ATTRIBUTES OVERVIEW

Attributes are customisable **characteristics** of geometry. **By default** a piece of geometry will have the **Point Attribute 'P'** that stores the **spatial position** for each **point** as **XYZ** coordinates.

As geometry gets modified, **other attributes can get added** to the geometry, **increasing the number of attributes** assigned to it. For example, when **Surface Normals** are activated on geometry via a **Point SOP**, the **Point Attribute 'N'** is created and added to the geometry's attribute list alongside 'P'.

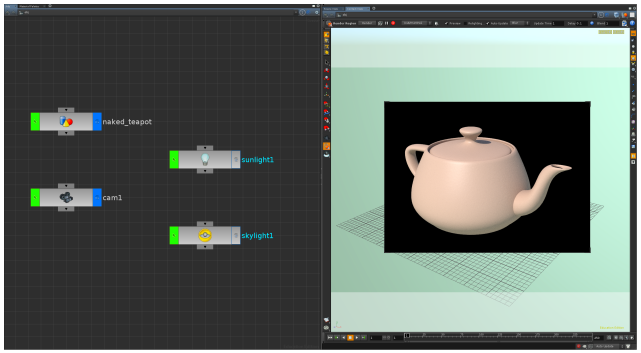
Custom Attributes can also be created for geometry as either standalone events, or as a way to override existing standard parameter behaviours.

As well as **Point Attributes** there are also **Primitive Attribute**, **Vertex Attribute** and **Detail Attribute classifications**. Attributes in any of these classifications can be stored as **Float**, **Integer**, **Vector** or **String** types.

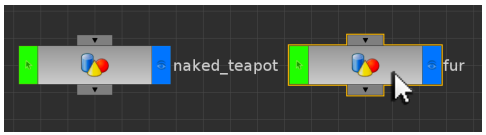
Attributes can also be **transferred** and **copied** between differing geometry, as well as being **promoted** from one attribute classification type to another.

The **more attributes assigned** to a piece of geometry, **the longer it can take to process**. Understanding how to work with and manage attributes is therefore a key part of an animation workflow.

Open the scene **naked_teapot_begin.hipnc.** This scene contains a naked polygon teapot and a simple light and camera setup.

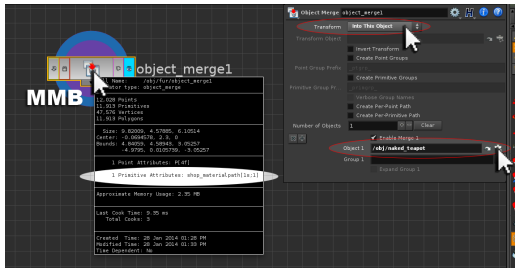


This example will examine the usage of attributes in Houdini, comparing a **hand configured fur setup** with the **automated Fur Shelf Tools**.



At **Object Level**, create a **new** piece of **geometry** and **rename** it to **fur**. Inside the fur Object, **delete** the default **File SOP** and in its place create an **Object Merge SOP**. This can be used to read in the naked teapot geometry. In the **parameters** for the **Object Merge SOP** specify:

Transform	Into This Object
Object 1	/obj/naked_teapot



MMB on the Object Merge node reveals that the teapot geometry being imported has a Primitive Attribute assigned to it denoting the skin material path location. As this attribute is not required for the fur, it can be deleted.

DELETING ATTRIBUTES

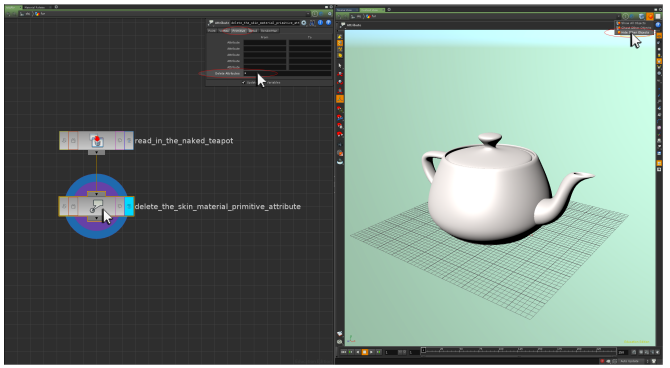
The Attribute SOP can be used to delete any excess attributes from geometry. It also has functionality for renaming attributes. Deleting any unnecessary or unwanted attributes can help speed up rendering and processing.

Append an **Attribute SOP** to the Object Merge SOP, and in the **Primitive section** of its **parameters** specify:

Delete Attributes *

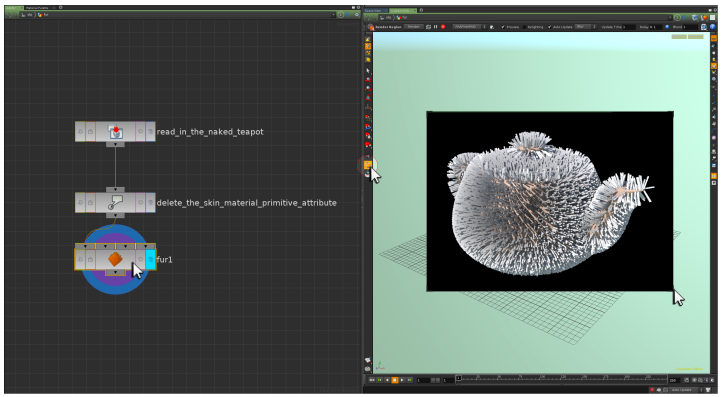
This will remove the skin material attribute from the teapot geometry, leaving a default grey teapot visible in the Viewer.

NOTE: The **Hide Other Objects** view option can be activated so only the contents of the fur object network are visible in the Viewer.



THE FUR SOP

As the teapot geometry is polygon based, it can be wired directly into a **Fur SOP**. Setting the View mode to **See All Objects** or switching over to a **Scene View** will allow for both the fur and the teapot to be seen together.



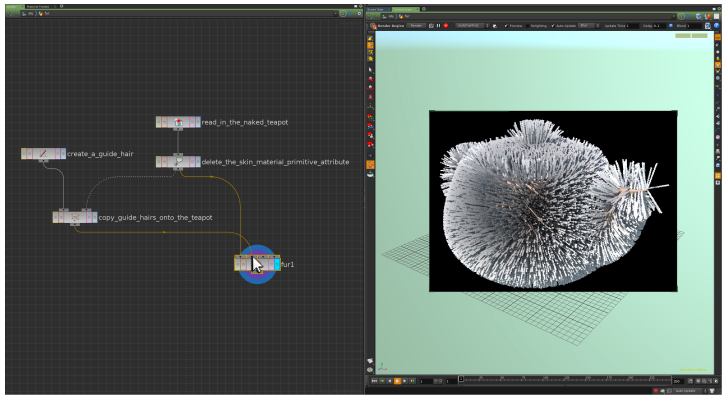
When the **scene** is **rendered**, the fur appears as a series of front facing rectangles.

REFINING THE FUR AESTHETIC

Create a **Line SOP**, and in its **parameters** specify:

Direction	**0**	**0**	**1**

Append to the Line SOP a **Copy SOP**, and wire the **output** of the **Copy SOP** as the **second input** of the **Fur SOP**. This copied Line SOP will create a series of guide hairs for the Fur SOP.



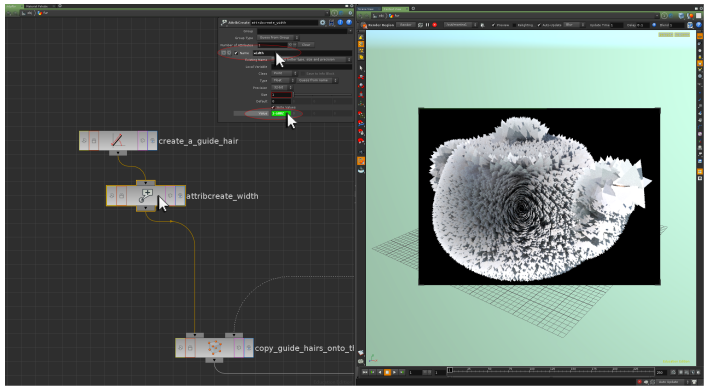
While the initial aesthetic of the fur render is similar to before, the network will now allow for the Line SOP guide hair to be customised in a bespoke way.

