



Lume shader set for Softimage (XSI) version 5 or higher.

These tutorials can also be found on line at
http://rowu-media.nl/content/prive/tutorials/Lume/Lume_Portal.html

All tutorials were based on the 1.3 release of the shaders and XSI 5.x.
Some material is slightly rewritten or things are added to address new features.

Find the latest version of the shaders at
<http://www.rowu-media.nl/content/web/tutsportal.html>

or
the appropriate thread on XSIBase.com

© Rob Wuijster, The Netherlands aka XSIBase/Rork, updated November 2009 (Lume Glare info)

To get the legal stuff out of the way, you're NOT allowed to copy from-, alter- and/or re-edit this document without an written agreement from me. You ARE allowed to digitally distribute this document, as long as the former is taken into account. That's it. Happy reading.

Creating an ocean with the Lume Toolset

Introduction:

For the ocean scene we start very simple with a polygon grid, a surface sphere and some copies of a polygon cylinder to create some wooden posts for an interesting scene.

So here it goes:

-1. *Get>Primitive>Polygon Mesh>Grid*. Make it fairly large, say UV length 2500x2500 units. No need to change the subdiv values. Rename it to "ocean". Assign a basic Phong material to it for now to get rid of the default scene material.

-2. *Get>Primitive>Surface>Sphere*. Make the radius 1250 units. No need to change the subdiv values. Rename it to "sky". Assign a basic Constant material to it.

-3. *Get>Primitive>Polygon Mesh>Cylinder*. Duplicate this a couple of times, scale them and create some wood posts. Create a *group* from all wood posts and assign a basic blinn material to the group.

-4. Move the camera around in the camera view until you have a decent frame.



Note:

You should get yourself in the habit of saving camera views, by working with the "Memocam" buttons. Not only you can switch quickly between camera angles, but also FOV values and more are stored with the camera angles. Also every view in every port has it's own settings.

Now we have a basic scene set up and are ready to set up shop... err.. shaders ;)

Starting of with the sky:

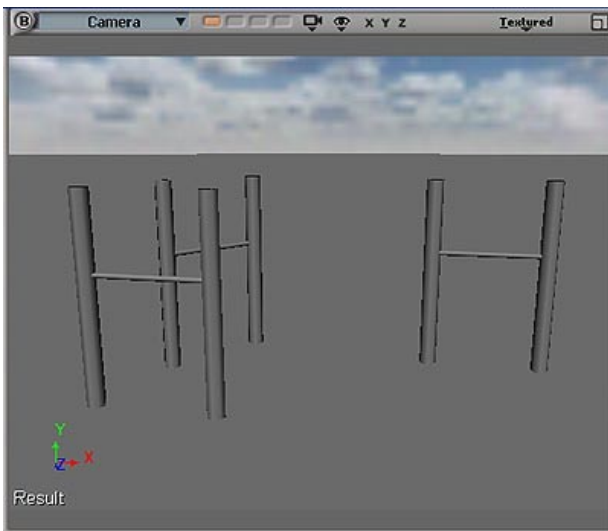
Select the "sky" geometry and in the "Render" module select "Modify>Shader". In the PPG change the color to full white.

Now assign a spherical texture projection by *Get>Property>Texture Projection>Spherical*. Also get a texture by *Get>Property>Texture*. Just choose a nice texture with some clouds. Keep the aspect ratio of the image in mind when wrapping the sphere or you get distorted clouds.

Assign the texture by going into the *Render tree* or by selecting "Modify>Shader" and pressing the red link next to the color sliders in the constant material PPG. Now click on the "New>New from File" button and browse to the location of the cloud texture.

Note:

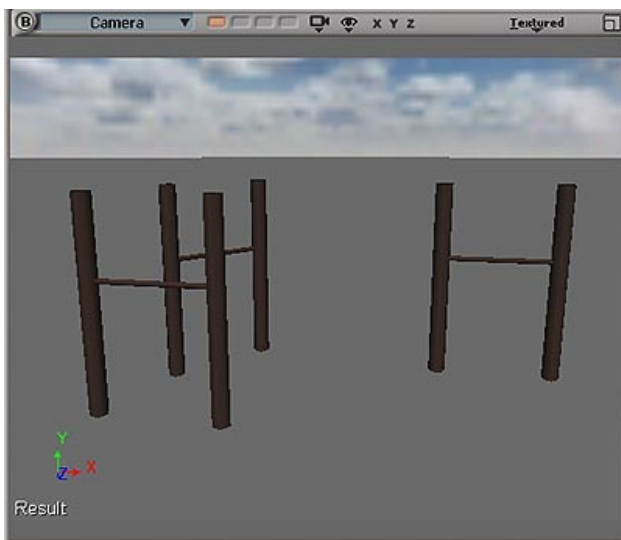
Open up the Texture Editor to see if the texture wraps nicely over the sphere, or have a look in a textured window. Remember the sea is reflective, so if your texture doesn't wrap well, this will show in the reflection.



Putting some color on the wood posts:

Select the "wood posts" group and open the shader by selecting "Modify>Shader". Change the diffuse and specular colors of the Blinn material into something brownish. Or you can put a nice wood texture on it, for now it's OK.

For creating a Dry-Wet effect for the wood posts we need the "Wet" shader, see "Making the wood posts look wet underwater" on page 8.



Creating the ocean:

This will have us working on the *Rendertree* because it's easier linking up the several shaders the ocean material is made of later on. So everybody still scared of the *Rendertree* this is a good time to get started ;)

It's a good idea to open the *Lumeshader shelf* so we can drag and drop the presets into the *Rendertree*. The shelf can be found under "*Application>Toolbars>LumeTools Collection*" in the topbar menu.

Open up the *Rendertree* by selecting the *ocean* geometry and pressing "7" on your keyboard or changing the front view into the *Rendertree* window.

Move the shelf over to the *Rendertree* for easy Drag&Drop of the presets. Now it's getting interesting :)

The *Ocean* shader is made up out of several shaders, all linked together to create the final material. We need the following shaders: "*-Ocean, -Submerge, -WaterSurface and WaterSurfaceShadow*" and a simple phong node to wrap things up.

Find the shaders in the shelf and D&D them into the *Rendertree* window. OK, now we have the basic shaders, let's connect them and set up the colors for them.

-1. Setting up the Submerge shader.

This will simulate the depth of the water. Grab the *output* of the "*Submerge*" shader and plug it into the *Volume slot* of the "*Material*" node. This is the the orange node most right of the *Rendertree*. You might have to open the list of available nodes by clicking on the "down-arrow" icon.

Open the "*Submerge*" shader by double clicking and set the following values:

- RGB: 0.00, 0.10, 0.20

- Vertical Gradation: 3.0

- Density: 8.0

This will get us a fairly dark blue environment that rapidly grows darker as you getting deeper.

-2. Setting up the "WaterSurface" shaders.

Start by grabbing the *output* of the "*WaterSurface*" shader and replace all connections of the Phong node into the "*Material*" node.

Link the output of the "*WaterSurfaceShadow*" node to the "*Shadow*" slot of the "*Material*" node.

For both nodes set the "*Transparency*" values to RGB 0.3, 0.3, 0.3

-3: Linking the Phong shader to the "WaterSurface" node.

Now link the output of the Phong node into the "*Surface_Material*" slot of both the "*WaterSurface*" node and the "*WaterSurfaceShadow*". Select and open the *Phong* shader and turn off the *Transparency* and *Reflection* values.

This stuff is covered in the "*WaterSurface*" shaders part of the PDF.

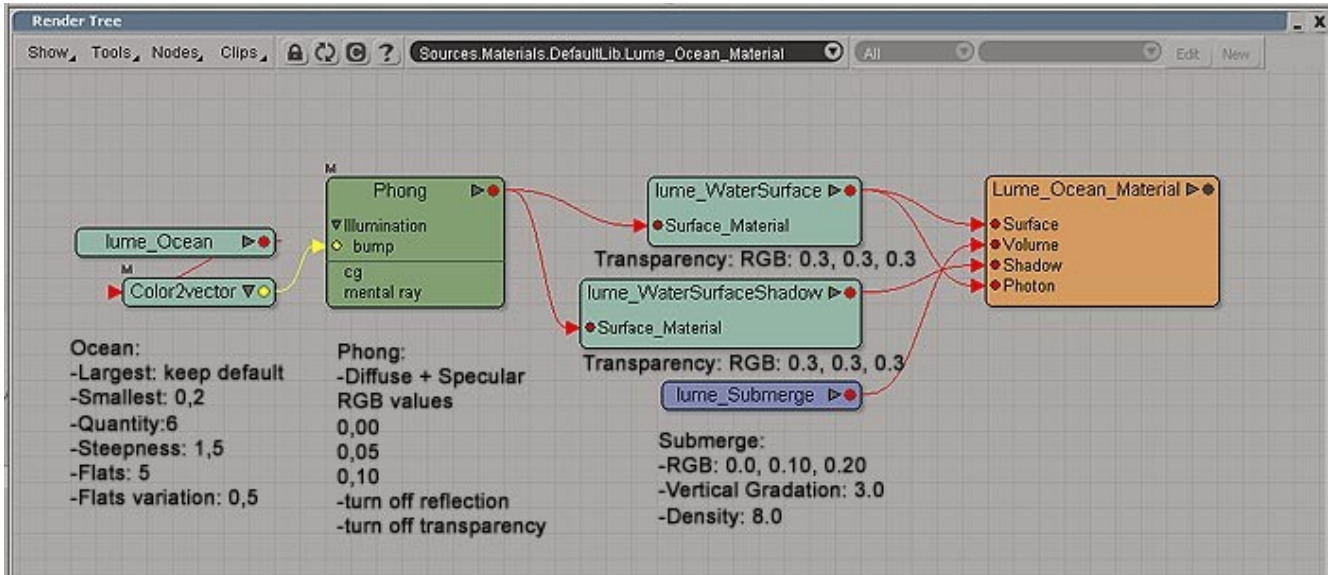
You can use the *Diffuse* color of the Phong node to color the ocean end result Set the value of the *Diffuse* color close to the color of the "*Submerge*" shader, maybe even a bit darker, like RGB 0.0, 0.05, 0.10. Copy this color over to the *Specular*.

