

Fun with Virtools!

Importing Basics

So we've discussed at some length in class the sorts of things you can import into Virtools. The upshot for now is: Polygonal Geometry and jpeg converted file textures. I also highly recommend reading the entries under "Maya Export" found in the Virtools help. There is a user guide to the export plug-in as well as a couple (sort of advanced) tutorials.

File types

.cmo – Virtools Composition file. This type of file will only open in the authoring environment. Allows you to create and edit Virtools content.

.vmo – Virtools web player file. Allows anyone with the Virtools web player browser extension to view and interact with Virtools content within a web page.

.nmo – Virtools Object file. This contains one or more Virtools objects with or without attached scripts (if you create one from Virtools). The export plugin to Maya also creates these files.

.nms – Virtools Script file. Saves Virtools VSL scripts. Not really covered in this class.

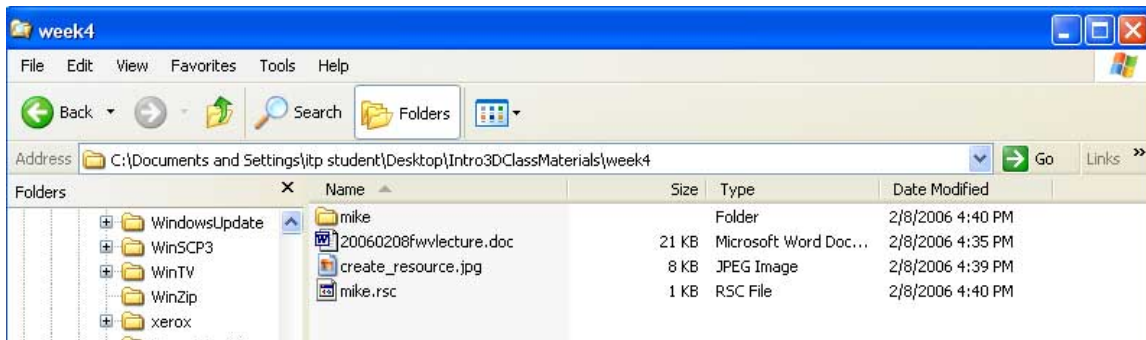
.rsc – Virtools resource file. A file describing the contents of a series of Virtools resource folders. Used for asset management. Speaking of which....

Resource Management

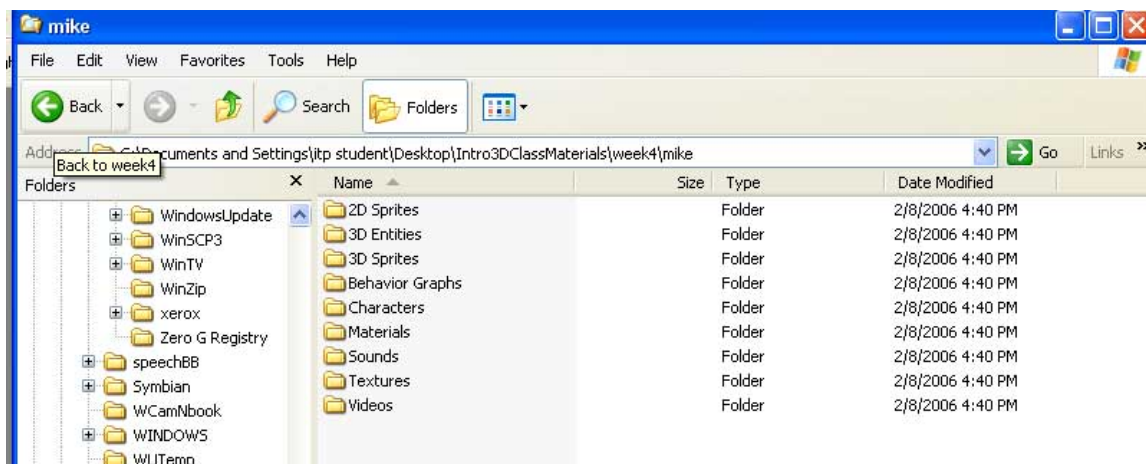
Virtools has implemented a simple system for managing your resources for use in creating 3D environments. To start this process, simply hit Resources-> Create New Data Resource .



You'll then get a file browse dialog wherein you can specify the name of your resource file and where to put it. Let's look at the files it creates:



Here we see there is a “Mike.rsc” file which is your actual resource file. You also see a “Mike” File folder which contains:



Folders for all the major classes of assets you might want to create your world. Some people find it helpful to have their content creation workflow save their assets into the appropriate Virtools Resource folders (IE Maya geometry -> 3D Entities, Photoshop textures -> Textures). If you do this then you can drag and drop assets into multiple .cmo's without having to separately import them all.

Scene Graph (3D Layout)

As indicated in our introductory lecture, the Virtools scene graph is very similar to Maya's (and all 3d scene graphs for that matter). While it is not a fully featured with respect to manipulation and layout, you can do many basic tasks involved in laying out a scene in Virtools, with perhaps the major exception that there's really no ability to create geometry out of primitives etc... So you'll see many Virtools tools parallel their Maya equivalents.

