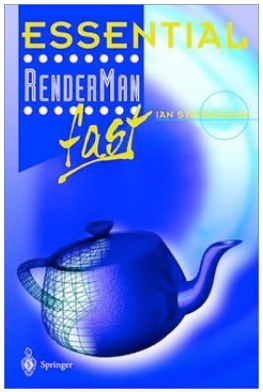
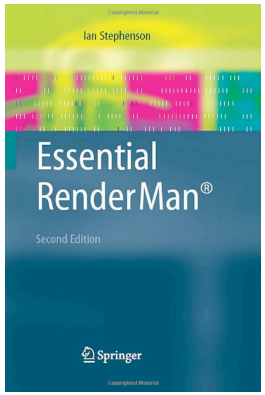
Shaders are code that describes a surface to a renderer. Shaders can represent any type of surface from metals to plastics, from skin to ceramics. Traditionally, shaders are text documents which are compiled into shader objects. These shader objects are then installed into a 3D system and assigned to geometry. Increasingly however, shaders can now be internally generated within 3D systems using a node based UI.

Within Houdini, it is possible to generate and install code based shaders for Mantra, Renderman and Mental Ray; and build node based shaders for Mantra and Renderman. Building shaders for Mantra and Renderman is fairly indistinguishable in terms of structure and configuration, either building shaders by code or by node network. Therefore learning to write shaders for Mantra will also teach you the principles for writing Renderman Shaders.

An excellent resource when learning to write shaders is Ian Stephenson's Essential Renderman Fast by Springer Press. This book contains lots of useful basic code for generating patterns. This code can be translated into node based networks and again forms a useful way to learn shader writing.



First Edition Cover



Second Edition Cover

EXAMPLE OF CODE BASED SHADER

Here is an example of a code based shader for Mantra. Mantra Shaders are written in VEX code and stored in a .vfl file. This code creates a simple electron microscope / x-ray shader.

FILE NAME: electron_xray.vfl

FILE CONTENTS:

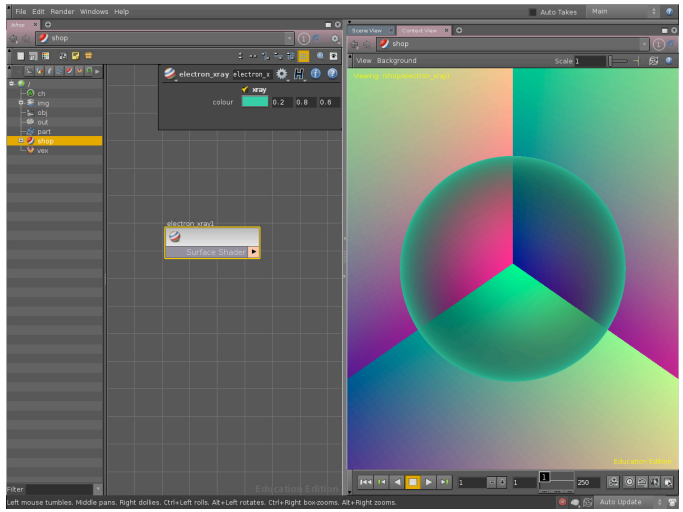
```
#pragma hint colour color
#pragma hint xray toggle

surface electron_xray (int xray = 0; vector colour = {0.2,0.8,0.6};)

{

float edge = 1-abs(dot(normalize(N),normalize(I)));

if (xray == 0)
        {
        Of = 1;
        }
else
        {
        Of = 1 * edge;
        }

Cf = colour * edge;


}
```

This VEX code can be compiled with the following command:

vcc -l electron_xray.otl electron_xray.vfl

This will compile the .vfl file into a .vex file and place it inside an otl. This otl can then be loaded into Houdini, where the variables xray and colour become parameters that can be interacted with by an end user.

NOTE: variables declared within the {} brackets of the main shader code (such as the edge variable) will not be visible to an end user.



THE RENDERMAN EQUIVALENT

FILE NAME: electron_xray.sl
FILE CONTENTS:

```
surface electron_xray( float xray = 0; color colour = (0.2,0.8,0.6); )

{

float edge = 1-abs(normalize(N).normalize(I));

if(xray == 0)
        {
        Oi = 1;
        }
        else
        {
        Oi = 1 * edge;
        }

Ci = colour * edge;

}
```

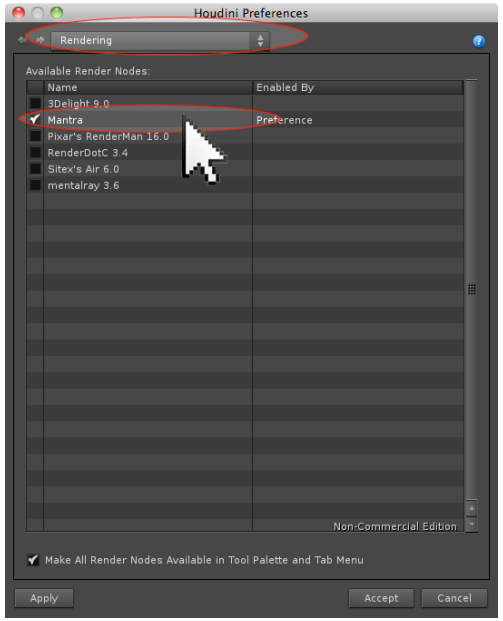
This .sl file can be compiled using the Renderman textport command:

shader electron_xray.sl

This will create a shader object file called electron_xray.slo. This can be converted into a Houdini Digital Asset using the command:

rmands –l electron_xray.otl electron_xray.slo

NOTE: Renderman and/or other renderers can be activated in Houdini by going to the main **Edit** menu and choosing **Preferences > Rendering**.

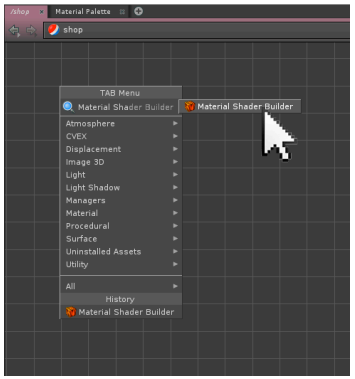


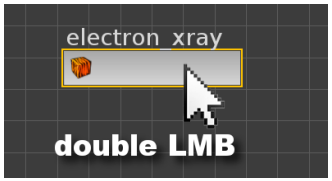
If Renderman is also activated, specific Renderman operators will appear both at SHOP Level and at OUTPUTS Level.

Shader Writing is very similar in many ways to compositing. Surface Colours of varying types are created within a shader and then blended between using black and white patterns. If you understand the principles of compositing, you should easily pick up the principles of shader writing.

RECREATING THE ELECTRON XRAY SHADER AS A NODE NETWORK

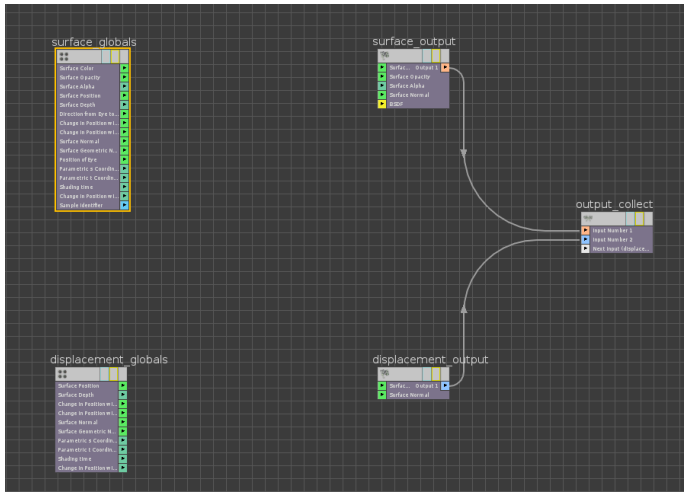
In Houdini switch to **SHOP Level** and create a **Material Shader Builder**. This will create a Material node that can be internally edited to create a custom shader.





Rename this **Material** to **electron_xray**, and **double LMB** click on it to go inside it. Inside the Material node is a **custom VOP Level** containing five operators: two Global Variables VOPs (one for Surface, the other for Displacement), two Output VOPs (again one for Surface and one for Displacement), and an Output Collect VOP (which combines both Surface and Displacement shader networks together).

The Global Variables VOP can be thought of in the same way as a Read Node in Nuke. These operators import all the standard attributes found on geometry into the shader network.