-4. Adding the waves with the Ocean shader.

The *Ocean* shader creates the actual waves. Grab the *output* of the *Ocean* shader and plug it into the *bump slot* of the *Phong* shader. It will automatically add a *color2vector* node in between.

Open up the *Ocean* shader and set the following values:

- Largest: keep default
- Smallest: 0.2
- Quantity: 6
- Steepness: 1.5
- Flats, size: 5
- Flats, variation: keep default.



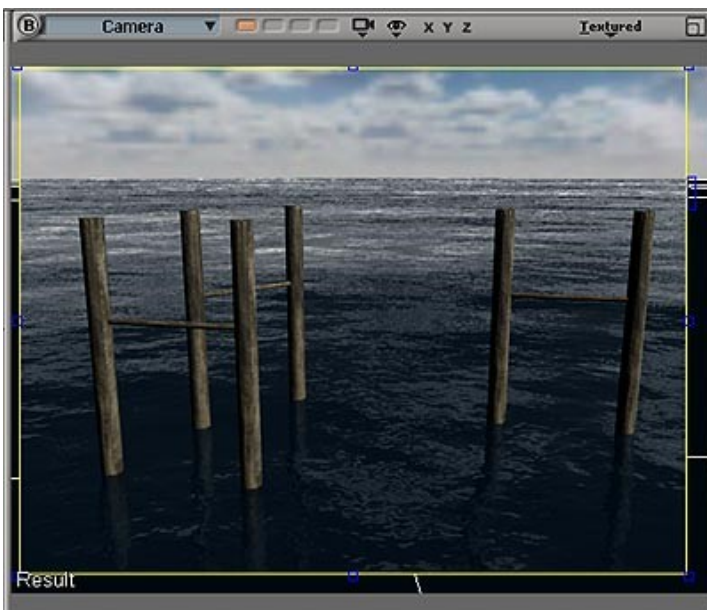
If you don't get it to work, try this [ocean preset](#) file.

Note:

Unless you're running Softimage(XSI) version 7.x or higher, with disconnected nodes, be sure not to click on the refresh button until you're finished, or your tree is gone. And nope, the Rendertree has NO undo !!!

Rendering:

Now we have set up all the shaders we can have a look with the render region Before we do that change the "Ambience" color to full black by selecting "Modify>Ambience". Now drag a render region in the camera view and see the ocean being rendered :)



Creating an ocean underwater scene with the Lume Toolset

Introduction:

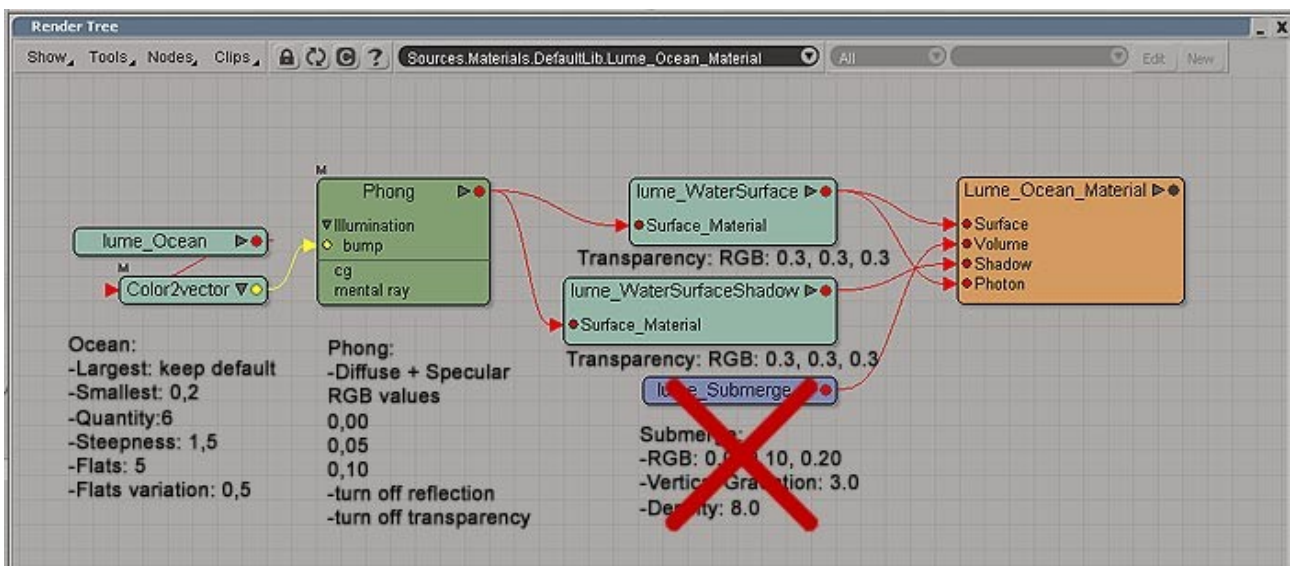
The setup for a underwater scene when working with the Lume shaders isn't that different from the previous ocean setup.

So here it goes:

Translate the camera so it's underwater. Then select the "ocean" geometry and open up the *Render tree*. Now disconnect the *Submerge* shader from the Material node. We're gonna use a different method to create "zee murky waters of zee deep" (insert footage of a certain Jacques Cousteau here) >:-)

Now we have a changed the basic scene set up and are ready to continue for the underwater effect.

Adding the Submerge shader as a Atmosphere:



Because we're gonna render an underwater scene we have to reverse some of the settings in the shaders or in the scene setup. We're going to use the *Submerge* shader as a *environment*, so we're adding it to the Pass. Select *Pass>Edit>Current Pass* and select the *Volume Shaders* Tab. Click on *Add*, select the *Submerge* shader and click OK to return to the PPG.

Be sure to have the *Submerge* shader selected and now click on the *Inspect* button. You can copy the RGB color setting from the old *Submerge* shader for now, you can always fiddle to get a different effect or colors. Change the "*Vertical Gradation*" and "*Density*" to 1.0.

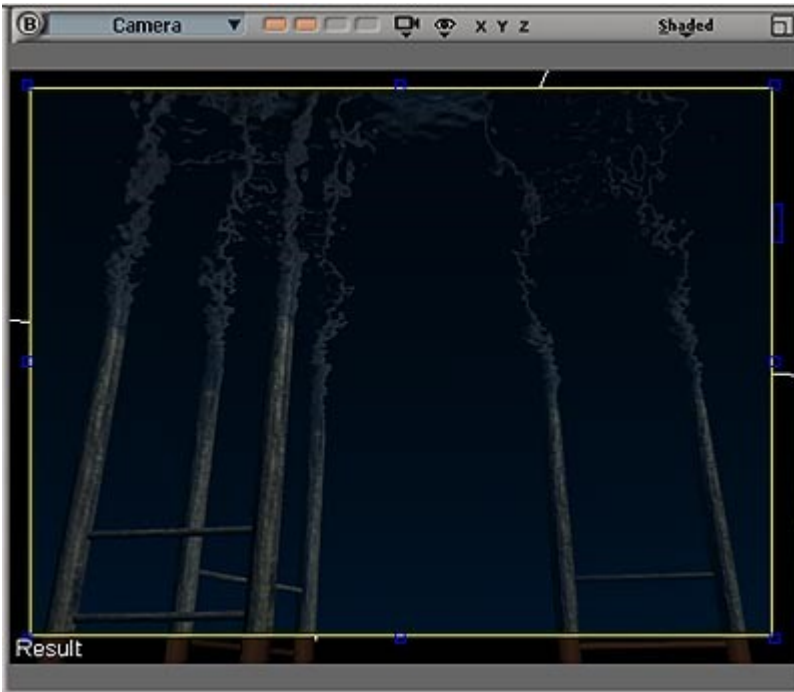
For the last change in the setup reselect the "ocean" grid, open up the *Render tree* and select the *WaterSurface* shader. Open it and in the PPG deselect "*Looking into the water*" and select "*Looking out of the water*".

There's really not more to it actually, you're done.

Rendering:

Now we have changed the setup of the shaders and the scene we can have a look with the render region. Be sure to have the "Ambience" color set to full black by selecting "Modify>Ambience". Now drag a render region in the camera view and see the underwater scene being rendered :)

That's it!!



Play with the "Density" and "Vertical Gradation" of the "Submerge" shader values to get more murky waters.

Making the wood posts look wet underwater

Introduction:

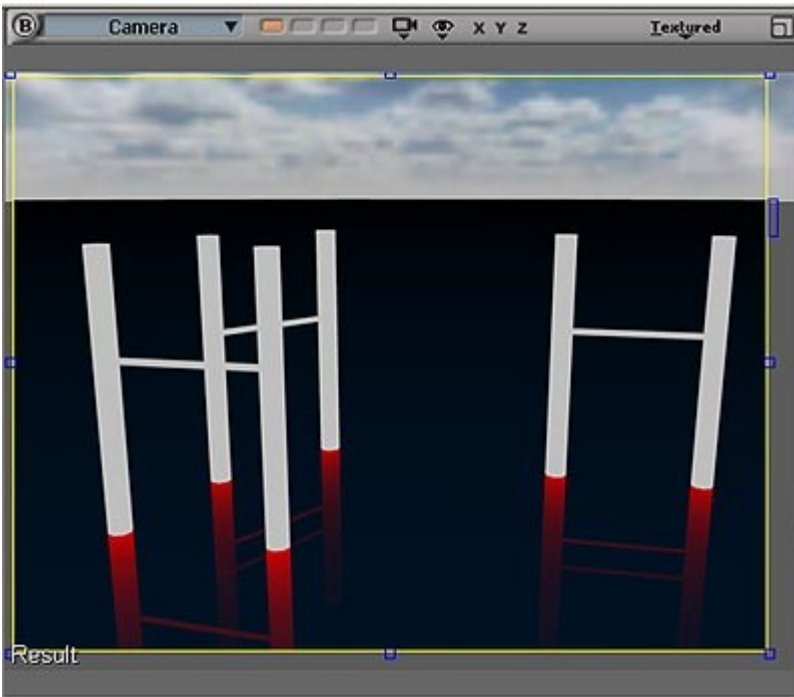
The "Wet" shader works in combination with the "WaterSurface" shader, so everything under the ocean will get a different shader/material/color assigned.

