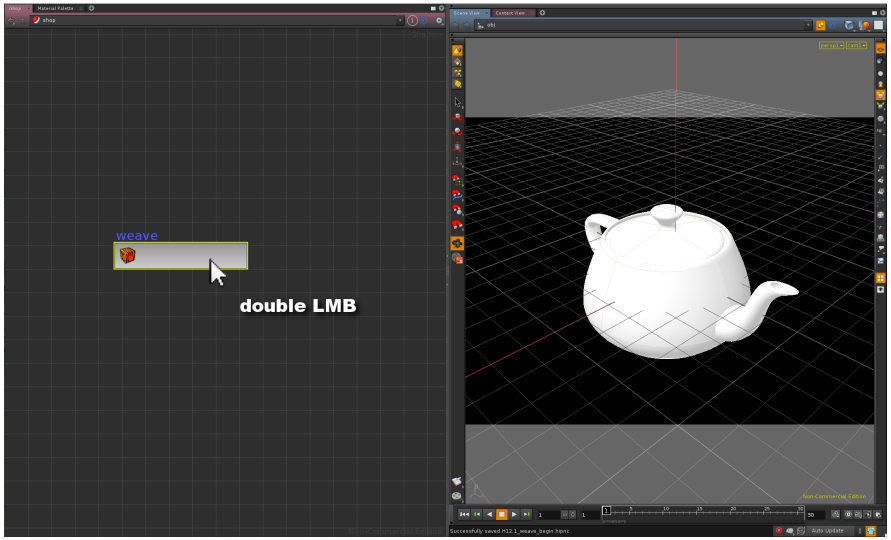
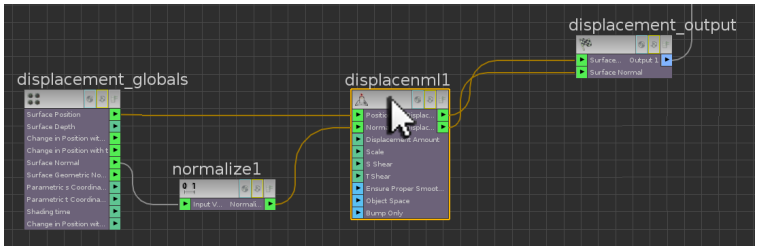
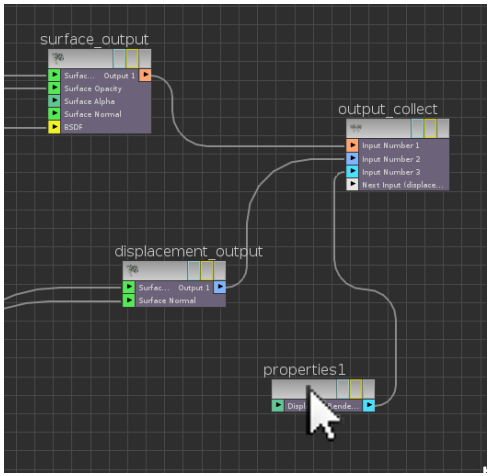
This example will look at creating a simple wicker weave shader to a preliminary level of shader engineering. **Open** the scene **H12.1_weave_begin.hipnc**. This scene contains a simple teapot setup for configuring a custom shader. An **empty Surface Model Material** called **weave** has been assigned to the teapot. Switch to **SHOP Level** and go inside the weave Material Shader Builder.



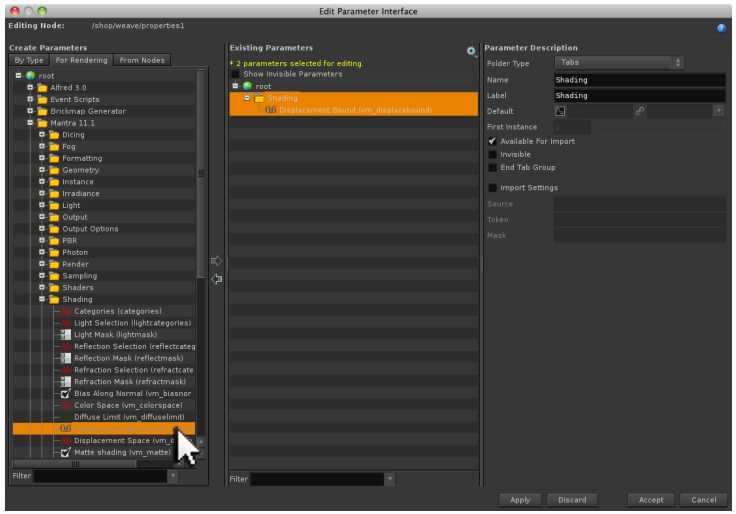
Here create a **Displace Along Normal VOP**, wiring its output directly into the **Displacement Output VOP**. For its **inputs**, wire **the Surface Position output** of the **Displacement Globals VOP** as its first input, with **Normalized Surface Normals** as its second input. This will create a displacement setup on which generated patterns can be tested.



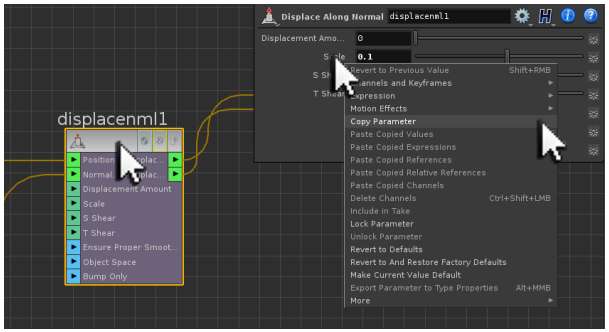
As a new node, create a **Properties VOP** and wire it into the **Output Collect VOP**. This node will be used to assign displacement bounding information to the shader for render time.



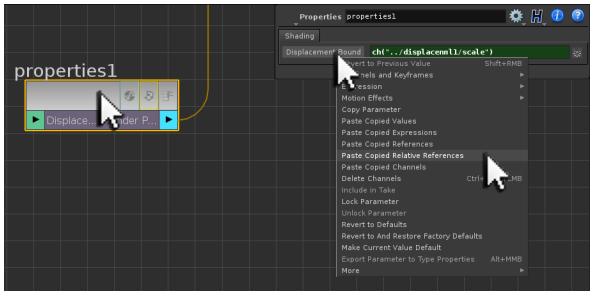
Using the **Properties VOP Cog button**, activate the **Edit Rendering Parameters** window and port across the **Displacement Bound parameter** found in **Create Parameters > For Rendering > Mantra > Shading**.



