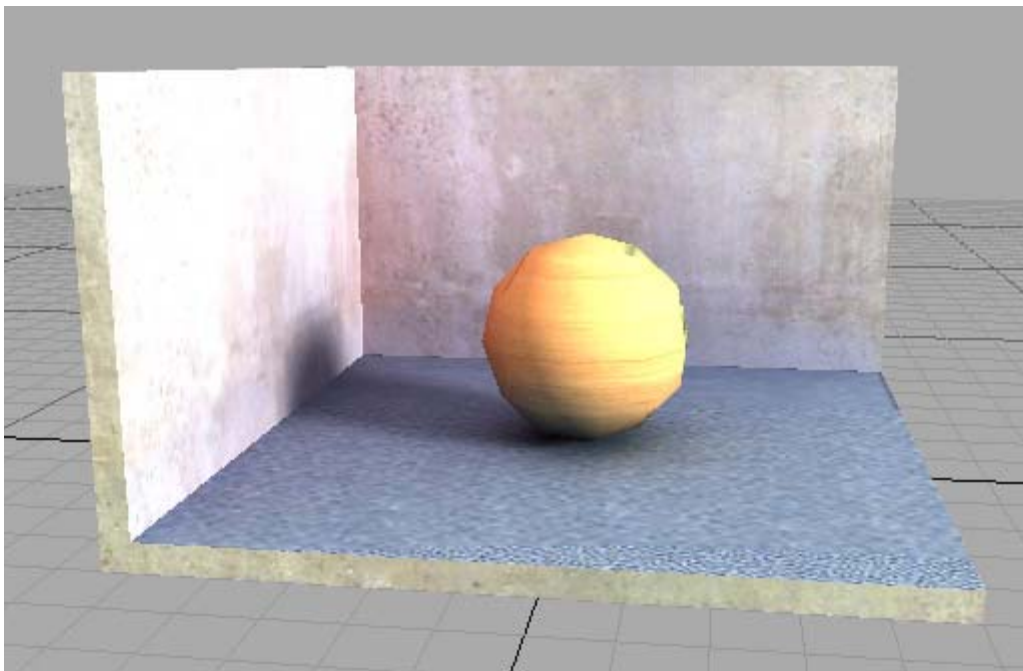


# Max2Ctor Tutorial

**“Get in the game!”**



**This tutorial details the workflow that allows Torque interiors to be modeled in 3dsmax and exported directly to Constructor.**

**Author: Ryan Mounts  
Date: 1/9/08**

## Introduction

In this tutorial, you will learn how to use the Max2Ctor macroscript to export a scene from 3dsmax directly to Constructor, as well as how to bake advanced lighting into your interiors. This workflow allows you to take advantage of 3dsmax's robust and versatile feature set to create your interiors... if you can model it, Max2Ctor can export it!

It should be understood from the beginning that Max2Ctor is not meant to replace Constructor as your sole brush editing tool, but rather to make it easier to create rich and complex interiors. Max2Ctor does NOT check for brush convexity, so the modeler must understand this principle and construct geometry accordingly. This is a powerful method, if used properly, and can be used to generate interiors that would be extremely difficult and time-consuming in Constructor, if not impossible. For example, suppose you want a gigantic tree in your game with roots that twist and curl everywhere, large enough for the player to run on and under. This sounds like a daunting task in Constructor, but it is actually quite straightforward in 3dsmax. You might make this by creating a large cylinder and then using the "extrude along path" feature of the Editable Poly to quickly generate the curvy, tapering roots, or maybe use a loft object instead. The resulting geometry is not convex, however. But you could go in and detach each segment of the root manually and cap the holes to make each piece convex! Voilà, you have collidable twisting roots, and what's even better... you have the proper texture coordinates generated for you automatically!

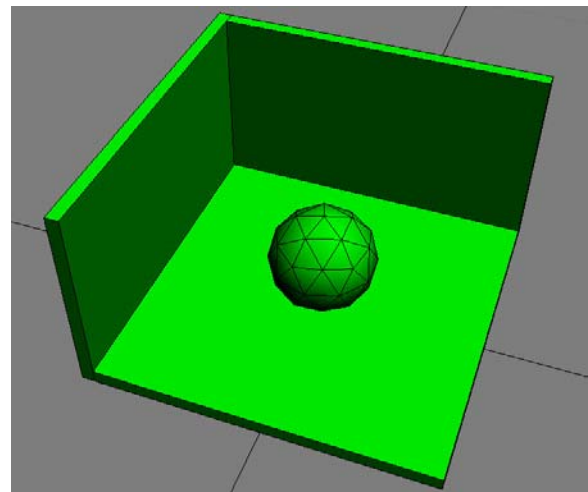
A more advanced use of this workflow might be to bake global illumination into your interior texture maps using the 3dsmax tool, Render To Texture. You could bake

ambient occlusion information into your interior textures and let Torque provide the direct lighting in the lightmap. Or you could bake a complete lighting solution into your textures, including direct light with soft shadows, ambient occlusion, indirect light with color bleed, etc. Another macroscript is available, called Explode Brushes, which can be used to bake textures for multiple brushes into a single texture map. Part 2 of this tutorial will demonstrate this technique. The intermediate 3dsmax user should be able to follow this tutorial without problems.

