Step 1 - Setup

Open the NodeEditorWindow.

Go to this site-

http://www.oera.net/How2/TextureMaps2.htm

Download

"EarthMap_2500x1250.jpg",
"EarthClouds_2500x1250.jpg",
"EarthNight_2500x1250.jpg",
"EarthMask_2500x1250.jpg",
"EarthElevation_2500x1250.jpg"

Scale up the imported images in Unity to 4096, for maximum clarity. If this slows your computer down too much due to memory issues, feel free to reduce them.

Make sure you make "EarthNormal" as a Normal Texture Type.

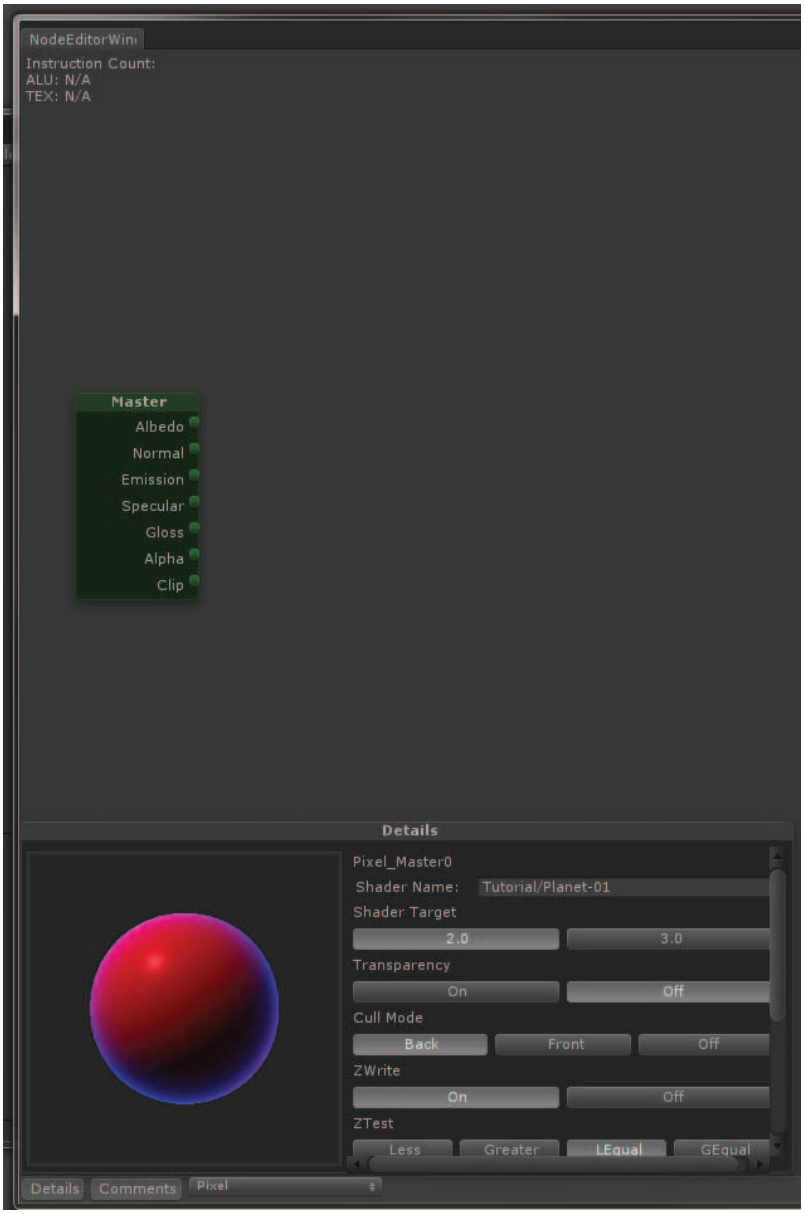
I'm going to take you through the classic Planet tutorial, made interesting through the use of some funky tricks possible in the Strumpy editor. It seems that making a nice, pretty Earth shader is becoming the Teapot of the shader world. That's good. It gives us something to compare. Through the use of this standard we can compare the complexity and efficiency of a tool.

Plus, I get to introduce you to my new best friend, the LERP. More on him later.

Of course, thanks to Texel and Stramit for making such an amazing editor, and then for making it free.