The best way to see this effect happening is to tune down some of the "WaterSurface" values. Select the "ocean" grid and open the *Rendertree*. Now double click the "WaterSurface" shader and in the PPG turn down the "Surface Material" to RGB 0,0,0 and the "Transparency" to RGB 1,1,1.

This will kill the reflection and transparency values for the ocean, leaving us with the "Submerge" effect and a look of the disappearing wood posts into the water.

Now for the Wet effect:

- 1. Select the group containing all wood posts we created earlier and open up the rendertree if not still open.
- 2. Get the "Wet" shader into the *Rendertree* and replace the blinn shader entirely for all connections to the "Material" node. Now for a good look what the "Wet" shader is doing, open it up and set the "Dry" material to a light gray color and the "Wet" material to a pure red color. Create a render region, it should look something like this:



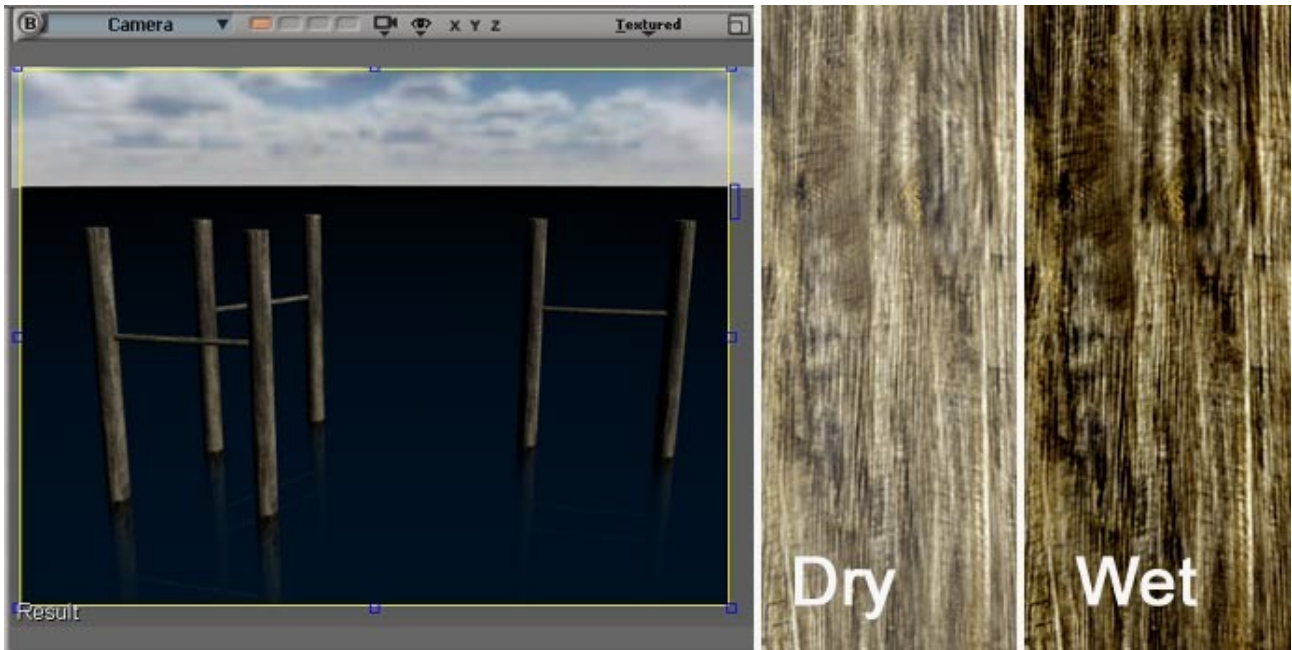
The combination of the "WaterSurface" and the "Wet" shader will "split" the materials in a above/under the water effect. Very nice indeed.

Creating the materials:

Now we know what's happening within the shader, it's now simple to get two extra nodes into the *Rendertree*. One blinn node for a more diffuse look of the dry wood, one phong node for the wet look of the wood underwater. After setting two sets of slightly different brownish colors, link the blinn node into the "Dry" slot of the "Wet" shader and the phong node into the "Wet" slot of the "Wet" shader.

For more realism you can also add two slightly different textures for the dry and wet look and add these to the blinn and phong material.

The render region would look something like this:



Lume Light shaders, how to create more realistic lighting

Introduction:

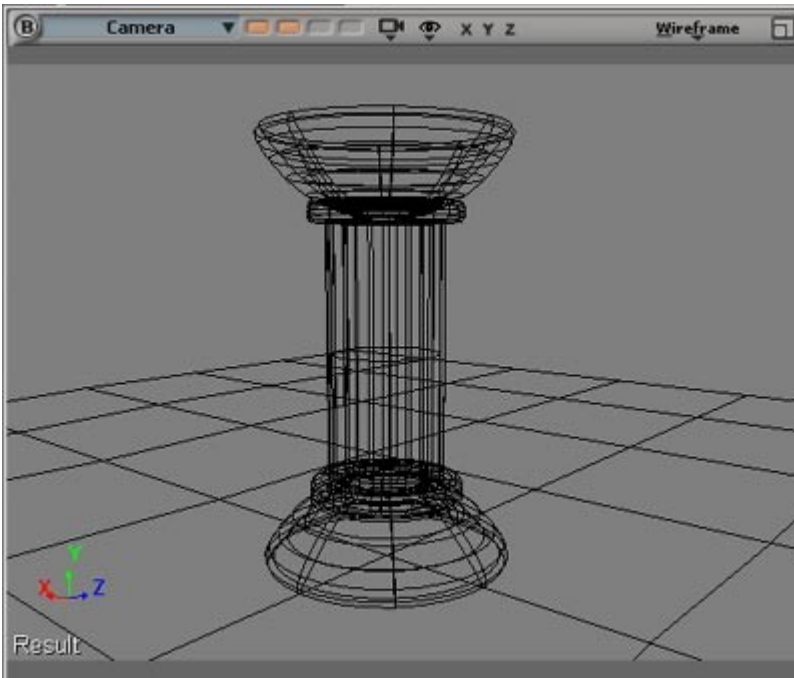
The Lume shaders contain some shaders to create more realistic lighting effects. The "*Lume_Illumination*" shader creates more realistic lighting, although since XSI some of the features are more or less obsolete (the inverse square falloff is now present in XSI as well) you still can make the shader to good use. Also you need it to get "*Lume_Glare*" working for the post process.

We'll look at the following steps:

- Scene setup*
- Lume_Illumination*
- Lume_Translucency*
- Lume_Glare*

So here it goes, scene setup:

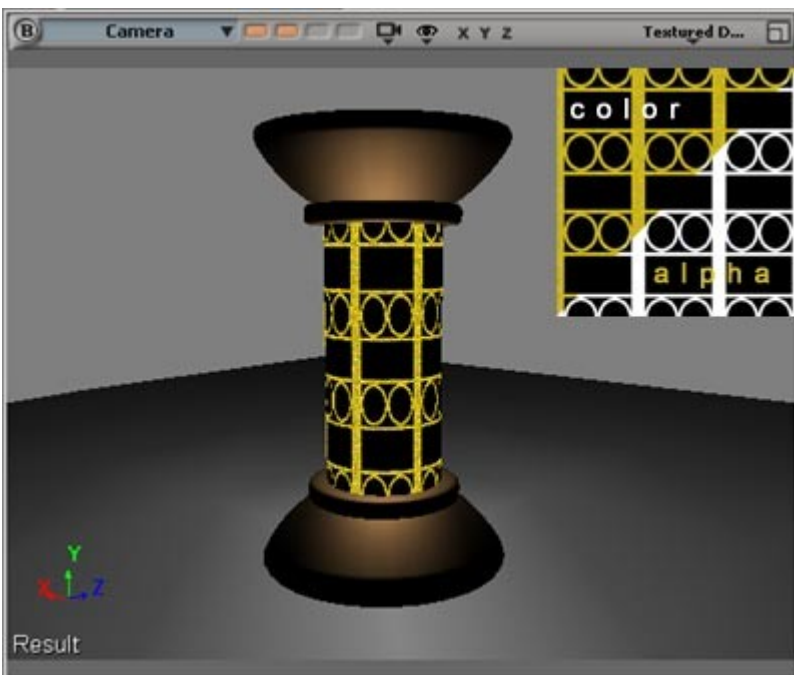
I created a simple scene based on the original Softimage|3D tutorial, containing a grid, two half spheres, two toruses and two cylinders. Stack these on top of each other so you get: grid, sphere, torus, cylinders, torus and sphere like in the image. If you don't like to build it, grab this [zipfile](#).



Before we start any rendering, set the scene *Ambience* to RGB 0,0,0 and delete the default scene light.

Now add some basic materials to each one of objects. Add a basic gray Lambert to the grid, and a golden yellowish Phong to the base parts of the lantern. Also add a basic phong to the outer cylinder and a basic phong to the inner cylinder, but with the transparency set to RGB 0.5,0.5,0.5.