## Part 1 – Creating, Mapping, and Exporting a Simple Scene

### Setting Up the Scene

First, follow the instructions in the "readme.txt" file provided with the macroscripts to install them in 3dsmax. When that has been completed, begin by creating a simple scene like the one shown in Figure 1, or simply open the 3dsmax9 scene, Max2Ctor\_tut.max, provided with this tutorial. If you create your own, just be



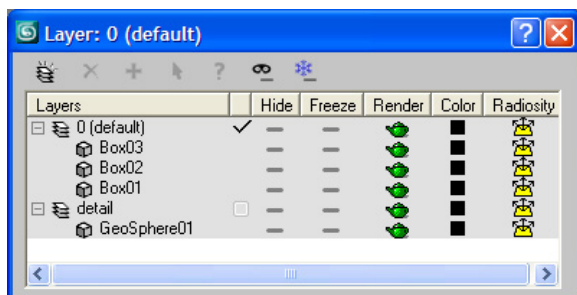
**Figure 1. Simple Scene with Convex Brushes**

sure that the geometry is convex. It should look just like how you would create an interior in Constructor, with brushes not overlapping, etc. Also, note that there is no scale feature for Max2Ctor, so do not make your geometry too big. Just remember that 1 unit is equivalent to about 1 meter in Torque. So if you make a box that is 50x50x50 units in size, it'll be HUGE in-game.

Brushes will be exported as structural entities by default. To change a brush's type, simply add it to the properly named layer. Allowable layer names are:

detail  
collision  
vehicle\_collision  
portal  
trigger  
InteriorTrigger  
PathedInteriorEntity  
Force\_Field  
Door\_Elevator

These are case-sensitive. Let's change the sphere into a detail brush by adding it to the "detail" layer. Open the Layer Manager and create a new layer. Rename it to "detail" and add the sphere to it, as shown in Figure 2. It's that easy.



**Figure 2. Layer Manager**

Now that we have a scene to work with, you could go ahead and click the Max2Ctor button and export the scene to Constructor.

It would open in Constructor texture-less, but each brush would still have the default 3dsmax texture coordinates, which is fine if you want to do your texturing in Constructor. Feel free to try it if you like.

## Texture Mapping

As far as this tutorial is concerned, however, we want some textures in 3dsmax! So let's start by creating some materials and applying them to our brushes. Create a Standard material with the provided wood texture map in the diffuse slot, enable "Display Map in Viewport," and apply it to the sphere. Then create a Multi/Sub-object material with three materials. Place the concrete texture map in the diffuse slot of the first, the brick texture map in the diffuse slot of the second, and the carpet texture map in the diffuse slot of the third, being sure to enable "Display Map in Viewport" for each texture. Now apply this Multi/Sub-Object material to the three boxes in the scene. Your material editor should look similar to Figure 3.

Now let's begin UV mapping our brushes. First, convert them all to Editable Polys. Select the box that represents the floor, enter Polygon sub-object mode and select all the polys. Change their Material ID to a value of 1, making them all concrete. Now select the top poly and change its Material ID to a value of 3, making it carpet. Exit sub-object mode. Select one of the side walls and enter Polygon sub-object mode. Select all polys and give them a Material ID of 1, making them all concrete. Now select the polys facing away from the sphere (on the outside of our little "room") and change their Material ID to a value of 2, making them brick. Repeat this for the other side wall.

Now let's fine-tune the textures. Select all three boxes and apply a UVW Map modifier to the group as a whole. Next, change the

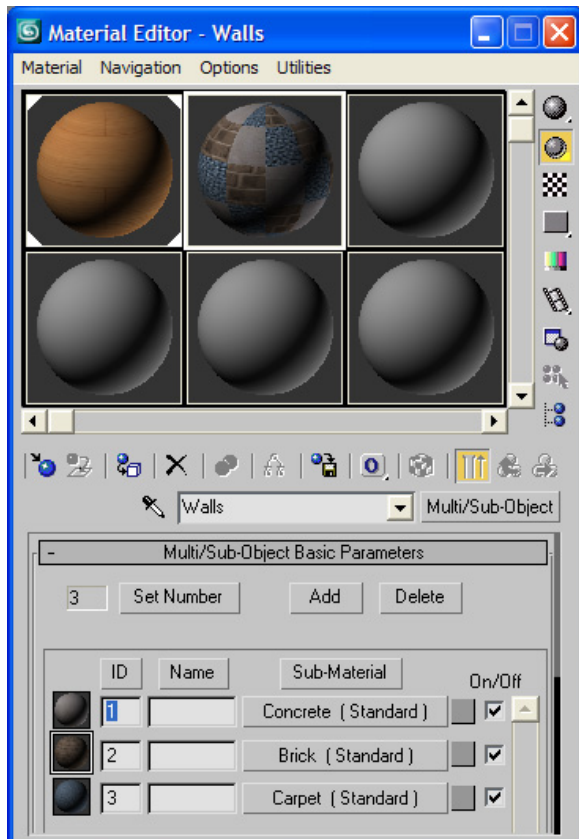


Figure 3. Material Editor

Mapping type to “box.” This will eliminate texture seams across adjacent brushes. But the carpet and the bricks are not tiled enough. Apply a Poly Select modifier to the selection. Enter polygon sub-object mode and select the top floor polygon. Without leaving sub-object mode, add a UVW Map modifier. This places a planar mapping gizmo on the carpet polygon. Change U and V Tile to 2 to make the carpet denser. Add another Poly Select to the modifier stack. Select the brick polygons as shown in Figure 4, and apply another UVW Map modifier. Change the Mapping to “box” and set the U Tile to 2 and V Tile to 1.4. Of course, you can play with these settings all you like, including moving, scaling, and rotating the gizmo. Keep in mind, though, that spherical and shrink wrap mapping cannot be preserved as seen in the 3dsmax viewport except for triangulated meshes, like the geosphere. Therefore, you should always use the geosphere primitive over the regular sphere primitive.

