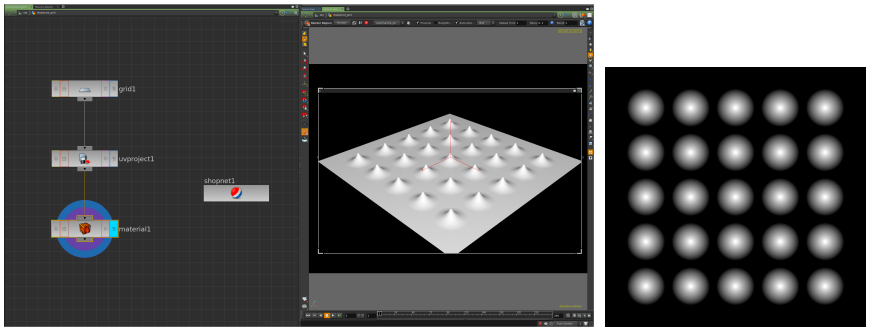
BAKING DISPLACEMENT ONTO GEOMETRY

Sometimes it is useful to visualise displacement maps as actual geometry. This can aid animation or Rigid Body Dynamics for example. Open the scene **displaceVOPSOP_begin.hipnc**.



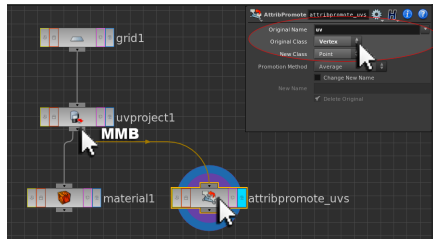
This scene itself contains a **Grid SOP** with a **Basic Displacement Material** assigned to it. When rendered the grid is displaced with an array of spikes. This **displacement** is a result of a **texture map**. The **UV Project SOP** is assigning the **uv** texturing co-ordinates as **Vertices**.

PROMOTING ATTRIBUTES

Before a geometry based displacement effect can be activated, **uv texturing** information must be **converted** to a **Point Attribute** rather than a **Vertices Attribute**. **An Attribute Promote SOP** can achieve this. **MMB** on the **output** of the **UV Project SOP** to **append** an **Attribute Create SOP** as a new network branch.

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

Original Name	**uv**
Original Class	**Vertex**
New Class	**Point**



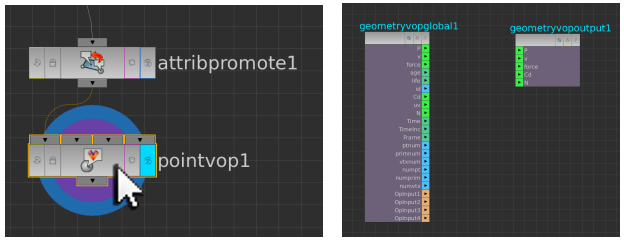
This promotion of the uv attribute can be confirmed by **MMB** on the **Attribute Promote SOP** to bring up the **node information card**.

THE POINT VOP SOP

A **Point VOP SOP** is a **special type** of **SOP** that contains an **internal VOP Network configured** for **generating custom SOP Level operators**. In this example, a **Point VOP SOP** can be configured to **bake** the **displacement** map onto the grid.

NOTE: As well as a **Point VOP SOP**, **other classes** of **custom operators** can also be **generated** using a **Primitive VOP SOP**, a **Vertex VOP SOP**, an **Attribute VOP SOP**, and a **Volume VOP SOP**. Equivalent custom operator nodes can also be found at every level of Houdini (for example a **POP VOP DOP** can be found at **DOP Level** for **creating custom particle operators**).

RMB append a **Point VOP SOP** to the output of the **Attribute Promote SOP**.

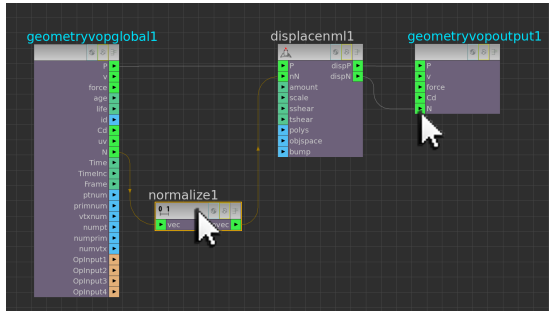


With the **VOP SOP selected**, press **ENTER** on the keyboard to go inside it. Here can be found a **Global Variables VOP** and an **Output VOP** specifically for inputting and outputting geometry data. Any VOP operators wired between these operators will modify the geometry this operator is assigned to.

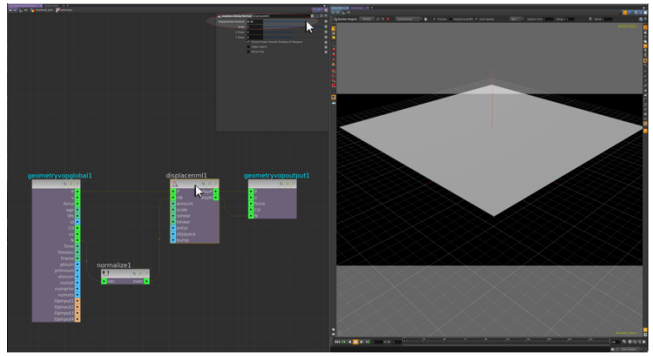
In VOPS **data types** are represented by **Input** and **Output colours** on the **VOP nodes**, with **bright green** for **Vectors**, **dark green** for **Floats** and **blue for Integers**. As a general rule, only matching data types can be wired together. **Hovering the mouse** over Input and Outputs colours will reveal the data they are passing.

BUILDING A CUSTOM TEXTURE DISPLACEMENT SOP

To displace the geometry, a **Displace Along Normal VOP** can be created between the **Global Variables VOP** and the **Output VOP**. The **Global Variables Ouput P** can be wired into the **Displace Along Normal P input**, and its **dispP output** can be wired into the **P input** of the **Output VOP**. This will take the incoming point data of the geometry, modify it, and then output it as point data back onto the original geometry.



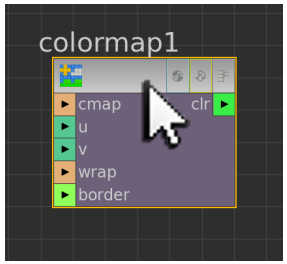
The **nN** input of **the Displace Along Normal VOP** denotes that any Surface Normal information passed through it should be **normalised** first to prevent errors. This can be done by **RMB appending** a **Normalize VOP** to the **Global Variables Output N** before it is **wired** into the **nN input** of **Displace Along Normal VOP**. The **dispN output** can then be wired into the **N** input of the **Output VOP**.

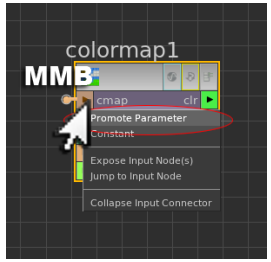


Adjusting the **Displacement Amount parameter** of the Displace Along Normal VOP now causes the grid to rise uniformly.

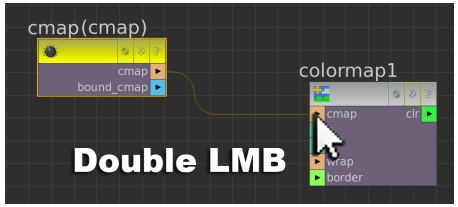
CREATING TEXTURE DISPLACEMENT

With a Displacement mechanism activated, texture displacement itself can now be created. A **Color Map VOP** can be created to read in the displacement image. As it is a displacement map with no alpha channel, the **Signature Parameter** can be left set at its **default RGB value**. In order to **create** a **SOP Level parameter** for **specifying** a **displacement map** to assign to the geometry, a **Parameter VOP** can be **automatically created** to do this.





