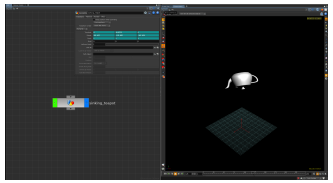
CREATING AN INTERACTIVE PARTICLE SUSPENSION

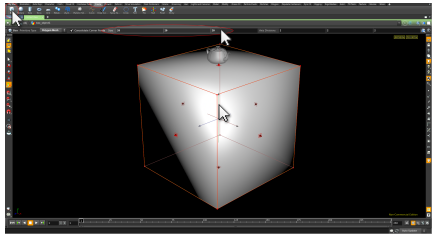
Making objects interact with each other is a core aspect of Houdini. For simple underwater FX work, a particle suspension can be made to interact with an object falling through its space. Open the scene **particle_suspension_begin.hipnc**.



This scene contains a simple key-framed animated sinking teapot. When **PLAY** is pressed, the teapot animates gently downwards as if sinking through water. In this example, a volume of suspended particles will be added and made to interact with the movement of the teapot.

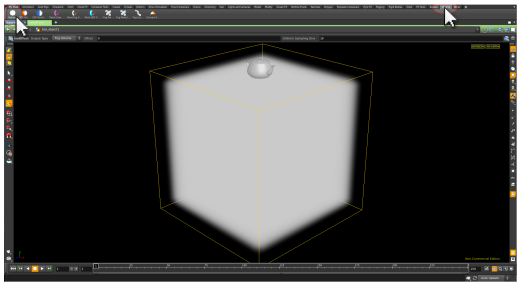
Maximise the **Viewer**, and **activate** the **Shelves**. From the **Create Shelf**, **LMB** the **Box Button** and **press Enter** to place it at the scene origin. In the Viewer **Parameters** for the **Box SOP**, specify:

Size	**20**	**20**	**20**



Using the **interactive handles raise** the **Box SOP up** so it sits just **underneath the teapot**.

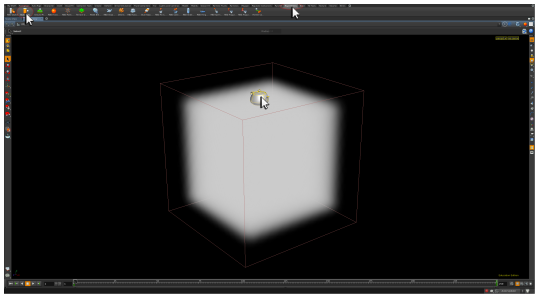
Activate the **Volume Shelf**, and **LMB** the **Volume Button**. This will convert the Box SOP into a Volume using an IsoOffset SOP.



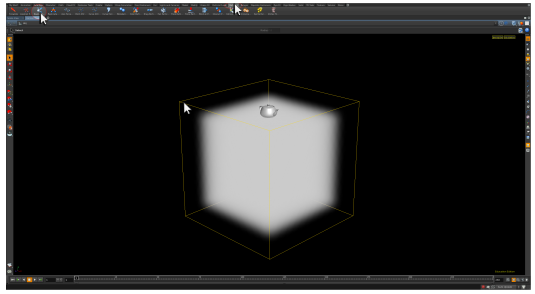
NOTE: The visual result of this operator can only be seen when the Viewer is set to a Smooth Shaded rather than Wireframe Shading view.

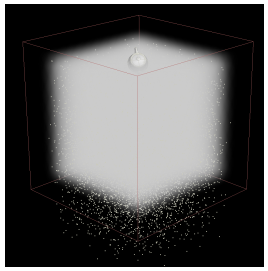
While volume representations of geometry are normally used for generating smoke or fog, they can also be used to create scattered points or a scattered particle suspension inside it.

Press U to return back to **Object Level**, **and Select** the **sinking_teapot object**. From the **Rigid Bodies Shelf** activate the **Static Object button**.



Next select the **particle_suspension object**, and from the **Particles Shelf**, activate the **Source Particle Emitter button**. When **PLAY** is pressed, particles rain down from the particle_suspension volume.

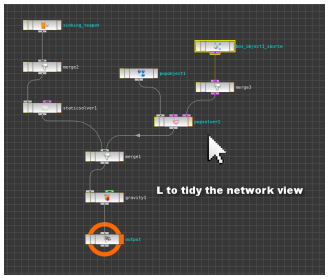




Behind the scenes, Houdini has created an **AutoDopNetwork** (a Houdini Dynamics Level - DOPs), where both the sinking_teapot and particle_suspension object have been imported.

See file particle_suspension_stage1.hipnc

Hide the Shelves and restore the **Network Editor**. Houdini should already be inside the **AutoDopNetwork** object. Pressing **L** on the keyboard with the mouse over the Network Editor will tidy up the node layout making the automated DOP (Dynamic Operators) nodes easier to see.

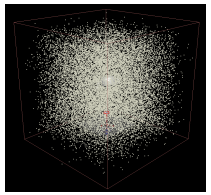


As the **sinking_teapot** is a **Static Object**, it **won't respond to forces** such as gravity, but simply **allow** for **other dynamic systems** to **interact with it**. The **volume box_object** is now a **source** for a **POP (Particles) Object**. **Both networks** are **processed** by their **own** respective **Solvers** before being merged together to allow for interaction to take place.

The **Gravity Force DOP** after the Merge DOP is currently responsible for creating the raining particles. As this raining effect is not required, **specify** its **parameters**:

Force	**0**	**-0.05**	**0**

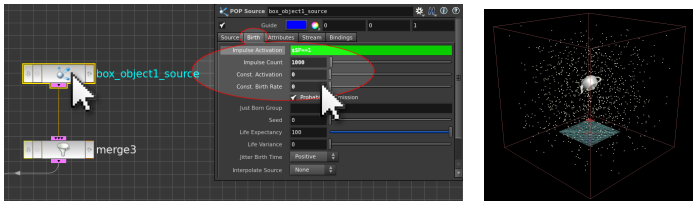
When **PLAY** is pressed, the raining effect no longer takes place; however many particles are continually being birthed from the source volume.



Locate the **box_object1_source DOP**, and under the **Birth** section of the **parameters** specify:

Impulse Activation	**$SF == 1**
Impulse Count	**1000**
Const. Activation	**0**
Const. Birth Rate	**0**

This will change the birth of the particles from time based (Constant Activation) to frame based (Impulse Activation). Setting the expression **$SF == 1** means that 1000 particles will be birthed only on the **first simulation frame**.

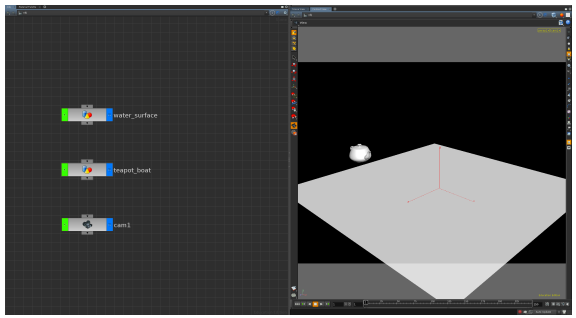


When **PLAY** is pressed, the particles birthed on frame 1 float around in space, but are moved around by the sinking teapot. The **Attributes > Variance parameter** of the **box_object1_source DOP** can also be modified to control the amount of random drift movement assigned to the particles.

