

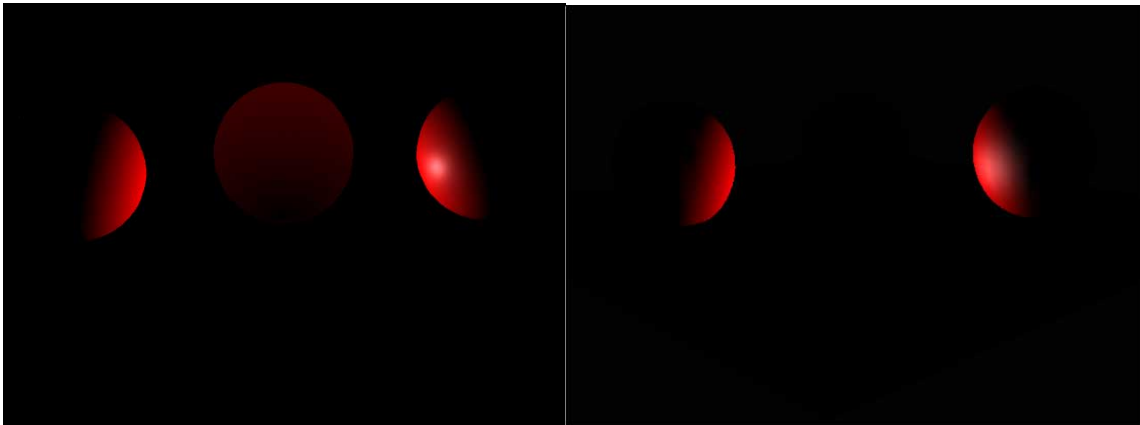
Lights, Camera, Action

While they may not be the primary actors in your mesmeric virtual environments, much as in the film world, good lighting and camera work can be the difference between brilliance and dreck.

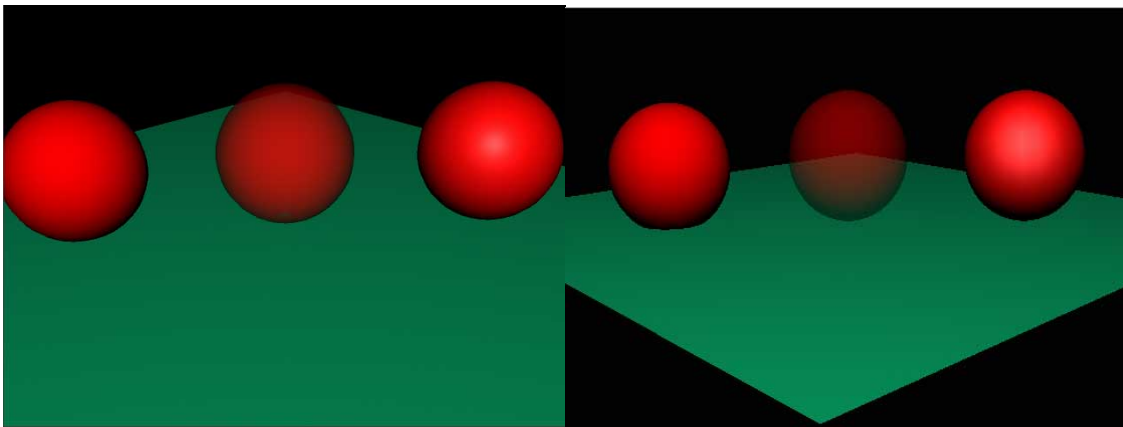
Types of lights

Point

A point light is the simplest kind of light. It simply radiates light out equally in all directions. The only real parameters in Maya you need to worry about are the light's color and intensity. These export to Virtools pretty well with the intensity value translated to "constant attenuation." Below are examples of simple point light in rendered in Maya [left] and then Virtools [right].

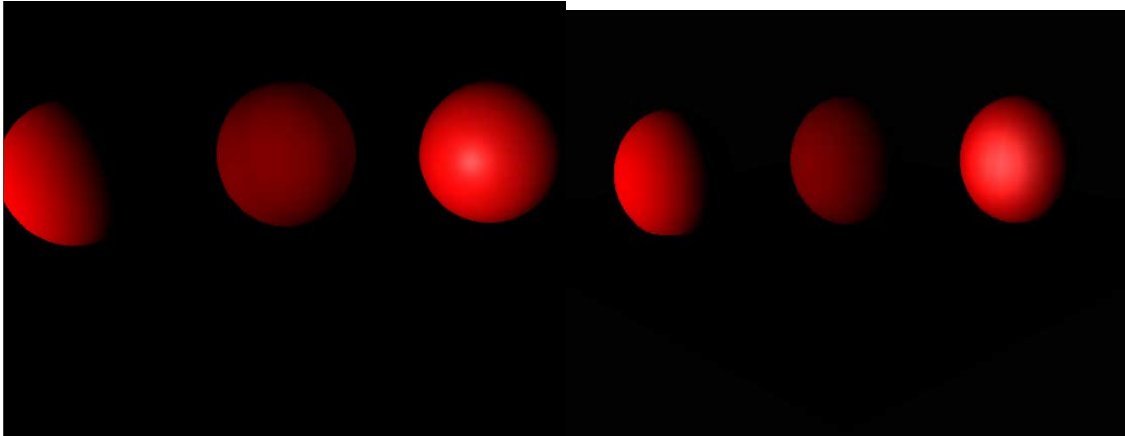


Above the point light is at the origin. You'll notice the middle sphere doesn't show up at all. This is because Virtools' rendering engine doesn't render polygons whose normals point away from a scene's only light source. If we move the point light then you'll see that they render quite similarly (though the camera angles are somewhat different).