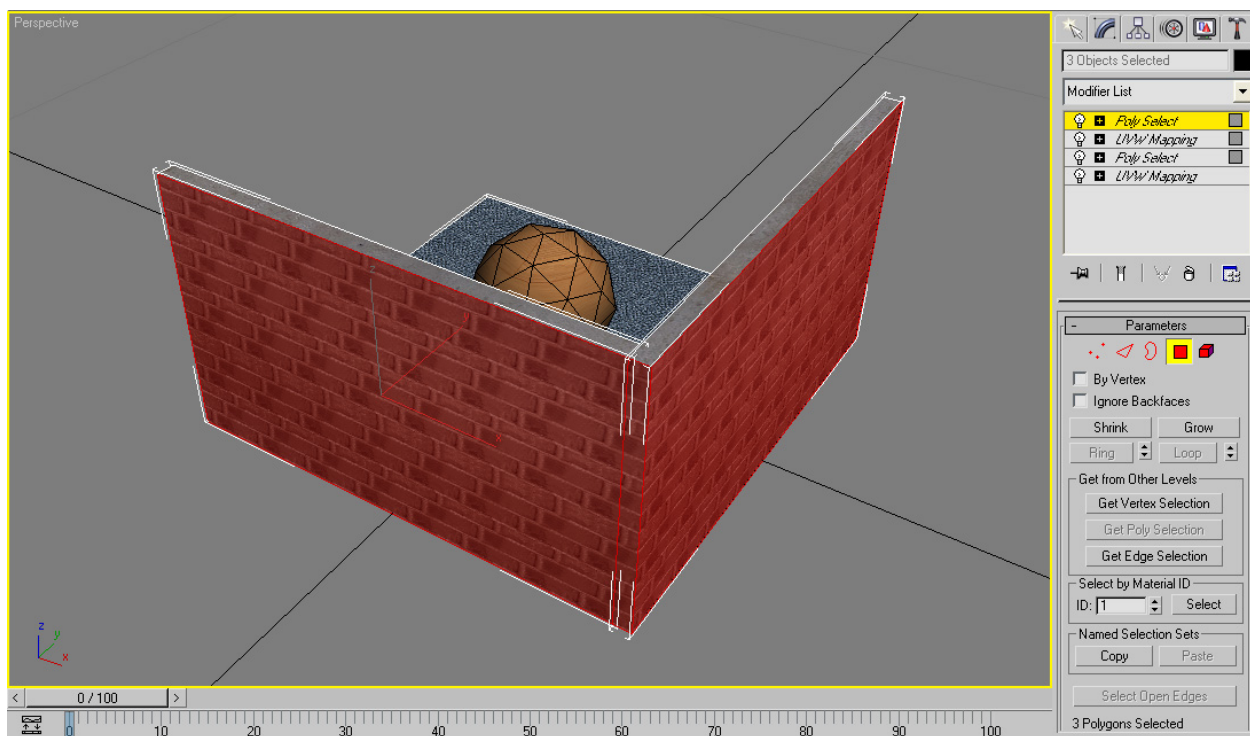


Figure 4. Selecting Polygons across Multiple Brushes

## Exporting the Scene to Constructor

This is the easy part. To export the 3dsmax scene to Constructor, first hide any objects or lights that you don't want exported. Only what's visible in your viewport gets exported. In our case, we want to export all three boxes and the sphere. So go ahead and click the Max2Ctor button on the main toolbar as shown in Figure 5, which you should have installed at the beginning of this tutorial.

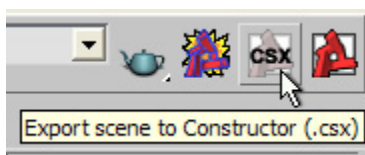


Figure 5. Max2Ctor Button

This should bring up the Export Scene dialog. Click the Browse button to choose your output file. If you performed the default Constructor install without modifying any directories, Max2Ctor should default to the "scenes" folder in your root Constructor folder. Name your file "tut\_part1" or something similar. You should see something like Figure 6. Our map channels should have all defaulted to 1, so just leave Map Channel on its default. Now just click Export Scene. The 3dsmax viewports will flicker slightly. This is simply due to a background "hold" and "fetch" call. Nothing to worry about. When the exporter is finished, a success dialog should appear.

*Note: Although Max2Ctor does considerable error checking, it is not completely bullet-proof. At the time this tutorial was written, there is one known bug. If a Multi/Sub-Object material is applied to an object, all the polygons of that object must have valid Material ID's. For example, by default our boxes have a different Material ID for each*

*polygon, 1 through 6. But our applied material only has 3 sub-materials. Therefore, all our boxes' polygons MUST have Material ID's in the range of 1 to 3. If you forget to change one of the polygon Material ID's and it happens to lie outside this range, a runtime error will occur in the Multi/Sub-Object portion of the code. This is an issue in Max2Ctor v1.0. All subsequent versions should not suffer from this bug.*