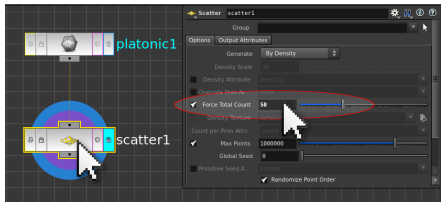
See file particle_suspension_end.hipnc

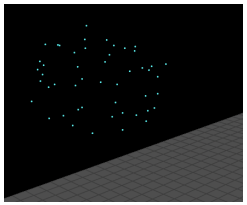
CREATING A WATER SURFACE

Open the scene **teapot_boat_begin.hipnc**. This scene contains an animated teapot that glides through the water_surface grid before sinking through it.



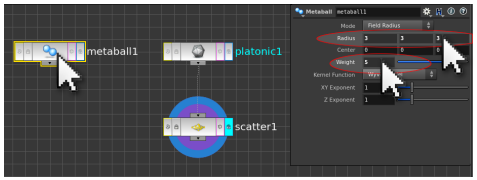
Go inside the **teapot_boat object** and **append** a **Scatter SOP** to the Platonic Solid SOP. **Specify** the **Force Total Count** parameter **to** scatter **50** points over the teapot geometry.



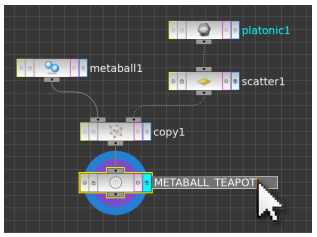


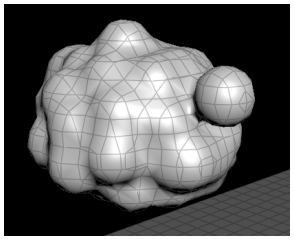
As a **new network chain**, create a **Metaball SOP**, and in its **parameters** specify:

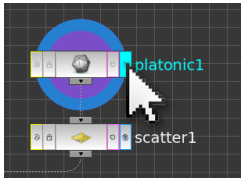
Radius	**3**	**3**	**3**
Weight	**5**		



The **Metaball SOP** can then be **copied** onto the **Scatter SOP** to create a blobby teapot.



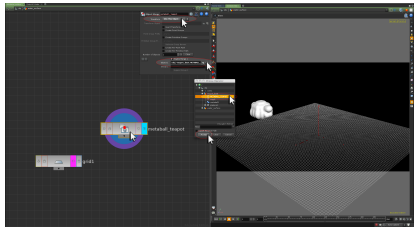




Append a **Null SOP** to the end of the network, and **rename** it to **METABALL_TEAPOT**. Finally **reset** the **Display / Render** Flag back to the **Platonic Solid SOP**.

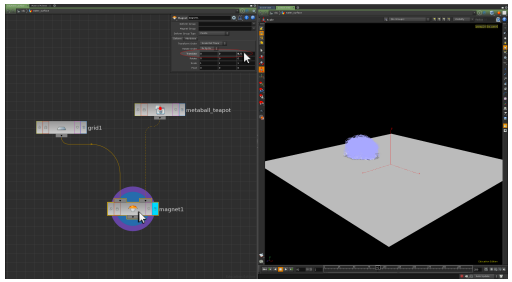
Inside the **water_surface object**, create an **Object Merge SOP** alongside the Grid SOP. In the **parameters** for the **Object Merge SOP**, specify:

Object 1	**/obj/teapot_boat/METABALL_TEAPOT**
Transform	**Into this Object**



To the **output** of the **Grid SOP**, append a **Magnet SOP**. Wire the **output** of the **Object Merge SOP** as its **second input**. In the **parameters** of the **Magnet SOP** specify:

Translate	**0**	**-0.2**	**0**

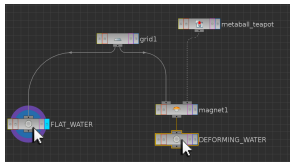


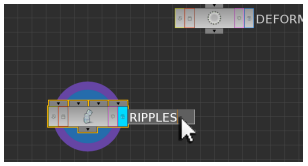
When **PLAY** is pressed, the grid geometry is gently deformed by the metaballs.

See file teapot_boat_stage1.hipnc

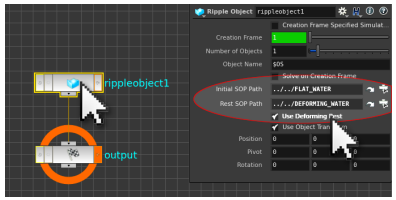
THE RIPPLE SOLVER

Houdini has a great tool for generating ripples. This is a Dynamics Tool called the **Ripple Solver**. To prepare the network for being passed into a Ripple Solver; **create two Null SOPs**. **MMB** append the first **Null SOP** to the output of the **Grid SOP** and **rename** it to **FLAT_WATER**. **RMB** append the **second Null SOP** to the output of the **Magnet SOP** and **rename** it to **DEFORMING_WATER**.





As a new network chain, create a **DOP Network**. This is a container for the Dynamics Level of Houdini. **Rename** this **DOP Network** to **RIPPLES**. **Double LMB** on this node to enter inside it.

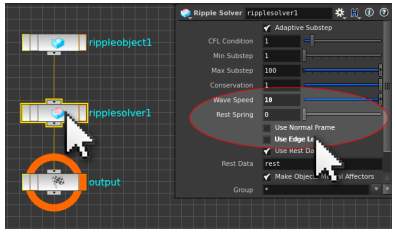


Inside the **RIPPLES DOP Network**, create a **Ripple Object DOP and wire it into the Output DOP**. This node will read in the FLAT_WATER and DEFORMING_WATER Null SOPs. In the **parameters** of the **Ripple Object** specify:

Initial SOP Path	**/obj/water_surface/FLAT_WATER**
Rest SOP Path	**/obj/water_surface/DEFORMING_WATER**
☑	**Use Deforming Rest**

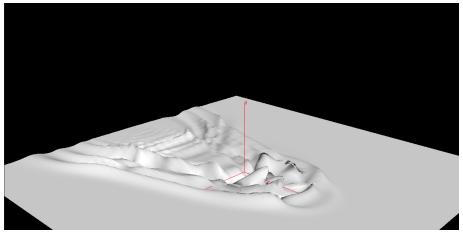
To the output of the **Ripple Object DOP**, append a **Ripple Solver DOP**. In its **parameters** specify:

Wave Speed	**10**
■	**Use Edge Len**



Deactivating the Use Edge Len parameter will treat the water surface as open water rather than contained water. If this option were activated, ripples would hit the edges of the geometry and rather than continuing, would simple bounce off the edges back into the ripple simulation.

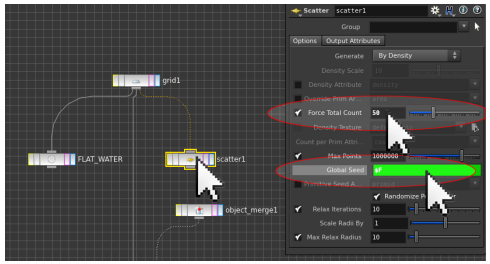
When **PLAY** is pressed, ripples will be calculated based upon the deforming water surface geometry. When the simulation has been fully calculated, **PLAY** can be pressed again to see the effect in real time playback.