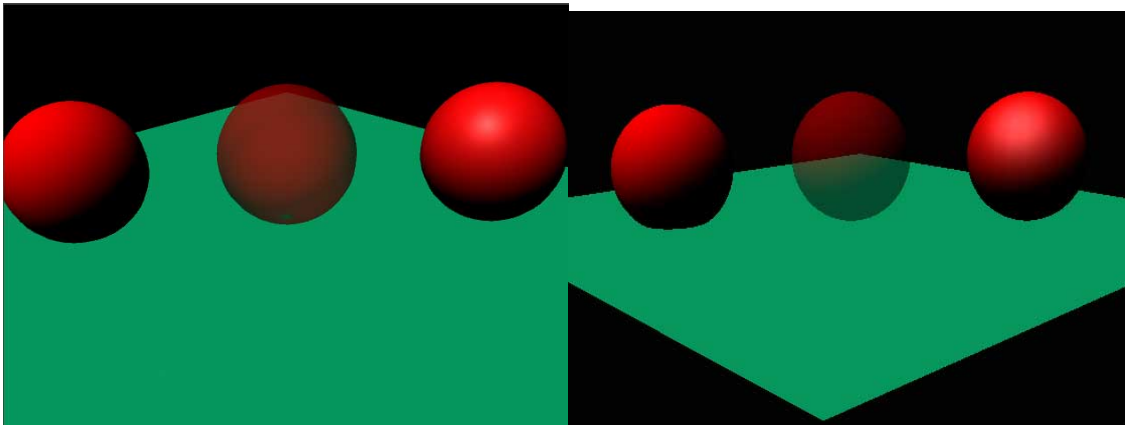


Directional

Directional lights produce parallel light rays across an entire scene. A common analogy is to sunlight (on earth) on a sunny day (despite the fact that the sun is more like a giant point light), since it seems to be coming from a pretty uniform direction. Again directional lights export pretty well from Maya.

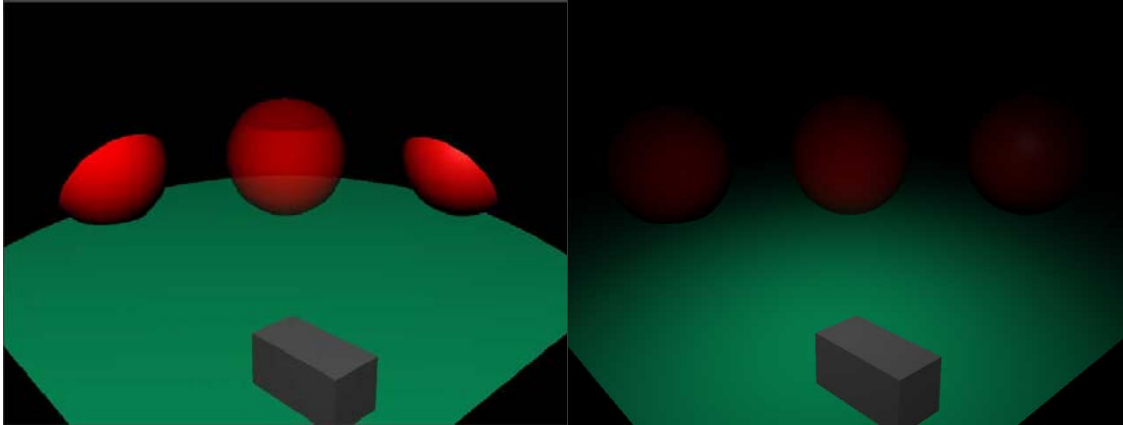


Here our directional light is oriented along the z axis. To get a better sense of the scene, I rotated the light 45 degrees.

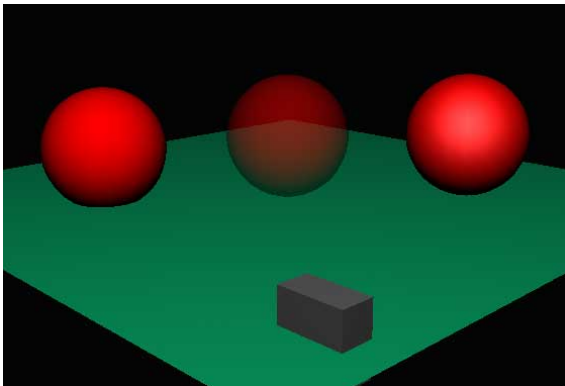


Spot

Spotlights are considerably more complex. Essentially they are a directed light source coming from a point with various focus options. For example, in Maya, here is our scene with a default spot light rotated 45 degrees in x and -45 in y and moved 10 units in each direction. Next to it is the same light with various "focus" parameters adjusted. The best way to familiarize yourself with these is just to play with them.