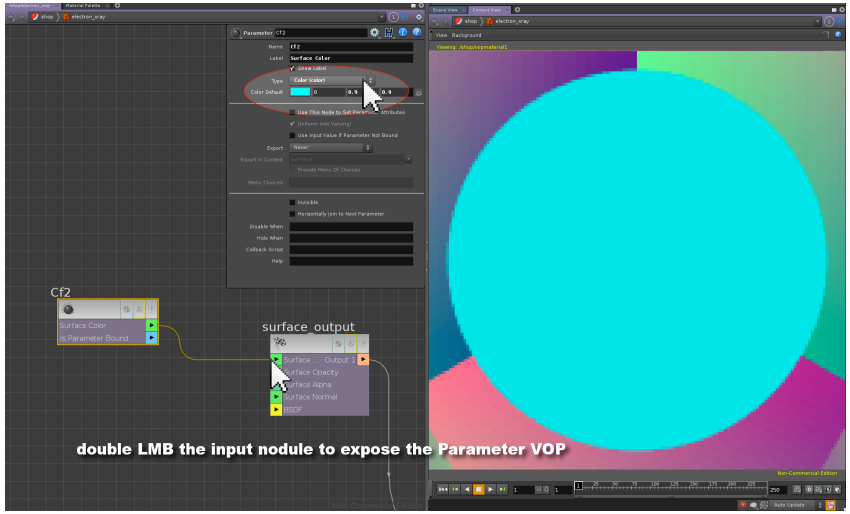


A custom Shader Network can be generated to link these inputs and outputs together using other VOP operators to describe the surface aesthetic.

NOTE: By default, a Material Shader Builder does not contain the node setup required for evaluating scene lighting. Such Lighting Model nodes have to be hand configured when using this operator. If lighting calculations are also required, please use a **Surface Model Material** found in the Material Palette, as this has these nodes preconfigured. As both the electron and xray shaders are not reliant upon scene lighting, the Material Shader Builder is fine for configuring these shaders.

CREATING THE BASE COLOUR

MMB on the **Surface Color (Cf) input** of the **Surface Output VOP** and from the resulting menu select **Promote Parameter**. This will automatically create a hidden Parameter VOP as a new node and wire it into the Surface Color (Cf) input. **Double LMB** on the resulting **input nodule** to expose the **Parameter VOP**.



In the **parameters** for the **Cf2 Parameter VOP** specify:

Parameter Type	**Color (color)**		
Color Default	**0**	**0.9**	**0.9**

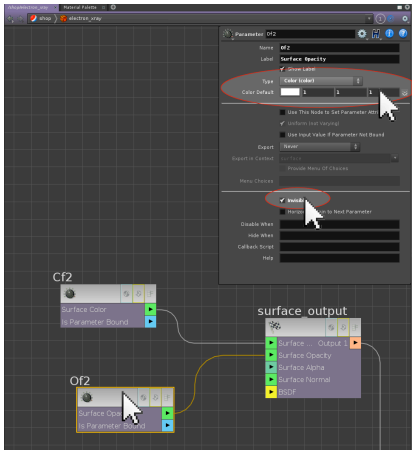
This will create a constant cyan base colour for the electron xray shader, which is visually returned in the VOP Viewer. Any Parameter VOP created will also automatically appear at SHOP/Material Level (unless activated as invisible). The Cf2 Parameter VOP is now explicitly declared as a colour parameter that an end user can interact with.

This node configuration equates to the following lines of VEX code from the electron xray shader script:

#pragma hint colour color

surface electron_xray (int xray = 0; **vector colour = {0.2,0.8,0.6}**;)

Brackets containing all of the end user parameters found at the Material Level normally follow the surface declaration of a shader script. Any operations created in the main script itself (ie outside of these brackets) are not visible to the viewer.



Repeat this operation for the **Surface Opacity (Of) input** of the **Surface Output VOP**. When a Parameter VOP is initially created for the shader, it will return a value of zero. This can be corrected by exposing the Parameter VOP node, and specifying in its **parameters**:

Parameter Type	**Color (color)**		
☑	**Invisible**		
Color Default	**1**	**1**	**1**

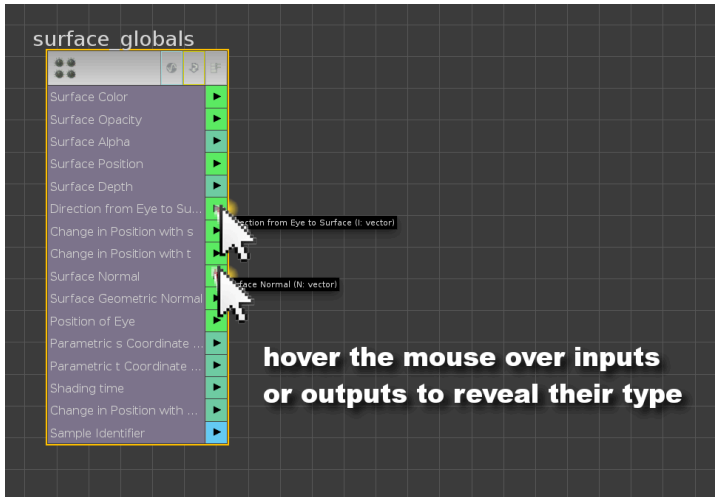
NOTE: Activating the Invisible option on a Parameter VOP will make the parameter invisible to the end user. This can be verified by returning temporarily back to the SHOP Level where only the Surface Colour parameter can be seen and modified.

CREATING THE ELECTRON EDGE

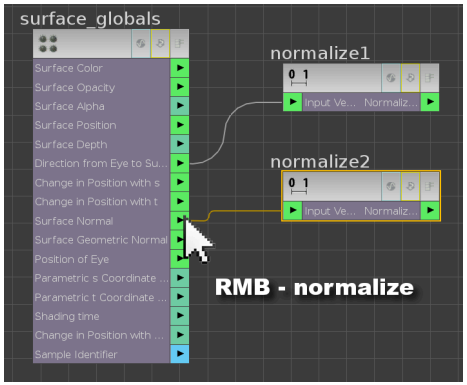
In the VEX shader code, the calculation for creating the electron edge is as follows:

```
float edge = 1-abs(dot(normalize(N),normalize(I)));
```

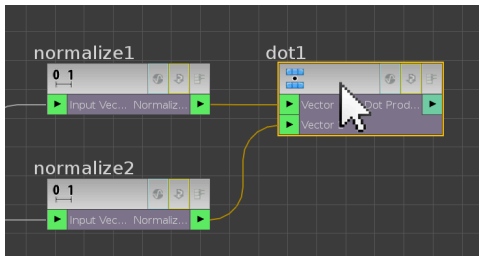
This calculates the value of the **dot product** between the **surface Normal (N)** and the camera's directional position to the **surface (I)**. **N** and **I** are both **Global Variables** and can therefore be identified on the **Surface Global Variables VOP**.



Hovering the mouse over any of the outputs or inputs of a VOP operator will reveal their type and any internal naming taking place. **RMB** on both the **I** and **N** outputs and in both cases append to them a **Normalise VOP**.

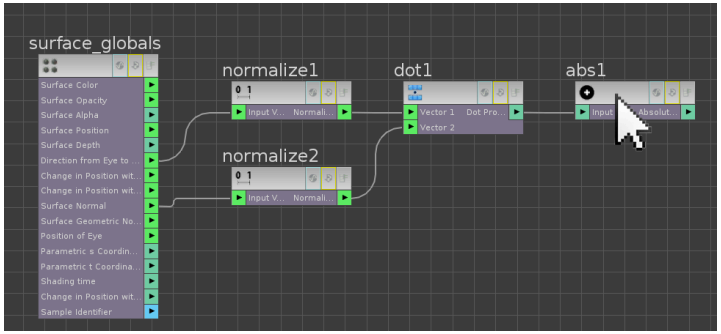


This is the equivalent of the **normalize(N)** and **normalize(I)** parts of the VEX Code from the edge calculation. Doing this will ensure any incoming vector data from the geometry will be normalised (re-ranged between 0 and 1). Normalised vector values help ensure correct shader calculations.