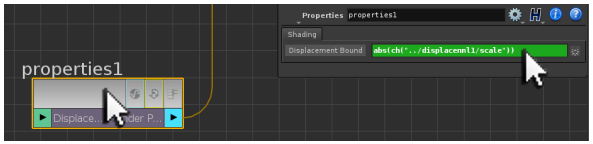
Press **Accept** to confirm the parameter editing. In the **Displace Along Normal VOP parameters**, set the **Scale parameter** to **0.1**, and **RMB** on it to **Copy** the **Parameter**.



In the **Properties VOP parameters**, **RMB** on the **Displacement Bound** parameter and choose **Paste Copied Relative References**.



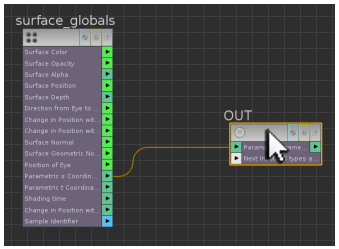
As a final step, wrap the channel reference in an **Absolute Function**, so that it always returns positive numbers.



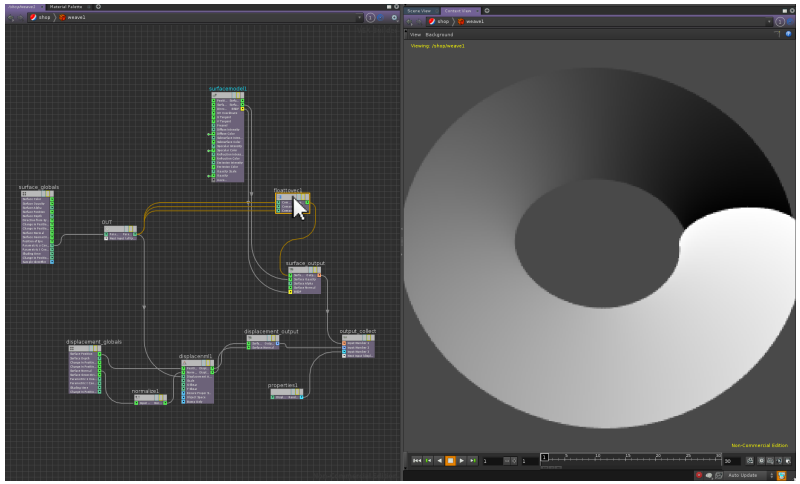
Now whatever patterns are piped into the Displace Along Normal VOP, they will displace and render correctly.

CONFIGURING A WEAVE PATTERN

As an initial step, append a **Null VOP** to the **Parametric s Coordinate** of the **Surface Globals VOP**. **Rename** this Null VOP to **OUT**.



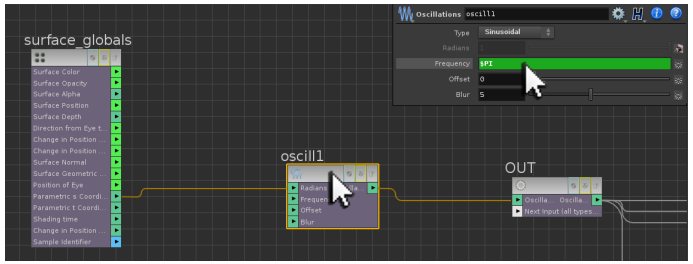
The **output** of this **Null VOP** can be fed into a **Float to Vector VOP** to drive the **Surface Output colour**; and into the **Displacement Amount parameter** of the **Displace along the Normal VOP** to drive the surface displacement.



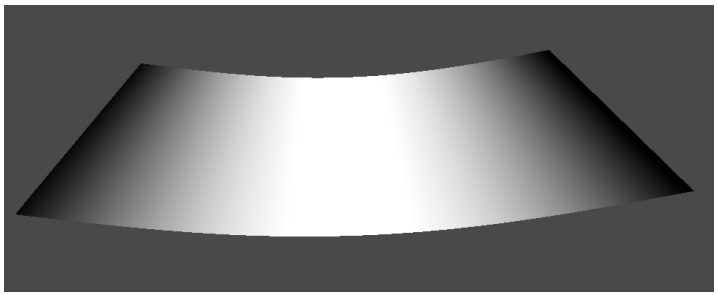
When the shader ball is examined as a torus (hotkey m), the effect of the **Parametric s Coordinate** can be seen as both colour and displacement, creating a simple horizontal ramp.

RMB on the **input** of the **OUT Null VOP** to insert an **Oscillations VOP**. In the **parameters** for the **Oscillation VOP** specify:

Type	**Sinusoidal**
Frequency	**$PI**

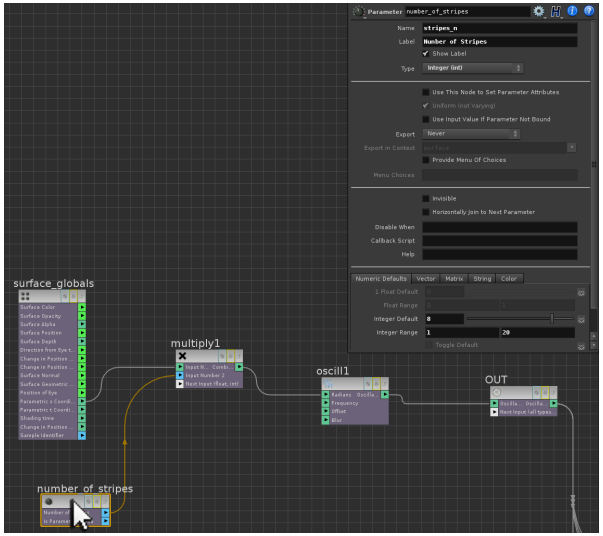


This will use the **Parametric s Coordinate** to drive a sine wave. Specifying a Frequency of $PI will ensure a sine wave arc from 0 to 1 to 0.



