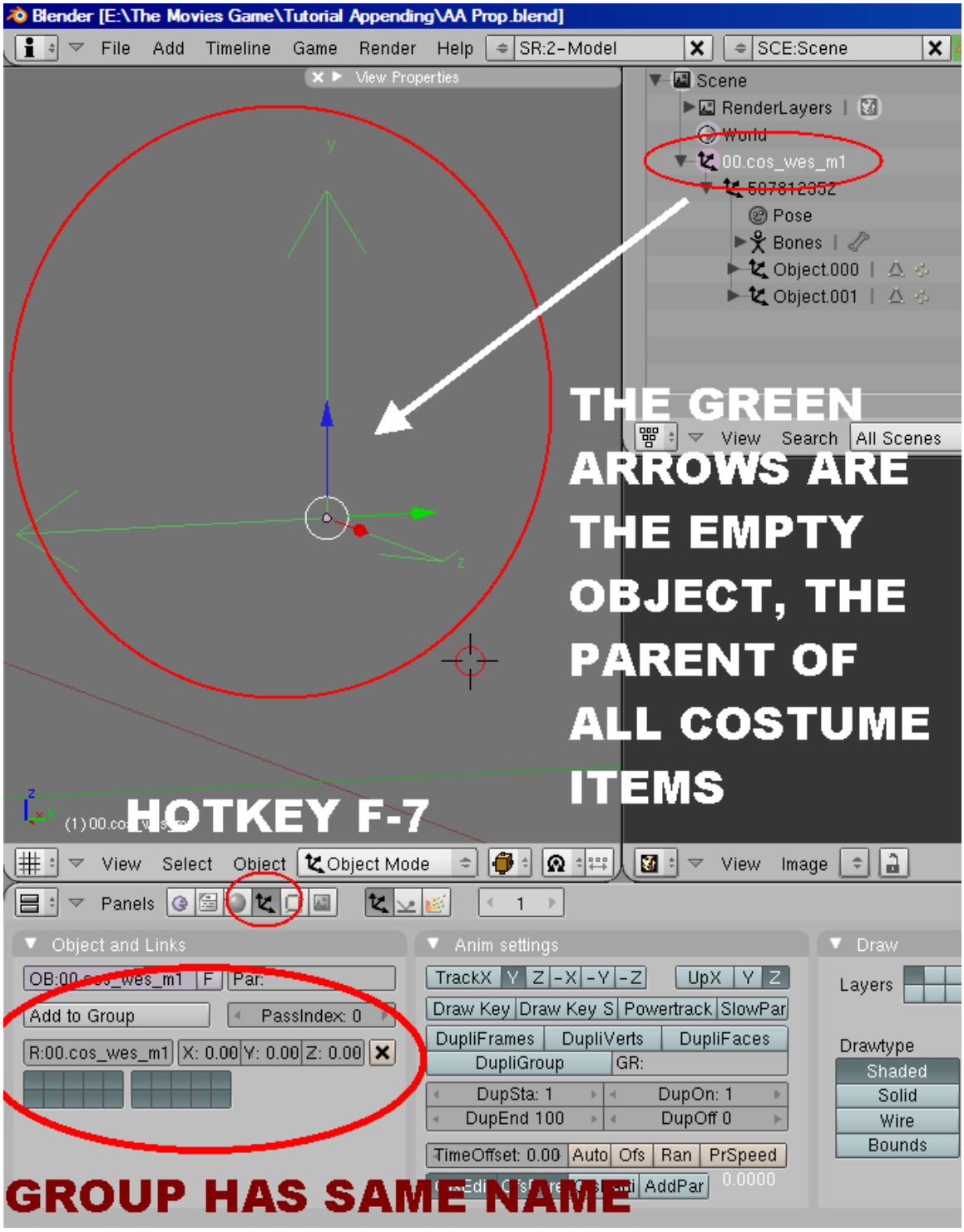
Underneath is the armature called **507812352**. Click the triangle next to it, too. Appended items get renamed if the objects already in the project have the same names. Object.000 under the armature is the costume object.
Object.001 is the badge object.
Click **Object.001** which is the badge
Now hold down **Shift** and click on **507812352.**
In 3D space press Hotkey **CTRL + P**, for *parent*.
Since the Armature (507812352) was selected second Blender will place the badge under it rather then the armature under the badge. A popup menu appears. Select **Armature**. Another popup menu. The badge already has weights so select **Don't Create Weights**. This way the badge is now under the influence of the armature. The armature can now move it.





One more thing to do here. Add the badge **Object.001** to the *Blend Group*. This is important or it won't export. All of the objects we append needs to be added to the **blend group**. What is a blend group? Blender allows you to group several items together. Our AA prop will only need one group. You can't really appreciate groups until you have more then one. Then things get interesting. Sets have lots of groups. Later, in the next example, we will append entire groups to a set.

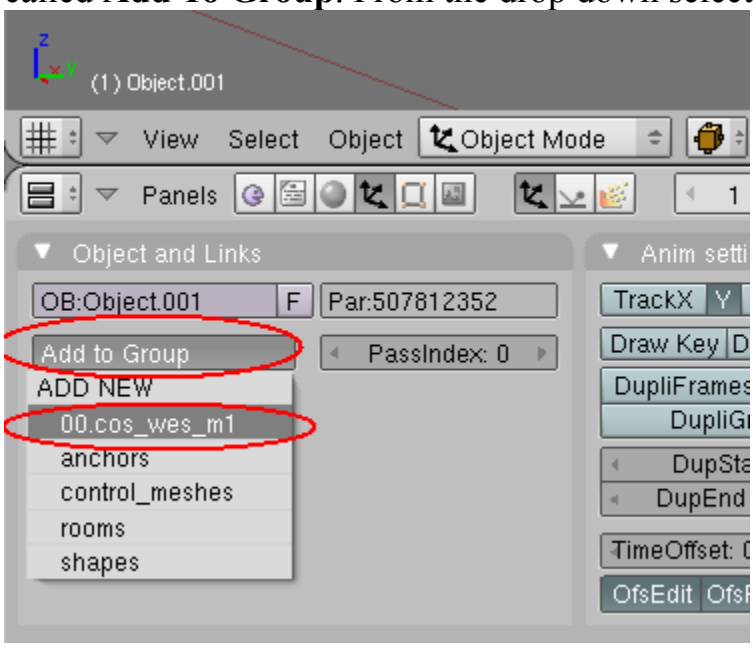
In this case the Blend Group has a name. Pay attention to this detail. It shares the same name with an object found in 3D space. So two items have the same name. And they always should have the same name. This would be different depending on which .blend file we started with. We started with the costume.blend file so the group name is: **00.cos_wes_m1**.



To see the *blend group* hit Hotkey **F7**.
In the *Outliner* click on the object called: **00.cos_wes_m1**.
In *Buttons* window, at far left is the group name.
Notice here that the 3 green arrows object (the *Empty* object) has the same name as the *blend group* which is 00.cos_wes_m1.

If you click on the costume object titled **Object.000** and look at the group name, it will also be **00.cos_wes_m1**, because it is a **child** of the **00.cos_wes_m1** object. But **Object.001** is not. It used to have a group in the badge.blend file. But since we appended it, only the object was appended and not it's group. This is a great advantage to appending. If the group did append then we would have to strangle Blender until it forgot the other blend group. But with appending this is not a worry.

Go ahead and click on **Object.001** (the badge object). Now down in the buttons window Hotkey **F7**, enter it into **00.cos_wes_m1**. Click the button called **Add To Group**. From the drop down select **00.cos_wes_m1**.



Now this object has been **appended** into the scene. It has become the **child** of the **armature**. And now it has been entered into the costume's **blend group.** We will do this with all of the objects we append. But first...

5. Save this session as another .blend file.

This way we won't overwrite the costume.blend file. Instead let's save it as **AA Prop.blend**.

6. Repeat until all items are appended to the AA Prop.blend file.

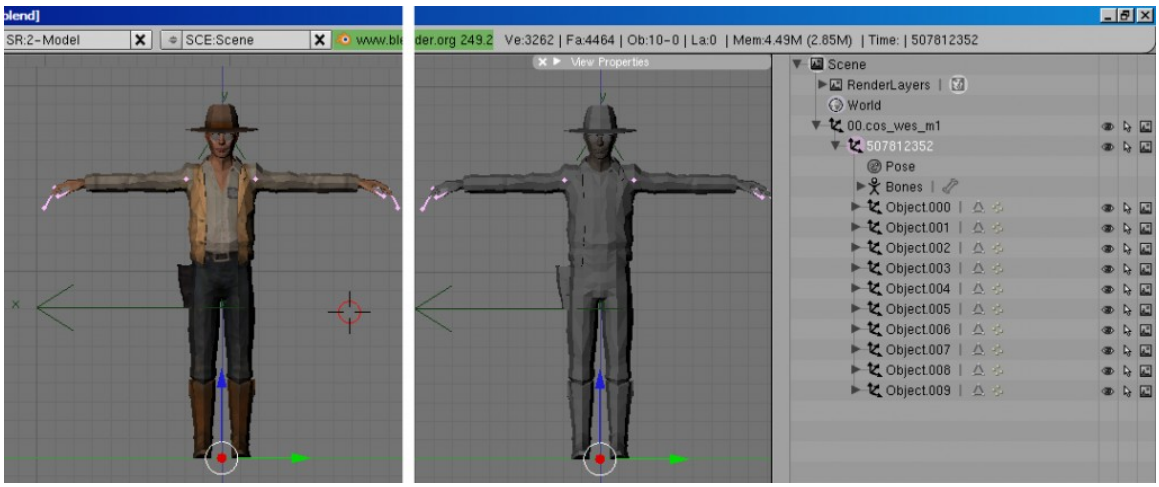
The next item is the **glasses.blend**. Go to **File**, select **Append or Link**, then navigate to where those .blend files are and select **glasses.blend**. From the next menu select **Object.** Now the glasses is made up of two objects. Select the ones titled **Object.000** and **Object.001**. With the right mouse button you can select both of them at the same time. If you need to go back and select the next one you can. Now click **Load Library**. The project already had Objects.000 and Object.001, so in the Outliner window Blender renamed them to **Object.002** and **Object.003**. We are basically going to keep repeating these steps.

Click on **Object.003**, hold down **shift**, and click **Object.002**. Keep holding down **shift** and finally click the armature object called **507812352**. With the armature selected last, it will become the *Parent*. Let go of shift key and hover mouse out in 3D space. Now hit Hotkey **CTRL + P** to *Parent*. From the popup select **Armature**, and the next popup select **Don't create groups.** Now **Object.002** and **Object.003** need to be entered into the blend group titled **00.cos_wes_m1**. First click **Object.003** and in *Buttons* window **F7** click the button called **Add To Group**. From the drop down select **00.cos_wes_m1**. Do this for **Object.002**. Now update this .blend file by **saving.**

The next one to append from is **hat.blend.** Same as before. **Parent** the hat to the **armature**. And enter it into the **blend group**.

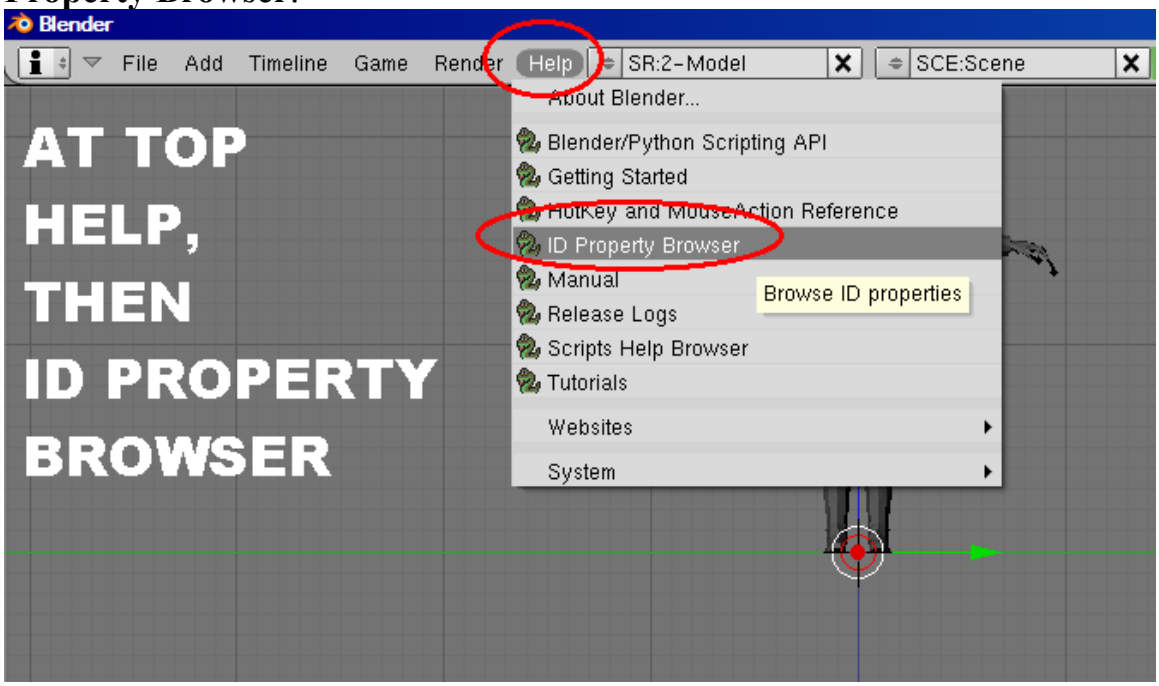
After the hat is the **head.blend** file. This has three objects. The head, eyes and mouth. Select them all at once with the right mouse button. Parent them to the armature and enter them into the blend group.

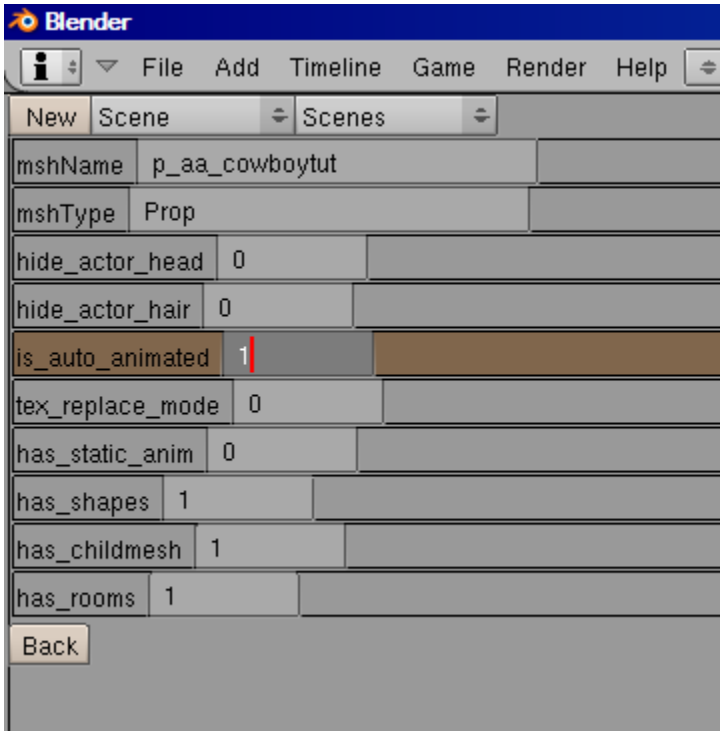
The last two blend files are **holster.blend** and **ring.blend**. Append those objects and repeat above steps.



7. Turn on the Auto Animated flag

Each .MSH file has a number of flags the game uses to know what to do with each item. One flag is the **auto-animated** flag. Before the Import/Export scripts, Meshmanip was the way to set this flag. But the 2009 update to the Blender scripts has a menu with lots of flags to set. At the top of 3D space is a tab called **HELP.** From the drop down menu click **ID Property Browser**.

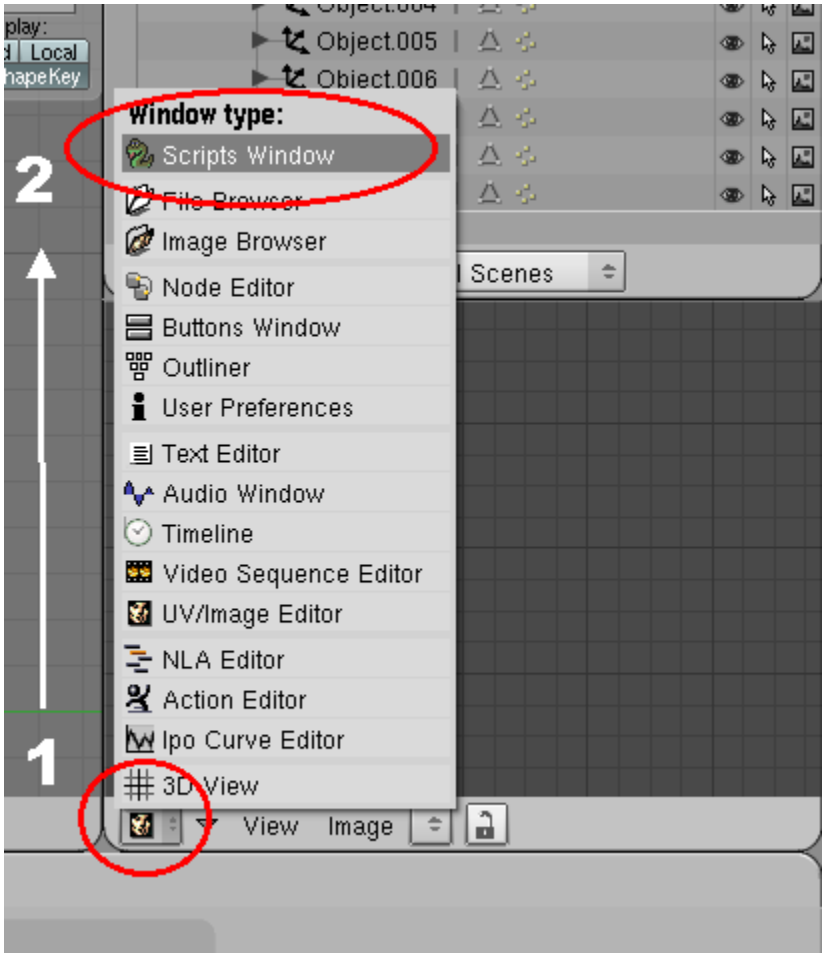


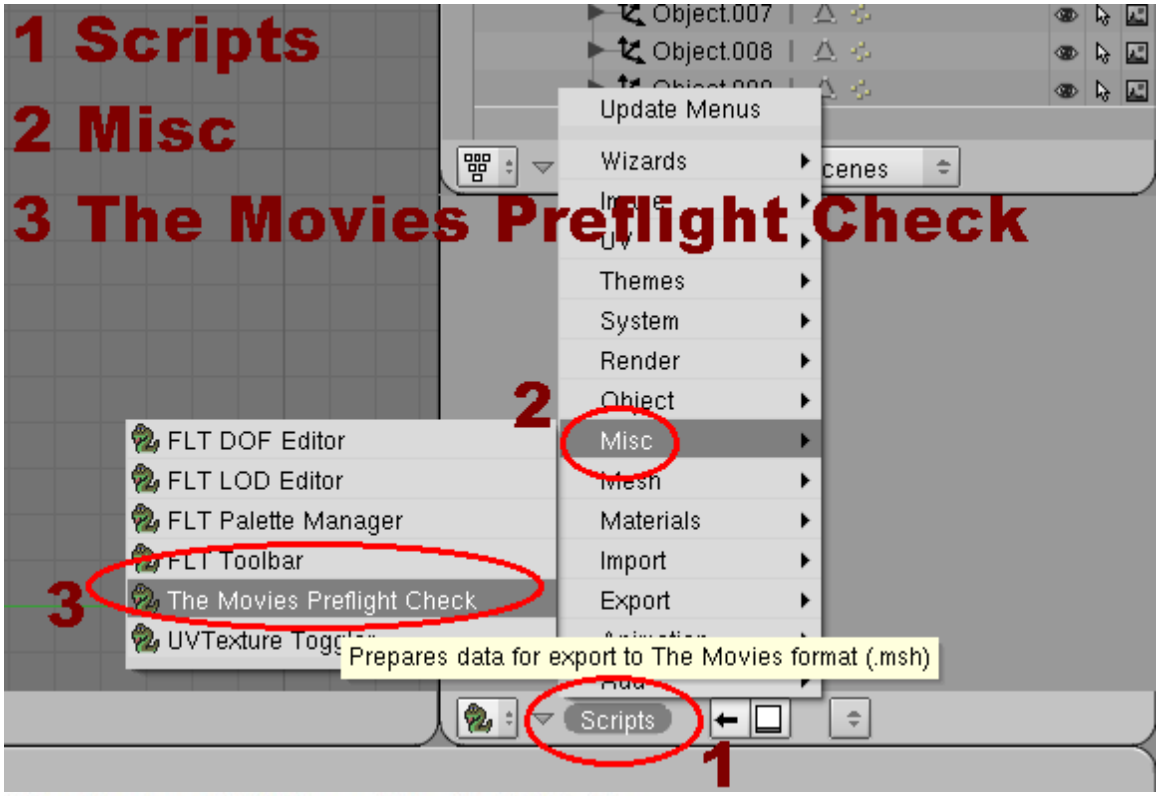


Where it says **Auto-Animated** is set to zero. Zero means NO! **One** means YES! So change 0 to 1.

8. Run The Movies Preflight Script

Is there a way to know if something went wrong? Yes. The Movies gods (DcModding) bestowed upon us the Movies Preflight Script. Running it will go over the entire project to see if the project is ready for export. I have to turn one of my windows into a *Scripts* window to run it. I will turn the *UV Image Editor* window into the *Scripts* window. At the bottom left of the *UV* window is a little icon. Clicking it brings up a popup menu. Select *Scripts* window**.** This changed the window. At the bottom of this window is a button called **Scripts.** Click it and select from the popup **Misc.** Now select **The Movies Preflight Check**.


Object.005
Object.006
Window type:
Scripts Window
File Browser
Image Browser
Node Editor
Buttons Window
Outliner
User Preferences
Text Editor
Audio Window
Timeline
Video Sequence Editor
UV/Image Editor
NLA Editor
Action Editor
Ipo Curve Editor
3D View
Scenes
View
Image
2
1



The Movies MSH Export Preflight Tests

Start Time: 2014-11-04 21:02:13
Duration: 0:00:01.856000
Status: Pass 3550

Results for checking validity and integrity of mesh to be exported to The Movies game format.

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
BasicTests: Verifies very basic requirements.	4	4	0	0	Detail
ArmatureTests: Tests several aspects of any Armatures in this mesh.	4	4	0	0	Detail
InGroupTest: Each Blender Object is in a Blender Group.	11	11	0	0	Detail
MultiGroupTest: Each Blender Object is in only ONE Blender Group.	11	11	0	0	Detail
GroupHasEmptyTest: Each populated Blender Group has an Empty pivot object.	1	1	0	0	Detail
GroupHasOrderNumberTest: Each Group has a number in the name, determining export order.	1	1	0	0	Detail
Group-ParentName: The name of the Group and the parent Empty are the same.	1	1	0	0	Detail
Group-PropertyName: The name of the Group and the IDProperty 'grpName' value are the same.	1	1	0	0	Detail
GroupHeirarchyTest: Meshes must be children of an Empty or an Armature.	10	10	0	0	Detail
BoneVertexGroupName: Vertex Groups must match Bone names.	10	10	0	0	Detail
Parent-PropertyName: The IDProperty 'grpName' and the name of the Empty are the same.	1	1	0	0	Detail
MaterialZeroTest: Each Grouped Mesh has a material in the first slot.	10	10	0	0	Detail
MaterialTypeTest: Textures in Materials must be Images.	16	16	0	0	Detail
UnusedMaterialsTest: Checks for unused Materials.	10	10	0	0	Detail
UnusedTexturesTest: Checks for unused Textures.	40	40	0	0	Detail
UnusedImagesTest: Checks for unused Images.	11	11	0	0	Detail
UVTest: Meshes must have UV mapping.	10	10	0	0	Detail
UVLayerNameTest: UV Layers must be named 'UVTex' or 'LightMap'.	10	10	0	0	Detail
NonFaceVertsTest: Checks for vertexes that are not in any faces.	10	10	0	0	Detail
VertWeightTest: Checks for vertexes that are not weighted to any Vertex Groups.	3378	3378	0	0	Detail
Total	**3550**	**3550**	**0**	**0**	

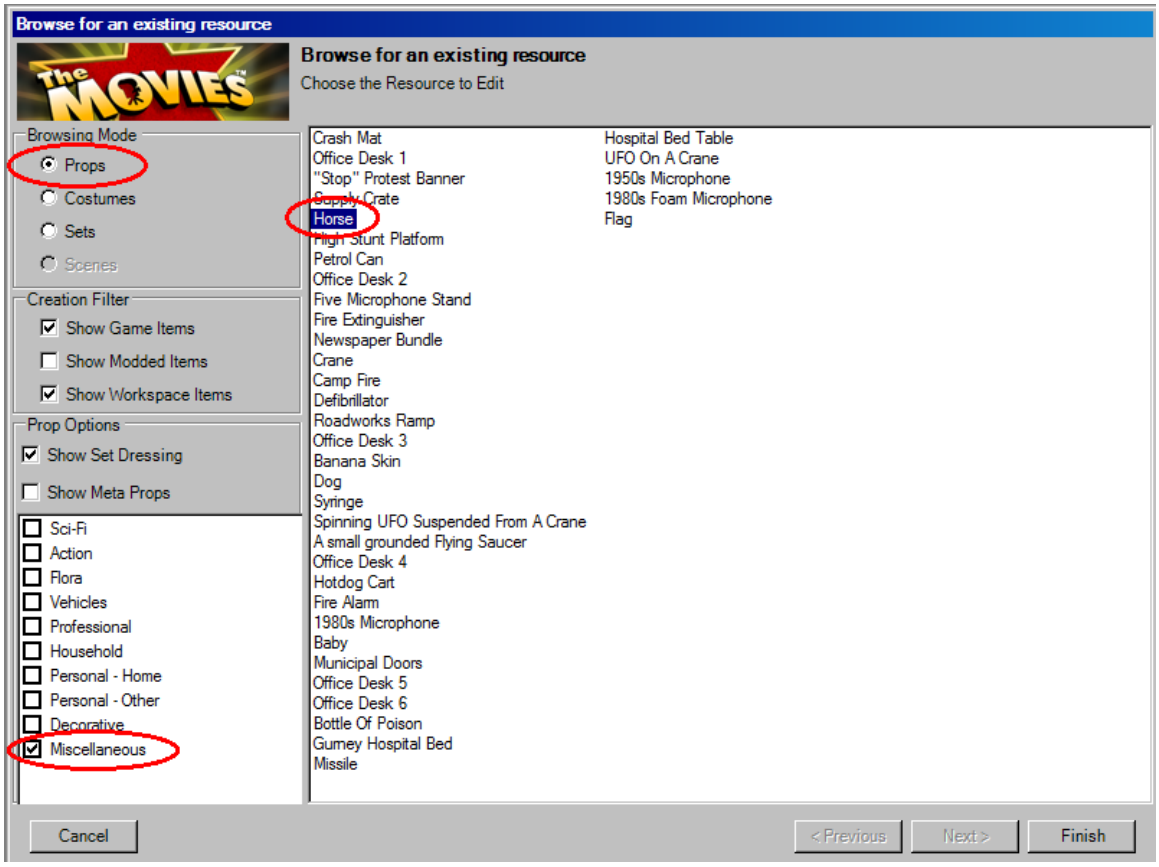
I got all green lights so I'm ready to export this prop.

Auto-Animated props are titled **aa_p_(name of prop).msh**. A p_aa_ in front of the name. The P is more important then the aa. But we will title it

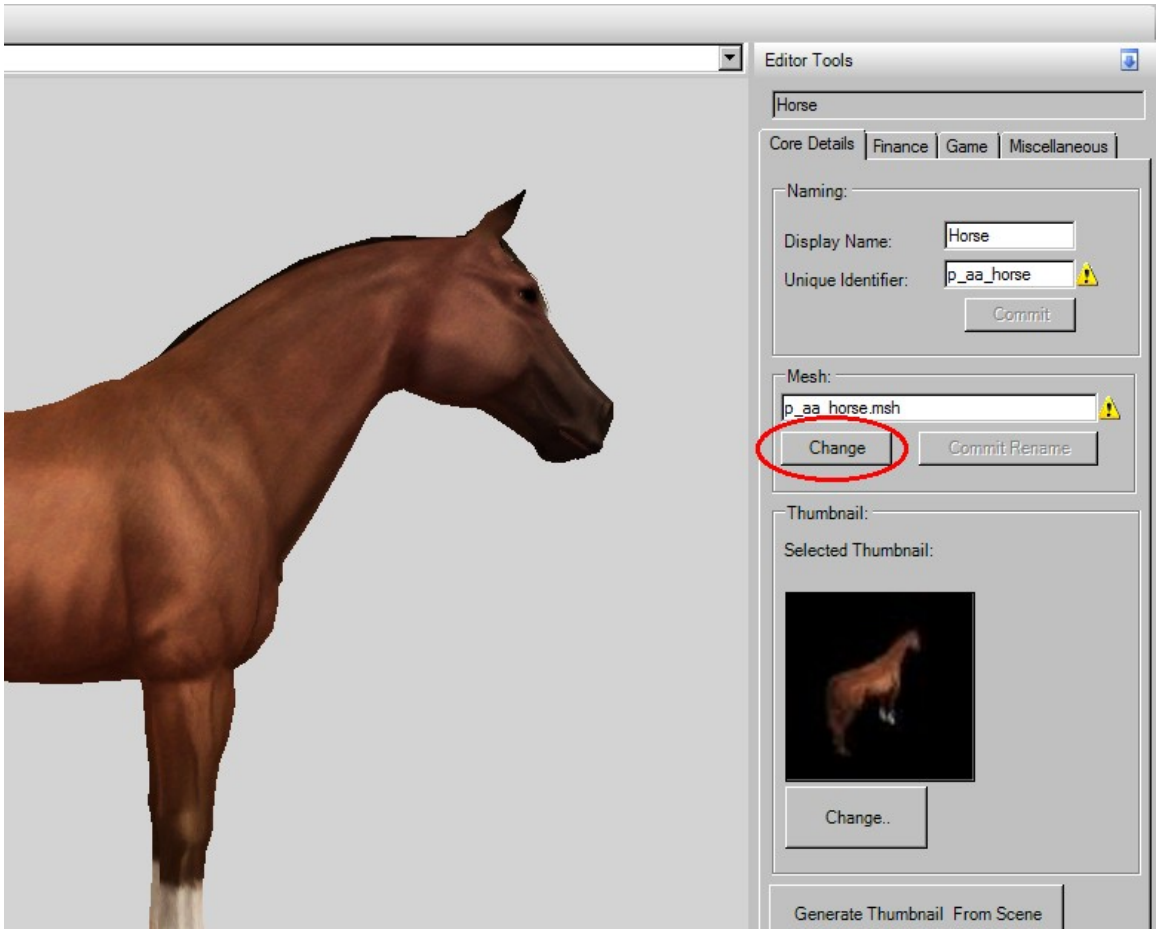
p_aa_cowboytut.msh. It goes in the data\meshes folder of your game.

9. Make the item available to the game.

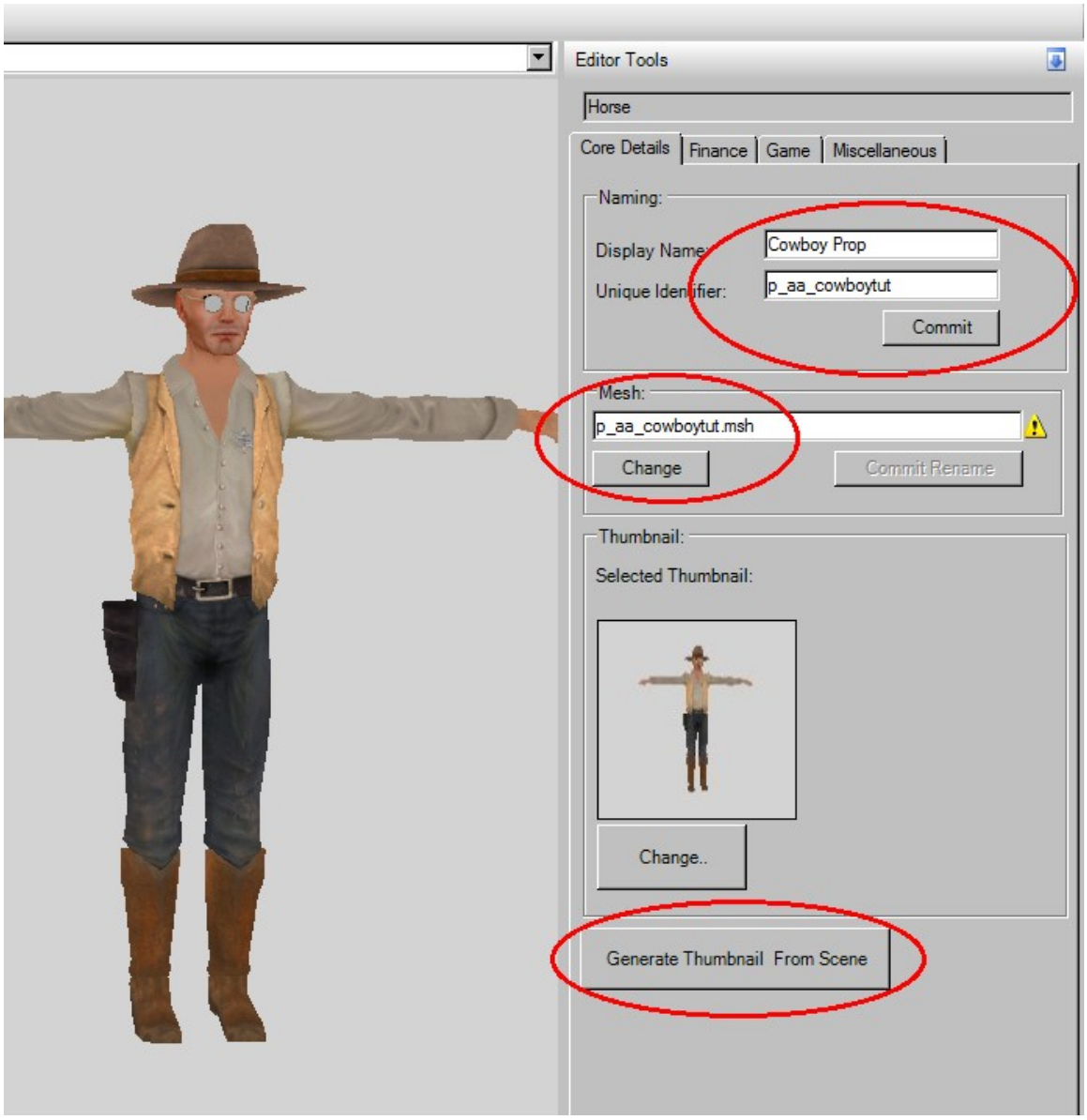
There is a number of ways to do this. It is a matter of making the right files in the right folders. But the easiest way is to use MED (the Movies Game Editor). Open MED and wait for the window that asks what do you want to do today. When it does select **Modify (Create From Existing Content).** By default the props are already selected. And the Sci-Fi props are already selected. Deselect Sci-Fi and select **Miscellaneous** instead. From the list of items select the **Horse.**



We are going to replace the horse with the cowboy we just made. At the left there is an entry for what **Mesh** file is presently being used with this prop. Notice the horse mesh is titled: **p_aa_horse.msh**. This horse is also an Auto-Animated prop, which is why I chose it. Click the button **Change** under that name. Select the name of the mesh we exported: **p_aa_cowboytut.msh.**



Change the **Display Name** and **Unique Identifier**. **Commit** the rename!!!! And then make the thumbnail by clicking **Generate Thumbnail From Scene** button.



Now commit to game or save it to workspace.

10. The Auto-Animated Prop Needs Animation

This prop is ready but it needs one more thing to work. One or more animation files. For human costumes there are tens of thousands of animations to choose from. All we need is one but you could have more then one if you wanted. If you have more then one the game will randomly choose from the list you give it. I will only give it one.

You can find **animation files** and extract them using MED. Use the **file extractor.** They will be at location: data\animations\high. After expanding all the plus signs scroll down till you find a file called: **wes_showd_hero_lk_around.anm**. Select it by adding a little **check mark** in the box. Next fill in the **address bar** to where you want the file extracted to. Then click **Finish**. Close Med. Now get to where you extracted the file to. Rename it to this: **aa_p_aa_cowboytut_v00.anm**. All animation files for Auto-Animated Props have a *aa_* in front of the name. The *_v00.anm* is the number of the animation to be picked by the game for animating the prop. If you select more animations you would rename them to end with a *_v01.anm* and _v02.anm. As many as you want. But keep in mind walking won't look good. Mostly standing or sitting animations will work best for human costumes.

After renaming the animation place it where the game can find it in it's own folders. If you don't have any of the sub-folders then just create them. Data\animations\high\autoanimated

Appending is very handy. Once I made a .blend file containing all .HD starmaker heads. Whenever I need a head I just append it from this .blend file.
Now that we know how to append objects, let's see how to use appending to make better sets in part two of this tutorial.