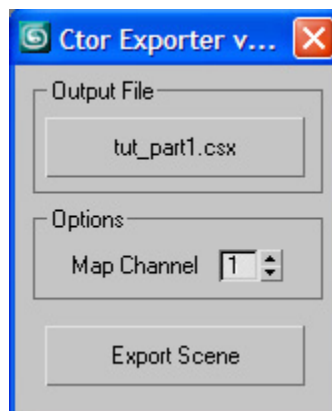


Figure 6. Export Dialog

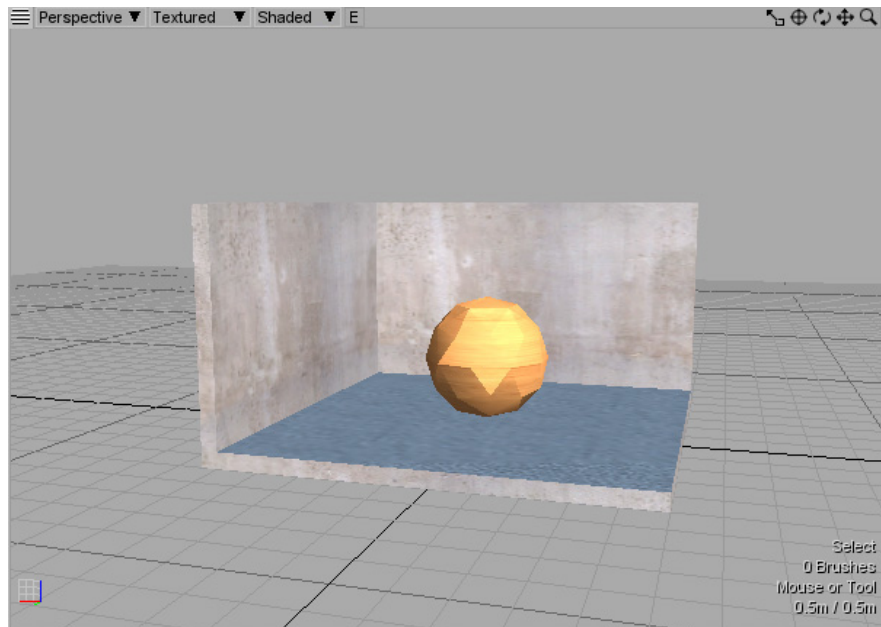
Now let's open up our scene in Constructor to verify that the export was successful. So startup Constructor, and first ensure that the materials used in your 3dsmax scene (wood, concrete, bricks, and carpet) are loaded in the Constructor material library. If they are not, your scene will open but have error textures.

*Note: Sometimes even with your textures properly loaded in Constructor's material library, Constructor will not be able to resolve a texture name. This happens for the bricks texture on my machine. I'm assuming this is a Constructor issue, or perhaps some naming convention that I'm not aware of. If some polygons have the error texture, I simply select them in Constructor and apply the proper texture. The texture coordinates are preserved.*

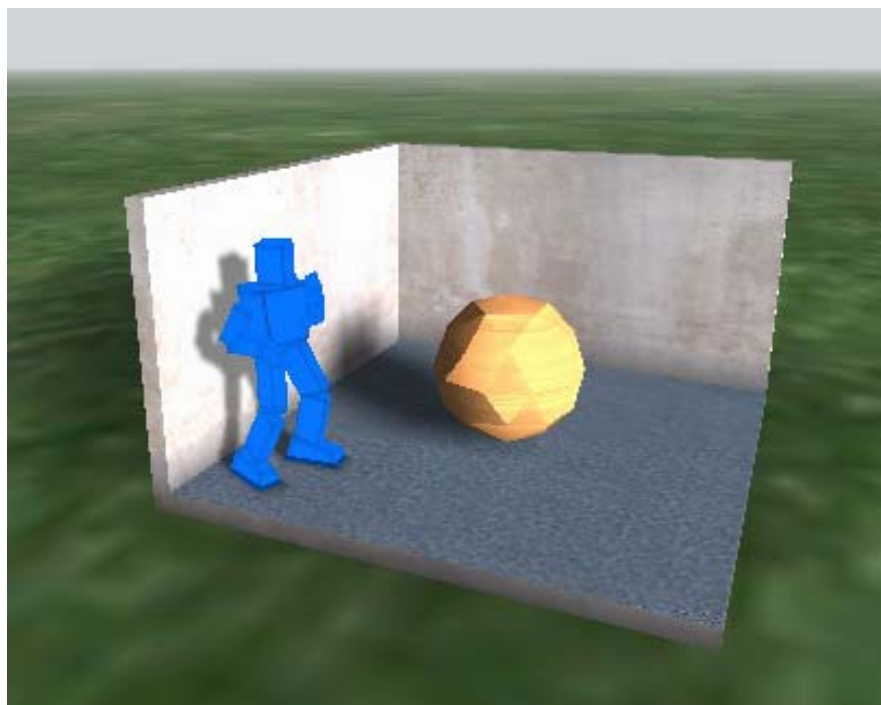


At this point, you should see something like Figure 7. You might try exporting the scene to DIF and dropping our little creation into Torque, as seen in Figure 8. It's always a good idea to verify that collisions are

functioning properly. The faceted shading occurs because Constructor does not use normals. There is a smoothing parameter in Constructor that should smooth the lighting, but I have never gotten it to work.



**Figure 7. 3dsmax Interior in Constructor**



**Figure 8. 3dsmax Interior in Torque**

## Part 2 – Advanced Tricks

### Incorporating Advanced Lighting

Well if you were thinking to yourself, “That doesn’t look anything like the image on the cover page of this tutorial!” then you’re absolutely right. That image was created by baking global illumination information directly into the brushes’ texture maps. Anyone familiar with Render To Texture should be comfortable with the term “baking.” And every game artist should have an intimate understanding of this tool, as it is very important to skinning, especially DTS models. Using Max2Ctor we can now incorporate the Render To Texture workflow into our interior creation, which leads to some interesting tricks!