CREATING VERTICAL STRIPES

This base sine wave pattern can be further enhanced by multiplying the Radians input by a specific number to create stripes. **RMB** on the **Parametric s Coordinate output** of the **Surface Globals VOP** to insert a **Multiply VOP**.

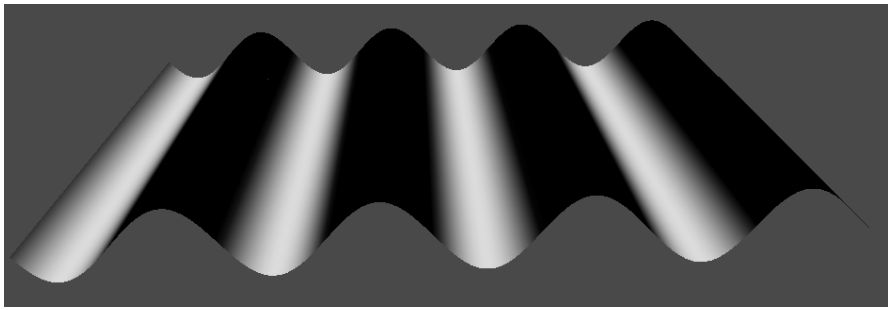


Promote the **second input parameter** of the **Multiply VOP**, and **double LMB** on the resulting **nodule** to **reveal the Parameter VOP** automatically created. In the **Parameters** for this **Parameter VOP** specify:

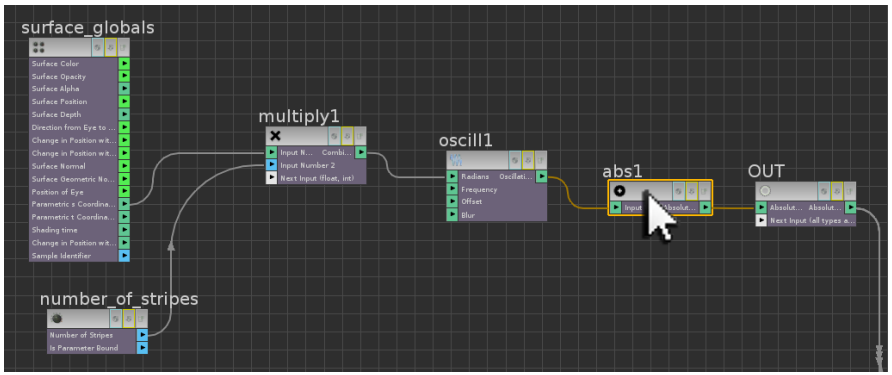
Name	**stripes_n**
Label	**Number of Stripes**
Type	**Integer**

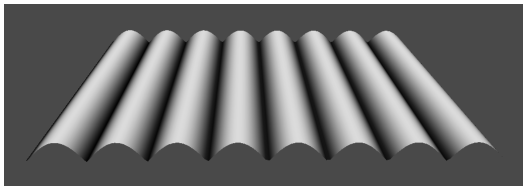
Numeric Defaults >

Integer Default	**8**	
Integer Range	**1**	**20**

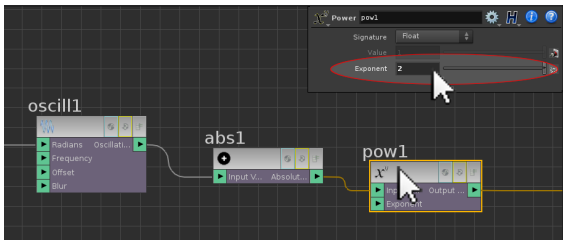


When the Shader Ball is examined using a Grid Display (hotkey m), a sine wave with peaks and troughs relative to the number of stripes setting is generated. This can be further enhanced by appending an **Absolute VOP** to the **output** of the **Oscillations VOP** to create ridged stripes.

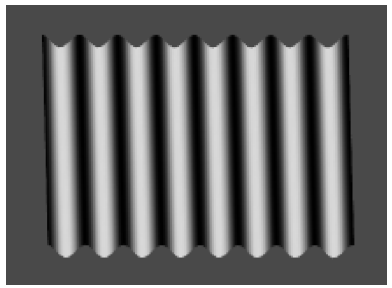




As a final step, a **Power VOP** can be introduced to control how the falloff from white to black should happen.

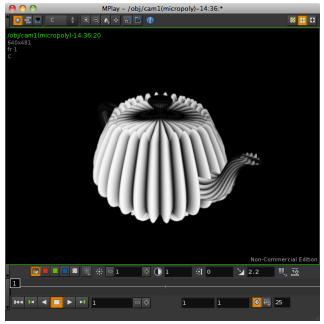


Setting an **Exponant** value of **2** will decrease the falloff from white to black giving thinner stripes with larger black spaces inbetween.



NOTE: Do not worry if the Shader Ball preview appears upside down relative to the displacement taking place. Simply adjust the view so it makes visual sense.

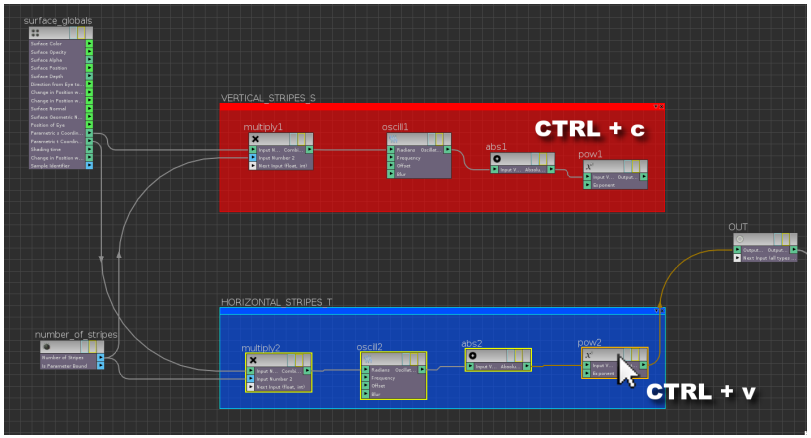
NOTE: The **Offset parameter** of the **Oscillations VOP** can be adjusted to ensure the end waves match (suggest **0.1** as an Offset value for 8 stripes). This Offset parameter may have to be manually adjusted depending upon the number of stripes specified. A procedural solution for generating the offset amount could become a higher-level shader writing task after its preliminary engineering.