Step 2 - The Basics

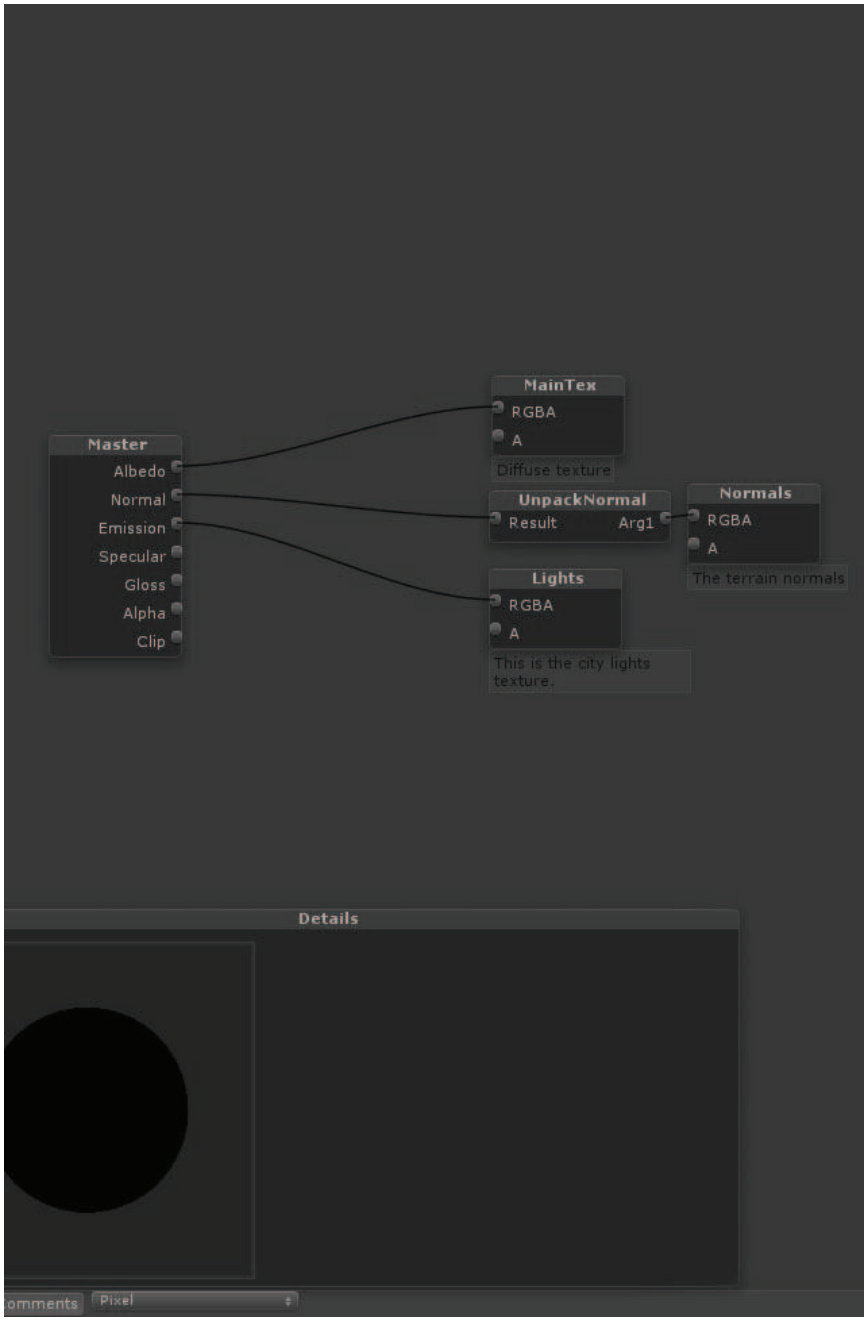
Rename the Master node "Tutorial/Planet-01".



In the details window there's a lot of options, but we can ignore them all for now. They are mostly beyond the scope of this tutorial. The only thing you need to worry about is the Shader Target. We'll use the 2.0 target for now, but later we'll be changing to 3.0, when the time comes. We do this because if you can make a shader that sticks to 2.0, it will be more compatible with a larger number of older graphics cards.

Step 3 - Some simple nodes

Make 3 **Sampled2D** nodes, and 1 **UnpackNormal** node. Rename the **Sampled2D** nodes as "MainTex", "Normals" and "Lights".

