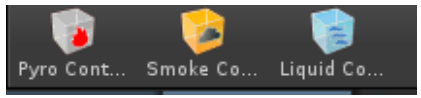
A SIMPLE SMOKE BOMB

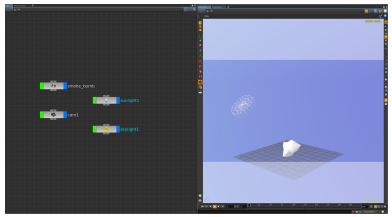
The **Pyro FX Shelf Tools** can create a number of presets for smoke, fire and explosion effects. Selected geometry can be assigned one of these presets, and Houdini will configure the associated networks automatically.

While these presets are very useful for quick effects setup; it is also useful to know a bit more about the components of dynamic volumetrics by creating a similar setup manually. This can be facilitated using the **Fluid Containers** and **Populate Containers** Shelf Tools.

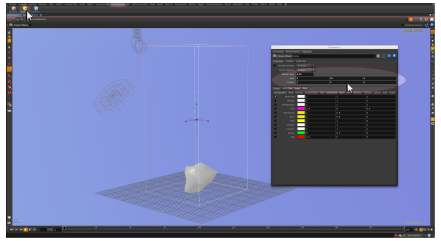


The **Fluid Containers Shelf** has three items: The **Pyro**, **Smoke** and **Liquid** Containers. These will create the digital equivalent of an Acquisition Water Tank to hold the fluid effect. Each of these tanks is a box with user-definable divisions to fill the box with voxels (3D pixels). As volumetric data is passed through these voxels, they are shaded accordingly. This means for example that volumetric smoke travelling through its container is actually static voxels being procedurally activated and deactivated to give the impression of moving smoke. A greater resolution of voxels will increase the visual quality of the fluid, but will result in longer render times.

Open the scene H15_smoke_bomb_begin.hipnc. This scene contains an animated sphere (the smoke bomb) and a simple camera and Sky Light setup. A Mantra PBR node has also been activated allowing for physically based rendering of the smoke bomb effect.



Maximize the **Scene View** and activate the **Shelves**. Ensure that the **Scene View's Tool Options** are set to **Create at Object Level**. From the **Fluid Containers Shelf** activate a **Smoke Container**. Press **ENTER** to **automatically place it** at the scene origin.



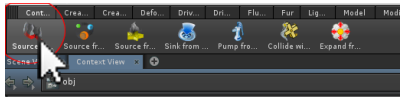
This will create an **AutoDopNetwork** with a **Smoke Object DOP** inside it. Pressing **p** with the mouse over the viewer can activate the parameters for the **Smoke Object DOP**.

In the **parameters** for the **Smoke Object DOP** specify:

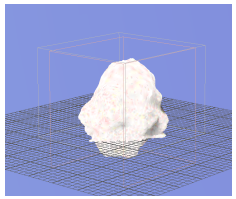
Division Size	**0.05**		
Size	**5**	**10**	**5**
Center	**0**	**5**	**0**

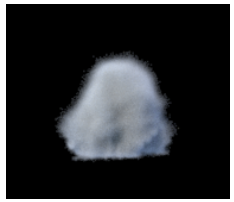
This will move and resize the smoke container so it surrounds the smoke bomb geometry, as well as increasing the voxel resolution of the container.

At **Object Level**, **select** the **smoke_bomb sphere** and activate the **Source from Surface** shelf button found on the **Populate Containers** Shelf. **When prompted**, **select** the Smoke Object **Container** and press **ENTER** to complete the setup. The **Source from… shelf tools** will automatically generate a fluid from an object, based upon the type of fluid container selected.



When **PLAY** is pressed, smoke emits from and surrounds the animated sphere.





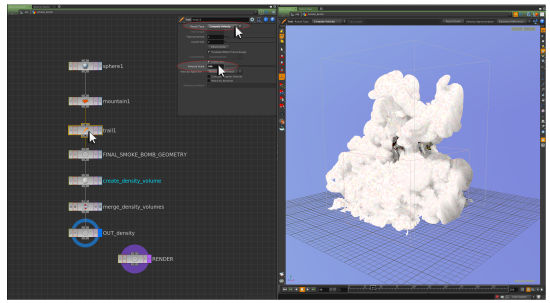
NOTE: The boundaries of the smoke container automatically resize relative to the smoke effect generated. This behavior can be deactivated if required.