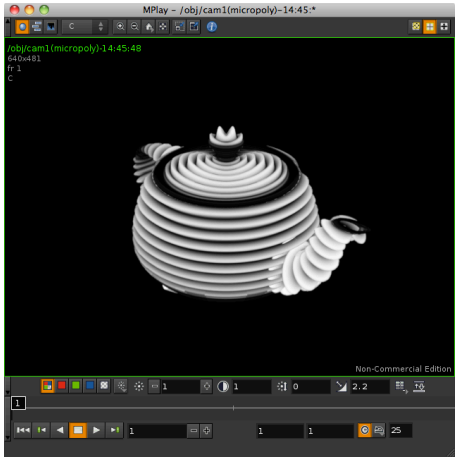
When the teapot is rendered, the displacement and colour effect of the stripes can be seen.

CREATING HORIZONTAL STRIPES

Horizontal Stripes can be generated by copying and pasting (CTRL + c, CTRL + v) the **Multiply**, **Oscillations**, **Absolute and Power VOPs**, feeding in the **Parametric t Coordinate** to drive their creation.



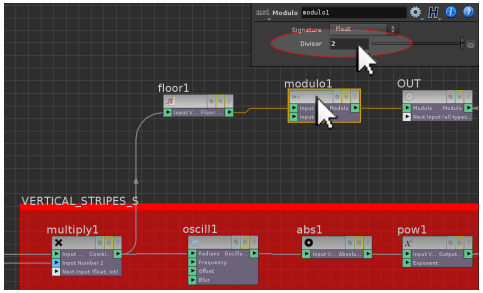
Netboxes can also be used to isolate these different areas of the shader network.



See file **H12.1_weave_stage1.hipnc**.

CREATING A CHECKER PATTERN MIX FOR THE STRIPES

MMB on the **output** of the **Vertical Stripes Multiply VOP** and create a **Floor VOP** as a new network branch. This will take the incoming float value and return its primary integer value.

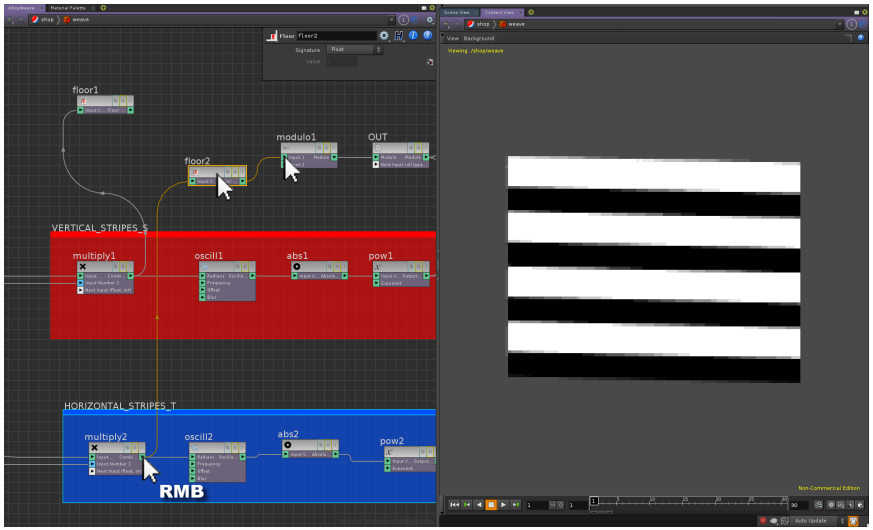


This in turn can be passed into a **Modulo VOP** to count the number of integers being produced by the Floor VOP. In the **parameters** for the **Modulo VOP** specify:

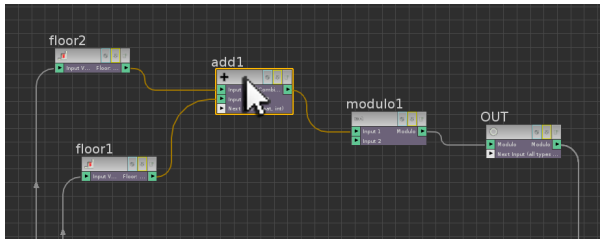
Divisor **2**

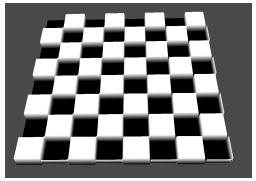
This will take the incoming integer values (0,1,2,3,4,5,6,7) and return either 0 or 1 as the result. When this network branch is fed into the **OUT Null VOP**, hard black and white alternating vertical stripes are created.

This effect can also be created from the **horizontal stripes**, by **RMB** appending a **Floor VOP** as a new network branch to the **output** of the **horizontal stripes Multiply VOP**; and feeding it into the **Modulo VOP**.

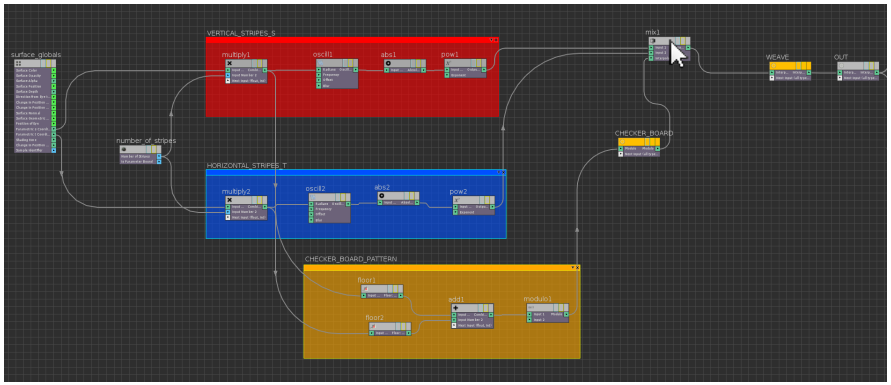


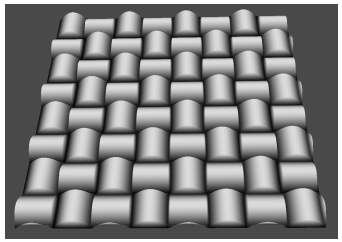
Adding the outputs of the **Floor VOPs** together before the result is passed into the Modulo VOP will generate a simple checkerboard pattern.