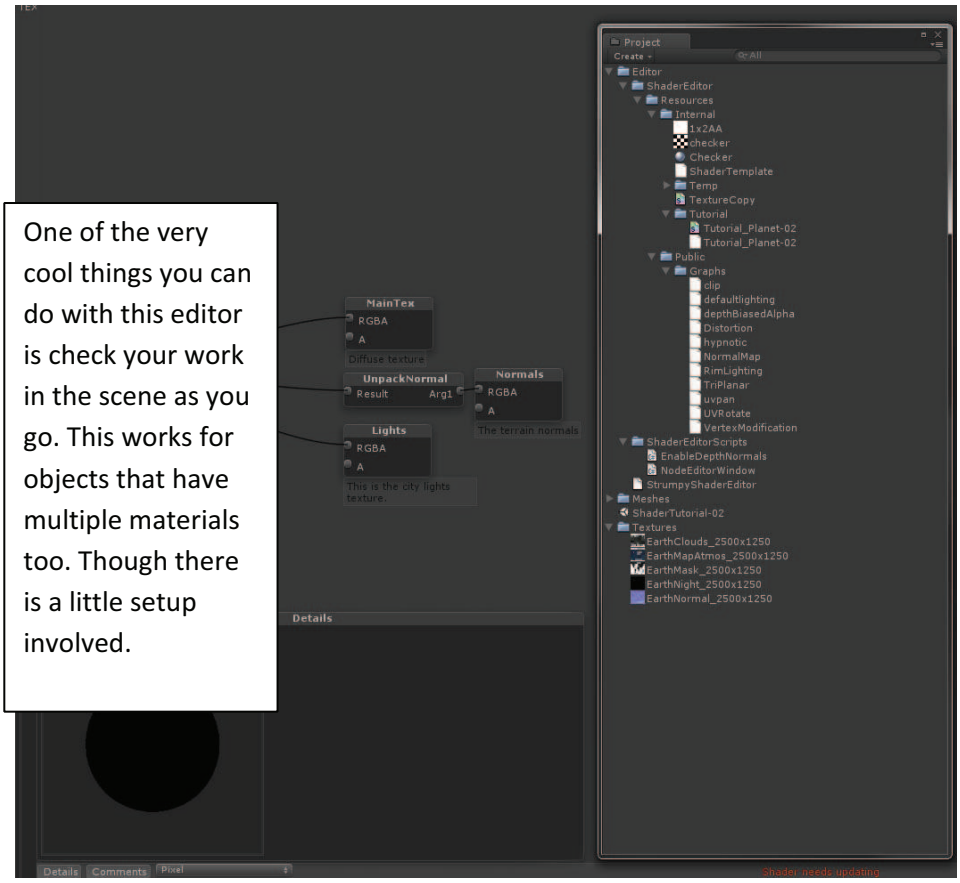


Any shader that has "MainTex" as the image going into the Diffuse (or Albedo as Unity calls it) channel will allow the image in that channel to interact with Beast. If you're using Beast, you want to call your main colour image "MainTex". Join this up to the Albedo input on the Master node. Though it's unlikely you'll be using Beast on a planet...

(Note- If you call your Normal map "BumpMap", Unity knows what this should be and will advise you if you use the wrong sort of image here. Also, if you call your Emission map "Emission" instead of lights, it would be something that Beast would take into account when doing lightmap renders. Again, no Beast, so it's not relevant, but it's a good thing to know.)

Step 4 - Setting up for in-scene feedback

Update, Save and Export the shader. Load the .sgraph back in. Assign the shader to your model. This will allow you to see updates in your scene.



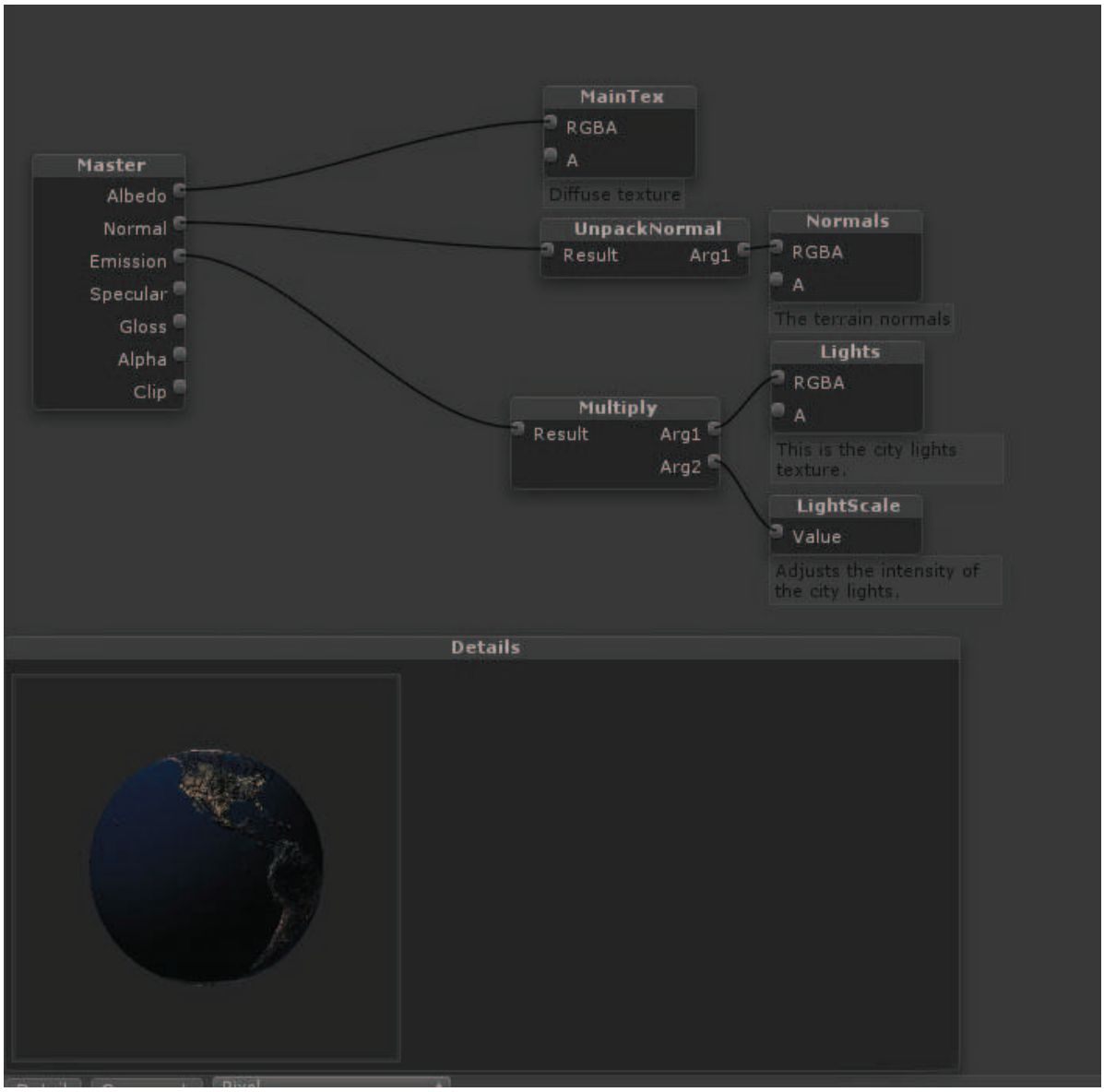
Make your scene's background black, and turn the ambient lighting to black.