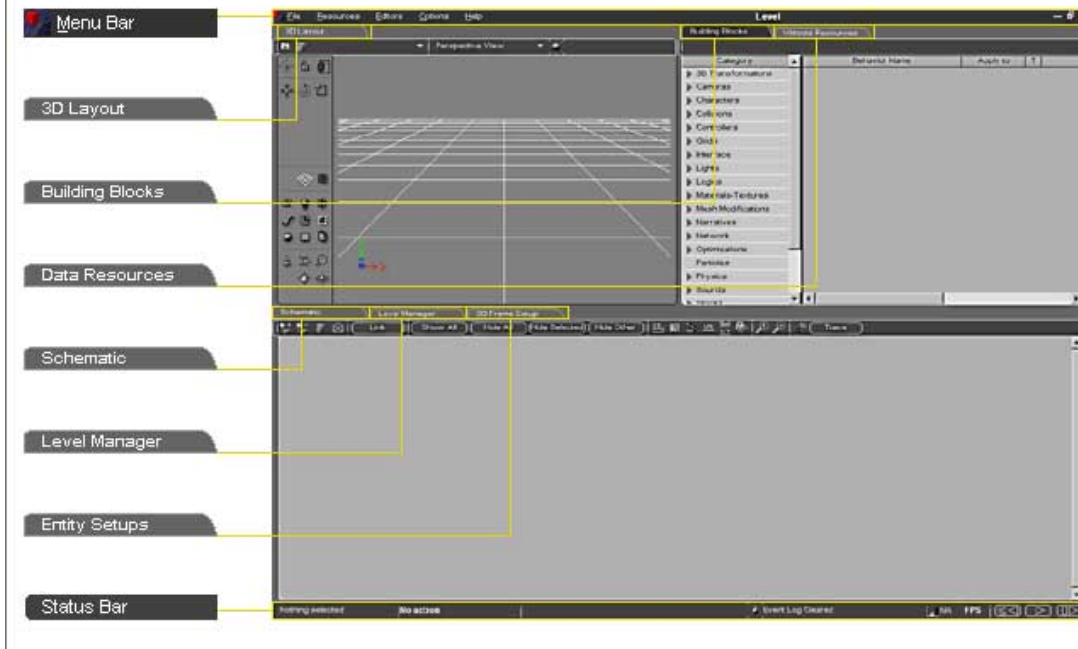
Most of the tools you'll use for working in the Scene Graph are aligned down the left side of Virtools' interface. From top to bottom, these are:

Interface

Overview

This section introduces and explains to you all elements of the Virtools Interface.

This section is divided in several parts corresponding to the **Menu Bar**, **3D Layout**, **Building Blocks Resources**, **Data Resources**, **Level Manager**, **Schematic** and **Entity Setups**, **Status Bar**.



View selector – lets you pick which camera or orthographic view you wish to see in 3D layout window. Next to it is an icon that allows you to bring up the Virtools Preferences.

Selection tools – Allows you to get back to pure selection mode, lock your selection, and change whether objects on your selection border get selected.

Move, Scale, And Rotate – similar to Maya. Note the space below which gets populated when you select one of these with items that allow you restrict transformations in specific directions or planes. N.B. scale is the first tool with a little triangle in its bottom right corner. This means that the tool has a variety of different modes, which you can see by clicking and holding on the button.

Reference Toggles – these two tools let you turn on & off the perspective guide and the screen guide.

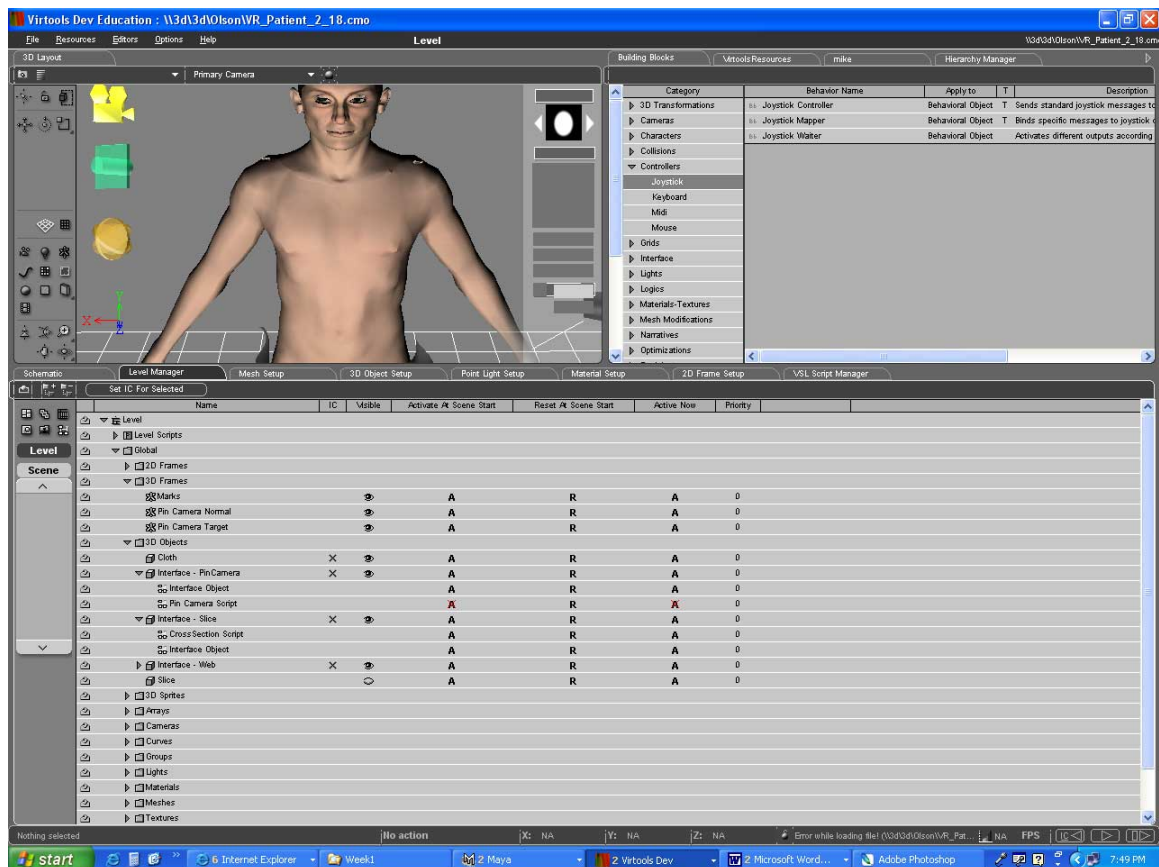
Create Panel – Lets you create cameras, lights, frames, curves, grids, 2d frames, materials, textures, portals (used in place management), and video objects. Note that the 3d objects among these get created at the location of the camera through which you're currently looking, so if you want to see what you've just created, simply dolly back slightly.