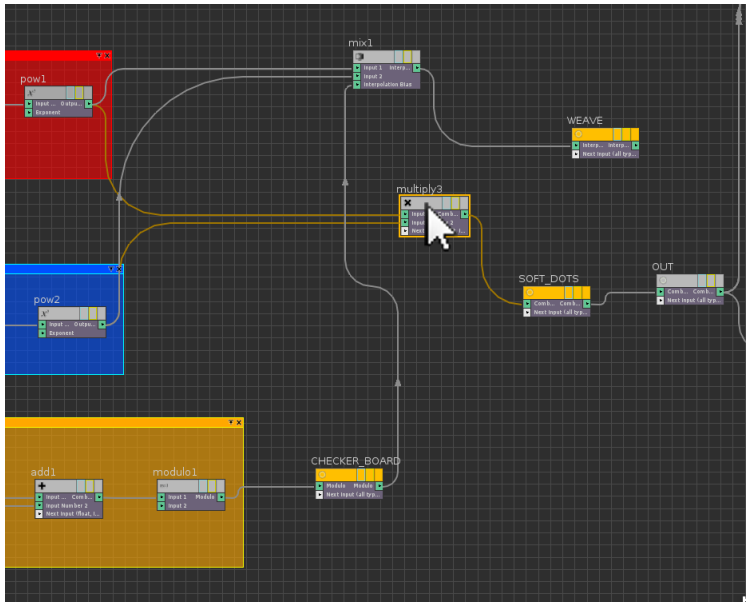
This checkerboard pattern can now be used to drive the **Interpolation Bias parameter** of a **Mix VOP**, alternating between the vertical and horizontal stripe patterns.



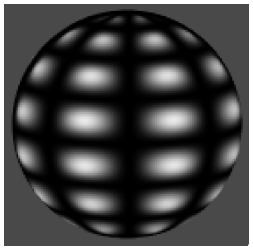


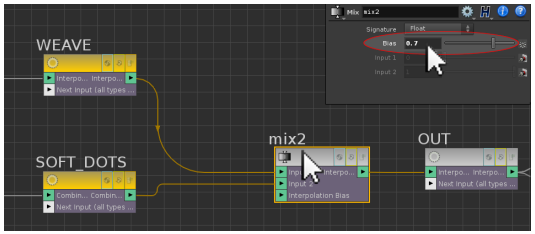
While this results in the weave pattern, the edges of each stripe area need to be naturalized to create a better surface flow. Currently each stripe area has a hard falloff creating a series of oriented tubes rather than a weave pattern in the proper sense.

NATURALISING THE WEAVE

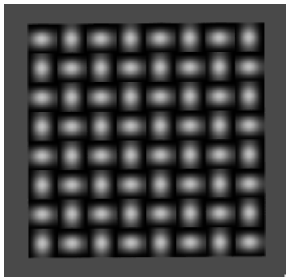


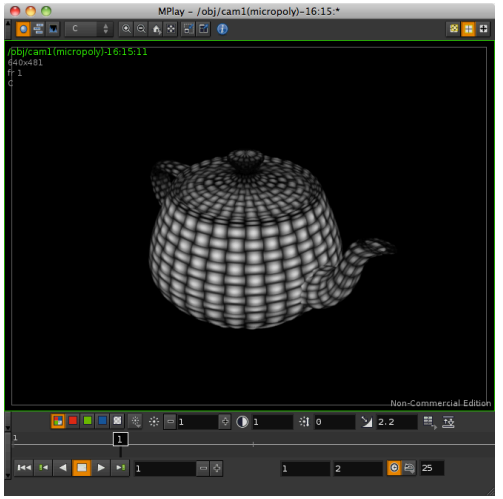
Use a Mulitply VOP to multiply together the Power VOPs of the Horizontal and Vertical Stripes Networks. Append to the Multiply VOP a **Null VOP** renamed to **SOFT_DOTS**.





RMB on the **input** of the **OUT Null SOP** to insert a **Mix VOP**. Wire the **WEAVE** and **SOFT_DOTS patterns** into its **inputs** and set **0.7** as an **Interpolation Bias amount**.

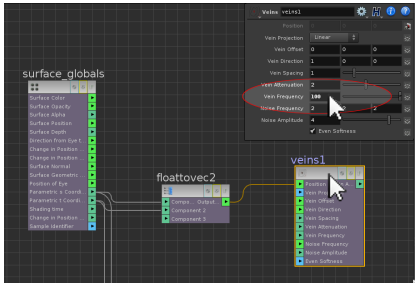




The displacement amount can be reduced by setting the **Scale parameter** of the **Displace Along Normal VOP** to **0.03** rather than **0.1**. Once a new Scale value has been set, the **Scale parameter** can be **promoted** as an end user control if required. See file **H12.1_weave_stage2.hipnc**

ADDING MORE DETAIL

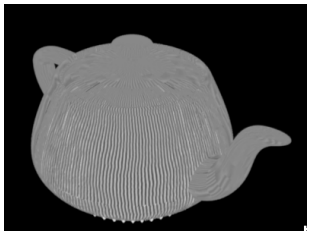
With the main weave pattern established, extra detail can be added into the surface to help convey wood. Pipe the **outputs** of the **Surface Global Parametric S & T Coordinates** into a **Float to Vector VOP**.