Ok, here's a bit that seems to need a little TLC from Stramit and Texel. Once I've saved and exported my shader, I then have to load the saved sgraph back into the NodeEditorWindow, re-Export As one last time. Now the Export button comes online. Once this happens, any change you make will show up automatically into the scene's version of the shader. It's a little fiddly, but it does make for great feedback.

If you've done all that, you should be ready to get going on the meat of this tute.

Step 05 - Light Scale

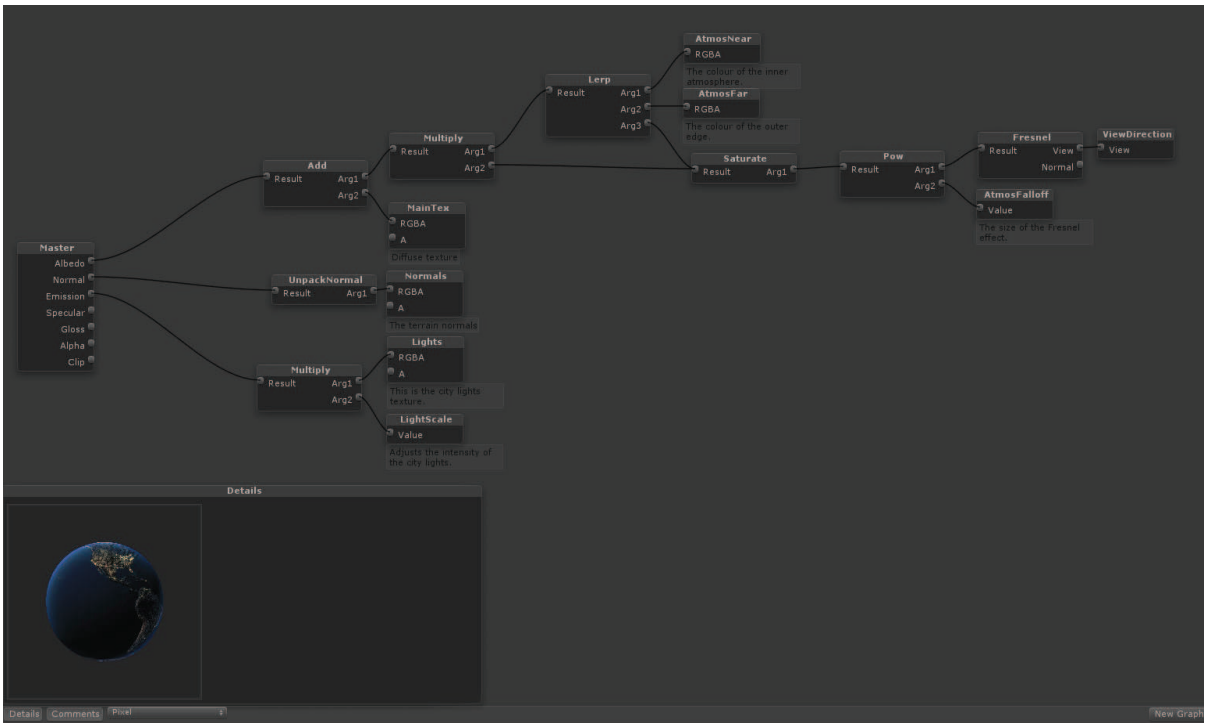
Add a **Multiply** and a **Float** to the Emission channel. Rename the **Float** "LightScale". Give the **LightScale** node a default value of 1.



Step 06 - Atmospheric Rim Lighting

Make the following nodes - **ViewDirection**, **Fresnel**, **Float**, **Pow**, **Saturate**, **Color**, **Color**, **Lerp**, **Multiply**, **Add**.

Wire the **ViewDirection**'s View output into the **Fresnel**'s View input. Take the result from the **Fresnel** and run it into the **Pow**. The other input of the **Pow** we run from the **Float**. We're going to call this **Float** "AstmosFalloff". I gave it a default value of 3.



The Result from the **Pow** runs to the input on the **Saturate**.

Hook the two colors to the first two inputs on the **Lerp**. Then wire the result of the **Saturate** into Arg3.