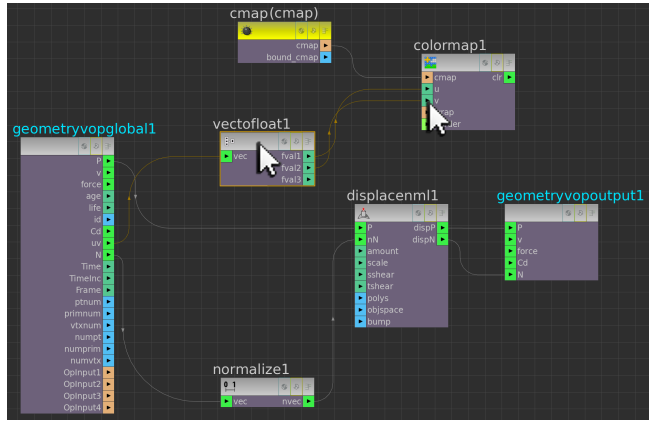
MMB on the **cmap input** of the **Color Map VOP** and choose **Promote Parameter**. This will create a **nodule input** denoting that a **hidden parameter VOP** correctly configured for this input **has been created**.



Double LMB on the **input nodule** can **reveal** the **Parameter VOP** if required.

CONVERTING VOP DATA TYPES

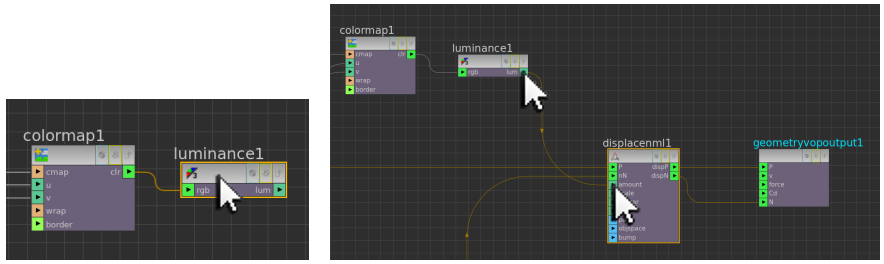
At present the **UV information** from the **input geometry** is being read in as a **vector** via the **Global Variables VOP**. The **Color Map VOP** has **two float attribute parameter inputs**, one called **u** and the other called **v**. In order to wire the **uv output** of the **Global Variables VOP** into these u and v inputs, it must be **converted** from a **vector** to a **series of floats**. **RMB** on the **uv output** of the **Global Variables VOP** and create a **Vector to Float VOP**.



Wire its first **two outputs** into the **u** and **v inputs** of the **Color Map VOP**. The third output of the Vector to Float VOP in this instance will return the w (depth) information of the UV coordinate system. As by default there is no depth specified to a uv coordinate system, this value will always return 0.

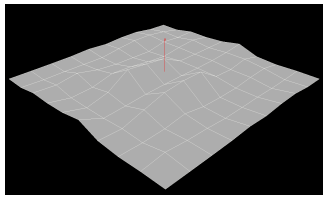
NOTE: A **full list of VOPs** responsible for **data conversion** can be located under the **Convert section** of the **TAB menu system**.

To the **output** of the **Color Map VOP** append a **Luminance VOP**. This VOP will automatically convert colour information from an image into a single greyscale output.



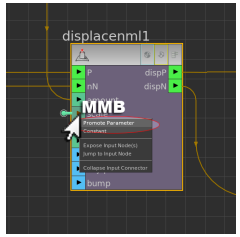
Wire the **output** of the Luminance VOP into the **amount input** of the **Displace Along Normal VOP**.

The grid geometry visible in the Scene Viewer will now be displaced based upon the default image being read into Color Map VOP (Mandril.pic).



NOTE: If necessary the **Force Compile button** of the **VOP SOP** can be pressed to refresh the internal VOP Network as changes are made.

As a final step, a **SOP Level parameter** can be created to control the **scale** of **displacement affecting** the geometry.

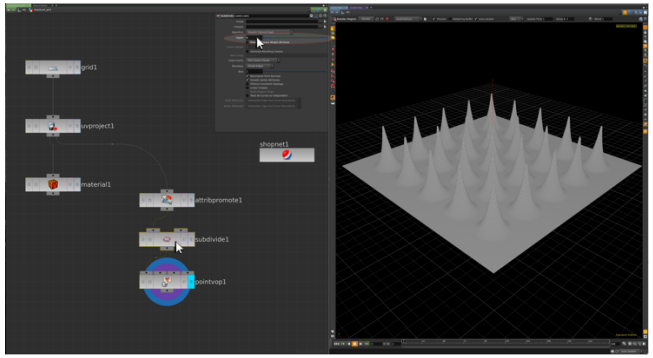


MMB on the **scale input** of the **Displace Along Normal VOP** to automatically create a **preconfigured Parameter VOP**. The scale of the displacement will now be adjustable at SOP Level. **See file displaceVOPSOP_stage1.hipnc**

CONFIGURING THE GEOMETRY DISPLACEMENT

With the VOP Network completed, return back up to **SOP Level** to **test** the **Point VOP SOP node**. The **default image** can be replaced by the **original displacement image** used by the Basic Displacement Material. The **scale of the displacement** can also be **increased** and **decreased**. There is however currently there is a lack of resolution to the assigned displacement due to a lack of geometry resolution on the grid being displaced. This can be rectified by **inserting** a **Subdivide SOP** before the **VOP SOP** node creating the geometry displacement. In the **parameters** for the **Subdivide SOP** specify:

Depth **5**

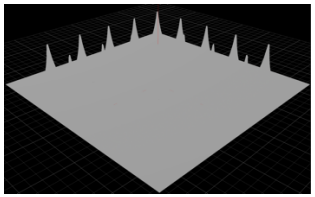


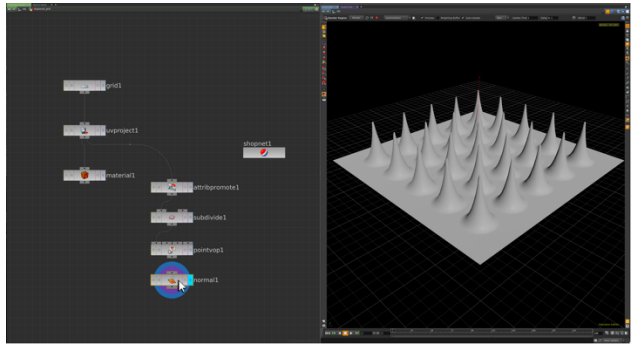
This will provide a suitably high enough division to geometry being displaced.

NOTE: For more complex geometry shapes requiring geometry displacement, the **second input** of the **Subdivide SOP** can also be used to **retain the original geometry shape** (by activating the **Crease Weight attribute** and associated parameter).

RE-COMPUTING THE NORMALS

Viewing the grid in **Smooth Shaded** mode reveals that although the displacement is taking place, the surface normals of the grid are not correct.

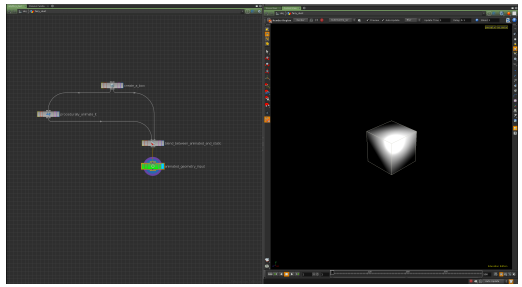




This can be rectified by **appending** a **Normal SOP** to the end of the network. This geometry displacement network can also be turned into a Digital Asset if required. **See file displaceVOPSOP_end.hipnc.**

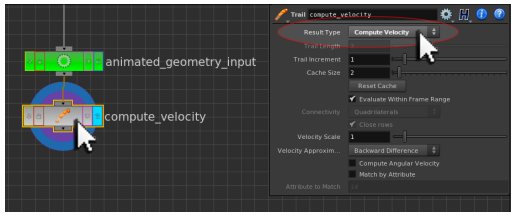
CREATING A CHOP INFLUENCED PARTICLE SYSTEM

Open the scene **fairy_dust_begin.hip**. This scene contains an animated box that stops and starts. This box will be used as the test input geometry for a particle system that reacts only to movement.