Camera Moves – These tools allow you to navigate around a scene with the active camera. Dolly moves your camera back and forth, FOV – changes your camera's field of view. Note this is not a good way to navigate as it will distort the perspective on the camera. Zoom, uh, zooms your camera. Note you can “zoom in on selection.” Pan moves the camera in the XY plane of your current view. Rotate by default (Saturn icon!) orbits the selected object which of course moves your camera in space. The other rotate mode rotates your camera around its own axes. It is important to note this difference in rotate mode.

Resources

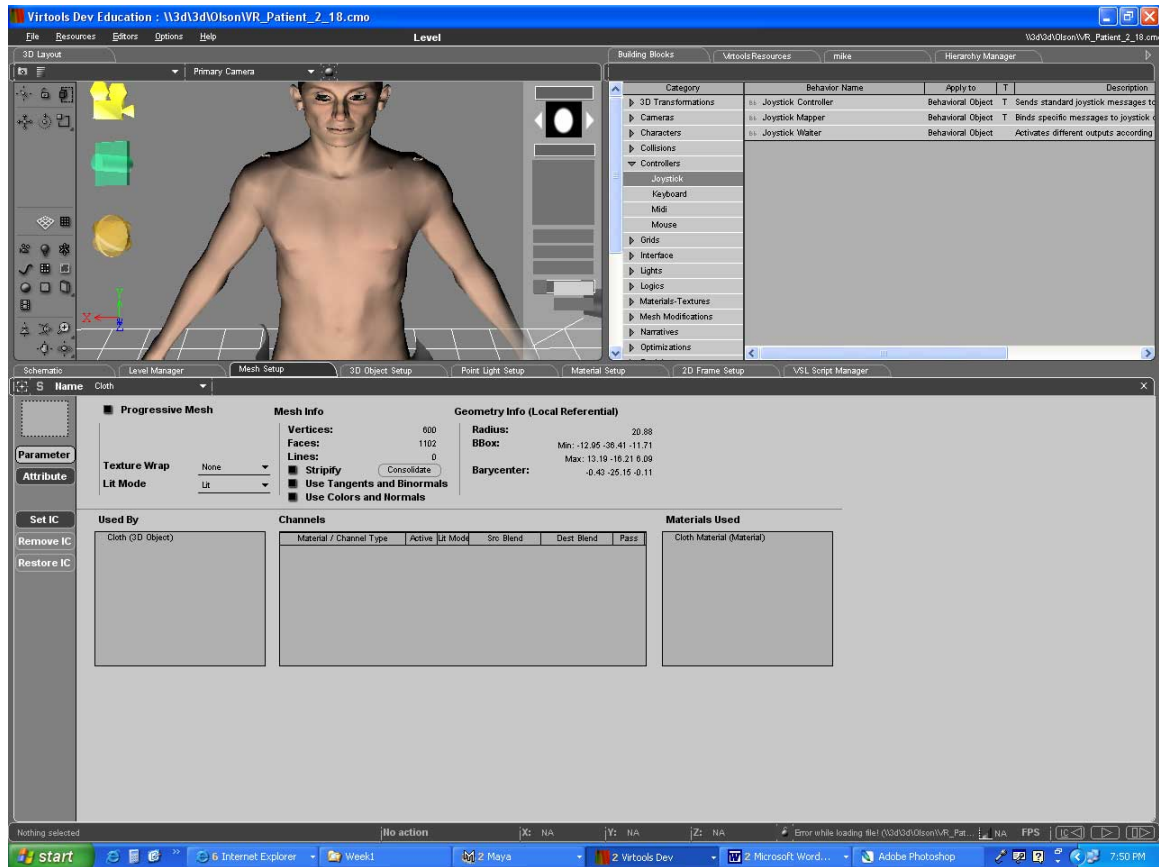
Next to the scene graph is a tabbed area that allows quick access to building block and any resource files you might import. Virtools will open with previously accessed resource files in this area, so feel free to close these if you don't need them. You'll also find that certain manager will open in this area such as the Hierarchy manager.

Level manager



The level manager contains a fairly comprehensive listing of all of the elements of you scene. It is very useful for selecting specific objects, and for drilling down into the details of how the various objects are created. It gives a quick interface for setting initial conditions for objects as well as showing / hiding them. You can access both the fundamental parameters for an object as well as its behavioral attributes by double clicking on an entry which will bring up:

Object Setups



By double clicking an item in the Level manager, you can access and modify its normal parameters. You can also open up other items in your scene by double-clicking the entry in the “used by” box for example. Note also that you can also access an objects ATTRIBUTES from here which will be discussed later.