Give the two **Color**s some really easy to see colors like bright red and green.

the **Color** going into the Arg1 "AtmosNear" and the one going into the Arg2, "AtmosFar".

Wire the **Multiply** in between the **Lerp**'s Result and the Albedo and put the Saturate into the second input. (Your Saturate should be running into both the Lerp's Arg3 and the Multiply's Arg2).

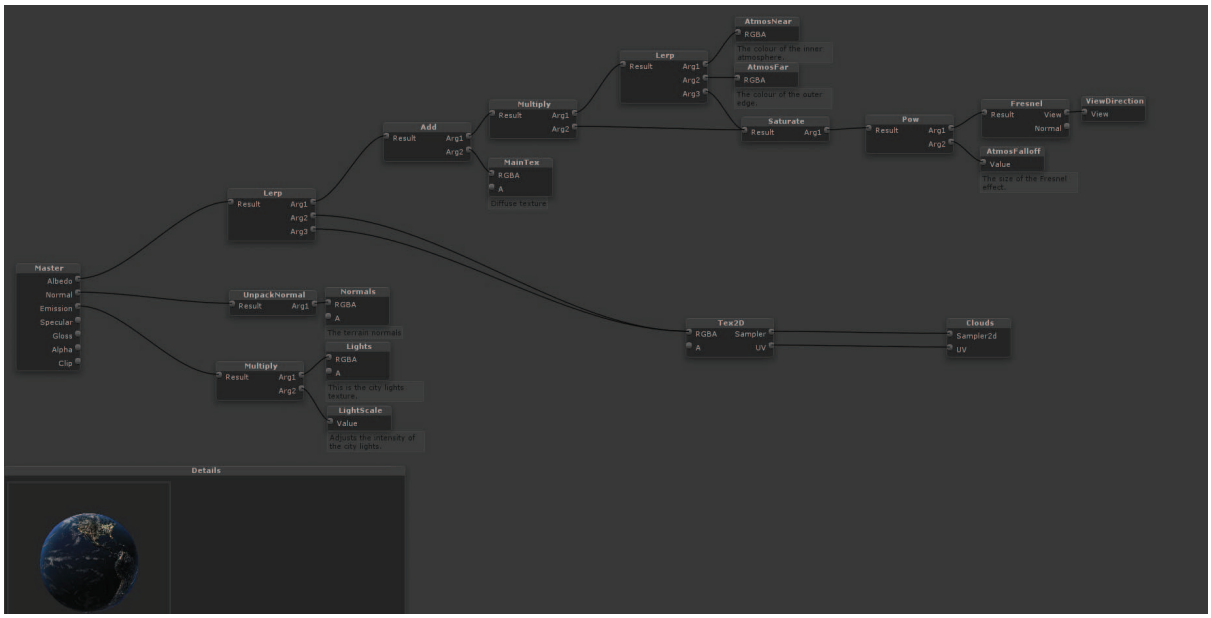
Add the **MainTex** back into this so they work together. Wire the **Add** into the path between the **MainTex** and the Albedo, and stick the **Multiply**'s Result into the other input.

Rebuild.

Step 7 - Adding the clouds

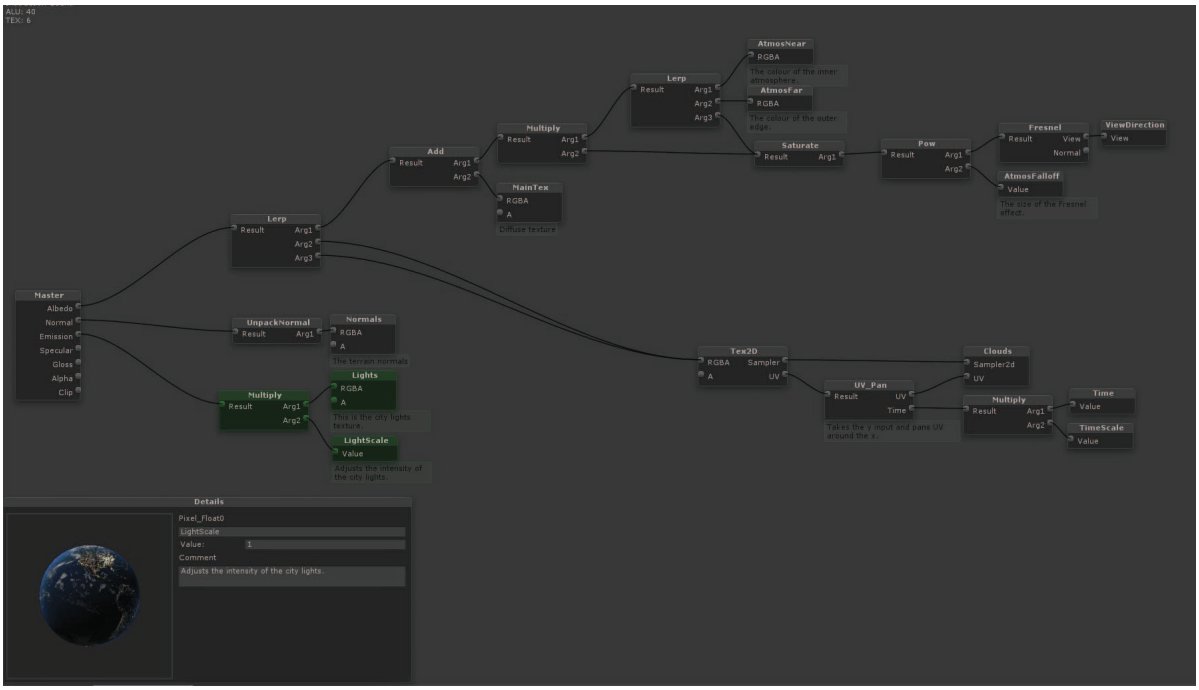
Make a **Tex2d** and a **Sampler2d**. Rename the **Sampler2d** "Clouds". Make the **Clouds** default color to be black. Wire the **Clouds**'s outputs to the **Tex2d**'s inputs. Then make a **Lerp** and put it between the **Add** and the Albedo.