Wire the output of the **Float to Vector VOP** into the Position input of a **Veins VOP**. In the **parameters** for the **Veins VOP** specify:

Vein Frequency **100**

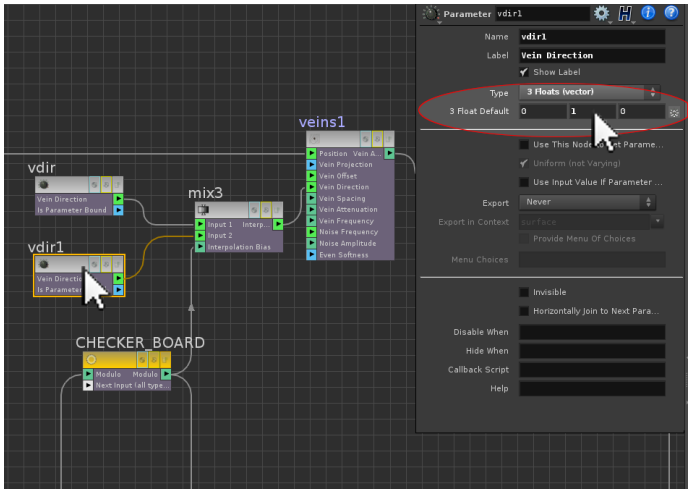
When the **output** of the **Veins VOP** is fed into the **OUT Null VOP** the effect of the veins can be seen in a render.



NOTE: The Veins VOP asks for a Position input. Had Global Variable P been used instead of the custom S & T Position, the vein direction would be akin to a carved block of wood rather than thin veins running in the vertical direction over the surface.

CONTROLLING THE VEIN DIRECTION

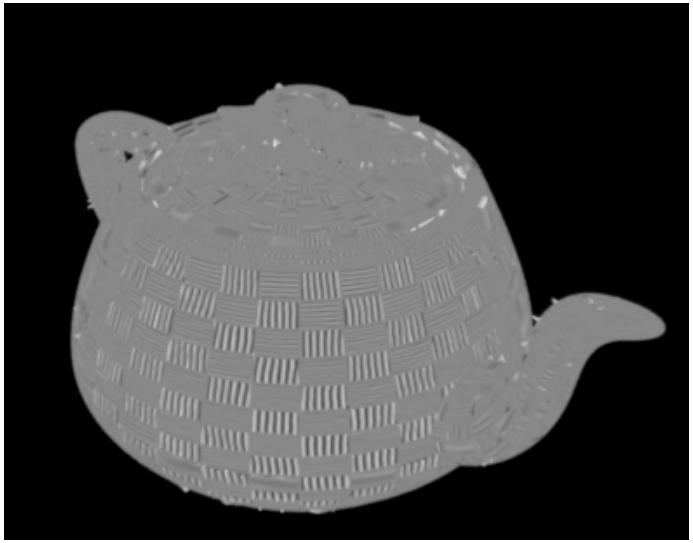
As this vein pattern needs to align with a weave pattern, the checker board pattern of the weave network can be used to drive the direction of the veins. **MMB Promote** the **Veins Direction parameter** and **double LMB** on the resulting **nodule** to **expose the node**. **RMB** on the **Viens Direction input** to **insert** a **Mix VOP**, and **Copy (CTRL + c)** and **Paste (CTRL + v)** the **vdir parameter node** to create a second instance of it.



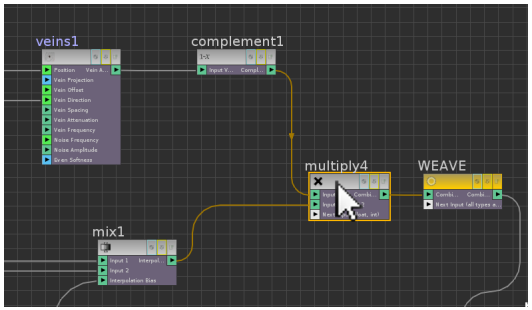
Give a **new name** to the **vdir1 node** to **unlock** its **parameters** and **specify** a 3 Float Default of:

3 Float Default	**0**	**1**	**0**

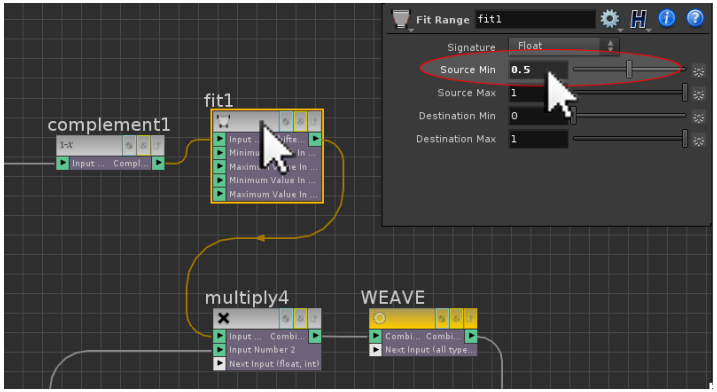
The **output** of the **CHECKER_BOARD Null VOP** can now be used as the **Interpolation Bias input** of the **Mix VOP**.



When the scene is rendered, the direction of the veins is driven by the checkerboard pattern.



The output of the veins network can now be inverted using a Compliment SOP to ensure the displacement of the veins go inward rather than outward; and multiplied into the WEAVE Null VOP.



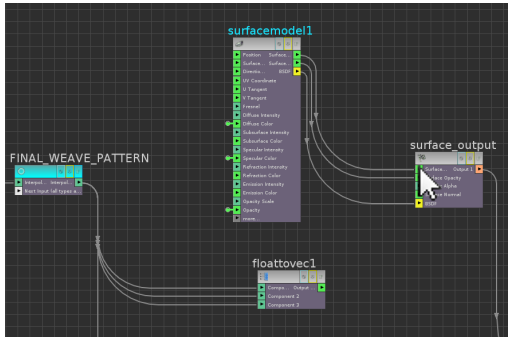
A **Fit Range VOP** can also be added to the vein pattern. Setting a **Source Min** parameter value will boost the effect of the vein pattern on the surface of the teapot.

The **OUT Null VOP** can now be **reset** to the **WEAVE and SOFT_DOTS Mix VOP** to see the effect of the combined pattern.