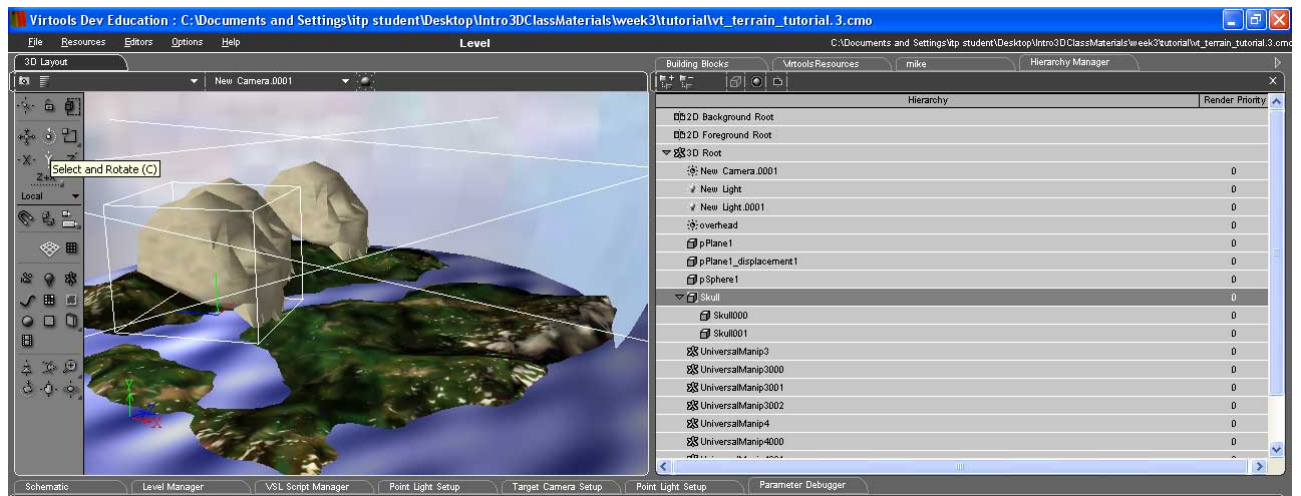
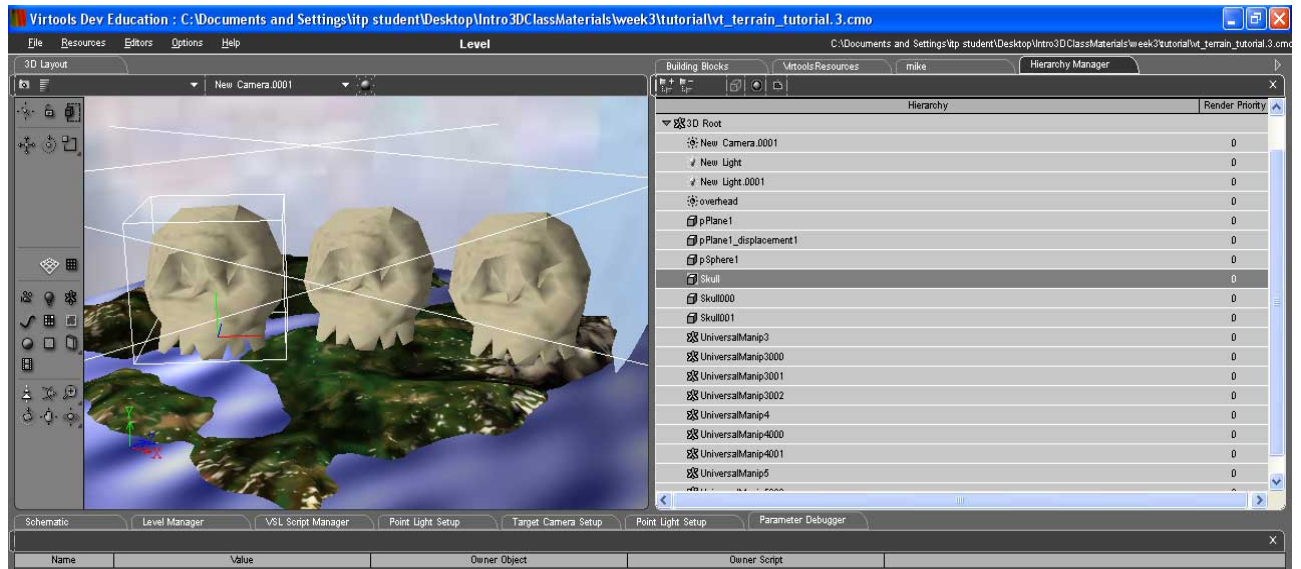
Initial Conditions

Initial Conditions is a very important concept in Virtools. Essentially it allows you to freeze any Behavioral object in a certain state to which it will revert when you hit the “rewind” button in Virtools. So, for example, if you apply a building block that translates a specific object, if you want to be able to snap that object back in place every time you start the scene, you need to have set its “IC.” Often you’ll want to move things around in your scene at first, but at the point that you have your scene more or less set up you should “set IC” on pretty much all the 3D entities (and any other dynamic object). If you then decide to make changes later, you can always set IC again on the objects you’ve adjusted.

Hierarchies

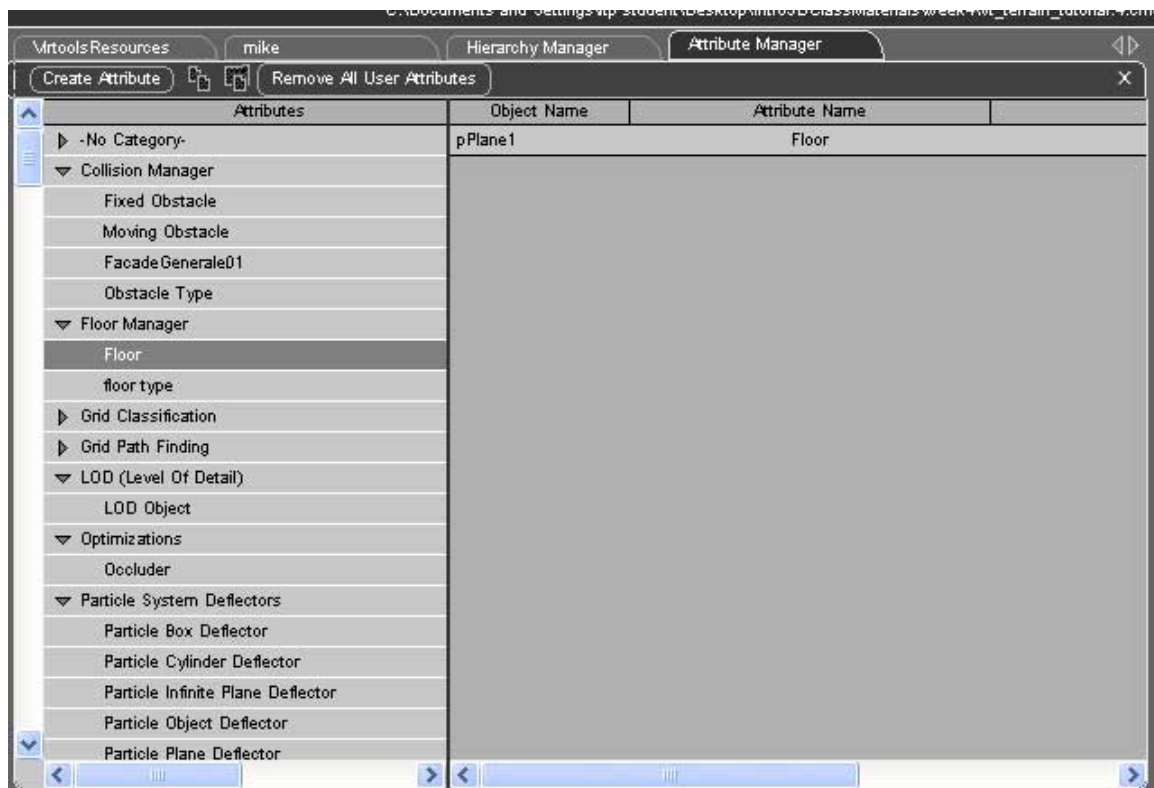
Hierarchies are important in establishing relationships between groups of objects in your scene. For example, if you were simulating a planet with a moon rotating around it, you would want to make the moon a child of the planet, so that no matter