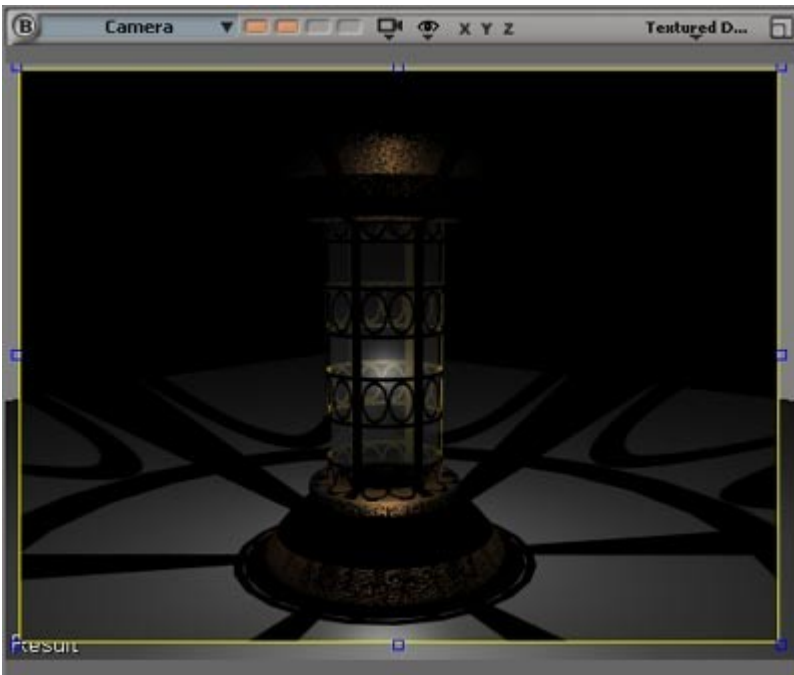
The outer cylinder will have a texture with an alpha channel driving the transparency of the object, so add the texture in the render tree.



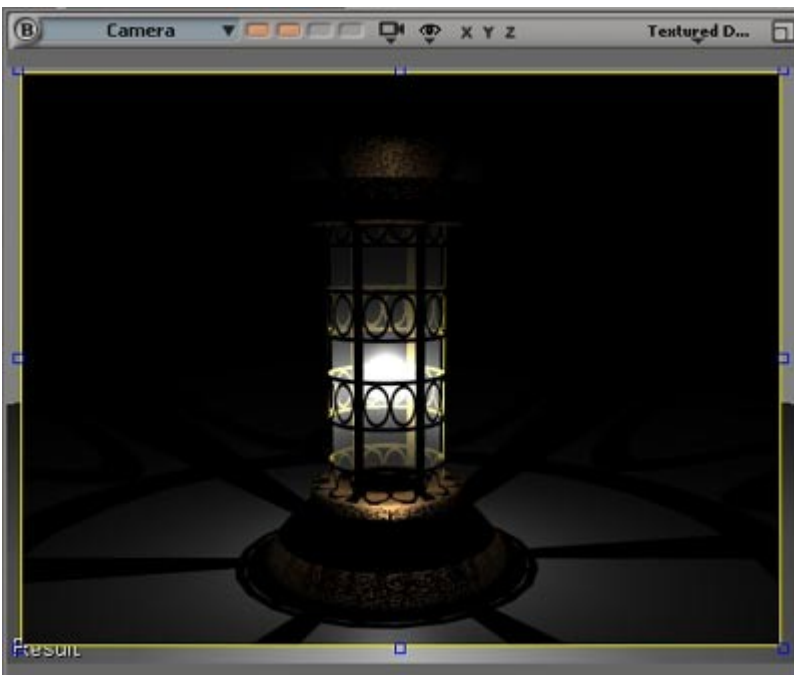
If you don't feel like going through all this trouble, download the [zipfile](#), it contains the model, the texture used and the presets for the rendertree setups.

Lume_Illumination

With all the objects given a basic material/texture, add a point light to the scene, and move it in the middle of the cylinders, about 1/3 up from the bottom. Open up the point light's PPG and set the intensity to 1, and turn on shadows, set the *Umbra* to RGB 0,0,0 to make the shadows black. Draw a renderregion to see what we got so far, nothing more realistic yet.



With the light still selected open the render tree and add the "*Lume_Illumination*" to the render tree. Now connect the "*Lume_Illumination*" shader to the *intensity* slot of the *soft_light* shader. Open the PPG of the "*Lume_Illumination*" shader and see what we have in there. Keep the *Brightness* to 1, but change the falloff type to *realistic*. In the render region you will see a glow where the point light is, and more natural looking shadows/falloff for the light. Changing the *Brightness* value will increase the amount of light, washing out areas of the render and increasing the falloff of the light.



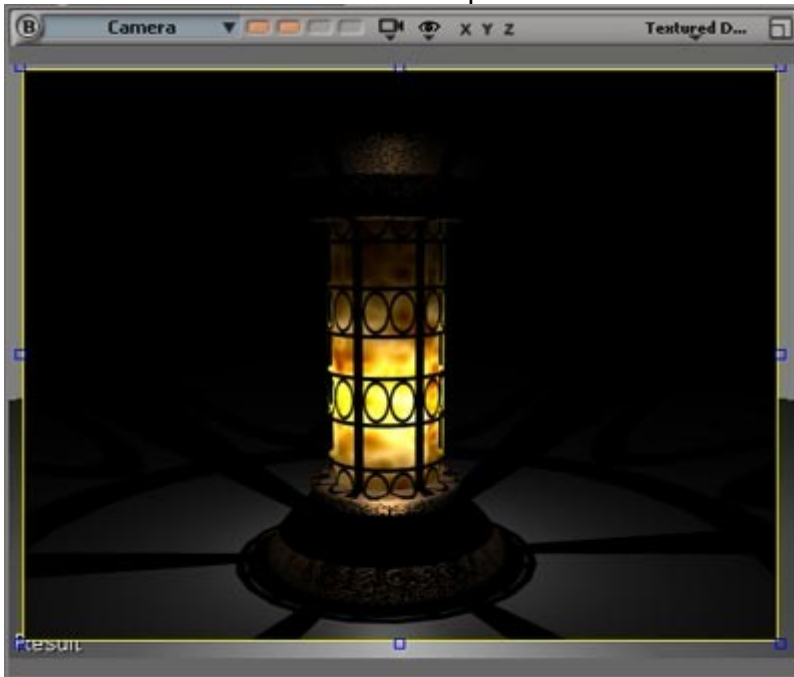
Lume_Translucency:

The inner cylinder still looks very dull, so we're going to spice that up. We can do that in two ways; the Lume shader or the XSI shader. During the Softimage|3D period there was no decent translucency solution available, so they created the Lume_Translucency shaders. In XSI we have one, so the Lume shaders are obsolete in this.

With the inner cylinder selected, open up the render tree and add the translucency and a cloud shader by *Nodes>Illumination>Simple_Translucent*, and *Nodes>Texture>Cloud*.

Now connect the *Cloud* shader into the *diffuse* port of the *Translucency* shader, and connect the *Translucency* shader into the Surface slot of the Material node. Keep the existing Phong node connected to the rest of the Material node.

Refresh the render region to see what we get. The translucency gives us a nice falloff in the cylinder, and the cloud texture makes a nice pattern.



Note:

If you still want to use the Lume shaders, add the *Lume_Translucent_Material* and the *Lume_Translucency* to the rendertree.

Now connect the *Lume_Translucent_Material* to the Surface slot of the Material node, connect the *Lume_Translucency* to the *diffuse*, *ambience* and *ambient* slots of the *Lume_Translucent_Material*.

Now connect the *Cloud* shader into the *surface_material* and *diffuse* ports of the *Translucency* shader. In the *Translucency* PPG add the point light in the *Lights* list.

Lume-Glare:

NOTE nov.2009: Since the introduction of XSI 7.0, the "Quality" slider only works with 3 and up. The "preview" settings of 1 and 2 don't work anymore. All the other options still work, but might need slightly different settings to get to the same result as shown in the images.

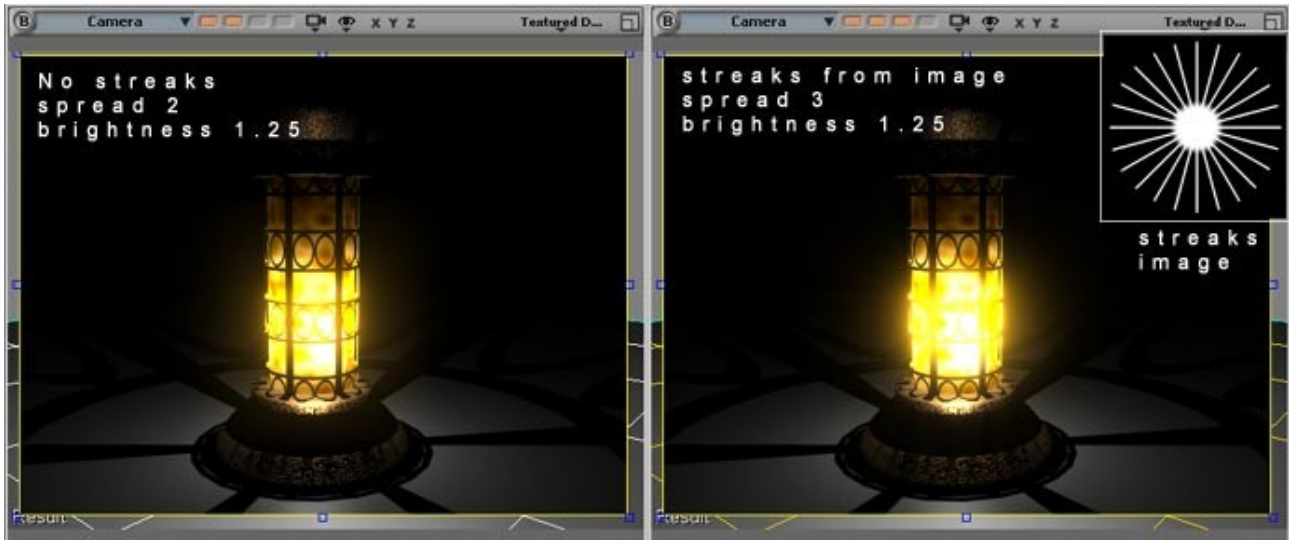
It already looks a lot better, but what we want is the overexposure we see in photographs. We can add that effect with the Lume_Glare shader. This shader will add to the render time, and setting the values too high can result in long render times.

Now go to the render panel and go "Pass>Edit>Current Pass" and open up the "Output Shader" Tab. Add the Lume_Glare shader by clicking the "Add" button and browsing to the Lume shader preset. The render region is updated directly with the newly added shader, and the overexposure is already

seen.

You can work with the overall effect in two ways. You can adjust the spread value in the Lume_Glare PPG, set the Quality drop down to Average for now, or adjust the Brightness value in the Lume_Illumination PPG. Maybe even both to tune the effect just right.