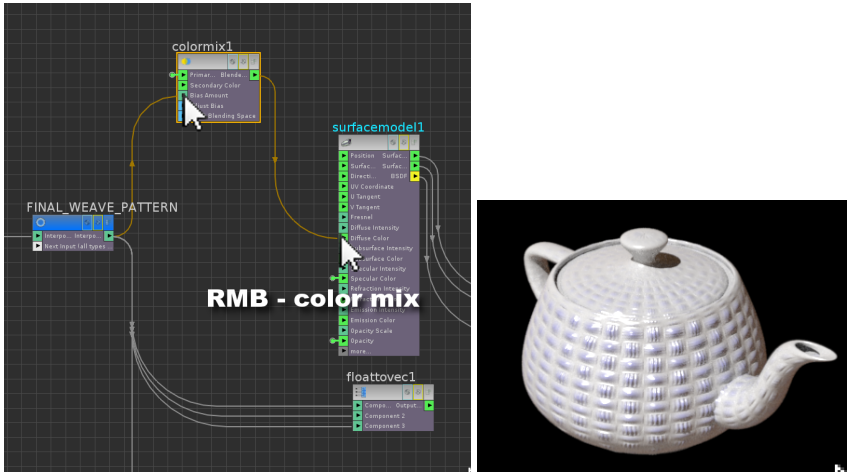
See file **H12.1_weave_stage3.hipnc**.

CONFIGURING THE SURFACE COLOUR

With the displacement of the material engineered to a primary level, its components can also be used to drive surface colour. Rename the **OUT Null VOP** to **FINAL_WEAVE_PATTERN**.



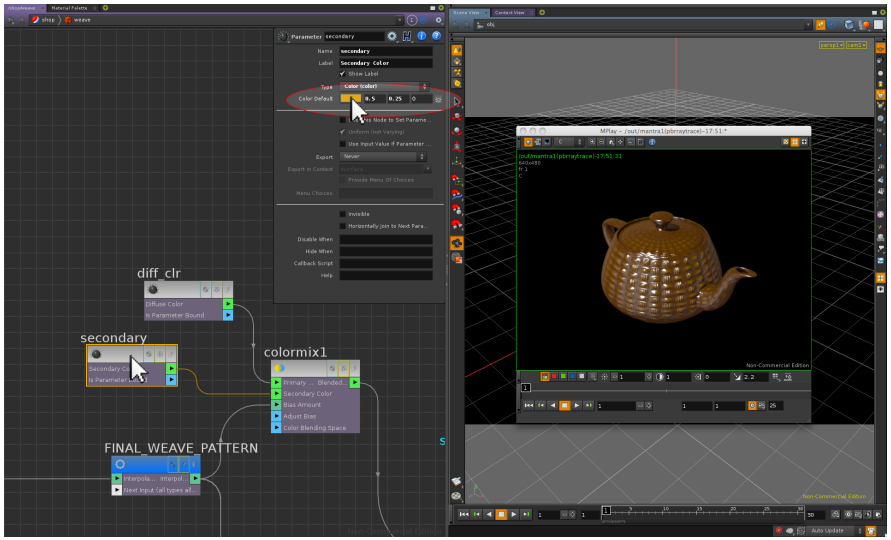
Disconnect the **Float to Vector VOP** passing the weave pattern into the **Surface Ouput VOP** and in its place **rewire** in the **Surface Color output** of the **Surface Model VOP**.



RMB on the **Diffuse Color input** of the **Surface Model VOP** to create a **Color Mix VOP**, wiring the **FINAL_WEAVE_PATTERN** as the **Bias Amount** input. When the teapot is rendered again, the Color Mix VOP now informs the surface shading.

NOTE: The original Diffuse Color input Parameter nodule is preserved when the Color Mix VOP is created.

The Color Mix VOP's primary colour can be adjusted to a dark brown, and its secondary color can be adjusted to a mid brown.

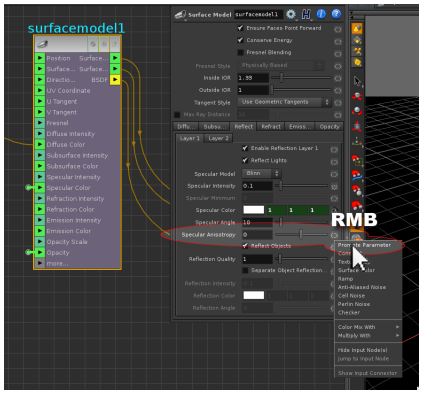


The Secondary Color parameter of the Color Mix VOP can also be promoted as end user controls.

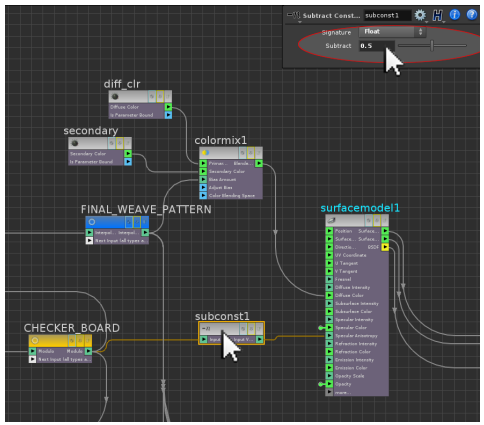
CONTROLLING THE SPECULAR ANISOTROPY

Currently the specular values of the surface are too intense, and also highlight the entire surface. This can be controlled in a couple of ways on the **Surface Model VOP**. **Specular Anisotropy** controls the vertical and horizontal direction of the specular highlight. In the case of the weave pattern, it would be useful to have specular highlights running in the same direction as the weave pattern.

RMB on the Specular Anisotropy parameter wheel to promote the Specular Anisotropy parameter.



The resulting parameter nodule can be exposed and its value adjusted to see the effect. Values less than 0 will sharpen the specular highlight horizontally while values larger than 0 will sharpen the specular highlight vertically. With this understanding of specular anisotropy, the **CHECKER_BOARD Null VOP** can be used to drive the Specular Anisotropy parameter in place of its promoted parameter.



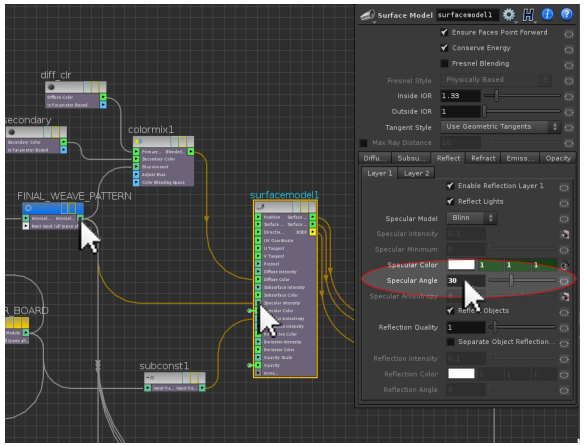
RMB on the **Specular Anisotropy input** to insert a **Subtract Constant VOP**. In the **parameters** for the **Subtract Constant VOP** specify:

Subtract **0.5**

This will re-range the checker pattern from 0-1 to -0.5-0.5, resulting in a discreet anisotropic controlled specular highlight over the surface of the teapot.