THE TRAIL SOP

In order to determine when the box is moving and when it is not, its velocity can be calculated. This can be done automatically by utilising a **Trail SOP**.

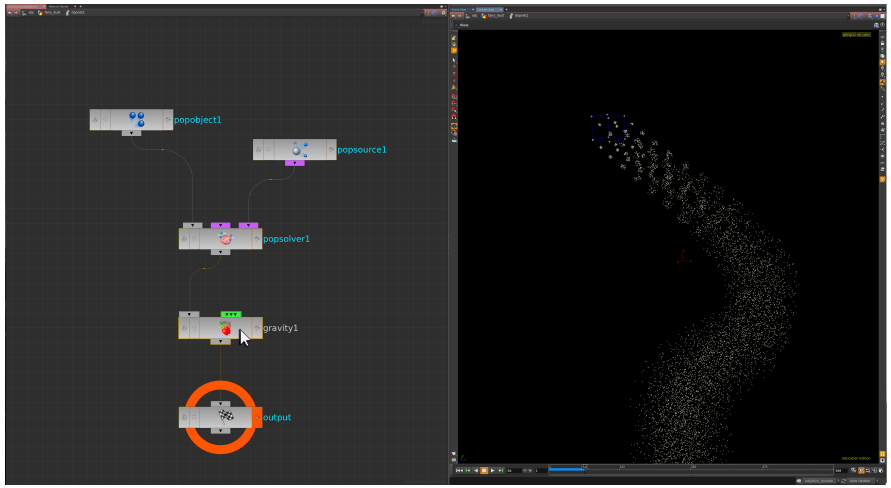


To **Null SOP**, append a **Trail SOP** and specifying in its **Parameters** for the **Result Type** to **Compute Velocity**. **MMB** on the **Trail SOP** node will display the added **Velocity Attribute**.

To the output of the **Trail SOP** append a **DOP Network**, and press **i** on the keyboard to enter into it. At **DOP Level**, create a **POP Source DOP** to read in the animated box as the source for the birth of the particles. In the **Parameters** for the **POP Source DOP** specify:

Source >	
Emission Type	**Points**
Geometry Source	**Use First Context Geometry**
Attributes >	
Initial Velocity	**Set Initial Velocity**

Complete the particle network setup by creating a **POP Object DOP** with a **POP Solver DOP** appended to it. Wire the **output** of the **POP Source DOP** as the **third input** of the POP Solver DOP.

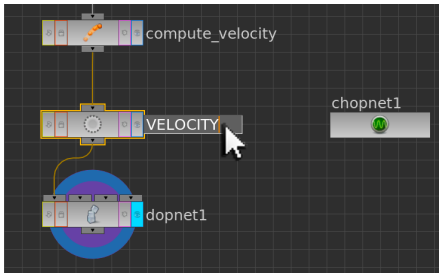


Append a **Gravity Force DOP** to the **output** of the **POP Solver DOP** and **wire** it **into** the **Output DOP**. This will create a simple particle system where the particles have some initial movement and fall under the influence of gravity.

When **PLAY** is pressed, a stream of particles emit continually from the box even when it is static. This is due to the **Const. Birth Rate** Parameter of the **POP Source DOP** being set to **5000**. This value can however be procedurally controlled by the computed velocity of the box.

IMPORTING THE VELOCITY DATA INTO CHOPS

Press **u** on the keyboard to jump back up to **SOP Level** and here create a **CHOP Network** (press **TAB** and type **chopnet**).



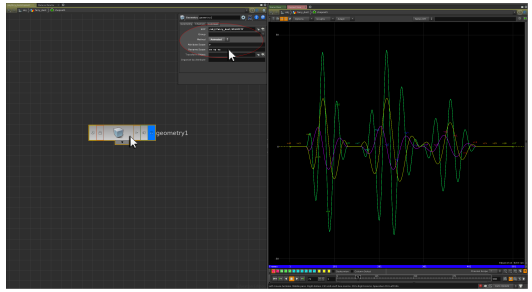
Insert a second **Null SOP** in-between the **Trail SOP** and the **DOP Network**. Rename this **Null SOP** to **VELOCITY**. This will be the reference point for importing the velocity data into CHOPs.

With the **CHOP Network selected**, press **i** on the keyboard to enter inside it. As with a DOP Network, data has to be initially imported into a CHOP Level in order to see it. As the Velocity Attribute of the animated box is a point attribute, a **Geometry CHOP** can be created to read this data in. In the **parameters** for the **Geometry CHOP** specify:

Geometry >

SOP	**/obj/fairy_dust/ VELOCITY**
Method	**Animated**
Attribute Scope	**v**
Rename Scope	**vx vy vz**

This will import only the Velocity Attribute for the animated box, separating out and renaming its components to vx vy and vz. This renaming is simply for clarity in the Viewer.



NOTE: Imported CHOP animation data is automatically resampled as per the **Sample Rate parameter** found under the **Channel section** of the **parameters**. A **low sample value** will result in jagged animation channels. A **higher sample value** will result in smoother animation channels. By default the Sample Rate parameter matches the Frames Per Second value specified in the **Global Animation Options**. If necessary this value can be increased for more accuracy.

MANIPULATING THE CHOP DATA

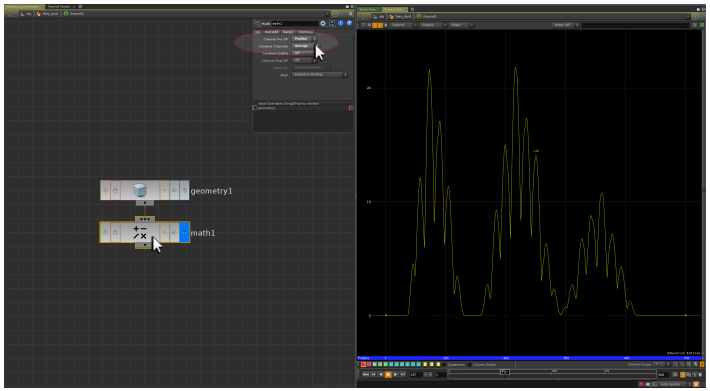
At present the Geometry CHOP is reading in the Velocity Attribute on a per point basis. This means that visually there is a graph line not only for each point of the incoming geometry data, but also a graph line for each x y and z component. This equates to a naming convention of:

vx0	**vy0**	**vz0**	
vx1	**vy1**	**vz1**	
vx2	**vy2**	**vz2**	etc.

As the **Const. Birth Rate Parameter** of the **POP Source DOP** will only accept a single positive floating-point number, these Velocity channels need to be combined together and converted to positive values.

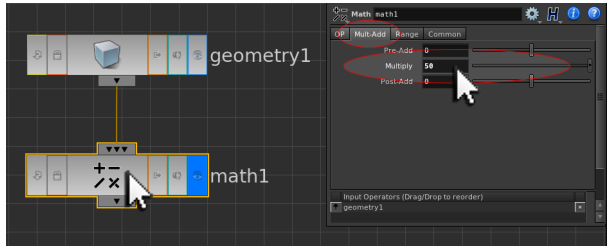
To the output of the **Geometry CHOP**, append a **Math CHOP** and in the **OP** section of its **Parameters** specify:

Channel Pre OP	**Positive**
Combine Channels	**Average**



This will make any negative values from the incoming data positive and combine the data channels into one by averaging them. The result is a single data channel ranging from **0** to **25**. While this channel is now in the correct format to be passed back into the DOP Network, the current number range will only allow a maximum of 25 particles per second to be born.