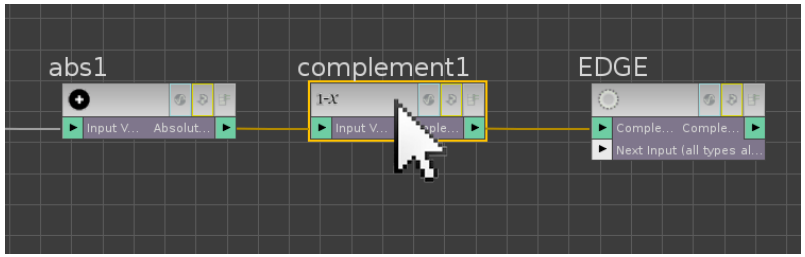
Create a **Dot Product VOP** as a new node, and wire the outputs of both Normalise VOPs into it. The Dot Product VOP will return a floating point number for each point on the surface. The value of this floating point number is the determined by the angle between N and I for each point on the surface. This node configuration is the equivalent of the **dot(normalize(N),normalize(I))** VEX Code.

RMB append to the **Dot Product VOP** an **Absolute VOP**. This will ensure all of the resulting numeric values are positive.



The VEX Code **abs(dot(normalize(N),normalize(I)))** has now been recreated as a node network.

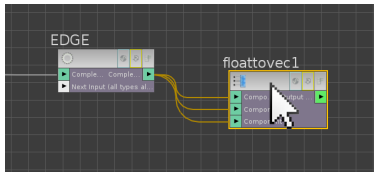
The **1-** abs(dot(normalize(N),normalize(I))) section of the VEX Code can be recreated using a **Compliment VOP**. This will invert the values calculated by the Absolute VOP.



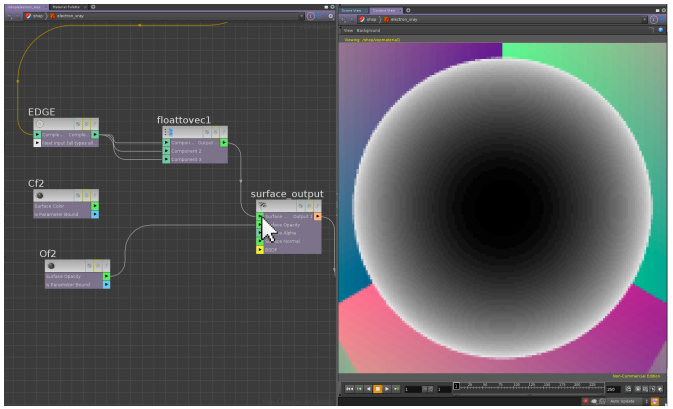
As a final step append a **Null VOP** to the **Compliment VOP**, renaming it to EDGE. The VEX Code **float edge = 1-abs(dot(normalize(N),normalize(I)));** has now been configured. **See file electron_xray_material_stage1.hipnc**

TESTING THE EDGE CALCULATION

Currently the EDGE calculation network has not been integrated into the surface output, and therefore its results cannot be seen.

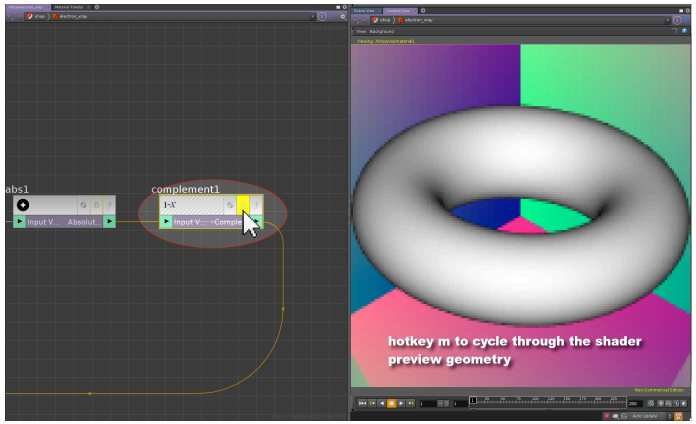


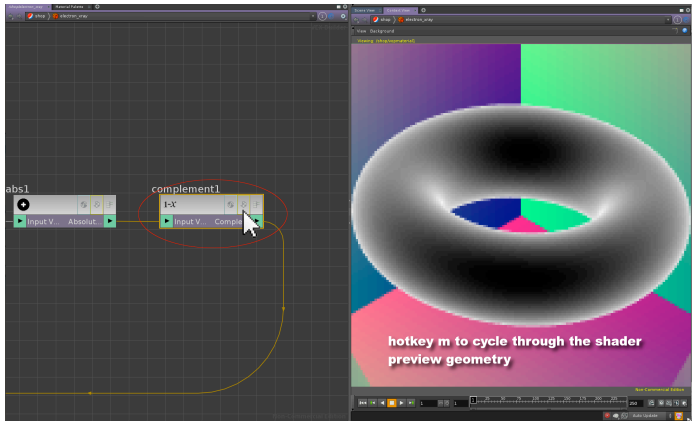
In order for this calculation to be visualised, the **output** of the **EDGE Null VOP** can be temporarily wired into a **Float to Vector VOP**. This will convert it into the correct data type for being wired into the **Cf input** of the **Surface Output VOP**.

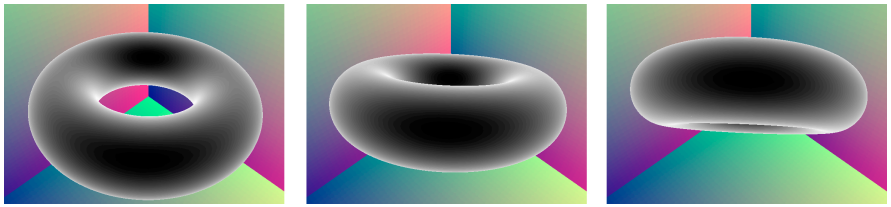


When the **Float to Vector VOP** Is wired into the **first input** of the **Surface Output VOP** the EDGE calculation can be seen. In general terms, this calculation will make the edges of the geometry white, and the geometry interior black relative to the direction of the viewing camera.

If the **Compliment VOP** is temporally **bypassed**, the **EDGE** calculation becomes the equivalent of the '**Head Light**' used by Mantra when rendering scenes with no lighting.







When the preview geometry is tumbled in the Viewer, the white areas will always appear on the edges of the geometry relative to the camera (if the Compliment VOP is bypassed, the white areas will always face towards the camera).

VOPS AND DATA TYPES

In order for nodes to be correctly wired together, VOPs utilises a simple colour scheme to denote data types. This colour scheme varies slightly for Renderman Material VOP Networks; however for Mantra Material Shaders the schema is as follows:

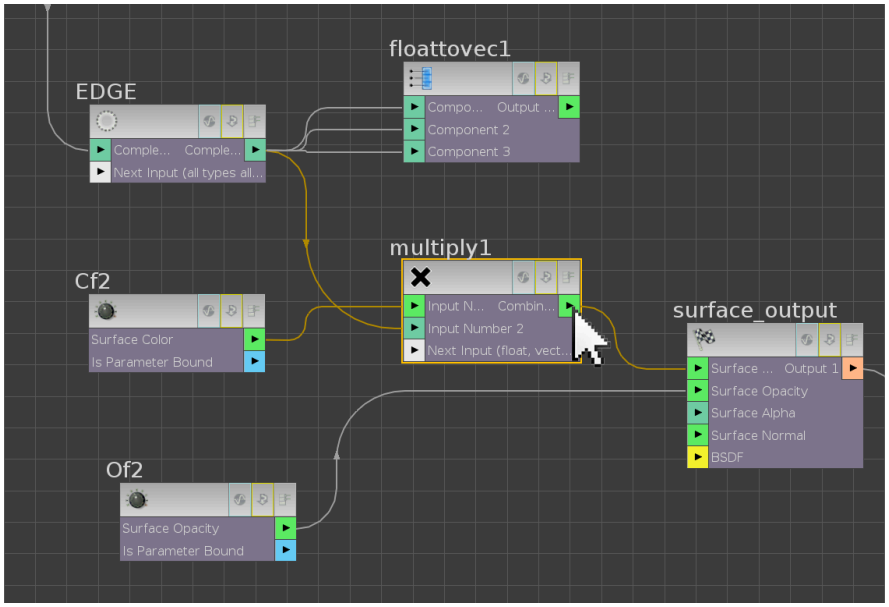
Bright Green	**Vector**
Light Green	**Vector4**
Pale Green	**Float**
Blue	**Integer**
Purple	**Matrix3**
Light Purple	**Matrix4**
Peach	**String / Text Input**

In this example, the **float output** of the **Absolute VOP** is wired into a **Float to Vector VOP** to convert its data type into a **vector** for visual output. **Other** useful **VOPs for converting data types** can be found under the **Convert** section of the **VOP Level TAB Menu System**.