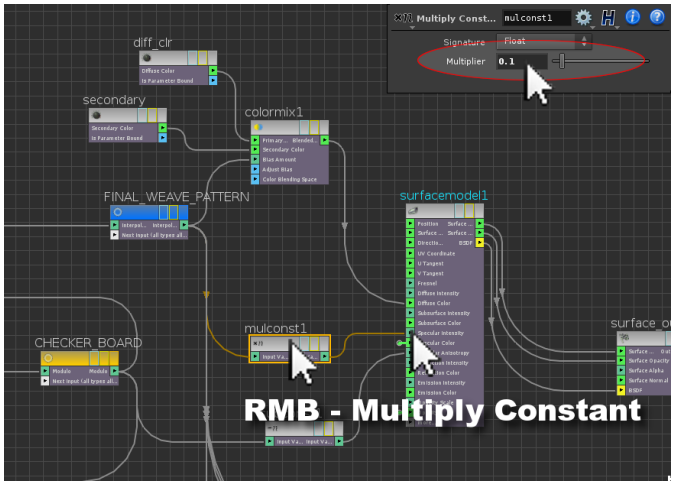
CONTROLLING THE SPECULAR INTENSITY

The **FINAL_WEAVE_PATTERN** can also be used to drive the **Specular Intensity input** of the **Surface Model VOP**. The **Specular Angle** of the **Surface Model VOP** can also be increased to **30** to give a more scattered specular return.

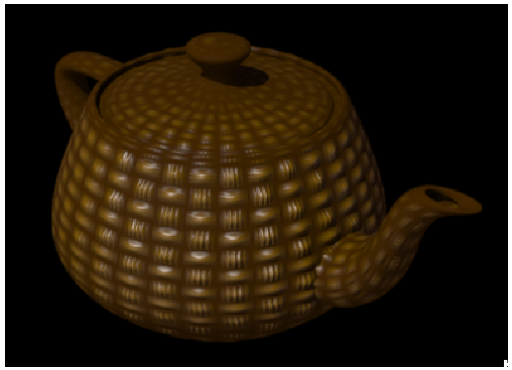


As a final step, the Specular Intensity can be diminished by **RMB inserting** a **Multiply Constant VOP** into the **Specular Intensity** input of the **Surface Model VOP** and in its **parameters** specifying:

Multiplier **0.1**



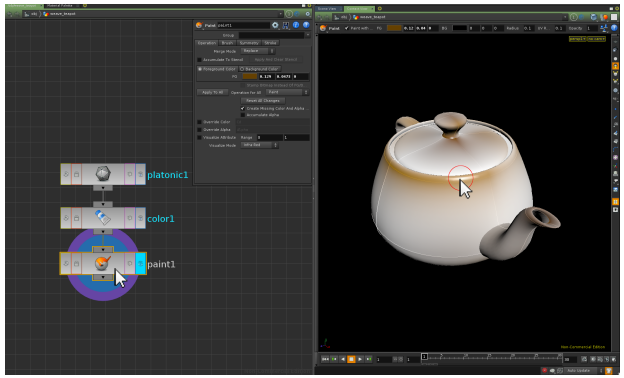
The base colours of the weave material can also be further adjusted to create a naturalized wood effect.



See file H12.1_weave_stage4.hipnc

AGEING THE SURFACE

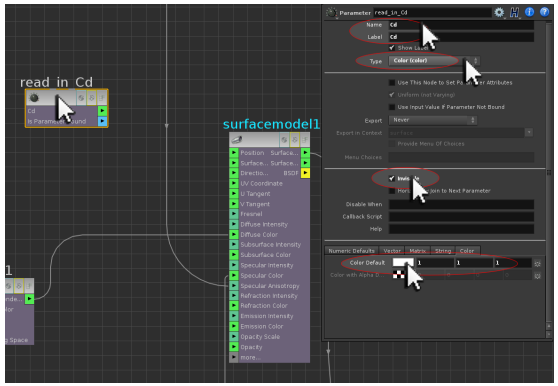
As a final step, areas of the teapot can be directly painted on to help create natural weathering. Go to the **Geometry Level** for the **teapot object**, and append a **Color SOP** to turn the surface **white**, and a **Paint SOP** to the **Color SOP**. This will allow the surface of the teapot to be painted on, creating a colour attribute (Cd) as a result.



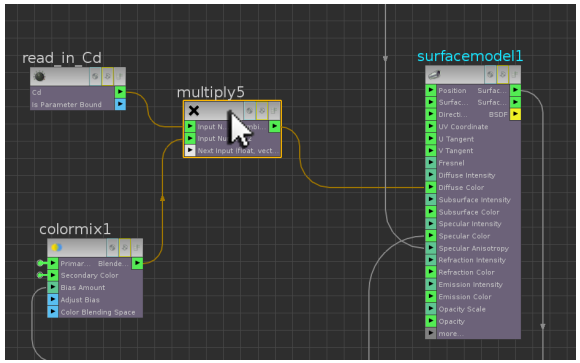
Back at Material Shader level, create a Parameter VOP to read in the Cd colour attribute near the Surface Model VOP. In the parameters for this new node specify:

Name	**Cd**		
Label	**Cd**		
Type	**Color**		
☑ **Invisible**			
Color >			
Color Default	**1**	**1**	**1**

Setting white as the default colour will help ensure that if no Cd attribute has been created on the geometry, the shader will still render as expected.



This **Parameter VOP** can now be **multiplied** with the **Diffuse Color input** of the **Surface Model VOP**.



When the teapot is rendered again, the effect of the paintwork can now be seen on the surface of the weave material.

All that remains after this initial shader engineering is to review and consolidate the end user parameters, and test the material relative to production in order to improve its aesthetic. **See file H12.1_weave_end.hipnc**