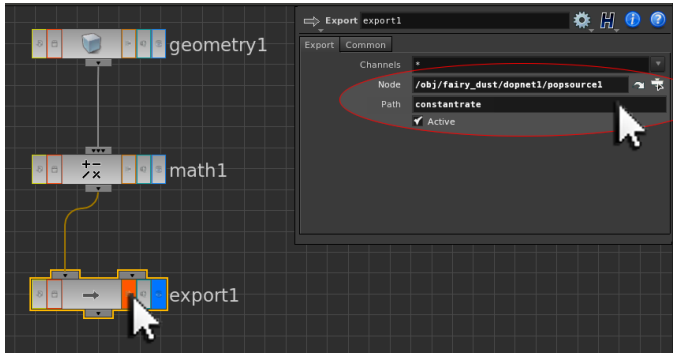
In order to give a visually pleasing result, this value can be multiplied up to a more appropriate range. This can be done in the **Mult-Add** section of the **Math CHOP**.



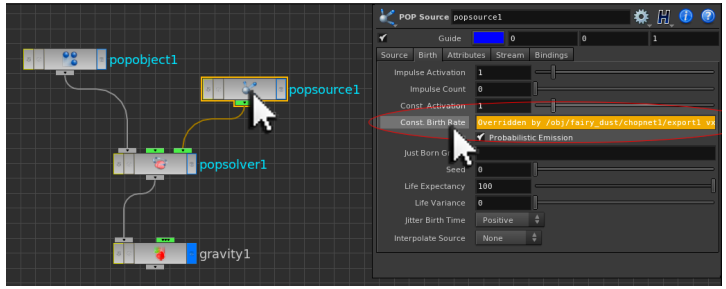
With the **Math CHOP** selected, go to the **Mult-Add** section of the **Parameters** and increase the **Multiply** Parameter to **50**. This will increase the number range from **0** to **1000**. Pressing **h** on the keyboard with the mouse over the Viewer will redraw the graph to fit.

This new data channel is now ready to be exported back into the Const. Birth Rate Parameter of the POP Source DOP. Appending an **Export CHOP** to the Math CHOP can do this. **Activate** both the **Blue Display** Flag and the **Orange Export** Flag.

In the **Parameters** for the **Export CHOP** set the **Node** Parameter to point to the **POP Source DOP** by using the **+** button to navigate to it. In the **Path** Parameter enter the word **constantrate**. This is the internal **Name** of the **Const. Birth Rate Parameter**.



With this done and the Orange Export Flag set on the Export CHOP, the data channel is now controlling the Const. Birth Rate Parameter. Navigating to the **POP Source DOP** and **examining** the **Const. Birth Rate parameter** can confirm this.

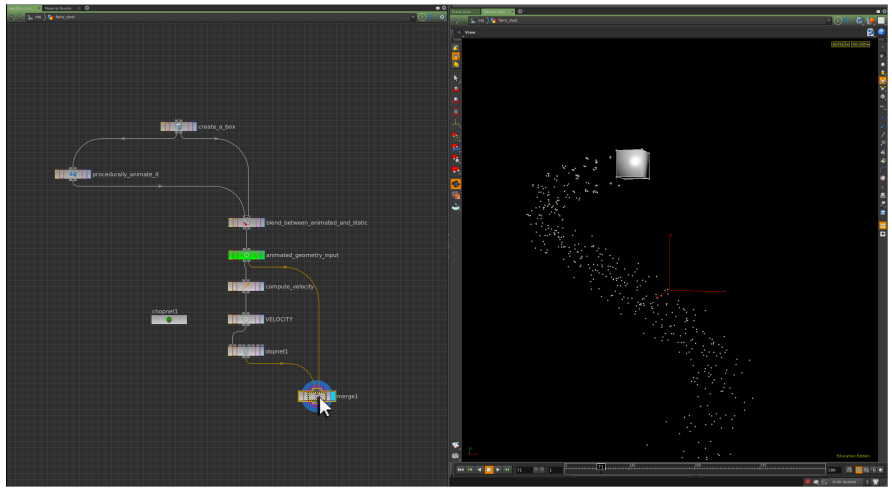


The value field of the **Const. Birth Rate Parameter** is now **orange**. When the Const. Birth Rate Parameter label is **LMB** clicked, the value is returned as:

Overridden by /obj/fairy_dust/chopnet1/export1

The visual end result is a series of particles that only emit when the input geometry is moving. The **Life Expectancy parameter** of the **POP Source DOP** can also be decreased from **100** to **5** for interactivity purposes.

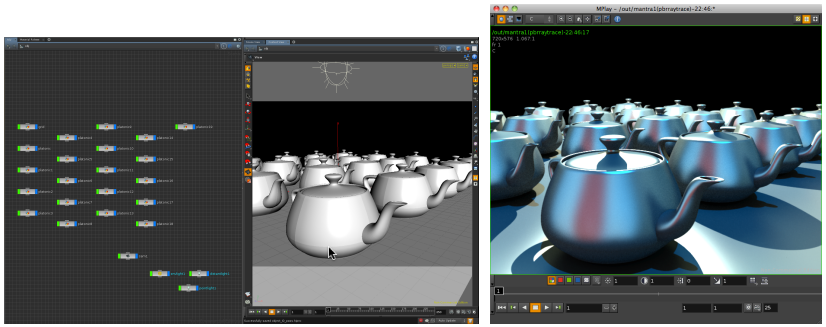
When **PLAY** is pressed more particles are born the faster the box travels. The advantage of this procedural method is that the animation geometry can be changed and the particles will update accordingly.



A **Merge SOP** can be used to see both the animated geometry and the particles simultaneously. See file **fairy_dust_end.hipnc**

OBJECT ID RENDER PASS

Another useful render pass for the composite is known as the ID pass. This is where a unique number is assigned to each scene object and stored as an image plane output from the main scene. Open the **scene object_ID_pass.hipnc**



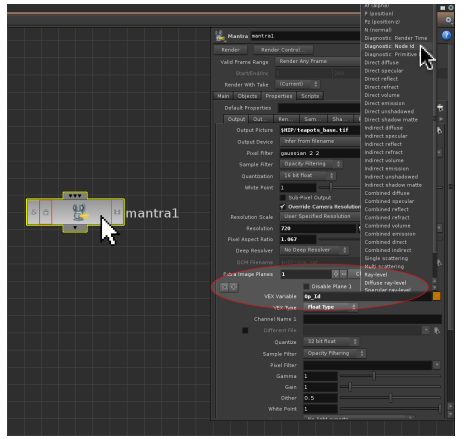
This scene contains a number of individual teapots on a grid. When the scene is rendered, the aesthetic of each teapot is identical.

An Object ID Pass can be used to isolate each individual teapot in the composite allowing for individual teapot compositing operations such as colour grading to be performed. Switch to the **Outputs Level** of Houdini, and examine the **Mantra ROP**. Currently, the render destination of the Mantra ROP is set to:

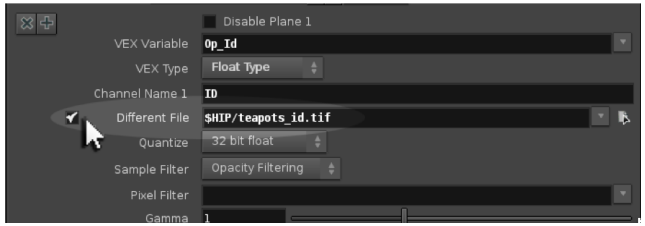
Output Picture **$HIP/teapots_base.tif**

While image planes can be assigned as layers in a .exr file, they can also be outputted as individual images in their own right. This behavior is determined by the file extension of the Output Picture destination.

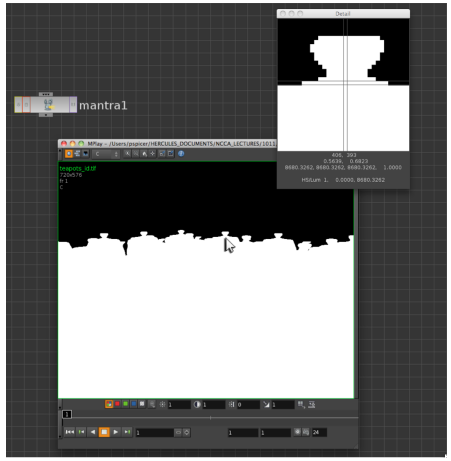
If the main render is created as a .exr file, all additional image planes will be automatically embedded into the .exr file. If however an alternate format such as .tif is used, image planes can also be set to render as separate .tif files in their own right.