NOTE: The output of the Copy SOP could also be passed into a dynamics network and simulated accordingly. The dynamic guide hairs would then form the second input of the Fur SOP, with any movement of the resulting fur controlled by this dynamic guide hair animation.

To the output of the Line SOP append an **Attribute Create SOP**. This node allows for the declaring and assigning custom characteristics to geometry in Houdini. In the **parameters** for the **Attribute Create SOP** specify:

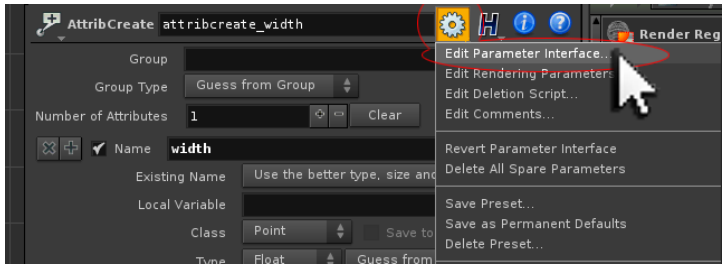
Name	**width**			
Value	**1 - $BBZ**	**0**	**0**	**0**

This will assign a width value from **1** to **0** down the length of the hair. **$BBZ** returns the **bounding box** information of the **Z Axis** from incoming geometry as a 0 to 1 range. Specifying **1-$BBZ** reverses the direction of the 0 to 1 $BBZ result to 1 to 0.

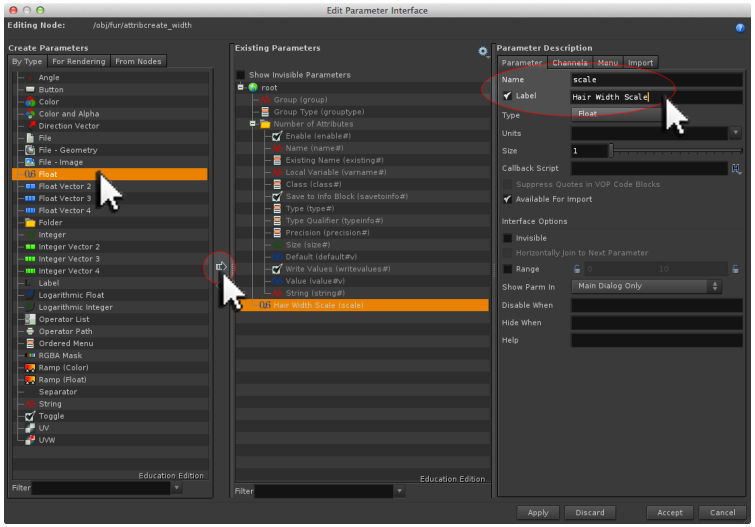


The fur now renders as large front-facing triangles. As the **Fur SOP anticipates** an **attribute** called **width**, this **custom width attribute** will automatically override the standard width attribute assigned in the Fur SOP. A **full list of** inherent **attributes** for the **Fur SOP** can be found in the **Fur SOP Help Card**; all of which can be overridden in this way.

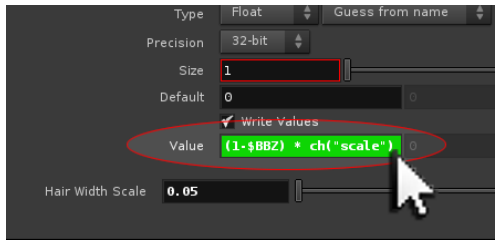
The width of the fur can be modified by creating a custom parameter on the Attribute Create SOP to scale down the triangular effect.



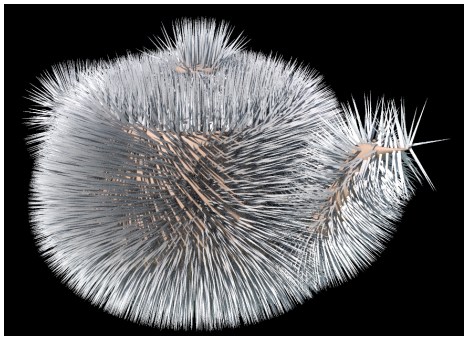
On the **parameters pane** for the **Attribute Create SOP**, activate the **Edit Parameter Interface** option of the **Cog Menu**.



Using the **Edit Parameter Interface** dialog window, port across a new **Float** type **parameter** to the **Existing Parameters list**, setting its **Name** as **scale**, and its **Label** as **Hair Width Scale**. When **Accept** is pressed, this custom parameter will now appear on the Attribute Create SOP's parameter pane.



Modify the **Value expression** to **(1-$BBZ) * ch("scale")**, and set the **Hair Width Scale** to a value of **0.05**.

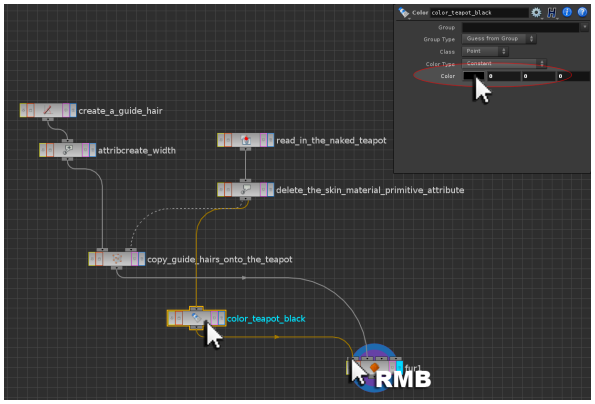


The resulting hair width of the render is now much more appropriate. **See file naked_teapot_stage1.hipnc**

SETTING HAIR COLOUR

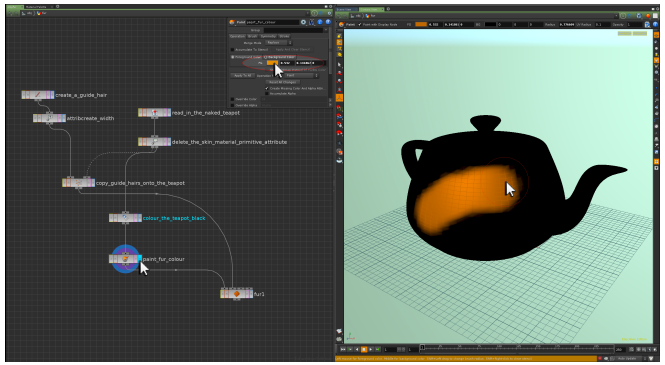
Hair colour for the fur can be activated by painting colour directly onto the teapot geometry. As a base for this colourisation, the teapot can be first turned black using a **Color SOP**. This will give a blank canvas on which colour can then be painted. **RMB** on the **first input** of the **Fur SOP** to **insert** a **Color SOP** into the network. In the **parameters** for the **Color SOP** specify:

Color	**0**	**0**	**0**

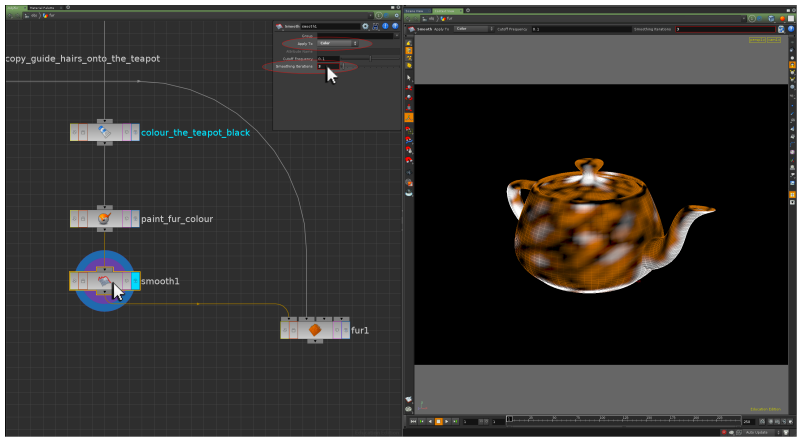


This will create the Point Attribute **Cd** on the teapot geometry, which stores this colour information. This base colour can now be painted on using a Paint SOP to create more bespoke fur colour.