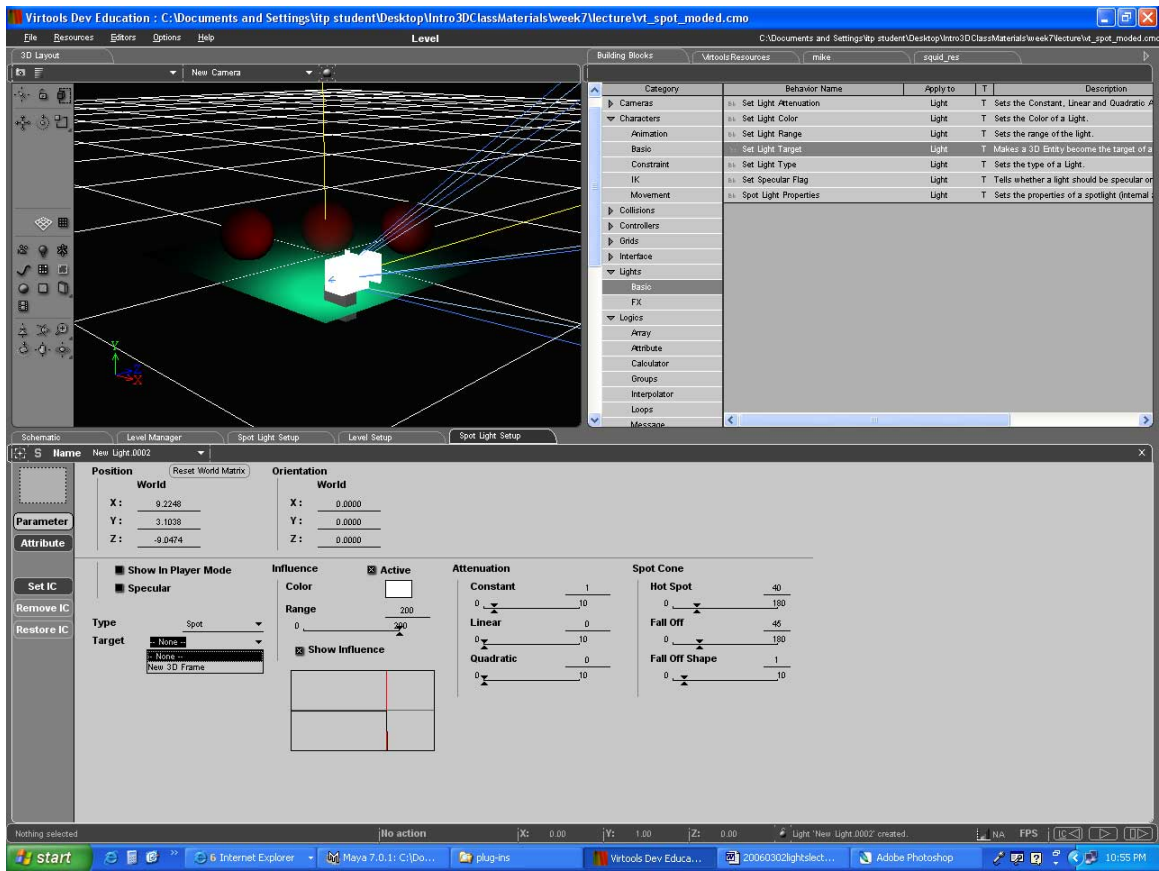


While spotlights DO export to Virtools, I don't recommend that you use them, since when they come into Virtools, a number of evil things can happen. Below for example, is the identical light in Virtools. Some of the spotlight attributes seem to come through (IE "cone angle" seems to go into "fall off") but Virtools' spotlight model seems somewhat different than Maya's. Beyond that, spotlights that come in from Maya appear to lack a significant amount of functionality possessed by their native Virtools counterparts. For example, it doesn't seem like they can be targeted.