MORE WATER INTERACTION

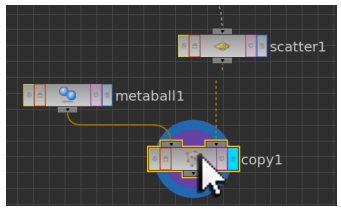
As well as the ripples generated by the teapot_boat; other ripples can be activated on the surface to represent wind or rain. **MMB** on the initial **Grid SOP** to create a **Scatter SOP** as a new network branch. In its **parameters** specify:

Force Total Count	**50**
Global Seed	**$F**

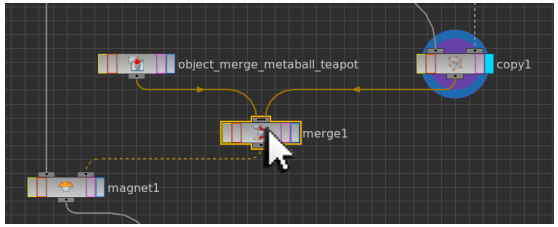


This will scatter 50 points over the surface, which randomly change position every frame (due to the setting of the Random Seed parameter to **$F** – **the current frame number**).

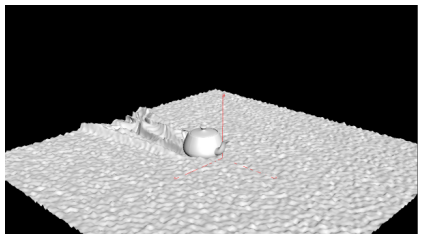
As a new network, create a **Metaball SOP**, and append to it a **Copy SOP**. Wire the **output** of the **Scatter SOP** as the **second input** to the Copy SOP. This will copy metaballs onto each random point, which can in turn be used to add additional deformation to the water surface before the ripples are calculated.



RMB append a **Merge SOP** to the output of the **metaball_teapot Object Merge SOP**. Wire the **output** of the **Copy SOP** as its **second input**. This will combine both sets of metaballs before they are passed into the Magnet SOP.



Reset the **Display and Render Flag** for this network back to the **RIPPLES DOP Network node**. When **PLAY** is pressed, the water surface ripples are generated from both the teapot_boat object and the scattered metaballs.

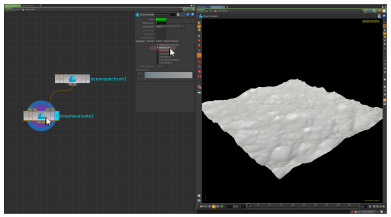


NOTE: Increasing the **Weight parameter** of the **Metaball SOP** copy scattered over the surface of the water can increase its influence on the ripple simulation. **The quality of ripples** can be improved by increasing the topology of the Grid SOP used to create the water surface, and adjusting the parameters of the Magnet SOP. Generating Flipbook renders will allow for the ripples to be viewed in real time.

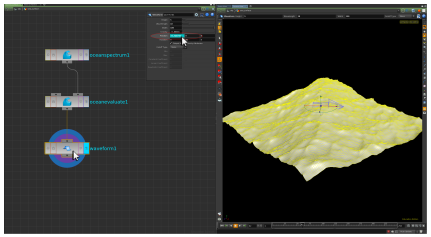
See file teapot_boat_stage2.hipnc

SEA SURFACES

The **Ocean Spectrum SOP** and **Ocean Evaluate SOP** are nodes that will generate a sea surface. Normally these nodes are used as part of the automated higher-level Ocean Shelf Tools, but they can be manually called in their own right. In a **new Houdini scene**, create a piece of geometry, and inside it **delete** the default **File SOP**. Create an **Ocean Spectrum SOP**, and wire it into the **second input** of an **Ocean Evaluate SOP**.



In the **parameters** for the **Ocean Evaluate SOP activate** the **Preview Grid tick box**. This will generate an animated sea surface. The **parameters** of the **Ocean Spectrum SOP** can be further adjusted to control the **aesthetic** of the resulting **sea surface**.

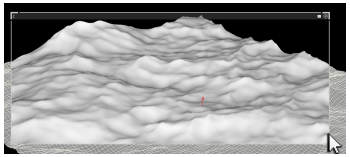


Large waves can also be passed through this animated sea surface by appending a **Waveform SOP** to the network. This node can be **key-frame animated** to control a bigger wave rolling through the sea surface.

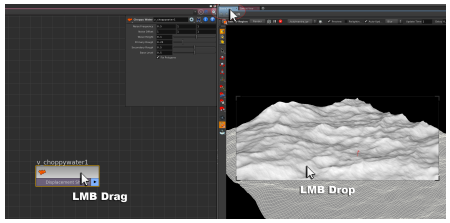
See file sea_surface.hipnc

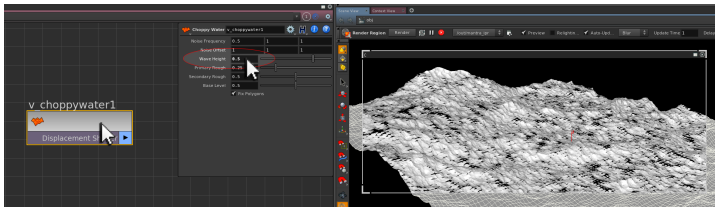
CHOPPY WATER DISPLACEMENT

With either method of water surface generation, a very smooth water surface is generated at render time. This can be seen with a **Render Region Preview**.



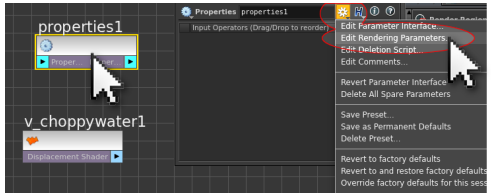
At **SHOP Level**, create a **Choppy Water Displacement Shader**, and **LMB drag and drop** it onto the **sea_surface object** by activating a **Scene View**. While initially this **finer waves displacement** appears to work, increasing the **Wave Height parameter** soon creates displacement tearing in the sea surface.



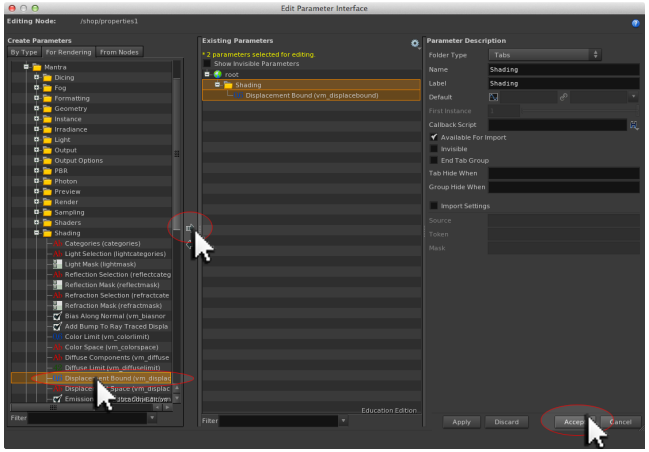


This is because Houdini needs to be told that displacement is occurring on a surface before it can be rendered correctly.

Alongside the Choppy Water Displacement Shader, create a **Properties Shader**. This node can be configured to add **Displacement Bounds** to the Choppy Water surface, to inform Houdini that surface displacement is taking place and should be anticipated at render time.

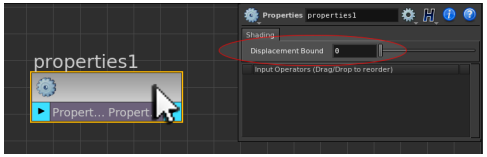


In the **parameters** for the **Properties Shader**, **activate** the **Cog Button Menu** and choose **Edit Rendering Parameters**. This will activate the Edit Parameter Interface window.

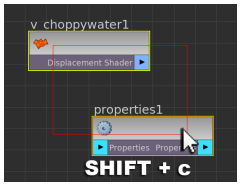


From the **Create Parameters > For Rendering list** go inside the **Mantra > Shading folder** and select the **Displacement Bound parameter**. Use the **right facing arrow** to **port** this currently inert parameter over to the **Existing Parameters list**.