CONSOLIDATING THE MATERIAL

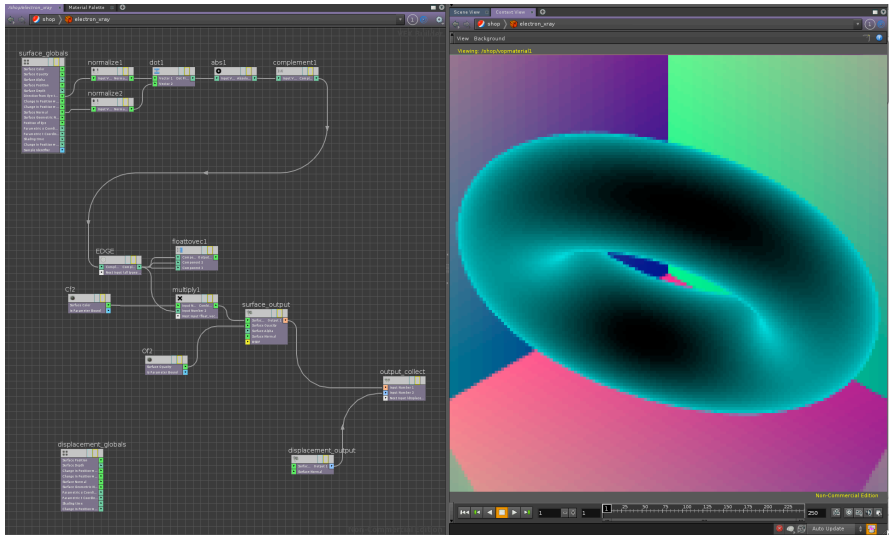
The **original colour Parameter VOP** can now be **multiplied** with the **EDGE calculation** in order to start **consolidating the material** into an electron microscope aesthetic.

To the **output** of the **Surface_Colour Parameter VOP** append a **Multiply VOP**. Wire the **output** of the **EDGE Null VOP** as the **second input** to the **Multiply VOP**.



NOTE: The Float to Vector VOP could also be utilised as the multiplier for the colour; or as in this example it can be ignored and deleted when no longer required.

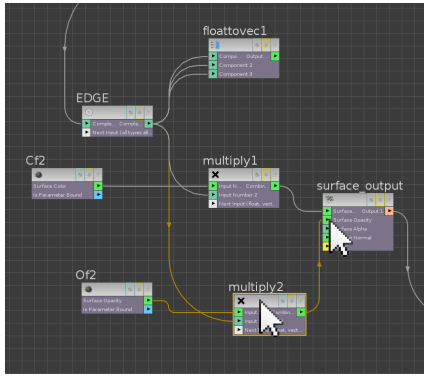
When the **output** of the **Multiply VOP** is fed into the **Surface Color input** of the **Surface Output VOP**, the effect of the shader network can be seen.

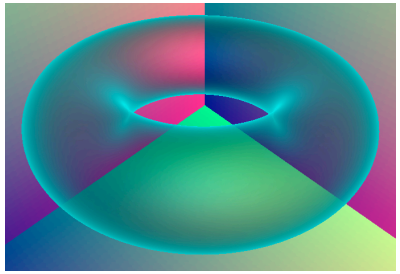


See file electron_xray_material_stage2.hipnc

CREATING THE XRAY SHADER

At present the shader behaves similar to the aesthetic generated by an electron microscope. A variation upon this is to use the edge calculation to also drive the opacity of the shader. If the **Surface Opacity VOP** is also multiplied by the output of the **EDGE Null VOP**, an **X-Ray effect** can be achieved.





NOTE: The Background menu of the VOP Viewer can also be used to set a background for the Viewer preview geometry.

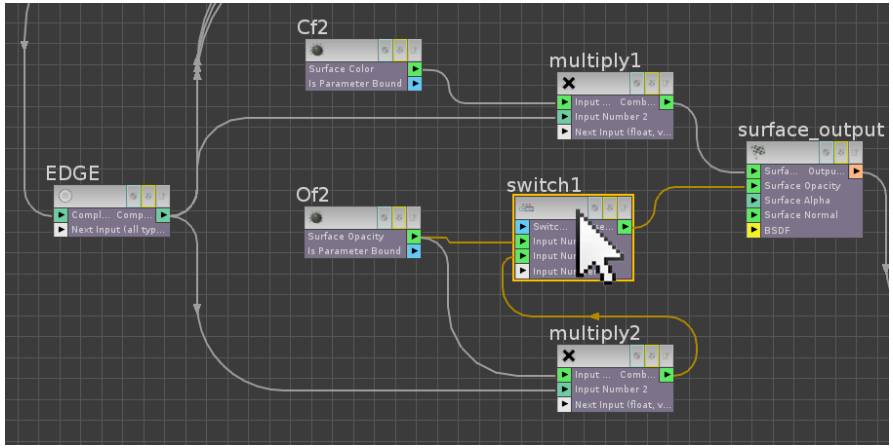
CREATING AN INTERACTIVE SWITCH

In order for an end user of the Material to switch between both electron and x-ray views, a **Swtich VOP** can be utilised. This is the equivalent of the VEX code based if-statement:

```
if (xray == 0)
        {
        Of = 1;
        }
else
        {
        Of = 1 * edge;
        }
```

THE SWITCH VOP

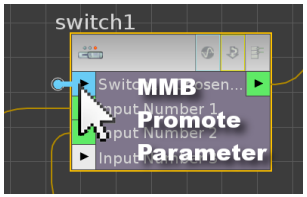
As a separate node, create a **Switch VOP**. **NOTE:** it's very first input is a blue integer input for creating an interactive switch for the end user.



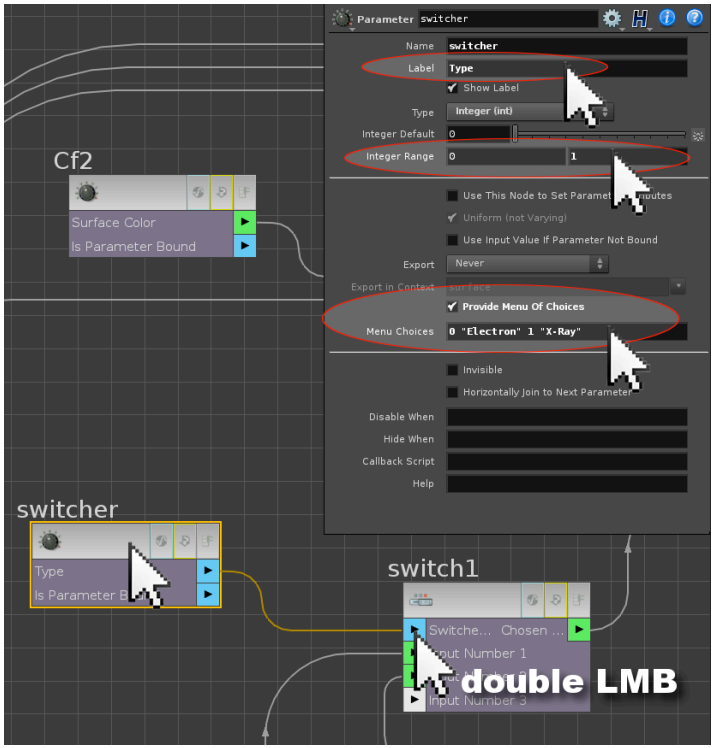
Wire the **Surface Opacity Parameter VOP** as the **second input** to the **Switch VOP** (**input1**). Wire the output of the **Multiply VOP** affecting the opacity as the **third input** to the **Switch VOP** (**input2**).

As a final step, wire the **output** of the **Switch VOP** as the **Surface Opacity** (**Of**) **input** of the **Surface Output VOP**. This will create two scenarios for the Material, where either the opacity is left as white (the electron effect) or the opacity is multiplied by the edge calculation (the x-ray effect).

In order for an end user to be able to specify which effect is required, a switch mechanism must be created. This can be done by **MMB** on **blue integer input** of the **Switch VOP** and choosing **Promote Parameter**. This will automatically create an integer Parameter VOP as a node nodule.



Double LMB on the resulting nodule to expose the **Switcher Parameter VOP**.