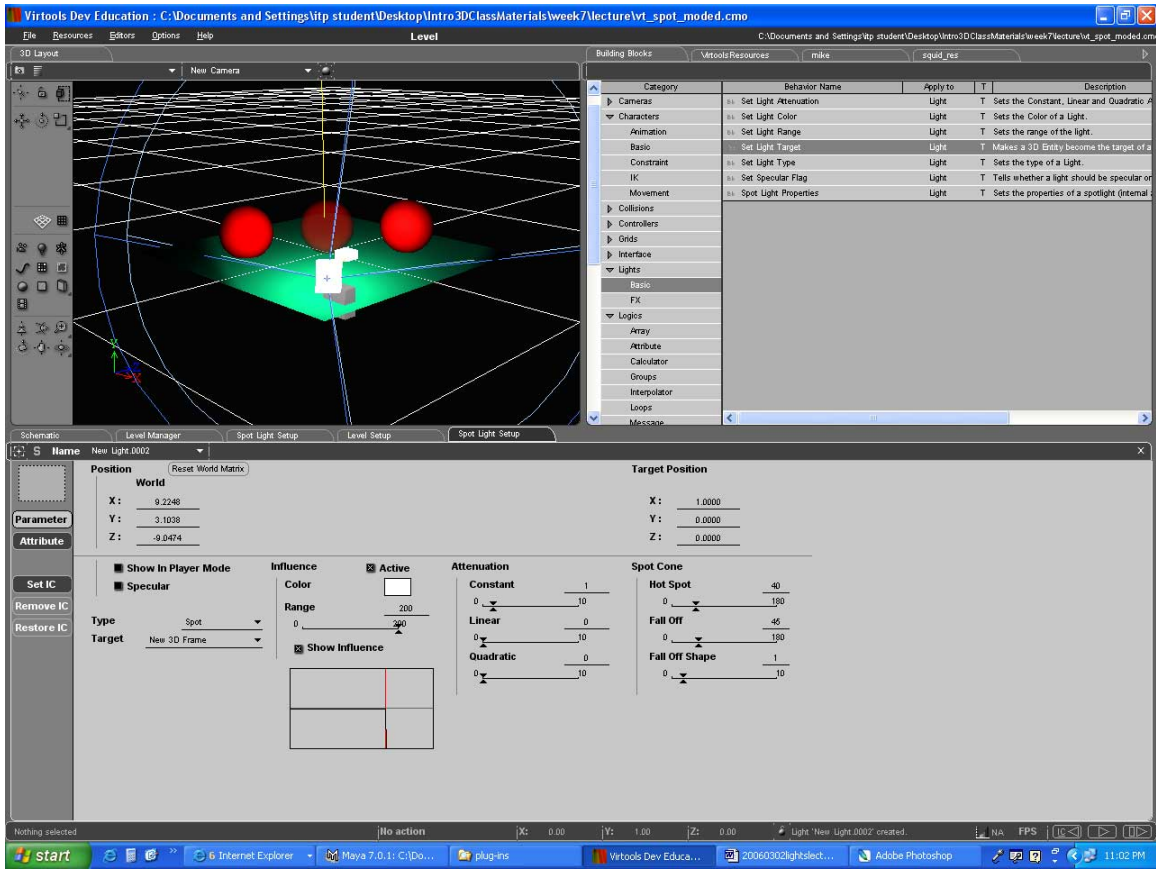


Targeting your Spot

Often you might want a light to continuously aim at a particular object or place regardless of whether either the light or the target moves. To demonstrate this in Virtools, start by creating a frame, and then create a light.



Set the Type Parameter for the light to be "spot," and select your new frame to be its target. You should see the camera immediately reorient itself to point at the frame.