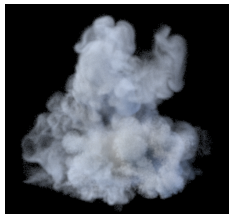
NOTE: Only objects within a container will respond to any fluid properties assigned to them. Once an emitter object moves outside of its container, its ability to produce fluid is culled.

CREATING MORE INTERESTING SMOKE

Currently the smoke effect is somewhat tepid. This is because any source geometry for the smoke must first be primed in an interesting way to generate interesting smoke effects. **Inside** the **smoke_bomb object**, insert a **Trail SOP** after the Mountain SOP animating the sphere. In the **parameters** for the **Trail SOP** specify:

Result Type	**Compute Velocity**
Velocity Scale	**500**





When **PLAY** is pressed, a more dramatic smoke effect is created.

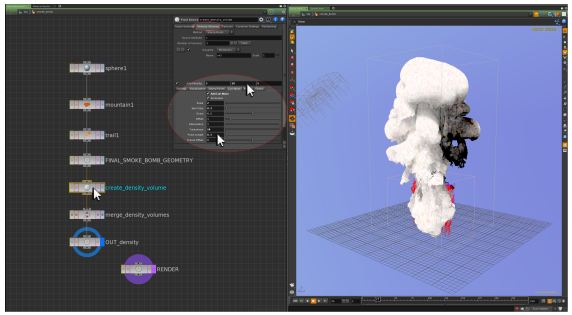
Modifying the **Fluid Source SOP** creating the **density volume** from the smoke bomb geometry can further enhance this smoke effect. In the **parameters** for the **Fluid Source SOP** specify:

Velocity Volumes >

☑	**Add Velocity**	**0**	**10**	**0**

Velocity Volumes > Curl Noise >

☑	**Add Curl Noise**	
	Scale	**2**
	Swirl Size	**0.3**
	Turbulence	**50**
	Pulse Length	**0.3**

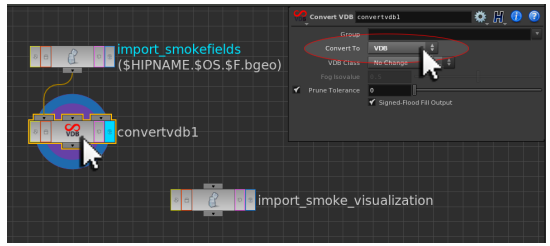


The **AutoDopNetwork's Smoke Solver DOP** can also adjust the overall fluidity of the smoke. Increasing **Viscosity** for example will create more **gooey smoke** aesthetics. The **Resize Container** node can determine which **sides of the fluid container** are deemed as **solid** or which of them are **holes** for the smoke fluid to escape out of the container. **See file H15_smoke_bomb_stage1.hipnc**

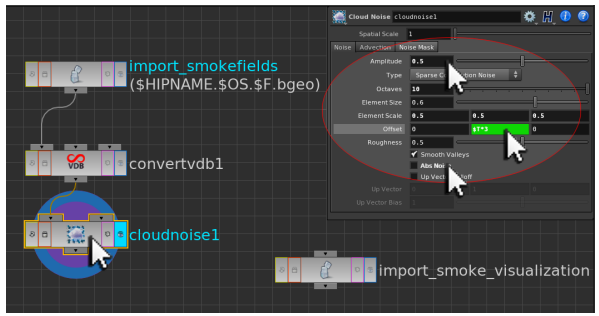
REFINING FLUID AESTHETICS

Once the overall shape of the smoke fluid has been established, its aesthetic can be refined. Inside the **import_smoke1 object**, append a **Convert VDB SOP** to the **import_smokefields SOP**. In the **parameters** for the **Convert VDB SOP** specify:

Convert To	**VDB**



Append to the Convert VDB SOP, a **Cloud Noise SOP**. This will allow cloud noise to be added to the smoke simulation, creating a greater sense of turbulence to the smoke aesthetic.