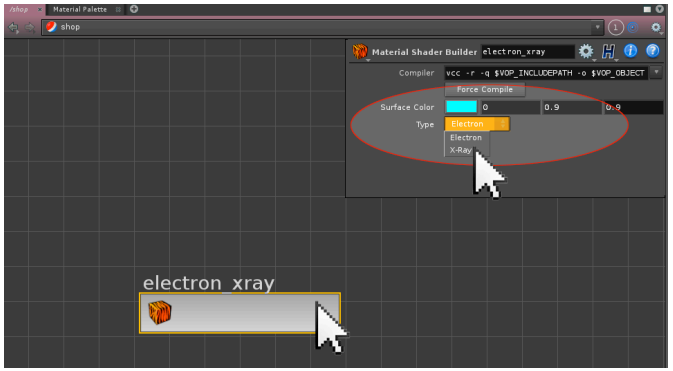
In the **parameters** for the **Switcher Parameter VOP** specify:

Parameter Label	**Type**
Integer Range	**0 1**
☑	**Provide Menu of Choices**
Menu Choices	**0 “Electron” 1 “X-Ray”**

This will create a simple menu for an end user to specify either an electron or x-ray shader.

TESTING THE ELECTRON XRAY MATERIAL

Press u on the keyboard to return back up to SHOP Level. Here test that the switch is working as expected.



See file electron_xray_material_stage3.hipnc

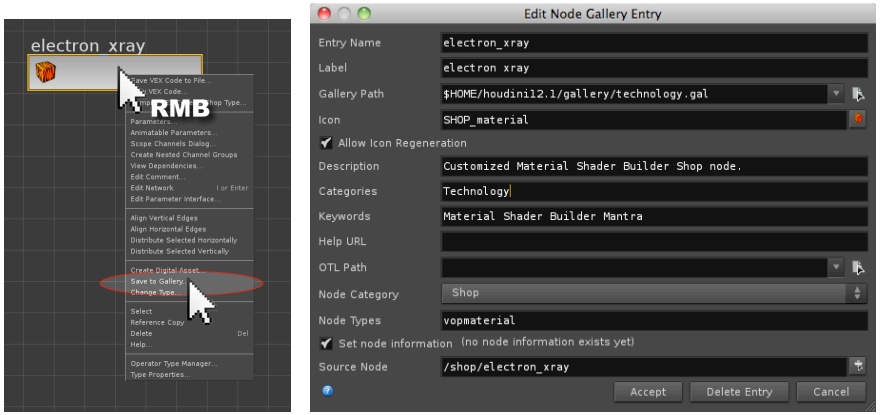
THE MATERIAL PALETTE, GALLERY FILES AND DIGITAL ASSETS

In Houdini it is possible to export materials to the Material Palette by storing them in a new Gallery. A Gallery file behaves in a similar way to .otl files where multiple custom Materials can be added to a single gallery much like multiple digital assets can be stored in a single .otl file. Custom Materials can also be exported as Digital Assets for ease of sharing.

If a custom material is stored directly into a Gallery file (with no conversion to a Digital Asset), the code for the shader will be embedded into the Gallery file. If a Digital Asset based custom Material is stored to a Gallery file, the Gallery file will reference the .otl file location where the Material is stored. This means if the current hip file doesn't have the correct .otl file loaded, any material created which references it, will return an error.

ADDING A CUSTOM MATERIAL TO THE MATERIAL PALETTE

RMB on the **electron_xray Material Node** and from the resulting menu choose **Save to Gallery…**

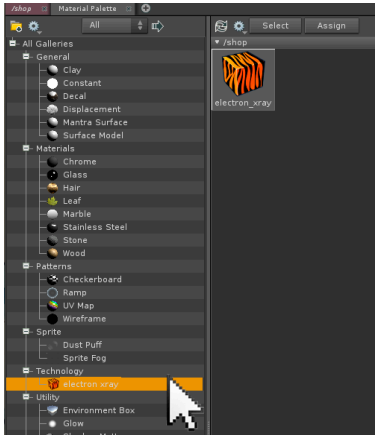


This will activate the **Edit Node Gallery** Entry window. In the **parameters** for this window specify:

Gallery Path	**$HOME/houdini12.1/gallery/technology.gal**
Catergories	**Technology**

NOTE: if your **$HOME/houdini12.1 preference folder** does not contain a **gallery directory**, please create one when specifying the **.gal** output location.

Press the **Accept** button to confirm this gallery. This will create a .gal file in the $HOME/houdini12.1/gallery preferences folder, storing the shader code embedded inside it. As this .gal file is stored in the Houdini preferences folder, it will automatically be accessible with every new Houdini scene.

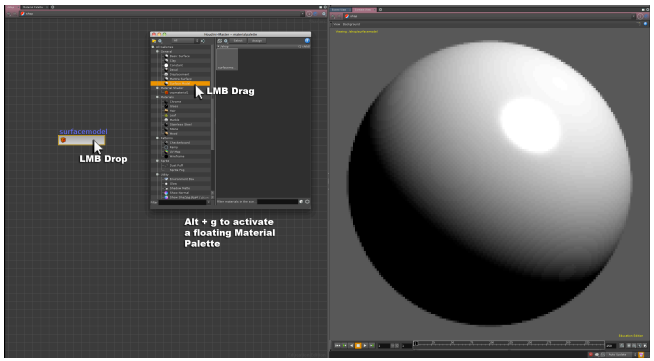


The Electron X-Ray Material will now be accessible through the material gallery within its own gallery section called Technology.

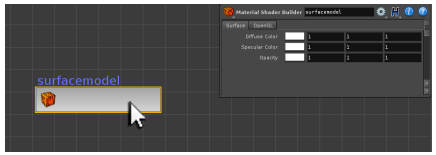
NOTE: If a custom gallery file is not automatically visible in the Material Palette, the gallery file can be manually added to the Material Palette by using the Folder icon at the top of the Material list.

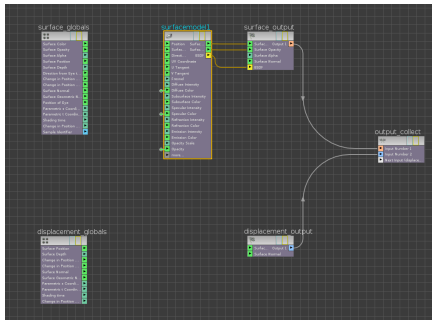
CREATING HIGHER LEVEL CUSTOM MATERIALS

In a new Houdini scene, go to the **SHOP Level** and activate a **floating Material Palette** (**Alt + g**). **LMB Drag** and **Drop** a **Surface Model Material** from the Palette **into the SHOP Network Editor**.



When the **parameters** for the **Surface Model Material** are examined, it appears to be a redundant operator, as only a few parameters exist.



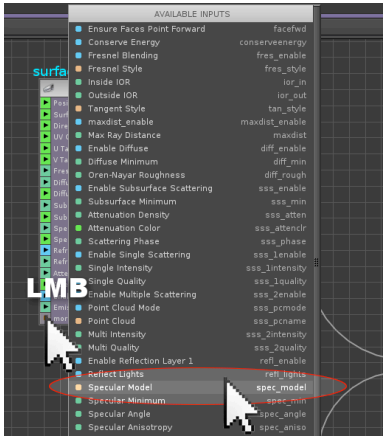


This operator is however similar to the Material Shader Builder; however already has a preconfigured Surface Model VOP inside it. The three Material Level parameters also appear as parameter nodules on the Surface Model VOP. The visual result of this network is a basic Material that will respond to scene lighting information.

The Surface Model VOP contains all of the parameters and shader functionality found in a standard Mantra Surface Material; however in the case of the Surface Model Material, these parameters are not yet activated. The idea with the Surface Model Material is that a custom shader network can be developed, and its core parameters plus any desired parameters of the Surface Model VOP can then be activated to an end user.

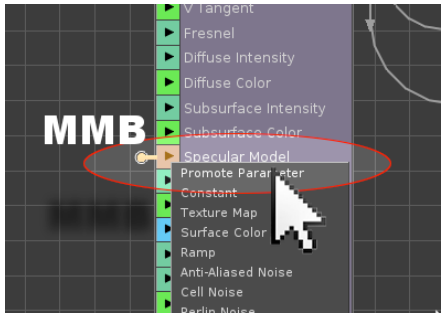
CREATING A LIGHTING MODEL END USER CONTROL

As an end user of this Material may require choice over the type of specular lighting model assigned to the surface, this option can be easily activated from the Surface Model VOP.