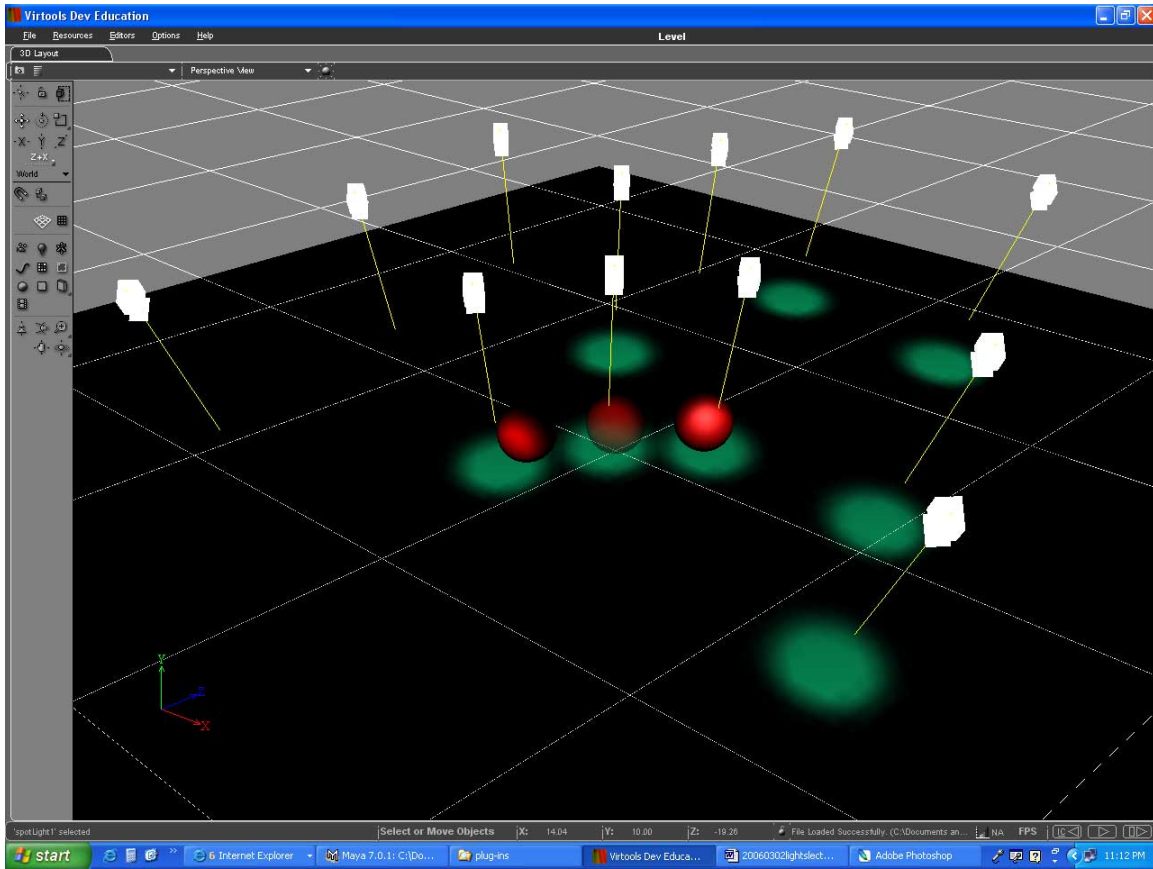


This frame could now be animated to produce a searchlight effect.

Other lighting issues

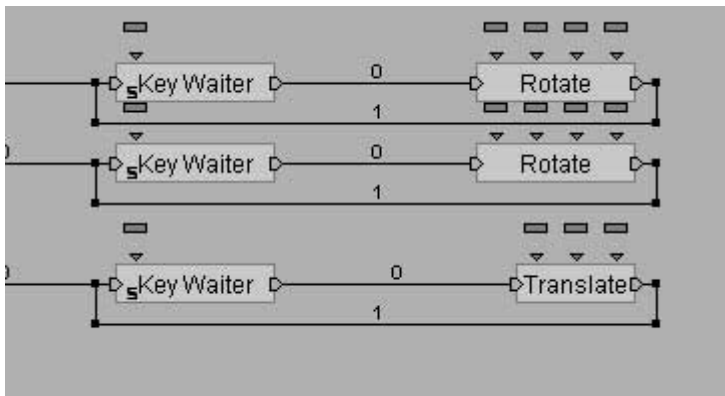
Rendering lights can be very processor intensive. While our new graphics cards don't necessarily place any limits on the total number of lights in a scene, some older cards limit you to 8 active lights. For example, my desktop has such a limit.

In general, it's a good idea to use lights fairly sparingly, both from a performance and hardware compatibility perspective.