Part 2, Appending Entire Groups

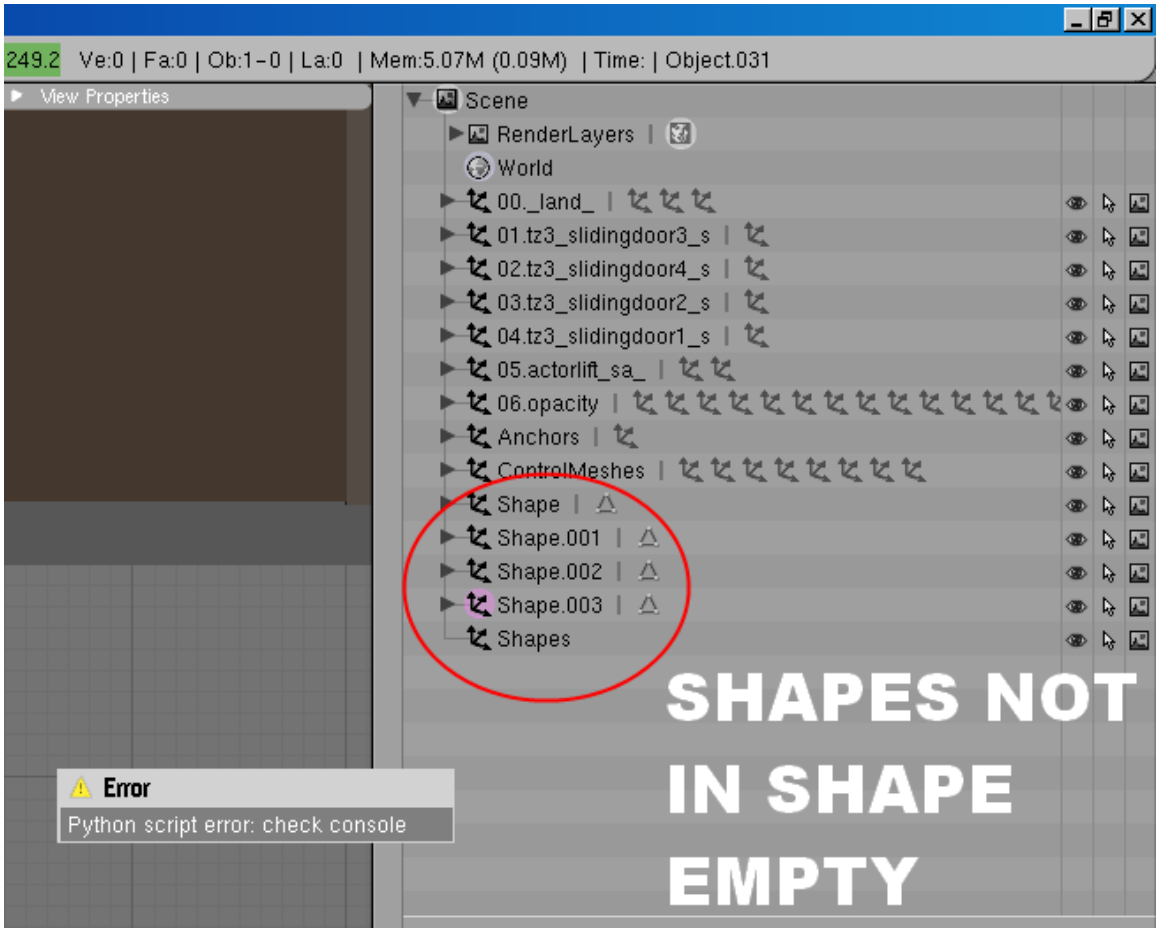
We know how to append Objects to your blender session, but what else can appending do? Append groups.

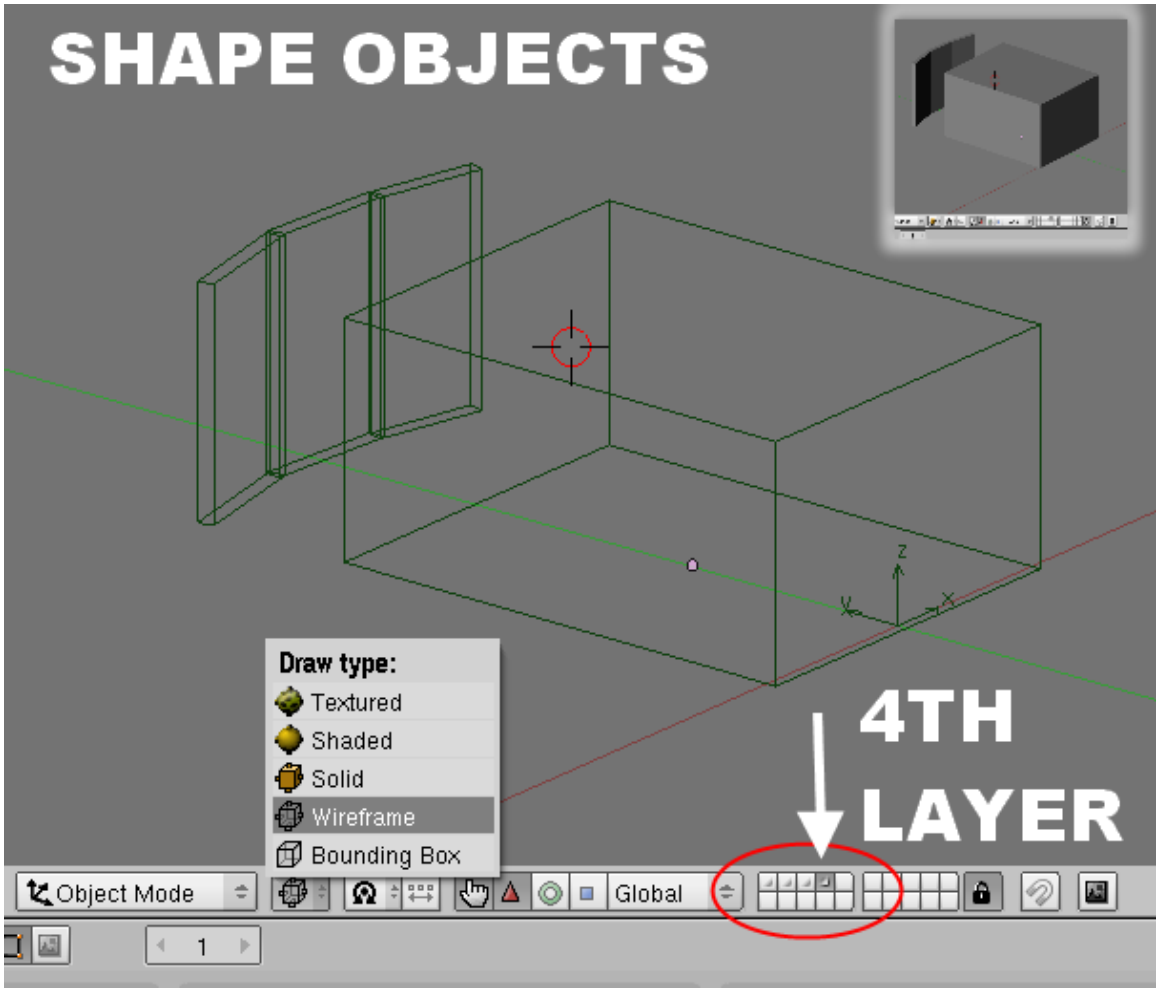
MED (The Movies Game Editor) allows us to make new sets out of existing sets. We can change the image textures of a Movies game set to get a new set. Medieval brick walls can turn the hotel bedroom into a castle bedroom. But MED leaves something out of sets that is very important to them. The **extrainfo file**. In the data\meshes folder is another sub-folder called **Extrainfo**. Here is a collection of files with the same names as the mesh files. Their purpose is to activate the advanced portions of the mesh file with the same name. The pistols and lasers have special effects activated by this file. Sets have doors and other layer items that are activated by this file. MED does not give a new or custom set this file. Chances are, if you have downloaded a set from 8eyedbaby, it did not come with this extrainfo file. The most annoying problem when this happens is you can not remove the set from the studio lot. Or move it around. Because you can not place a builder on the icon to have it moved or destroyed. If this happens to you, find another set's extrainfo file, like **set_stage.inf** and rename it to the same name as the custom set mesh file. Start your game and you can now drop a builder on the icons. After the set is gone go ahead and delete the renamed extrainfo file. Problem solved.

A good update to MED would be the option to copy and rename an extrainfo file for your set. Saving you the work of doing it yourself. What's the big deal? Not everyone knows about the extrainfo file. And this would give them the opportunity to learn and not make the mistake later.

This would also work with new weapons. Another blend group for weapons decides what special effect the game needs. The name of the pivot and group decides where and what special effect happens. The extrainfo activates this stuff. And does so for sets, too.

An actual bug of MED is how you extract a set from the game. If you are working on one in MED and save it to workspace or commit it to game, all of the shapes will be removed from the shape empty. However, if you extract a set with mesh extractor, then the shapes stay inside the shape empty. It's not a big problem. When you import such a set into Blender, you will get a python error. All you have to do is parent the shapes to the shape empty. Or just delete them.


249.2 Ve:0 | Fa:0 | Ob:1-0 | La:0 | Mem:5.07M (0.09M) | Time: | Object.031
View Properties
Scene
RenderLayers
World
00._land_
01.tz3_slidingdoor3_s
02.tz3_slidingdoor4_s
03.tz3_slidingdoor2_s
04.tz3_slidingdoor1_s
05.actorlift_sa_
06.opacity
Anchors
ControlMeshes
Shape
Shape.001
Shape.002
Shape.003
Shapes
SHAPES NOT IN SHAPE EMPTY
Error
Python script error: check console



Much of the stuff for the sets is in other layers. For instance, the outline of the set when it is placed on the studio lot. zHeight, which is where actors really stand. If you edit the *zHeight* of a set, you could make people start walking in mid air. Why? Actors can't walk up stairs. Actors can walk up zHeight and collision. The *outline* of the set to prevent it from being placed on or near another set (this can be changed, lol.) Or the shape that knows what is around it adds or decreases the value of the set once it is built on the studio lot. None of this stuff is active without the extrainfo file.

For this tutorial I will extract a **Sci-Fi Corridor**. I opened it in MED and saved it to workspace, but you could also use mesh extractor, which would be better as the shapes will be in the shape empty and no python error when importing in Blender. Use file extractor to extract also the extrainfo file.

The idea of this tutorial is to append all of the extra layer items to your custom set. You can do this simply by appending the groups that this stuff is already assigned to. This way you do not have to build them from scratch.

You can of course. Often I will replace a collision object and zHeight with others. The problem with doing this however is the extrainfo file is designed to accommodate the original shape of a set. Keeping that in mind you can do it anyways and still make fine custom sets. You could add more planes to the collision and zHeight to expand the boundaries of the set. What I do is after selecting the *Collision* in the second Blender layer, go into edit mode and add some more plains, lining them up where they need to be. But here is an important tip. The script breaks all quads into tris upon import and on export converts all tris back into quads. Except the second layer faces. Those end up losing a tri. Why? A bug maybe. I don't know. To prevent this, before export, convert all quads in the second layer into tris yourself and then the script will export them properly.

The other reason to use an original set as a source of extra layer data, besides saving you from having to do it yourself, is that scenes are designed to fit in a set's shape. If you make a set that goes beyond the boundaries of a normal set then no actors will ever visit that part of the set. For those areas you would have to make your own scenes with FLM Reader Zero. Which is time consuming but sometimes the only way to achieve what you are after.

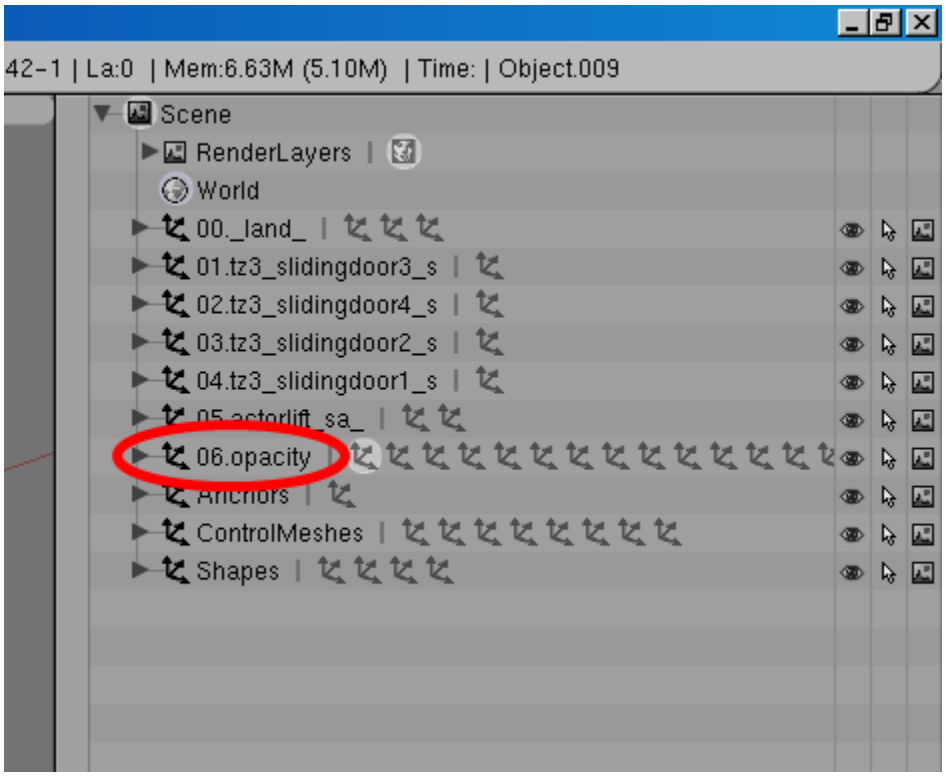
The example I will give here is to make a dungeon set that is the same shape as a Sci-Fi corridor set. I won't make sliding doors, which would be nice but this isn't going to be a mod, just a way to show you how to append groups. All of the walls and backdrop and floors and stuff belong to one group. In part 1 of this tutorial you know how to see the group name in the buttons window Hotkey **F7**. The names match the names of the green arrow objects to which all of these objects are parented to. The sliding doors of the Sci-Fi corridor set belong to their own group. The green arrows are targeted by the game. And activated by the extrainfo file mentioned earlier. What ever objects are parented to the green arrows of the doors, will move when the scene for those doors is run. If you parented a 3D elephant object to them then the elephant, too, would slide sideways when the doors opened.

Here is a very important point about set group names. They follow an order. There are numbers in the name. These numbers and names can not change if you are going to use the extrainfo file. Another important point is just because you are not going to use sliding doors does not mean you should get rid of those empty arrows or their groups. Removing them will renumber all of the rest of the objects and your new set will loose functionality in the game. All names and numbers must remain the same. Therefore you must

append every group to your new set.

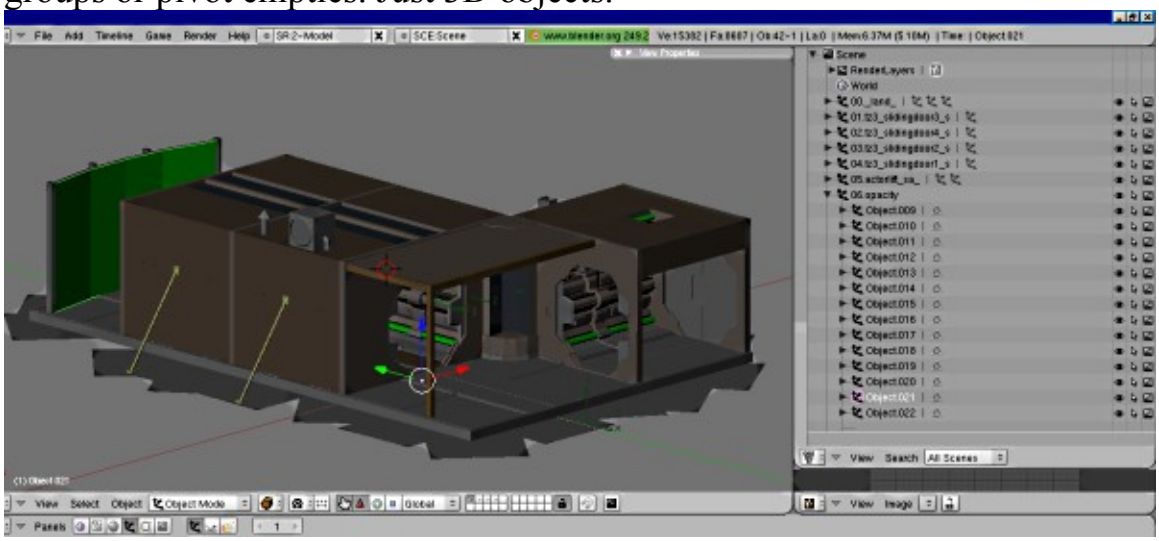
There is a trick here. We do not want any of the set's walls or doors, or floors or anything. You can keep the actor lift if you want, (if your set hides the object like the original does.) We only want the items found in the extra layers. After importing the set, I am going to delete all of these objects. You may join them all together and delete their materials, then set their appearance to wire view and also set them to X-ray. You can use this as a dummy object so that you know where everything is. The problem with deleting them is they are not really gone. If you delete them, then saved the set as a .blend file, then appended the group they belong in, all of the objects show up again. Why? Blender is so powerful that it remembers things it no longer needs to remember. Sometimes a real problem. What can we do? Save over the .blend file three times and load it three times. In other words save it. Restart Blender and load that same .blend file again. Then repeat. Save and load three times. And this will clear away Blender's memory. Moooost of the time. This tutorial will do that.

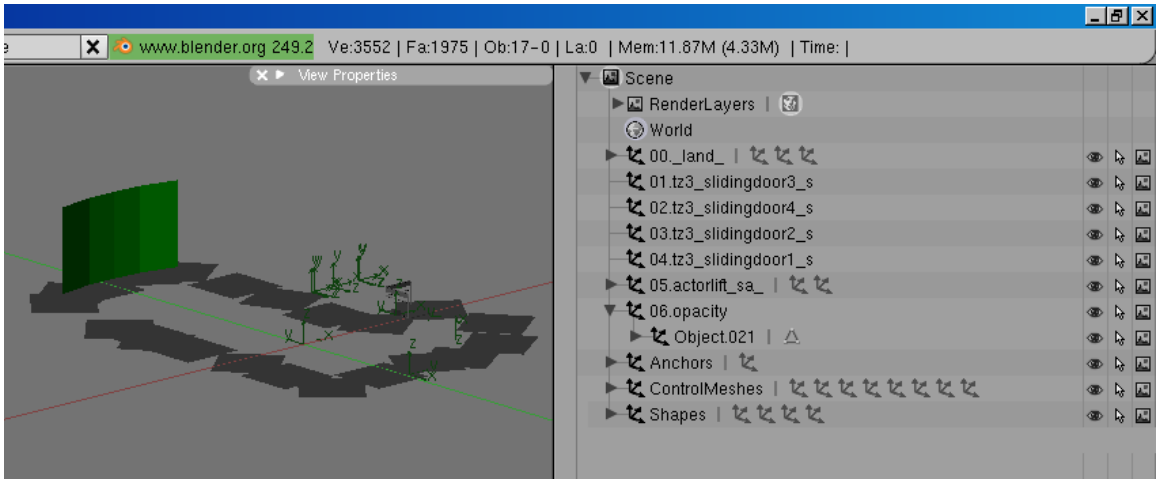
First I extract **set_corridortz3_v2.msh** using MED. Then import it into Blender.



In the outliner we can see the **empties**. We also know that each has a **group** that has the same name. Only one of these contain items we do not need:

06.opacity. The stuff from the set I don't need is parented to this empty. Expand it by clicking the triangle. Object.021 is that backdrop. I want to keep that so I'll turn it's visibility off. To delete everything else listed under the 06.opacity just right click it in the *Outliner* and select delete. I will also delete the sliding doors. They are parented to the items called:
01.tz3_slidingdoor3_s
02.tz3_slidingdoor4_s
03.tz3_slidingdoor2_s
04.tz3_slidingdoor1_s
I won't delete the green arrow objects they are parented to. Never delete any groups or pivot empties. Just 3D objects.





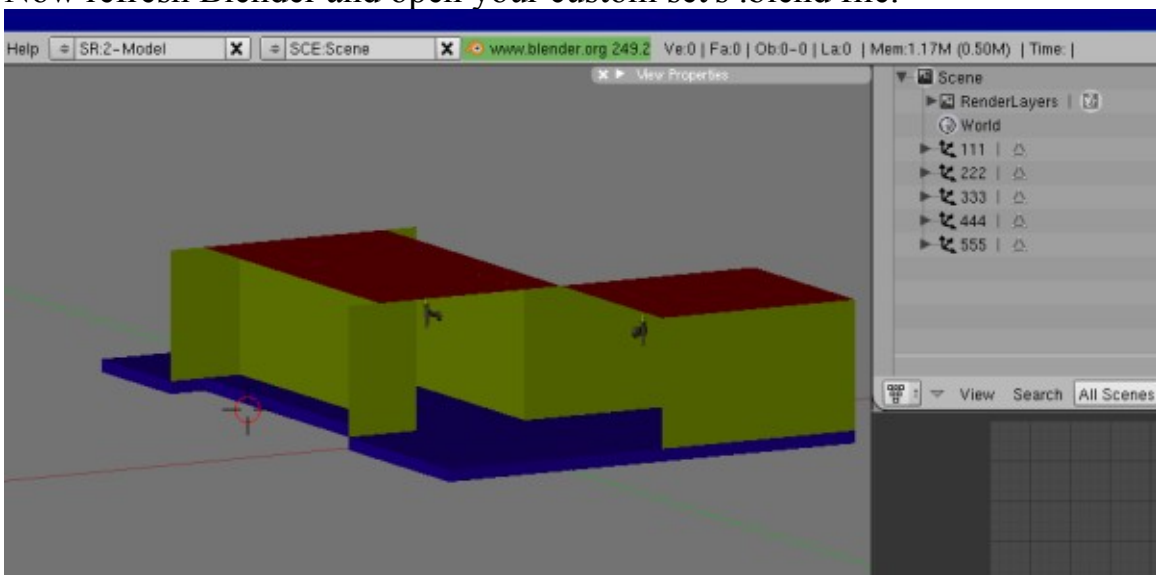
Here is what the model now looks like. And what the *Outliner* looks like. The actor lift is still there and it is set to the lightmap that originally came with the game. You don't have to but you can exchange it for your custom set's lightmap, which is what I did. Also kept is the backdrop. Also changed

it's lightmap to my custom set's lightmap. Both in the uvmap and in the material button settings.

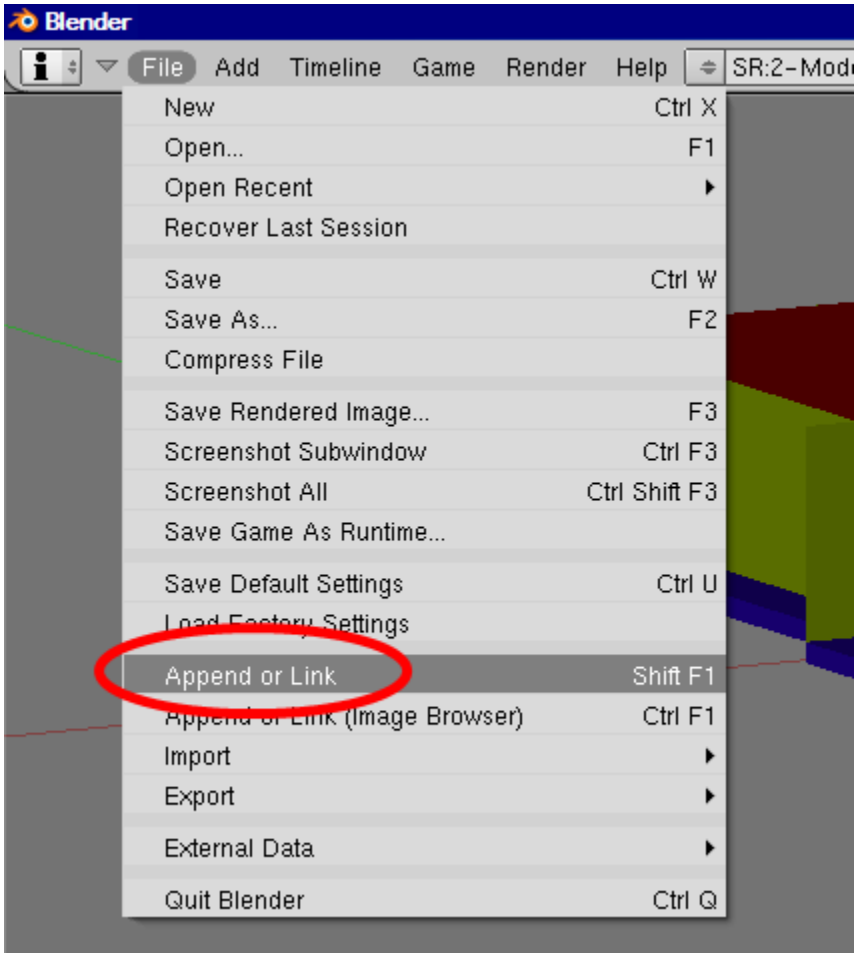
The next step is to save it as a **.blend** file. Something like **Temp Append Set Stuff.blend.** I did the saving and loading 3xs to clear out Blender's memory as I mentioned earlier.

By this time you should already have your own set. And let's assume your set is like mine in this tutorial, meaning that it matches the shape and layout of the Sci-Fi corridor. The corridors are a bit higher then the ground. If you are using the Sci-Fi set then you would have the floor a bit higher then the ground. This tutorial is for appending, not how to build sets.

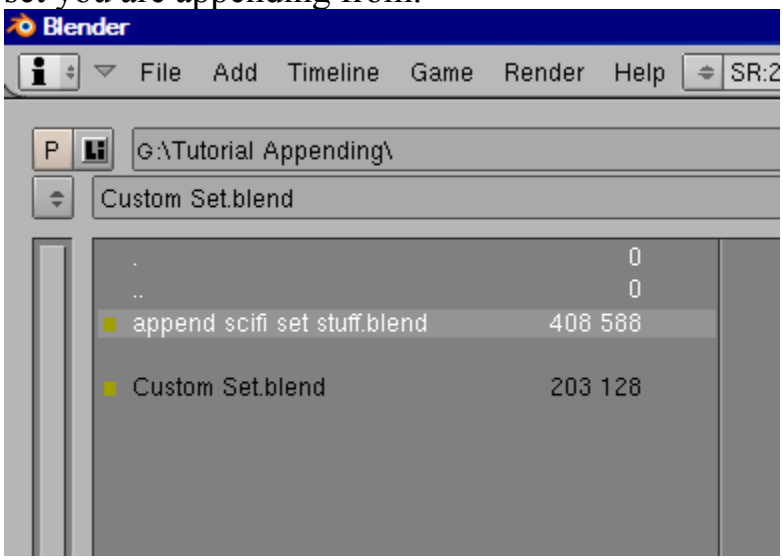
Now refresh Blender and open your custom set's .blend file.



Now go up to **File** and from the drop down menu select **Append or Link.**

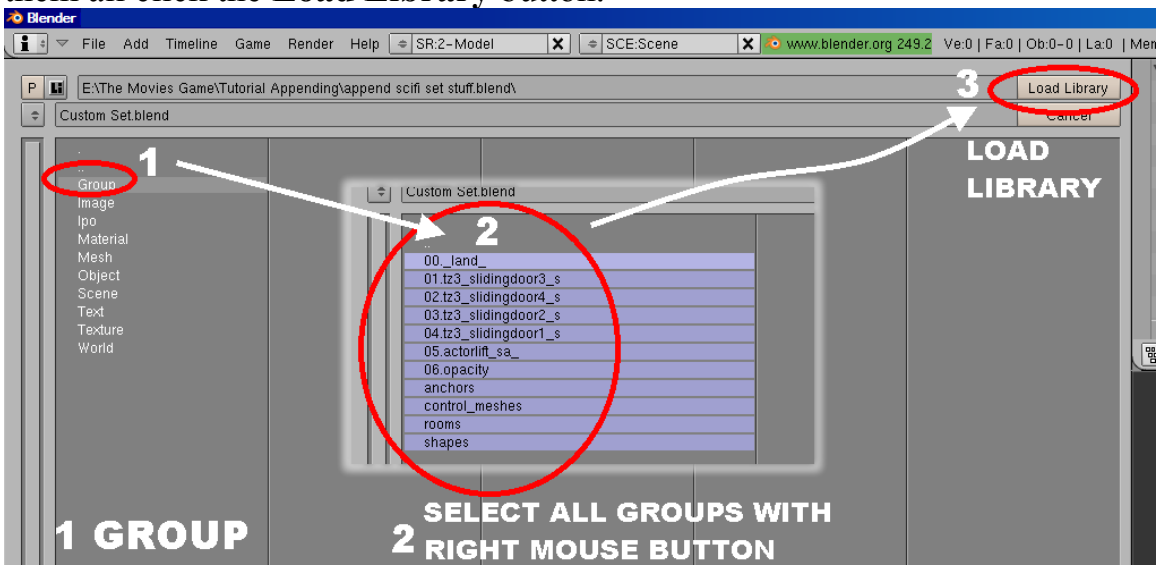


Navigate the browsers to the Sci-Fi set .blend file, or whatever Movies game set you are appending from.

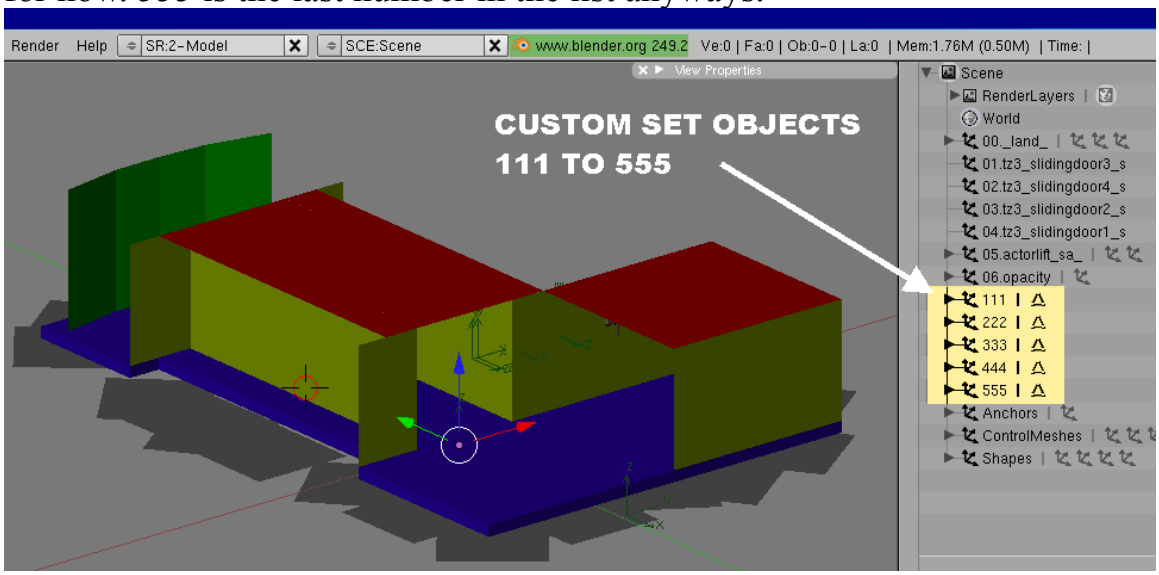


Another menu. Select **Group.** From the next menu select all of the names in the list. These are all of the Blender groups for the Sci-Fi corridor set. There are no more set objects save a few. But all the groups are still there and so

are the remaining objects assigned to these groups. They all will be appended just by selecting the group they are assigned to. After selecting them all click the **Load Library** button.

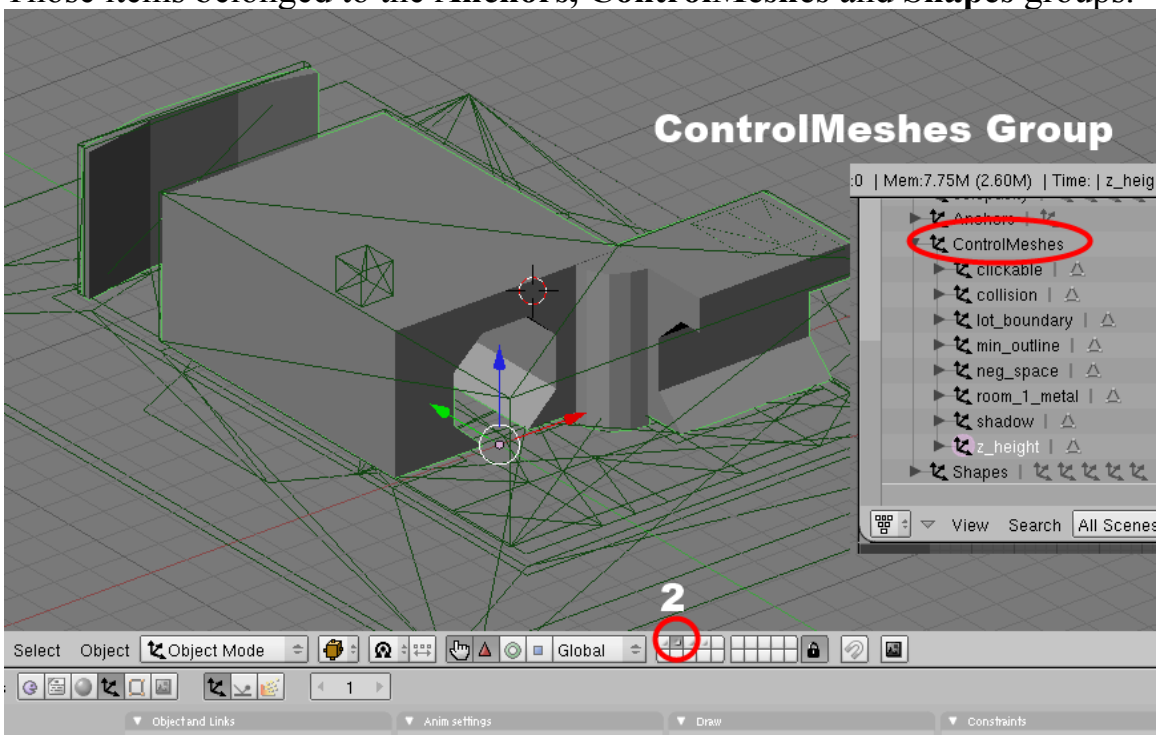


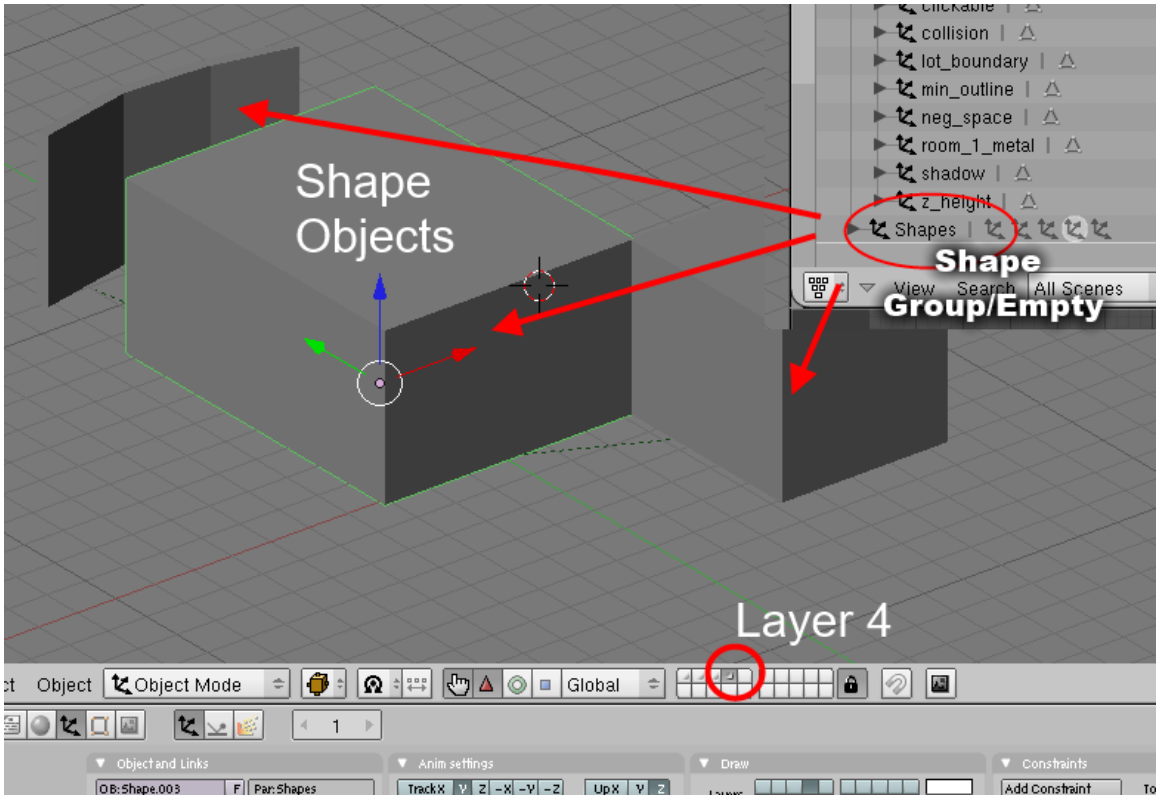
There are 5 objects to my custom set titled 111, 222, 333, 444, 555. All Movies game object names need numbers in them. It can just be a number, too. What the number is in this case is not important. It would be if I had a window or something transparent. Such an object needs to be rendered last in the game during rendering so that the other objects show up on the other side of the window. I have a fire object but it is so small I don't care about it for now. 555 is the last number in the list anyways.



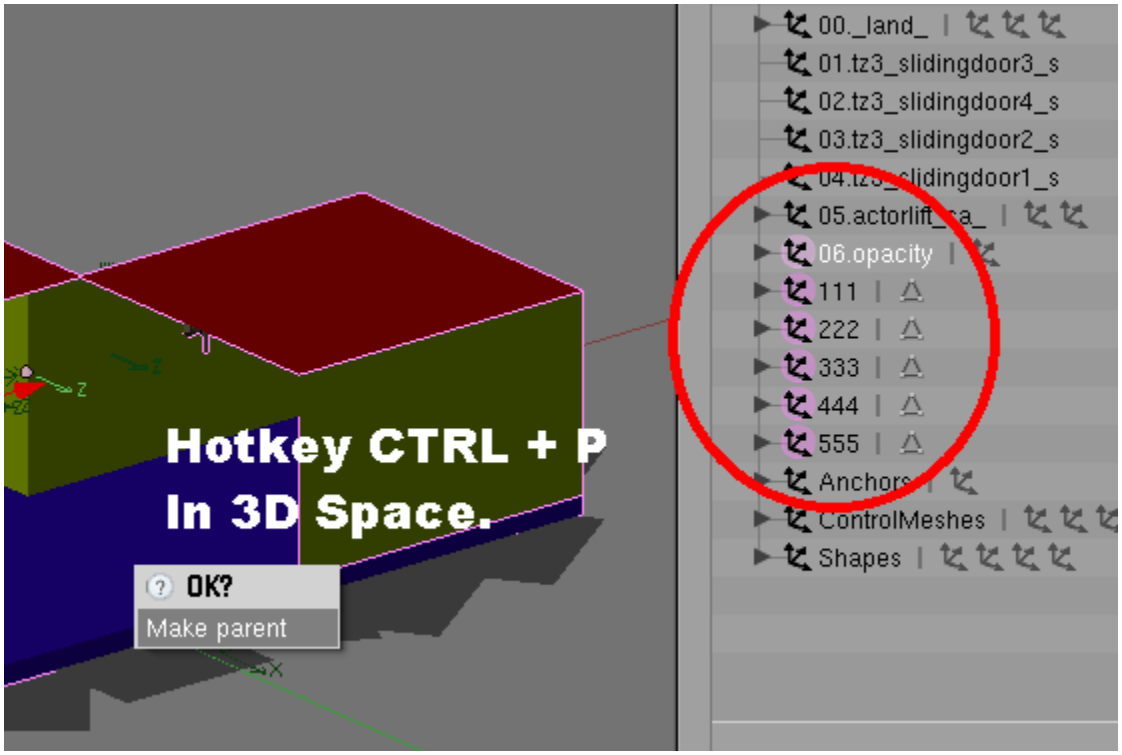
Before the Appending, all I had in this session was the colored objects in the above picture. My custom set. But after the append all of the objects from the Sci-Fi corridor set came with it. Because those objects were assigned to those groups we appended.