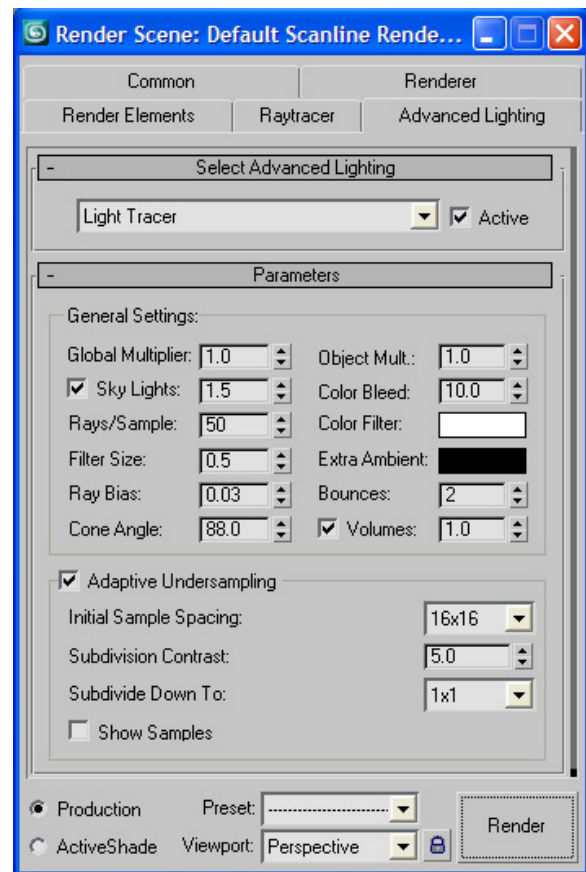
Let’s go back to 3dsmax and continue building on top of our scene. Create two lights: one Skylight and one Target Direct. Enable Cast Shadows for the Skylight. Position the Target Direct where you want your sun to be. Turn Shadows on for the Target Direct and change the shadow type to Area. Under the Area Shadows rollout, change the type to Disc and set the Length and Width to 2. This will give us very soft shadows. Under the rollout labeled Intensity/Color/Attenuation, increase the multiplier to 1.3. Feel free to play with any of these parameters. Now create a large plane beneath our little room to represent the ground. Either change the plane’s color to a dark green or add a grass material to it. We want light to bounce off the ground and light our scene to create a more realistic look.

We can now setup up the advanced lighting. You can use any renderer you want, as long as it supports Render To Texture.

*Note: Mental Ray unfortunately only supports occlusion maps in Render To*

*Texture. Hopefully this will change in the future. V-Ray is a good third party renderer alternative that fully supports Render To Texture.*

Let’s use Light Tracer to produce our lighting. On the main menu, select Render > Advanced Lighting > Light Tracer. You can set up your parameters like Figure 9, or play around as you like.

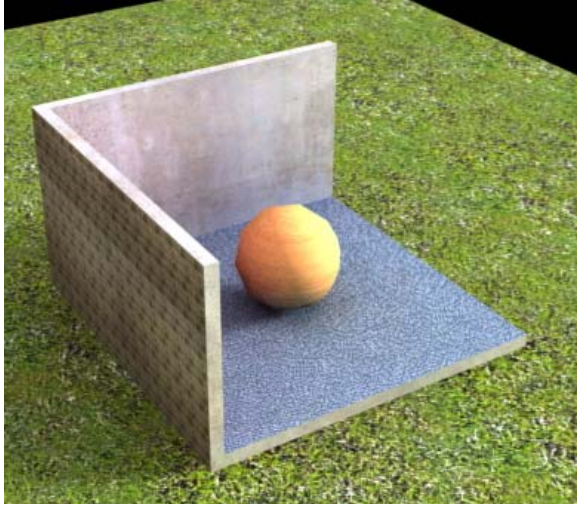


**Figure 9. Light Tracer Setup**

Do some test renders and tweak the lighting to your satisfaction. You should have something like Figure 10. Once you are ready to move on, we can prepare our brushes for texture baking.

### Unwrap UVW

There are a couple of options here; you could bake a texture for each individual



**Figure 10. Light Tracer Test Render**

brush, which would be fine for this simple scene, but in general would not be a very efficient method. A complex scene with many brushes would require too many textures and would quickly eat up precious texture memory when used in-game. So another option would be to bake all the textures and lighting for the whole interior into one or more large textures. This method should also reduce the number of texture state changes required to display your interior. This is a good thing! The more triangles an engine can draw without having to swap textures, the better. The trade-off here is the potential to lose fine details when viewed up close in-game. Another trade-off is wasting some texture real estate. We need to try to minimize these two problems.

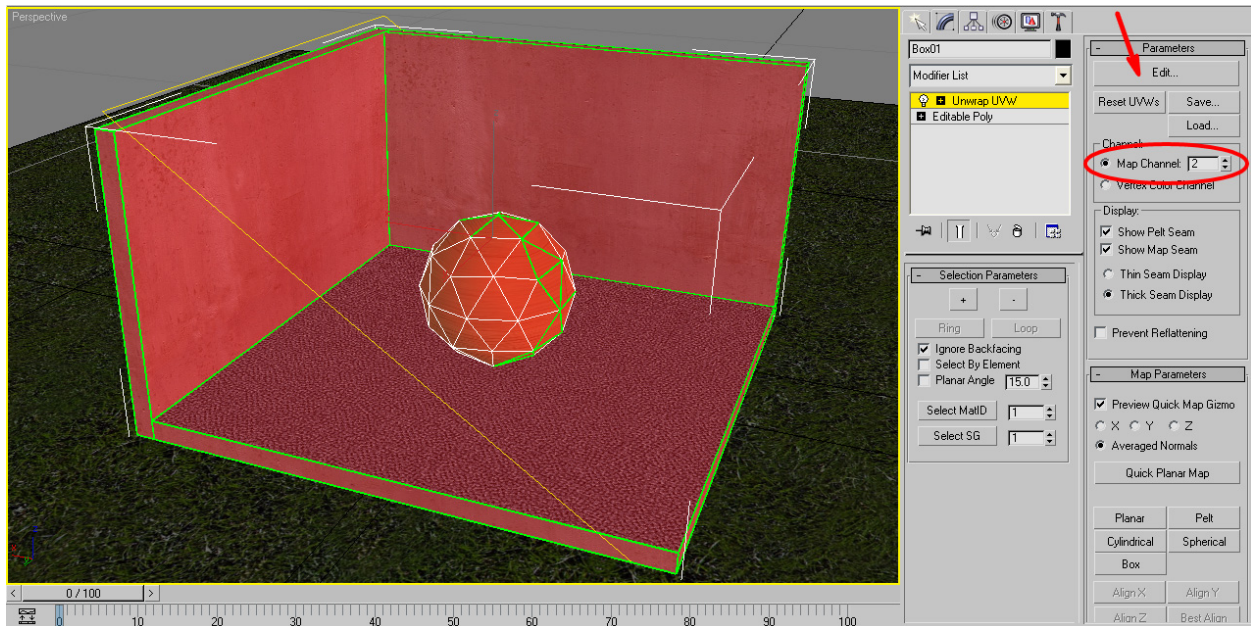
I'll show you how to bake everything for the whole scene into one texture, but if you understand the process, you could easily extrapolate this procedure to bake maybe the two walls into one texture, and the floor and sphere into another to retain more fine detail. First select the floor brush and convert it to an Editable Poly. Attach the walls and sphere so that you end up with only a single Editable Poly object, in