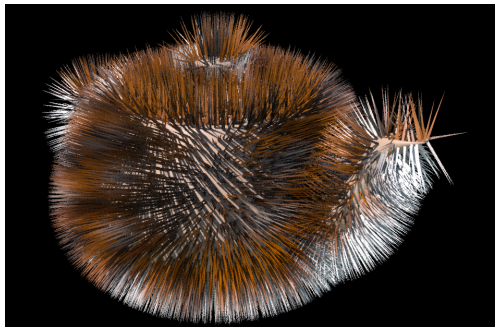
Append a **Paint SOP** to the **Color SOP**, and press **ENTER** with the **mouse over the Viewer** to activate its **tool mode**. **SHIFT + LMB dragging** the brush across the Viewer will interactively adjust the radius of the brush.



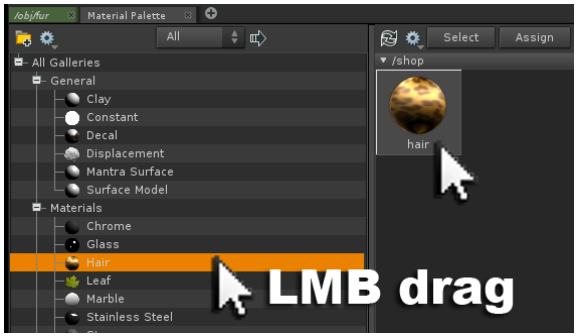
Appending a **Smooth SOP** to the **Paint SOP** can further refine the paintwork of the teapot geometry. By **default** a Smooth SOP will smooth the **shape** of the teapot, however **Color smoothing** can be activated instead of geometry smoothing.



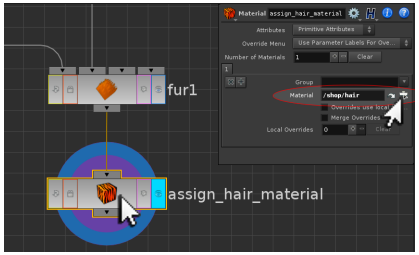
When the **Fur SOP** is rendered, the colour information is passed onto the fur.



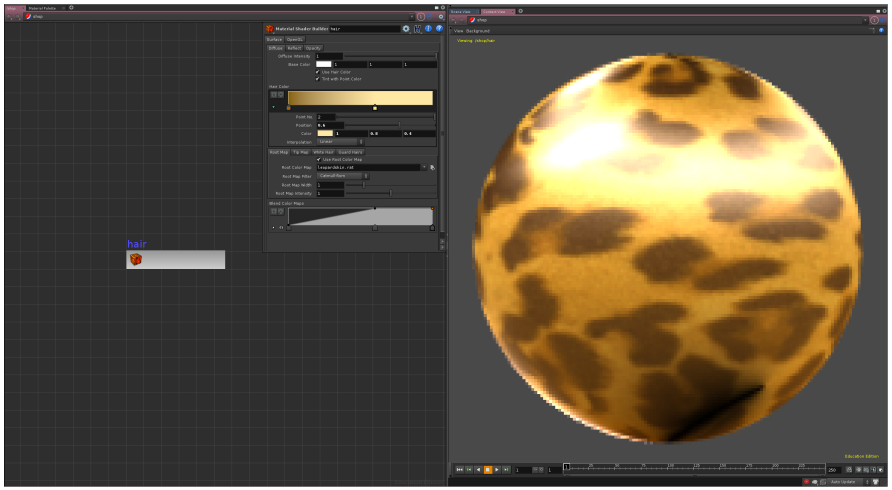
The naturalism of the fur render can also be improved upon by formally assigning a Hair Material to the fur geometry.



Switch the Network Editor view over to the **Material Palette** and **LMB Drag** and **Drop** a **Hair Material** into the **/shop palette** region. This material can then be assigned to the fur geometry using a Material SOP.



When the **scene** is **rendered** again however, the resulting aesthetic is now slightly different to before. This is due to the **default settings** of the **Hair Material**. When the **Hair Material** is **examined** at **SHOP Level**, its **leopard print default** pattern is responsible for this aesthetic change.



In the **parameters** for the **Hair Material** specify:

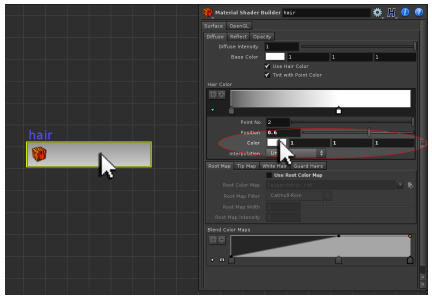
Root Map >

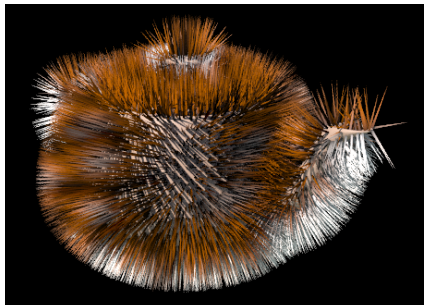
- ☐ **Use Root Color Map**

Tip Map >

- ☐ **Use Tip Color Map**

This will strip out the leopard skin print from the hair material aesthetic.



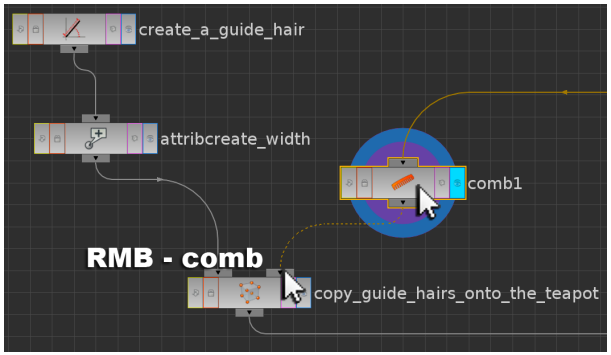


The **Hair Color** Ramp can also be adjusted to **grey** and **white** values, which in turn will be multiplied by the custom painted fur colour at render time. The resulting render is now much closer to what was occurring before; however now the fur has a natural shiny hair aesthetic. **See file naked_teapot_stage2.hpinc**

STYLING THE FUR

Currently the guide hair is being copied onto the teapot geometry perpendicular to its surface in accordance with its surface normal information. By modifying this surface normal information, hair styling can take place.

In the **Network Editor RMB** on the **second input** of the **Copy SOP** to **insert** a **Comb SOP**. This operator can be used to comb the direction of Surface Normals before the hair geometry is copied onto the teapot.



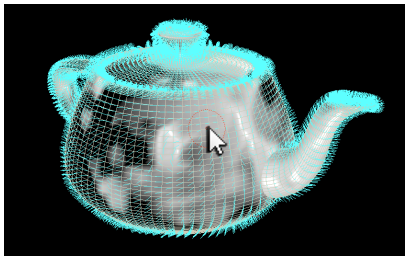
The Comb SOP will also automatically create the **Normals attribute**, usually activated using a Point SOP. The **Display** of **Surface Normals** can be **activated** in the **right hand Viewer stow bar** to make the combing process simpler.

In the **parameters** of the **Comb SOP** specify:

Comb Lift **0.5**

This will ensure the combed surface normals are never fully flattened or pushed through the teapot geometry surface.

The surface normals of the teapot geometry can now be combed to create swirling fur patterns.

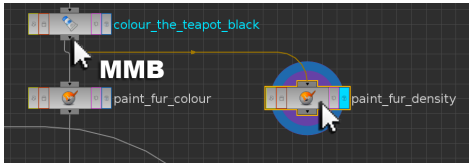


When the Material SOP is rendered, the effect of the styling can be seen. **Shortening** the **Distance parameter** of the **Line SOP**, and increasing the **Density parameter** of the **Fur SOP** can also improve the aesthetic of the fur. **Copy Stamping** can also be used to **vary** the **guide hair length** over the teapot surface.