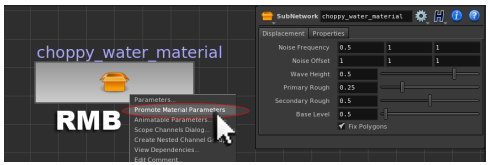
When **Accept** is pressed, this **Displacement Bound parameter** will now appear as an active parameter on the **Properties Shader parameter pane**.



In the **Network Editor**, **select both** the **Choppy Water Displacement Shader** and the **Properties Shader** and **press SHIFT + c** to convert them into a **Material**. Rename this material to **choppy_water_material**.

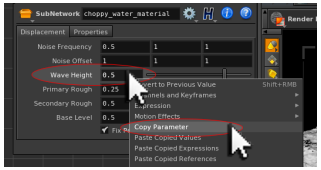


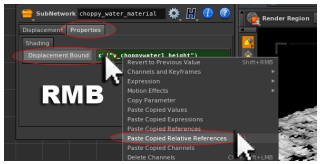
RMB on the **choppy_water_material** and from the resulting menu choose **Promote Material Parameters**.



This will grab all of the parameters from both internal shader nodes and port them to the top level of the material. A **channel reference** can be used on the **Displacement Bound parameter** to procedurally control the amount of displacement anticipation required at render time.

RMB on the **Wave Height parameter**, and from the resulting menu choose **Copy Parameter**.

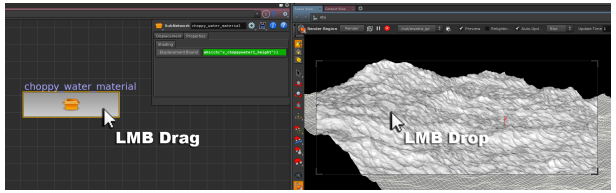




Go to the **Properties** section of the **choppy_water_material parameters**, **RMB** on the **Displacement Bound** parameter and from the resulting menu choose **Paste Copied Relative References**. **Modify** this **channel reference** from:

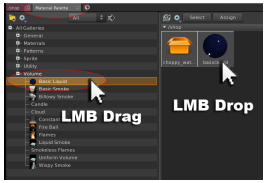
Displacement Bound	**ch("v_choppywater1_height")**
to:	
Displacement Bound	**abs(ch("v_choppywater1_height"))**

Now the **Wave Height parameter** of the choppy water displacement **will automatically set additional displacement bound space** on the geometry at render time. **Wrapping** the **channel reference** within an **absolute function abs()**, will always ensure the resulting value is always positive.



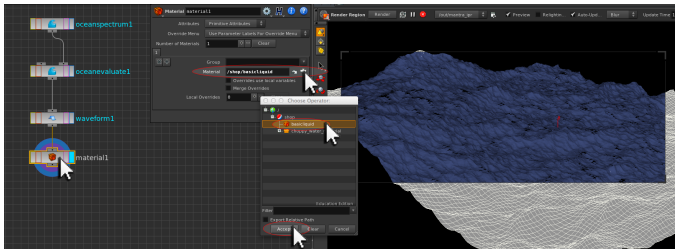
When the **choppy_water_material** is **LMB dragged and dropped** onto the **sea_surface** (overriding its previous shader assignment), the effect of the displacement can be seen.

A **Basic Liquid Material** can also be assigned to the sea_surface to colour the geometry for reflections and refractions. **This needs to be done at Geometry Level** so **not to override** the **choppy_water_displacement material** assigned to the **sea_surface at Object Level**.



Activate the **Material Palette** and **LMB Drag** and **Drop** a **Basic Liquid Material** into the **Palette region** next to the choppy_water_material.

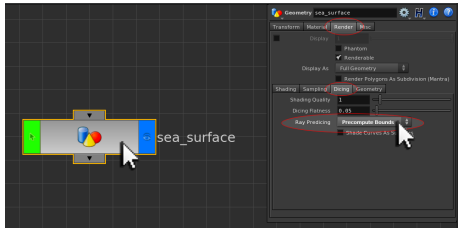
Inside the **sea_surface object**, append a **Material SOP** to the end of the sea surface construction network. The **Material parameter** can then be set to the **Basic Liquid Material** by activating the **Choose Operator button**.



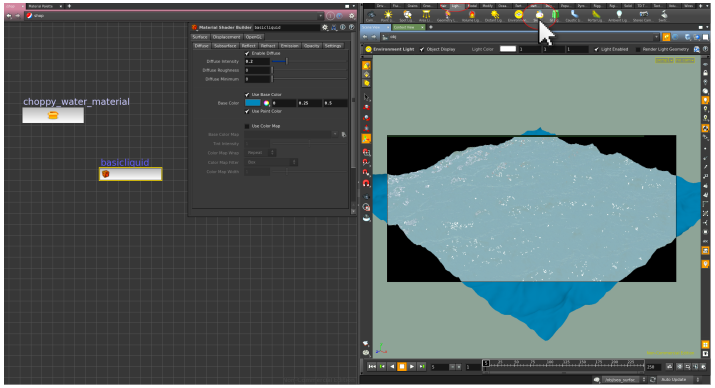
As a final step, go to the **Object Level parameters** of the **sea_surface object**, and from the **Render > Dicing** section specify:

Ray Predicing **Precompute Bounds**

This will ensure Houdini calculates all the displacement-bounding values first before rendering the sea surface, preventing any triangulation render errors.



The aethestic of the water surface can further be enhanced by adding a **Sky Light** from the **Lights and Camera Shelf**.

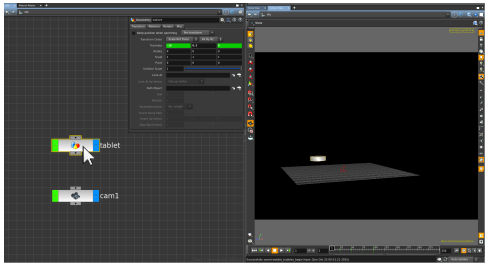


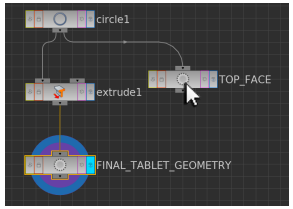
This will create a linked **Environment Light** (the skylight object) and a **Distance Light** (the sunlight object) to illuminate the scene.

See file sea_surface_choppy_water.hipnc

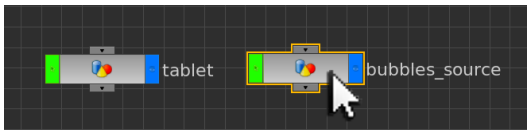
WOBBLY BUBBLES

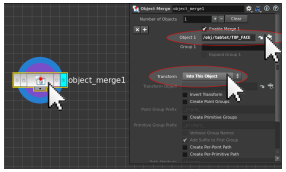
Open the scene **wobbly_bubbles_begin.hipnc**. This scene contains a simple tablet object that slide animates through the scene.





Inside the **tablet object**, alongside the main network is a **Null SOP** isolating the **top face** of the geometry. This can be used as a reference address to generate particle bubbles from.