Activate an **Extra Image Plane**, setting the **VEX Variable parameter** to **Diagnostic: Node id**. Specify a Channel Name of ID, and activate the **Different File** tick box and set a destination for the object id render.



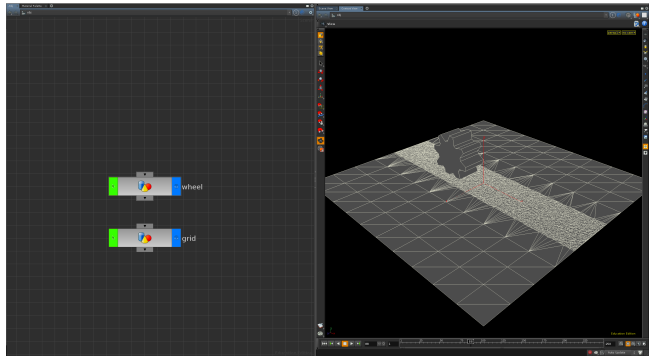
When the **Render button** is activated, both the **main render** and the **Object ID Pass** will be created on disc.



Examination of the Object ID Pass using MPlay reveals a white image akin to an alpha channel. Pressing **m** over the **MPlay** window will reveal a magnifier that can be used to identify the individual numbers assigned to individual teapots. The Object ID Pass can be used to colour grade each teapot individually in the composite.

CREATING A TYRE TREAD EFFECT

Open the scene **H15_tyre_tread_flat_terrain_begin.hipnc**.

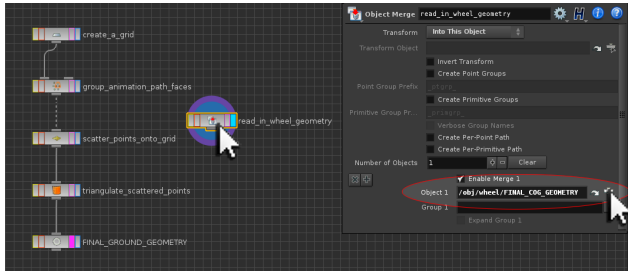


This scene contains a simple wheel animated across a grid. Currently the wheel position intersects with the grid as it rolls along. The grid has been traingulated along the path of the wheel to give sufficient geometry for tyre-tread deformations. This example will look at how the topology of the grid can be procedurally deformed in accordance to the movement and shape of the wheel.

Inside the **grid object**, create an **Object Merge SOP** alongside the main network. In its **parameters** specify:

Object 1	**/obj/wheel/FINAL_COG_GEOMETRY**
Transform	**Into This Object**

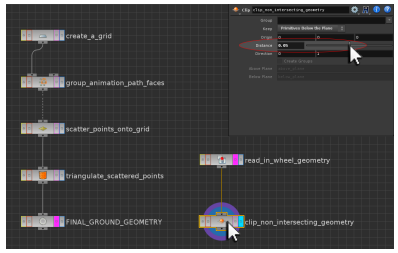
This will read in a duplicate of the wheel that be used to generate particles, which in turn will deform the grid.

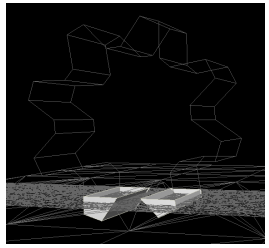


To the **output** of the **Object Merge SOP** append a **Clip SOP**. This will be used to delete any wheel geometry not directly intersecting with the grid. In its **parameters** specify:

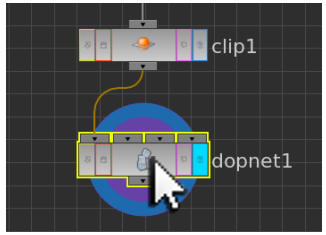
Keep	**Primitives Below the Plane**
Distance	**0.05**

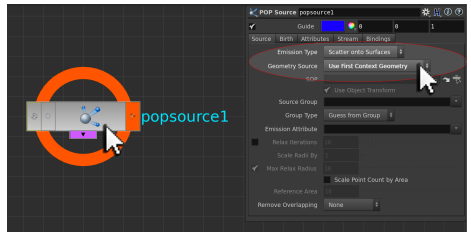
This will activate the clip slightly higher above the grid creating troughs from the wheel geometry.





When **PLAY** is pressed, only the areas of the wheel intersecting with the grid remain.





To the **Clip SOP** append a **DOP Network** and **Double LMB on it to go inside it.** At DOP Level, create a **POP Source DOP**. In its **parameters** specify:

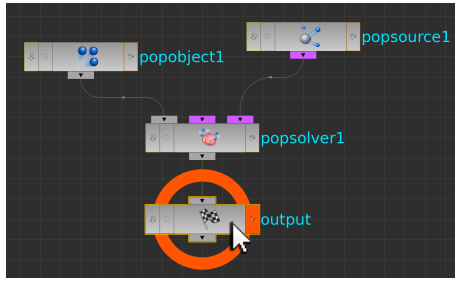
Source >

Emission Type	**Scatter onto Surfaces**
Geometry Source	**First Context Geometry**

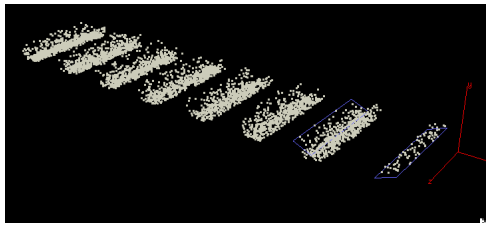
Birth >

Const. Birth Rate	**2000**

Complete the particle network setup by creating a **POP Object DOP** with a **POP Solver DOP** appended to it. Wire the **output** of the **POP Source DOP** as the **third input** of the POP Solver DOP. Wire the **POP Solver DOP** into the **Output DOP**.

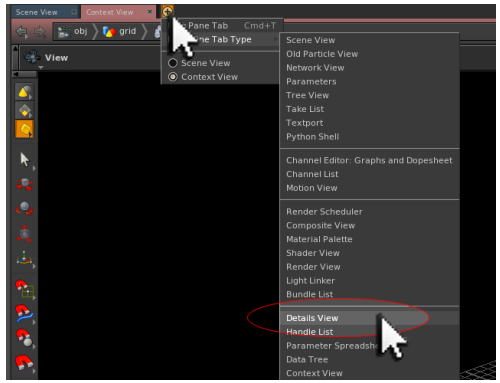


When **PLAY** is pressed, useful trough shapes of particles are generated whenever the clipped wheel geometry intersects with the grid.

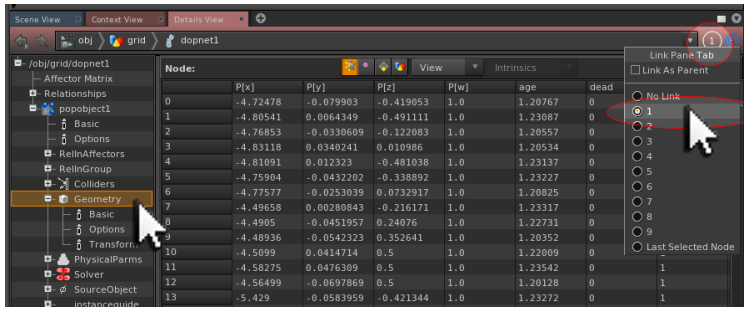


COLOURING PARTICLES

In order to create a displacement effect on the grid, these trough particles can be coloured relative to their distance from the grid. These values can be examined numerically by activating a **Details View** as a new **Pane Tab Type** over the Viewer. A **Details View** is a **spreadsheet** containing all information about a selected node.



Using the **Link Pane button**, activate the link value to **1**. This will point the **Details View** to return the current node of the **Network Editor** (also set to the link value of 1).