addition to the ground plane. If the Attach Options dialog pops up, just click OK to select the default option.

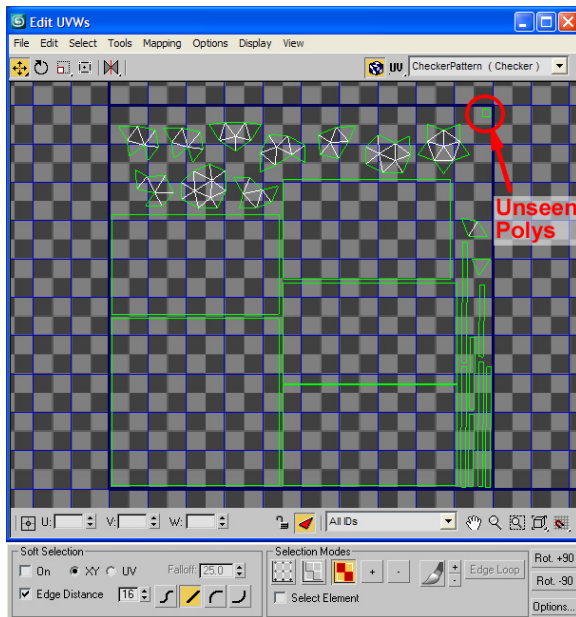
Now add an Unwrap UVW modifier to the single Editable Poly object. **This is very important: change the Map Channel to 2.** Click OK to dismiss the warning dialog that pops up. The value of 2 is not set in stone; it simply needs to be something different from our other map channels. Enter Face sub-object mode and select only the polygons that would be visible to a player in-game. For example, do NOT select the polygons that are touching the ground; no one will ever see them, so they should not be taking up valuable space in our texture. Now click the Edit button as shown in Figure 11.

The Edit UVWs floater should open, and you should see in red your selected polygons. Enable Filter Selected Faces (the triangle to the right of the lock icon) to unclutter our work area. From the menu bar, select Mapping > Flatten Mapping. Just click OK to keep the default settings, and you should see the selected polygons separate out into small chunks that each get their own texture real estate. In the 3dsmax menu bar, select Edit > Select Invert. This will select all those unseen polygons. Use the scale tool to shrink them way down and move them to an unused corner of the texture. Now disable Filter Selected Faces so that we can see all the polygons. The default map flattening is okay, but a lot of the texture is wasted space. Manually scale, move, and rotate the polygons to better utilize our texture real estate, as shown in Figure 12. When satisfied, close the Edit UVWs floater and exit Face sub-object mode. Now that our brushes are unwrapped into one texture, we are ready to start baking.





**Figure 11. Unwrap UVW Modifier**



**Figure 12. Flattened Mapping**

## Render To Texture

Be sure the Editable Poly object is selected and open the Render To Texture floater by pressing “0” (zero) or selecting from the menu bar Rendering > Render To Texture. Under Object to Bake you should see your

object’s name. We want to use the Unwrap UVW mapping coordinates, so change Channel to 2 as shown in Figure 13. Now scroll down to the Output rollout. Click the Add button, select CompleteMap from the list, and click Add Elements. You should see the element added to the output list. Click the “...” button to the right of File Name and Type to change the output directory and filename to whatever you desire. Click the 1024x1024 button to set the texture size. We want a big texture! The bigger the texture, the better it will preserve the fine details. Scroll down to the Baked Material rollout and check the Render to Files Only option. Your setting should look similar to Figure 14. To bake the texture, simply click Render at the bottom of the Render To Texture floater. A virtual frame buffer should appear, and your texture will render before your very eyes. Your result should look something like Figure 15.

*Note: Be careful not to set your advanced lighting quality settings too high before you Render To Texture. Baking textures can*

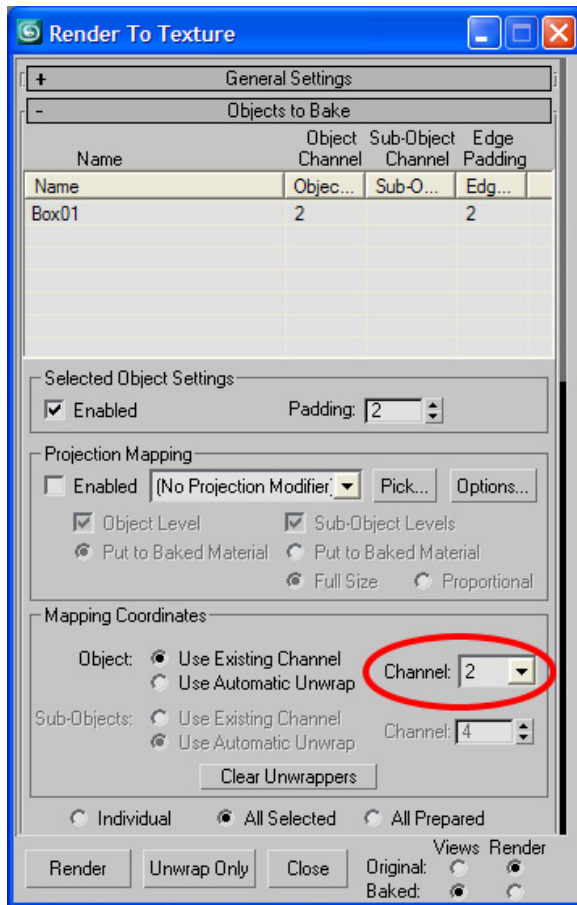


Figure 13. Render To Texture (Part 1)