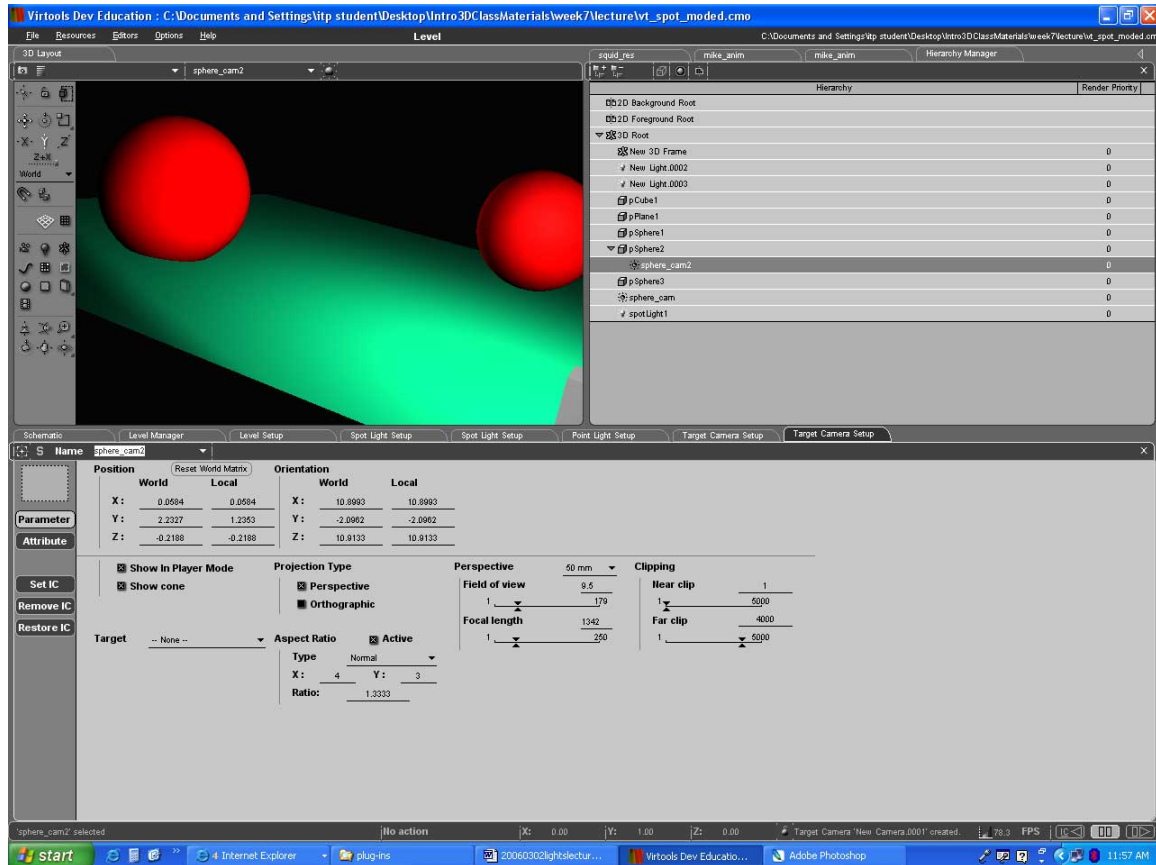


Follow Cameras

Often in designing a 3D experience you will want to assign a camera to follow a character around a scene. To experiment with various ways of accomplishing this, let's enable one of our sphere's for motion by adding key waiters and basic transformation BB's to enable move forward and rotate left and right. The script is below, but make sure on the translate bb that you set the Referential to be the sphere you're trying to move. Set IC on this sphere as well.

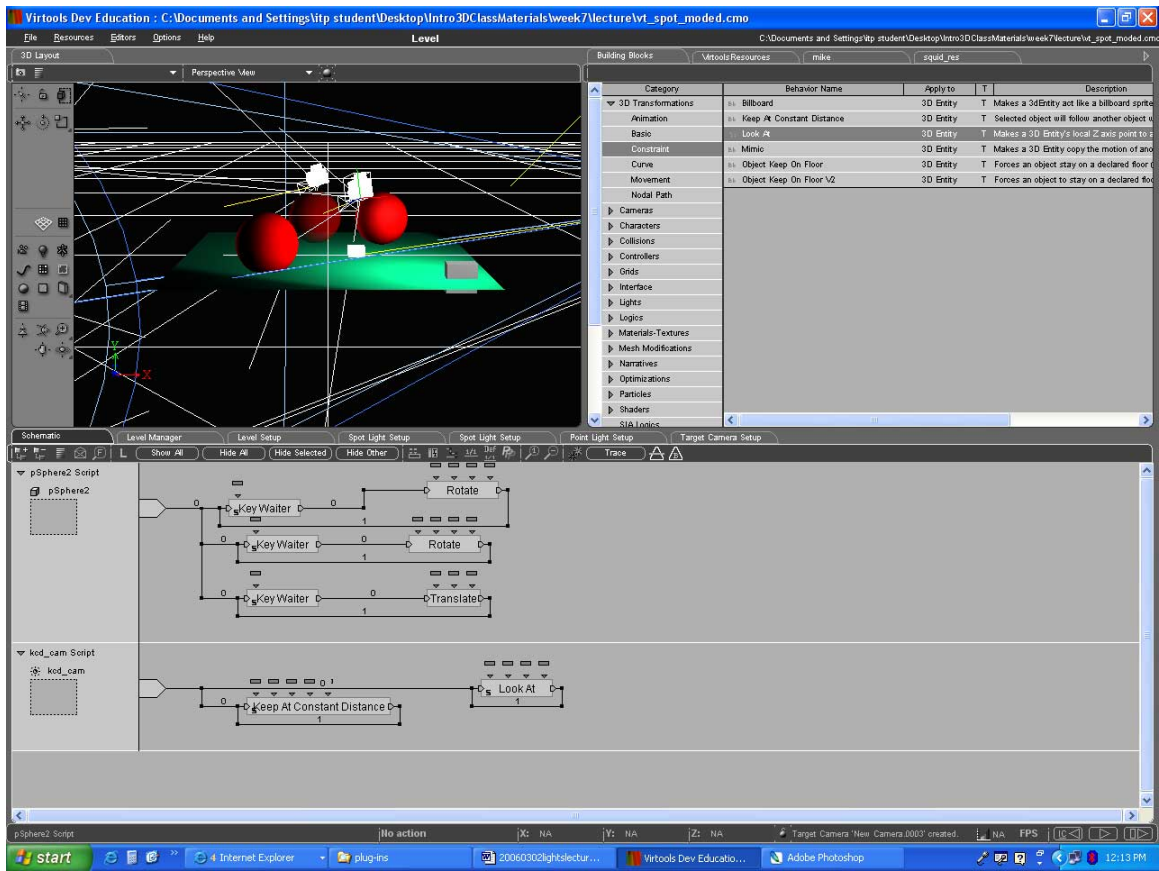


Now let's try perhaps the easiest thing, simply creating a camera that will be the child of the sphere. So create a new camera called sphere_cam. Position it in the orthogonal views so that it is either just above, or inside, but sticking out of your mobile sphere. Now hit Editors -> Hierarchy Manager and drag the entry for your sphere_cam onto the entry for the sphere you selected. Be sure to set IC for your sphere_cam at this point. Now if you set your view to be the sphere_cam, and move about your terrain, you should see things from the perspective of the sphere.