Wire the Cloud's **Tex2d** into both the Arg2 and Arg3, while putting the **Add** into the Arg1.



Step 8 - Making the clouds spin

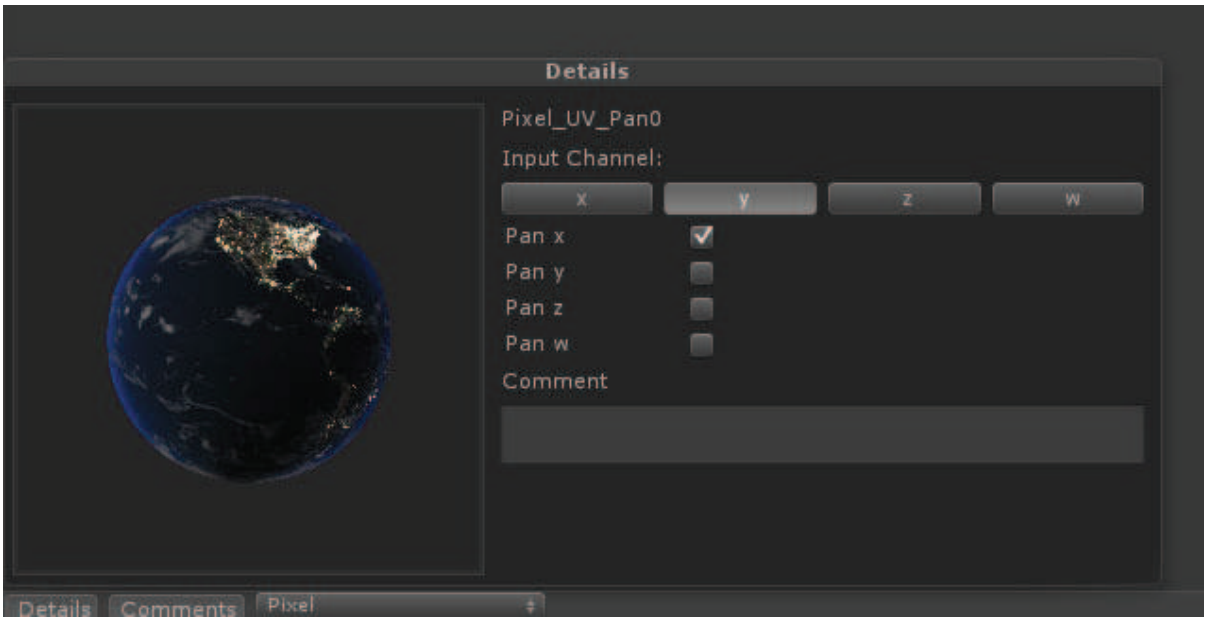
Make a **UV_Pan**, a **Multiply**, a **Float** and a **Time** node. Rename the **Float** to be called "TimeScale.



Go into the **UV_Pan** node.

Tell the **UV_Pan** node to only accept the 2nd input channel. Tick the "Pan x" tag.