You can see the grey squares in the picture. That is the land object which I also kept. They are sand. The sand that shows up around a set when you place it on the studio lot. Those objects were in the **00._Land_** group. And came with the land group when it was appended.
And I definitely needed the objects found in the extra layers of Blender. Those items belonged to the **Anchors, ControlMeshes** and **Shapes** groups.

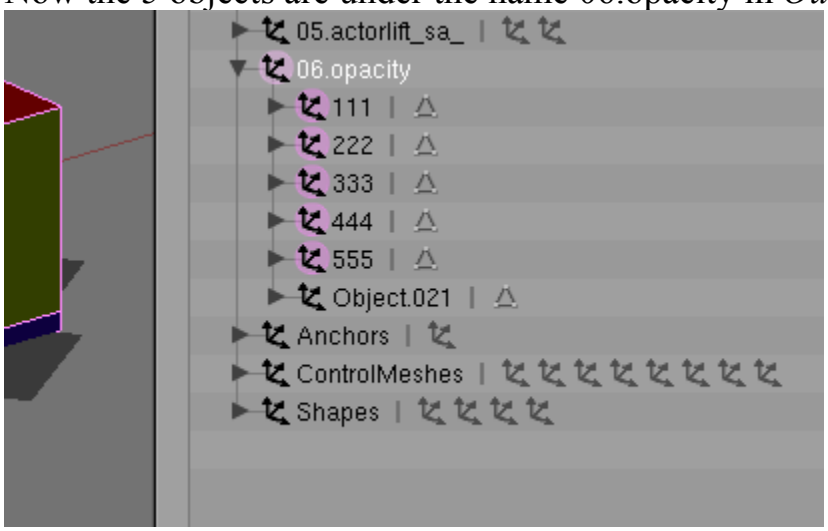




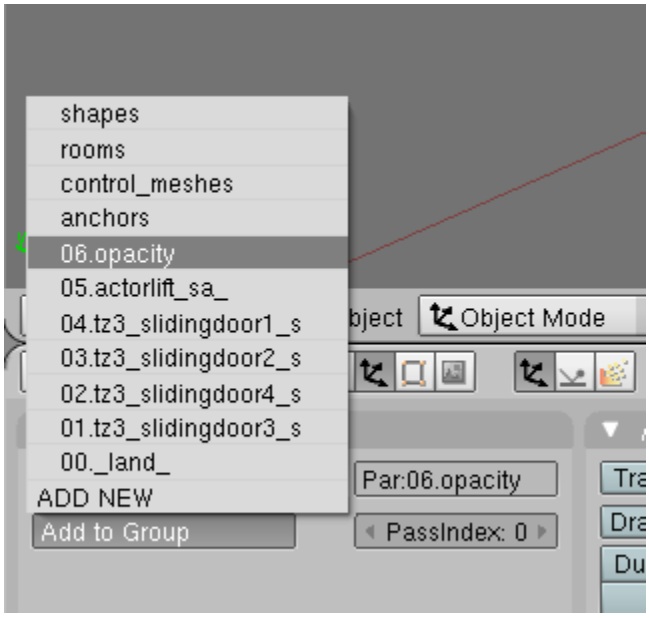
Now that the groups have been appended, I need to *Parent* my own custom set's 5 objects to one of the Empty green arrows thing. I will select all five of them at the same time. Using shift button while clicking allows for more then one item to be selected at the same time. The last item I will select is the green arrows empty object I need my set to be *Parented* to. The same one the set's original back drop is already parented to, which is **06.opacity**. 06.opacity will be selected last using the **shift** key.



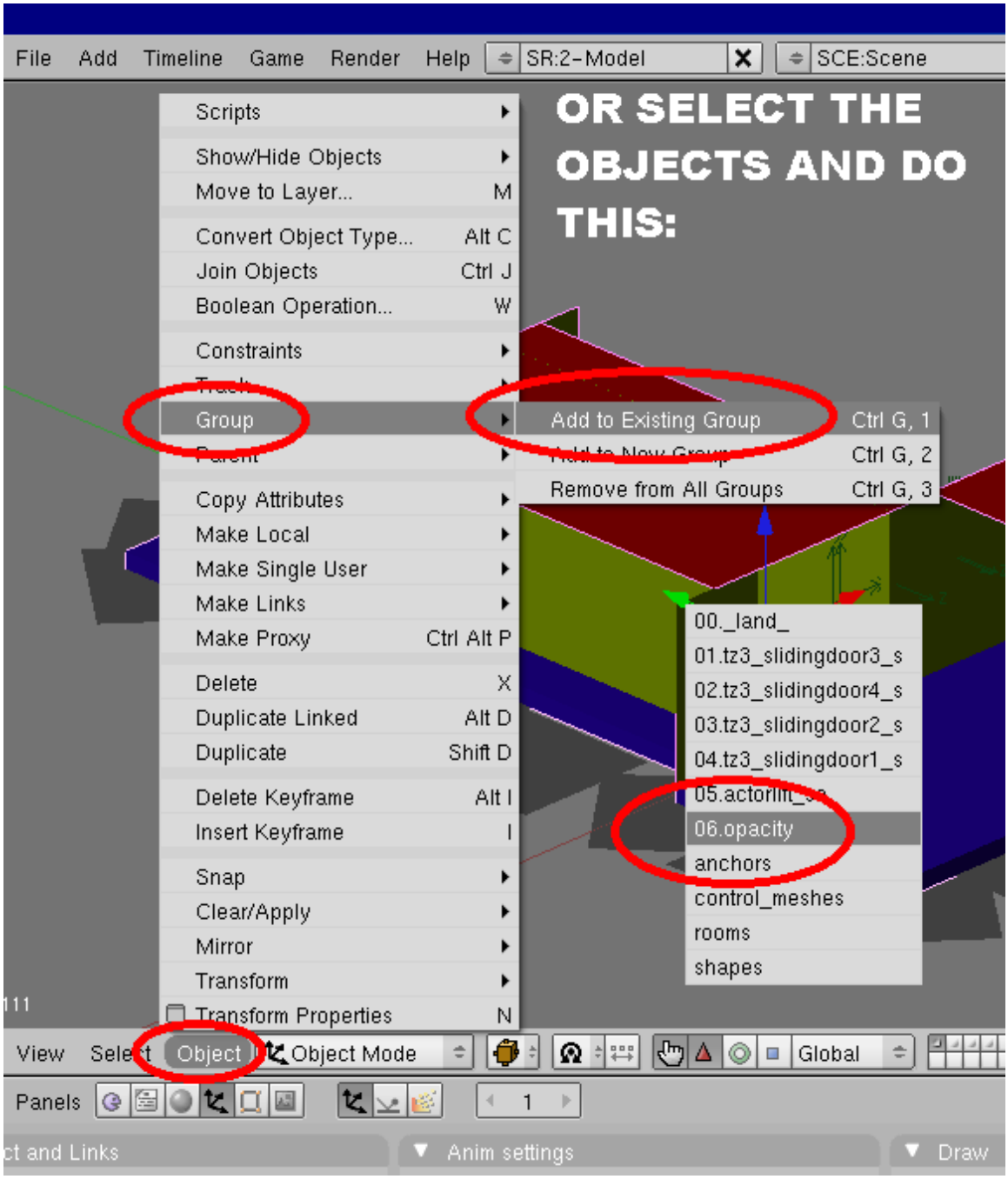
After the selections, in *3D* window press hotkey **CTRL + P** for *make parent*. Now the 5 objects are under the name 06.opacity in *Outliner* window..



The next step is to enter the 5 custom set objects into that *06.opacity group* also. Do so in the buttons window Hotkey **F7**. Click the first object, in this case 111, and down at the bottom click the button **Add To Group**. The group to choose, in this case is **06.opacity**. Do this for all of the objects. 111 - 555.

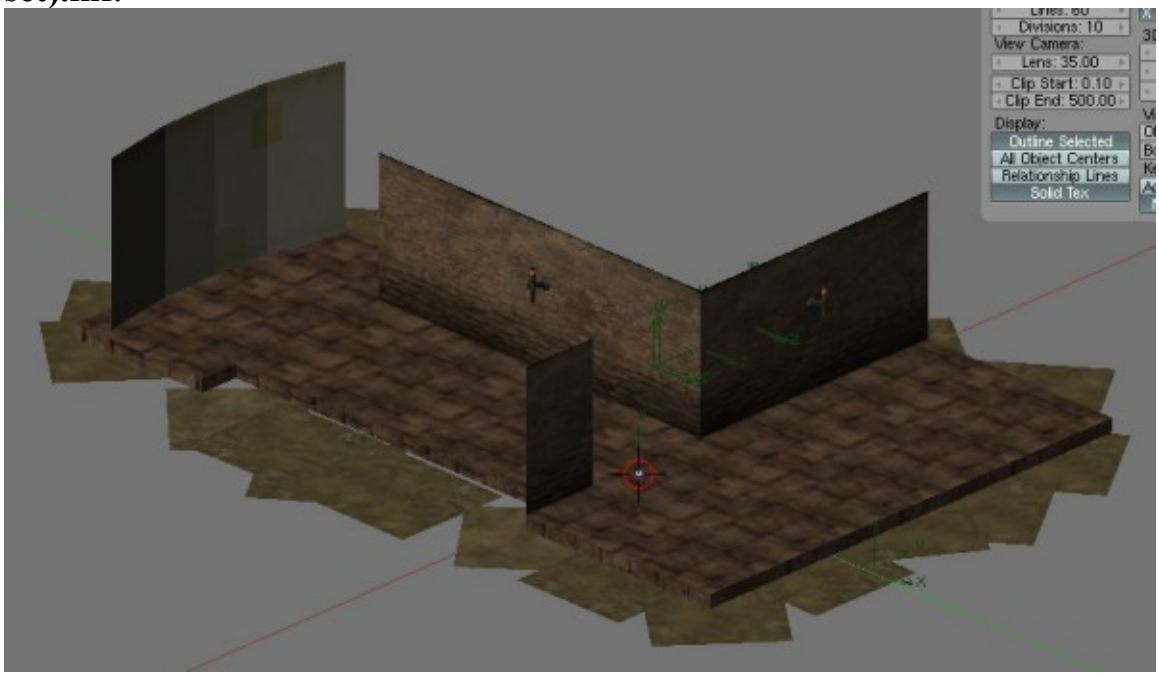


Or... You can do it faster. With a custom set that has 20, 40 or even a hundred objects, you can do it all at once.



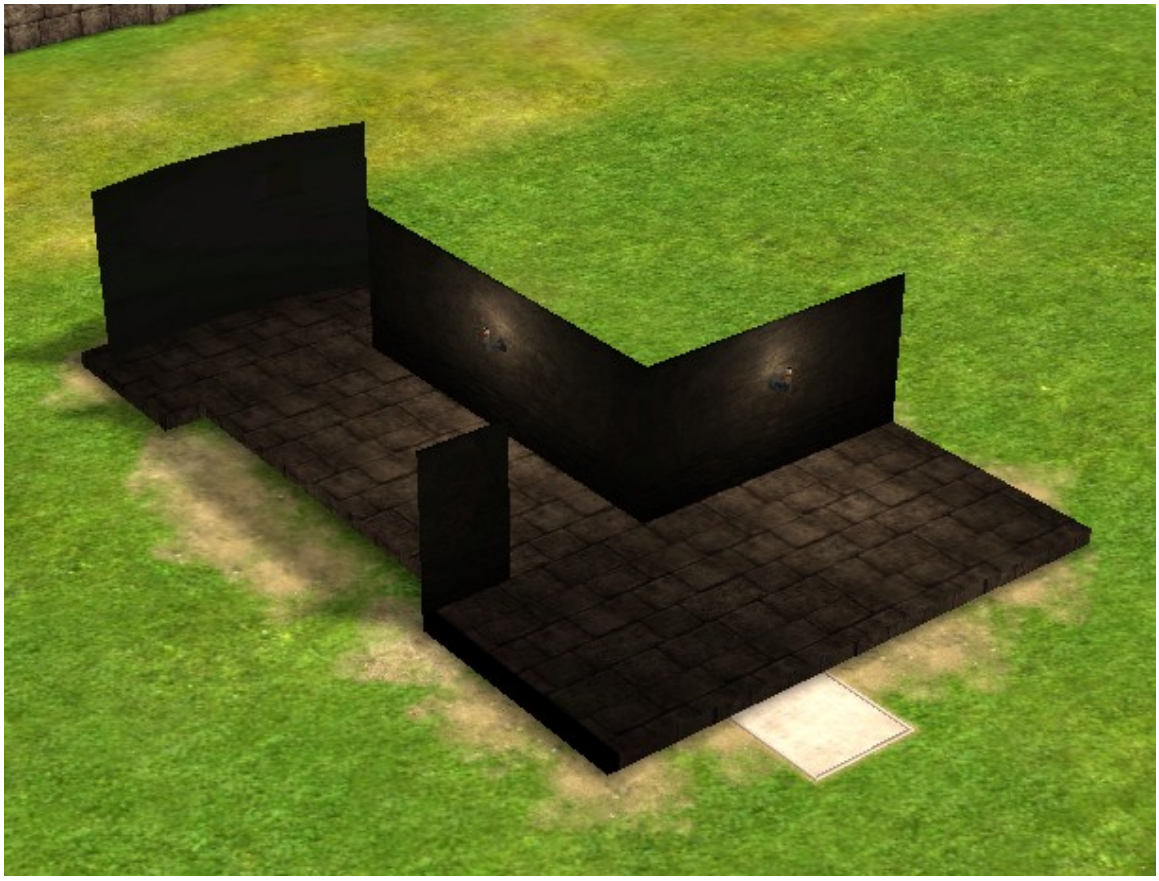
From here I would run **The Movies Game Preflight Script** to see if anything went wrong or still needs to be. Then export it to the game. Mesh files go in the game's data\meshes folder. Using MED I would load an existing Sci-Fi corridor set and replace the mesh file with the set I just made. Change the unique identifier to a new name and commit the new name. Make a thumbnail. Then *commit* the whole thing to the game. Also making sure it has an extrainfo file, which is to copy one of the Sci-Fi set corridor's .INF file found in the data\meshes\extrainfo folder. You would

extract one using MED's file extractor. The set .MSH file and the .INF file must have the same name. Like **set_(name of set).msh** and **set_(name of set).inf**.



Again this is not a tutorial for making sets. It was a tutorial for how to use the append function to **append entire groups**. The benefit for sets is that we can utilize all of a game set's advanced features without having to make them ourselves from scratch.

To summarize:
1) Have your set ready and saved it as it's own **.blend** file
2) Restart Blender
3) Import a Movies game set that closely matches your own, delete unwanted objects
4) Save it as a .blend file, something like "Append from this.blend"
5) Restart Blender
6) Open your custom set's .blend file again
7) **Append** all of the groups from the Movies game set's .blend file that we titled “Append from this.blend”
8) Enter your custom set objects into one of the **groups**
9) **Parent** the custom objects to the **empty** with the same name as the group
10) Run preflight check
10) Export to game



11. Converting A Game Character To The Movies Costume

Do you have a character model from another game and want to make your own movies with it? Or perhaps you have made your own custom character model and want to inject it into The Movies. This tutorial will show you what your model needs to be exported from Blender as a (.MSH) file. And how to inject it into The Movies.

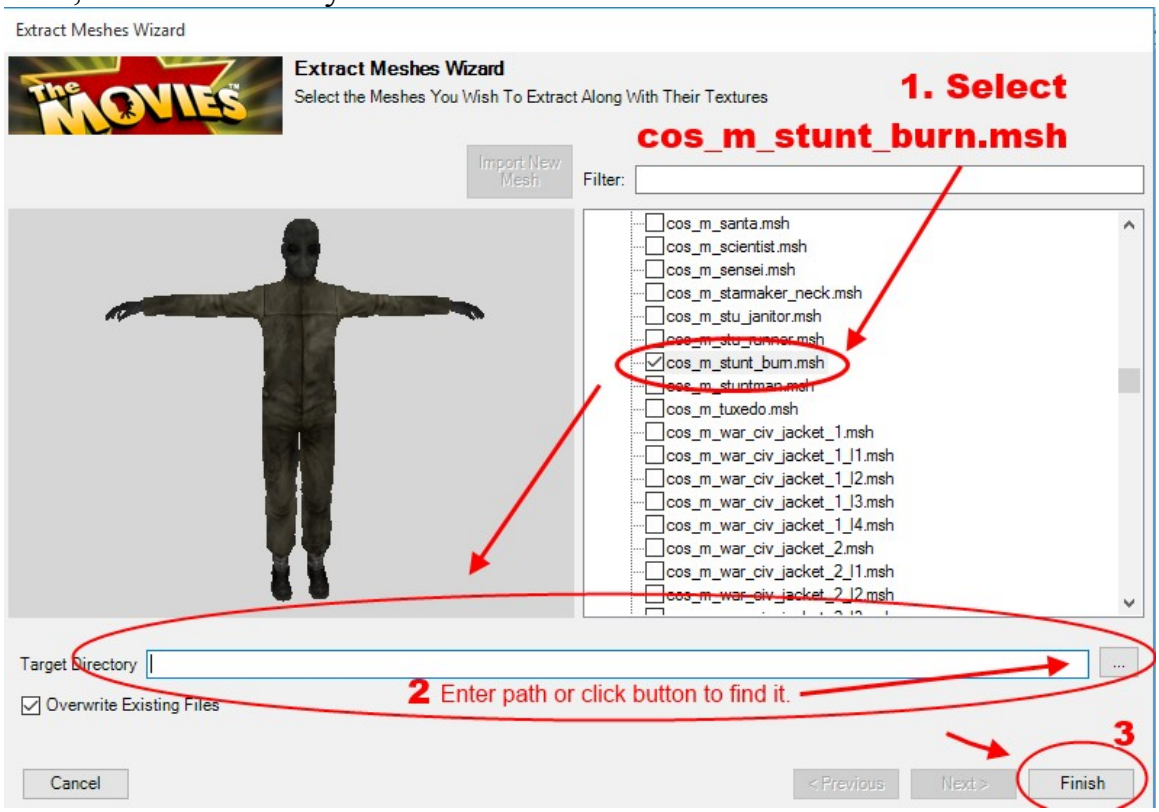
The first thing you will need is a game model. If you have made your own custom model then great, move on to next step. I did a quick Google search for a random model. I finally settled on Han Solo from some Star Wars game. Not sure which but it was in a *T-position* which is great for a beginner. I will use this for the example but will assume you have your own character

model. When you have your game model loaded in Blender save it as a **.blend** file to *append* from later.

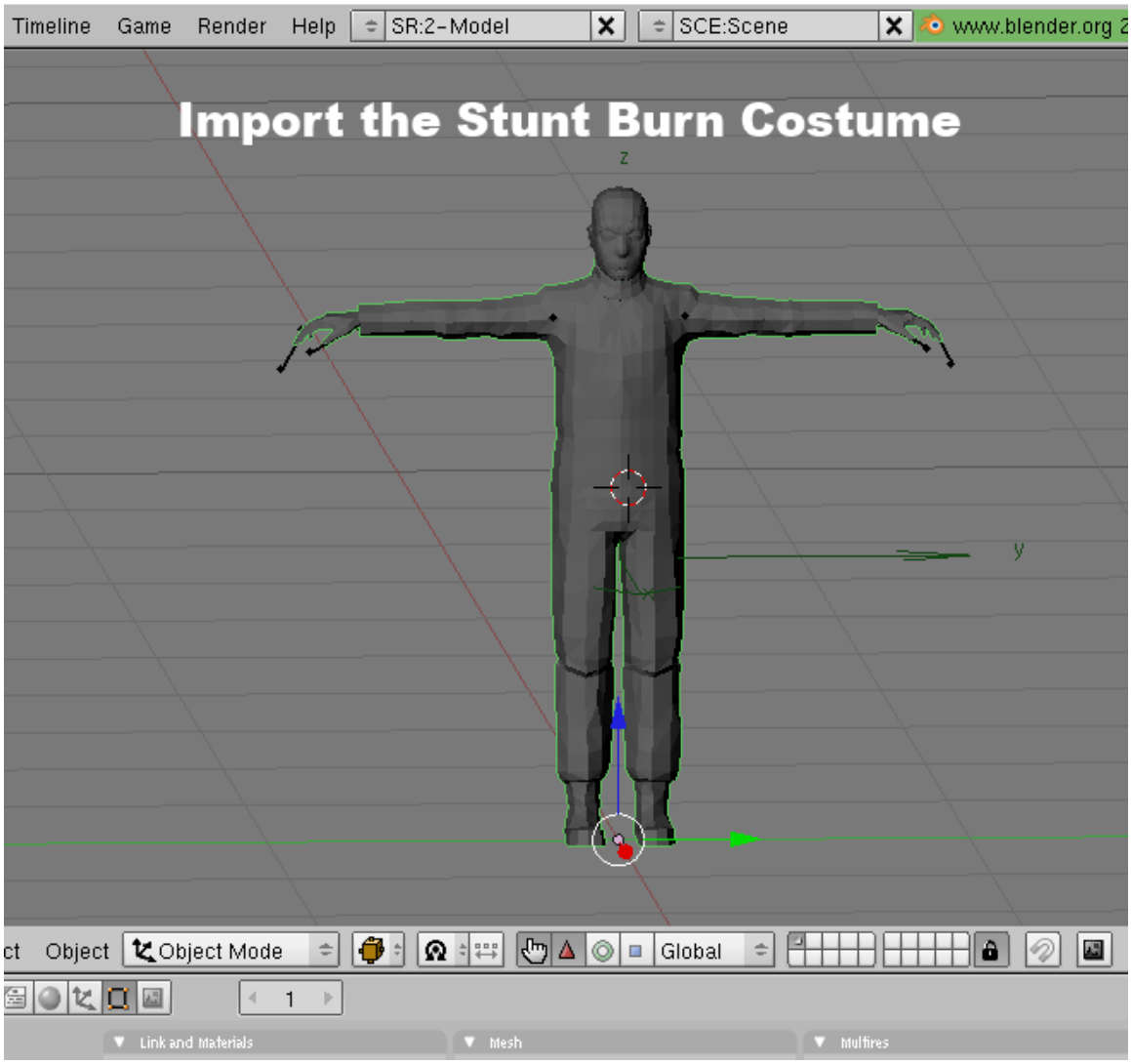
The next thing the model will need is bone weights so that it can move in The Movies. The fastest way is doing a *Bone Weight Transfer*. This is a handy python script that will copy the weights of one model to another. The requirement is that one model has weights already (a Movies costume), and one that does not have any (our game model. I.E. Han Solo). And finally that both objects be right on top of the other. Each obscuring the view of the other. Ensuring all the minute bits line up. Like fingers and elbows. Chins and groins. Making sure every limb and area is aligned is the most important part.

After a *Bone Weight Transfer* you can delete the Movies costume because it is no longer needed. But since it was imported, we can barrow it's *Empty*, it's *armature* and *blend group*.

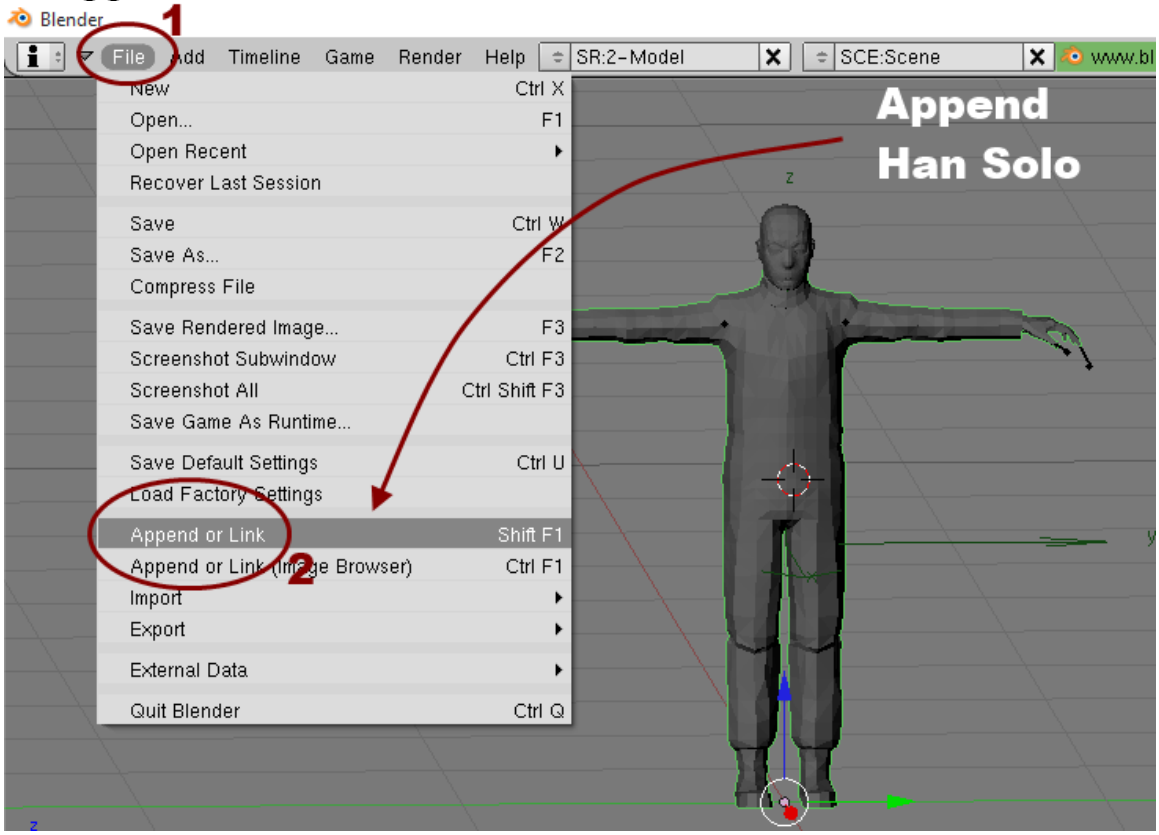
1. Use MED (The Movies Editor) to extract a costume that also has a head. I chose the stunt burn costume that came with Stunts & Effects expansion pack. If you do not have S&E then find a costume that does, or extract a body and head model.



2. Import the burn costume into Blender.

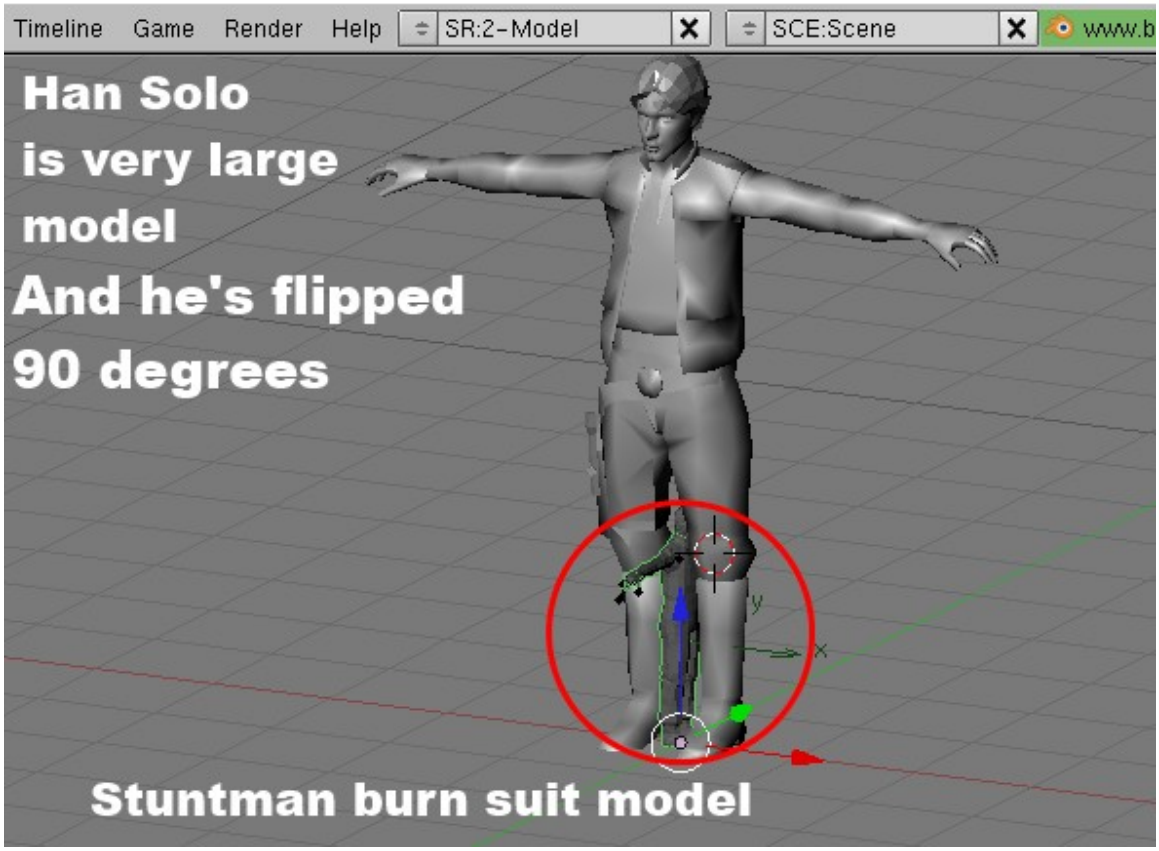


3. Append Your Game Model.



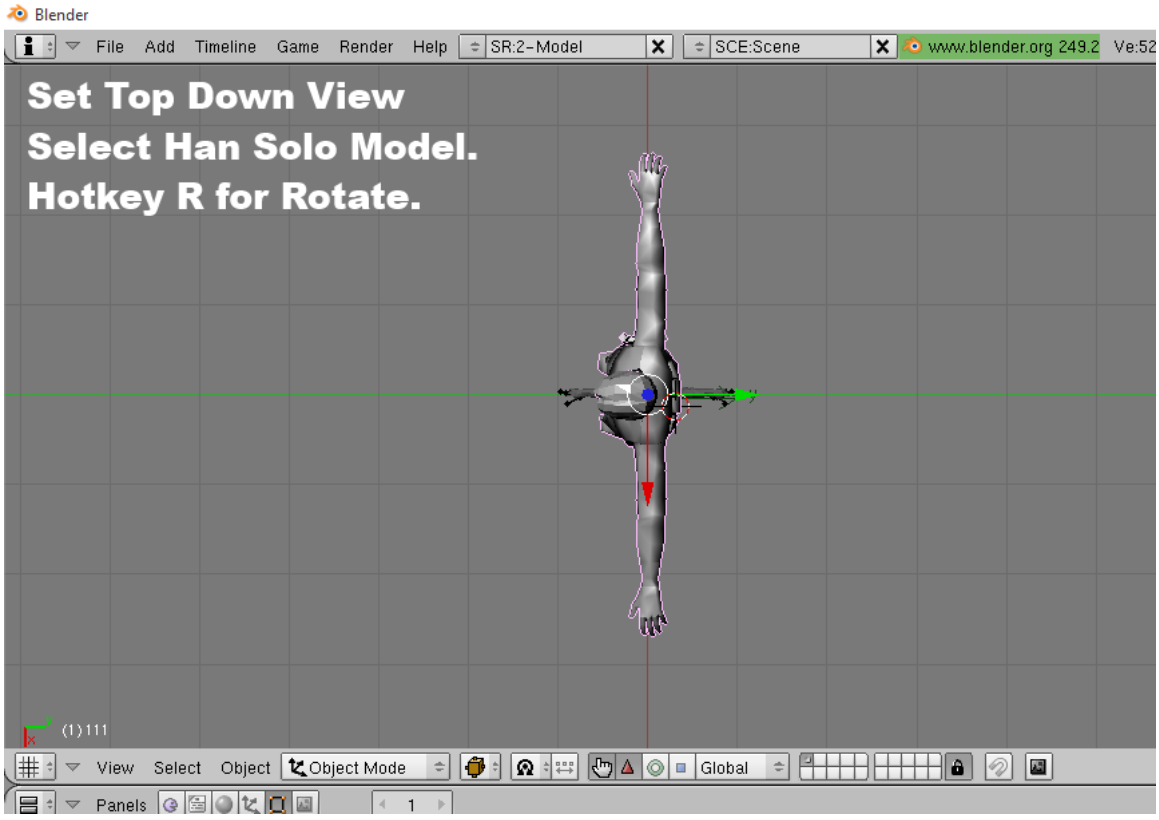
4. Scale and Rotate the model

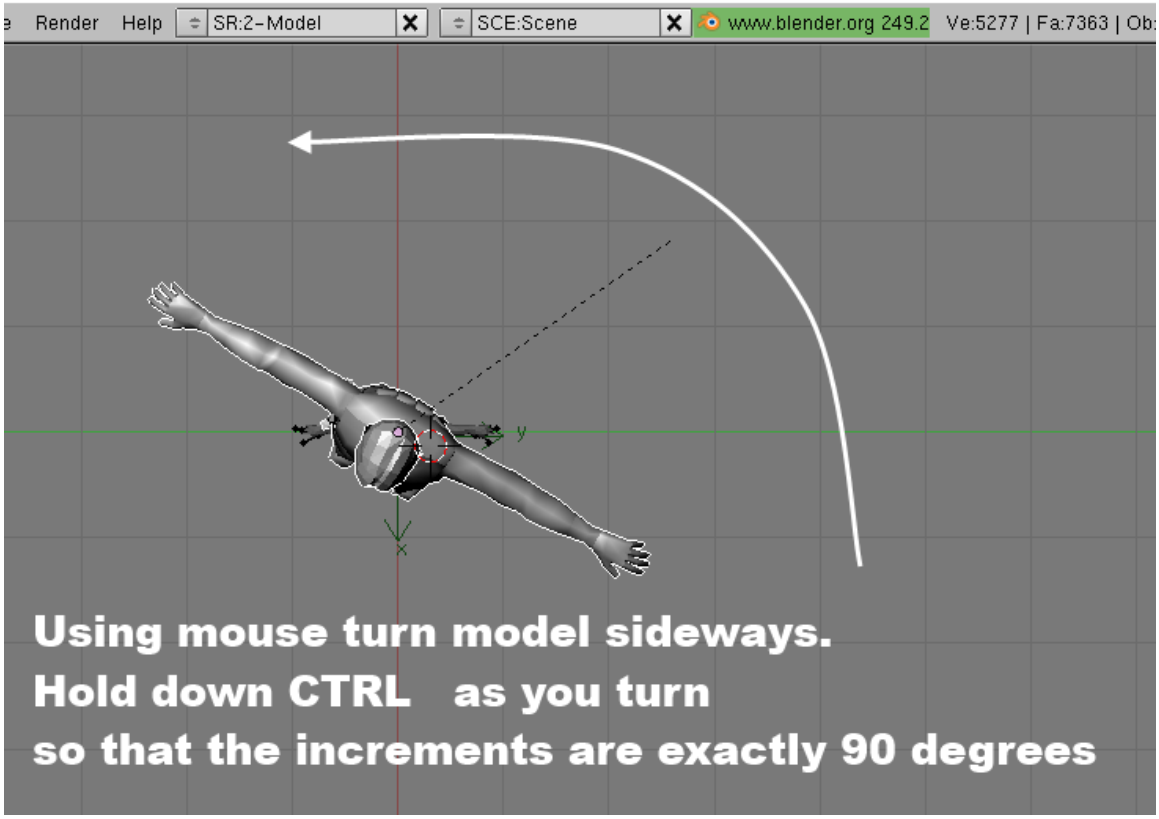
Your game model might be too large or too small. It may also be flipped on an axis. Take a bit of time to *rotate* it and *scale* it to match the size and location of the burn costume object.

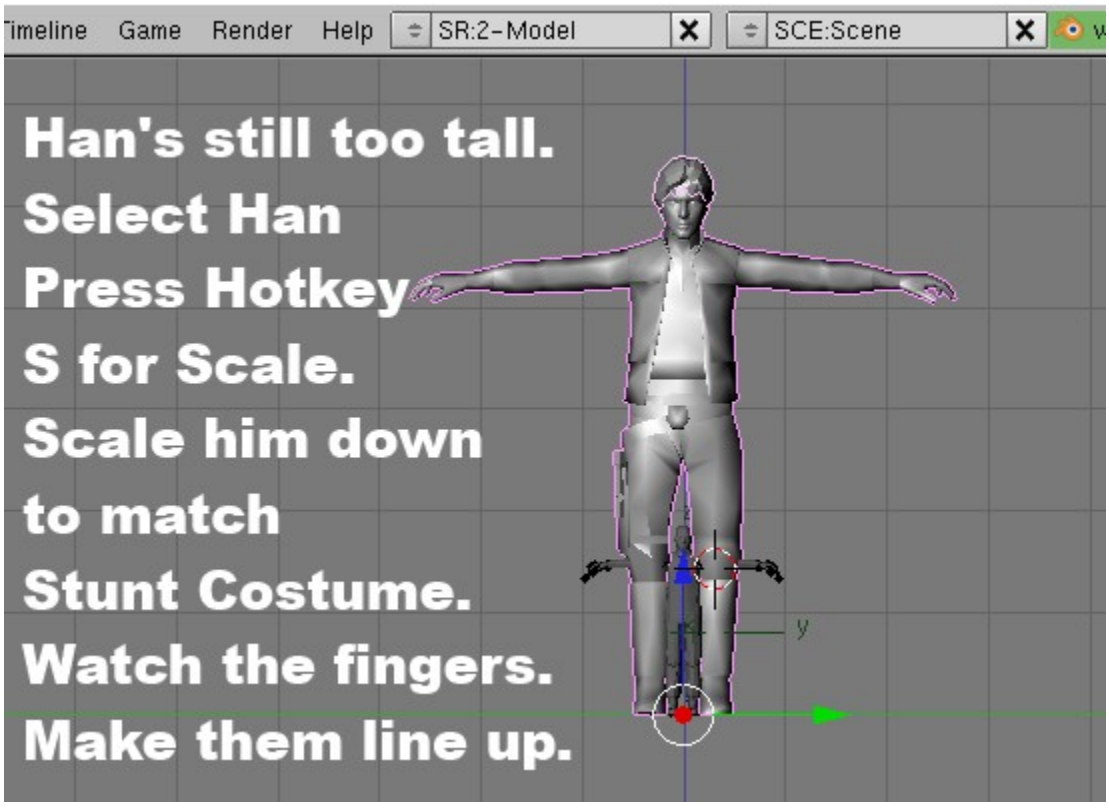


Here Han Solo is facing the wrong direction. Near the bottom you can see the burn costume's arm poking out. We need Han to face the correct direction. The best way for me is to view the scene from the top down. There is a Hotkey to achieve this, number **7** on the NumPad. But I just used the mouse and **CTRL** key to snap it into a perfect downward view.