You can achieve a similar effect with the 3D Transformations -> Constraint -> Keep at Constant Distance BB. So let's create a new camera called KCD_cam. Move it somewhere near the moving sphere. Then create a script for the camera, and drag the KCD BB onto it. Set the position to (0,1,1) (this will initially put the camera above and behind the sphere for an "over the shoulder" view), the referential to be the moving sphere, and the distance to be 5. If you try it now, you'll probably be disappointed with the effect. To improve it, use the 3D Transformations -> Constraint -> Look At BB with the referential = your moving sphere. This will force your camera to always look at the moving ball. Both of these BB's need to be externally looped.

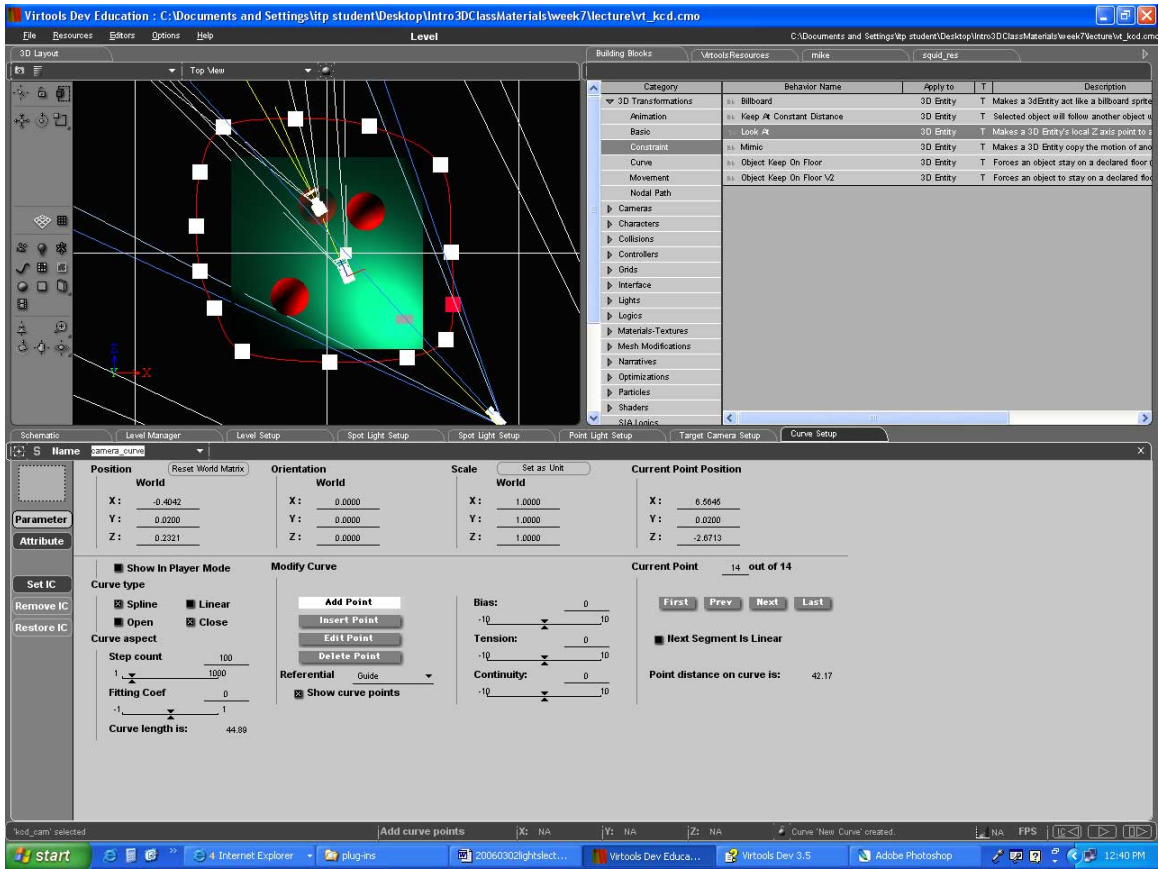
You could alternatively position a frame within the sphere, make it a child of the sphere, and set that frame to be the camera's target.



Path cameras

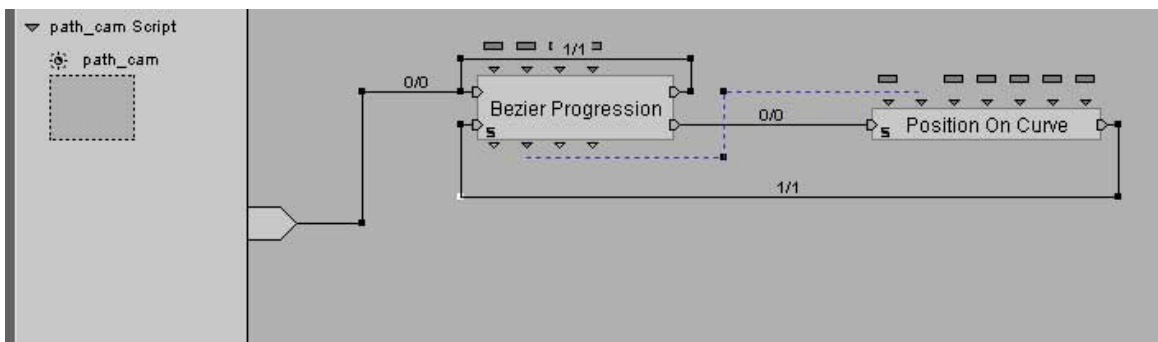
Another thing one frequently wants to do with cameras (and other things) is to set them in motion along a predefined path. Luckily, Virtools handles this pretty easily.