In its **parameters** specify:

Noise >			
Amplitude	**0.5**		
Octaves	**10**		
Element Scale	**0.5**	**0.5**	**0.5**
Offset	**0**	**$T*3**	**0**
☐ **Abs Noise**			

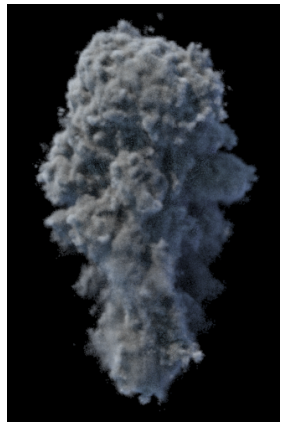
When the simulation is played again or rendered, the plume of smoke now has greater sense of visual scale due to the additional surface definition created by the Cloud Noise SOP.

Modifying the parameters of the **Billowy Smoke Material** (automatically created when the smoke_bomb object was sourced by the smoke fluid container) can also help increase the sense of scale to the smoke.

At **SHOP Level**, locate the **Billowy Smoke Material** and in its **parameters** specify:

Smoke Intensity	**2**		
Density >			
Smoke Density	**5**		
Shadow Density Multiplier	**3**		
Noise >			
☑ **Do Noise**			
Frequency	**0.5**	**0.5**	**0.5**
Amplitude	**0.2**		
Offset	**0**	**$T**	**0**

When the simulation is rendered again, and much denser plume of smoke is generated.



Adjusting parameters of the **Mantra PBR ROP** can also refine the render aesthetic of the smoke. A key parameter for rendering volumes is the **Volume Limit parameter**. This parameter controls **how many times light can bounce internally through a volume** giving a greater sense of internal volume illumination. This can be useful when creating more naturalistic volume effects (for example sunlight permeating through a cloud). At **OUTPUTS Level**, specify in the **Mantra PBR ROP**:

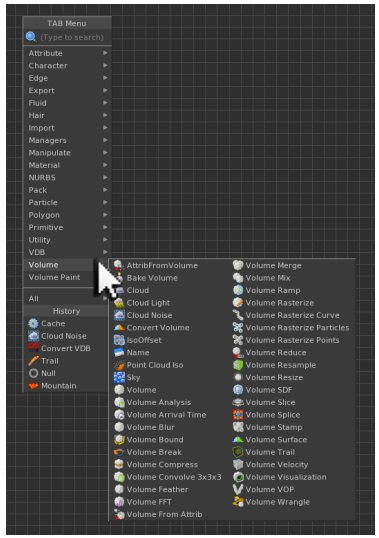
Rendering > Sampling
Volume Quality **1**
Rendering > Limits
Volume Limit **3**

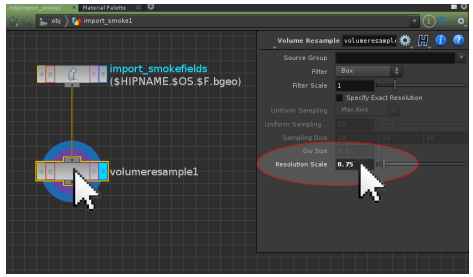
When the scene is rendered again, the smoke plume has a greater sense of internal lighting and overall quality; however render times have increased.

See file H15_smoke_bomb_stage2.hipnc

OTHER USEFUL VOLUME TOOLS

The **SOP Level** of Houdini also has a number of operators that can modify volumes; all of which are worth exploring. These can be found under the **Volumes section** of the **TAB Menu System**.