Once your are looking down select your model (assuming it has the same problem) and in 3D space press Hotkey **R** for *rotate*. While rotating if you hold down **CTRL** it will turn in increments. And do so until a perfect 90 degree turn is achieved.

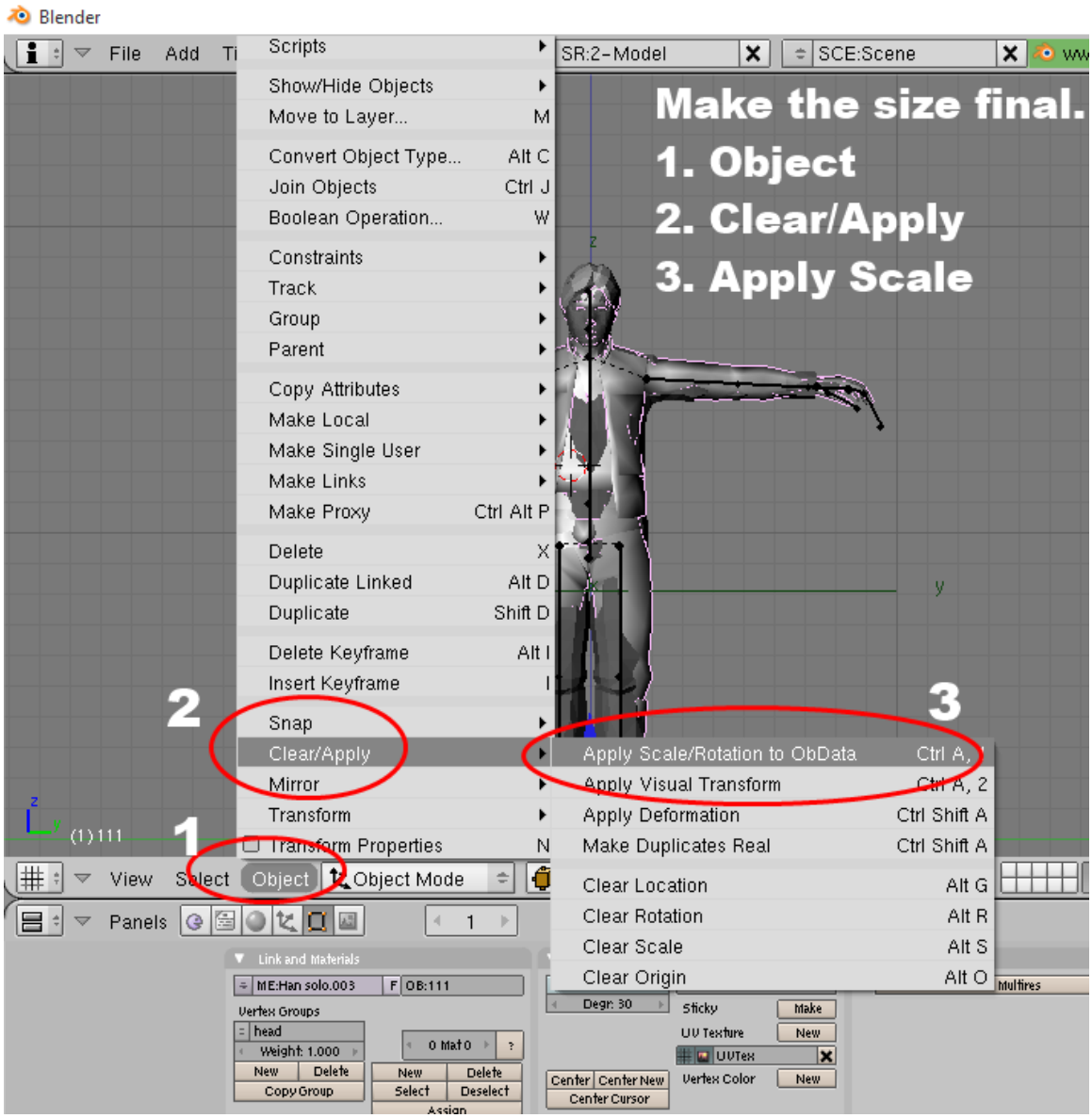

Blender
File Add Timeline Game Render Help
SR:2-Model
SCE:Scene
www.blender.org 249.2
Set Top Down View
Select Han Solo Model.
Hotkey R for Rotate.
View Select Object
Object Mode
Global
Panels


Render Help
SR:2-Model
SCE:Scene
www.blender.org 249.2
Ve:5277 | Fa:7363 | Ob
Using mouse turn model sideways.
Hold down CTRL as you turn
so that the increments are exactly 90 degrees



We have to ensure that both costumes occupy the exact same space in 3D window. Both heads from each model need be exactly aligned. Same with the fingers, knees, elbows and everything. Sometimes the model we choose does not line up exactly in some parts. Maybe the neck is too long compared to the other. Or the hands are farther out then a Movies costume. I got lucky picking Han Solo and everything seems to line up exactly.

With game model selected press hotkey **S** for *scale*. Move mouse in or out to change model's size. Here I *scale* it down to the same size as the stunt burn costume.

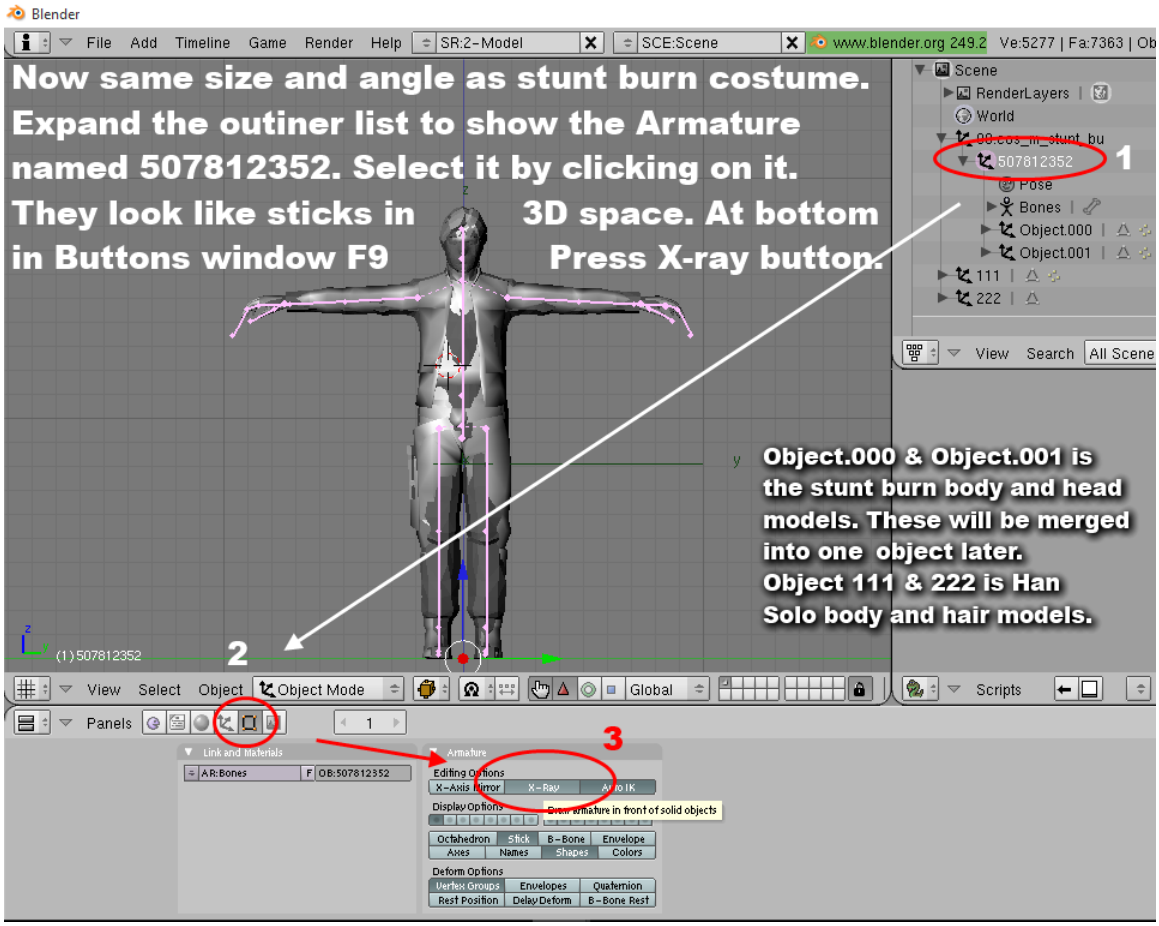


Now that the two objects are even in size and angle we need to tell Blender to make it's size official or permanent. Blender can forget what we just did otherwise. This can be done in Object Mode. Click the Object button. Clear/ Apply. And Apply scale/Rotation to ObData.

5. Turn on X-ray button for the Armature

In the outliner window we can expand the items listed to reveal their child objects. The Empty object called 00.cos_m_stunt_bu possesses the armature. The armature object called 507812352 possesses the stunt costume objects, the body and head. The armature is inside the costume so it makes it hard to see in 3D space. To view the armature inside the costumes we can click the *X-ray* button. First select the armature object called 507812352. Then down

in Buttons window (**F9**) find and click a button called **X-ray**.

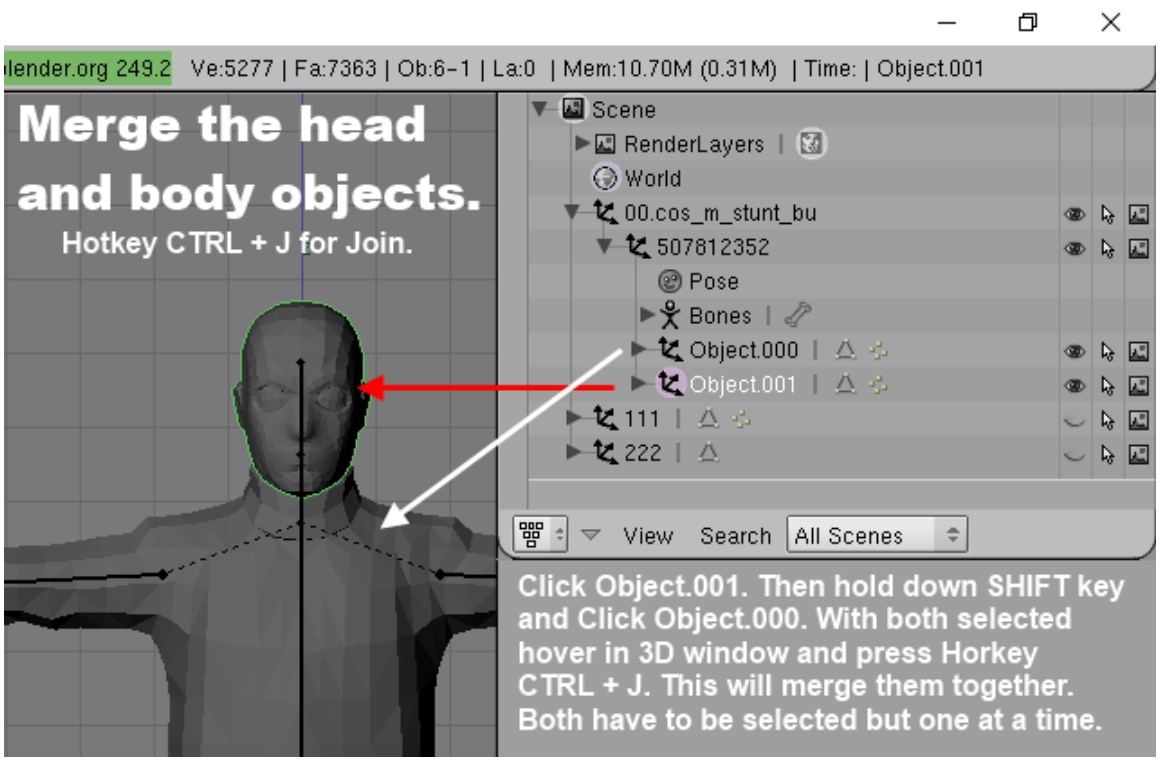


Seeing the Armature will make the coming steps easier.

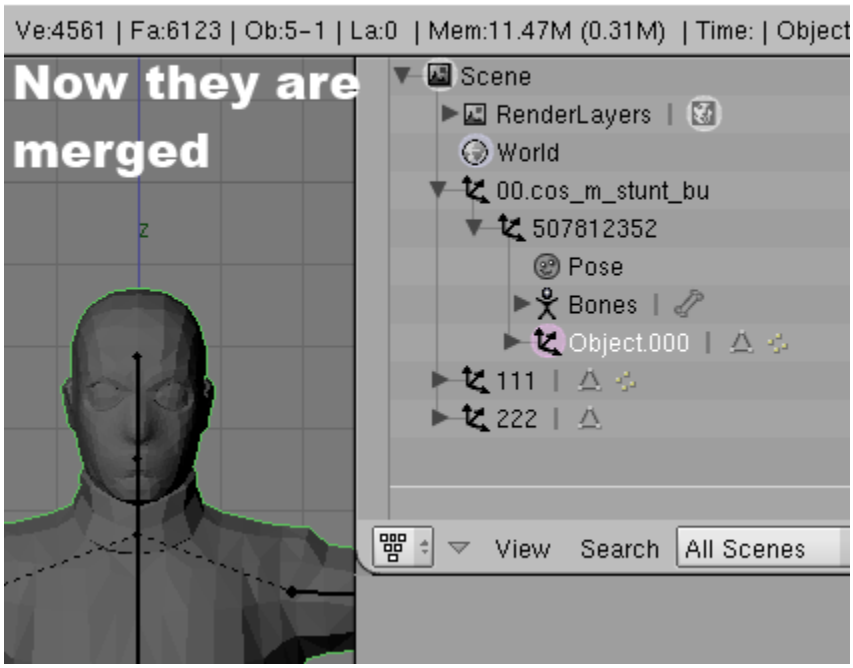
6. Merge the weight source object's Head & Body

Soon we will be doing the *Bone Weight Transfer*. The object we want to get weights from must have a head. This is because the game model we are converting also has a head. If it didn't then our bone weight source wouldn't need it. But we do in this example.

The stunt burn costume imported with the head and body as separate objects. But need only one object to get weights. One that has a head. So we have to merge the stunt burn costume's head and body. Being one object will allow us to steal the weights from head to toe.



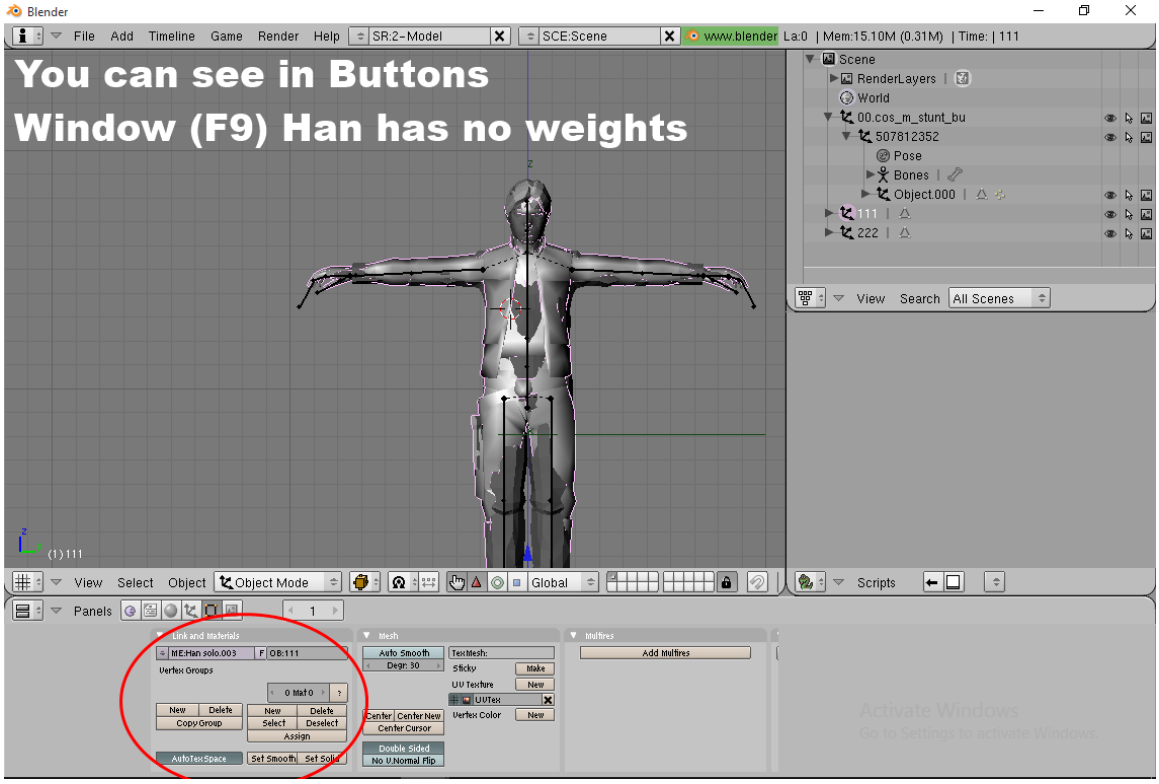
When *Joining* the last object remains and the first object merges with it. To join the 2 stunt burn objects select Object.001 first. Then hold down **SHIFT** and select Object.000. In 3D space press Hotkey **CTRL + P** for *Make Parent*. Object.001 will disappear in the Outliner window. But will also have been added to Object.000.



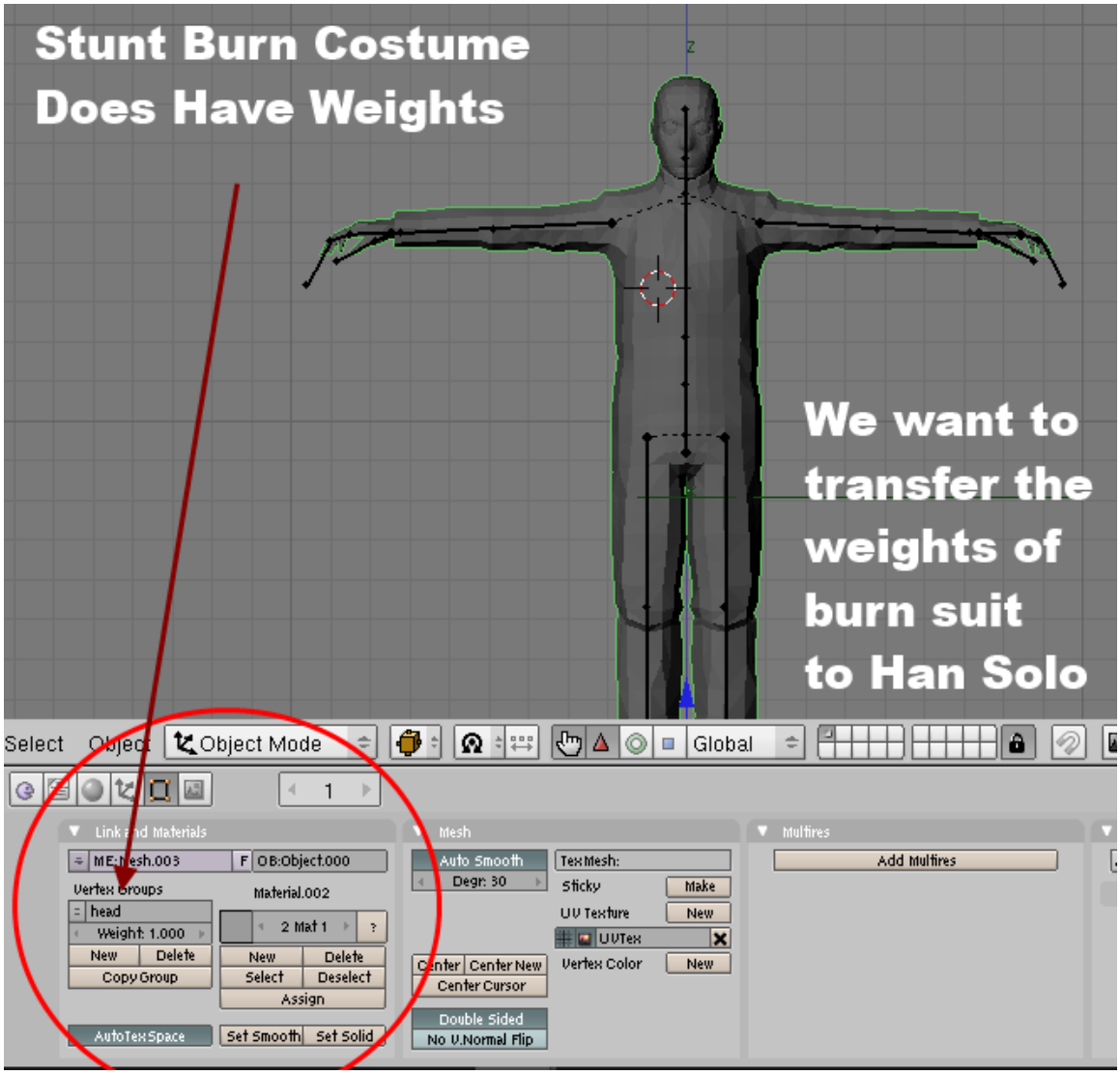
7. Do a *Bone Weight Transfer*

We now have one object with bone weights. And we also have an object without weights which is our game model. Han Solo in this example. To view if an object has weights or not we can see them in the Buttons window (**F9**).

Click on your game model first and then press Hotkey **F9**. In the Buttons window at the far left we see the area under *Vertex Groups* is blank. No weights.

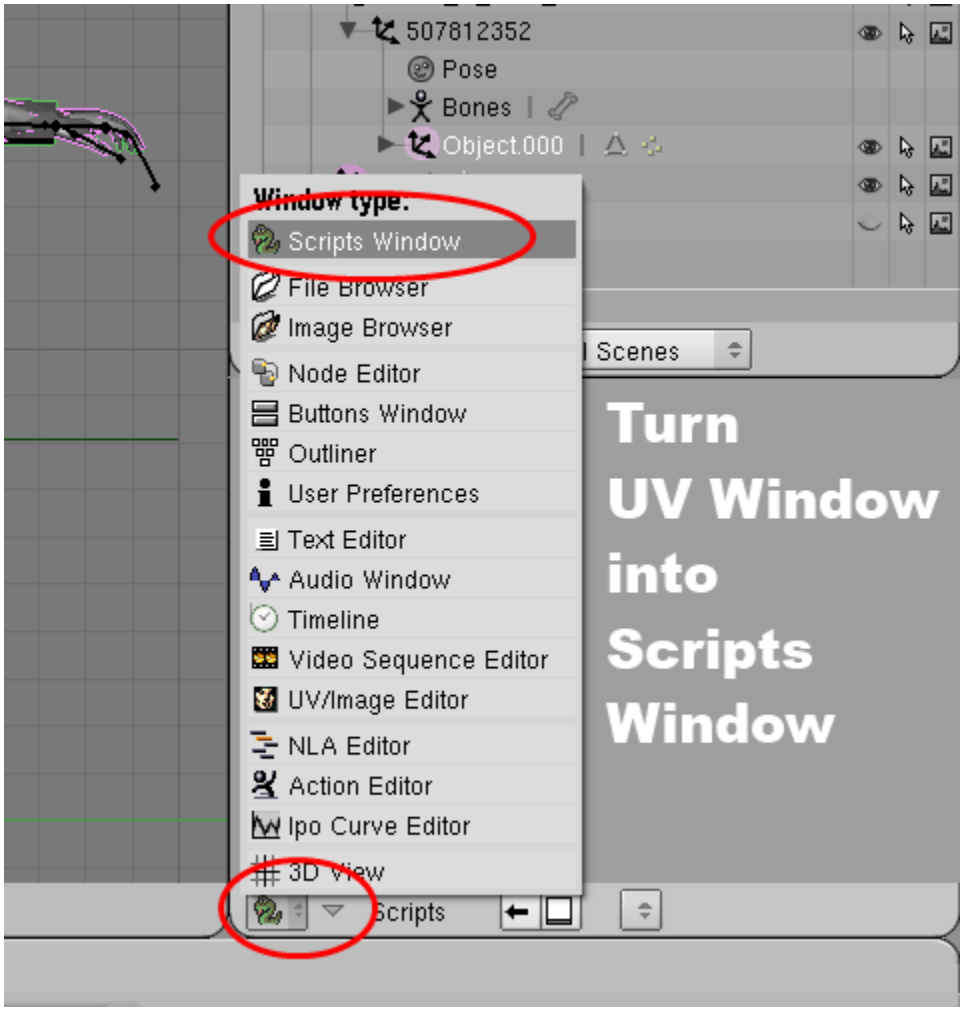


On the other hand go ahead and click on the burn costume which is Object.000 and do the same. Look at it in the Buttons window.

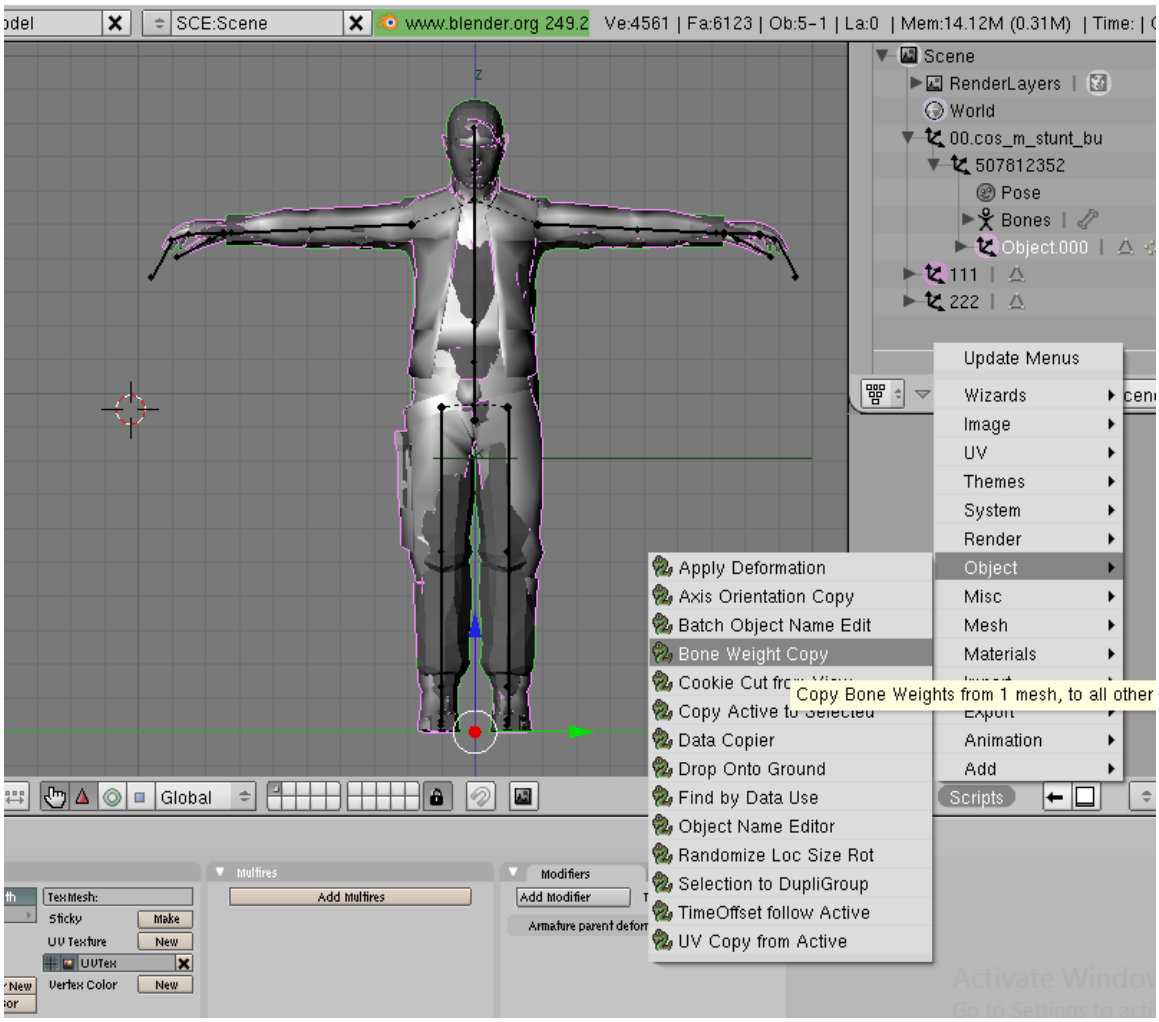


Ok the 3 conditions have been met.
1. We have a model with no bone weights.
2. We have a model that does have bone weights.
3. Both models occupy the same space.

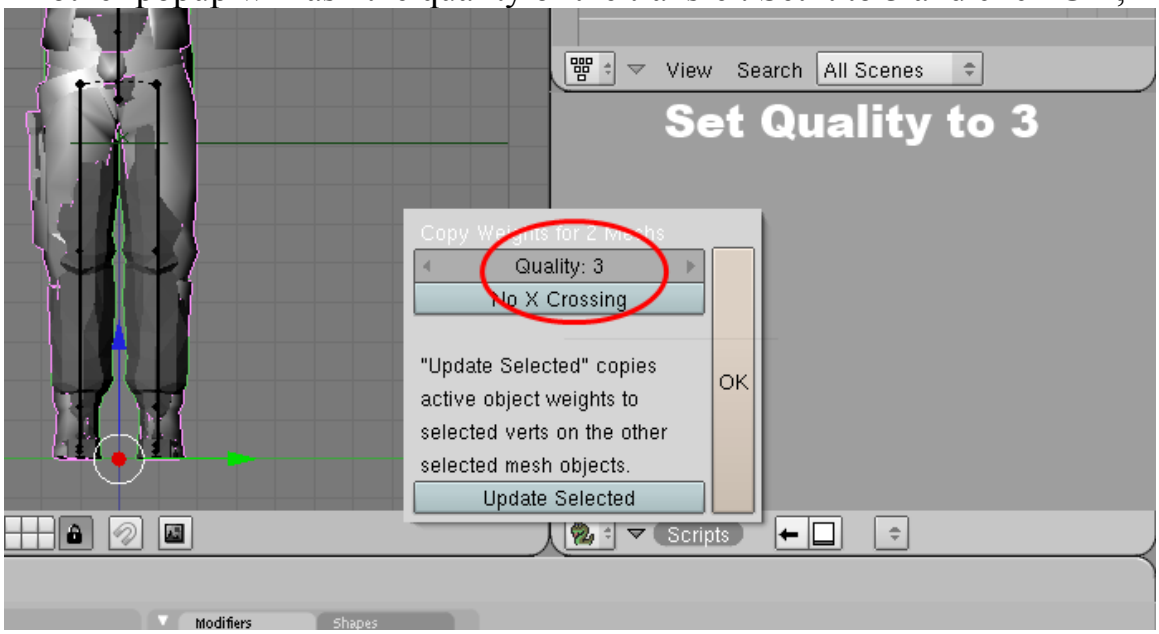
The *Bone Weight Transfer* is launched from the Scripts window. Let's turn the UV Edit window into a Scripts Window.



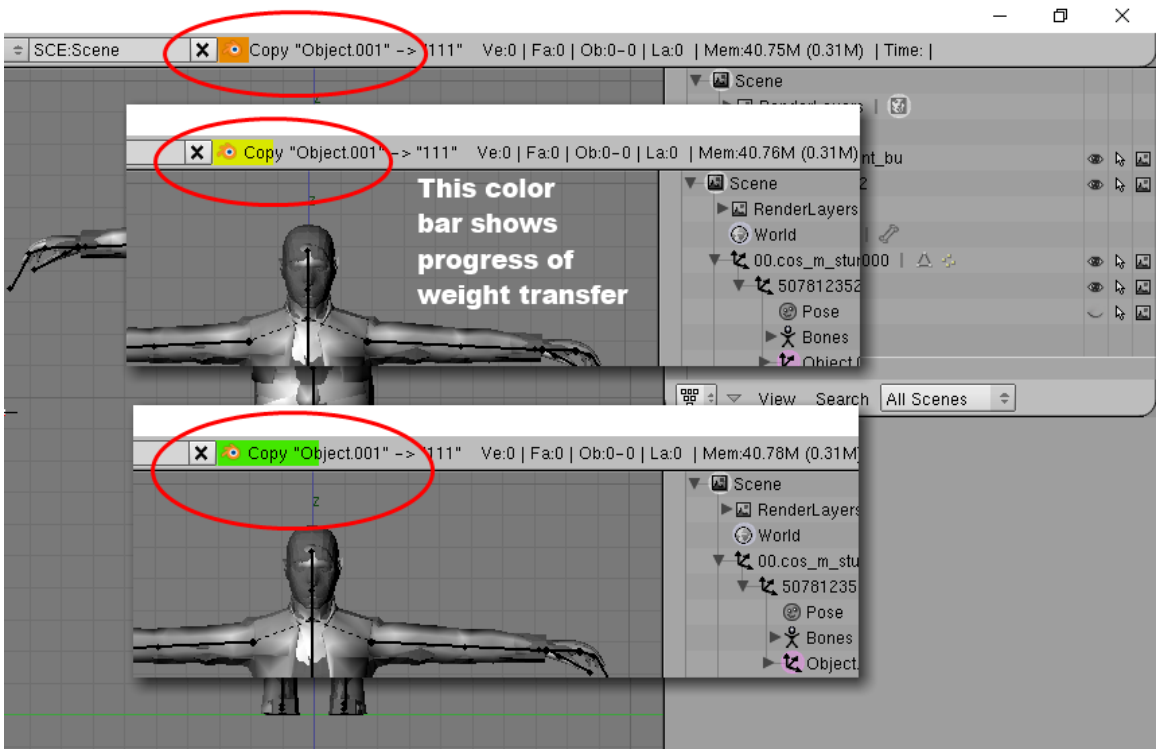
There is a button or tab called **Scripts**. Click it to reveal a popup menu. Select **Object** from the list. Then select **Bone Weight Copy**.



Another popup will ask the quality of the transfer. Set it to **3** and click **OK**,



Now it starts. It may take some time depending how many verts your game model has. You will know by a progress bar at the top changes color and gets larger. Don't click anything and wait for Blender to complete the transfer.

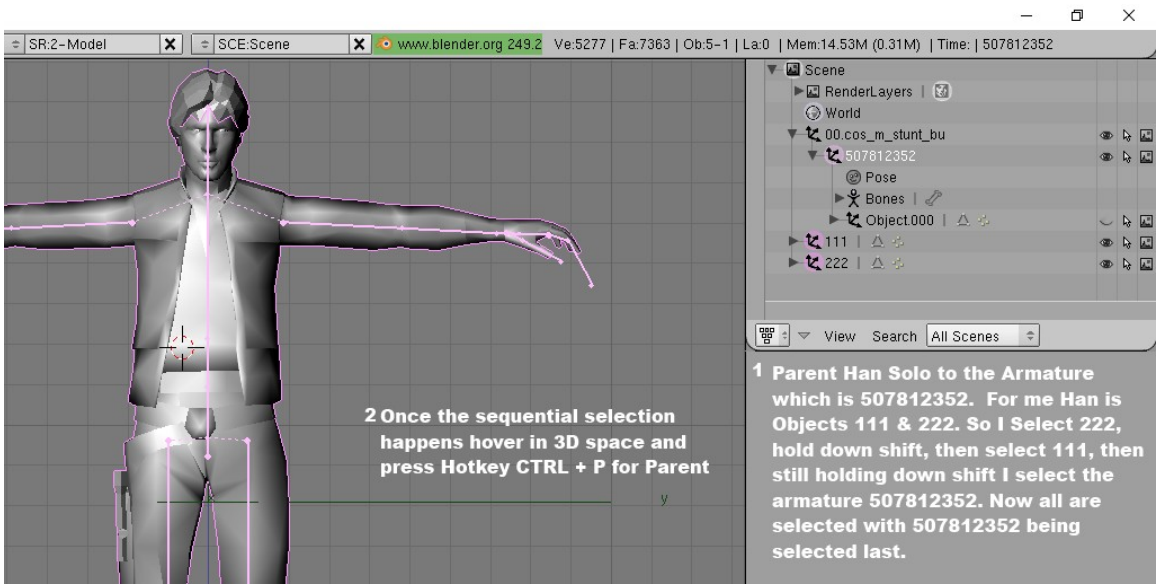


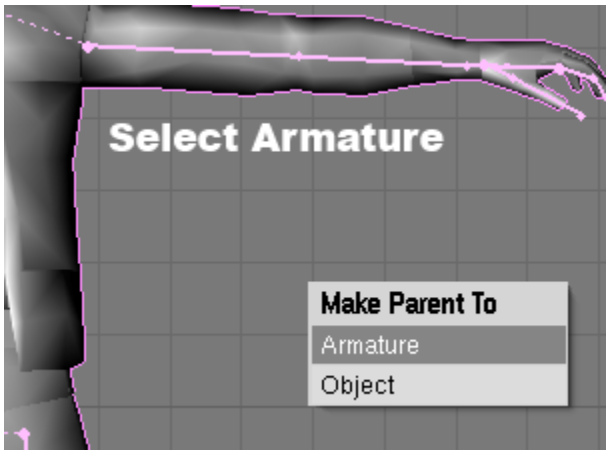
8. Parent your game model to the Armature.

After the *Bone Weight Transfer* we need to test how the transfer went. We cannot test this unless your game model can be influence by the bones or the armature.

In the meantime I turned off the visibility of the burn costume object which in the Outliner is Object.000. I could delete it but if there was something we missed or did in error we could do the transfer again. So turn its visibility off for now. You can turn an item's visibility off in the outliner window. There is a little eyeball next to the name of each item listed there. Clicking on it hides the object from view. You could also just press Hotkey **H** for *Hide* if the stunt burn item is the one selected. Don't hide your game model.

Now to parent your game model to the armature. As we did when merging two objects, we need to select the game model and the armature at the same time and in sequence. Click on your game model. I will click on Han Solo first. Then hold down **SHIFT** and click on the armature last. Then in 3D space press Hotkey **CTRL + P** for *Parent*. If you have more then one game model object as I do, you can click one, then another and lastly the armature.