what happened to the planet, the moon would continue in it's orbit. The Hierarchy manager provides an easy drag and drop interface to set up these relationships. You can demonstrate this principle to yourself by importing 3 skulls from the Virtools Resources 3d Entities -> 3d Objects section. Note that if you translate or rotate one, nothing happens to the others. If however, you open the Hierarchy manager from Editors -> Hierarchy Manager, and drag the entries for 2 of the skulls onto the third, you'll see that when you move and rotate the third, the other two go with it. This is the essence of the parent-child relationship. That, and unconditional love.



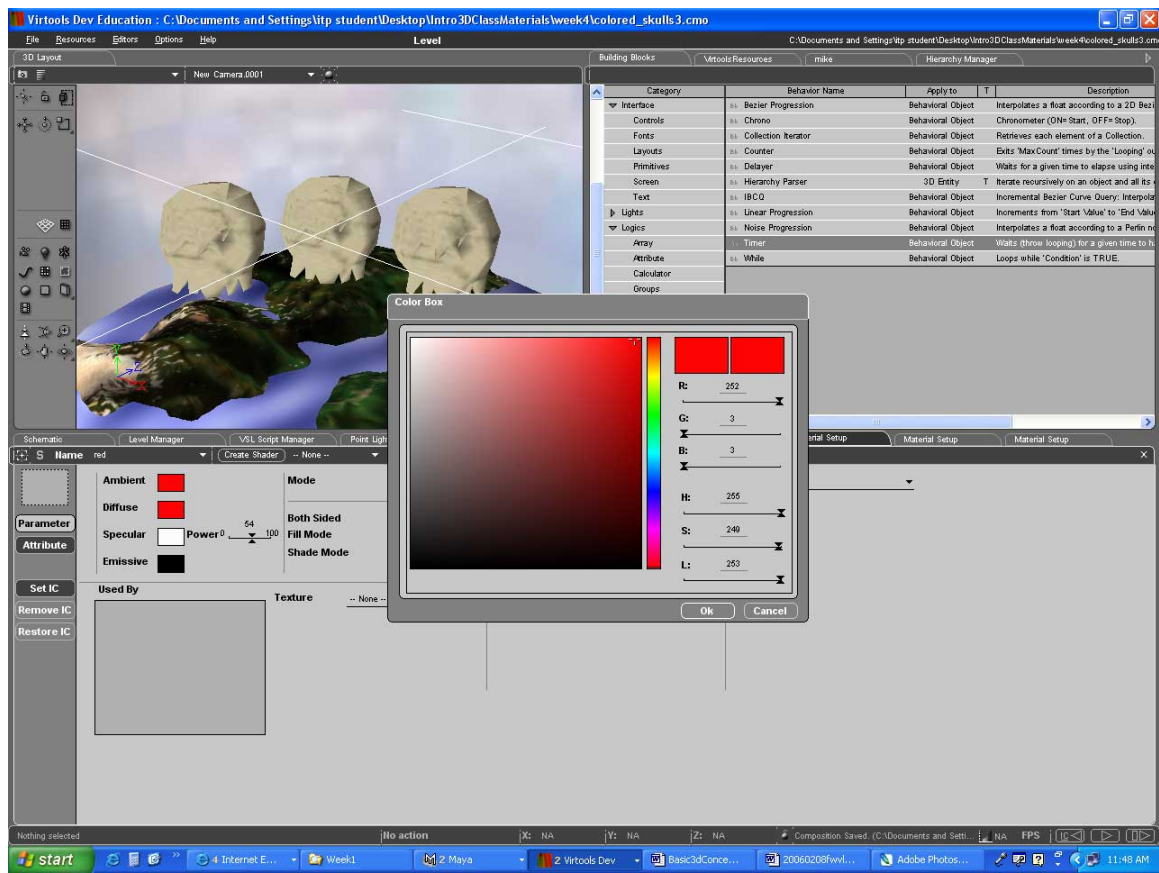
Attributes

Attributes are somewhat different from an objects basic parameters. Generally attributes are used by Virtools' Behavior Engine in conjunction with certain Building Blocks or processes. For example, you have to assign the attribute "floor" to at least one object if you want to use the "keep on floor" building block.



Groups

Another way you can organize Virtools objects is by grouping them. For example create a couple new materials for your scene by clicking the "Crate Material" button in the create bar. It will bring up a "set up" page for a new material. Change its diffuse color to red by clicking the grey rectangle next to "Diffuse." Hit F2 and rename the material "Red. Then make a White and a Blue material.



Now, go back to the level manager and look for the little create icons at the far left of the interface just below the “Schematic” tab. These are create icons for items that do not show up in the 3D layout view. Find the group icon, and hit it to create a group. Name the group “colors. Then LMB-drag your colored materials onto the “colors” group line entry in the level manager. You should then see something like this:



Being able to cycle through groups of objects is an important skill we’ll use a great deal.