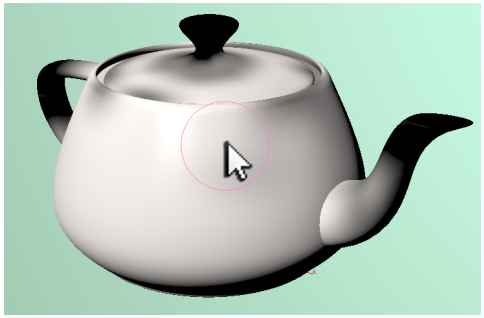
Increasing fur density may however impede interface interactivity, so should be done at the final rendering stage. **See file naked_teapot_stage3.hipnc**

MODIFYING THE FUR DENSITY

Areas of hair growth can also be determined on the surface geometry. This is done by overriding the **furdensity attribute** of the **Fur SOP**. This attribute is utilised as a multiplier for the Density parameter, and if not set returns a value of 1 instead.

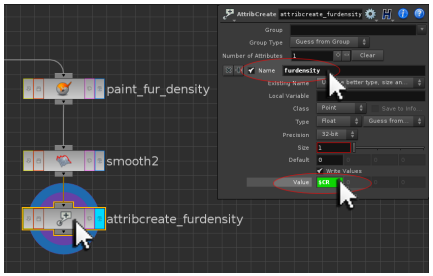


MMB on the output of the **Color SOP** setting the **teapot** to **black**, and append a **second Paint SOP** as a new network branch. **Rename** this Paint SOP to **paint_fur_density**.



In the **Viewer** paint the teapot, creating **areas of white** (where the fur distribution will be most dense) and **areas of grey** (where fur distribution will be less dense). **Areas left black** will become bald in the final render. Appending a **Smooth SOP** to this network branch can also soften the density colour.

To the **Smooth SOP** append an **Attribute Create SOP**. This can be used to set this custom fur density colour as an attribute expected by the Fur SOP.



In the **parameters** for the **Attribute Create SOP** specify:

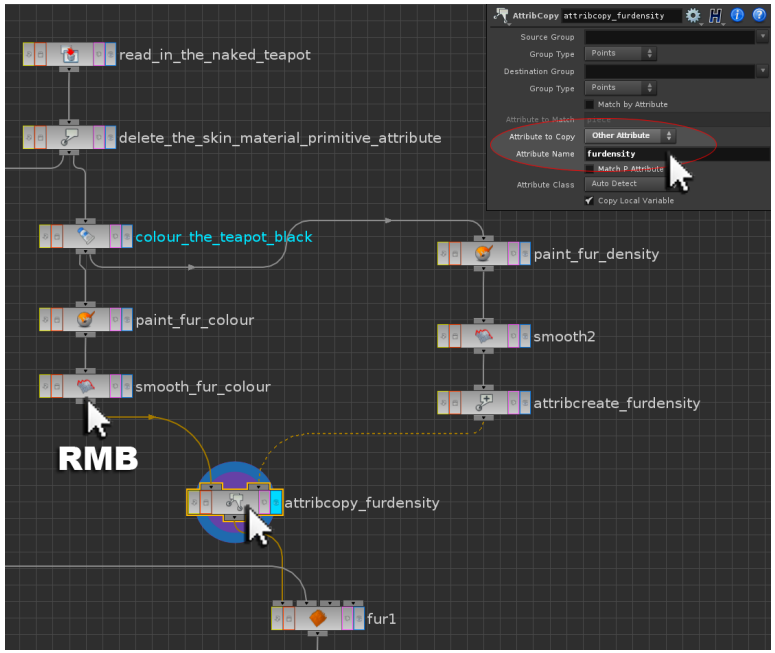
Name	**furdensity**
Value	**$CR**

As the colour for the fur density is monochrome, the colour values of each RGB channels will be the same. Any of the three colour channels variables ($CR, $CG, $CB) could be set as this attribute value. Alternatively the expression ($CR + $CG + $CB) / 3 could be used instead.

COPYING ATTRIBUTES

Currently this new network branch stands alone from the main network. Attributes can however be copied back into the main network, if there are matching topologies between the geometries being copied from and to. In cases where differing topologies occur, a proximity based **Attribute Transfer SOP** can be used instead.

RMB on the output of the fur colour **Smooth SOP** to insert an **Attribute Copy SOP**. Wire the **output** of the **fur density Attribute Create SOP** as its **second input**.

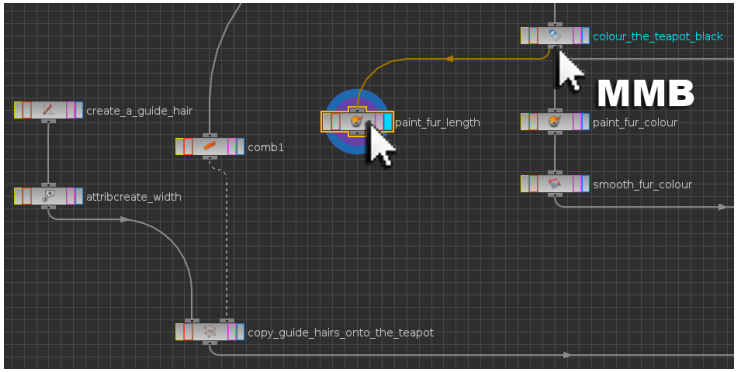


In the **parameters** for the **Attribute Copy SOP** specify:

Attributes to Copy	**Other Attribute**
Attribute Name	**furdensity**

When the scene is rendered again, the areas of the teapot with no fur density assigned are now bald.

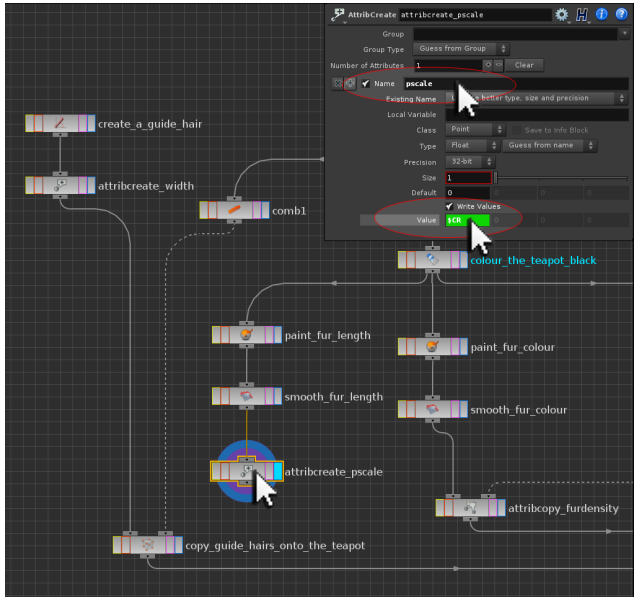
This principle of attribute copying can also be done to further modify the fur length. As the Fur SOP expects fur length to be controlled by guide hairs; there is no formal furlength attribute that can be overridden. Nor is it directly possible to create a length attribute to override the Fur SOP's length parameter. Instead a **pscale** attribute can be created, and used to **modify the length** of the copied guide hairs before the fur is assigned.



MMB on the output of the **Color SOP** colouring the teapot black to create a **Paint SOP** as a new network branch.

NOTE: Paint colour values of above 1,1,1 can also be used to create extra length to the hair. For example a paint colour value of 2,2,2 will double the length of the hair.

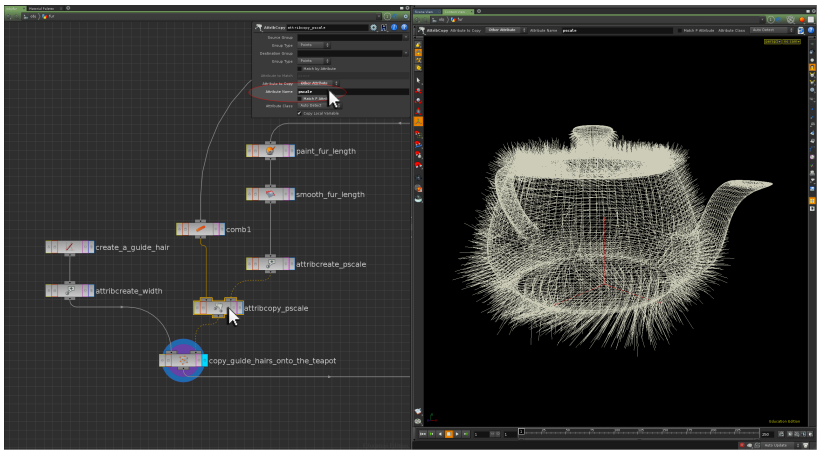
Append a **Smooth SOP** to this Paint SOP, to smooth out the fur length paintwork. Make sure its **Apply to** parameter is set to **Color** rather than **Point Position**.