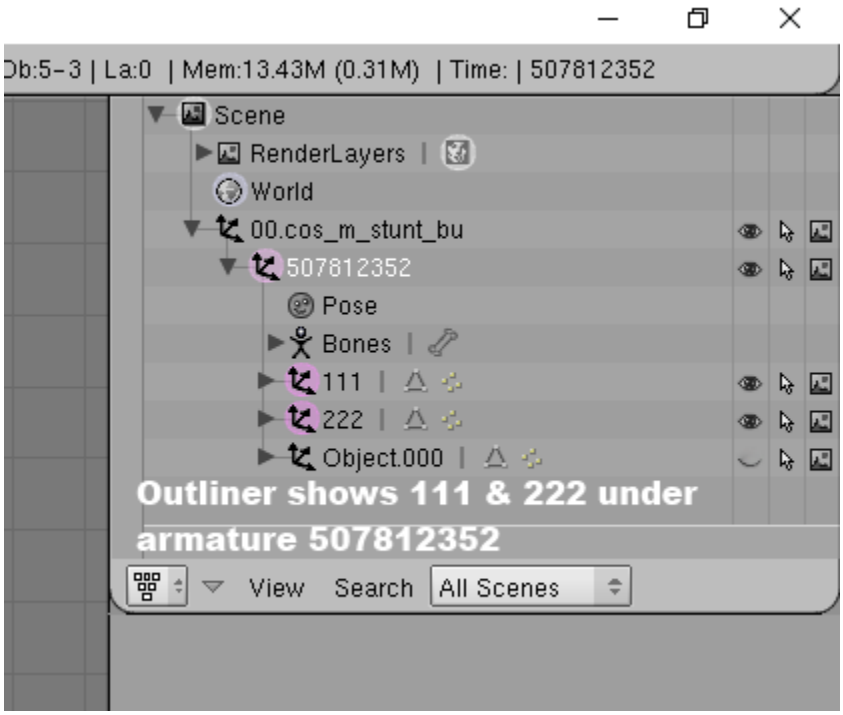




Hotkey CTLR + P>Armature>Don’t Create Groups

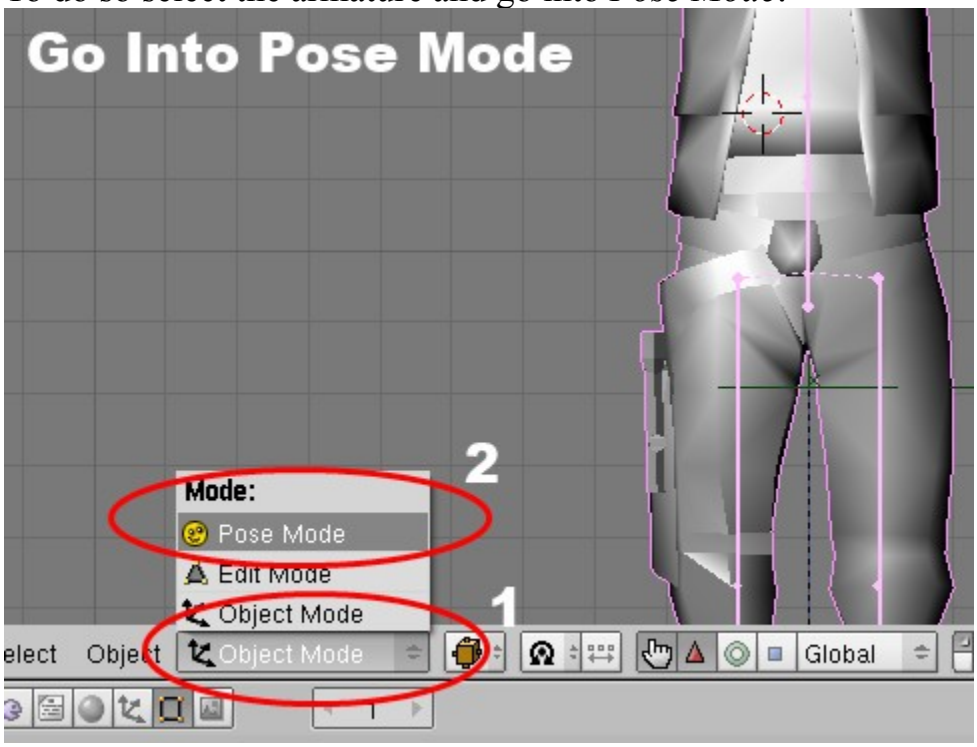
We just transferred the weights so no need to let Blender create weights

arbitrarily.



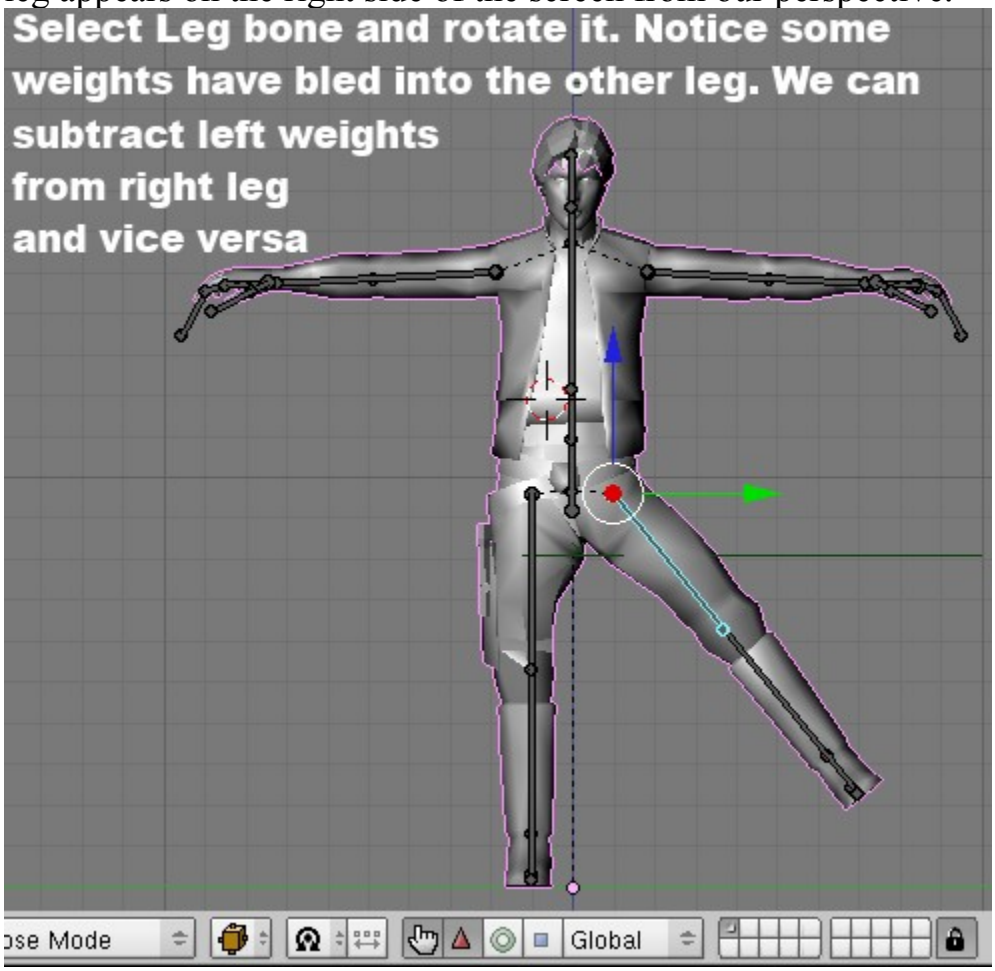
9. Test the game object's bone weights

To do so select the armature and go into *Pose Mode*.



In *Pose Mode* the bones change color. They are now selectable individually.

The things to look for if one limb was very close to another, like the legs, then some of the weights from those bones may bleed over onto the other or nearer item. Select the left leg bone. Since the costume is facing us the left leg appears on the right side of the screen from our perspective.

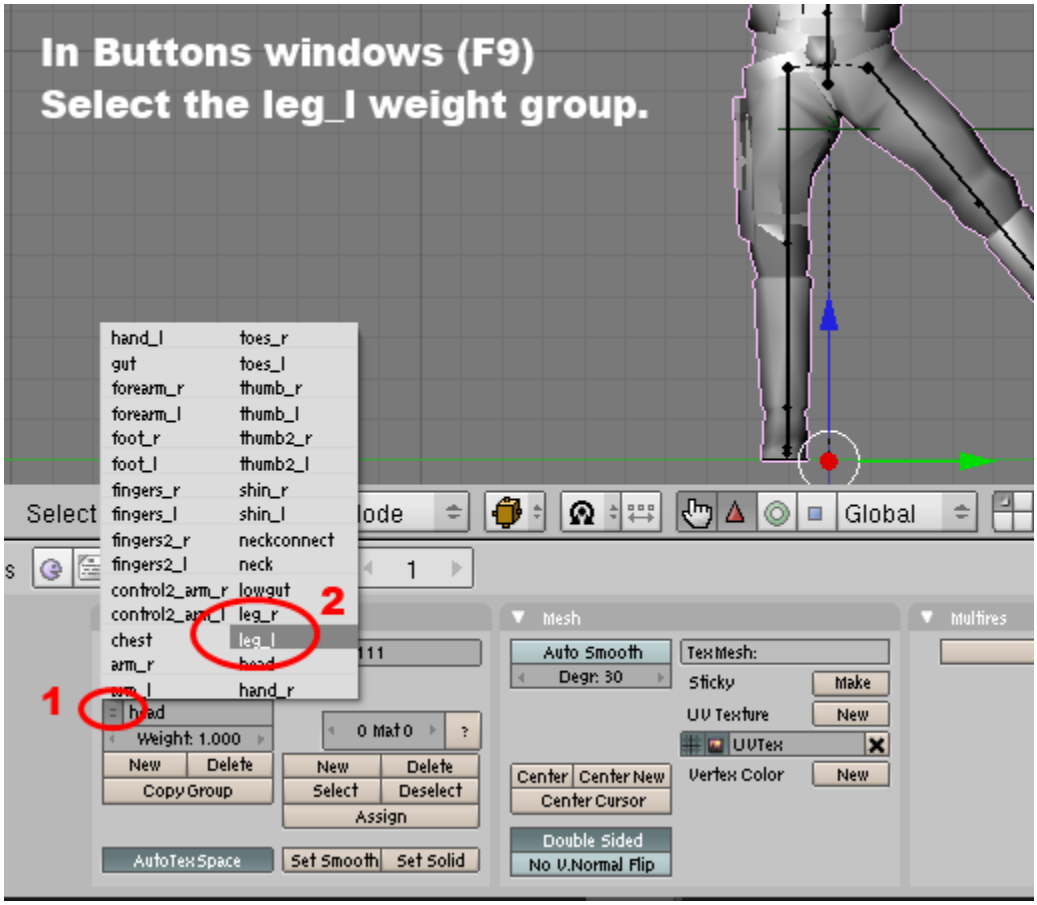


After selecting a bone, press **R** for *Rotate*. Move it out to see if any part of the other legs moves as well. Near the groin we can see that a little bit has. The legs will have to be fixed.

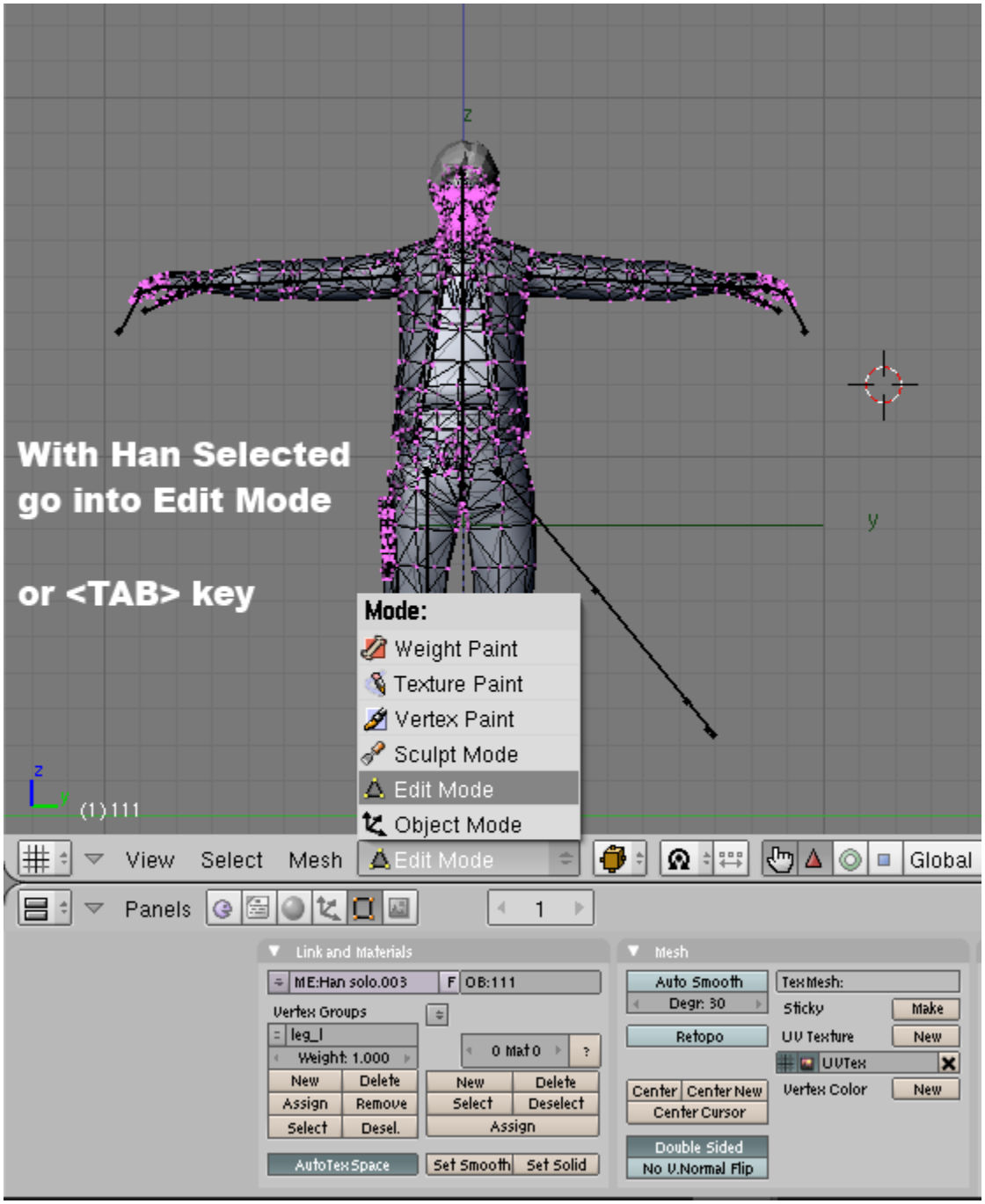
10. Fix the weights

There are two ways to see the weights on the model. And each way allows you to fix them. You can use the Buttons window or Blender's *Weight Paint Mode*.

Select your game object. Then in Buttons window (**F9**) select the left leg weight group which is titled **l_leg**.

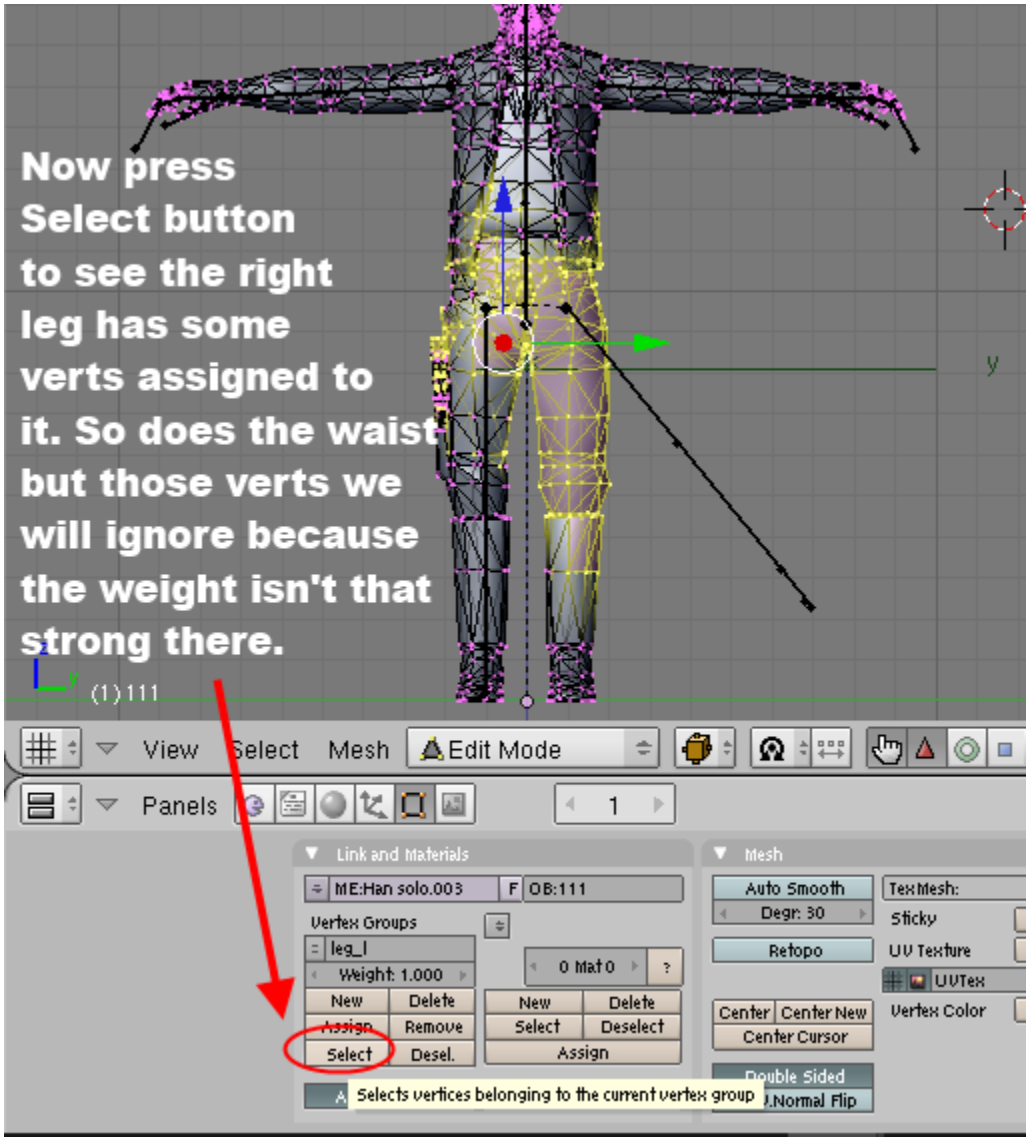


Now go into *Edit Mode*.



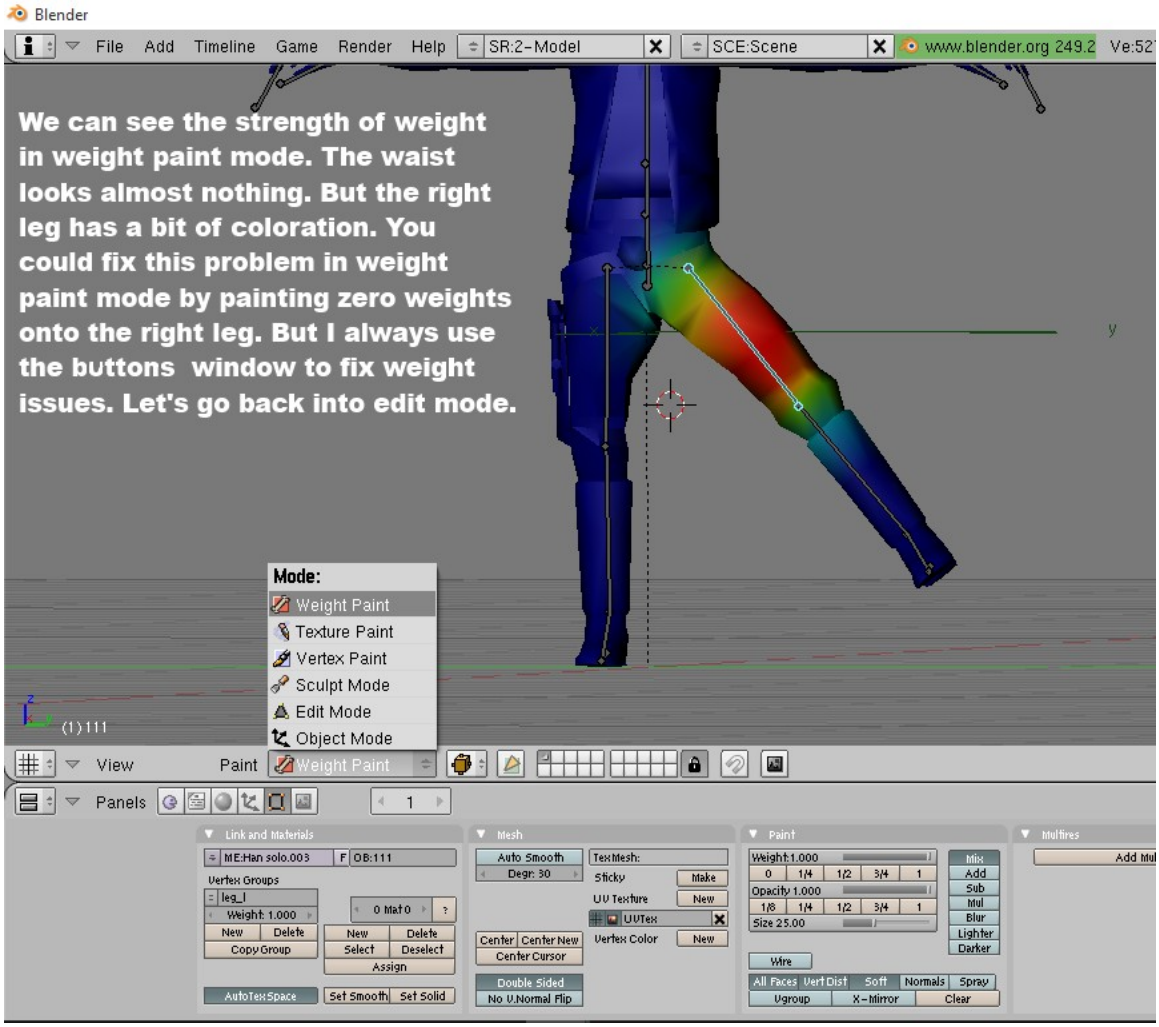
In Edit Mode we see the model's verts. *Weight Groups* are assigned to a select number of them. All in different grades of intensity or percentages. We can't view the percentages in *Edit Mode*. But it is a conveniently fast way to correct some errors.

Once we are in *Edit Mode*, and in buttons window we have selected the **leg_l** group, go ahead and click the button called **Select.** This will select the verts that are assigned to the **leg_l** group.



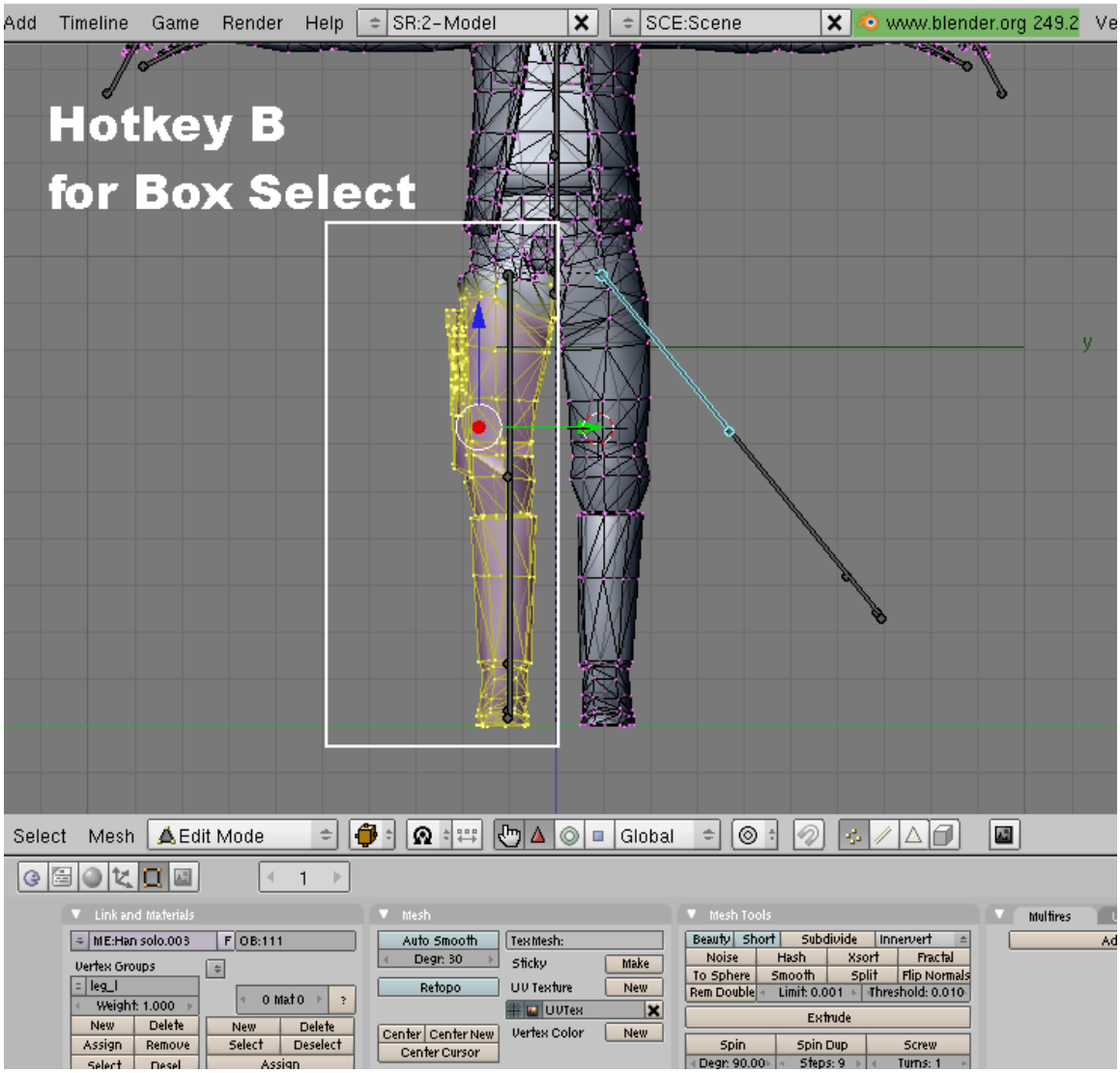
Take much of what you see with a grain of salt. Even though a good portion of the costume's waist appears to have a heck of a lot of verts assigned to the **leg_l** group, it is a very small unnoticeable percentage. They will be ignored. However we also see a good deal of verts from the right leg is assigned to the left leg bone. Most too are very minuscule but the verts near the groin are very noticeable.

Let's see these percentages in Blender's *Weight Paint Mode,*

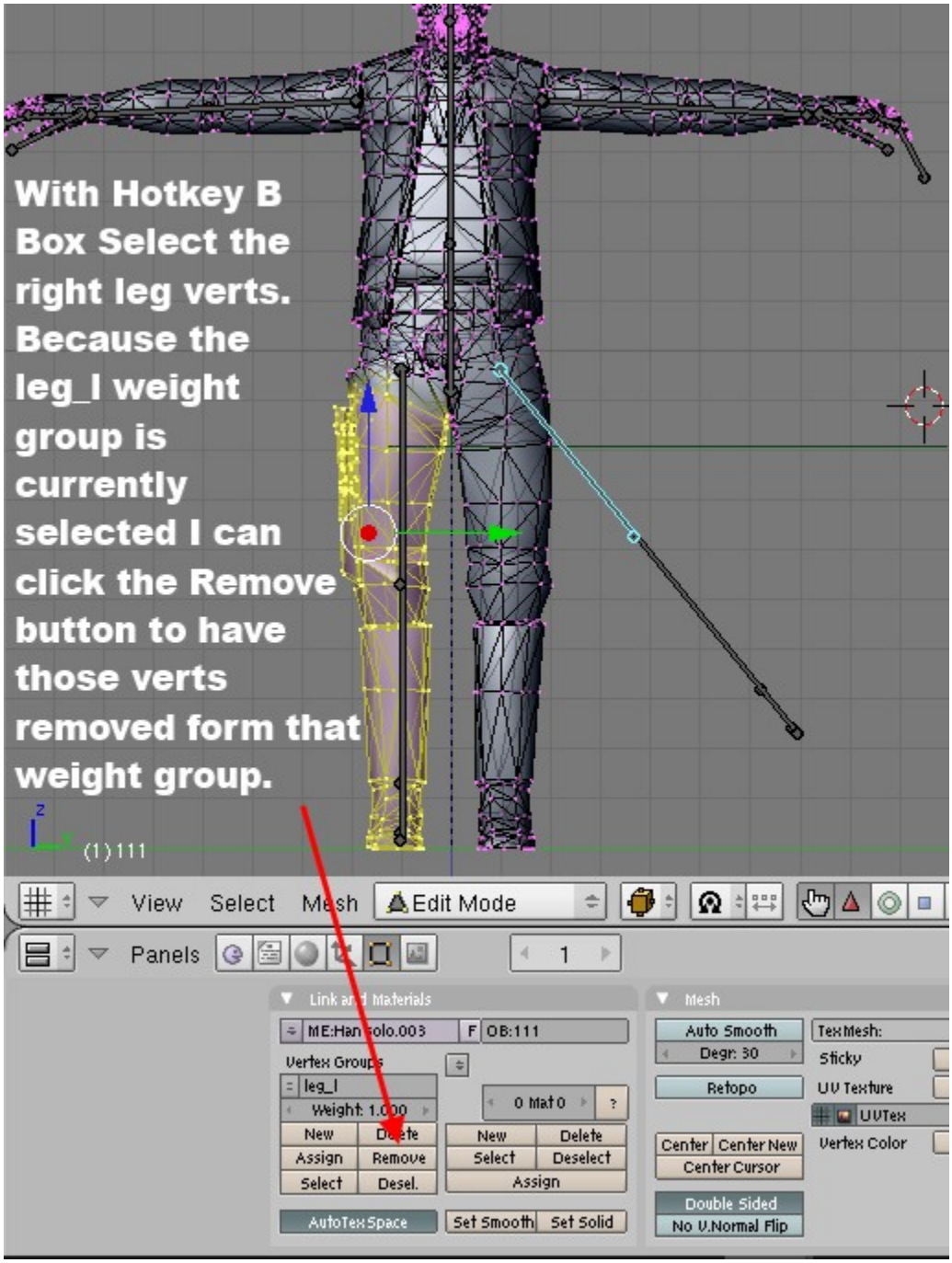


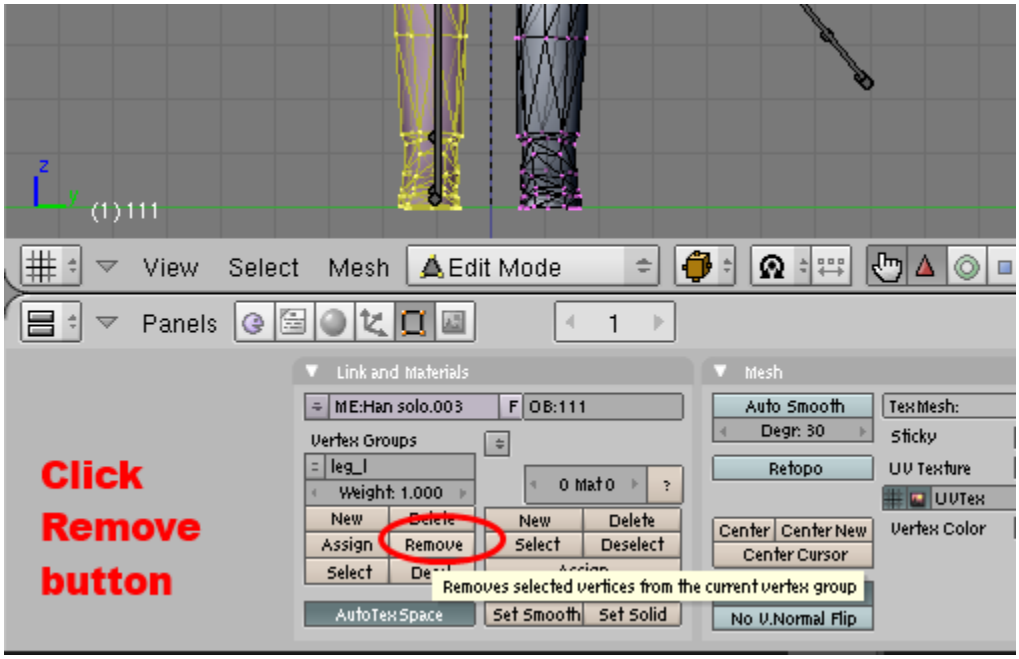
You can see a bit of color on the right leg. Dark blue is zero weights and any other color is a percentage. Go back into *Edit Mode*.

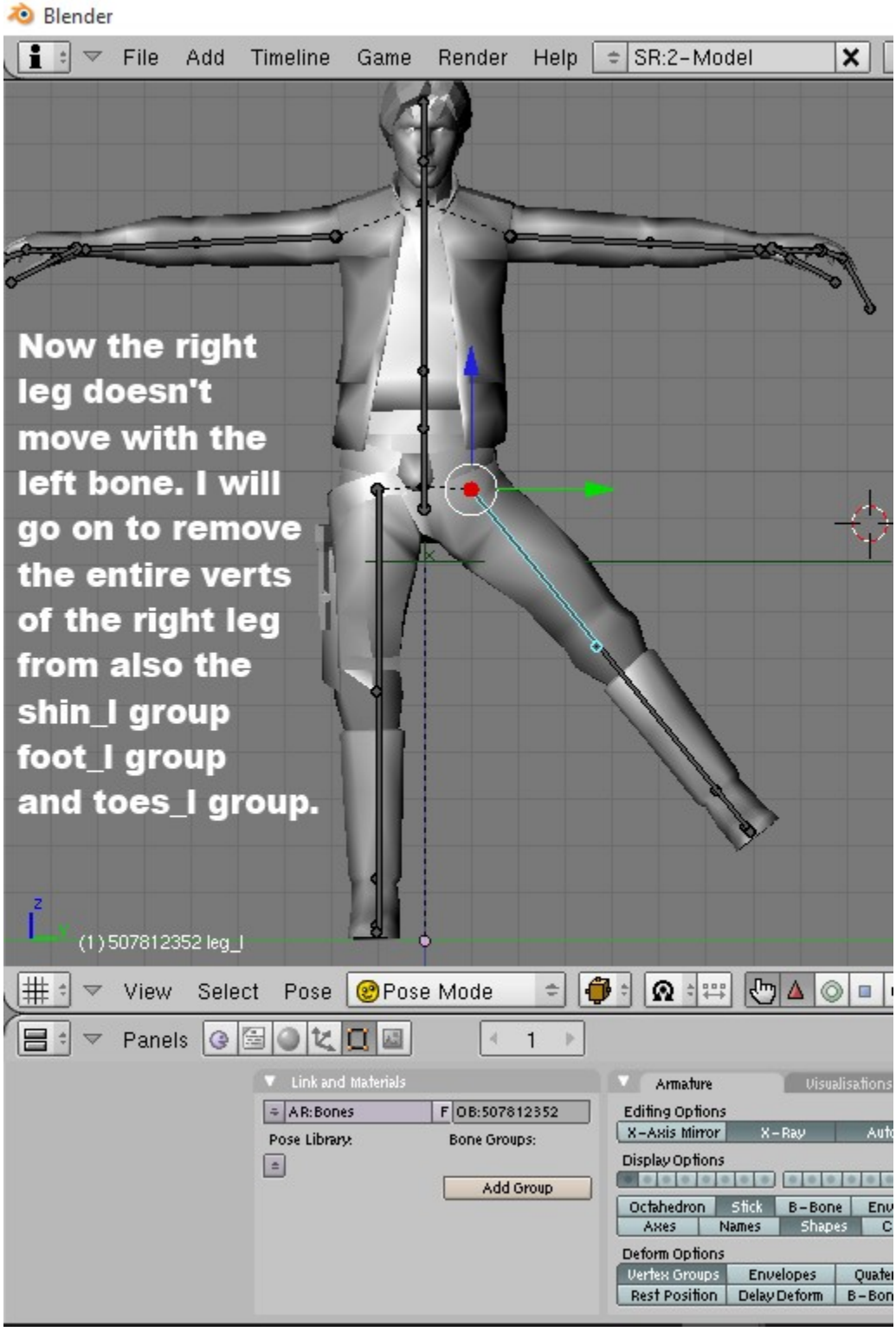
Deselect all verts with Hotkey **A**. Use Hotkey **B** for Box Select to select only the verts of the right leg.