	P[x]	P[y]	P[z]	P[w]	age	dead
0	-4.72478	-0.079903	-0.419853	1.0	1.20767	0
1	-4.80541	0.0064349	-0.491111	1.0	1.23087	0
2	-4.76853	-0.0330609	-0.122083	1.0	1.20557	0
3	-4.83118	0.0340241	0.010986	1.0	1.20534	0
4	-4.81091	0.012323	-0.481038	1.0	1.23137	0
5	-4.75904	-0.0432202	-0.338892	1.0	1.23227	0
6	-4.77577	-0.0253039	0.0732917	1.0	1.20825	0
7	-4.49658	0.00280843	-0.216171	1.0	1.23317	0
8	-4.4905	-0.0451957	0.24076	1.0	1.22731	0
9	-4.48936	-0.0542323	0.352641	1.0	1.20352	0
10	-4.5099	0.0414714	0.5	1.0	1.22009	0
11	-4.58275	0.0476309	0.5	1.0	1.23542	0
12	-4.56499	-0.0697869	0.5	1.0	1.20128	0
13	-5.429	-0.0583959	-0.421344	1.0	1.23272	0

The **Details View list** contains information for each individual particle in the **Geometry section** of the **popobject1 listing**. This includes all of their attributes such as position (P), velocity (v) and Identity Number (id).

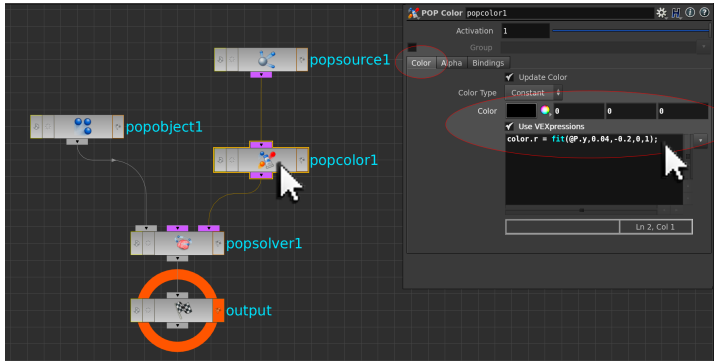
When the **P[y]** column is examined for the particle system, position values between approximately **0.04** (particles born just above the grid) and **-0.2** (particles born at the lowest point beneath the grid) can be seen. These values can be used to add colour to each of the particles.

To the output of the **POP Source DOP** append a **POP Color DOP**. In its **parameters** specify:

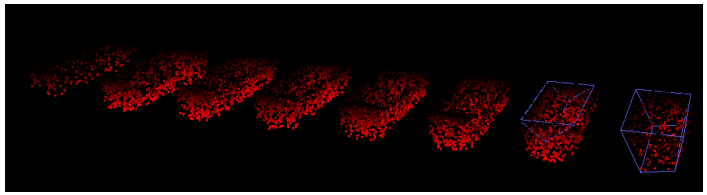
Color 0 0 0

☑ Use VEXpressions

color.r = fit(@P.y,0.04,-0.2,0,1);

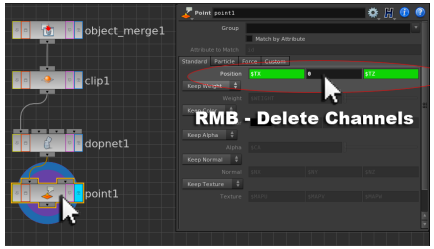


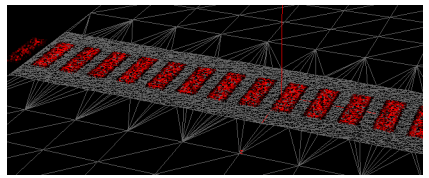
When **PLAY** is pressed, each particle is now coloured red based upon their Y distance from the grid. The further away the particle, more red is assigned.



Return back up to **Geometry Level** by pressing **u** on the keyboard, and append a **Point SOP** to the **DOP Network**.

In the **Standard > Position parameter** of the **Point SOP**, **RMB** on the **Position Y** parameter and choose **Delete Channels** from the resulting menu. This will flatten all the particles, aligning them with the grid.





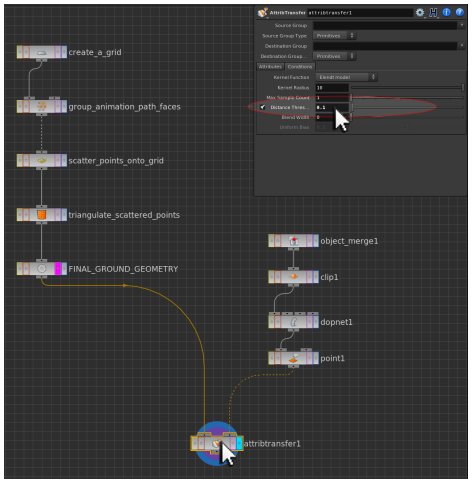
This colour information from the particles can be **transferred onto the grid** using an **Attribute Transfer SOP**. When attributes are transferred from one piece of geometry to another, this is done by a radius-based proximity between the first and second geometries. If the particles were left un-flattened, the bright red particles would not transfer onto the grid as much as the darker particles nearer the grid surface. By flattening the particles, all the particle colour information will transfer in the same way.

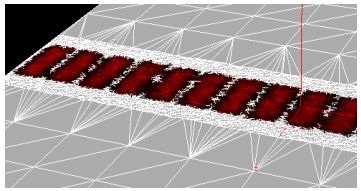
To the **output** of the **FINAL_GROUND_GEOMETRY Null SOP** append an **Attribute Transfer SOP**, wiring the **output** of the **Point SOP** flattening the particles as the second input.

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

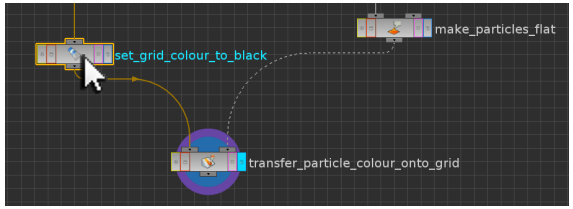
Attributes >
☑ **Points** **Cd**
Conditions >
☑ **Distance Threshold** **0.1**

This will transfer the colour information of the particles onto the grid geometry.

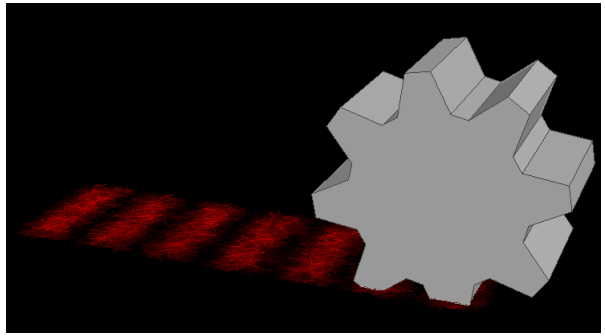




A **Color SOP** can also be **appended** to the **FINAL_GROUND_GEOMETRY Null SOP** to **set the colour of the grid to black**. This will ensure any surface deformation driven by colour will only utilize the red information.



When **PLAY** is pressed, 'heat' areas from the wheel animation are left on the grid geometry.