Append to the Smooth SOP an **Attribute Create SOP**. In its **parameters** specify:

Name	**pscale**			
Value	**$CR**	**0**	**0**	**0**

This pscale attribute can now be copied into the Comb SOP network branch.



When the Display/Render Flag is set to the Copy SOP, the effect of the pscale attribute can be seen on the copied guide hairs. This customisation will in turn be passed into the Fur SOP. **See file naked_teapot_end.hipnc**

THE POWER OF THE SHELVES

This entire exercise can be quickly recreated using the **Fur Shelf**. This Shelf contains a series of **predefined tools**, all designed to make the process of generating fur systems much **simpler** and **faster** than generating bespoke networks.

Generating a **bespoke network** by placing individual nodes can be seen philosophically as a **bottom-up** approach. Generating **automated networks** using the Shelves can be seen philosophically as the opposite **top-down** approach. **Understanding both bottom-up and top-down workflows** is key to unlocking the **power** and **future** of Houdini.

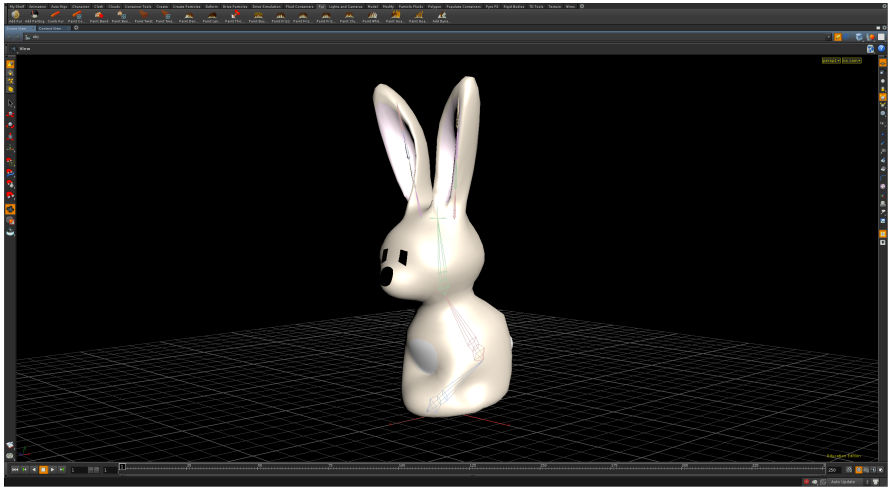
While initially the Shelf based top-down approach may seem rapid and efficient; it too can be laborious after the initial automatic networks have been generated. When refining the aesthetics of an automated setup for example, the resulting nodes still have to be explored and refined. This can take a great deal of time, and it is therefore vital that the bottom-up knowledge of node creation is also known and understood to make navigating the automated networks as simple as possible.

The Shelves are however the future of Houdini, as slowly and surely more low level operators are being wrapped up into higher-level tools and objects. In time, the Shelf tools will also start being combined into their own higher-level systems.

Exploration of the Shelves should therefore only really begin when a Houdini artist has a good foundation of the bottom-up node creation process. This will ensure full understanding of what the Shelves are doing and the automated networks they create.

THE FUR SHELF

Open the scene **bunny_ears_fur_begin.hipnc**. This scene file contains a simple rigged bunny with simple dynamic ears.

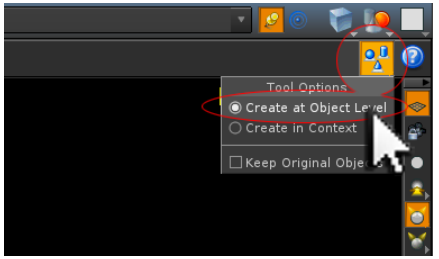


Maximise the **Scene Viewer**, and **activate** the **Fur Shelf**.

SCENE VIEW VS CONTEXT VIEW

For bottom-up node create, the **Context View** is great as it automatically follows the Network Editor around, allowing for the internal refinement of objects. **When working directly with the Shelves**, it is better to use the **Scene View**. This is simply because the Shelves are primarily **designed to work at Object Level**, and as Shelf tools are activated, the Scene View will follow their instruction accordingly.

Another aspect of Shelf based working to consider is a **setting** on the **Viewer Pane** itself, which allows for nodes to be **Created in Context** (inside an object) or **Created at Object Level** instead.



With the **Viewer** set to a **Scene View**, use the **Tool Options** button in the top right hand corner of the Viewer to specify that tools should be created at Object Level.

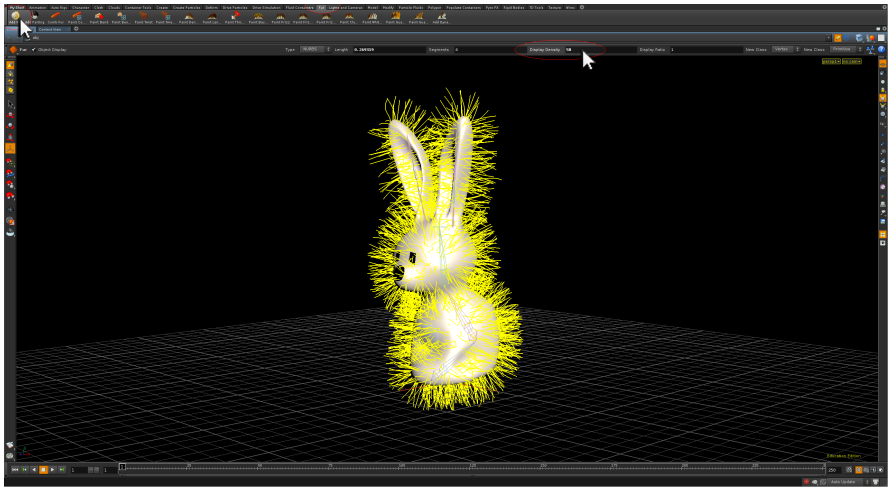
NOTE: A default Tool Options setting can be specified in the main Edit > Preferences > Objects and Geometry settings. This default cannot currently be set on a per Viewer basis, only globally.

VIEWER BASED WORKING

When working in a **Maximised Viewer**, the following keyboard commands can be used:

p	**will activate a floating Parameters Pane for the current tool**
i	**will go inside any selected object to its geometry level**
u	**will go from inside an object back up to object level**
Alt + g	**will activate a floating Material Palette**

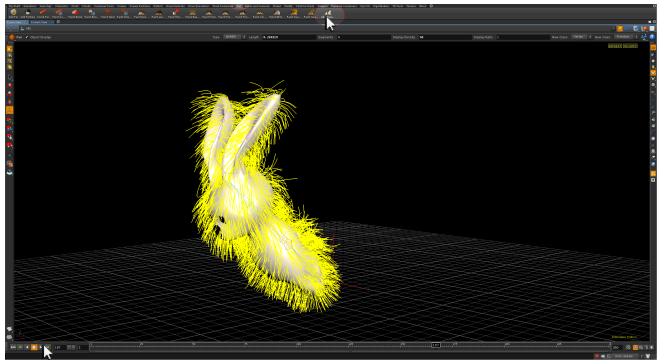
In the **Viewer**, **select** the **bunny** geometry and from the **Fur Shelf** select the **Add Fur** button.



From the Fur parameters list at the top of the Viewer, set the Display Density parameter to 50. This will keep the fur viewer display quite low for interactivity purposes; however will not affect the rendering density of the fur.

With the **fur** still **selected**, **activate** the **Add Dynamics button** of the **Fur Shelf**.