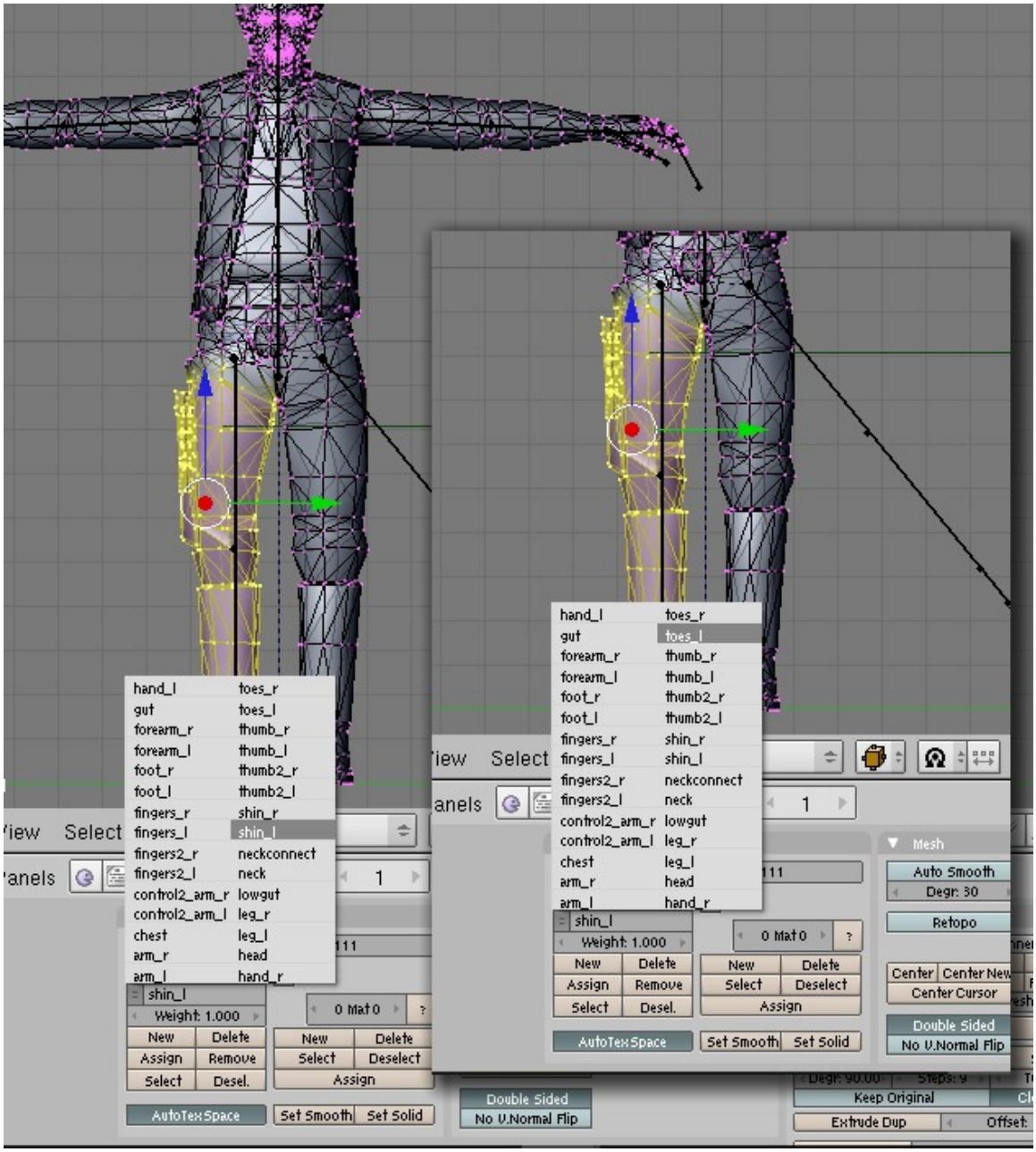


Now all it takes is a simple click of a button. This step is actually easier then you may think. Down in buttons window click **Remove**. Any verts that are selected will get removed from whatever group is showing in the little group box. Be sure to only remove verts from the group that doesn't need them.

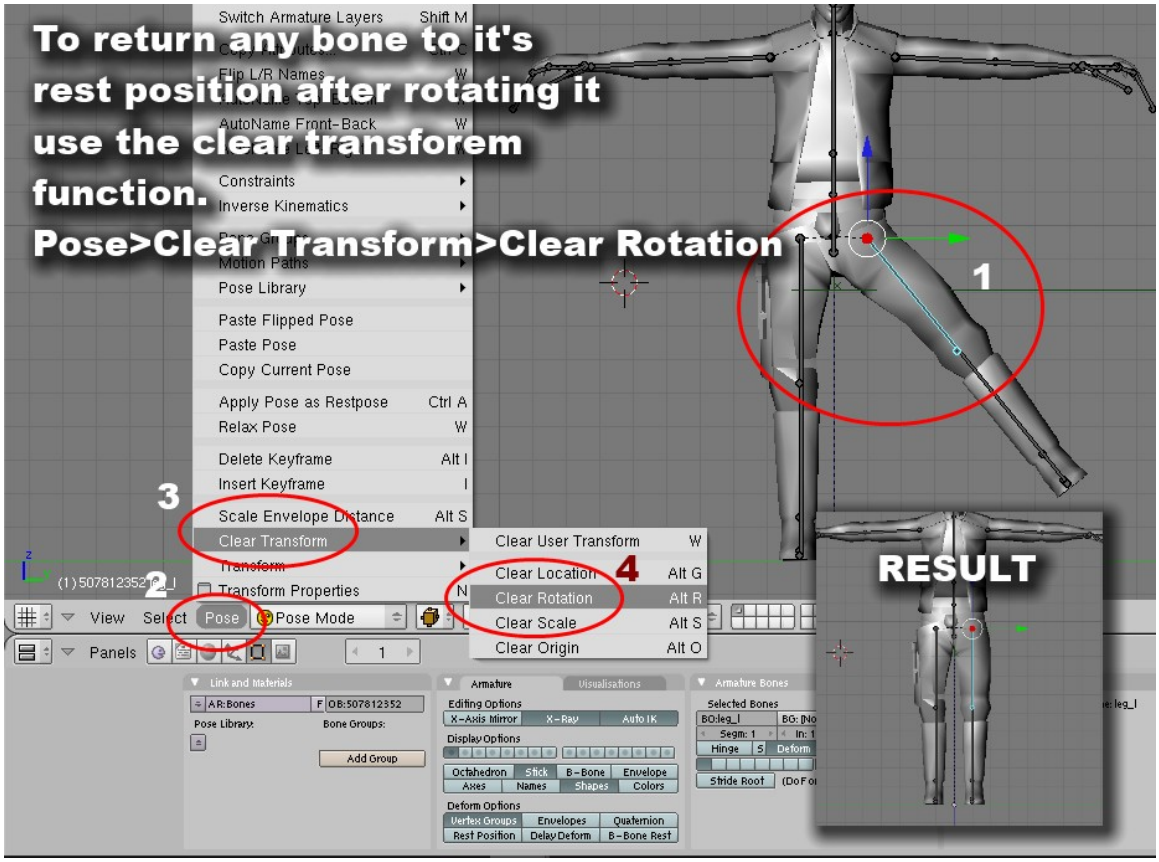

With Hotkey B Box Select the right leg verts. Because the leg_l weight group is currently selected I can click the Remove button to have those verts removed form that weight group.
(1) 111
View
Select
Mesh
Edit Mode
Panels
1
Link and Materials
ME:Han Solo.003
F
OB:111
Vertex Groups
leg_l
Weight: 1.000
New
Delete
Assign
Remove
Select
Desel.
0 Mat 0
?
New
Delete
Select
Deselect
Assign
AutoTexSpace
Set Smooth
Set Solid
Mesh
Auto Smooth
Degr: 30
Retopo
Center
Center New
Center Cursor
Double Sided
No V.Normal Flip
TexMesh:
Sticky
UV Texture
UVTex
Vertex Color


(1) 111
View
Select
Mesh
Edit Mode
Panels
1
Link and Materials
ME:Han solo.003
F
OB:111
Vertex Groups
leg_l
Weight: 1.000
0 Mat 0
New
Delete
Assign
Remove
Select
Deselect
Removes selected vertices from the current vertex group
AutoTexSpace
Set Smooth
Set Solid
Mesh
Auto Smooth
Degr: 30
Retopo
Center
Center New
Center Cursor
No V.Normal Flip
TexMesh:
Sticky
UV Texture
UVTex
Vertex Color
Click
Remove
button

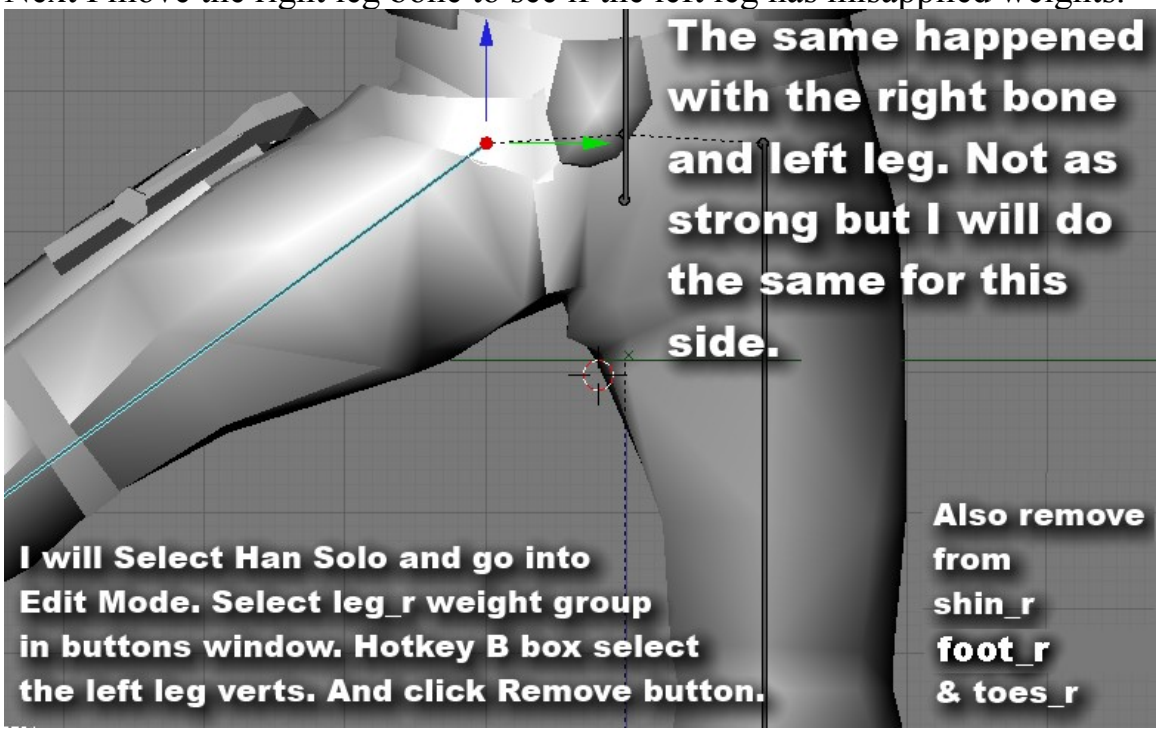

Blender
File Add Timeline Game Render Help SR:2-Model
Now the right leg doesn't move with the left bone. I will go on to remove the entire verts of the right leg from also the shin_l group foot_l group and toes_l group.
(1) 507812352 leg_l
View Select Pose Pose Mode
Panels 1
Link and Materials
AR:Bones F OB:507812352
Pose Library:
Bone Groups:
Add Group
Armature Visualisations
Editing Options
X-Axis Mirror X-Ray
Display Options
Octahedron Stick B-Bone
Axes Names Shapes
Deform Options
Vertex Groups Envelopes
Rest Position Delay Deform



In Edit Mode I did the same with the other groups that may cause a problem. The toes_l, shin_l and foot_l weight groups.
I left the left leg bone extended so I will return it to it's *rest position*.

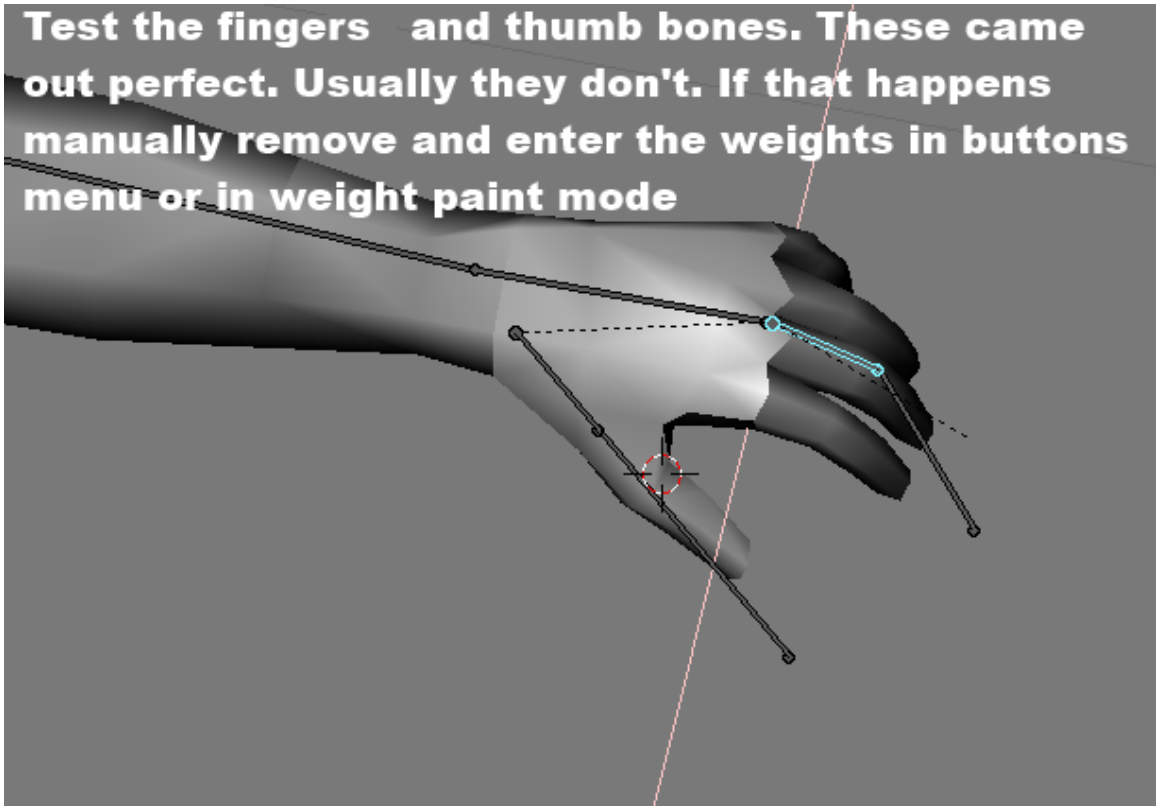


In Pose Mode>Pose>Clear Transform>Clear Rotation.
Next I move the right leg bone to see if the left leg has misapplied weights.

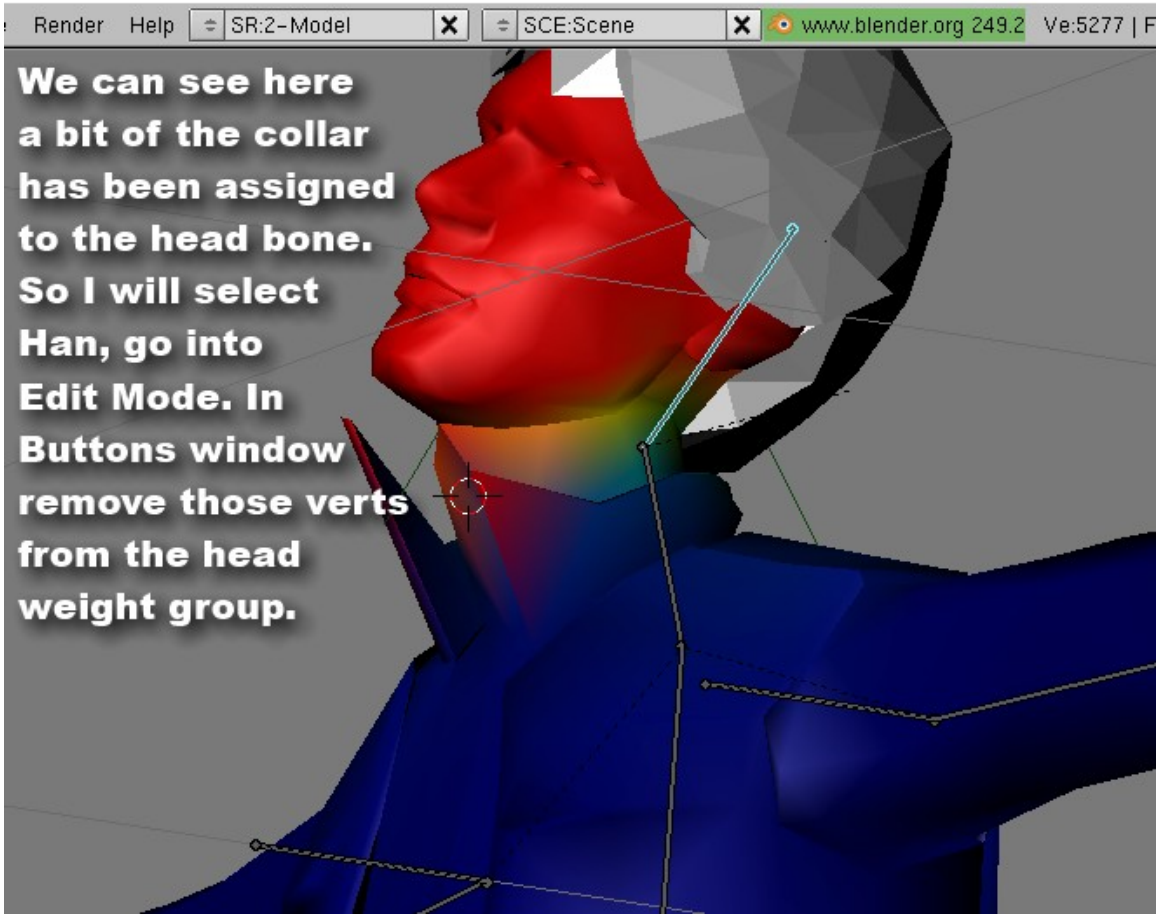


Just as before but with the left leg. Once the legs are fixed lets go see how

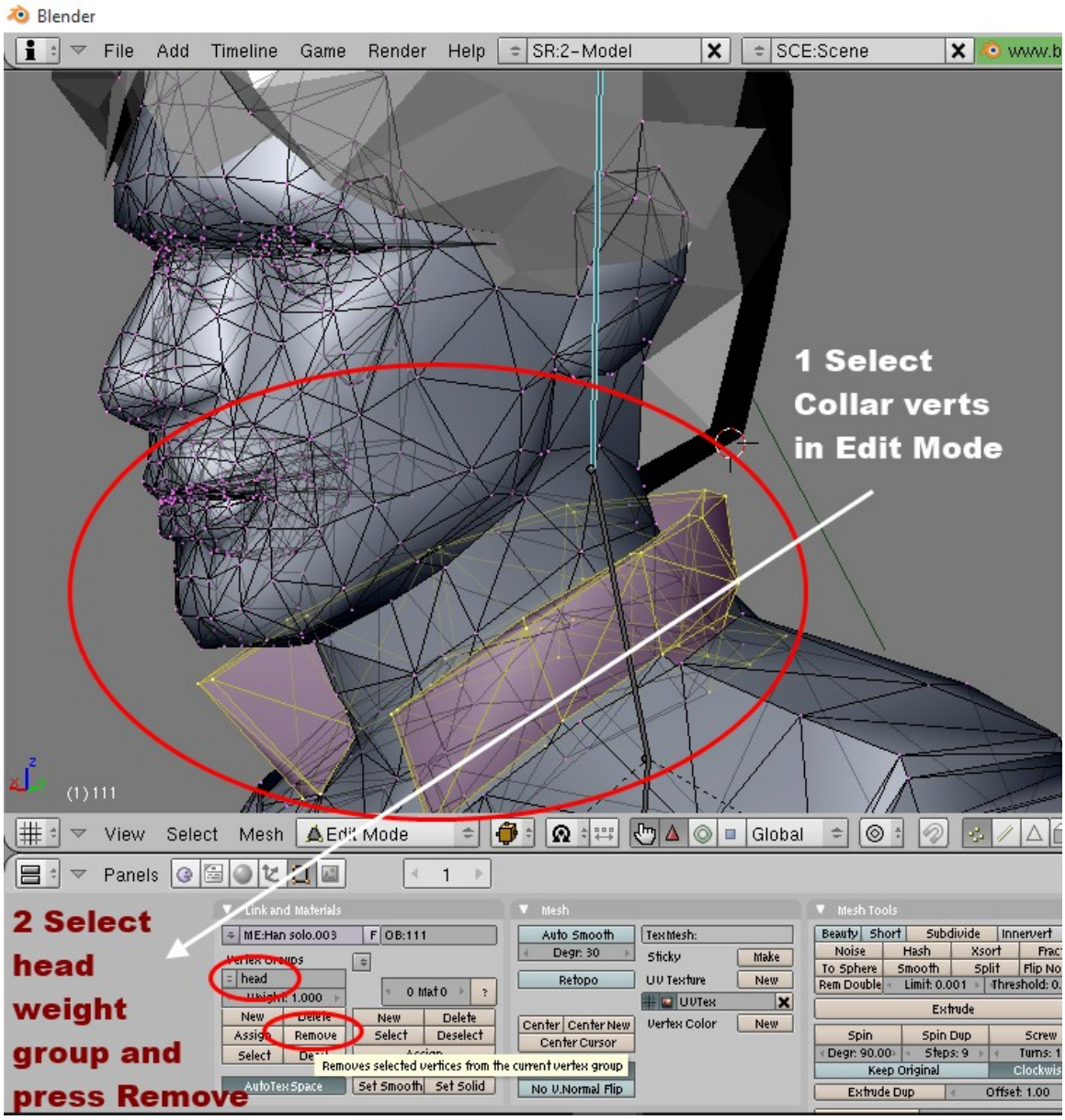
the hands turned out.

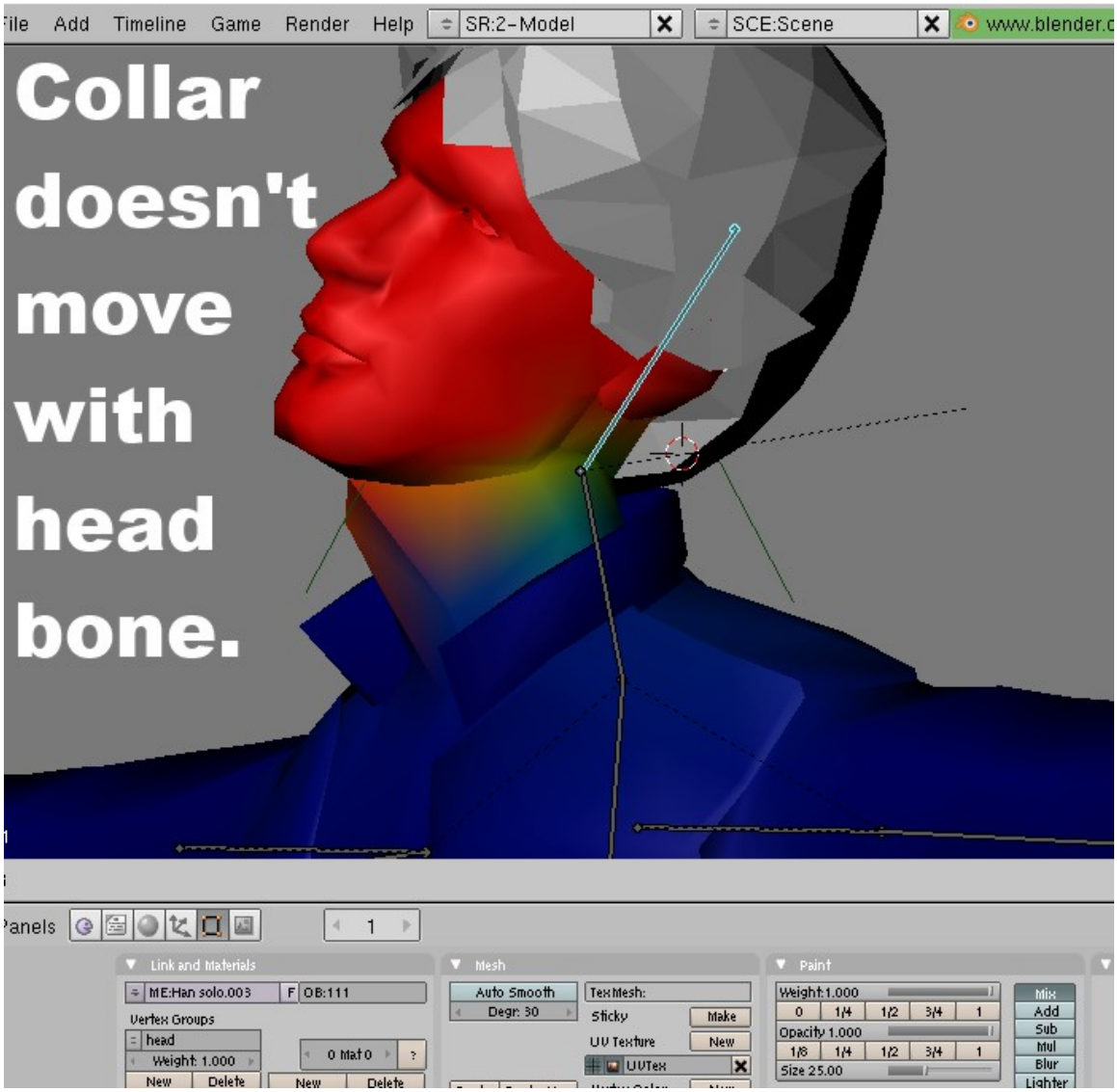


Now test the neck and head bone weights.



The verts at the front of the collar have head bone weights and as before I will remove them in the same way in *Edit Mode.*

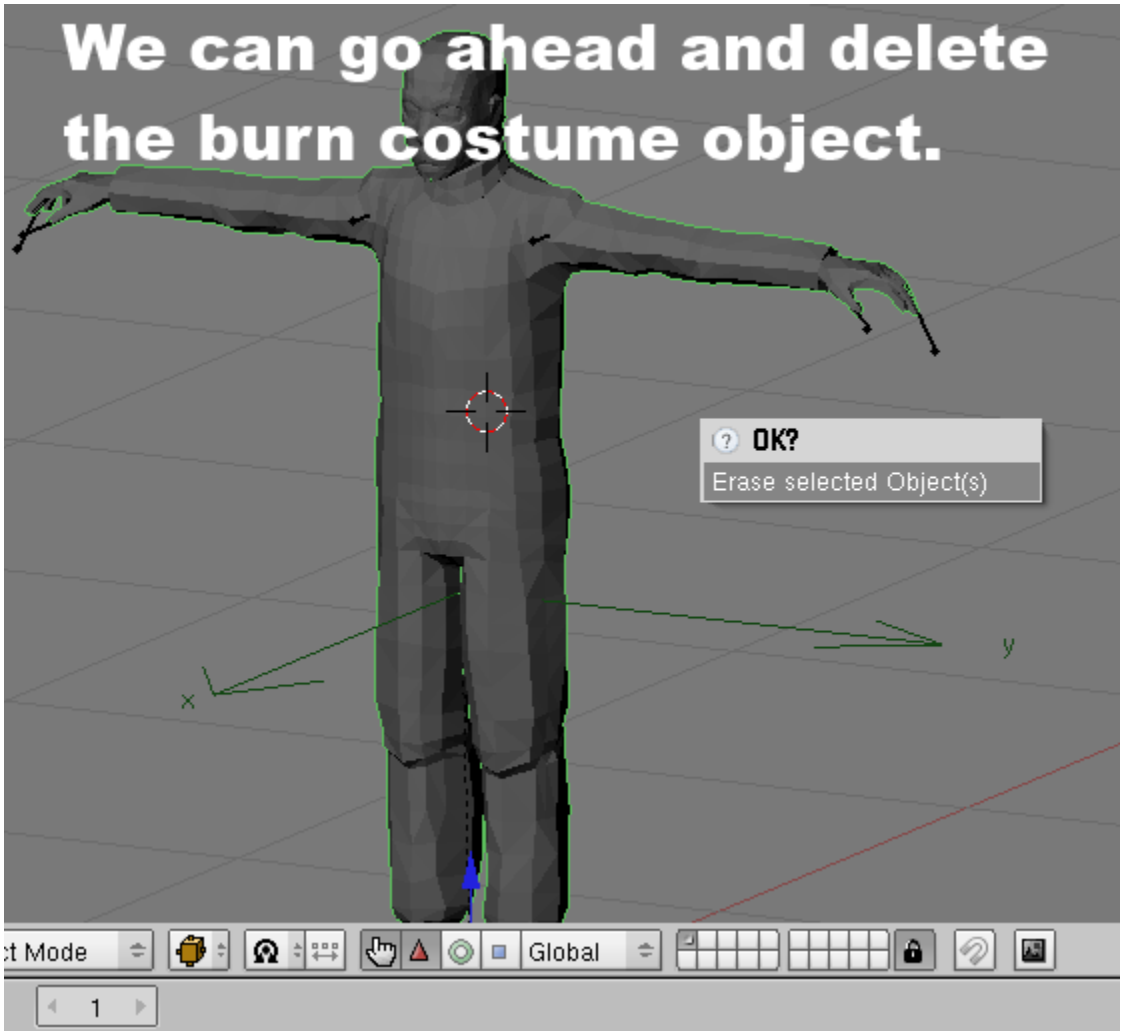

Blender
File Add Timeline Game Render Help
SR:2-Model
SCE:Scene
1 Select
Collar verts
in Edit Mode
View Select Mesh Edit Mode
Global
Panels
2 Select
head
weight
group and
press Remove
Link and Materials
Mesh
Mesh Tools
Removes selected vertices from the current vertex group



Go on to test other bones.

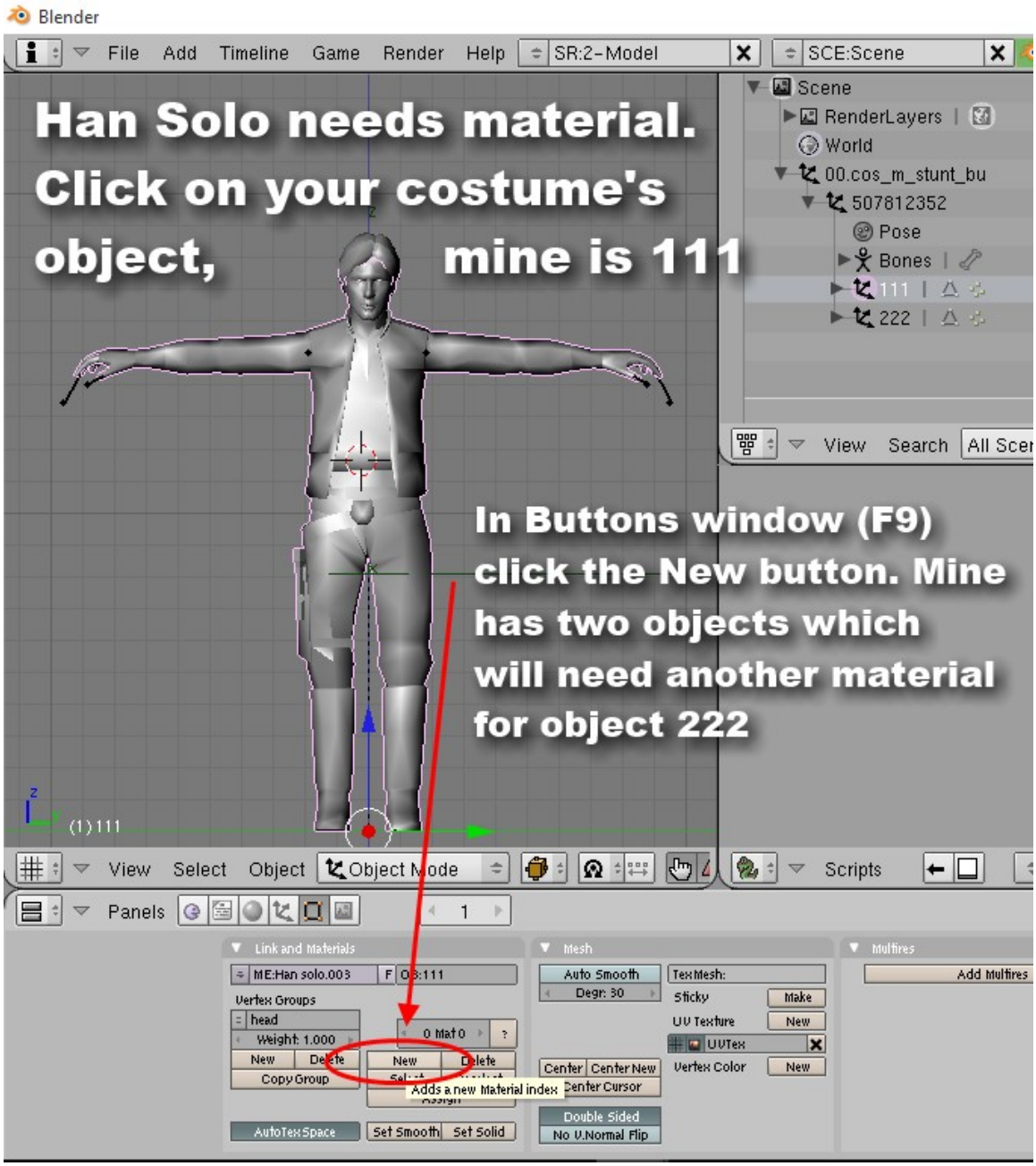
11. Delete the stunt burn costume

The transfer in this example went well so there is no more need to keep the stunt burn costume, which is our bone weight source object. Delete it.

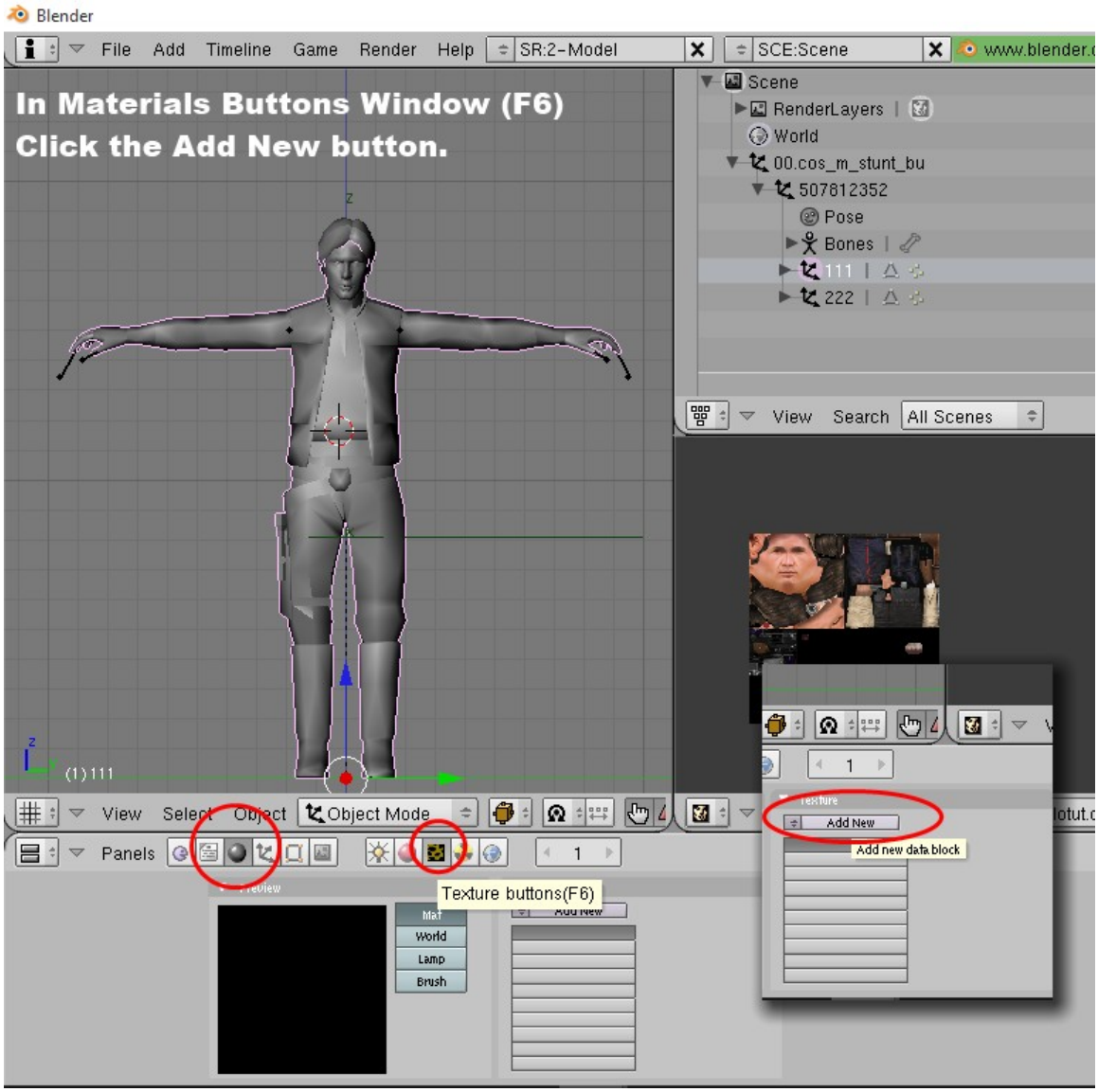


12. Prepare your project for export

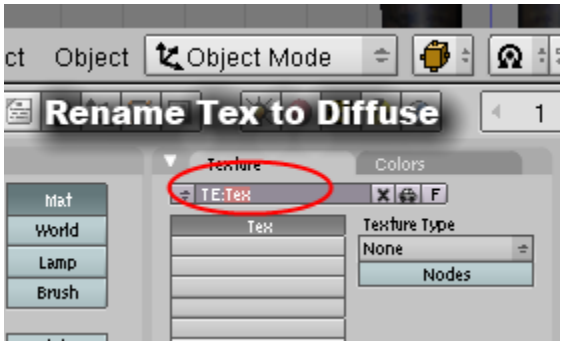
From here on it is standard procedure. Give the objects *materials* and enter them into *blend groups*.

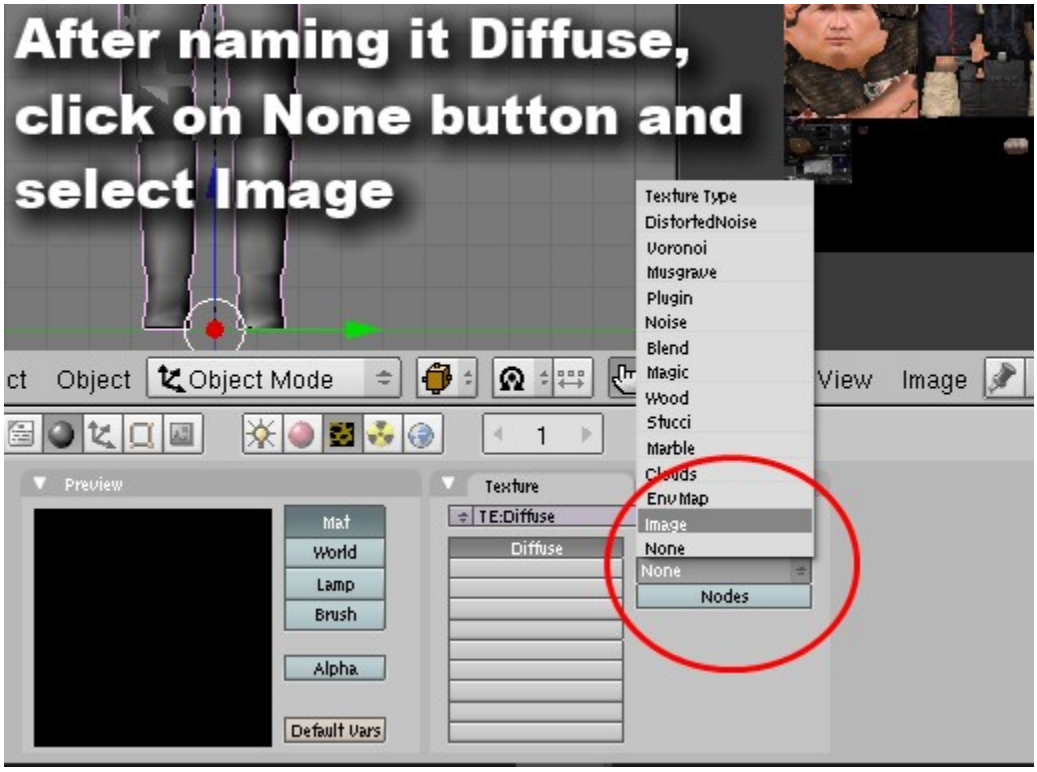


I use the **F9** buttons window to generate new *materials* by clicking the **New** button. However in *materials* button window **F6** is where *materials* get the settings The Movies needs.