*require much longer render times than a normal viewport render. This can be a problem, since there is no way to cancel Render To Texture after you hit Render. Once you start, you just have to wait it out. I suggest starting with very low quality settings and increasing them incrementally if render times are not unbearable.*

We now need to apply our baked textures to our Editable Poly object. Create a Standard material with the newly rendered texture in the diffuse slot. **Important: change the Map Channel of the diffuse texture to 2.** Be sure to enable Show Map in Viewport. Also set the material's self-illumination to 100 so that the scene lights do not affect it. Apply the material to the Editable Poly object and you should see advanced lighting right in your viewport!

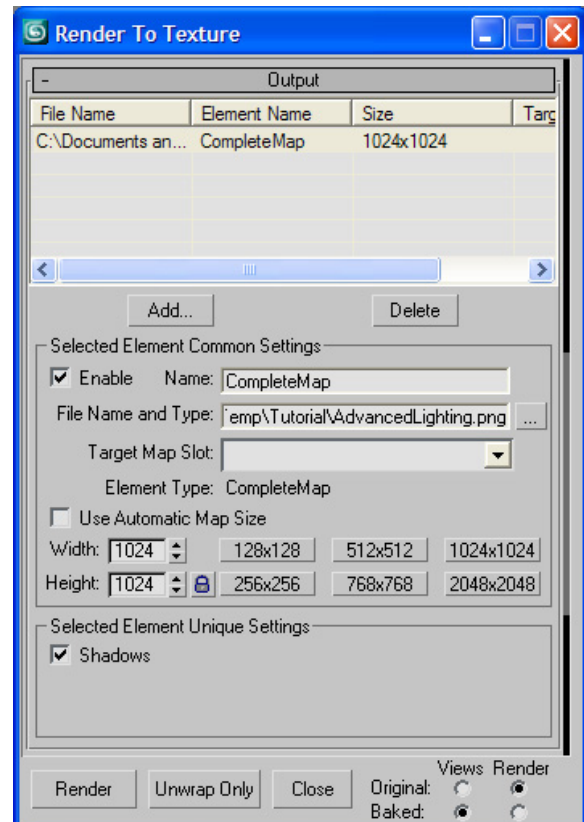


Figure 14. Render To Texture (Part 2)

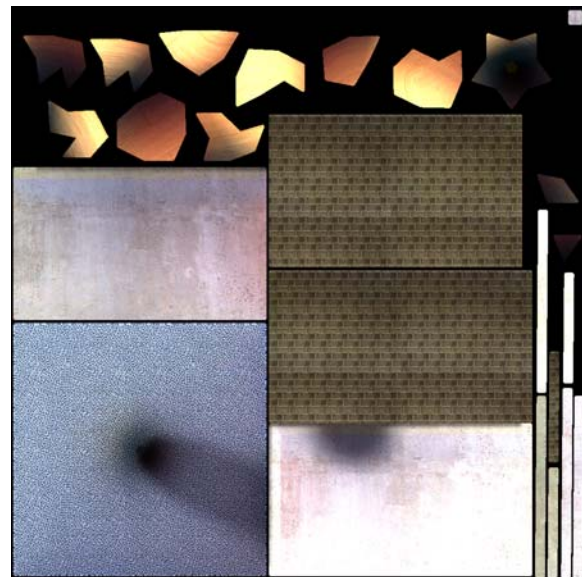


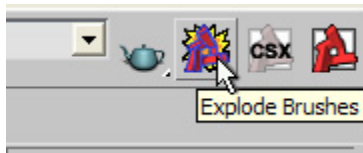
Figure 15. Render To Texture Output

If desired, you can touch up this texture or add details by hand with your favorite paint program. I actually touched up a small area

directly beneath the sphere that did not receive shadows properly. You could easily add decals or adjust colors, saturation, brightness, etc.

## Exporting the Scene

We only want to export our little room, so hide everything else, like the ground plane, lights, etc. The scene cannot be used as is, because all the brushes have been collapsed into a single Editable Poly. We need to get our brushes back out. One way to do this would be to enter Element sub-object mode and individually detach each brush, but you can see how this could become very tedious and time consuming for a more complex scene. So let's use the helper macroscript, Explode Brushes, to do the work for us. Simply select all objects you want decomposed into their individual elements (in this case, just the little room) and click the Explode Brushes button on the main toolbar, which should have been installed at the beginning of this tutorial. See Figure 16.



**Figure 16. Explode Brushes**

Voilà, we have our individual brushes back. Add the sphere back to the detail layer and then use Max2Ctor to export the scene to Constructor, **with Map Channel set to 2**.