Give your **TimeScale** node a default value of 0.01

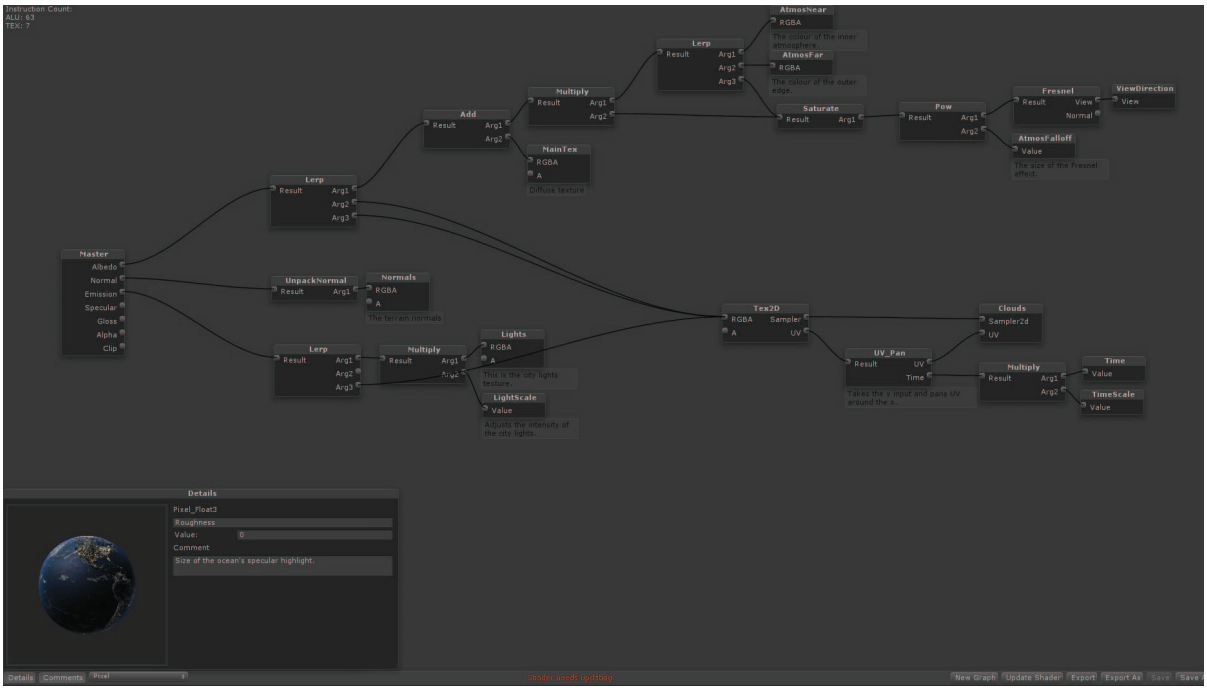


Step 9 - Making the clouds block out the city lights

Make one **Lerp**. Move the **Lights**, **LightScale** and **Multiply** nodes to the right a little and place a **Lerp** between the **Mulitply** and the **Master** node. Wire the Result from the **Multiply** into the Arg1 of the **Lerp**.

Run the RGBA output of the **Tex2d** of the Clouds into the Arg3 of this **Lerp**.

Wire the result of the **Lerp** into the Emission channel of the **Master** node.

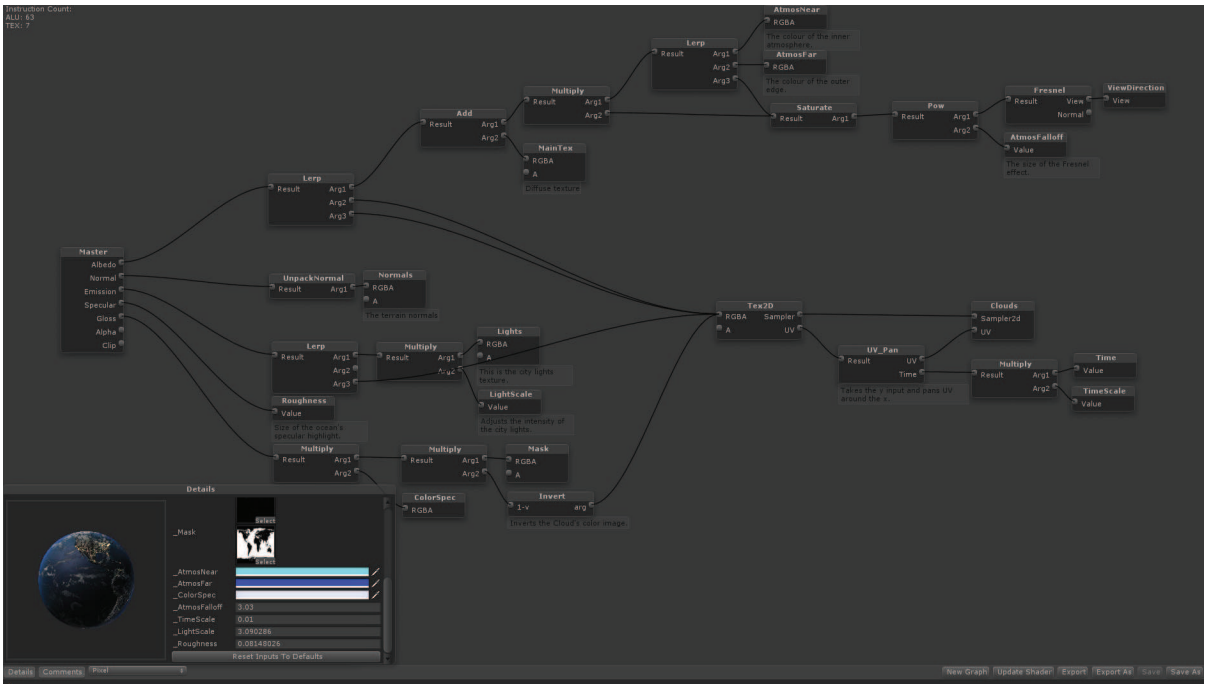


Step 10 - Gloss and Spec

Make a **Float.** Rename it "Roughness". Wire it into the Specular input on the Master node.