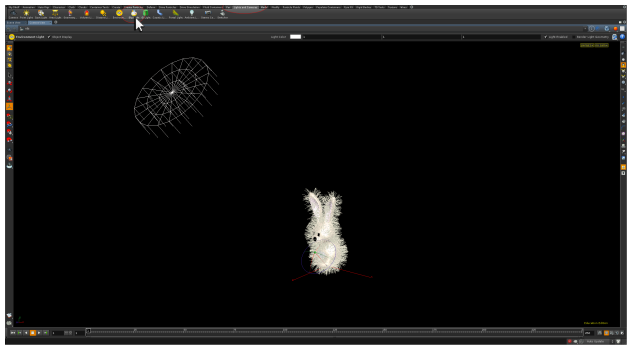
NOTE: The Scene View will automatically relocate itself to inside the fur object in case refinement of the guide hairs is required. Pressing u twice with the mouse over the Viewer will reset the View back to Object Level. Alternatively, the Viewer navigation bar can also be used to reset back to other Houdini levels.



When **PLAY** is pressed, the fur flops around in accordance to the animation of the bunny.

REFINING THE FUR AESTHETIC

Activate the **Lights and Cameras Shelf**, and **press** the **Sky Light** button. This will add a **preconfigured Environment Light** and **directional Sun Light** to the scene. A Mantra PBR node is already present in the scene for rendering purposes.



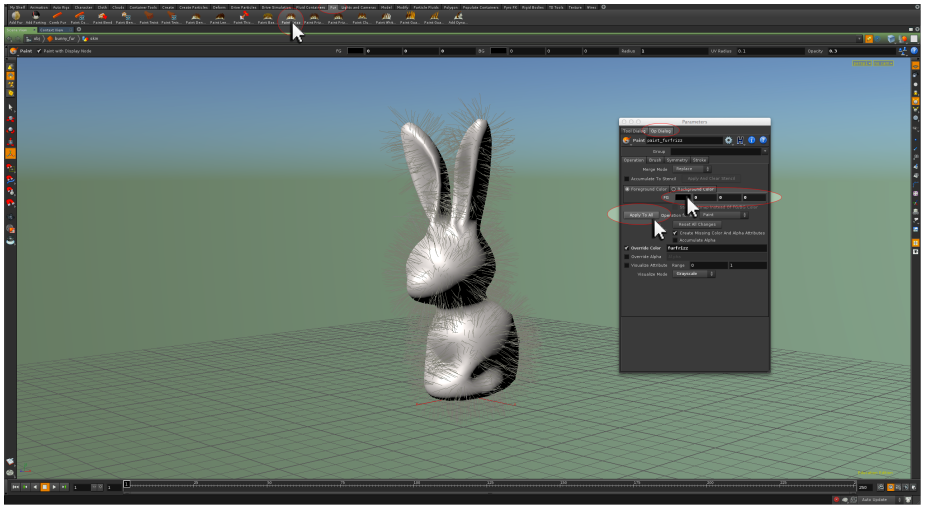
A **Render Region Preview** reveals the initial aesthetic of the fur. Make sure the **Normal Lighting mode** is **activated** from the right hand side stow bar of the Viewer to see the effect of the lit fur.

The **Fur Shelf** can now be returned to, and **tools** such as **Paint Density**, **Paint Length** and **Paint Frizz** can be played with. These tools will automatically create and configure the attributes shown in the earlier example.

The **fur geometry** can also be **re-selected** to get access to the **primary Fur parameters** such as global length.

Whenever using these Shelf Tools, the **timeline** should be set to **Frame 1**. Alternatively, the Edit > Objects > IK Rest option can be activated to temporarily remove animation.

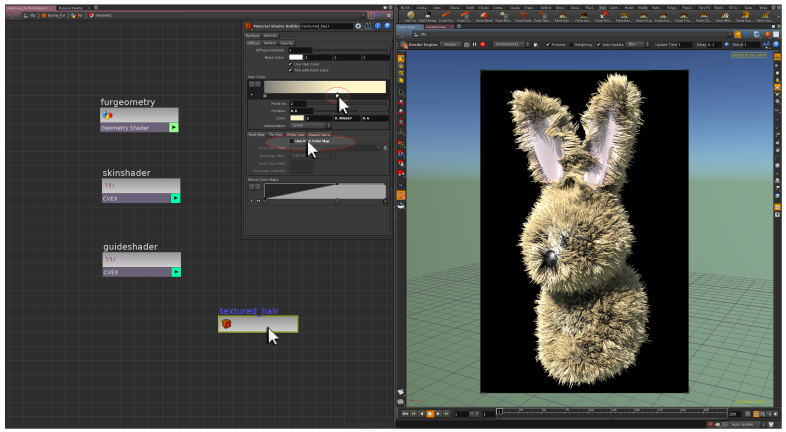
If for example, no frizz is required for the fur, the **Paint Frizz tool** can be activated, and the surface geometry painted as black.



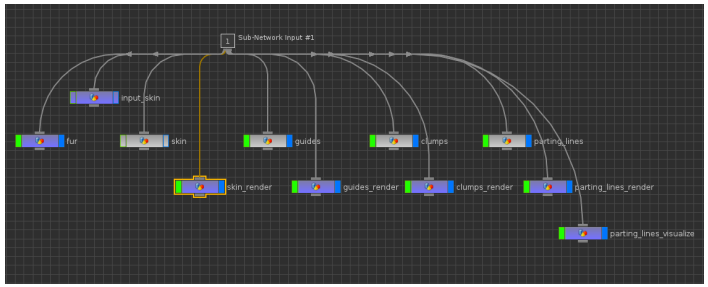
With the **Paint Frizz tool activated**, a **floating Parameters Pane** can be activated by pressing **p** on the **keyboard** with the **mouse over the Viewer**. The **Op Dialog** section of the floating **Parameters Pane** will give the standard parameters list for the **Paint SOP**, where the **Apply to All** button can be activated if required to completely remove the frizz.

NOTE: All of the automated fur Paint Tools have a **default Opacity setting** located in the **parameters** list at the top of the **Viewer**. This value should be set to **1** if absolute colour values for paintwork are required. **RMB** on the **Viewer** whilst in **Paint Tool mode** will reveal options for **smoothing painted colour**.

The Fur Material located inside the Fur Object can also be modified to remove the leopard skin texture; allowing for the geometry colour to influence the render.



It is also worth exploring the fur setup inside the Fur Object to better understand the automated nodes created, and as a comparison to the handmade network.



See file bunny_ears_fur_end.hipnc

CREATING CUSTOM SHELF TOOLS – SOME THEORY

Once networks have been established, they can become Shelf Tools in their own right. This is done by dragging and dropping a group of selected nodes directly onto a shelf. In the **Creation Desk UI**, the first shelf presented is called **My Shelf**, and is designed to store custom tools and networks.

In the case of the furry bunny network, the bunny nodes can either be **LMB dragged** directly onto **My_Shelf** as **individual nodes** or as a **single subnetwork node** by pressing **SHIFT + c** with the mouse over the Network Editor before attempting to place the nodes on the shelves.

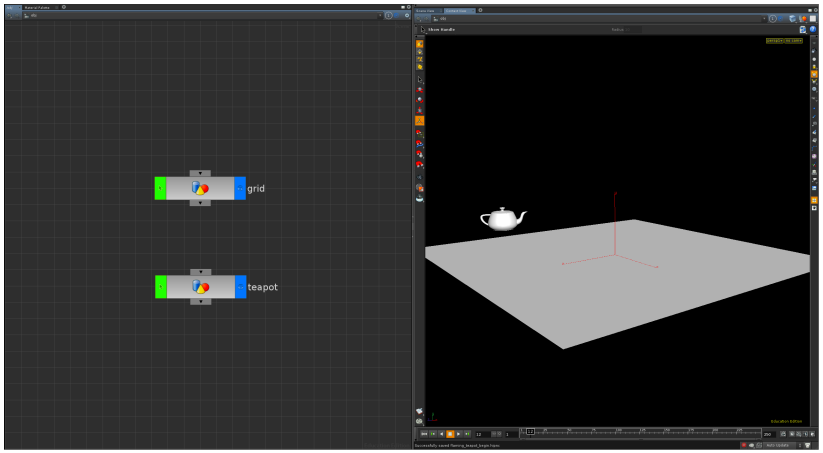
In either case, a shelf tool replicating this network will be created, and while initially everything looks good to go; creating a new instance of this network from the resulting Shelf Tool will result in errors being returned… The reason for this is that many of the furry bunny network nodes make absolute path references (IE: /obj/bunny) rather than relative path references (IE: ../../bunny). A new instance of these nodes would result in bunny1 being created instead.