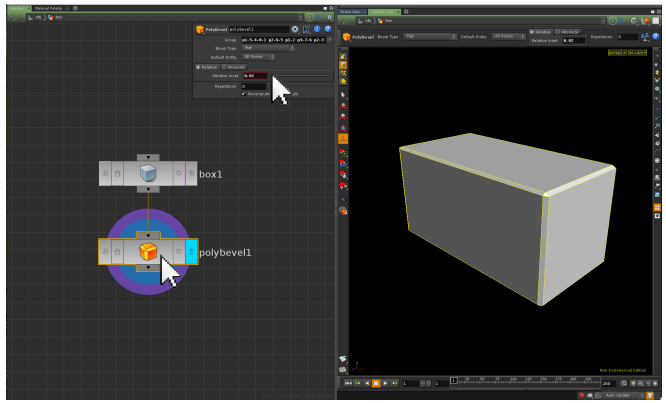
One such operator is the **Volume Resample SOP**. This operator can be used to increase or decrease the number of voxels in a volume, and can be useful for optimizing volume render times. **MMB** on the Volume Resample SOP node will **reveal** the resulting **Voxel Count**.

RBD POINT INSTANCING

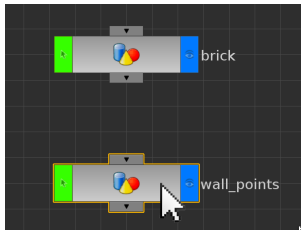
RBD Point Instancing can be used for generating multiple dynamic objects, either of the same or different geometry. In a **new Houdini scene**, generate a **Box Object**, and inside at its Geometry Level specify:

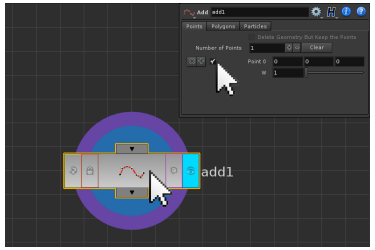
Size	**2**	**1**	**1**

Select all the faces of the **Box SOP** in the **Viewer** and **activate** a **Poly Bevel**. Reduce the **Relative Inset parameter** of the **Poly Bevel SOP** to **0.02**. This will give a simple brick that can be instanced onto point data at DOP Level.



Return back up to **Object Level**, and rename the **Box Object** to **brick**. Alongside it create a **new** piece of **geometry** called **wall_points**.





At **Geometry Level** for the **wall_points object**, **delete** the default **File SOP** and in its place create an **Add SOP**.

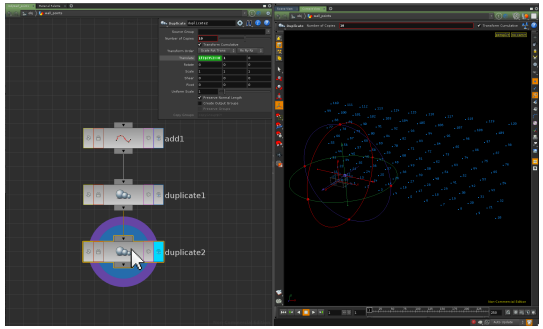
In its **parameters**, activate the **Points > Point 0 toggle option** to create a single point at the origin of the scene. Append to the **Add SOP**, a **Duplicate SOP**. In its **parameters** specify:

Number of Copies	**10**		
Translate	**2**	**0**	**0**

This will create a row of brick points. Append to the **Duplicate SOP**, a **second Duplicate SOP**. In its **parameters** specify:

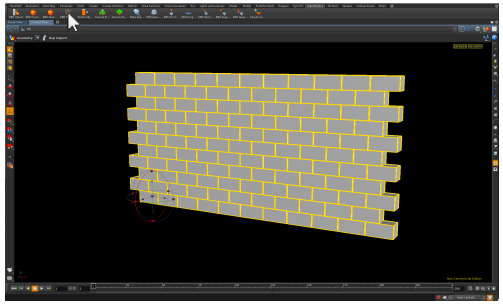
Number of Copies	**10**		
Translate	**if($CY%2==0,1,-1)**	**1**	**0**

This will create a wall of points for the brick to be instanced onto.



At **Object Level**, select the **wall_points object**, and using the **Rigid Bodies Shelf**, activate the **RBD Point Object button**. When asked **'What Type of point object?'** choose **RBD Point Object**.

The **Help Prompt** will ask the object for instancing to be selected. Select the brick object and press **ENTER** to confirm the selection.