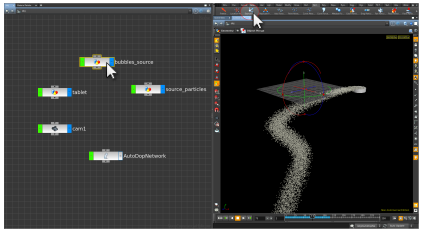


Back at **Object Level**, create a **new piece** of **geometry** and rename it to **bubbles_source**. Inside it, **delete** the **default File SOP** and in its place create an **Object Merge SOP**. In the **parameters** for the **Object Merge SOP** specify:

Object 1	**/obj/tablet/TOP_FACE**
Transform	**Into This Object**

This will extract the top face geometry of the tablet into the bubbles_source object, plus also read in its Object Level animation data.

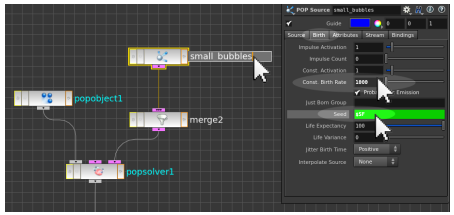
Return back up to **Object Level**, and with the **source_bubbles object** selected; press **ENTER** with the **mouse over the Viewer** to activate its **tool mode**. Activate the **Particles Shelf**. **LMB** the **Source Particle Emitter Button** will generate an **AutoDopNetwork** containing the particle system, and a **source_particles object** reading in the particles from DOP Level. When **PLAY** is pressed particles emit from the bubbles_source object.



Inside the **AutoDopNetwork**, rename the **POP Source DOP** to **small_bubbles**. In its **parameters** specify:

Birth >

Const. Birth Rate	**2000**
Seed	**$SF**

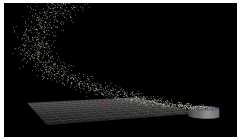


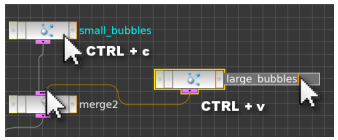
This will emit 2000 particles to each second, with a per simulation frame (**$SF**) randomisation over the surface, as well as varying direction for each particle.

Locate the **Gravity Force DOP** and in its **parameters** specify:

Force	**0**	**9.80665**	**0**

This will cause the particles to emit upwards when **PLAY** is pressed.





Select the **small_bubbles DOP** and **Copy** (**CTRL + c**) and **Paste** (**CTRL + v**) it into the network. **Rename** this pasted node to **large_bubbles** and **wire it into the Merge DOP** feeding into the POP Solver. In the **parameters** for the **large_bubbles DOP** specify:

Birth >

Const. Birth Rate	**500**		
Seed	**$SF * 2**		

Attributes >

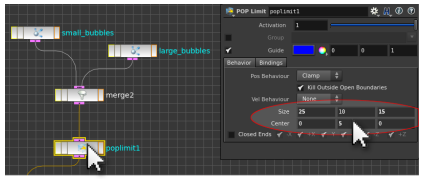
Velocity	**0**	**2**	**0**
Variance	**2**	**2**	**2**

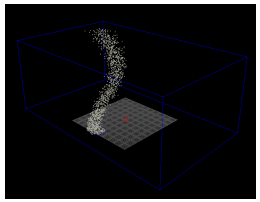
This will create a second stream of large_bubbles particles whose behaviour is slightly different from the small_bubbles particle system.

To the output of the **merge2 DOP RMB insert** a **POP Limit DOP**. In its **parameters** specify:

Size	**25**	**10**	**15**
Centre	**0**	**5**	**0**

This will create a bounding box around the particle systems, automatically culling any particles that go beyond its reach.

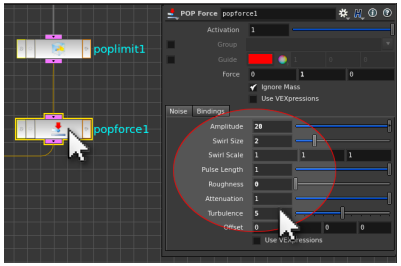




ADDING BUBBLE MOVEMENT

In order to add more swirling movement to the particles, a **POP Force DOP** can be **appended** to the **POP Limit DOP**. Under the **Noise** section of the **parameters** for the **POP Force DOP** specify:

Amplitude	**20**			
Swirl Size	**2**			
Roughness	**0**			
Turbulence	**5**			
Offset	**1**	**0**	**0**	**0**

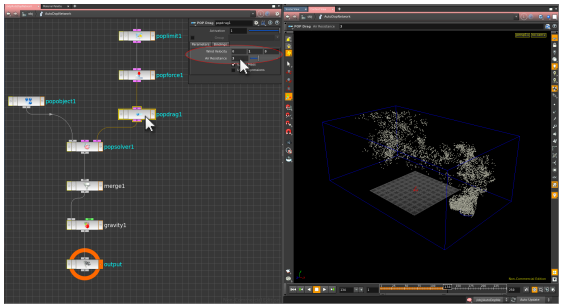


When **PLAY** is pressed, the particles move and swirl albeit somewhat fast. This movement can be calmed down by **appending** a **POP Drag DOP** to the **POP Force DOP**.

In the **parameters** for the **POP Drag DOP** specify:

Wind Velocity	**0**	**1**	**0**
Air Resistance	**3**		

Now when **PLAY** is pressed, the swirling particles appear to have the resistance of being underwater.

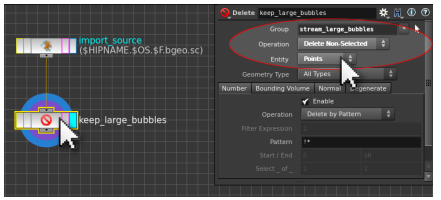


See file wobbly_bubbles_stage1.hipnc

BUILDING BUBBLE GEOMETRY

Back at **Object Level**, go inside the **source_particles object** and to the **output** of the **import_source SOP** append a **Delete SOP**. This can isolate either the big_bubbles or small bubbles so they can be processed separately. In the **parameters** of the **Delete SOP** specify:

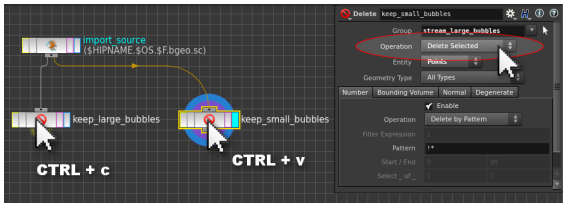
Group	**stream_large_bubbles**
Operation	**Delete Non-Selected**
Entity	**Points**



This will isolate the **large_bubbles particle stream** as a new network chain.

Copy (**CTRL + c**) and **Paste** (**CTRL + v**) this **Delete SOP** to create a **second instance** of it. To **isolate** the **small_bubbles particle stream**, **rename** this second **Delete SOP** to **keep_small_bubbles**, and in its parameters specify:

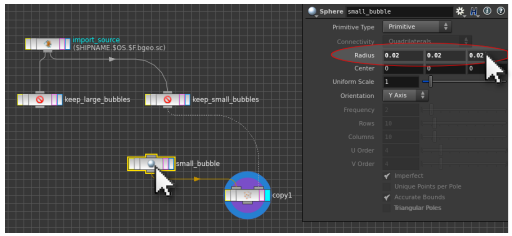
Operation	**Delete Selected**

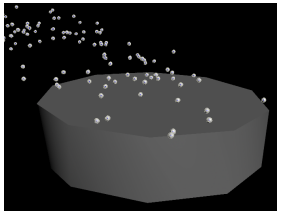


Create a **Sphere SOP** and rename it to **small_bubble**. In its **parameters** specify:

Radius	**0.02**	**0.02**	**0.02**

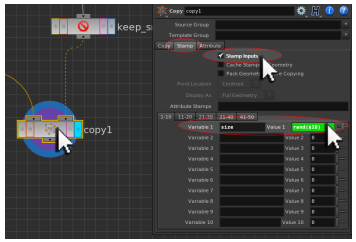
Use a **Copy SOP**, to **copy** the **small_bubble sphere** onto the **small_bubbles particle stream**.