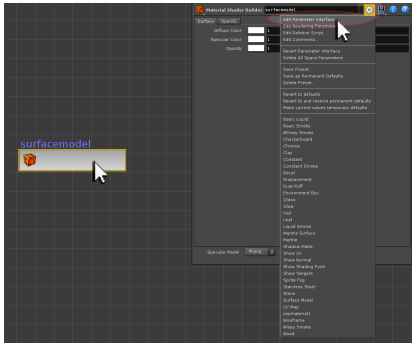


LMB on the **more… input** of the **Surface Model VOP** to reveal the other **Available Inputs**. From the resulting list, select the **Specular Model** input.

The Specular Model input will now appear on the inputs list of the Surface Model VOP.



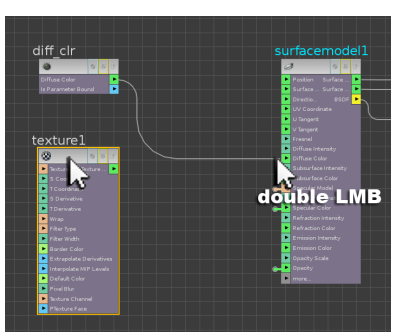
MMB on the **Specular Model input** and from the resulting options choose **Promote Parameter**. This will create a hidden Parameter VOP nodule as well as porting the parameter up to material level.

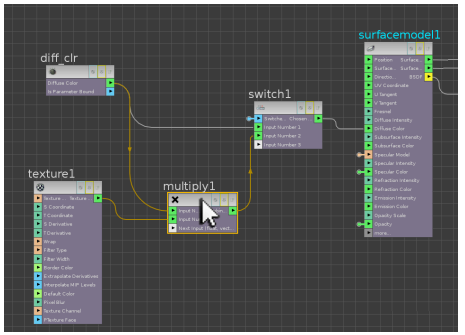


Return back to **SHOP Level** and examine the **parameters** for the **Surface Model Material**. By default newly ported parameters appear right at the **bottom of the parameter list**. This can be rectified by using the Parameter Pane **Cog Button** to **Edit the Parameter Interface**. Other parameter inputs found on the Surface Model VOP can also be activated in this way.

CREATING A DIFFUSE TEXTURE INPUT

Double LMB on the **Diffuse Color input nodule** of the **Surface Model VOP** to expose the associated Parameter VOP. Alongside it also create a **Texture VOP**.

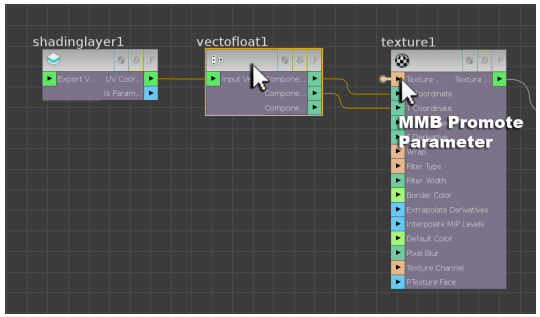




The **output** of the **diff_clr Parameter VOP** can be **multiplied** with the **output** of the **Texture VOP**; and **both** the **outputs** of the **diff_clr Parameter VOP** and the **Multiply VOP** can then be fed into a **Switch VOP** as inputs. An **end user control** can also be activated on the **Switch VOP** to allow for switching between a colour and a texture.

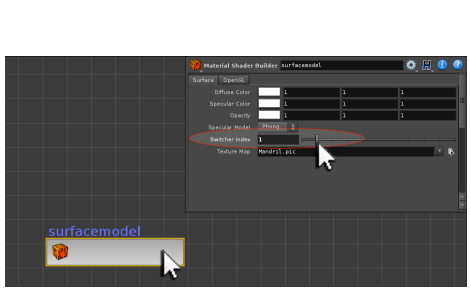
As UV information is not part of the Surface Global Variables VOP (or indeed assigned as standard to geometry), any UV information assigned to geometry needs to be explicitly read into VOPs to make it accessible. This can be done by using a **Shading Layer Parameter VOP**.

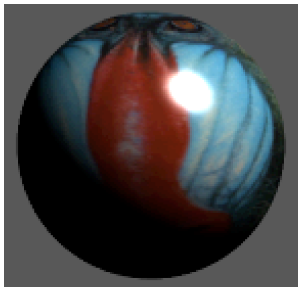
Alongside the **Texture VOP**, create a **Shading Layer Parameter VOP**. Use a **Vector to Float VOP** to access the **UV** vector **components**, wiring its outputs into the respective **S Coordinate** and **T Coordinate inputs** of the **Texture VOP** (**u** should go into the **S Coordinate Input** and **v** should go into the **T Coordinate Input**).



The **Texture Map input** of the **Texture VOP** can also be assigned as an **end user control** by **MMB** on the input to **promote the parameter**.

When the Material is examined at SHOP Level, the end user can determine whether the base colour or a texture map can be activated. With a Switcher Index value of 1, the display geometry is assigned the Houdini Mandril image.



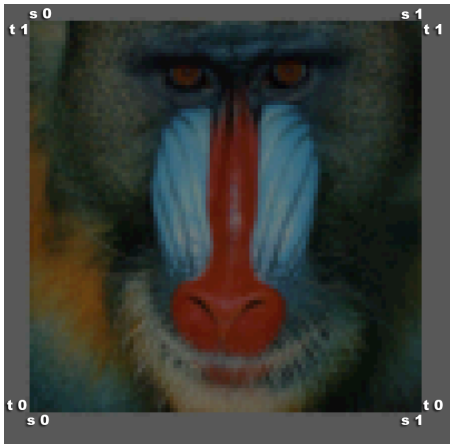


See file surface_model_stage1.hipnc

GLOBAL S & T VS UVS

The placement of textures on geometry can either be controlled by **UV information** derived from the **Shading Layer Parameter VOP** or **S & T information** derived from the **Surface Global Variables VOP**. The Texture VOP indicates that it will accept S and T information, which visually links directly to the S and T outputs of the Global Variables VOP.
Both **S & T** and **U & V** relate to **X & Y** where **S, U and X** return horizontal information, and **T, V, and Y** all return vertical information.

S & T information returns **texture coordinates** of an image or pattern stored within the standard UV square. This also relates to the Viewer geometry preview too. When the Viewer Geometry Preview is set to a grid, the S and T coordinates can be determined.



The S & T coordinate system allows for patterns to be constructed relative to this 0-1 square. For example, a checker pattern might have 8 stripes set relative to S & T. U and V information however returns surface coordinates of any geometry UV Projected into the standard UV square. As this texture shader will ultimately be positioned through some form of UV Projection operation, it is appropriate to use UV information to drive the placement.

NOTE: The **Shading Layer Parameter VOP** will **import UV**, **Colour** or **Alpha attributes** found on the incoming geometry. A **standard Parameter VOP** can also be configured to import these attributes or other custom attributes, as long as its Name parameter and Type parameter matches the name and type of the attribute being imported.

PTEX AND MATERIALS

As well as the **Texture VOP**, there is also a **PTexture VOP**. This operator can be used to read in ptex image files.

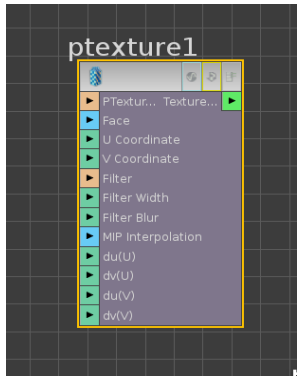


IMAGE PLANE EXPORTS

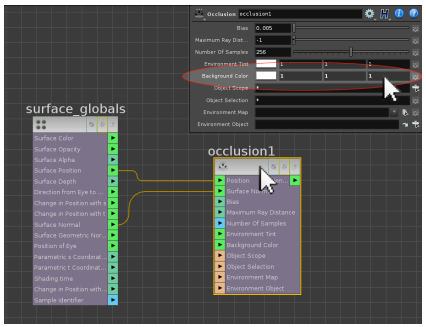
The Surface Model VOP has pre-configured capacity to export all of the standard image planes declared on a Mantra ROP. Other custom image plane exports can also be configured, either derived from the main shader network or from separate networks created inside the material. One example is to configure Ambient Occlusion into the shader network, and have it export as an image plane.

Locate the **Surface Global Variables VOP** and next to it create an **Occlusion VOP**. Wire the **Surface Position output** from the **Surface Global Variables VOP** into the **Position input** of the **Occlusion VOP**. Wire the **Surface Normal ouput** from the **Surface Global Variables VOP** into the **Surface Normal input** of the **Occlusion VOP**.