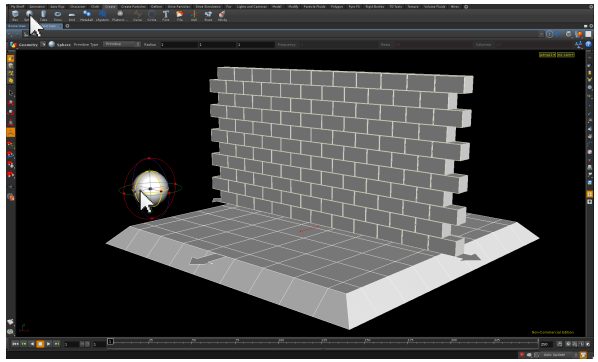


This will rebuild the wall as dynamic objects, and turn off the Display Flag on the original brick object.

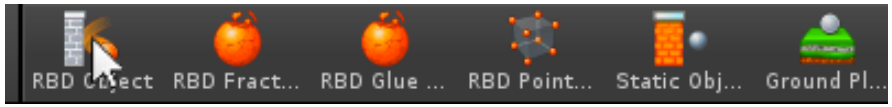
From the **Rigid Bodies Shelf**, create a **Ground Plane DOP**, and then position the dynamic wall so that is sits on top of it. The parameters for the wall can be activated by pressing p with the mouse over the Viewer to allow for precise numeric positioning.

From the **Create Shelf**, create a **sphere object**, and position it in front of the wall with a Translate value of:

Translate	**0**	**5**	**15**



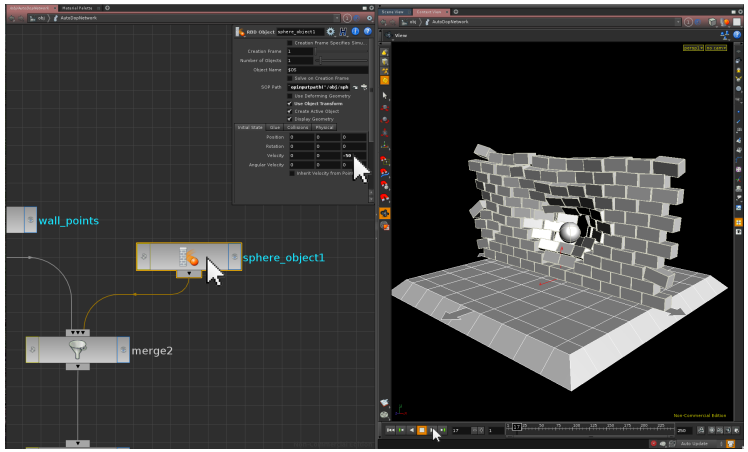
From the **Rigid Bodies shelf**, activate the **sphere object** as a **RBD Object**.



Inside the **AutoDopNetwork**, locate the **RBD DOP** reading in the **sphere** and specify in its **parameters**:

Initial State >

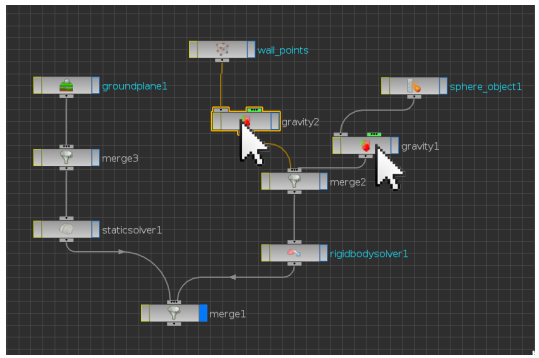
Velocity **0** **0** **-50**



When **PLAY** is pressed, the ball fires at the wall causing it to break apart.

BESPOKE GRAVITY

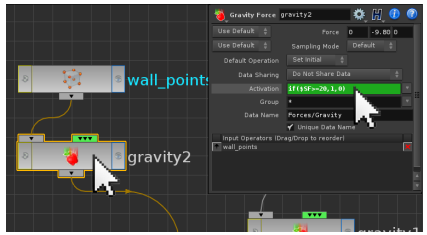
Currently the wall and sphere are under the effect of gravity. This gravity force is causing the bricks at the edges of the wall to wilt when PLAY is pressed. This breaks the illusion of the wall geometry. In the **Network Editor**, move the **Gravity Force DOP** from the end of the network, to under the **sphere_object1 RBD DOP**.