COPY STAMPING

An ability of the Copy SOP is to add per copy variation. This is known as **Stamping**. In the context of the small bubbles, **copy stamping** can be used to **vary the size** of the small bubble as it gets copied onto the particles.



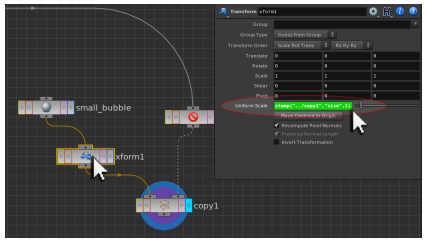
Go to the **Stamp** section of the **parameters** for the **Copy SOP**, and specify:

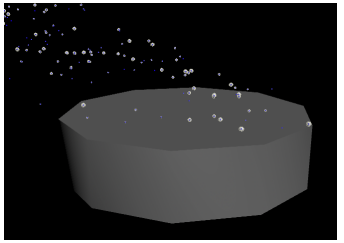
☑	**Stamp Inputs**		
Variable 1	**size**	**Value 1**	**rand($ID)**

This will create an internal **variable** called **size**, assigning a 0-1 random number to each individual particle ($ID). This variable can be passed upstream to create variation. **RMB** append a **Transform SOP** to the **small_bubble Sphere SOP**, and in its **parameters** specify:

Uniform Scale	**stamp("../copy1","size",1)**

This will resize the small_bubble Sphere SOP based upon it's existing size multiplied by the 0-1 random number created by the Copy SOP stamping.





When **PLAY** is pressed, random sized small bubble spheres have been copied to the particle stream.

NOTE: The **final number** assigned to the **stamp() function** is the value the parameter should use if it cannot find a value through copy stamping. If necessary the **Radius of the Sphere SOP** can be **temporarily increased** to see the effect more clearly.

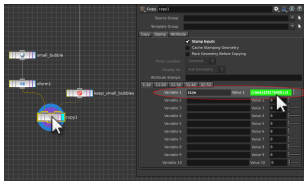
CREATING GROWING BUBBLES

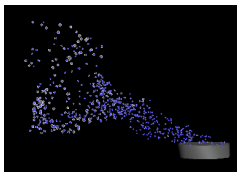
Return to the **Copy SOP** and under the **Stamp parameters**, modify the **size** variable expression to read:

Variable 1	**size**	**Value 1**	**(rand($ID)*$AGE)+1**

$AGE is a **variable** created on particle systems that **returns** the **current age** of each particle over time. The net effect of multiplying $AGE with rand($ID) is that each randomly sized particle will get bigger over time.

NOTE: The use of **+1** at the end of the size expression simply ensures the size of the particles when birthed is not absolute 0.





Now when **PLAY** is pressed, the **bubbles get larger** relative to the age of the particles.

See file wobbly_bubbles_stage2.hipnc

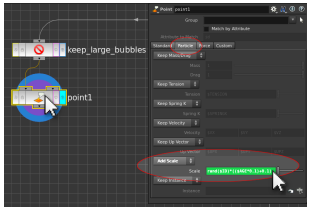
WOBBLY BUBBLES

While a copied stamped sphere is fine for generating small bubbles with size variation; for bigger bubbles a different technique can be used. This is where the big_bubbles particle stream can get explicitly surfaced as a particle fluid in order to generate bubbles that blob together to form more complex shapes. To the **keep_large_bubbles Delete SOP** append a **Point SOP**. In its **parameters** specify:

Particle >

Add Scale

Scale **rand($ID)*(($AGE*0.1)+0.1)**



This will assign a growing random scale attribute to each particle that will get automatically recognised by other Houdini operators.

Append to the **Point SOP** a **Particle Fluid Surface SOP**. This operator will generate a fluid mesh around each particle that can blob together. In the **parameters** for the **Particle Fluid Surface SOP** specify:

Surface >

Point Radius Scale **3**

Surface Tightness **1**

Speed Stretching >

☑ **Enable Speed Stretching**

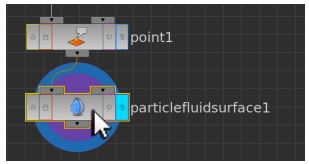
Particle Duplicates **10**

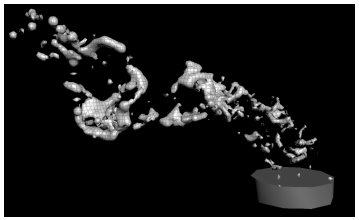
☑ **Scale Stretch Length**

Stretch Fraction **0.4**

Maximum Stretch **0.6**

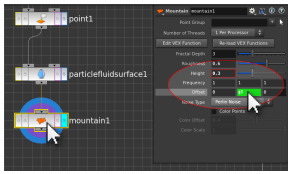
Particle Size Ratio **0.2**

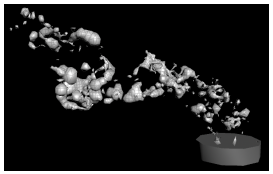




When **PLAY** is pressed, a bobbly fluid surface is assigned to the big_bubbles particle stream. The Speed Stretching parameters of the Particle Fluid Surface SOP will create subtle tails to the blobs as they rise, and can be further adjusted if required to create longer tails.

To make the fluid surface wobble as the bubbles rise, **a Mountain SOP** can be **appended** to the **Particle Fluid Surface SOP**. This will add a bit of **noise-displaced** volume to the top regions of the blobs, and this **node can** also **be animated** to create the **wobbling effect**.

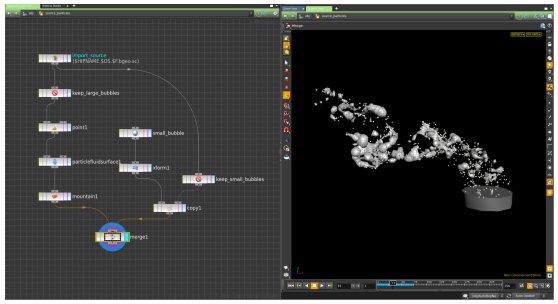




In the **parameters** of the **Mountain SOP** specify:

Height	**0.25**		
Offset	**0**	**$T**	**0**

Using **$T** (**time**) will cause the displacement noise to animated upwards as the bubbles rise.



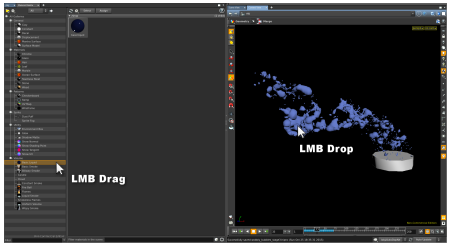
As a final step, the two geometry networks can be merged together with a Merge SOP.

NOTE: The yellow warning error generated by the Merge SOP is due to the different merged networks have different attributes. This warning can be ignored.

See file wobbly_bubbles_stage3.hipnc

RENDERING THE BUBBLES

At **Object Level**, **activate** the **Material Palette** and **LMB Drag and Drop** a **Basic Liquid Material** onto the Object Level bubbles.