For now open up the Lume_Illumination PPG and change the spread value to 1,25. This will add a little bit more falloff to the shadows and will overexposure the centre of the light even more. The Lume_Glare shader will add the visible halo around the light.



Open the Lume_Glare PPG again to see what more we can do. The Quality slider will improve the quality of the effect. The downside is that the render times will increase dramatically, it's up to you what kind of effect you want.

The Overlay Only will show you just the Glare effect, but AFTER the entire render is done. It will stay a post effect.

It's a good way to see the effect separate from the final render, it's easier to tweak that way.

The Verbose option will show you the status of the Glare render, you have to have Logged Messages>Progress turned on in the render region option to see this.

The Streaks Tab will let you add an B/W image to the Glare effect. Turn on the Streaks option and add a new image with the New button. Because of the fact that the image is mixed with the Glare effect (by the contrast slider), the total result will be somewhat dimmer. You'll have to turn up the Spread and even the Brightness value a bit to see the effect better.

The Scale slider doesn't seem to do much with the rendered result.

That's it, a more realistic light setup with the help of some of the Lume shaders.

Lume Material shaders setup

Introduction:

Material shaders replace the default material in the XSI Rendertree for most cases. Although with all the new shaders coming out for the new MR versions in XSI, a lot of these shaders can still be made to good use and in some cases are still unique. Not bad for a bunch of shaders written for the game "Riven" published in 1997!!

In this tutorial we discuss most of the shaders:

- Lume_Facade
- Lume_Glass
- Lume_Glow
- Lume_Edge
- Lume_Metal
- Lume_Stain
- Lume_Translucency
- Lume_WaterSurface
- Lume_Wet

So here it goes, *Lume_Facade*:

-The Facade shader was meant to do the following thing. Assigned to a plane, cylinder or cube, and "colored" with a texture the shader would always have the texture face the camera. Also the shadow would always be correct to the light sources. Think forest of trees (build up with grids) with a free moving camera and never see the illusion distorted by perspective changes. Unfortunately this shader seems to be broken beyond repair after version 6 was released :-)

It should work like this:

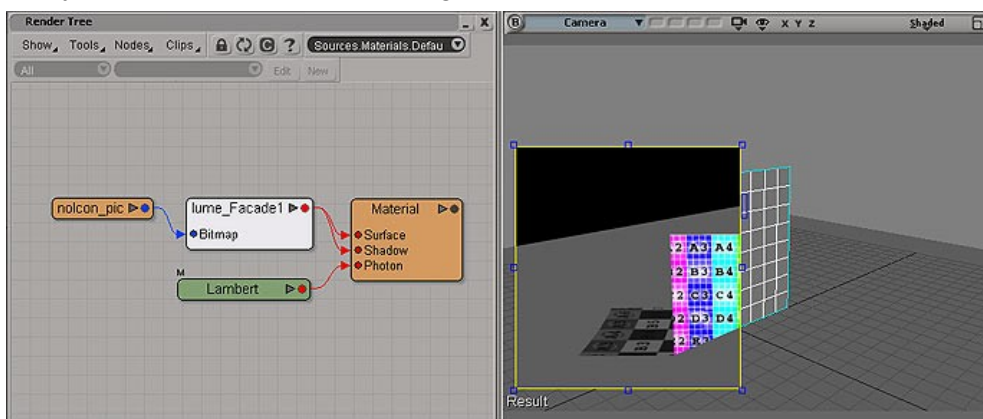
-Create a *polygon grid*, and scale it up to 50x50. Create *another polygon grid* and rotate this 90 dgr on it's X axis, so it's standing upright. Move this grid to align it with the "ground floor"

Assign any material to this grid, it doesn't really matter what you choose. Now open the "*Rendertree*" and open the "*LumeTools Toolbar*" and drag the "*Facade*" shader into the "*Rendertree*". Connect the "*Facade*" shader to the "*Surface*" and "*Shadow*" slots of the "*Material*" node.

Open the "*Facade*" shader and assign a material to it, for testing the "*Nolcon*" would do fine. Now turn on the shadow in the default light in the scene.

Draw a render region and play with the camera. As you can see the shadow does work, but the texture is projected in the wrong way. I've tested several ways to fix this, including "Texture Space Generators" but this doesn't fix the problem.

If anyone comes up with a working solution I will add it to this PDF and re-release it.



Lume_Glass:

Soon.....As is Lume_Metal.

Lume_Glow:

See "*Lume Light shaders, how to create more realistic lighting*" on page 9.

Lume_Edge and Lume_Edge_Shadow:

The "*Lume_Edge*" shader is a "quick 'n dirty" way to get rough edges off of a bumpmap, or sort of a "poor man's" displacement. But the accompanying "*Lume_EdgeShadow*" will also show the effect in your shadows of the object.

So create a sphere and set a spherical projection, and create a grid. Assign a *Phong* material to both objects and open the *RenderTree*.

Now select the sphere and get a "*Lume_Edge*" and "*Lume_EdgeShadow*" node.

Assign the "*Lume_Edge*" to the "*Surface*" and "*Photon*" slots of the "Material" node. Connect the "*Lume_EdgeShadow*" node to the "*Shadow*" slot of the "Material" node.

Reconnect the existing Phong node to the "*Surface_Material*" slot of both the "*Lume_Edge*" and "*Lume_EdgeShadow*" node. Also connect the Phong node to the "*Transparency*" slot of the "*Lume_EdgeShadow*" node.

We need something for bump, so create a *Fractal* node and connect it to the "*Bump*" slot of the Phong. Turn on "*bump*" and set the UV repeats to something useful.

If you now render a region, the effect is not visible yet.

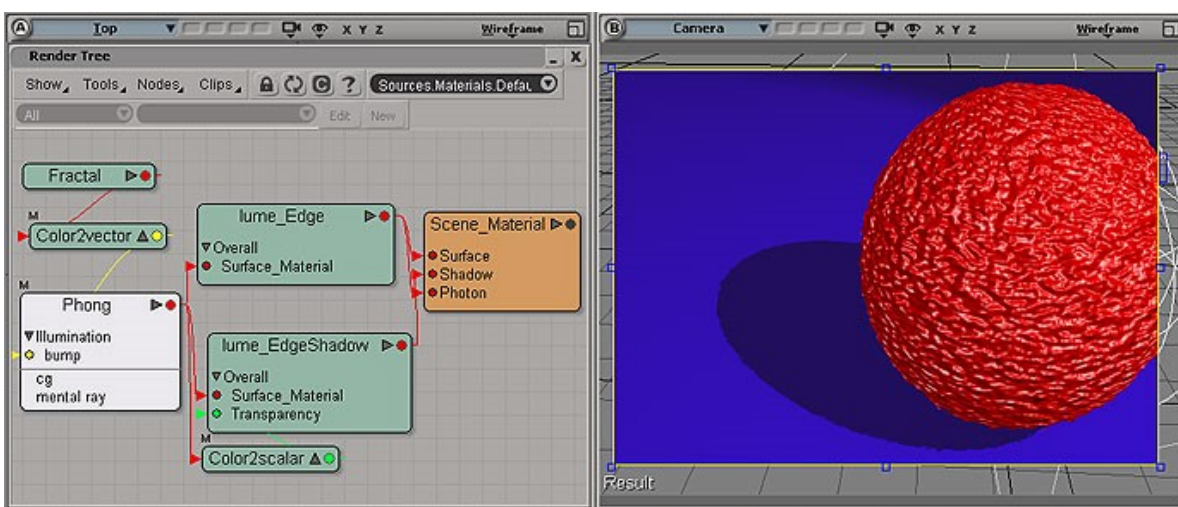
Open the "*Lume_Edge*" PPG and set the "*Amount*" to 3. Leave the "*Transparency*" set to 1. "*Blur*" will blur the edges of the effect. Small values will do, keep it set to 0.

Select the "*Effects & Mixing*" tab and activate the "*Noise Effect*". Keep the "*Influence*" to 1, and set the "*roughness*" to 0.5 and the "*Scale*" to 0.1.

Activate the "*Color Effect*" and set the diffuse color the same as the one from the phong shader. Drag and drop would do fine here..... ;-)

Use the same values for the "*Lume_EdgeShadow*" PPG, and also add a spotlight.

Re-render the region and look at your shadows, they have rough edges now too...!!



You can get the preset [here](#).

Lume_Metal:

Soon...As is Lume_Glass.

Lume_Stain:

The "Lume Stain" is an interesting shader. Assigned to an object it will change the values of the camera rays as soon as they enter the shaded object. To do something with this change we can use it in combination with the "Wet" shader.

If you have looked at the "Dry-Wet tutorial" already, you've seen you can change shaders by assigning the "Wet" shader. The "Stain" shader does the same thing, but can be assigned as a separate shader instead of the "Ocean" setup.

As an example we create a simple polygon cylinder. Scale it a bit in Y and copy it three times. Now scale these a little bigger in XZ and a lot in Y so you get three rings. Assign all objects a simple basic shader, like phong or lambert.