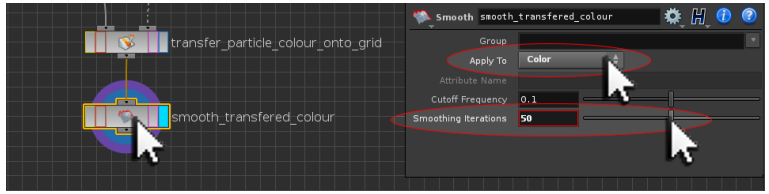


See file **H15_tyre_tread_flat_terrain_stage1.hipnc**

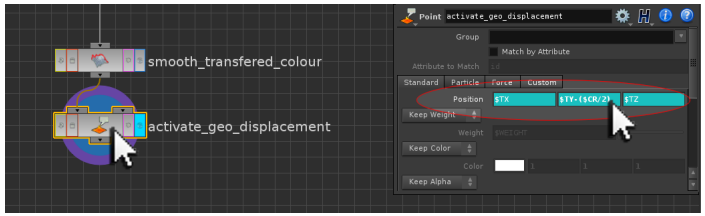
DEFORMING THE GRID

Before deformations take place it is a good idea to slighty blur the colour information in order to create a smoother deform. To the **output** of the **Attribute Transfer SOP**, append a **Smooth SOP**. In its **parameters** specify:

Apply To	**Color**
Smoothing Iterations	**50**



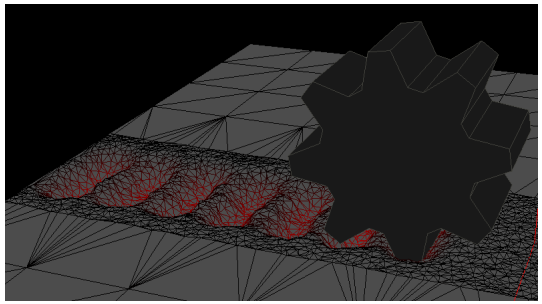
This will soften the transferred colour values, allowing for better deformations to occur. To the **output** of the **Smooth SOP** append a **Point SOP**. This can be used to deform the grid based upon its colour information.



In its **parameters** specify:

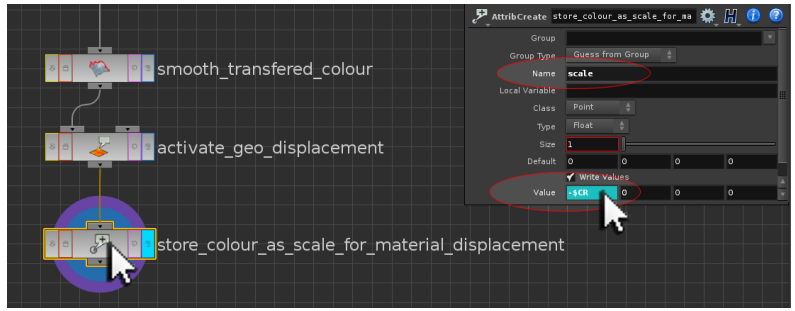
Position	**@P.x**	**@P.y-(@Cd.r/2)**	**@P.z**

When **PLAY** is pressed, the colour information from the particle system now deforms the grid. Setting the **Viewer** to **Hidden Line Ghost** display will better reveal the effect taking place.



USING COLOUR TO DRIVE MATERIAL DISPLACEMENT

This red colour information can also be used to procedurally drive material displacement. This will help add a little bit more naturalism to the deformation effect, where noise based displacement can be assigned to the red areas only. Again, the greater the value of red, the more noise displacement will occur.



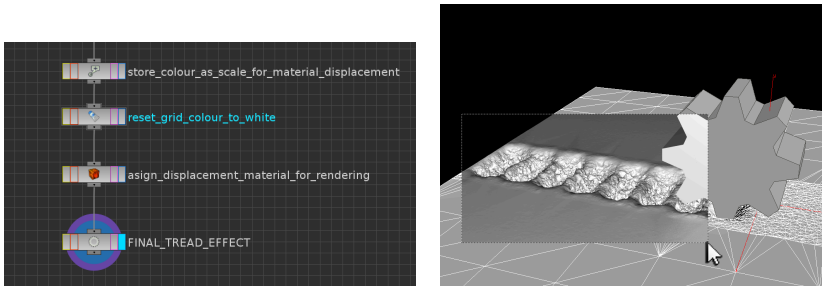
To the output of the **Point SOP** creating the grid deformations, append an **Attribute Create SOP**. In its **parameters** specify:

Name	**scale**			
Value	**-@Cd.r**	**0**	**0**	**0**

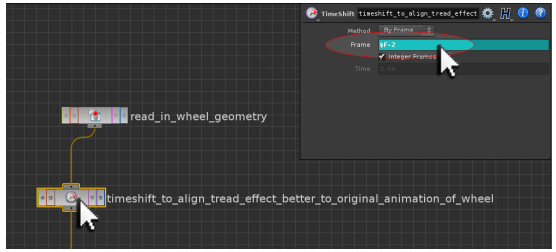
This attribute will automatically override the Scale parameter found on a Displacement Material (already configured at SHOP Level for this scene).

A **Color SOP** can then be appended to the network chain to reset the grid colour back to white. A **Material SOP** pointing to the **Displacement Material** located at **SHOP Level** can then also be assigned for rendering.

A **NULL SOP** can be added to denote the end of the tread effect network.



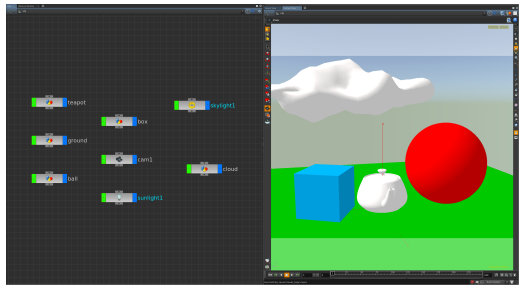
When a **Render Region Preview** is drawn over the Viewer, the effect of the noise displacement can be seen in the regions coloured red by the particle system. As a finishing step, the deformation effect can also be biased so that it aligns more tightly with the original wheel animation. This can be done using a **Time Shift SOP inserted** after the **Object Merge SOP** reading in the animated wheel.



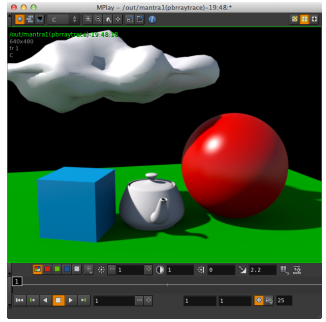
Specifying a **Frame parameter value** of **$F-2** will slightly nudge the deformation effect backwards in time, so that it appears more synchronized with the animated wheel. **See file H15_tyre_tread_flat_terrain_stage2.hipnc**

CLOUDS

Houdini now has a number of procedural tools for generating clouds. **Open the scene clouds_begin.hipnc**



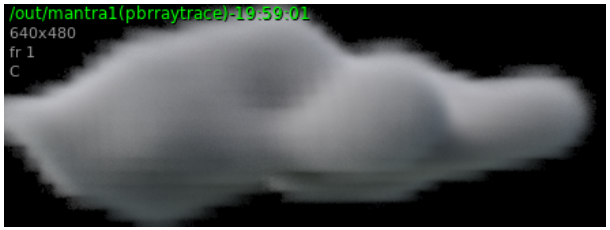
This scene contains a number of simple objects including a **mountain-ised polygon sphere** creating a **cloud**. A **Skylight** has also been added to the scene.



When the scene is rendered, a very basic aesthetic is generated. This example will look at generating the cloud as well as improving the overall render aesthetic.

Maximise the Viewer, and **select** the **cloud object**. From the **Cloud FX Shelf**, activate the **Cloud button**.

This will convert the **mountain-ised polygon sphere** into a **cloud volume**.

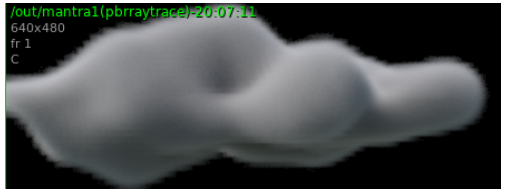


In the **cloud object's Geometry Level**, a **Cloud SOP** has been added to the network. In the **parameters** for the Cloud SOP specify:

Volume >

Uniform Sampling	**By Size**
Div Size	**0.05**

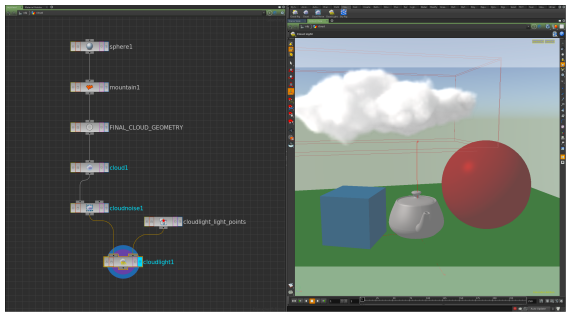
This will reduce the level of voxel-isation. When the scene is rendered again, the effect of the Cloud SOP refinement can be seen.