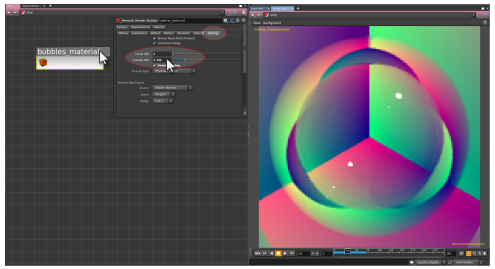
At **SHOP Level**, rename this material to **bubbles_material** and in its **parameters** specify:

Surface > Diffuse

Enable Diffuse

Surface > Settings

Inside IOR	**1**
Outside IOR	**1.344**



This will create a bubble in water aesthetic.

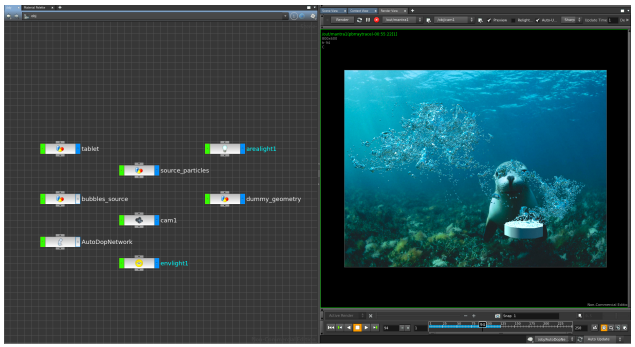
A **simple lighting, camera and Mantra PBR setup** can be added to the scene in order to test the aesthetic of the bubbles, and the amount and size of large and small bubbles can be adjusted accordingly.

At **Object Level**, the **bubbles** can also be **subdivided at render time** by activating the **source_particles > Render > Render Polygons as Subdivision (Mantra)** option. This will smooth out any artefacts caused by the Mountain SOP creating the wobble effect.

The **Background Plate** can also be **camera-projected as a Constant Material** onto either **Dummy Geometry or** a **large grid** placed at the back of the scene in order to **enable accurate refractions** in the bubbles. **Light linking** and can be used to ensure it doesn't respond to scene lighting or cast shadows over the scene. **Phantom Rendering** can be activated so **Dummy Geometry only appears in reflections and refractions**.

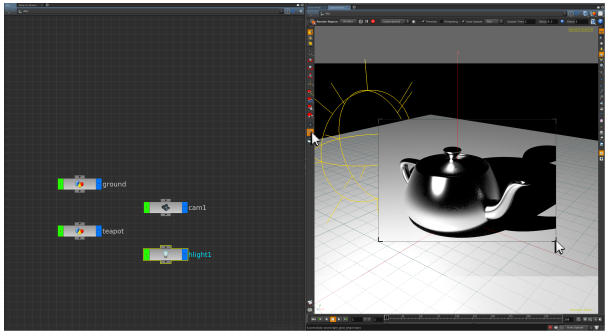
NOTE: These principles will be covered more in depth in a future lecture; however are presented here for reference.

See file wobbly_bubbles_complete.hipnc

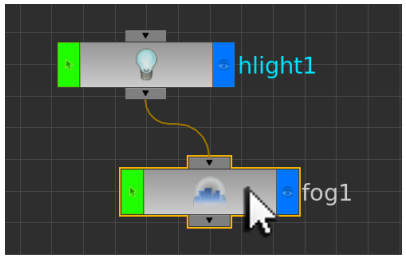


CREATING A LIT FOG LIGHT GLOW

As well as creating spotlight beams, lit fog can also be assigned to point lights to create soft glows akin to light bulbs. Open the scene **light_glow_begin.hipnc.**



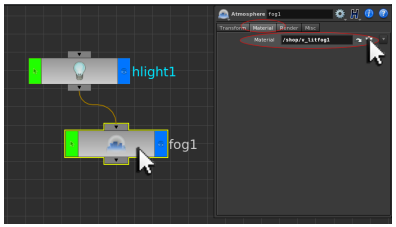
This is a simple scene with a point light illuminating a teapot and a grid. A **Mantra PBR ROP** has been added to the scene for Physically Based Rendering. A **Render Region Preview** reveals that the **teapot** has a **Mantra Surface Material** assigned to it configured as a **reflective surface**. Note that the **point light source reflects** in this mirror surface, but does not appear in the scene itself.

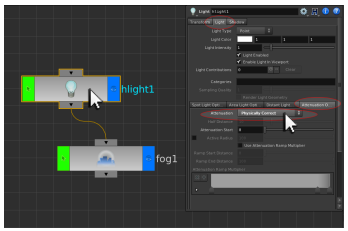


To the **output** of the **point light RMB** append an **Atmosphere object**. This object can be used to assign a lit fog glow to the point light.

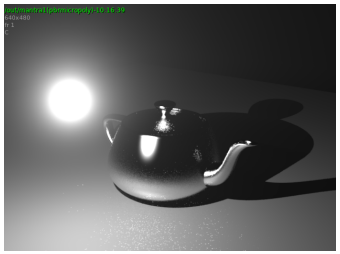
At **SHOP Level**, create a **Lit Fog shader** alongside the **Mantra Surface Material**.

Back at **Object Level**, assign the **Lit Fog Shader** to the **Atmosphere Object's Material parameter**.





In the **parameters** for the **point light** activate the **Light > Attenuation Options > Attenuation parameter** as **Physically Correct**. When the scene is rendered, a lit fog glow appears around the point light source; however it is incredibly slow to render.



See file light_glow_stage1.hipnc

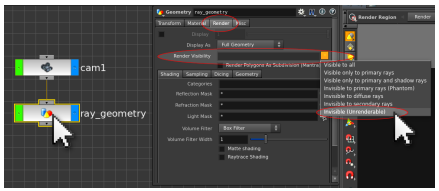
MEASURING LIT FOG

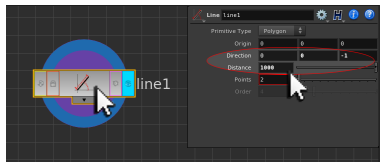
In order to improve render times for lit fog, its configuration needs to be measured relative to the scene itself and also the parameters of the Lit Fog shader. The **Houdini Help** for the **Lit Fog Shader** states:

`This fog shader computes lit fog by marching a ray from the eye to the surface being shaded. The illumination in the scene is computed at each step along the ray. If shadows are turned on, this will cause shadow shaders to be evaluated possibly increasing render times substantially.`

This means a ray (or a line) is being fired out from the camera at render time, and at each point along that ray (or line), Houdini is performing the lit fog calculation. Understanding this ray (or line) configuration is therefore key for generating optimal lit fog render times.

At **Object Level**, **RMB** append a **Geometry Object** to the **Camera**. **Rename** this object to **ray_geometry**. In the **Render** section of its **parameters**, **set** the **Renderable Visibility** parameter to **Invisible (Unrenderable).** This will prevent the ray_geometry from appearing in any renders.