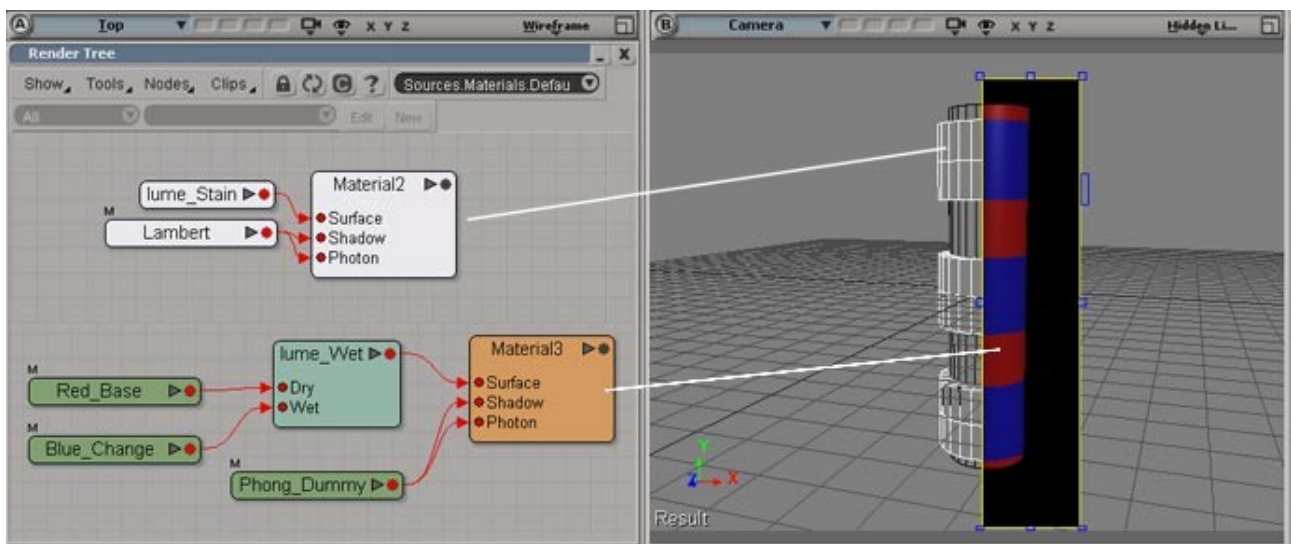
Again open the "Rendertree" and open the "LumeTools Toolbar". First start with the cylinder sticking inside the rings.

Drag the "Wet" shader into the "Rendertree" and connect it to the "Surface" slot of the "Material" node. Next get two new Phong nodes from the "Nodes>Illumination" menu. Make one red and one blue, and assign these to the "Dry" and "Wet" slots of the "Wet" shader.

If you now draw a render region nothing really happens, it will render the cylinder in the color assigned to the "Dry" slot and the rings the default gray.

Now select one of the rings, drag and drop the "Stain" shader into the "Rendertree" and connect it to the "Surface" slot of the "Material" node.

If you still have the render region active you'll see the ring disappear and render the overlapping area of the cylinder in the blue 'wet' color. Copy the material over to the other two rings and see two more blue rings appear. This might be a handy shader for changing shaders/textures of an object moving through another object.



Note:

You could turn down transparency on the dummy materials just to be sure the rings don't end up in the alpha channel in some custom pass ;-)

Lume_Translucency:

See "*Lume Light shaders, how to create more realistic lighting*" on page 9.

Lume_WaterSurface:

See "*Creating an ocean with the Lume Toolset*" on page 2.

Lume_Wet:

See "*Making the wood posts look wet underwater*" on page 8.

Lume Output shaders setup

Introduction:

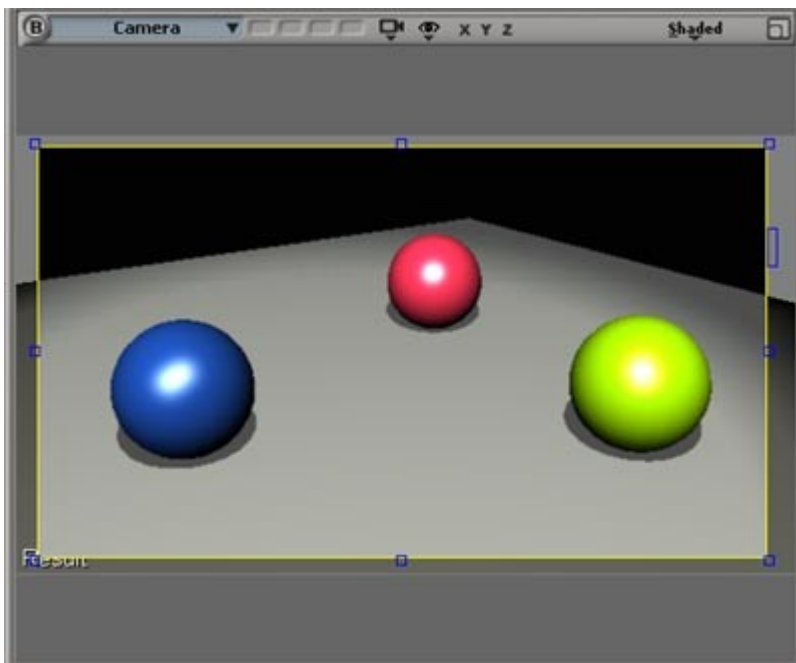
Output shaders are calculated **after** the rendering has finished, as a post-processing effect. So that means the shader is not added in the scene itself, but in the scene pass. In this tutorial we discuss all three output shaders:

- Lume_Adjustments
- Lume_Bumpcapture
- Lume_Glare

So here it goes, Lume_Adjustments:

-1. Create a simple scene with a grid and three spheres. Add some basic materials to each one of them, a plan color for the grid, some bright red, blue and green for the spheres. Add a spotlight to light the scene.

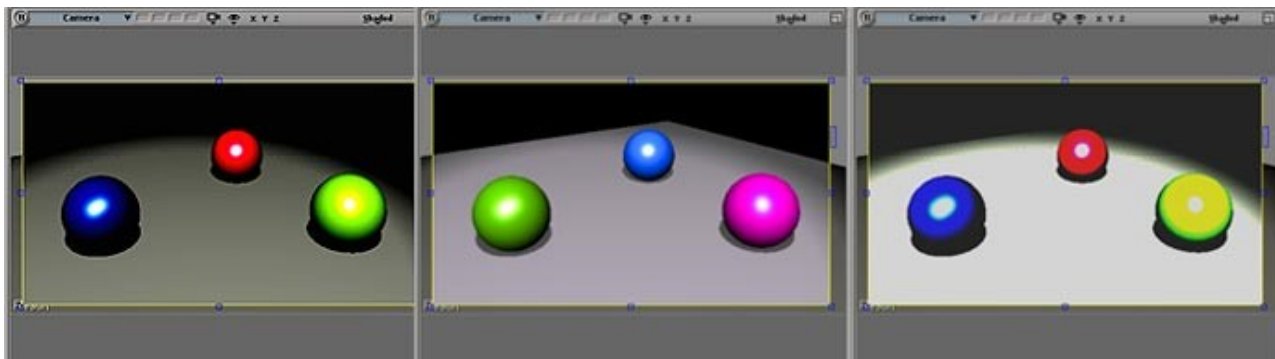
After that set up a render region to see what you got so far.



-2. OK, now got to the render panel and go "*Pass>Edit>Current Pass*" and open up the "*Output Shader*" Tab. Add the Lume_Adjustments shader by clicking the "Add" button and browsing to the Lume shader preset. Select it in the Output Shader Stack and click on the "Inspect" button.

-3. Now it's time to play with the sliders, they work as expected when you're used working with Photoshop or alike.

Here you have changed Brightness/Contrast, Hue/Saturation and Levels:

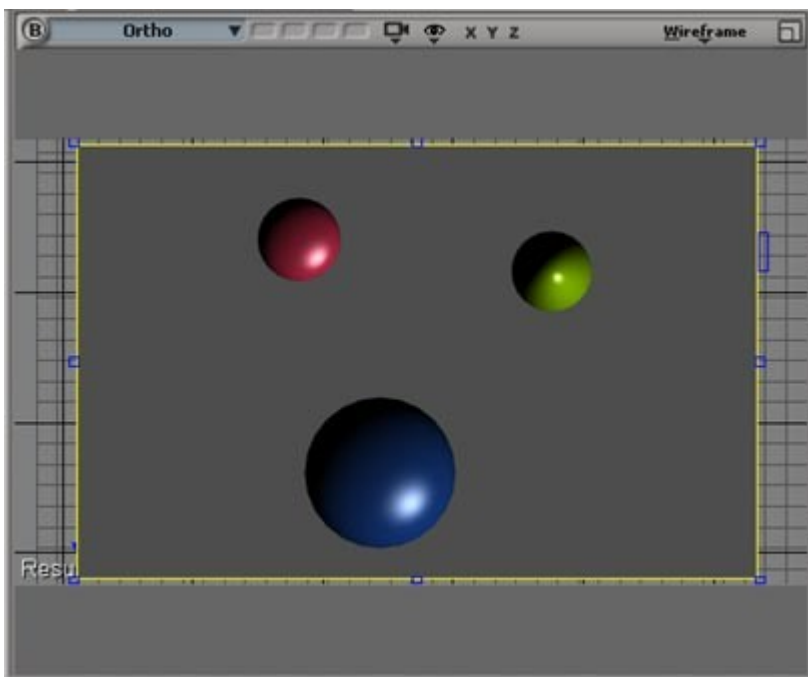


Lume_Bumpcapture, creating the texture and using it as a displacement:

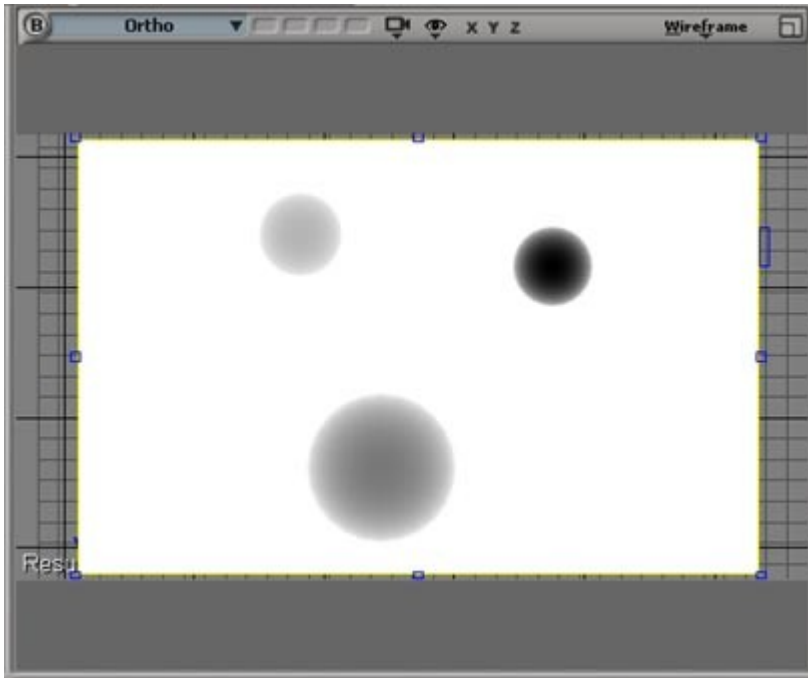
The bumpcapture shader renders a B/W depthbased image. so it can be used a texture for the "Bumpmap generator or as a displacement texture.