Scripts

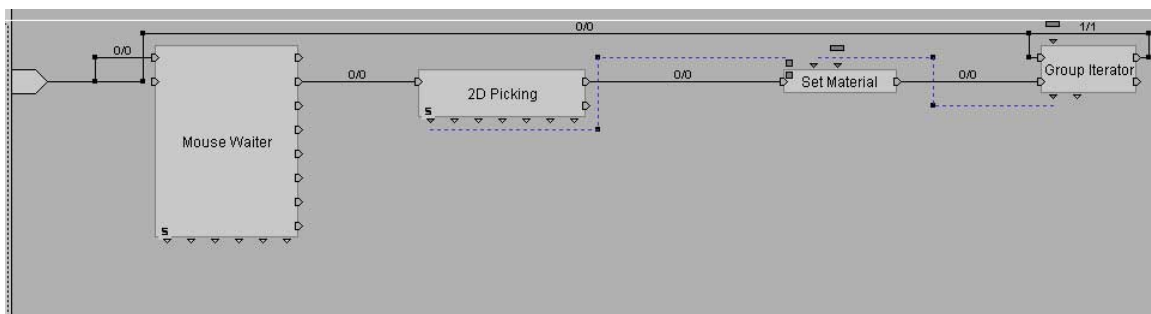
Objects in Virtools can have behaviors, which cause them to interact in various ways when you “play” a scene. These behaviors are represented in the Virtools Schematic as scripts.

Scripts tend to be arrangements of building blocks. Building blocks have Behavioral Inputs on the left, which typically turn them on or cause them to execute one loop. Behavioral outputs get triggered when the BB finishes what it's doing. Parameter in's allow you to feed data to your BB, and Parameter outs allow the BB to spit out data.

We've already seen some simple scripts in action such as simple key waiters, translates, and keyboard camera orbits. Let's look at another form of user input.

Right click on the “Level” entry in the level manager and select “Create Script” from the shortcut menu. Then switch to the schematic view and drag the “Mouse Waiter” building block from Controllers -> Mouse. You'll see this building block has a number of outputs corresponding to your basic Mouse Events such as “Left Button Down.” Let's say that we want to change the color of a skull based on the user clicking on it. So we can grab mouse clicks with the Mouse Waiter, the next thing we need to do is determine which skull is getting clicked. Luckily Virtools has a building block that does just this. Drag in the “2D Picking” building block from Interface -> Screen. This building block tells you which 3D entity or 2D sprite the mouse is over when the building block is activated. Then based on this information we can change the material of the selected object with the Materials-Textures -> Basic -> Set Material building block. So drag this BB in as well. Finally perhaps we want to change the color on successive mouse clicks. To do this we will use the group of materials we created earlier and the Logics -> Groups -> Group Iterator BB. Note that the “Logics” section of the BB manager is very important for basic programming functions.

Now that we have our building blocks we need to connect them in such a way as to achieve our desired effect. Examine the arrangement below:



Program Flow

The program flow is fairly straightforward with three exceptions. 1) we want to start the group iterator at the beginning of the program (or when we get an LMB) so it will have the first member of the group ready for when we want to do something with it and 2) We need to loop the group iterator back on itself when it is done iterating through the group so that it will start again with the first member 3) we want “Set

Material” to hit the “loop in” input of the group iterator so that it will advance to the next member as opposed to starting over.

This brings up an important point. Building blocks in Virtools come in 3 basic flavors:

Single Action – these BB’s have only 1 B-in and complete their processing within 1 frame. Most BB’s are like this including 2Dpicking and set Material Above

Internally Looped – these building blocks are activated and remain active until they are turned off. Examples include most “waiter” type building blocks including the Mouse Waiter above.

Externally Looped – these building blocks are activated, and then depend on external input to advance through their processing, Typically they will have a standard B-in and then a Loop-In, which gets activated according to some external process loop.

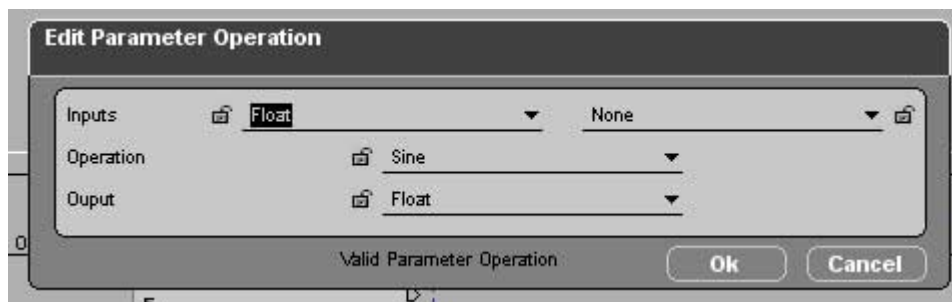
Note there are also hybrid BB like Bezier Progression that combine internally looped and Externally Looped behavior. More on that later.

Data Flow

The data flow above is fairly simple: 1) the 2d picking BB sends the object picked to the Set Material BB which 2) refers to the current output of the Group Iterator to determine which material should be applied. When the Set Material is done, it tells the Group Iterator to go to the next item in the list.

Parameters and ParamOps

In many cases, you’ll want to manipulate data as it travels through your program flow. Virtools provides a convenient way to do this with its Parameter Operations functionality. You can add ParamOps to a script by simply right-clicking in the script area and selecting “Add Parameter Operation” or by hitting alt-P . This will then bring up a little dialog that asks for input types, an operation, and an output type. The dialog will tell you when you’ve selected a valid combination. You can also look through the list of valid ParamOps in the help according to various sorting criteria.