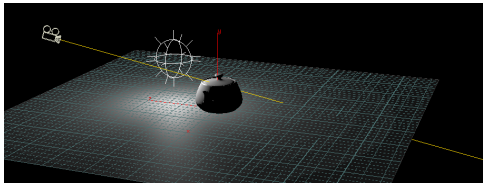




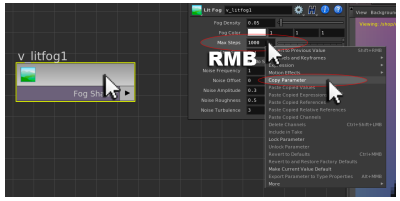
Inside the **ray_geometry object**, **delete** the **default File SOP** and in its place create a **Line SOP**. In the **parameters** for this Line SOP specify:

Direction	**0**	**0**	**-1**
Distance	**1000**		

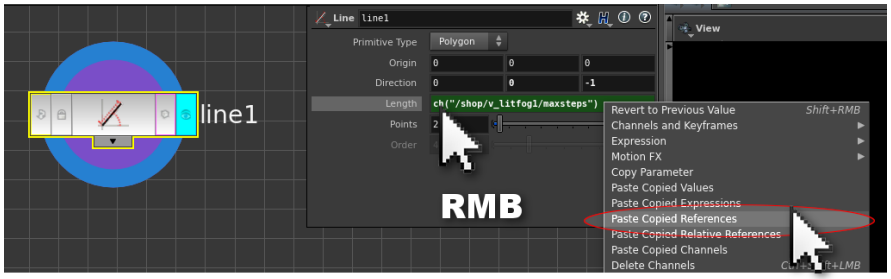
When the scene is examined, the ray_geometry object fires a very long line (or ray) out from the camera into the scene.



Switch to **SHOP Level**, and **RMB** on the **Lit Fog shader's Max Steps parameter**. From the resulting menu choose **Copy Parameter**.



Back inside the **ray_geometry object**, **RMB** on the Line SOP's **Length parameter**, and from the resulting menu choose **Paste Copied References**.



Modify this **Channel Reference** from:

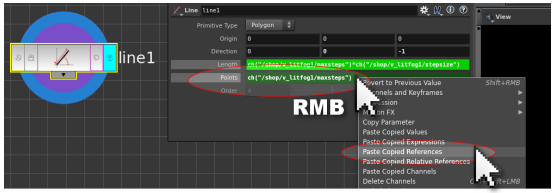
Length **ch("/shop/v_litfog1/maxsteps")**

to:

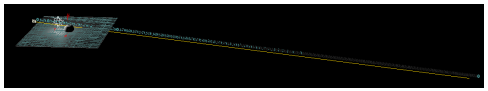
Length **ch("/shop/v_litfog1/maxsteps") * ch("/shop/v_litfog1/stepsize")**

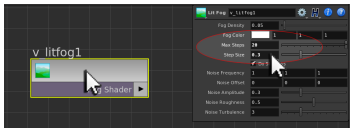
NOTE: Pressing **ALT + E** when editing a Channel Reference or Expression will activate an Edit Expression window for easier typing.

As a final step, **RMB Paste** the **Copied References** into the **Line SOP's Points** parameter.

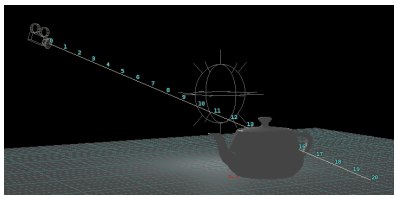


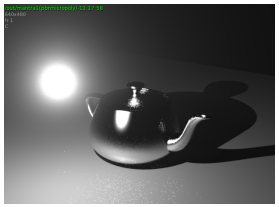
Examination of the scene now reveals why the initial lit fog glow render took so long to compute. Based on the default lit fog settings, the lit fog calculation was happening at each point along the ray_geometry well beyond the scope and scale of the scene.





The **parameters** of the **Lit Fog shader** can now be adjusted to values more appropriate to the scene (for example **Max Steps 20**; **Step Size 0.3**).



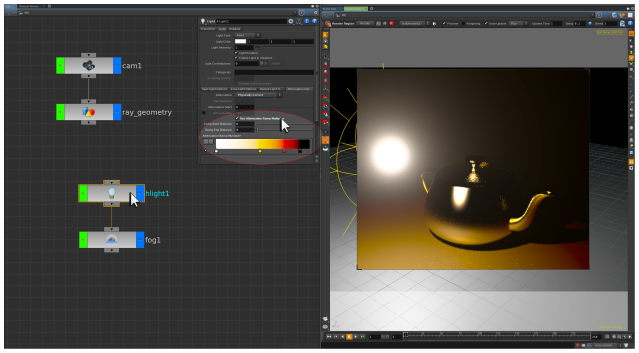


This will shorten the length of the ray_geometry to within the scene itself, and reduce the number of steps along the line that the lit fog will have to calculate. When the scene is rendered again, the lit fog appears as before; however render times are significantly reduced.

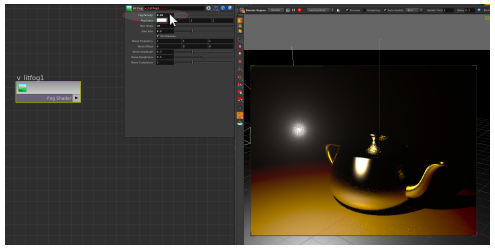
See file light_glow_stage2.hipnc

CONTROLLING THE LIT FOG AESTHETIC

Once the configuration of the lit fog has been set correctly, its aesthetic can be adjusted. This is a combination of the point light's parameters as well as the Lit Fog shader's parameters.



Activating the **Use Attenuation Ramp Multiplier parameters** in the Point Light will allow for coloured light fall off to occur over the scene.

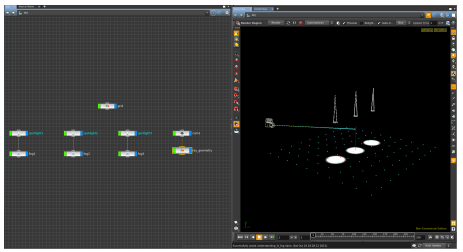


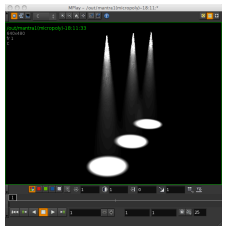
Adjusting the **Fog Density parameter** of the **Lit Fog Shader** will determine how large or small the lit fog glow is.

See file light_glow_complete.hipnc

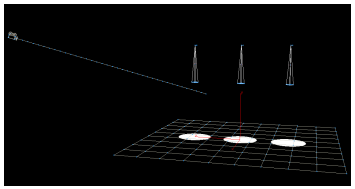
UNDERSTANDING LIT FOG

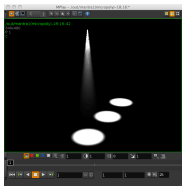
Open the file **understanding_lit_fog.hipnc**. This file contains a simple setup where three spotlights with lit fog are illuminating a ground plane. A ray_geometry object has been configured and parented to the camera to control the parameters found on the Lit Fog Shader.





When the scene is rendered from the scene camera, all three spot lights appear with lit fog.





If however the Max Steps Parameter and the Step Size Parameter of the Lit Fog Shader are adjusted so the line only enters the first spot light region; only the first spot light will appear with lit fog. **NOTE: the ray_geometry object has been set not to render.**

The **quality of the fog** is being controlled by the **Step Size** parameter. The lower this value, the better the fog; however the **Max Steps** parameter will need to be increased so that the ray being fired from the camera still hits the spot lights. This simple way of visually measuring lit fog can be used to create optimum render settings.

IMPORTANT NOTE: The Lit Fog parameters are applied to every camera in the scene. This includes the standard persp1 view. This means if the camera view is moved away from cam1, the Lit Fog may disappear depending upon the distance of the persp1 camera from the scene being rendered.

TIPS FOR SUCCESSFUL LIT FOG
To get lit fog to appear directly emitting from the light source, Physically Correct Attenuation should be used on scene lights.

The Max Steps Parameter and the Step Size Parameter can also be animated so the ray fired from the camera is only ever rendering what it needs to. This is especially important when creating lit fog for animated objects (for example car headlights).

See file lit_fog_animated.hipnc