-1. Stick with your simple scene of the grid and three spheres. Just scale every speher a bit in Y and move them halfway down till the center is stuck on the grid.

You want to have a Top view render of the scene, so make the Top view full screen (F12 hotkey) and do a print screen after a render region, or get a orthographic camera. I'll do the last thing, so "Get>Primitive>Orthographic Camera" and change the camera view to the new camera. Set the Camera Interest to 0,0,0 and the Camera itself to 0,10,0. If the camera flips, select it and rotate -90 degrees in the X-0axis. Grab the entire hierarchy at the root and move it so its about center of your scene. Open the Camera PPG and play with the "Ortho Height" slider for the amount of camera view.



2. OK, now go to the render panel and go "Pass>Edit>Current Pass" and open up the "Output Shader" Tab. Add the Lume_Bumpcapture shader by clicking the "Add" button and browsing to the Lume shader preset. The render region is updated directly with the newly added shader.



-3. The rendering can best be used a displacement map. It's also best to use this shader for simple geometry replacements.

Lume-Glare:

See the *"Lume Light shaders, how to create more realistic lighting"* tutorial on page 9.

Lume Lens shaders setup

Introduction:

Lens shaders are a way to adjust the rays send and seen by the camera. In this tutorial we discuss all three lens shaders:

- Lume_Distortion
- Lume_Night
- Lume_Wraparound

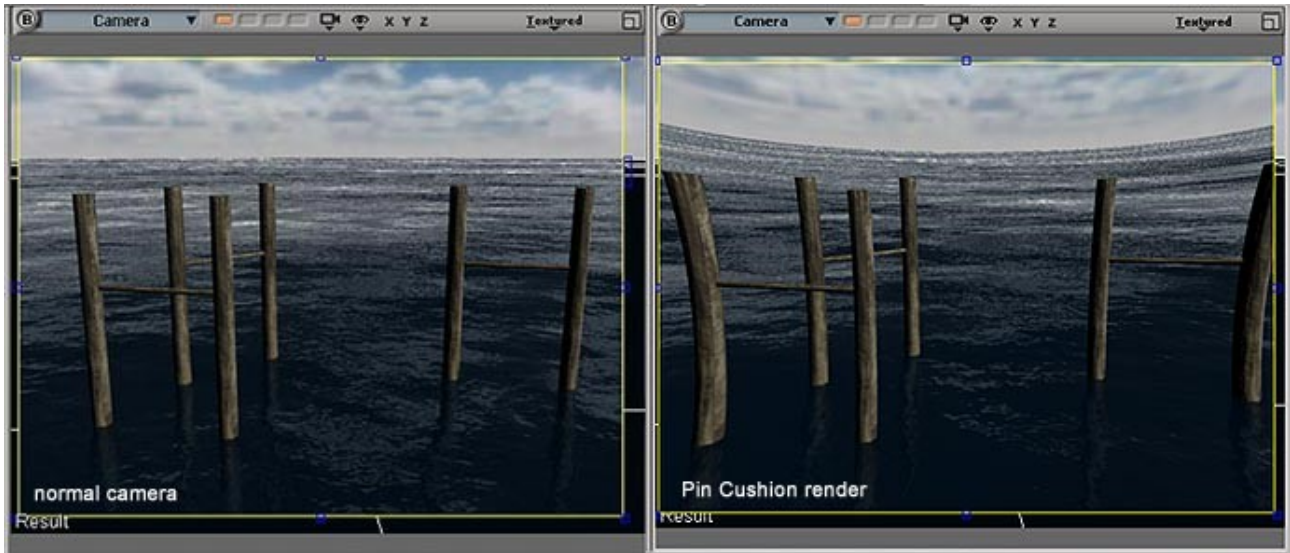
So here it goes, Lume_Distortion:

The *"Lume_Distortion"* lens shader is one of the most basic ones, with only three parameters. What it does is "bend" the rays so you get a "real life" warping of your render due to the way light is warped by using different lenses, e.g. like a fish eye lens.

I opened up my Ocean tutorial scene for this. Now select your camera in the explorer, and open up the *"Lens Shader"* Tab. Click the *"Add"* button and browse to the *"Lume_Distortion"* preset. Select it, and click *"Inspect"*.

We can adjust the *"Pin Cushion"* effect, this will "warp" the image towards the center while keeping the corners locked. *"Barrel"* will do the opposite, warping the image away from the center.

The *"Amount"* slider will determine the strength of the effect. Any values above 4 will seriously mess up your render, especially with the *"Pin Cushion"* option. The effect will be more visible away from the center of the image.



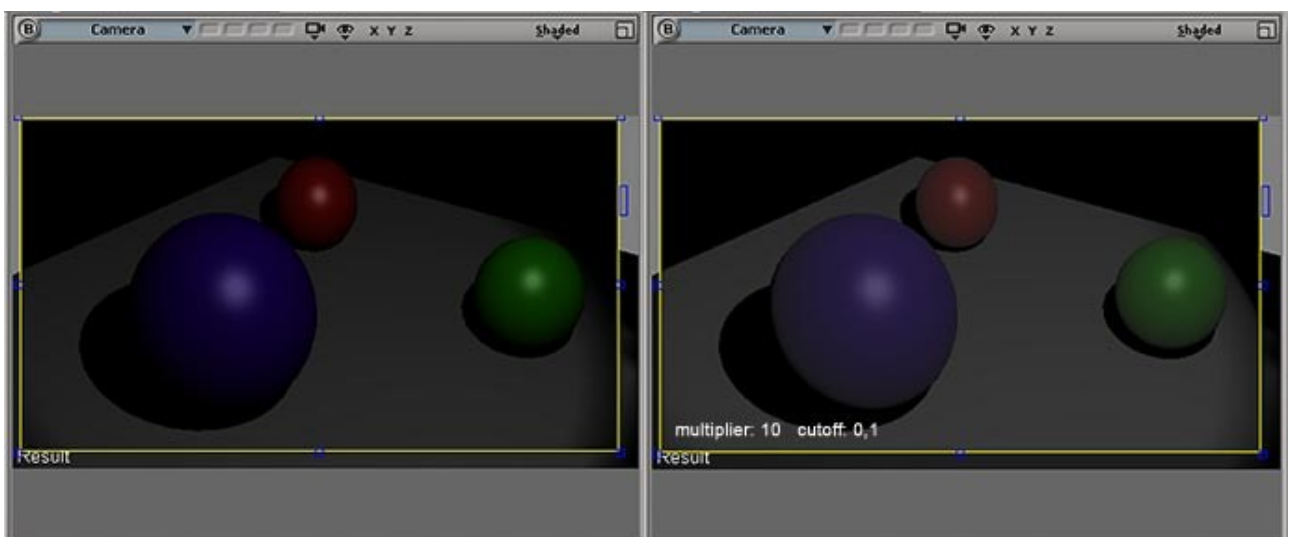
Lume_Night, toning down your colors:

From the Lume manual:

When designing a nighttime scene, it is essential that "Lume_Night" is activated before lighting is set. "Lume_Night" has a significant effect on dim lights: what looks good before activating "Lume_Night" will need to be adjusted after activating "Lume_Night". It should be noted that "Lume_Night" simulates the way human eyes work in dim lighting, and not the way a camera captures dimly lit images; do not use "Lume_Night" if the rendered image is intended to mimic a photographic look.

I opened up my *bumpcapture* tutorial scene for this, and adjusted my light color to a low 0.25,0.25,0.25. Now select your camera in the explorer, and open up the "Lens Shader" Tab.

Click the "Add" button and browse to the "Lume_Night" preset. Select it, and click "Inspect". We can adjust the "Multiplier" and "Cutoff" effect, the first will boost the lightlevel but desaturate shadows in the process, the latter will desaturate the the colors of your materials.



Lume_Wraparound:

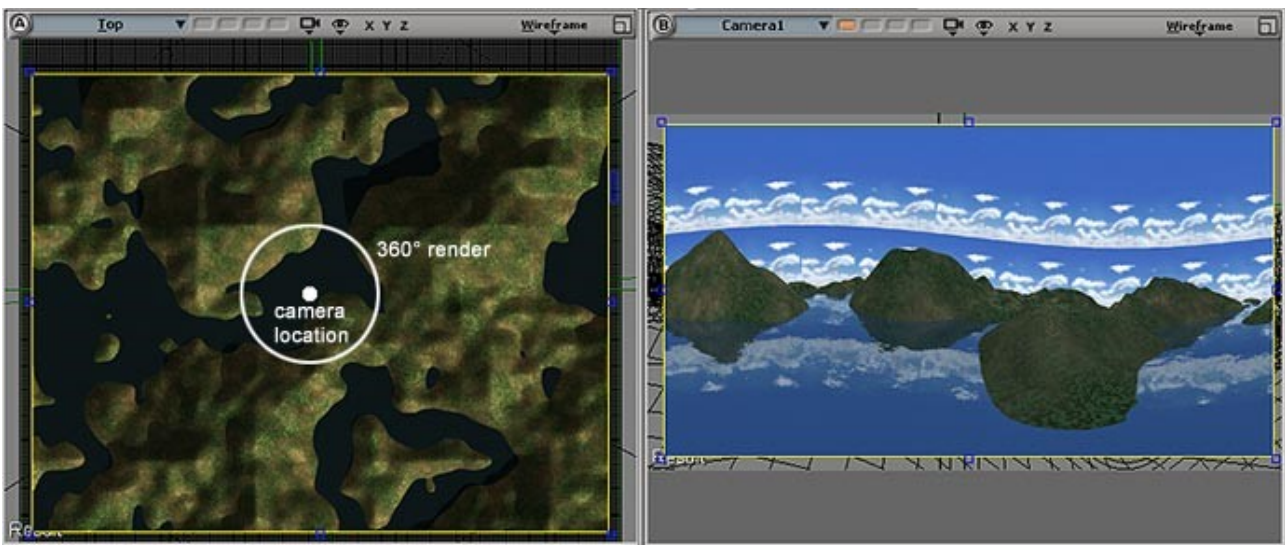
The "*Lume_Wraparound*" lens shader is one of the most basic ones. You cannot change any of the parameters, because there are none to adjust.