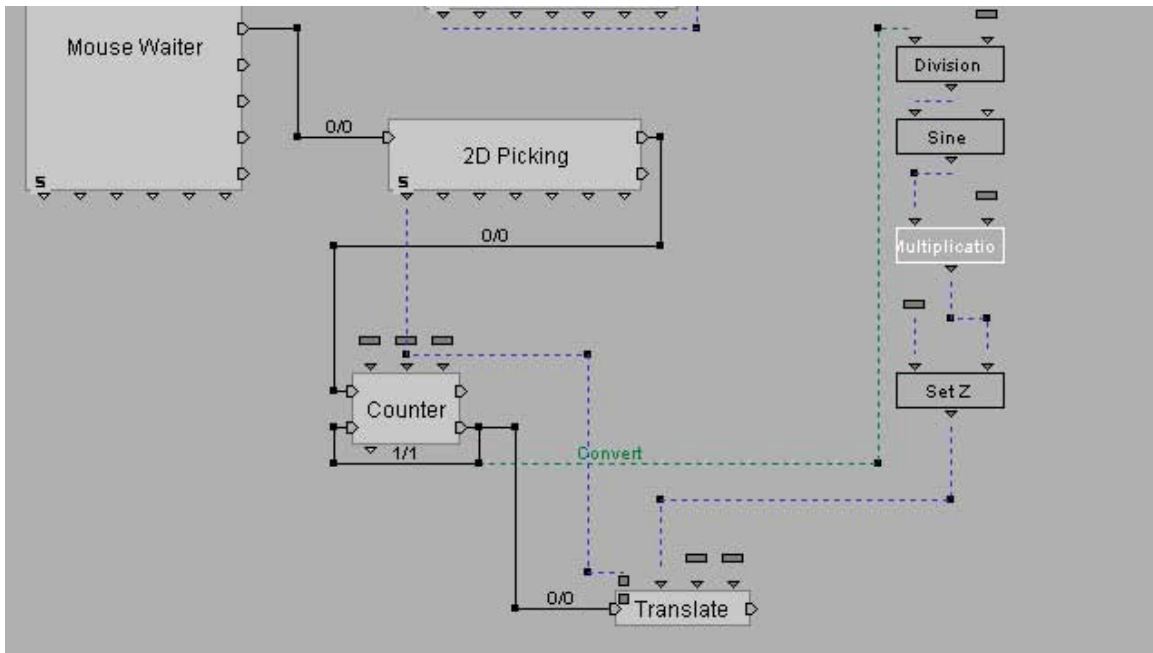
You’ll then see a little box with two inputs and an output distinguished by the name of the operation you selected Like those depicted below.

Let's say we wanted to add a little procedural animation to our skulls whenever we right clicked on one of them. There are a wide variety of ways to accomplish this, so let me suggest a paramOps intensive one for the time being.

First of all we'll need to capture the user's RMB, which we can do with our existing Mouse Waiter. Then we'll need to determine which skull was selected using another 2D picking BB. And finally we'll need to effect some change on the skull. Let's use its position for the time being, and so drag in the 3D Transformations -> Basic -> Translate.

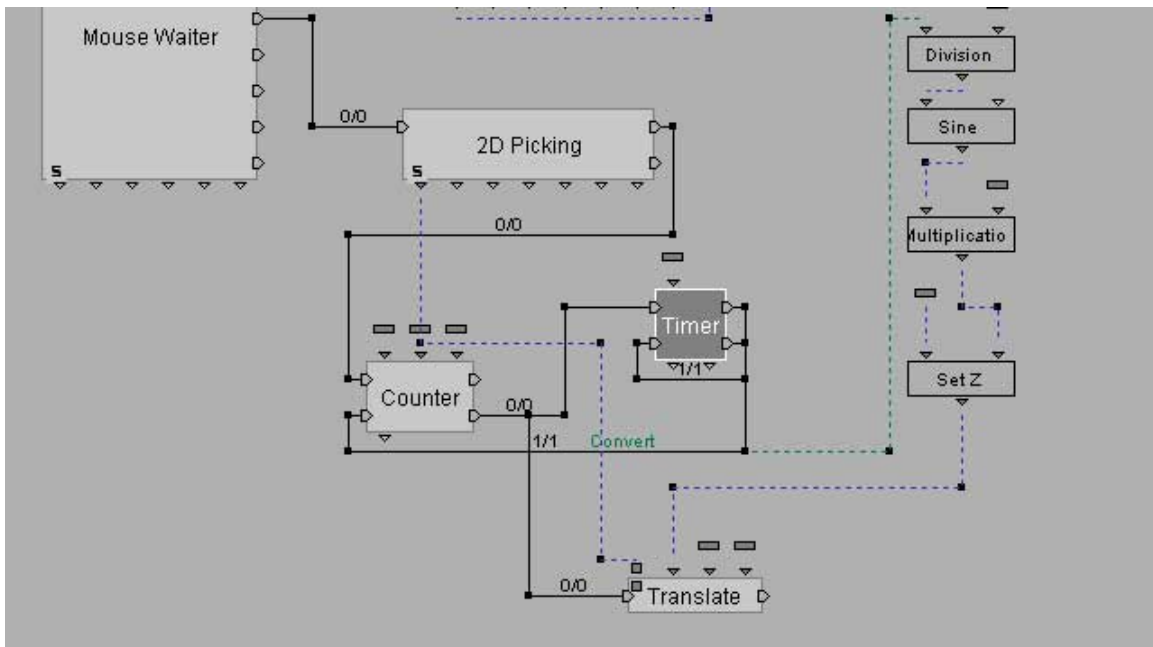
Loops

In order to trigger a "Procedural" animation (one produced quantitatively in Virtools as opposed to exported from Maya) from a single click as opposed to holding down a key so that the input is captured on every frame, one generally will need to use some sort of looping BB. Let's start with perhaps the simplest: from Logics -> Loops -> Counter. This BB allows you to simply specify an number up to which to count and the "step" by which you want to get there. Let's say we want our skull to sort of move back and forth, but eventually return to its original position. This suggests a sort of sinusoidal motions. So perhaps we'll want to count up to 360 degrees while moving the skull along the z axis according to a sine function. To do this is a bit complicated. First of all, Virtools ParamOps tend to work with radians not degrees, so the first thing we'll do is take the output of our counter and divide by 2π or 6.28 . Then we'll run that float through the sine function to get a movement amount. If we want to be able to easily scale this movement, we'll multiply the result of the sine function by a certain amount. This will be the amount by which we translate our skull, which should fluctuate from positive to negative values. However, by looking at the help, or the BB itself you'll notice that the Translate BB needs a vector, not a float, which necessitates us 1) Adding a local parameter of type vector (0,0,0), and 2) connecting it to the ParamOp "Set Z" which will take this vector and set its z value to the result achieve above. This can then be fed into the translate, so that your final script will look like this:



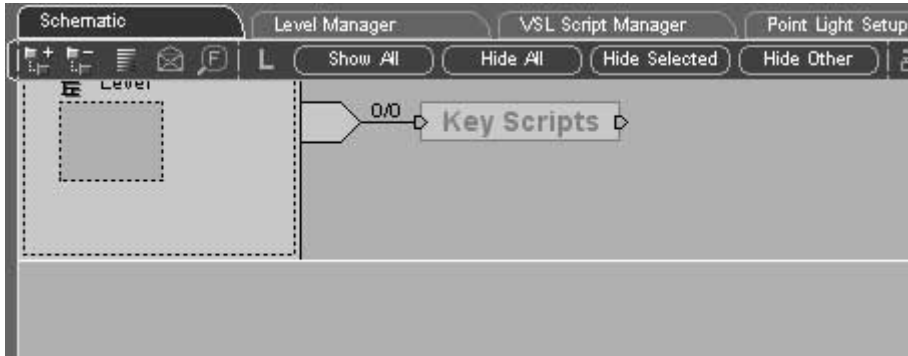
Timers

If you test the above script, perhaps you think the animation happens too fast. It's easy to add a delay to any process loop by adding the Logics-> Loops -> Timer BB. This simply counts up to a certain amount of time and then activates it's b-out. We'll put it between the counter loop out and loop in to control how quickly the counting happens. Note that you have to loop the timer on itself. Now you can adjust how quickly the animation executes.

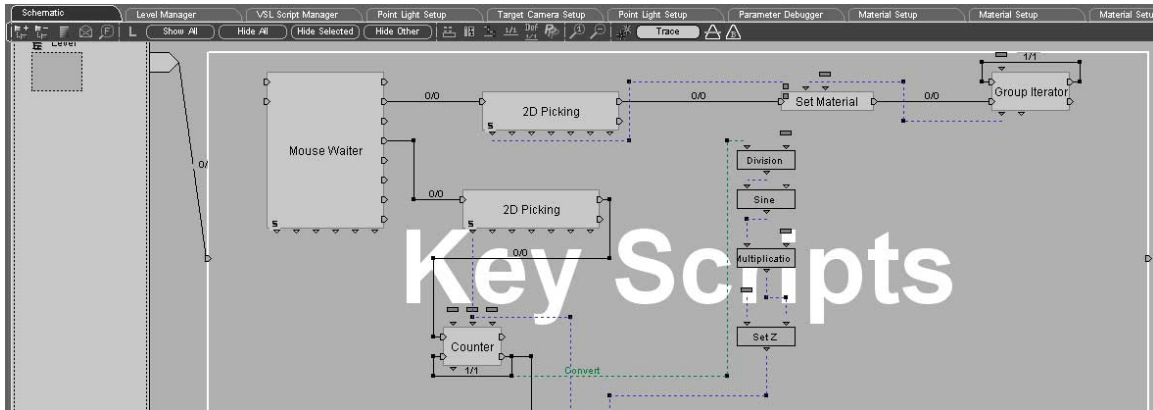


Behavior Graphs

Now that we have a sort of large and tangled script working the way we want, sometimes it's nice to be able to “file away” some of the clutter. One way Virtools provides to do this is by drawing behavior graphs. You can do this by either right clicking in the schematic area and selecting from the shortcut menu “Draw Behavior Graph” or by hitting the “g” key. This will allow your cursor to draw a selection box in the schematic and encapsulate part of your script. If you then double click inside this area the script will collapse into what looks like a single BB.

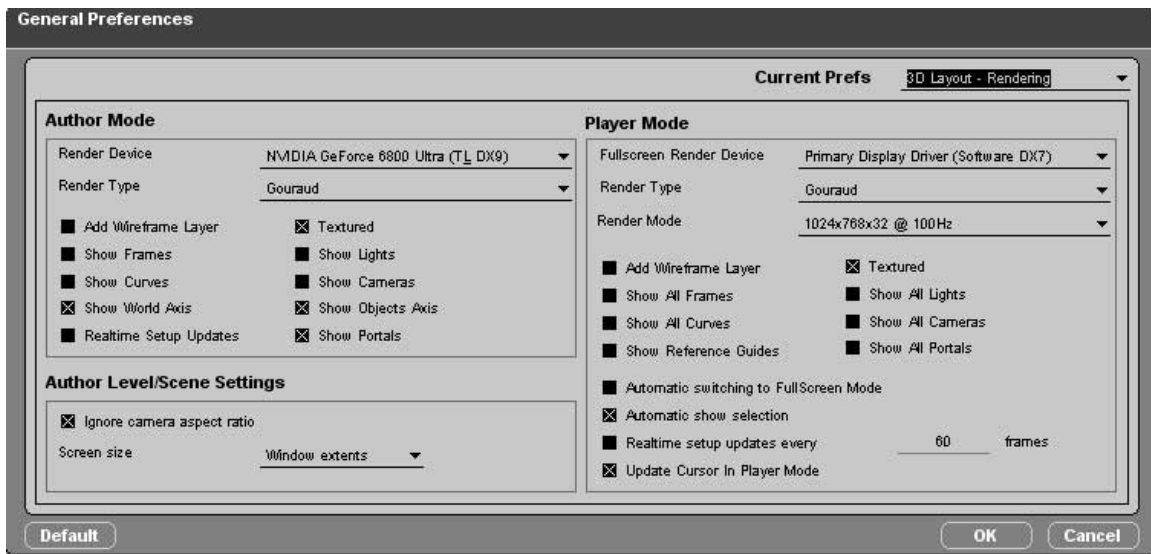


You can rename it, and then double click again to re-open it.



Preferences

Another element of working with Virtools that can be helpful is the ability to effectively use the Virtools preferences. You can access them by hitting control-P or by selecting options -> General Preferences .



Note the primary thing you'll do with this dialog is select which elements of your virtual scene you want to see both in the Author Mode and in The Player mode. For example often its tedious to work with all the lights, frames, and cameras you might have showing. Note also the "Current Prefs" selection box in the upper right. This will get you to several other preference menus.