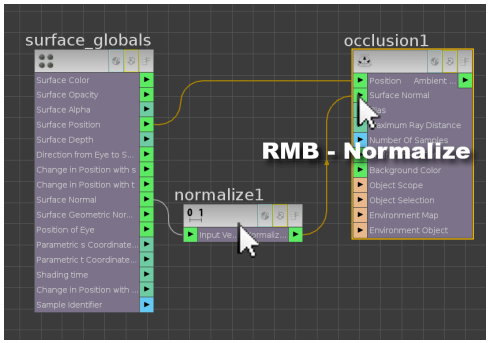
In the **parameters** for the **Occlusion VOP** specify:

Background Color	**1**	**1**	**1**

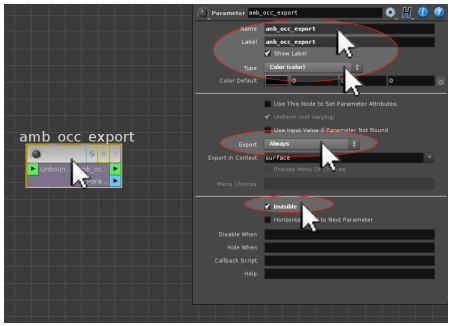
The other parameters of the Occlusion VOP can be left at their defaults, and if desired parameter inputs for the Occlusion VOP can be promoted to Material Level as end user controls.



To ensure correct normals calculations, a Normalize VOP must also be inserted before the Surface Normal input of the Occlusion VOP. **RMB** on the **Surface Normal input** and from the resulting **TAB Menu system** type normalize to activate a Normalize VOP.



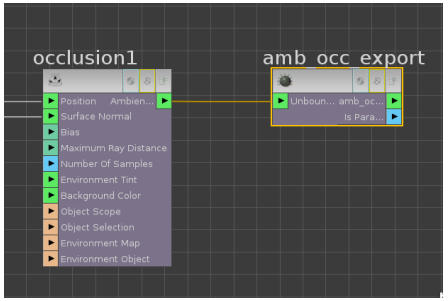
In a new part of the Network Editor, create a **Parameter VOP**. Set the **Name**, **Label** and **node name** as **amb_occ_export**.



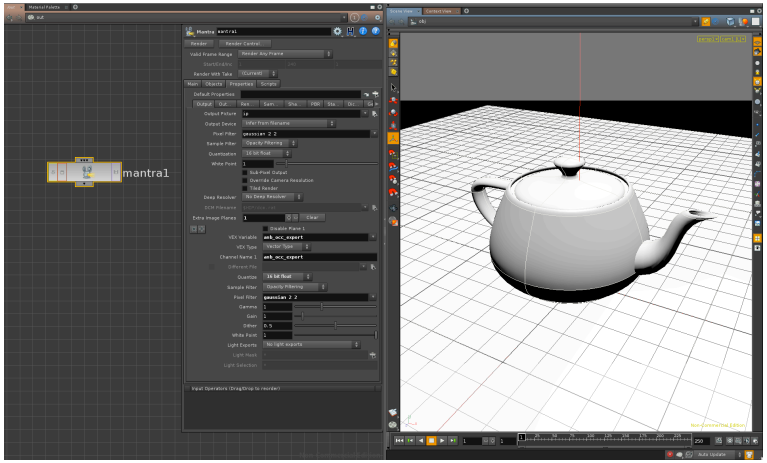
In the **parameters** for the **amb_occ_export Parameter VOP** specify:

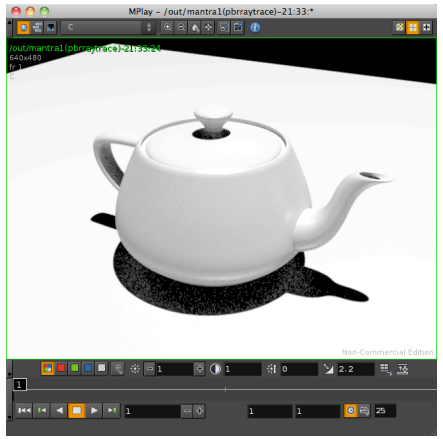
Type	**Color (color)**
Export	**Always**
☑	**Invisible**

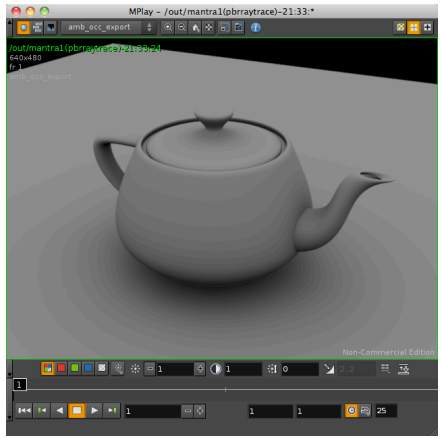
The **output** of the **Occlusion VOP** can now be wired into **input** of the **amb_occ_export Parameter VOP**.



A simple scene can be configured to test the custom material and the amb_occ_export image plane. A **Sky Light** can be also used for lighting the main scene render. See file **surface_model_stage2.hipnc**

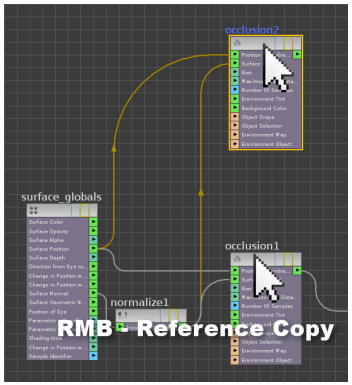






REFLECTION OCCLUSION

A variation on **Ambient Occlusion** is **Reflection Occlusion** (Twist, S; NCCA – 2009). This is where a Fresnel (or edge) calculation is made to drive the bias the surface normals of the occlusion calculation. This render pass is especially useful for surfaces such as metals or fluids, where additional shading is required to help retain the surface shape of highly reflective materials.

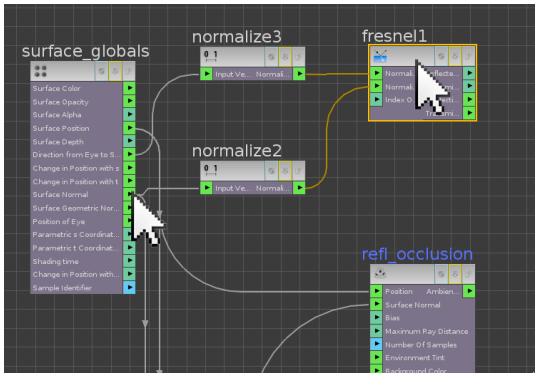


Back inside the shader network, **RMB** on the **Occlusion VOP** to create a **Reference Copy** of it. This means that the primary parameter configuration of the [Ambient] Occlusion VOP will also drive the visual output of the Reflection Occlusion Pass.

CREATING A FRESNEL RENDER PASS

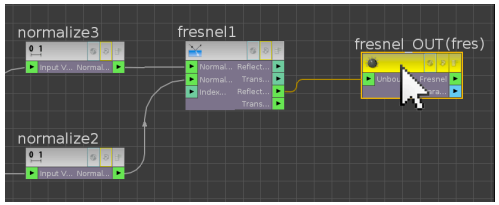
Reflection Occlusion takes its shading information from **Fresnel** information on a surface. This is where the **camera-facing edge outline** of the geometry is used to bias the direction of a standard ambient occlusion.

As a new node, create a **Fresnel VOP**. From the **Global Variables VOP** wire a **Normalised Direction from Eye to Surface output** into the **first** input of the **Fresnel VOP**.



Similarly, wire a **Normalized Surface Normal output** from the **Global Variables VOP** as the **second input** of the **Fresnel VOP**.

The output of a Fresnel VOP can also be exported as an Extra Image Plane as a way to control the reflective properties of shiny surfaces. While its export is not vital for this example, it is included for completeness.