What it does is "*bend*" the rays so you get a - 360 degree - image of your scene. This a perfect shader for heavy scenery replacement or alike. Build your scene, place a camera in the middle of the scene, add the shader to the camera, render, hide original geometry, put rendered image on sphere, re-render, same result with a fraction of the render time.

Also a good shader to use for creating a QTVR movie.

So..... select your camera in the explorer, and open up the "*Lens Shader*" Tab. Click the "*Add*" button and browse to the "*Wraparound*" preset. Select it, you're done. Remember to render an image with the correct aspect ratio, so it will fit nicely on your sphere. So render a square image to wrap entirely on your sphere, or a 2:1 image to wrap on the top part of your sphere, like a dome.

Top view render left and rendered lens shader on the right, you can see the distortion of the bend rays:



Lume Volume shaders setup

Introduction:

Volume shaders are added at the Pass level. In this tutorial we discuss two volume shaders:

- Lume_Beam
- Lume_Mist
- Lume_Submerge, see "*Creating an ocean underwater scene with the Lume Toolset*" on page 6.

So here it goes, Lume_Beam:

The "Lume_Beam" shader adds glows to lights or geometry. It does not calculate shadows, so beware of that fact. This is the main reason it is so fast.

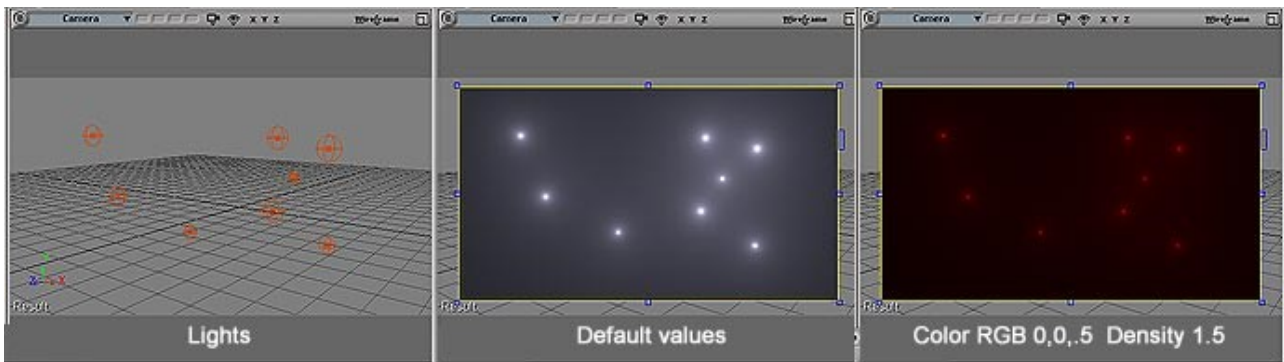
1. Adding the "Lume_Beam" shader to lights.:

Create a new scene and place a couple of point lights in front of the camera.

Go to the render panel and go "*Pass>Edit>Current Pass*" and open up the "*Volume Shader*" Tab. Add the "Lume_Beam" shader by clicking the "Add" button and browsing to the Lume shader preset. Select it in the "*Output Shader Stack*" and click on the "Inspect" button.

Select all lights by clicking the "*Pick Item*" button and selecting all lights in the drop down menu. Draw a render region. Adjust the color of the glow effect, or the density. Remember the effect is additive, so all lights will add to the effect. Also the color values work this way. It's easy to completely overblow the lights this way.

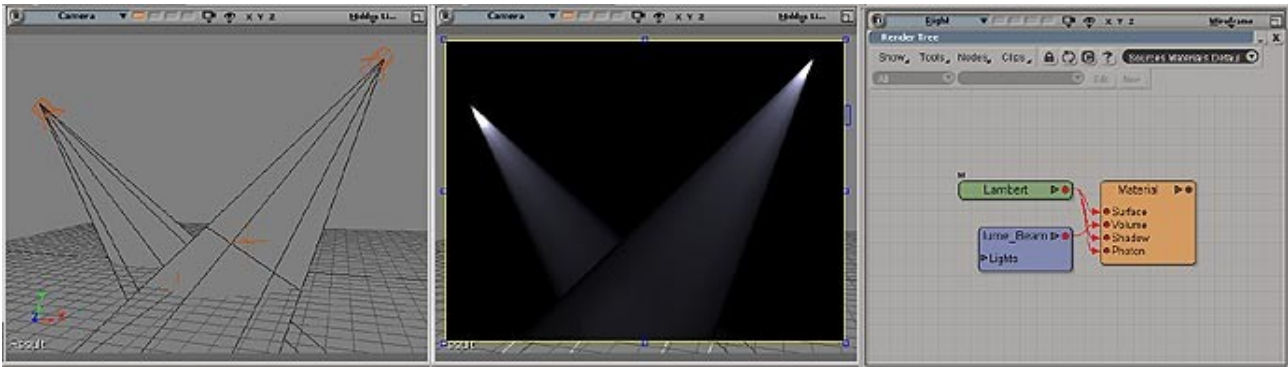
It's a nice shader for rendering lot's of point lights on a building or city view or something in a separate pass.



2. Adding the "Lume_Beam" shader to an object:

Open up a new scene and create two spot lights, and place them visible within the camera. Create two cones and place them in the same direction as the spotlights. Place the start of the cone inside the spot icon. Use the "B" shortcut on the spots to see the "*coneangle / spreadangle*". Add a material to the cones, open up the PPG and turn on the transparency to 0,0,0.

Now select one of the cones and open up a Rendertree. Add the "Lume_Beam" and connect it to the "volume" slot of the "Material" node. Do the same to the other cone. Draw a render region.



There are some drawbacks with this shader, you cannot have intersecting geometry as this will give artifacts and this shader doesn't do shadows.

For transparency of beams behind each other set the refraction values in the render setting higher than 4. So extra lights are necessary for shadows or multiple passes are needed when cones are intersecting.

Lume_Mist:

1. Create a scene:

Create a grid of 10x40 units, and a cylinder, radius 0,5 units and a height of 10 units. Move the cylinder on top and to the left side of the grid. Duplicate it to the right side. Select both cylinders and duplicate the set to 10 and 20 in Z, and -10 and -20 in Z. You now have a grid with 10 cylinders in 2 rows. Move the camera to 0,5,-30 and the interest to 0,5,0 so we're looking straight into the "corridor"

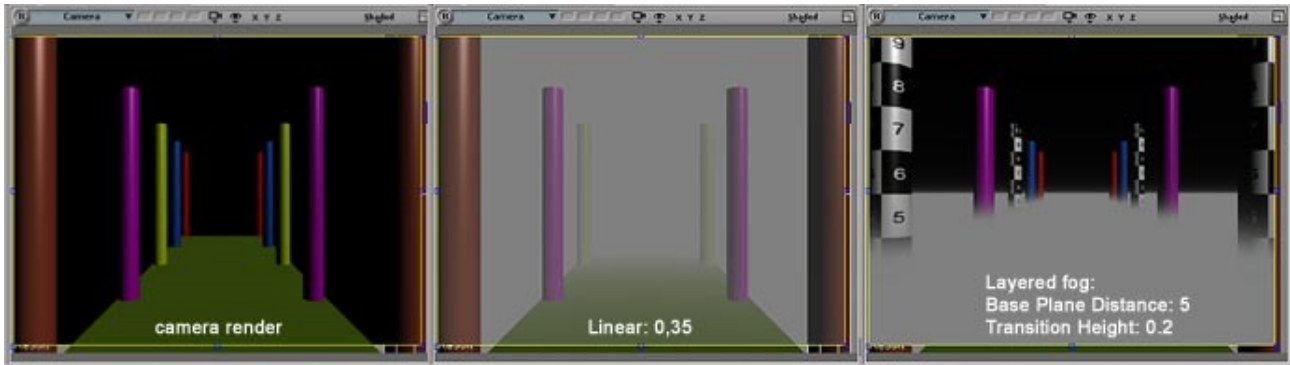
Go to the render panel and go *"Pass>Edit>Current Pass"* and open up the *"Volume Shader"* Tab. Add the *"Lume_Mist"* shader by clicking the *"Add"* button and browsing to the Lume shader preset. Select it in the *"Output Shader Stack"* and click on the *"Inspect"* button.

2. Setup:

Keep everything under the *"Overall"* Tab default and move on to the *"Layering and Falloff"* Tab. The default start with *"Realistic"*, play with the *"Density"* value to see the fog thicken. Also type in values higher than 1.

Switch to *"Linear"*, and set the values to 0 and 35 for *"Linear Start"* and *"Linear End"*. It may not be obvious at this point, but the fog is calculated from the camera viewpoint AND in XSI units, so keep this in mind with adjusting values. Also the *"Start"* value defines the transparency, the *"End"* value the opacity. With these values we can just see the cylinders in the middle show through the mist.

Switching to *"Custom"* will do the same, with an extra slider to place the 50% transparency exactly where you want it to be.



3. Change settings:

Switch back to "*Realistic*", be sure "*Density*" is set to 0,5 and turn on "*Layering*". To be able to see the layering in the height set the "Base Plane Normal" to 0,1,0 as in XYZ.

Set the "Base Plane Distance" to 5 and the "Transition Height" to 0.01. You will see that the mist will stop at 5 XSI units from the ground plane at 0. The Transition height is – **multiplied**- with the "Base Plane Distance", so changing the "Transition Height" to 0.2 will result in a transition of 1 units, so the mist will start around 4 and end at 5.

Smaller values will result in a sharper transition. Don't make it too small or the effect will be strange, especially when the camera is – **in** - the Mist.

Final Notes:

Keep in mind that all shader values as stated in the tutorial are not "set", and a lot of different results can be achieved by just changing some values.

Happy rendering with the Lume Shaders!!