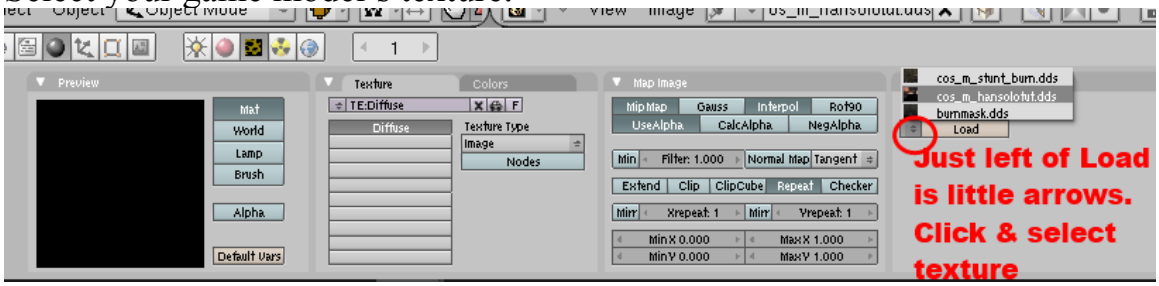


There is a button called **Add New**. Click it. This turns on one of the layered boxes just below it. It will be called *Tex*. It needs to be renamed to **Diffuse**. Click on the name above that stack called TE:Tex to rename it.

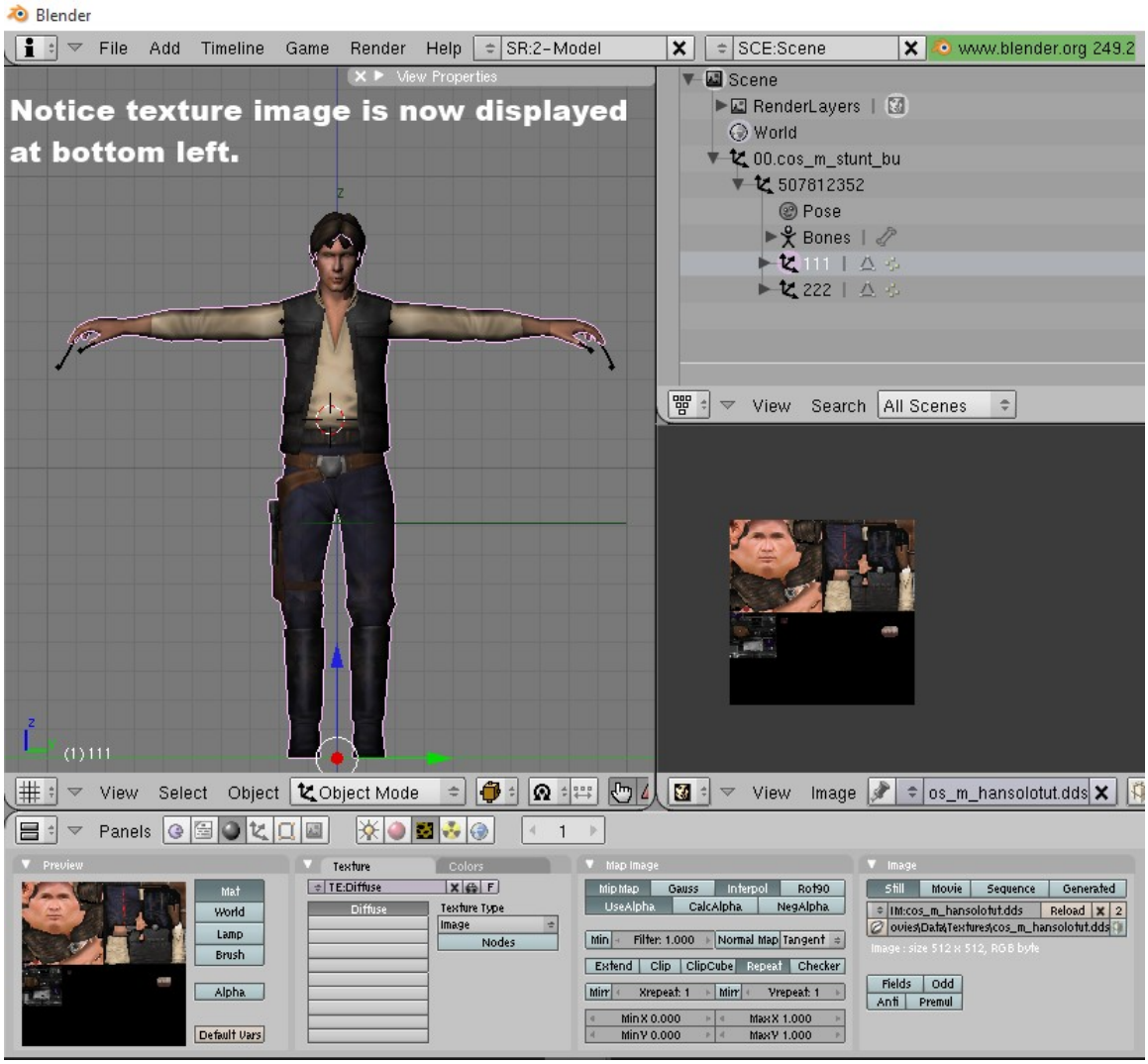




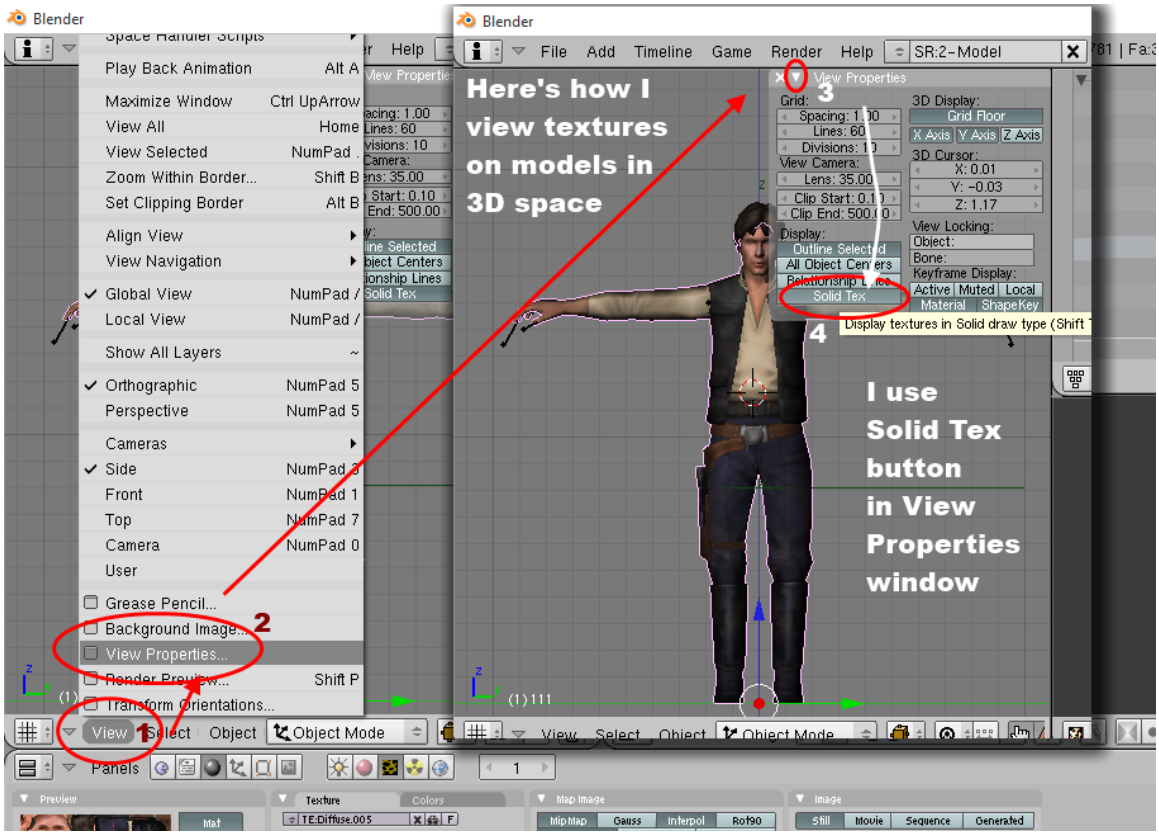
Now the Tex below says **Diffuse**. The Diffuse needs an image assigned to it. The button to the right called **None**, click that and select **Image.** More options keep being made available on the right. At the far right is a *Load* button. **But don't press it**. (Unless your model hasn't a texture yet.) Assuming the texture is already loaded into Blender, click the **little arrows** next to the *Load* button to reveal a little menu of images to select from. Select your game model's texture.



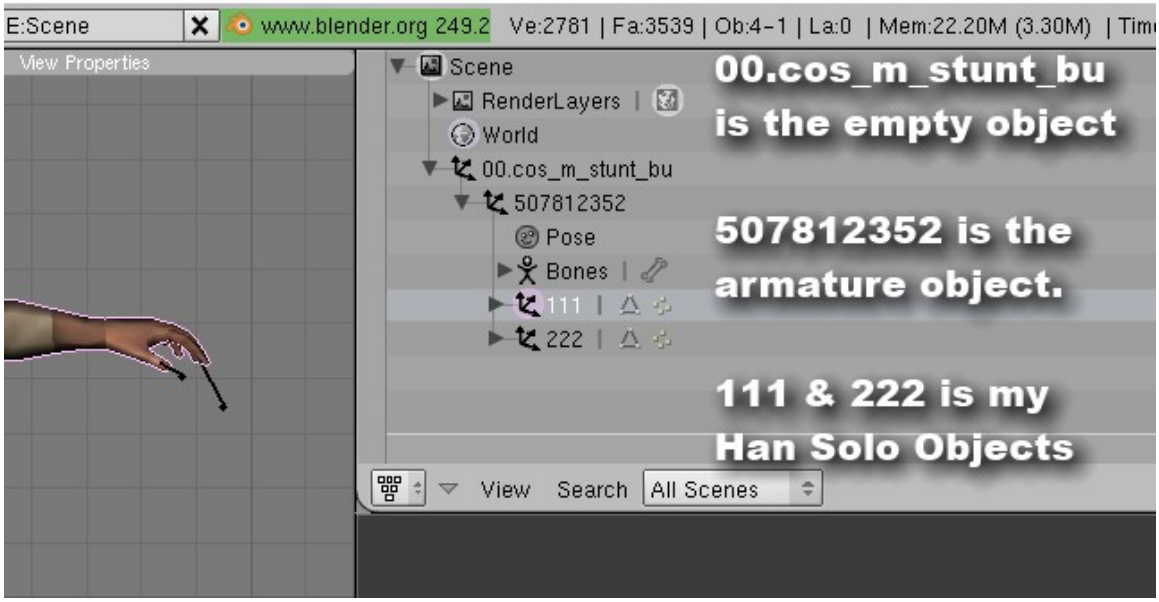
This will assign the image to the Diffuse block. And it will show up at the far left where we started.



Not required for the export but there are a few ways to see the texture over your model. A quick way is to use the *View Properties Window*.

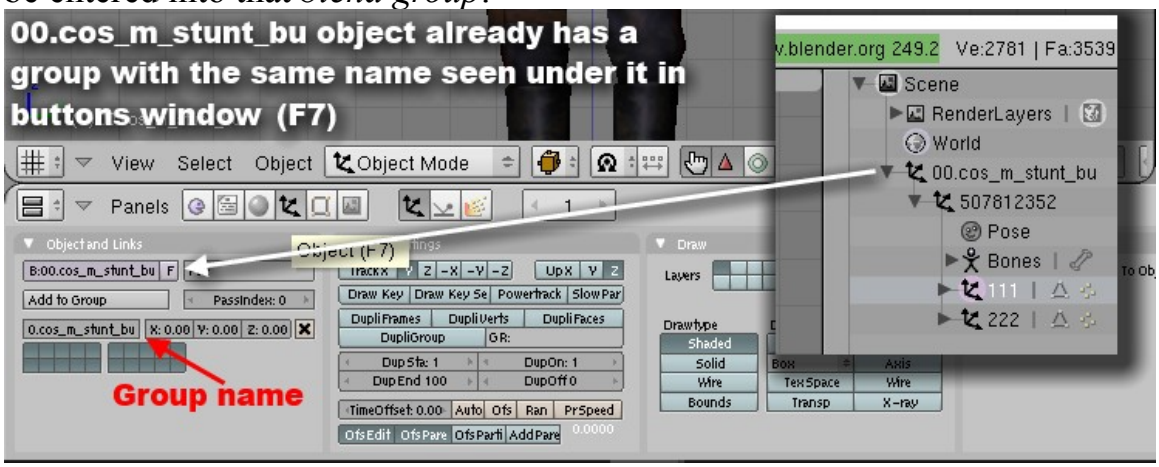


In the Outliner window we can see if the Parent/Child hierarchy is correct.
The Empty object (00.cos_m_stunt_bu) is the *Parent* of everything.
The first child is the *armature* 507812352.
Which is the *Parent* of the game model objects.
In my case 111 & 222, Han Solo's Body and Hair.

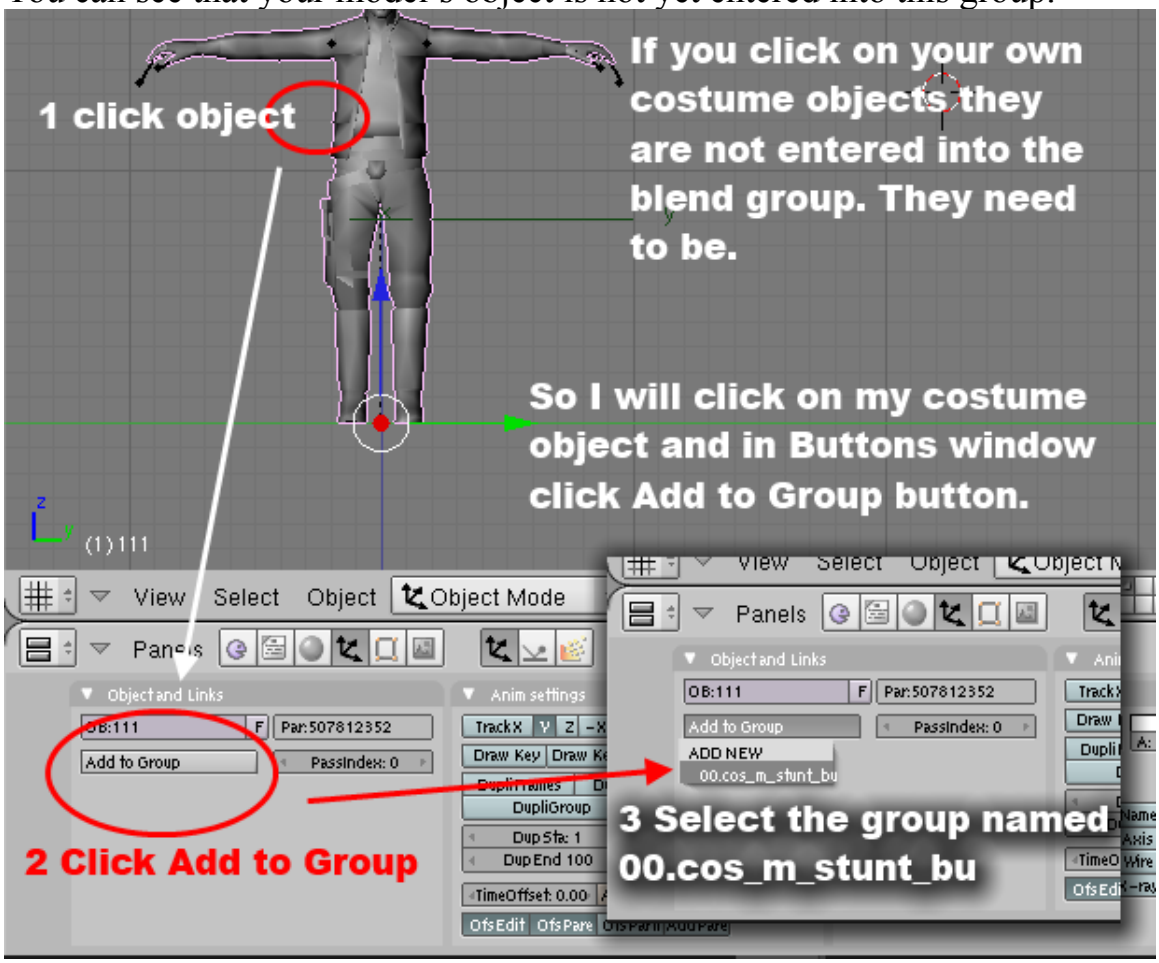


The stunt burn costume has a *blend group* called **00.cos_m_stunt_bu** which

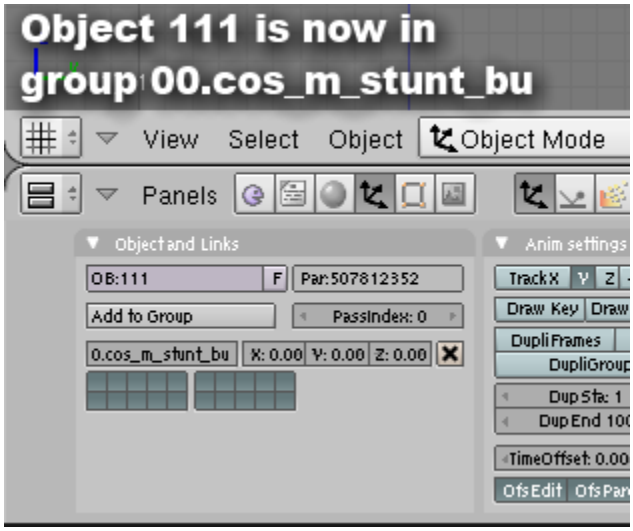
is also the same name as the *Empty* object. Your game objects also need to be entered into that *blend group*.



You can see that your model's object is not yet entered into this group.



Select you model object and click the button called **Add to Group**. From the drop down select **00.cos_m_stunt_bu**.

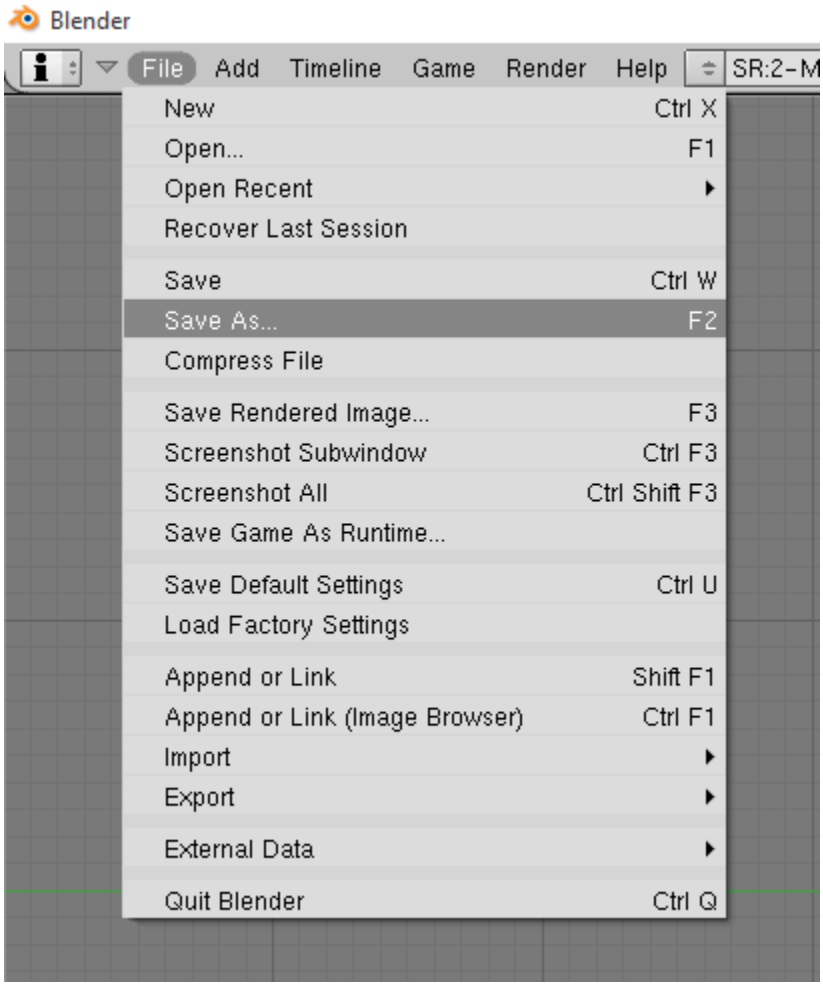


13. Save & Load 3xs

Before you run the Preflight Check there is something you should do to clean your project from the unneeded items still lingering in Blender's memory. We deleted the stunt burn costume but the texture and material it possessed are still hanging around. Blender doesn't let go of things easily. But there is a way to flush Blender's memory of items that no longer have any use.

It's saving and loading your project 3 times.

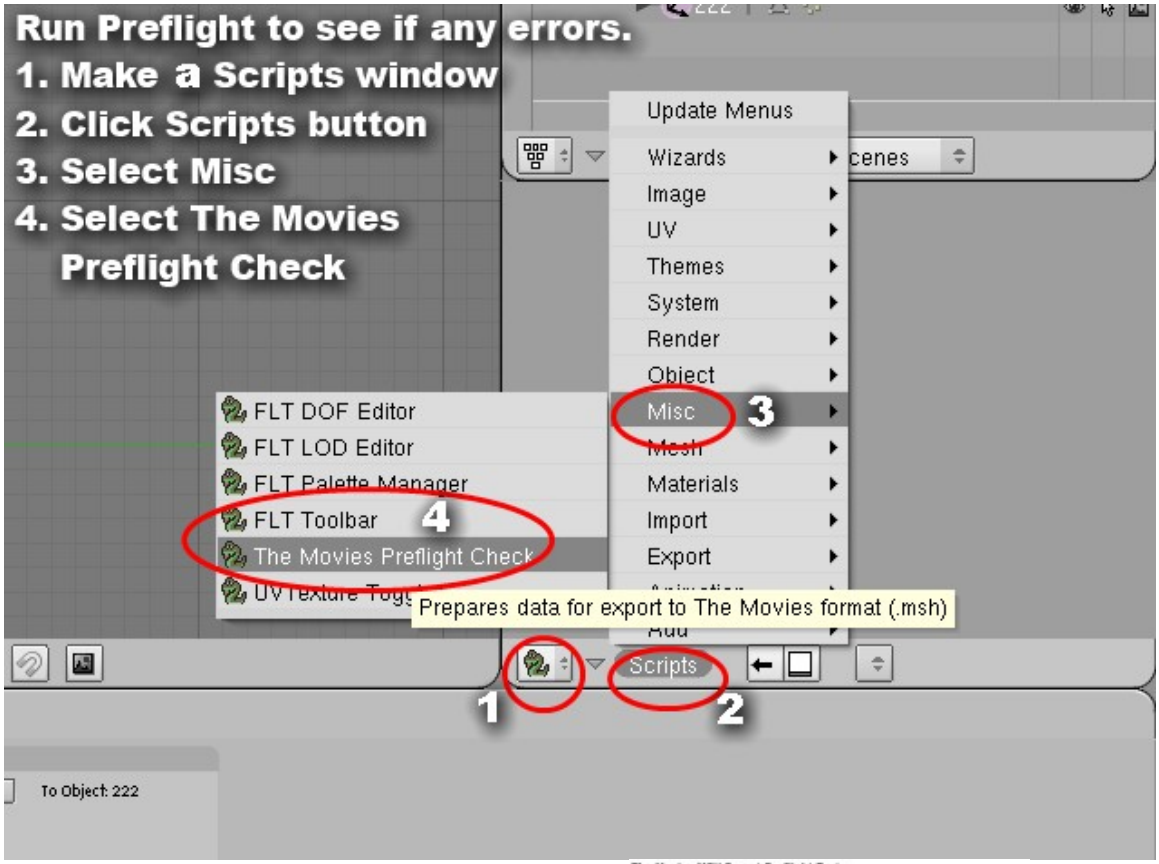
It sounds hacky but it works.



Pretty basic. Save your project as a .blend file. Like myprojectbeforeexport.blend. After it is saved. Go up to file and LOAD this same .blend file. Now save it again. And you get the idea. Save and load 3xs. Afterwards unneeded items are gone.

14. Run The Movies Preflight Check

Turn the *UV Edit window* into the *Scripts window*. Click the **Scripts** button, select **Misc** and select **The Movies Preflgiht Check.**



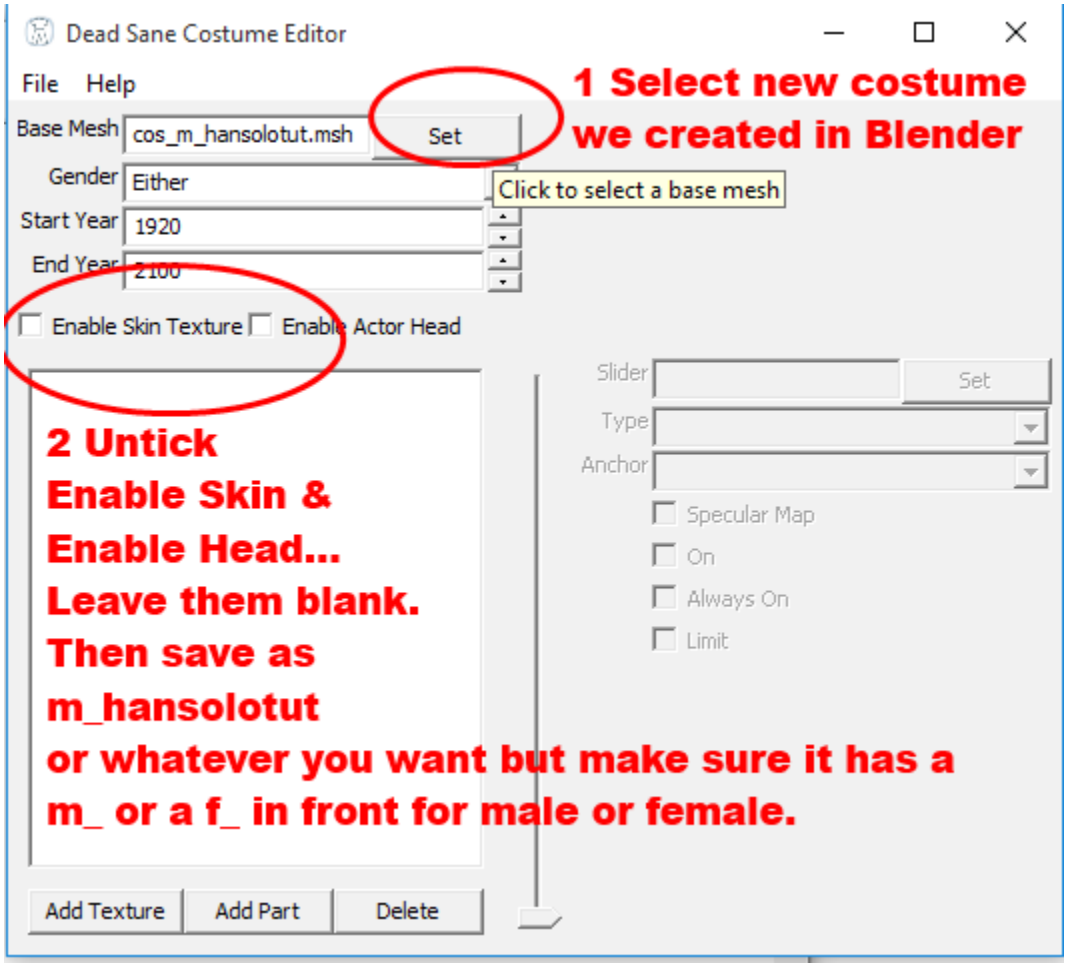
If all went well you get all green lights.

15. Export the model to the game.

The Movies Export scripts generates (.MSH) files. They go into your game's data\meshes folder. Costumes have a *cos_m* or *cos_f* in front of the name. Mine is **cos_m_hansolotut.msh**. M is male f is female.
C:\Program Files\Lionhead Studios Ltd\The Movies\Data\Meshes

16. Make the model available as a costume.

I wanted to use MED (The Movies Editor) to inject the costume, but this Windows10 is having issues. MED will not load for me any file found in the *program files* folder. I can view and extract the .PAK content but not much else. So I will use Dead Sane's CosEd to inject the costumes.
If you have CosEd, fire it up.



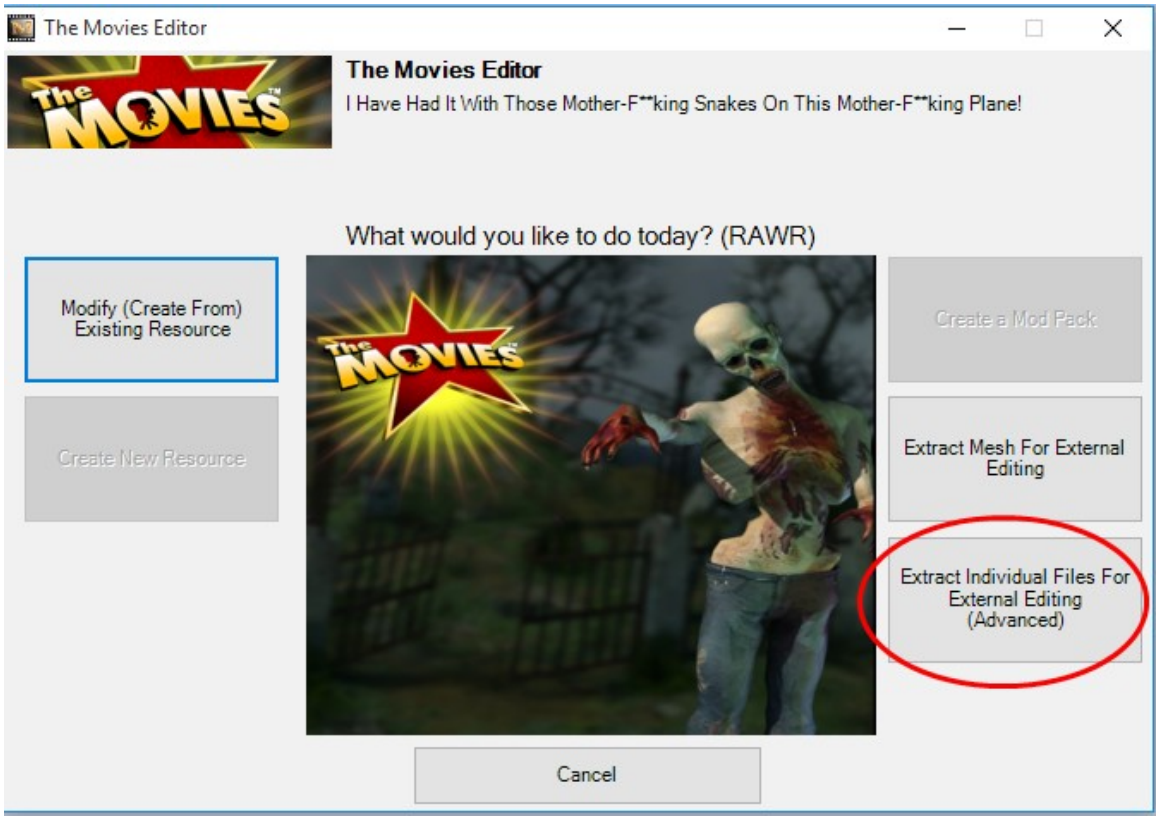
All we have to do is put our new costume (.MSH) into the *Base Mesh* field. Click the **Set** button, navigate to The Movies data\meshes folder and select your new costume. Untick the boxes for *Head* and *Skin*. Han's *Gender* is male. Save the new (.COS) file in The Movies data\costume\datas folder. Create any of these subfolders if you don't have them yet.

I saved mine as m_hansolotut.cos

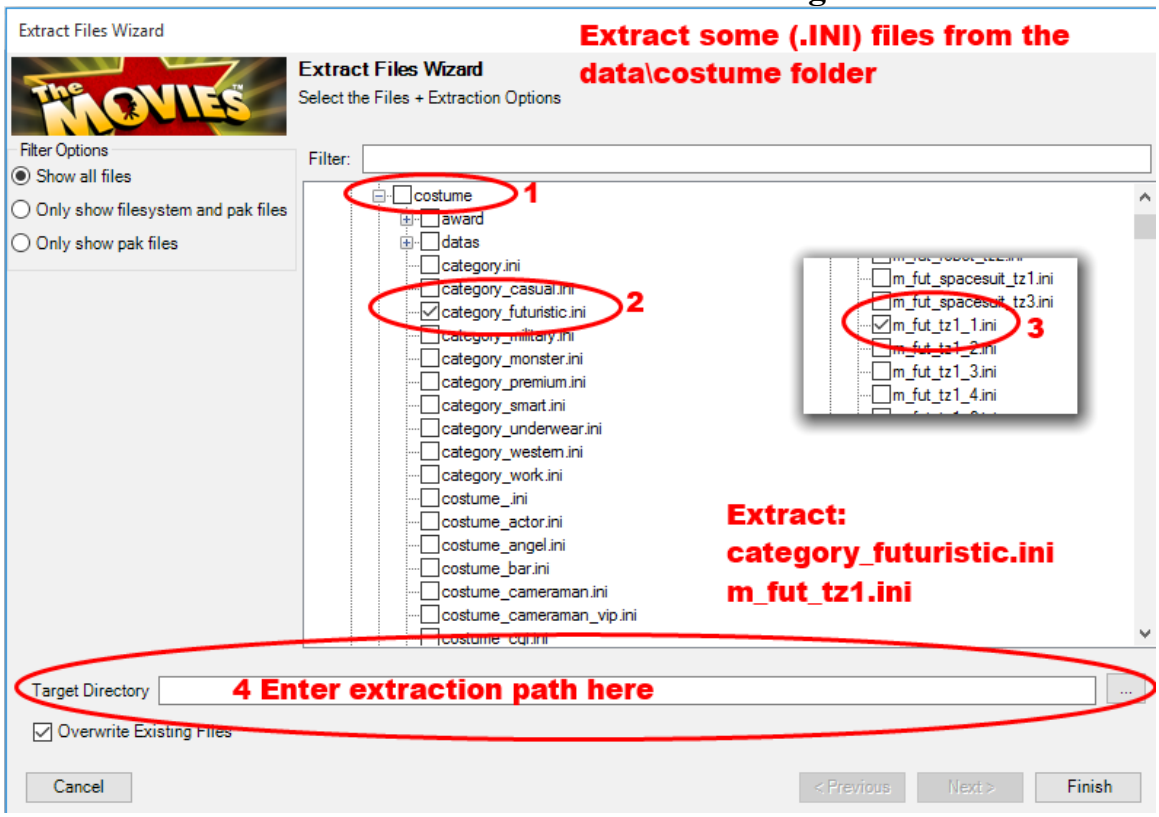
Since I can't use MED to inject the costume, I will use MED to extract the two (.INI) files I will need to manually inject the costume.

Costumes are made available for selection with three files. The (.COS) file, created by CosEd. And two (.INI) files. One is a category file which is a list of costume names to be selected by genre. The game has 7 or 8 costume categories and there are only 7 or 8 of these files. The other tells the game the costume exist and has the same name as the (.COS) files. If your (.COS) is called m_spiderman.cos then the (.INI) file will be called m_spiderman.ini.

We need to extract two files then. One is a category file and the other we will rename to our mod's file name. Start MED.

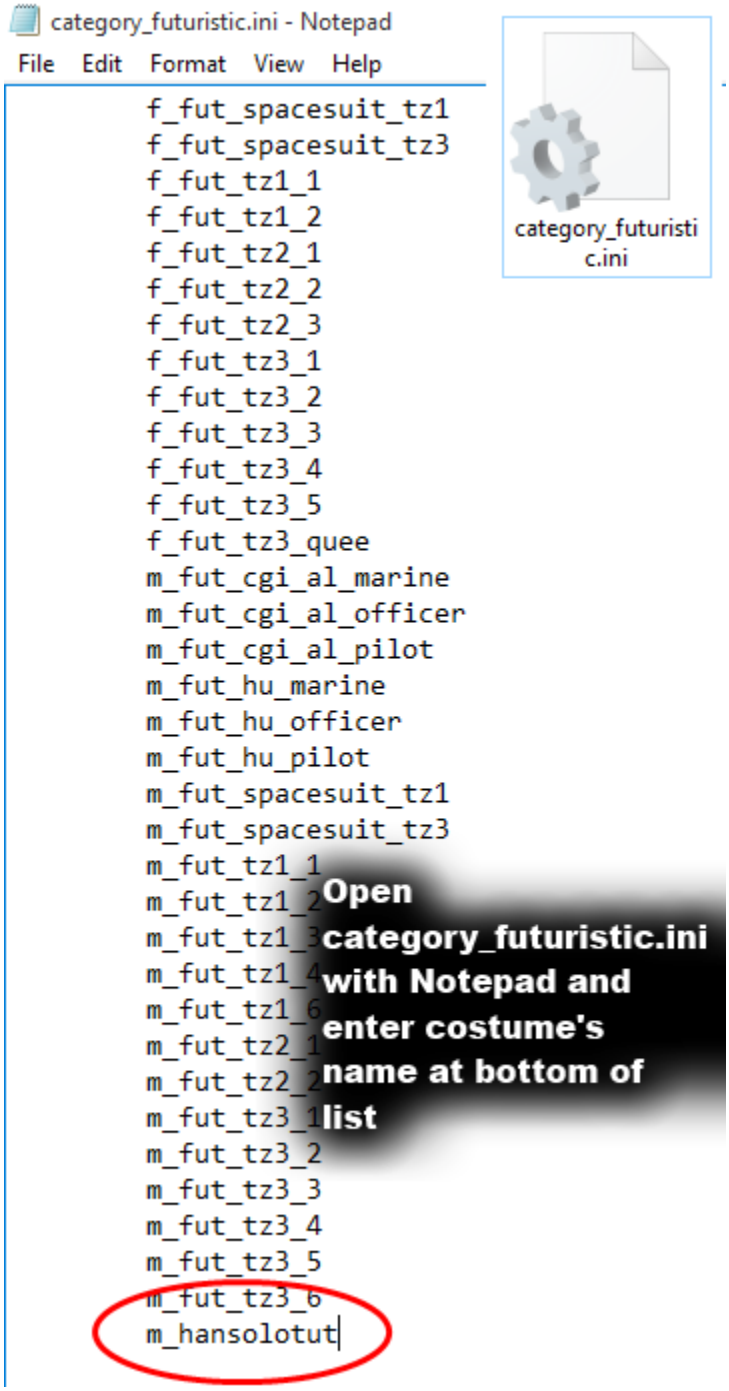


Select **Extract Individual Files For External Editing.**



Expand the list. Select **category_futuristic.ini** and **m_fut_tz1.ini** files. You

put a check mark in the little boxes. Then enter the extraction address in the bar at the bottom. Finally click **Finish** button.
Close MED. Now find the **m_fut_tz1.ini** file you extracted and rename it to your costumes name. Mine is m_hansolotut.ini. Next open the category_futuristic.ini file with Notepad.