Let's begin by creating a somewhat circular path around our scene. Switch to the scene's top view. Then, to the left of the 3D Layout window, click on the "create curve" tool. You can then begin clicking in the 3D layout to create points on a curve. Select the "closed" option in the curve setup, and click out points that roughly orbit around the scene's platform.



You can then switch views and move the whole curve slightly upward. Now create a new camera called path_cam, position it along the curve facing inward, set its IC, and create a script for it.

Drag the 3D Transformations -> Curve -> Position on Curve and Logics -> Loops -> Bezier Progression BB's into the script. And connect them as follows:

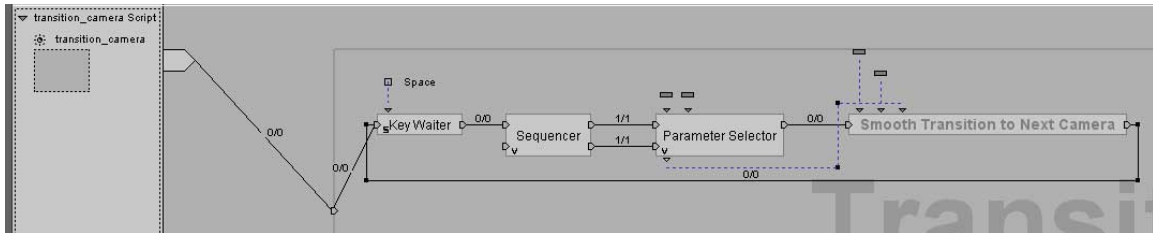


Set the duration parameter in the Bezier progression to be 15 seconds. Connect the BP's Value Out to the POC's Progression Pin. Set the curve in the POC BB to be the curve you just created. Now when you play, you'll see the camera begin to follow the curve, but for much of this time, you won't see anything because the camera's orientation doesn't change. To fix this either create a frame to serve as its target, or use the Look At BB as we did above. I set it to look at my moving sphere.

Transition Cameras

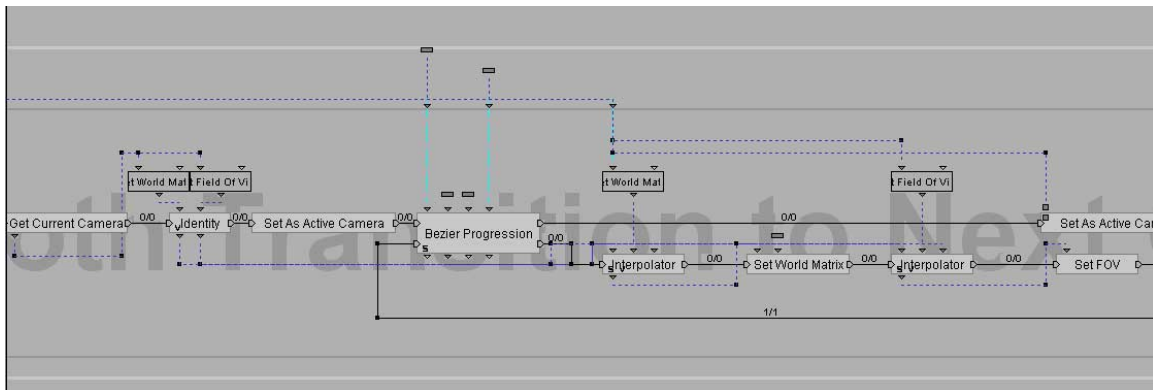
Another interesting camera effect you can use is called a “transition camera” this behavior interpolates the position of 3 cameras (A,B, and transition) such that there is a smooth transition between the views of cameras A &B.

Let’s begin by creating a new camera called “transition_cam.” Set its IC, and create a script for it. Right click in the script and select “import behavior graph” from the context menu. Browse to the Class 7 lecture folder and choose Transition Camera.nms. Double click on the behavior graph to open up its first level. You should see:



Note the key waiter, which looks for user input to trigger the transition. You can substitute some programmatic condition for this. Next there is the Parameter Selector which chooses to which of the cameras you’ll be transitioning. Select for example, your path_cam and your sphere_cam.

Now open the Smooth Transition... Behavior Graph.



This is a fairly complicated script, but suffice it to say, what we’re doing here is basically interpolating both the world matrices (which includes position and rotation values) and fields of view for our two cameras along a Bezier Progression, and setting these interpolated values as the data for the transition camera along each step. The best way to see this effect is just to try it out.

Now let your inner Orson Welles go bananas.