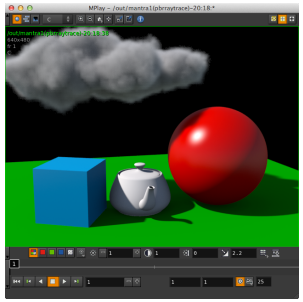


ADDING CLOUD NOISE AND LIGHTING INTERACTION

Return back to Object Level, and with the cloud object selected in the Viewer activate the **Cloud Noise button** and **Cloud Light button** from the **Cloud FX Shelf**.

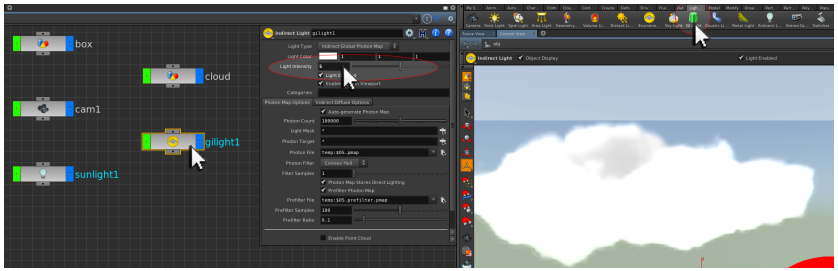
This will create fluffiness to the cloud as well as illuminate the cloud with diffuse lighting.





IMPROVING THE RENDER

Increasing the amount of Bounce Light affecting the scene, as well as increasing the amount of internal lighting to the cloud can improve the overall aesthetic of the scene. Whilst **Mantra PBR gives bounce light by default**, its value is fixed relative to physical phenomena. **It can however be overridden by** the addition of **a GI Light** to the scene.



From the **Lights and Camera Shelf**, activate a **GI Light**. In its **parameters**, specify a **Light Intensity** value of **5**.

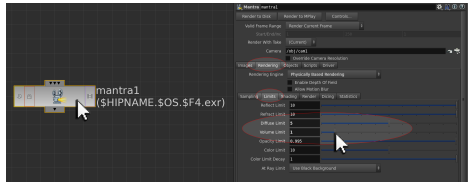
VOLUME LIMITS AND DIFFUSE LIMITS

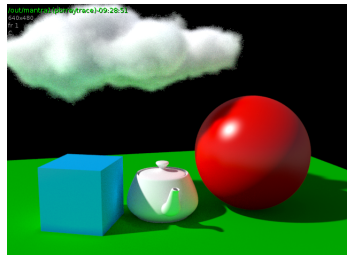
The effect of the **GI Light** and **Cloud Light SOP** needs to be formally activated in the render, by increasing the **Volume Limit** and **Diffuse Limit parameters** of the **Mantra PBR ROP**. These parameters control **how many times light** will either **bounce through a volume** (giving it internal illumination) or **bounce upon regular surfaces** (increasing the Bounce Light of the scene).

At **Outputs Level**, locate the **Mantra PBR ROP**, and in its **parameters** specify:

Rendering > Limits

Diffuse Limit	**5**
Volume Limit	**1**

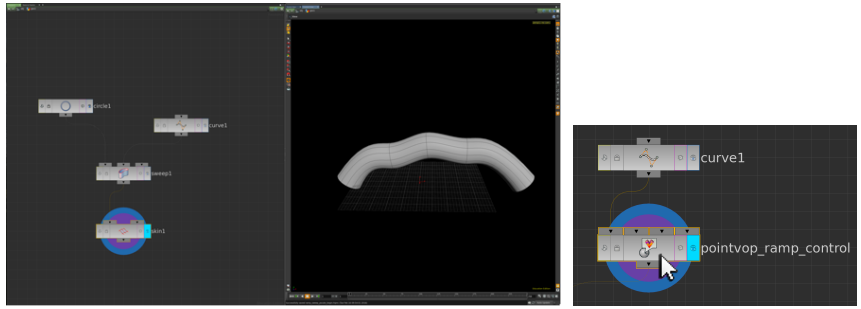




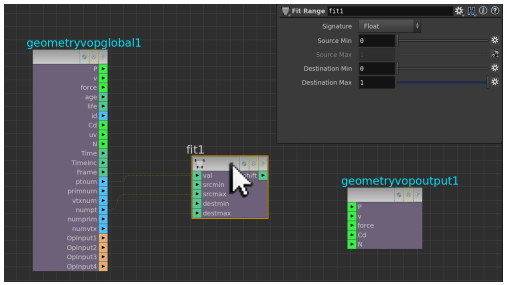
When the scene is rendered again, the effect of both bounce light and cloud lighting can be seen. A **Billowy Smoke Material** has also been assigned to the cloud and can be adjusted to refine the cloud aesthetic even further. **See file H15_clouds_end.hipnc**

RAMP CONTROL

Open the scene **ramp_sweep_pscale_begin.hipnc**. This scene file contains a curve swept with a NURBS Circle and then skinned. This example will look at activating a Ramp based control to adjust the radius of each swept circle. To the **output** of the **Curve SOP append** a **Point VOP SOP**, and **double LMB** on the node to **go inside it**.



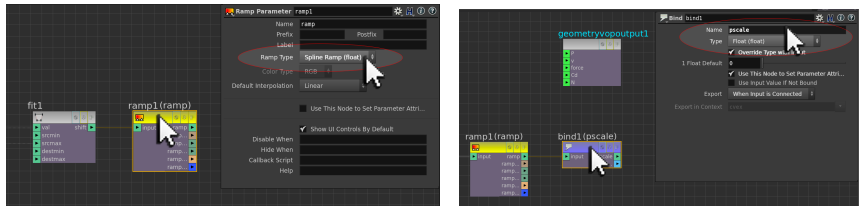
Create a **Fit VOP**, and wire the **ptnum output** of the **Global Variables VOP** into its **val (Value) input**, and the **numpt output** of the **Global Variables VOP** into its **scrmax (Source Mx) input**.



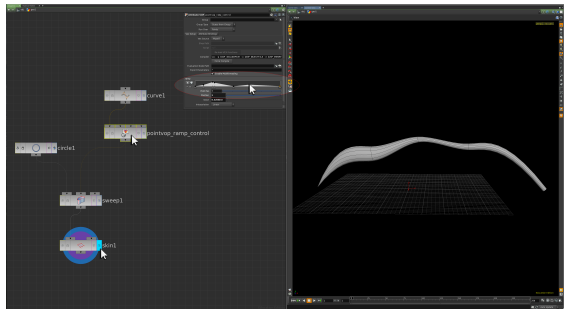
This will re-range (fit) each curve point (ptnum) relative to the total number of points (numpt) into a 0-1 number range that can be used to drive a ramp controller.

RMB on the **output** of the **Fit VOP** and **append** a **Ramp Parameter VOP**. This will automatically configure a Ramp Parameter control on the Geometry Level parameters of the Point VOP SOP.

In its **parameters**, set the **Ramp Type** to **Spline Ramp (float)**. This will activate a single channel black and white based ramp instead of the default RGB colour ramp.



To the **output** of the **Ramp Parameter VOP append** a **Bind Export VOP**. This will allow for the output of custom attributes not normally found on the Output VOP. In the **parameters** of the Bind Export VOP specify a **Name** of **pscale**, and a **Type** of **Float**.



When the **Ramp Parameter** is played with at **Geometry Level**, it now controls the scale of each swept circle along the curve. **See file ramp_sweep_pscale_end.hipnc**