When creating Shelf Tools it is important to test and debug them thoroughly. This is done by creating a new broken instance of a network using the new shelf tool, and then going through the nodes to find out what paths require adjusting. Similarly, some internal nodes may not work as expected due to their internal parameters (for example the Mirror SOP creating the dynamic ears curves would need to be replace in order to properly translate a new instance of the bunny around the scene. Currently the Mirror SOP makes a hard reference to the origin point of the scene for mirroring purposes).

ATTRIBUTE TRANSFERS

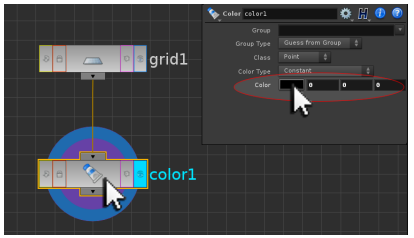
As well as copying geometry attributes, it is also possible to **transfer attributes**. As the name indicates this is a transfer process rather than a straight copying of data; meaning the **physical proximity** between **two pieces of geometry** is used to determine whether or not an attribute is passed on. The advantage of Attribute Transfers is simply that attributes can be transferred between two **completely different pieces of geometry**. A standard Attribute Copy by comparison requires identical geometry sources for the copying process.

When **Attribute Transfers** are also combined with the **Solver SOP**, a useful mechanism for the higher-level control of effects can be achieved. Open the scene **flaming_teapots_begin.hipnc**

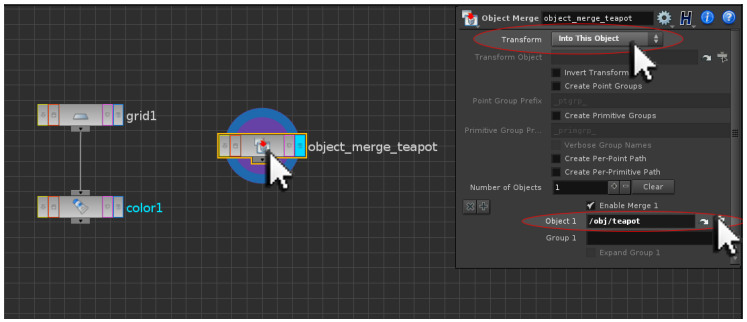


This scene contains a grid and a white teapot. When **PLAY** is pressed, the teapot slides over the grid.

Inside the **Grid Object**, append a **Color SOP** to the **Grid SOP**. Set the **colour** of the grid to **black**.



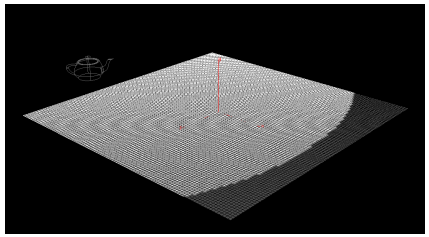
NOTE: The **Hidden Line Ghost** shading mode can be used to better see the black grid in the **Viewer**.



Alongside this network chain, create an **Object Merge SOP** to read in the animated teapot. In the **parameters** for the Object Merge SOP specify:

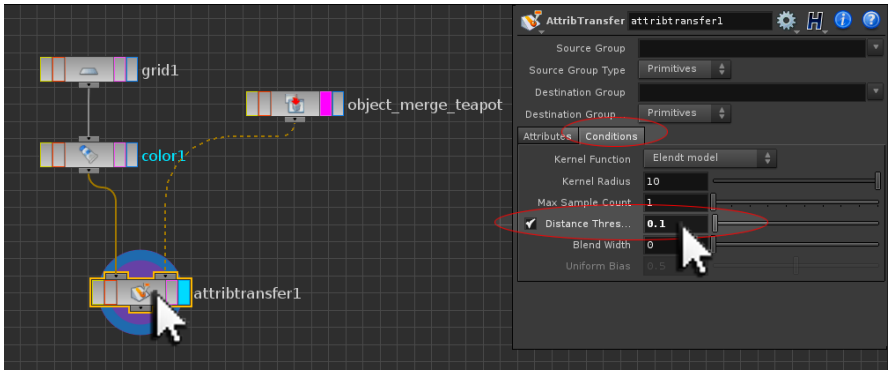
Transform	**Into This Object**
Object 1	**/obj/teapot**

As a new node, create an **Attribute Transfer SOP** wiring the **output** of the **Color SOP** as the **first input**, and the **output** of the **Object Merge SOP** as its **second input**. When the **Viewer** is examined, the white colour of the teapot is being proximity transferred onto the grid. The radius of transfer is however far too extreme.

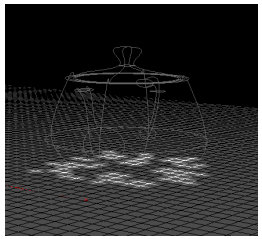


In the **Conditions** section of the **parameters** for the **Attribute Transfer SOP** specify:

Distance Threshold **0.1**

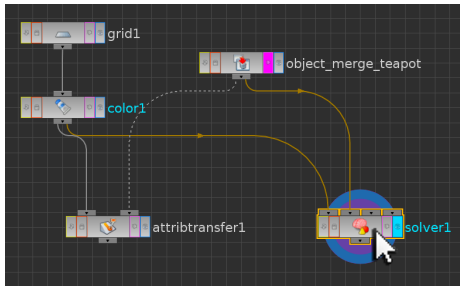


This will reduce the transfer radius, resulting in only points on the grid directly under the teapot geometry being coloured white.



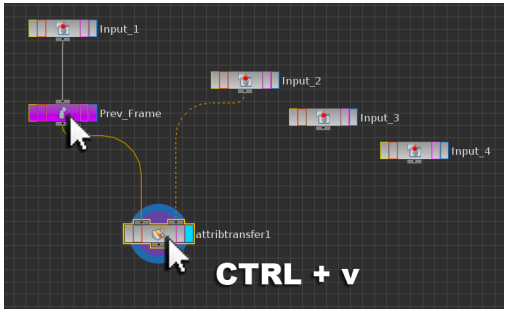
CREATING A STREAK OF COLOUR

In order to create a streak of white colour as the teapot slides over the grid, a **Solver SOP** can be used. This will **record** a **history** for the attribute transfer onto the grid.

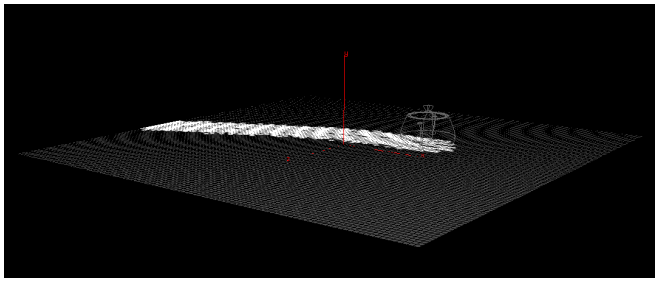


As a new node, create a **Solver SOP**; wiring the **output** of the **Color SOP** as the **first input**, and the **output** of the **Object Merge SOP** as its **second input**. The original Attribute Transfer SOP can be left as a comparison for this new network configuration.

Copy (**CTRL + c**) the original **Attribute Transfer SOP** into memory, and then head **inside** the **Solver SOP**. **Paste** (**CTRL + v**) the **Attribute Transfer SOP** into the Solver SOP network, wiring the **Input 1** and **Input 2** nodes respectively as the first and second inputs.



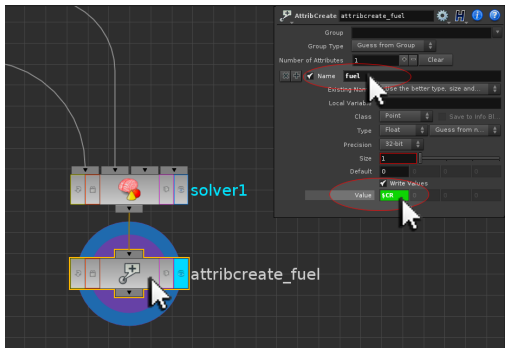
The **Prev_Frame node** (responsible for recording the history) can be inserted into the network by **dragging** and **dropping** it onto the **first input connection wire**.