*Note: All layer information is lost when we attached everything into one Editable Poly. If you use this workflow, setup your layers **after** exploding the brushes to avoid that “Crud, I gotta redo all that?” moment.*

Open up Constructor and load the baked texture into its material library. Open the exported scene to see our room. Since the

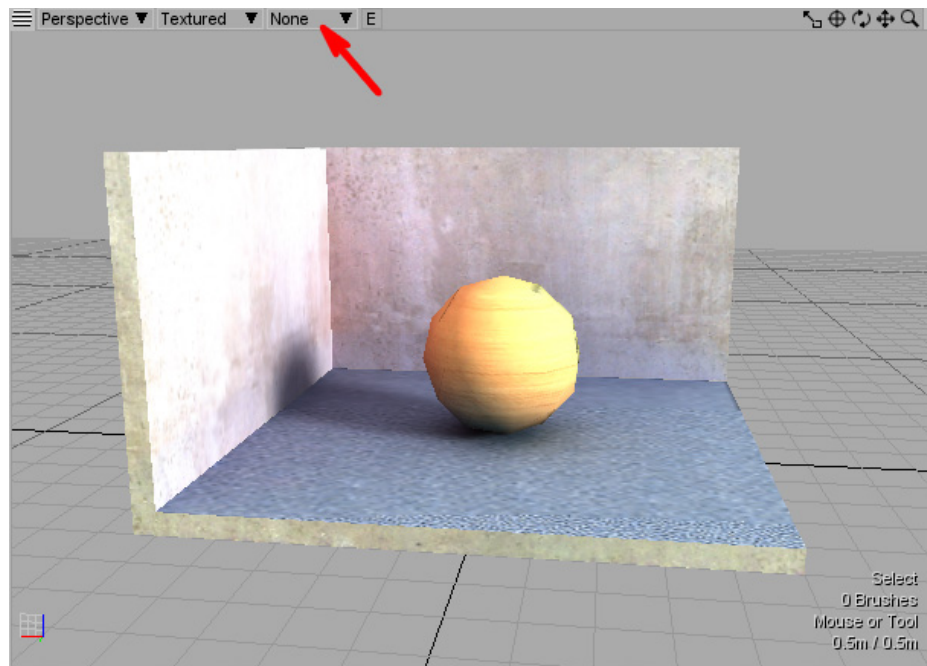
lighting information is baked into the textures, change the viewport lighting mode to None as shown in Figure 17. This is basically like setting the self-illumination to 100.

To see the room in-game, export the scene to DIF and drop it into a Torque simulation (I'm using TGE 1.5.2). In the World Editor, select the room and check the “useGLLighting” option to disable its lightmap. The lighting's in the texture, remember? To keep the scene looking right, change the sun's position to mimic the placement of the Target Direct we created in 3dsmax. You should have something like Figure 18.

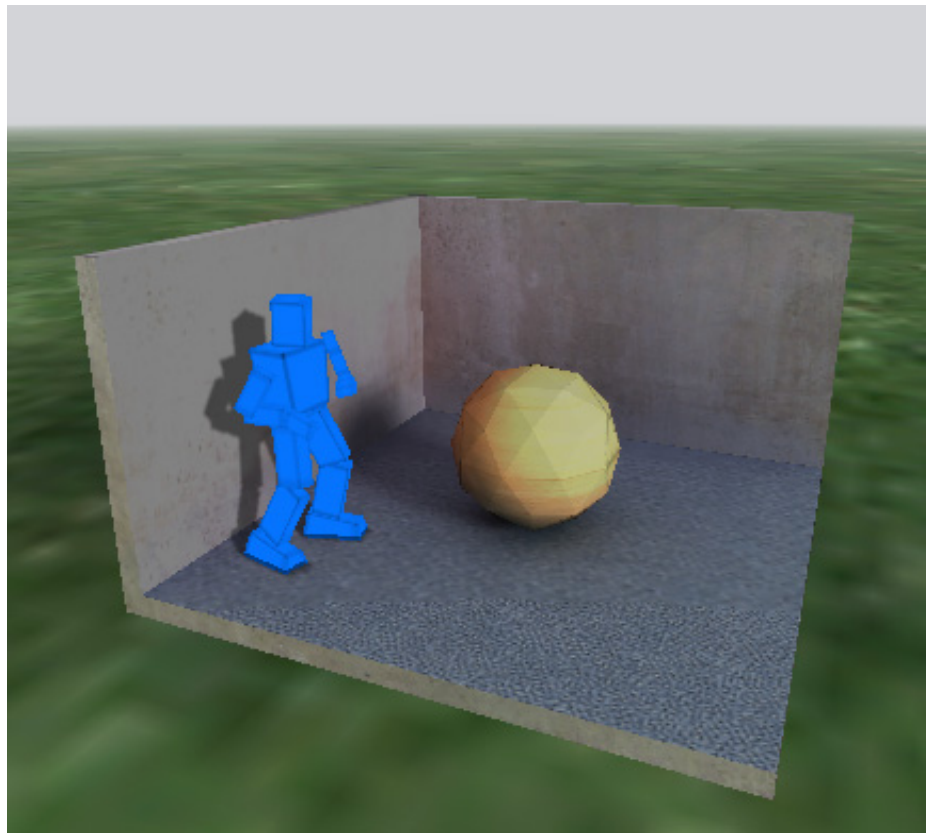
There is something slightly amiss, however, which you probably noticed if you have a good eye. The room does not look as vibrant as it did in Constructor; it's darker, slightly desaturated, and the sphere is faceted. The faceted look is due to the GL lighting, which is superfluous anyway because the lighting is already in the texture. The other issues are due the Torque Lighting Kit and the way it extends the dynamic range of the lighting, I believe. The ideal solution would be to change the engine code to add another check box in the World Editor which would expose the ability to make the interior completely self-illuminated. That would make it look like it did in 3dsmax and Constructor, taking full advantage of our baked advanced lighting. It does not seem like a difficult task upfront... perhaps that will be my next little project. ☺

If anyone has any suggestions on how to improve this tutorial or the scripts, feel free to let me know. I hope you enjoyed this tutorial and that these tools will allow game artists more freedom and the ability to create more complex and rich environments easier and faster.





**Figure 17. Baked Advanced Lighting in Constructor**



**Figure 18. Baked Advanced Lighting in Torque**