Make the following nodes- **Multiply**, **Multiply**, **Sampled2d**, **1-v**, **Color**. Rename the Sampled2d node to "Mask", and the Color to "ColorSpec".

Wire them up as shown in the image below.



Notice that the Clouds **Tex2d** output is wired into the **Invert**.

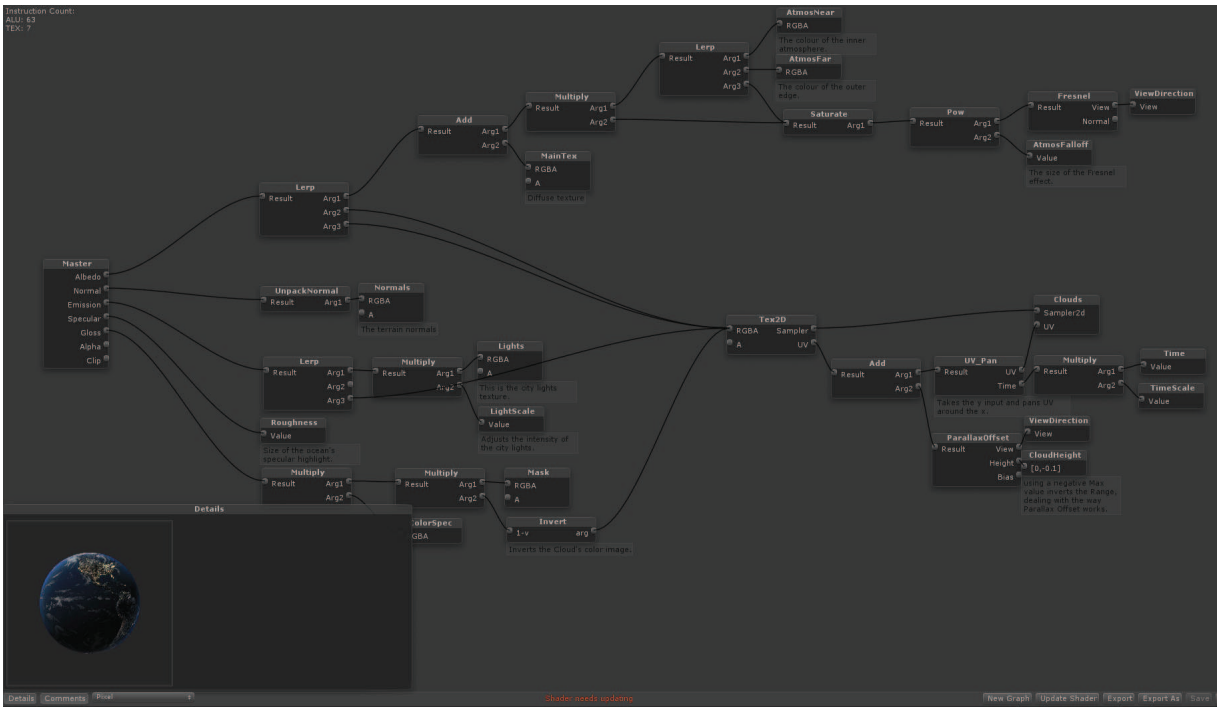
Step 11 - Clouds in the air

Make an **Add**, a **Range**, a **ViewDirection** and a **Parallax** node.

Looking at the UV_Pan output, place the new Add node between there and the Tex2D of the Clouds branch. Wire it in so it's between the UV_Pan and the Tex2d. Wire the result of the ParalxOffset into the second input of the Add... and then run the ViewDirection into the View input on the ParallaxOffset.

Rename the Range "CloudHeight" and wire it into the Height input on the ParallaxOffset, and then go to its details.

Change the default range from (0 to 1) to (0 to -0.1).



When you press Update Shader, you should see a bright pink ball. This means we've come to the limit of what Shader Model 2.0 shaders can do. Click on the Master node, chose Shader Target 3.0 and update the shader.

Step 12 - Aniso Cloud Power

Make the following nodes - **Lerp**, **Lerp**, **Normalize**, **ViewDirection**, **Float4Const**, **-v** and **Range**.

Rename the Range, "AnisoCloudPower".

Move the **Normal** and **UnpackNormal** modes to the right a bit, so we have room to add the rest in. Refer to the image below.

Place 1 Lerp between the **UnpackNormal** and the **Master** node. Wire the **UnpackNormal** into the Arg1 of this **Lerp**. The wire the other new **Lerp** into the Arg2 of the previous one. Run the **Normalize** into the second **Lerp**'s Arg1, and the **ViewDirection** into the **Normalize**. Now wire the **Float4Const** into the Arg2 of the second **Lerp**. In the details for this constant, change the z value to 1. This simulates an image where the RGB's B channel is blue, and the red and green channels have no data.

Run the AnisoCloudPower into the -v and then run that into the second Lerp's Arg3 input. The AnsioCloudPower is the slider you can use to tune the way the clouds receive light. Go into its details and make its range 0 to 0.1.

Now wire the Cloud's Tex2d RGBA output into the last remaining available channel in the first Lerp. Once you wire this Lerp into the Normal channel on the Master node, we're all done.