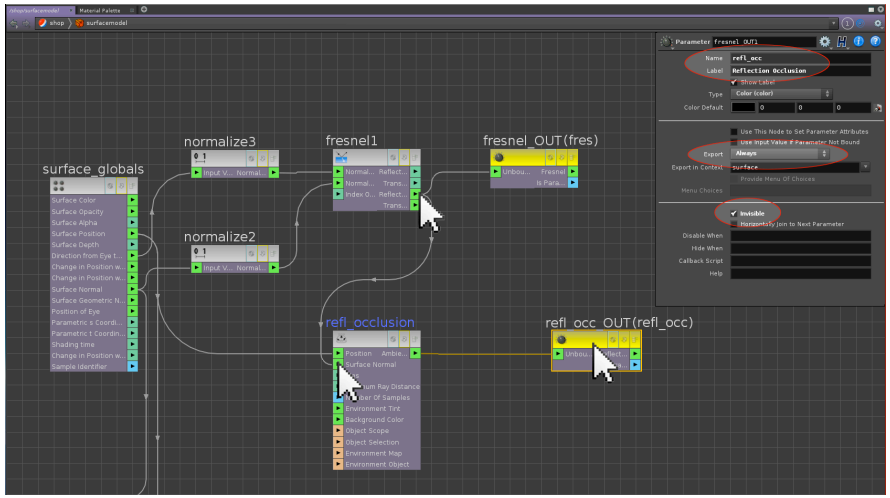
As before create an **export Parameter VOP** creating an **image plane** called **fres** from the Reflection Vector output. When the scene is rendered, a Fresnel pass can be seen in **MPlay**.

NOTE: The Reflected Light float output of the Fresnel VOP can also be exported, as this will isolate the blue channel of the Fresnel calculation.

The role of a Fresnel pass is to isolate the oblique edges of geometry (represented by blue in the render), so that reflectivity of surfaces can be controlled relative to real world phenomena. For mirrored surfaces the **Fresnel Effect** is not very noticeable; however can be observed in lesser reflective surfaces. When looking at a car for example, its reflectivity around its edges will be slightly brighter than the reflections directly facing the observer.

CREATING REFLECTION OCCLUSION

This Fresnel information can also be used to create a Reflection Occlusion render pass. Back at the **shader level**, wire the **Reflection Vector** output of the **Fresnel VOP** into the **Surface Normal input** of the second **Occlusion VOP**.



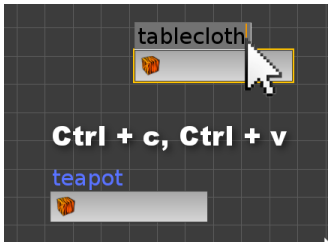
As before create a **Parameter VOP export** for the second Occlusion VOP called **refl_occ**, and activate it as an **Extra Image Plane** output on the **mantra_pbr ROP**. When the scene is rendered, a reflection occlusion pass can be seen in **MPlay**.

See file **surface_model_stage3.hipnc**

ADDITIONAL CONFIGURATION

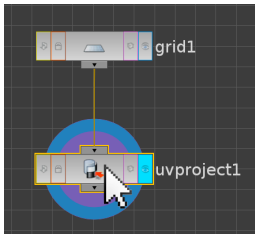
With this basic setup of a custom material complete, a new copy of it can be created; allowing for different materials to be further configured for both the grid and the teapot. As both custom materials have additional exports already assigned inside their networks, these exports will still all appear as Extra Image Planes in the resulting render.

At **SHOP Level**, **Copy** (**CTRL + c**) and **Paste** (**CTRL + v**) the **Surface Model Material**. **Rename** the **original** material to **teapot**, and the **copy** to **tablecloth**.

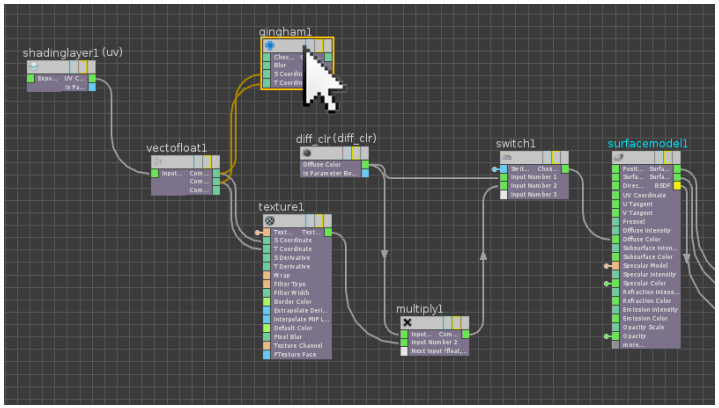


LMB Drag the **tablecloth material** onto the **grid** object in the **Viewer**, and **modify** the **teapot material** to **display** its **texture** in order to confirm the assignment of individual materials to the scene.

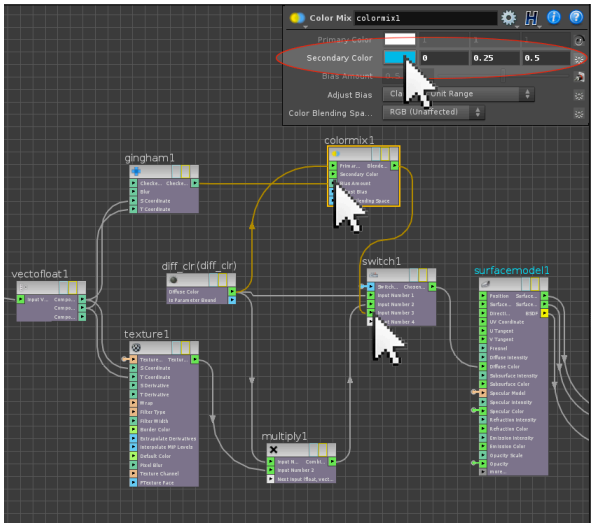
Inside the **grid_object**, append a **UV Project SOP** to the **output** of the **Grid SOP**.



Back at **Shader Level**, locate the **Vector to Float VOP** controlling the **Texture VOP**, and wire its **U** and **V** outputs into the respective **S** and **T inputs** of a **Gingham VOP**. The **Gingham VOP** is one of a **number preconfigured patterns** in VOPS, that can be used to drive a Color Mix. See the **TAB Menu – Patterns** section for the full pattern list.



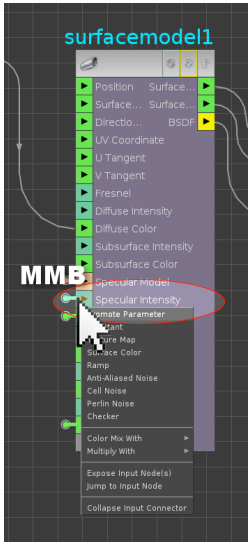
Create a **Color Mix VOP**, using the **Gingham VOP** as the **Bias Amount input**. Wire the default **Diffuse Color Parameter VOP** as the **Primary Color input** for the Color Mix VOP; and **set** the **Secondary Color** parameter to a **mid blue**.

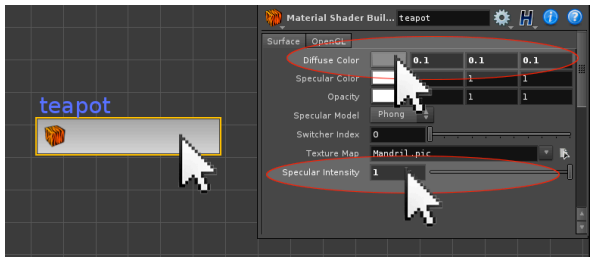


Wire the **output** of the **Color Mix** as the **third green input** for the **Switch VOP**. When the **Switcher Index** value is set accordingly, the **Gingham tablecloth** will appear on the grid geometry. See file **surface_model_stage4.hipnc**

NOTE: The blue nodule of the Switch VOP may need to be remade in order to get the switch mechanism working again. This may be due to modifying a copy of the original material.

Inside the **teapot material**, **MMB** promote the **Specular Intensity** parameter. This will control the extent of reflection in the teapot surface.





At **SHOP Level** set the **Diffuse Color** parameter to a **dark grey**; and the **Specular Intensity** parameter to **1**. When the scene is rendered, the teapot has a soft chrome aesthetic.

A ROUGH GUIDE TO MATERIAL TYPES

Creating metal, plastic or diffuse base surfaces can be done by modifying the Surface Model VOP when configuring a material, or by adjusting any of its promoted parameters at SHOP Level. As a rough guide:

Metal >			
Diffuse Intensity	**0**	**Specular Intensity**	**1**
Plastic >			
Diffuse Intensity	**0.5**	**Specular Intensity**	**0.5**
Diffuse >			
Diffuse Intensity	**1**	**Specular Intensity**	**0**

NOTE: In Houdini, Specular Intensity controls both specular highlights as well as reflections.

AMBIENT OCCLUSION V. REFLECTION OCCLUSION

With a chrome teapot configured a render can be tested in the composite and multiplied by the different occlusion passes. See file **surface_model_end.hipnc**