It is a list of all costumes found in the futuristic category. At the bottom of the list enter the name of your costume. Make sure there are no blank lines

between names. And make sure the name has a <TAB> space in front of it. Save it. The category_futuristic.ini & m_hansolotut.ini (or your own named costume) goes into your game's data\costume folder. Start The Movies.

To Summarize:

1. Import and save your new costume as it's own .blend file.
2. Use MED to extract the stunt burn costume.
3. Restarting Blender, import the stunt burn costume.
4. Merge the stunt burn costume's head and body objects.
5. *Append* your new game model.
6. *Scale* & *rotate* game model to same size and orientation as stunt costume.
7. Do a *Bone Weight Transfer*.
8. Delete the stunt burn costume.
9. *Parent* the game model to the *Armature*.
10. Test your new model's weights by posing bones.
11. Fix any weight issues you find, if any.
12. Give game model a Material>Diffuse block>Image>Texture image.
13. Enter game model into the *blend group*.
14. Save and Load your Blender project 3 times.

14. Run **The Movies Preflight Check**.
15. Export model and make it available to The Movies game.

About This Tutorial

This is the Archive version of MovieBandit's Tutorials. Please ask permission before posting it.

You should only follow this version. You are welcome to download it for your personal viewing.

The Movies game is the property of Lionhead Studies LTD.

I am not Lionhead Studios LTD or Activision.

Some of the items presented as downloads in this tutorial may no t be available.

List Of Hotkeys For Blender 3D

Window HotKeys

Certain window managers also use the following hotkeys. So ALT-CTRL can be substituted for CTRL to perform the functions described below if a conflict arises.

CTRL-LEFTARROW. Go to the previous Screen.
CTRL-RIGHTARROW. Go to the next Screen.
CTRL-UPARROW or CTRL-DOWNARROW. Maximize the window or return to the previous window display size.
SHIFT-F4. Change the window to a Data View
SHIFT-F5. Change the window to a 3D Window
SHIFT-F6. Change the window to an IPO Window
SHIFT-F7. Change the window to a Buttons Window
SHIFT-F8. Change the window to a Sequence Window
SHIFT-F9. Change the window to an Outliner Window
SHIFT-F10. Change the window to an Image Window
SHIFT-F11. Change the window to a Text Window
SHIFT-F12. Change the window to an Action Window

Universal HotKeys

The following HotKeys work uniformly in all Blender Windows, if the Context allows:

CTRL-LMB. Lasso select: drag the mouse to form a freehand selection area.

ESC.
This key always cancels Blender functions without changes.
or: FileWindow, DataView and ImageSelect: back to the previous window type.
or: the RenderWindow is pushed to the background (or closed, that depends on the operating system).

SPACE. Open the Toolbox.
CTRL-SPACE. Opens menu with manipulator type choices. (Press 2, 3, 4, or 5 to change the widget type) *
ALT-SPACE. Opens menu with manipulator orientation choices. *
TAB. Start or quit EditMode. Toggles between horizontal and verticle splitting when the user is splitting the window.
F1. Loads a Blender file. Changes the window to a FileWindow.

SHIFT-F1. Appends parts from other files, or loads as Library-data. Changes the window to a FileWindow, making Blender files accessible as a directory.

F2. Writes a Blender file. Change the window to a FileWindow.

SHIFT-F2. Exports the scene as a DXF file

CTRL-F2. Exports the scene as a VRML1 file

F3. Writes a picture (if a picture has been rendered). The fileformat is as indicated in the DisplayButtons. The window becomes a File Select Window.

CTRL-F3 (ALT-CTRL-F3 on MacOSX). Saves a screendump of the active window. The fileformat is as indicated in the DisplayButtons. The window becomes a FileWindow.

SHIFT-CTRL-F3. Saves a screendump of the whole Blender screen. The fileformat is as indicated in the DisplayButtons. The window becomes a FileWindow.

F4. Displays the Logic Context (if a ButtonsWindow is available).

F5. Displays the Shading Context (if a Buttons Window is available), Light, Material or World Sub-contextes depends on active object.

F6. Displays the Shading Context and Texture Sub-context (if a ButtonsWindow is available).

F7. Displays the Object Context (if a ButtonsWindow is available).

F8. Displays the Shading Context and World Sub-context (if a ButtonsWindow is available).

F9. Displays the Editing Context (if a ButtonsWindow is available).

F10. Displays the Scene Context (if a ButtonsWindow is available).

F11. Hides or shows the render window.

F12. Starts the rendering from the active camera.

LEFTARROW. Go to the previous frame.

SHIFT-LEFTARROW. Go to the first frame.

RIGHTARROW. Go to the next frame.

SHIFT-RIGHTARROW. Go to the last frame.

UPARROW. Go forward 10 frames.

DOWNARROW. Go back 10 frames.

ALT-A. Change the current Blender window to Animation Playback mode. The cursor changes to a counter.

ALT-SHIFT-A. The current window, plus all 3DWindows go into Animation Playback mode.

IKEY. Insert Key menu. This menu differs from window to window.

JKEY. Toggle the render buffers. Blender allows you to retain two different rendered pictures in memory.

CTRL-O. Opens the last saved file.

QKEY. OK? Quit Blender. This key closes Blender. Blender quit is displayed in the console if Blender is properly closed.

ALT-CTRL-T. TimerMenu. This menu offers access to information about drawing speed. The results are displayed in a pop-up.

CTRL-U. OK, Save User defaults. The current project (windows, objects, etc.), including UserMenu settings are written to the default file that will be loaded every time you start Blender or set it to defaults by pressing CTRL-X.

CTRL-W. Write file. This key combination allows you to write the Blender file without opening a FileWindow.

ALT-W. Write Videoscape file. Changes the window to a FileWindow.

CTRL-X. Erase All. Everything (except the render buffer) is erased and released. The default scene is reloaded.

CTRL-Y. Redo. Mac users may use CMD-Y.

CTRL-Z. Undo. Mac users may use CMD-Z.

SHIFT-CTRL-Z. Redo. Mac users may use SHIFT-CMD-Z

Object Mode HotKeys

These hotkeys are mainly bound to the 3D Viewport Window, but many work on Objects in most other windows, like IPOs and so on, hence they are summarized here.

HOME. All Objects in the visible layer are displayed completely, centered in the window.

PAGEUP. Select the next Object Key. If more than one Object Key is selected, the selection is shifted up cyclically. Only works if the AnimButtons->DrawKey is ON for the Object.

SHIFT-PAGEUP. Adds to selection the next Object Key.

PAGEDOWN. Select the previous Object Key. If more than one Object Key is selected, the selection is shifted up cyclically. Only works if the AnimButtons->DrawKey is ON for the Object.

SHIFT-PAGEDOWN. Adds to selection the previous Object Key.

ACCENT. (To the left of the 1KEY in US keyboard) Select all layers.

(N.B. SINGLE-QUOTE-KEY on UK, ÖKEY on DE & SE, ÆKEY on DK, ØKEY on NO, ÒKEY on IT and ÙKEY on AZERTY keyboard)

SHIFT-ACCENT. Revert to the previous layer setting.

(N.B. SHIFT-SINGLE-QUOTE-KEY on UK, SHIFT-ÖKEY on DE & SE, SHIFT-ÆKEY on DK, SHIFT-ØKEY on NO, SHIFT-ÒKEY on IT and SHIFT-ÙKEY on AZERTY keyboard)

TAB. Start/stop EditMode. Alternative hotkey: ALT-E.

AKEY. Selects/deselects all.

CTRL-A. Apply size and rotation. The rotation and dimensions of the Object are assigned to the ObData (Mesh, Curve, etc.). At first glance, it appears as if nothing has changed, but this can have considerable consequences for animations or texture mapping. This is best illustrated by also having the axis of a Mesh Object be drawn (EditButtons->Axis). Rotate the Object and activate Apply. The rotation and dimensions of the Object are 'erased'.

SHIFT-CTRL-A. If the active Object is automatically duplicated (see AnimButtons->DupliFrames or AnimButtons- >Dupliverts), a menu asks Make duplis real?. This option actually creates the Objects. If the active Mesh Object is deformed by a Lattice, a menu asks Apply Lattice deform?. Now the deformation of the Lattice is assigned to the vertices of the Mesh.

SHIFT-A. This is the AddMenu. In fact, it is the ToolBox that starts with the `ADD' option. When Objects are added, Blender starts EditMode immediately if possible.

BKEY. Border Select. Draw a rectangle with the LeftMouse; all Objects within this area are selected, but not made active. Draw a rectangle with the MiddleMouse to deselect Objects. In orthonormal ViewMode, the dimensions of the rectangle are displayed, expressed as global coordinates, as an extra feature in the lower left corner. In Camera ViewMode, the dimensions that are to be rendered according to the DisplayButtons are displayed in pixel units.

SHIFT-B. Render Border. This only works in Camera ViewMode. Draw a rectangle to render a smaller cut-out of the standard window frame. If the option DisplayButtons->Border is ON, a box is drawn with red and black lines.

CKEY. Centre View. The position of the 3DCursor becomes the new centre of the 3DWindow.

ALT-C. Convert Menu. Depending on the active Object, a PopupMenu is displayed. This enables you to convert certain types of ObData. It only converts in one direction, everything ultimately degrades to a Mesh! The options are:

Font -> Curve
MetaBall -> Mesh The original MetaBall remains unchanged.
Curve -> Mesh
Surface -> Mesh

CTRL-C. Copy Menu. This menu copies information from the active Object to (other) selected Objects.

Fixed components are:

Copy Loc: the X,Y,Z location of the Object. If a Child is involved, this location is the relative position in relation to the Parent.

Copy Rot: the X,Y,Z rotation of the Object.
Copy Size: the X,Y,Z dimension of the Object.
DrawType: copies Object Drawtype.
TimeOffs: copies Object time offset.
Dupli: all Duplicator data (Dupliframes, Dupliverts and so on)
Mass: Real time stuff.
Damping: Real time stuff.
Properties: Real time stuff.
Logic Bricks: Real time stuff.
Constraints: copies Object constraints.

If applicable:

Copy TexSpace: The texture space.

Copy Particle Settings: the complete particle system from the AnimButtons.

For Curve Objects:

Copy Bevel Settings: all bevelling data from the EditButtons.

Font Objects:

Copy Font Settings: font type, dimensions, spacing.

Copy Bevel Settings: all bevelling data from the EditButtons.

Camera Objects:

Copy Lens: the lens value.

SHIFT-C. CentreZero View. The 3DCursor is set to zero (0,0,0) and the view is changed so that all Objects, including the 3Dcursor, can be displayed. This is an alternative for HOME.

DKEY. Draw mode menu. Allows to select draw modes exactly as the corresponding menu in the 3D viewport header does.

SHIFT-D. Add Duplicate. The selected Objects are duplicated. Grab mode starts immediately thereafter.

ALT-D. Add Linked Duplicate. Of the selected Objects linked duplicates

are created. Grab mode starts immediately thereafter.

CTRL-D. Draw the (texture) Image as wire. This option has a limited function. It can only be used for 2D compositing.

ALT-E. Start/stop EditMode. Alternative hotkey: TAB.

FKEY. If selected Object is a mesh Toggles Face selectMode on and off.

CTRL-F. Sort Faces. The faces of the active Mesh Object are sorted, based on the current view in the 3DWindow. The leftmost face first, the rightmost last. The sequence of faces is important for the Build Effect (AnimButtons).

GKEY. Grab Mode. Or: the translation mode. This works on selected Objects and vertices. Blender calculates the quantity and direction of the translation, so that they correspond exactly with the mouse movements, regardless of the ViewMode or view direction of the 3DWindow. Alternatives for starting this mode:

LMB to draw a straight line.

The following options are available in translation mode:

Limiters:

CTRL: in increments of 1 grid unit.

SHIFT: fine movements.

SHIFT-CTRL: in increments of 0.1 grid unit.

SHIFT-X: Locking the X axis in place. Moving along Y, Z axis only. *

SHIFT-Y: Locking the Y axis in place. Moving along X, Z axis only. *

SHIFT-Z: Locking the Z axis in place. Moving along X, Y axis only. *

MMB toggles: A short click restricts the current translation to the X,Y or Z axis. Blender calculates which axis to use, depending on the already initiated mouse movement. Click MiddleMouse again to return to unlimited translation.

XKEY, YKEY, ZKEY constrains movement to X, Y or Z axis of the global reference.

a second XKEY, YKEY, ZKEY constrains movement to X, Y or Z axis of the local reference.

a third XKEY, YKEY, ZKEY removes constraints.

NKEY enters numerical input, as well as any numeric

key directly. TAB will switch between values, ENTER finalizes, ESC exits.

ARROWS: These keys can be used to move the mouse cursor exactly 1 pixel.

Grabber can be terminated with:
LMB SPACE or ENTER: move to a new position.
RMB or ESC: everything goes back to the old position.
Switching mode:
GKEY: starts Grab mode again.
SKEY: switches to Size (Scale) mode.
RKEY: switches to Rotate mode.

ALT-G. Clears translations, given in Grab mode. The X,Y,Z locations of selected Objects are set to zero.
CTRL-ALT-G. Translation Manipulator. *

SHIFT-G. Group Selection
Children: Selects all selected Object's Children.
Immediate Children: Selects all selected Object's first level Children.
Parent: Selects selected Object's Parent.
Shared Layers: Selects all Object on the same Layer of active Object

IKEY. Insert Object Key. A keyposition is inserted in the current frame of all selected Objects. A PopupMenu asks what key position(s) must be added to the IpoCurves.
Loc: The XYZ location of the Object.
Rot: The XYZ rotation of the Object.
Size: The XYZ dimensions of the Object
LocRot: The XYZ location and XYZ rotation of the Object.
LocRotSize: The XYZ location, XYZ rotation and XYZ dimensions of the Object.
Layer: The layer of the Object.
Avail: A position is only added to all the current IpoCurves, that is curves which already exists.
Mesh, Lattice, Curve or Surface: depending on the type of Object, a VertexKey can be added

CTRL-J. Join Objects. All selected Objects of the same type are added to the active Object. What actually happens here is that the ObData blocks are combined and all the selected Objects (except for the active one) are deleted. This is a rather complex operation, which can lead to confusing results, particularly when working with a lot of linked data, animation curves and hierarchies.
KKEY. Show Keys. The DrawKey option is turned ON for all selected

Objects. If all of them were already ON, they are all turned OFF.

SHIFT-K. A PopupMenu asks: OK? Show and select all keys. The DrawKey option is turned ON for all selected Objects, and all Object-keys are selected. This function is used to enable transformation of the entire animation system.

LKEY. Makes selected Object local. Makes library linked objects local for the current scene.

CTRL-L. Link selected. Links some of the Active Object data to all selected Objects, the following menu entry appears only if applicable.

To Scene: Creates a link of the Object to a scene.
Object IPOs: Links Active Object IPOs to selected ones.
Mesh data: Links Active Object Mesh data selected ones.
Lamp Data: Links Active Object Lamp data to selected ones.
Surf Data: Links Active Object Surf data selected ones.
Material: Links Active Object Material to selected ones.

SHIFT-L. Select Linked. Selects all Objects somehow linked to active Object.

Object IPO: Selects all Object(s) sharing active Object's IPOs.
Object Data: Selects all Object(s) sharing active Object's ObData.
Current Material: Selects all Object(s) sharing active Object's current Material.
Current Texture: Selects all Object(s) sharing active Object's current Texture.

MKEY. Moves selected Object(s) to another layer, a pop-up appers. Use LMB to move, use SHIFT-LMB to make the object belong to multiple layers. If the selected Objects have different layers, this is ORed in the menu display. Use ESC to exit the menu. Press the "OK" button or ENTER to change the layer seting. The hotkeys (ALT-)(1KEY, 2KEY, ... - 0KEY) work here as well (see 3DHeader).

CTRL-M. Mirror Menu. It is possible to mirror an Object along the X, Y or Z axis.

NKEY. Number Panel. The location, rotation and scaling of the active Object are displayed and can be modified.

ALT-O. Clear Origin. The `Origin' is erased for all Child Objects, which causes the Child Objects to move to the exact location of the Parent Objects.

SHIFT-O. If the selected Object is a Mesh toggles SubSurf on/ off. CTRL-1 to CTRL-4 switches to the relative SubSurf level for display purpouses.

Rendering SUbSurf level has no HotKey.

CTRL-P. Make selected Object(s) the child(ren) of the active Object. If the Parent is a Curve then a popup offers two choices:

Normal Parent: Make a normal parent, the curve can be made a path later on.

Follow Path: Automatically creates a Follow Path constraint with the curve as target. If the Parent is an Armature, a popup offers three options:

Use Bone: One of the Bones becomes the parent. The Object will not be deformed. A popup permits to select the bone. This is the option if you are modeling a robot or machinery

Use Armature: The whole armature is used as parent for deformations. This is the choiche for organic beings.

Use Object: Standard parenting. In the second case further options asks if Vertex groups should not be created, should be created empty or created and populated.

ALT-P. Clears Parent relation, user is asked if he wishes to keep or clear parent-induced transforms.

Clear Parent: the selected Child Objects are unlinked from the Parent. since the transformation of the Parent disappears, this can appear as if the former Children themselves are transformed.

... and keep transform: the Child Objects are unlinked from the Parent, and an attempt is made to assign the current transformation, which was determined in part by the Parent, to the (former Child) Objects.

Clear Parent inverse: The inverse matrix of the Parent of the selected Objects is erased. The Child Objects remain linked to the Objects. This gives the user complete control over the hierarchy.

SHIFT-P. Push/Pull Transformations. It is a bit like scaling, except that every element is moved the same distance toward or away from the center. The distance they move is controlled by moving the mouse toward you (pull away from center) or away from you (push toward center). *

RKEY. Rotate mode. Works on selected Object(s). In Blender, a rotation is by default a rotation perpendicular to the screen, regardless of the view direction or ViewMode. The degree of rotation is exactly linked to the mouse movement. Try moving around the rotation midpoint with the mouse. The rotation pivot point is determined by the state of the 3DWiewport Header buttons. Alternatives for starting this mode:

LMB to draw a C-shaped curve.

The following options are available in rotation mode:

Limiters:

CTRL: in increments of 5 degrees.

SHIFT: fine movements.

SHIFT-CTRL: in increments of 1 degree.

SHIFT-X: Locking the X axis in place. Rotating along Y, Z axis only. *

SHIFT-Y: Locking the Y axis in place. Rotating along X, Z axis only. *

SHIFT-Z: Locking the Z axis in place. Rotating along X, Y axis only. *

MMB toggles: A short click restricts the current rotation to the horizontal or vertical view axis.

XKEY, YKEY, ZKEY constrains rotation to X, Y or Z axis of the global reference.

a second XKEY, YKEY, ZKEY constrains rotation to X, Y or Z axis of the local reference.

a third XKEY, YKEY, ZKEY removes constraints.

NKEY enters numerical input, as well as any numeric key directly. ENTER finalizes, ESC exits.

ARROWS: These keys can be used to move the mouse cursor exactly 1 pixel.

Rotation can be terminated with:

LMB SPACE or ENTER: move to a new position.

RMB or ESC: everything goes back to the old position.

Switching mode:

GKEY: switches to Grab.

SKEY: switches to Size (Scale) mode.

RKEY: starts Rotate mode again.

ALT-R. Clears Rotation. The X,Y,Z rotations of selected Objects are set to zero.

CTRL-ALT-R. Rotation Manipulator. *

SKEY. Size mode or scaling mode. Works on selected Object(s). The degree of scaling is exactly linked to the mouse movement. Try to move from the (rotation) midpoint with the mouse. The pivot point is determined by the settings of the 3D Viewport header pivot Menu. Alternatives for starting scaling mode:

LMB to draw a V-shaped line.

The following options are available in scaling mode:

Limiters:

CTRL: in increments of 0.1.

SHIFT-CTRL: in increments of 0.01.

SHIFT-X: Locking the X axis in place. Scaling along Y, Z axis only. *

SHIFT-Y: Locking the Y axis in place. Scaling along X, Z axis only. *

SHIFT-Z: Locking the Z axis in place. Scaling along X, Y axis only. *

MMB toggles: A short click restricts the scaling to X, Y or Z axis. Blender calculates the appropriate axis based on the already initiated mouse movement. Click MMB again to return to free scaling.

XKEY, YKEY, ZKEY constrains scaling to X, Y or Z axis of the local reference.

a second XKEY, YKEY, ZKEY removes constraints.

NKEY enters numerical input, as well as any numeric key directly. ENTER finalizes, ESC exits.

ARROWS:These keys can be used to move the mouse cursor exactly 1 pixel.

Scaling can be terminated with:

LMB SPACE or ENTER: move to a new position.

RMB or ESC: everything goes back to the old dimension.

Switching mode:

GKEY: switches to Grab.

SKEY: starts Size mode again.

RKEY: switches to Rotation.

ALT-S. Clears Size. The X,Y,Z dimensions of selected Objects are set to 1.0.

CTRL-ALT-S. Scale Manipulator. *

CTRL-SHIFT-S. To Sphere Transformations. An interactive version of the To Sphere tool, working both in Edit Mode (on all data types) and in Object Mode. *

SHIFT-S. SnapMenu:

Sel->Grid: Moves Object to nearest grid point.

Sel->Curs: Moves Object to cursor.

Curs->Grid: Moves cursor to nearest grid point.

Curs->Sel: Moves cursor to selected Object(s).

Sel->Center: Moves Objects to their barycentrum.

TKEY. Texture space mode. The position and dimensions of the texture space for the selected Objects can be changed in the same manner as described above for Grab and Size mode. To make this visible, the drawingflag EditButtons->TexSpace is set ON. A PopupMenu asks you to select: "Grabber" or "Size".

CTRL-T. Makes selected Object(s) track the Active Object. Old track method was Blender default tracking before version 2.30. The new method is the Constrain Track, this creates a fully editable constraint on the selected object targeting the active Object.

ALT-T. Clears old style Track. Constraint track is removed as all constrains are.

UKEY. Makes Object Single User, the inverse operation of Link

(CTRL-L) a pop-up appears with choices.

Object: if other Scenes also have a link to this Object, the link is deleted and the Object is copied. The Object now only exists in the current Scene. The links from the Object remain unchanged.

Object & ObData: Similar to the previous command, but now the ObData blocks with multiple links are copied as well. All selected Objects are now present in the current Scene only, and each has a unique ObData (Mesh, Curve, etc.).

Object & ObData & Materials+Tex: Similar to the previous command, but now Materials and Textures with multiple links are also copied. All selected Objects are now unique. They have unique ObData and each has a unique Material and Texture block.

Materials+Tex: Only the Materials and Textures with multiple links are copied.

VKEY. Switches in/out of Vertex Paint Mode.

ALT-V. Object-Image Aspect. This hotkey sets the X and Y dimensions of the selected Objects in relation to the dimensions of the Image Texture they have. Use this hotkey when making 2D Image compositions and multi-plane designs to quickly place the Objects in the appropriate relationship with one another.

WKEY. Opens Object Booleans Menu.

XKEY. Erase Selected? Deletes selected objects.

ZKEY. Toggles Solid Mode on/off.

SHIFT-Z. Toggles Shaded Mode on/off.

ALT-Z. Toggles Textured Mode on/off.
PAD-/. Toggles layer buttons on/off.

Edit Mode - General

Again, Most of these hotkeys are useful in the 3D Viewport when in Edit Mode, but many works on other Blender Object, so they are summarized here. Many Object Mode keys works in Edit mode too, but on the selected vertices or control points; among these Grab, Rotate, Scale and so on. These hotkeys are not repeated here.

TAB or ALT-E. This button starts and stops Edit Mode.
CTRL-TAB. Switches between Vertex Select, Edge Select, and Face Select modes. Holding SHIFT while clicking on a mode will allow you to combine modes.
AKEY. Select/Unselect all.
BKEY-BKEY. Circle Select. If you press BKEY a second time after starting Border Select, Circle Select is invoked. It works as described above. Use NUM+ or NUM- or MW to adjust the circle size. Leave Circle Select with RMB or ESC.
CTRL-H. With vertices selected, this creates a "Hook" object. Once a hook is selected, CTRL-H brings up an options menu for it.
NKEY. Number Panel. Simpler than the Object Mode one, in Edit Mode works for Mesh, Curve, Surface: The location of the active vertex is displayed.
OKEY. Switch in/out of Proportional Editing.
SHIFT-O. Toggles between Smooth and Sharp Proportional Editing.
PKEY. SeParate. You can choose to make a new object with all selected vertices, edges, faces and curves or create a new object from each separate group of interconnected vertices from a popup. Note that for curves you cannot separate connected control vertices. This operation is the opposite of Join (CTRL-J).
CTRL-P. Make Vertex Parent. If one object (or more than one) is/are selected and the active Object is in Edit Mode with 1 or 3 vertices selected then the Object in Edit Mode becomes the Vertex Parent of the selected Object(s). If only 1 vertex is selected, only the location of this vertex determines the Parent transformation; the rotation and dimensions of the Parent do not play a role here. If three vertices are selected, it is a `normal' Parent relationship in which the 3 vertices determine the rotation and location of the Child together. This method produces interesting effects with

Vertex Keys. In EditMode, other Objects can be selected with CTRL-RMB.

CTRL-S. Shear. In EditMode this operation enables you to make selected forms `slant'. This always works via the horizontal screen axis.

UKEY. Undo. When starting Edit Mode, the original ObData block is saved and can be returned to via UKEY. Mesh Objects have better Undo, see next section.

WKEY. Specials PopupMenu. A number of tools are included in this PopupMenu as an alternative to the Edit Buttons. This makes the buttons accessible as shortcuts, e.g. EditButtons-> Subdivide is also `WKEY, 1KEY'.

SHIFT-W. Warp. Selected vertices can be bent into curves with this option. It can be used to convert a plane into a tube or even a sphere. The centre of the circle is the 3DCursor. The mid-line of the circle is determined by the horizontal dimensions of the selected vertices. When you start, everything is already bent 90 degrees. Moving the mouse up or down increases or decreases the extent to which warping is done. By zooming in/out of the 3Dwindow, you can specify the maximum degree of warping. The CTRL limiter increments warping in steps of 5 degrees.

EditMode - Mesh

This section and the following highlight peculiar EditMode Hotkeys.

ALT-B. Select portion of viewing area to only be visible.

CTRL-NUM+. Adds to selection all vertices connected by an edge to an already selected vertex.

CTRL-NUM-. Removes from selection all vertices of the outer ring of selected vertices.

ALT-CTRL-RMB. Edge select.

CKEY. If using curve deformations, this toggles the curve Cyclic mode on/off.

EKEY. Extrude Selected. "Extrude" in EditMode transforms all the selected edges to faces. If possible, the selected faces are also duplicated. Grab mode is started directly after this command is executed.

SHIFT-EKEY. Crease Subsurf edge. With "Draw Creases" enabled, pressing this key will allow you to set the crease weight. Black edges have no weight, edge-select color have full weight.

CTRL-EKEY. Mark LSCM Seam. Marks a selected edge as a "seam" for unwrapping using the LSCM mode.

FKEY. Make Edge/Face. If 2 vertices are selected, an edge is created. If 3

or 4 vertices are selected, a face is created.

SHIFT-F. Fill selected. All selected vertices that are bound by edges and form a closed polygon are filled with triangular faces. Holes are automatically taken into account. This operation is 2D; various layers of polygons must be filled in succession.

ALT-F. Beauty Fill. The edges of all the selected triangular faces are switched in such a way that equally sized faces are formed. This operation is 2D; various layers of polygons must be filled in succession. The Beauty Fill can be performed immediately after a Fill.

CTRL-F. Flip faces, selected triangular faces are paired and common edge of each pair swapped.

HKEY. Hide Selected. All selected vertices and faces are temporarily hidden.

SHIFT-H. Hide Not Selected: All non-selected vertices and faces are temporarily hidden.

ALT-H. Reveal. All temporarily hidden vertices and faces are drawn again.

ALT-J. Join faces, selected triangular faces are joined in pairs and transformed to quads

KKEY. Knife tool Menu.

Face Loop Select: (SHIFT-R) Face loops are highlighted starting from edge under mouse pointer. LMB finalizes, ESC exits.

Face Loop Cut: (CTRL-R) Face loops are highlighted starting from edge under mouse pointer. LMB finalizes, ESC exits.

Knife (exact): (SHIFT-K) Mouse starts draw mode. Selected Edges are cut at intersections with mouse line. ENTER or RMB finalizes, ESC exits.

Knife (midpoints): (SHIFT-K) Mouse starts draw mode. Selected Edges intersecting with mouse line are cut in middle regardless of true intersection point. ENTER or RMB finalizes, ESC exits.

LKEY. Select Linked. If you start with an unselected vertex near the mouse cursor, this vertex is selected, together with all vertices that share an edge with it.

SHIFT-L. Deselect Linked. If you start with a selected vertex, this vertex is deselected, together with all vertices that share an edge with it.

CTRL-L. Select Linked Selected. Starting with all selected vertices, all vertices connected to them are selected too.

MKEY. Mirror. Opens a popup asking for the axis to mirror. 3 possible axis group are available, each of which contains three axes, for a total of

nine choices. Axes can be Global (Blender Global Reference); Local (Current Object Local Reference) or View (Current View reference). Remember that mirroring, like scaling, happens with respect to the current pivot point.

ALT-M. Merges selected vertices at barycentrum or at cursor depending on selection made on pop-up.

SHIFT+CTRL+ALT+MKEY Selects unclosed edges / holes in the mesh (Edge select mode only).

CTRL-N. Calculate Normals Outside. All normals from selected faces are recalculated and consistently set in the same direction. An attempt is made to direct all normals `outward'.

SHIFT-CTRL-N. Calculate Normals Inside. All normals from selected faces are recalculated and consistently set in the same direction. An attempt is made to direct all normals `inward'.

ALT-S. Whereas SHIFT-S scales in Edit Mode as it does in Object Mode, for Edit Mode a further option exists, ALT-S moves each vertex in the direction of its local normal, hence effectively shrinking/fattening the mesh.

CTRL-T. Make Triangles. All selected faces are converted to triangles.

UKEY. Undo. When starting Edit Mode, the original ObData block is saved and all subsequent changes are saved on a stack. This option enables you to restore the previous situation, one after the other.

SHIFT-U. Redo. This let you re-apply any undone changes up to the moment in which Edit Mode was entered

ALT-U. Undo Menu. This let you choose the exact point to which you want to undo changes.

WKEY. Special Menu. A PopupMenu offers the following options:

Subdivide: all selected edges are split in two.

Subdivide Fractal: all selected edges are split in two and middle vertex displaced randomly.

Subdivide Smooth: all selected edges are split in two and middle vertex displaced along the normal.

Merge: as ALT-M.

Remove Doubles: All selected vertices closer to each other than a given threshold (See EditMode Button Window) are merged ALT-M.

Hide: as HKEY.

Reveal: as ALT-H.

Select Swap: Selected vertices become unselected and vice versa.

Flip Normals: Normals of selected faces are flipped.

Smooth: Vertices are moved closer one to each other, getting a

smoother object.

Bevel: Faces are reduced in size and the space between edges is filled with a smoothly curving bevel of the desired order.

XKEY. Erase Selected. A PopupMenu offers the following options:

Vertices: all vertices are deleted. This includes the edges and faces they form.

Edges: all edges with both vertices selected are deleted. If this `releases' certain vertices, they are deleted as well. Faces that can no longer exist as a result of this action are also deleted.

Faces: all faces with all their vertices selected are deleted. If any vertices are `released' as a result of this action, they are deleted.

All: everything is deleted.

Edges and Faces: all selected edges and faces are deleted, but the vertices remain.

Only Faces: all selected faces are deleted, but the edges and vertices remain.

YKEY. Split. This command splits the selected part of a Mesh without deleting faces. The split parts are no longer bound by edges. Use this command to control smoothing. Since the split parts have vertices at the same position, selection with LKEY is recommended.

EditMode - Curve

CKEY. Set the selected curves to cyclic or turn cyclic off. An individual curve is selected if at least one of the vertices is selected.

EKEY. Extrude Curve. A vertex is added to the selected end of the curves. Grab mode is started immediately after this command is executed.

FKEY. Add segment. A segment is added between two selected vertices at the end of two curves. These two curves are combined into one curve.

HKEY. Toggle Handle align/free. Toggles the selected Bezier handles between free or aligned.

SHIFT-H. Set Handle auto. The selected Bezier handles are converted to auto type.

CTRL-H. Calculate Handles. The selected Bezier curves are calculated and all handles are assigned a type.

LKEY. Select Linked. If you start with an non-selected vertex near the mouse cursor, this vertex is selected together with all the vertices of the same curve.

SHIFT-L. Deselect Linked. If you start with a selected vertex, it is deselected together with all the vertices of the same curve.

MKEY. Mirror. Mirror selected control points exactly as for vertices in a Mesh.

TKEY. Tilt mode. Specify an extra axis rotation, i.e. the tilt, for each vertex in a 3D curve.

ALT-T. Clear Tilt. Set all axis rotations of the selected vertices to zero.

VKEY. Vector Handle. The selected Bezier handles are converted to vector type.

WKEY. The special menu for curves appears:

Subdivide. Subdivide the selected vertices.

Switch direction. The direction of the selected curves is reversed. This is mainly for Curves that are used as paths!

XKEY. Erase Selected. A PopupMenu offers the following options:

Selected: all selected vertices are deleted.

Segment: a curve segment is deleted. This only works for single segments. Curves can be split in two using this option. Or use this option to specify the cyclic position within a cyclic curve.

All: delete everything.

EditMode - Metaball

MKEY. Mirror. Mirror selected control points exactly as for vertices in a Mesh.

EditMode - Surface

CKEY. Toggle Cyclic menu. A PopupMenu asks if selected surfaces in the `U' or the `V' direction must be cyclic. If they were already cyclic, this mode is turned off.

EKEY. Extrude Selected. This makes surfaces of all the selected curves, if possible. Only the edges of surfaces or loose curves are candidates for this operation. Grab mode is started immediately after this command is completed.

FKEY. Add segment. A segment is added between two selected vertices at the ends of two curves. These two curves are combined into 1 curve.

LKEY. Select Linked. If you start with an non-selected vertex near the mouse cursor, this vertex is selected together with all the vertices of the

same curve or surface.
SHIFT-L. Deselect Linked. If you start with a selected vertex, this vertex is deselected together with all vertices of the same curve or surface.
MKEY. Mirror. Mirror selected control points exactly as for vertices in a Mesh.
SHIFT-R. Select Row. Starting with the last selected vertex, a complete row of vertices is selected in the `U' or `V' direction. Selecting Select Row a second time with the same vertex switches the `U' or `V' selection.

WKEY. The special menu for surfaces appears:
Subdivide. Subdivide the selected vertices
Switch direction. This will switch the normals of the selected parts.
ALT + MKEY. Mirror. Asks for axis (X,Y or Z) then mirrors the selected vertices

XKEY. Erase Selected. A PopupMenu offers the following choices:
Selected: all selected vertices are deleted.
All: delete everything.

VertexPaint Hotkeys

SHIFT-K. All vertex colours are erased; they are changed to the current drawing colour.
UKEY. Undo. This undo is `real'. Pressing Undo twice redoes the undone.
WKEY. Shared Vertexcol: The colours of all faces that share vertices are blended.

EditMode - Font

In Text Edit Mode most hotkeys are disabled, to allow text entering.

RIGHTARROW. Move text cursor 1 position forward
SHIFT-RIGHTARROW. Move text cursor to the end of the line.
LEFTARROW. Move text cursor 1 position backwards.
SHIFT-LEFTARROW. Move text cursor to the start of the line
DOWNARROW. Move text cursor 1 line forward
SHIFT-DOWNARROW. Move text cursor to the end of the text.
UPARROW. Move text cursor 1 line back.
SHIFT-UPARROW. Move text cursor to the beginning of the text
ALT-U. Reload Original Data (undo). When EditMode is started, the

original text is saved. You can restore this original text with this option.
ALT-V. Paste text. The text file /tmp/.cutbuffer is inserted at the cursor location.

UV Editor Hotkeys

EKEY. LSCM Unwrapping. Launches LSCM unwrapping on the faces visible in the UV editor.
PKEY. Pin selected vertices. Pinned vertices will stay in place on the UV editor when executing an LSCM unwrap.
ALT-PKEY. Un-Pin selected vertices. Pinned vertices will stay in place on the UV editor when executing an LSCM unwrap.

EdgeSelect Hotkeys

ALT-CLICK. Selects an Edge Loop.

FaceSelect Hotkeys

ALT-CLICK. Selects a Face Loop.
TAB. Switches to EditMode, selections made here will show up when switching back to FaceSelectMode with TAB.
FKEY. With multiple, co-planar faces selected, this key will merge them into one "FGon" so long as they remain co-planar (flat to each other).
LKEY. Select Linked UVs. To ease selection of face groups, Select Linked in UV Face Select Mode will now select all linked faces, if no seam divides them.
RKEY. Calls a menu allowing to rotate the UV coordinates or the VertexCol.

UKEY. Calls the UV Calculation menu. The following modes can the applied to the selected faces:
Cube: Cubical mapping, a number button asks for the cubemap size
Cylinder: Cylindrical mapping, calculated from the center of the selected faces
Sphere: Spherical mapping, calculated from the center of the selected faces
Bounds to x: UV coordinates are calculated from the actual view, then scaled to a boundbox of 64 or 128 pixels in square
Standard x: Each face gets default square UV coordinates

From Window: The UV coordinates are calculated using the projection as displayed in the 3DWindow