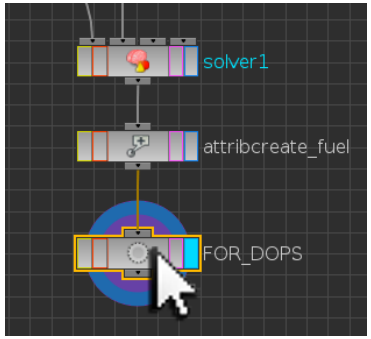
When the scene is examined again back outside of the Solver SOP, the effect of the trail can be seen. This trail can now be used to drive a simple fire effect.

To the **Solver SOP** append an **Attribute Create SOP**. In its **parameters** specify:

Name	**fuel**			
Value	**$CR**	**0**	**0**	**0**

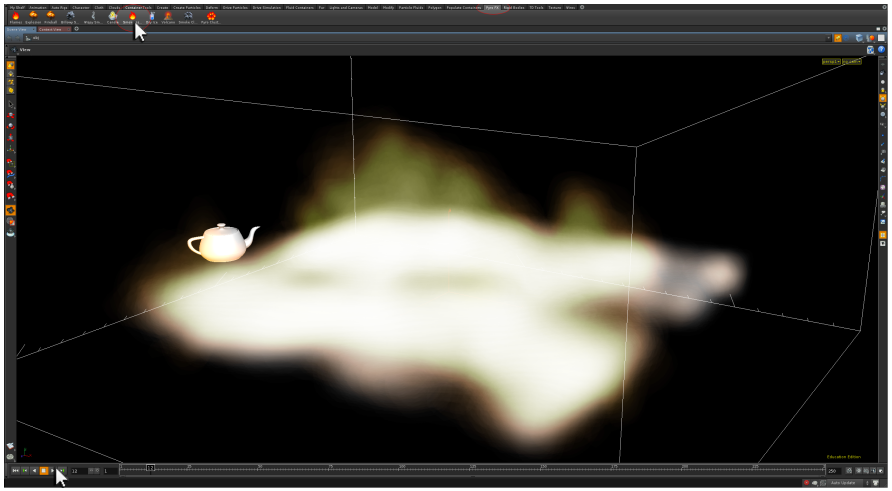


This will store the white coloured streak as a custom attribute anticipated by Houdini's Pyro Tools. As a final step, append a Null SOP called FOR_DOPS to the network. **See file flaming_teapot_stage1.hipnc**



CREATING A FIRE TRAIL

Return back to **Object Level**, and with the **grid selected maximise** the **Viewer** and **expose** the **Pyro FX Shelf**.



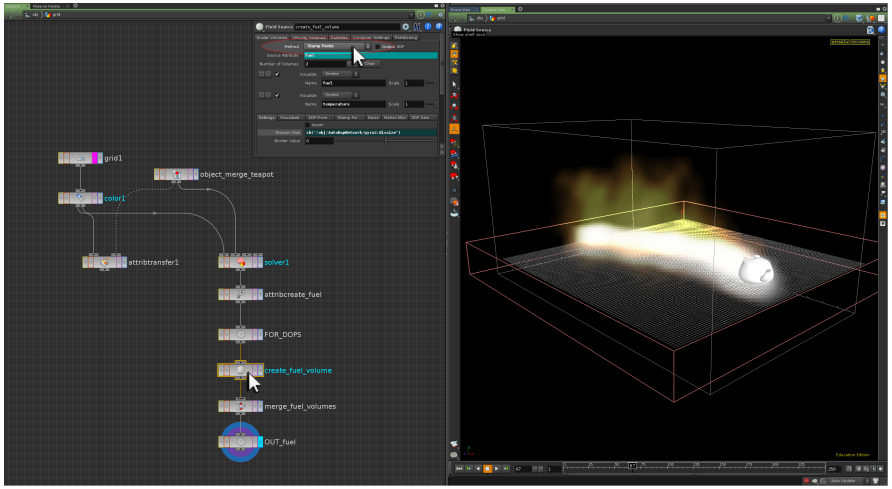
From the **Pyro FX Shelf** activate the **Smokeless Flame** button. When **PLAY** is pressed at **Object Level** a blanket flame effect can be seen across the entire surface of the grid.

NOTE: The Viewer must be set to a Smooth Shaded mode in order to see the flame effect in the Viewer.

In order to get the fire only appearing on the white trail, one of the automated nodes created by the Pyro Tools need to be modified.

Head back inside the **Grid Object**, and locate the **create_fuel_volume node**. Under the **Scalar Volumes** section of the **parameters** specify:

Method **Stamp Points**



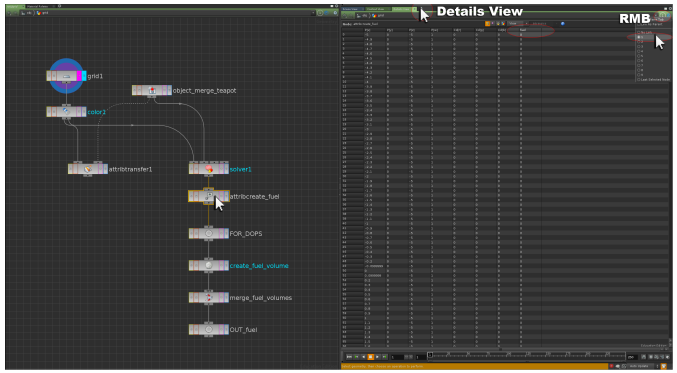
Now when **PLAY** is pressed, the fire effect only occurs from the white trailing streak. As a final step, the **Display / Render flag** can be reset to the original **Grid SOP**, so that the grid will render.

NOTE: The fire effect itself is being created in the AutoDopNetwork node created by the Pyro Tools, with the import_pyro_build object responsible for displaying it at Object Leve.

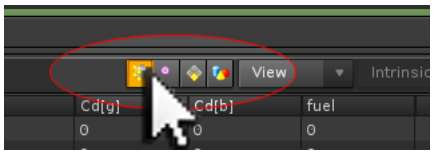
See file flaming_teapot_end.hipnc

ATTRIBUTE UTILITIES - The Details View

When working with attributes, it can be useful to see the numeric values created and stored within a custom attribute. This can be done using a **Details View**, which can be loaded as a **New Pane Tab Type alongside** the **Scene** and **Context Viewers**.



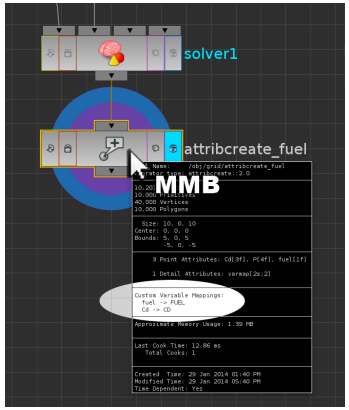
Setting the **Pane Number** of the Details View to **1** will link it to the **Network Editor**, meaning that any selected node's attributes will appear in the list.



The different classifications of attributes (Point, Primitive, Vertex and Detail) can also be switched between by choosing the appropriate attribute display button at the top of the Details View.

Attribute Mappings

When an attribute is created, the name of the attribute is also mapped as a $ATTRIBUTENAME variable. For example the custom **fuel attribute** created can also be called using **$FUEL**.



Verification of an attribute mapping can be done by **MMB** on the node to see the **Information Card**. The **Custom Variable Mappings** are listed after the main attributes list. Theoretically $ATTRIBUTENAME should be available to all nodes created downstream of the attribute declaration. Sometimes however downstream nodes do not recognise custom mappings. If this occurs, then the **point()**, **prim()**, **vertex()** and **detail() expression functions** can be used to retrieve attribute information from upstream nodes.

IMPORTANT NOTE: Always Delete Excess Attributes before production rendering takes place. This will save processing time, and can be done using the Attribute SOP.