Append to the **wall_points RBD Point Object DOP** a second **Gravity Force DOP**. When **PLAY** is pressed, the simulation runs as before; however new values for the gravity force affecting the wall can now be given. In the **parameters** for the **second Gravity Force DOP** specify:

Activation **if($SF>=20,1,0)**

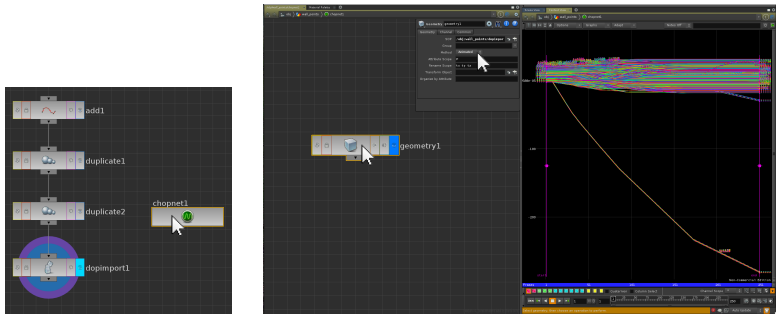
This will turn on the wall gravity after the ball has impacted with it.



See file H15_rbd_wall.hipnc

CREATING A TIME FREEZE EFFECT

Go to the **Geometry Level** of the **wall_points object**, and alongside the main network create a **CHOP Network**. This can be used to retime the incoming dynamics simulation.



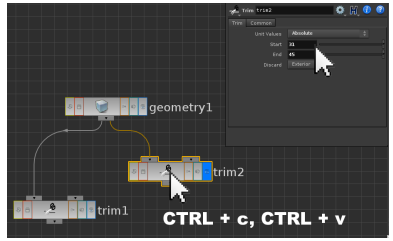
Inside the **CHOP Network**, create a **Geometry CHOP** to read in the **DOP Import SOP** for the dynamic wall. In its **parameters** specify:

SOP	**/obj/wall_points/dopimport1**
Method	**Animated**

Append to the **Geometry CHOP** a **Trim CHOP**. In its **parameters** specify:

Trim >	
Unit Values	**Absolute**
Start	**1**
End	**30**
Common >	
Units	**Frames**

This will create a clip of the first part of the animation sequence where the ball is impacting with the wall.



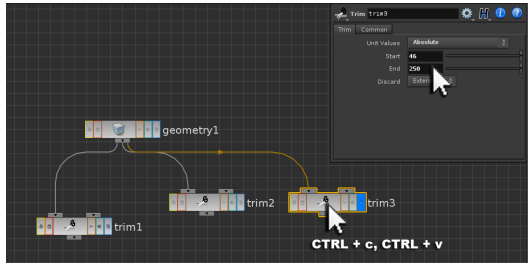
Copy and **Paste** (**CTRL + c**, **CTRL + v**) the **Trim CHOP** to create a second version of it. In its **parameters** specify:

Trim >	
Start	**31**
End	**45**

This will create a clip of animation where the wall begins to crumble.
Copy and **Paste** (**CTRL + c**, **CTRL + v**) the **Trim CHOP** to create a **third** version of it. In its **parameters** specify:

Trim >	
Start	**46**
End	**250**

This will create a clip of animation where the wall finishes collapsing.

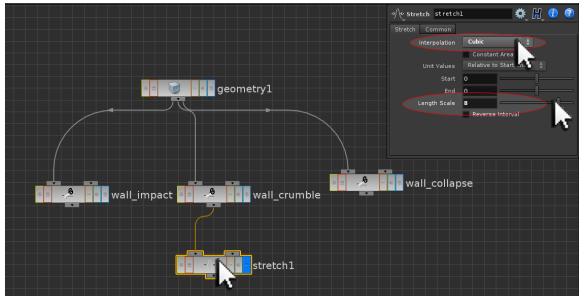


THE STRETCH CHOP

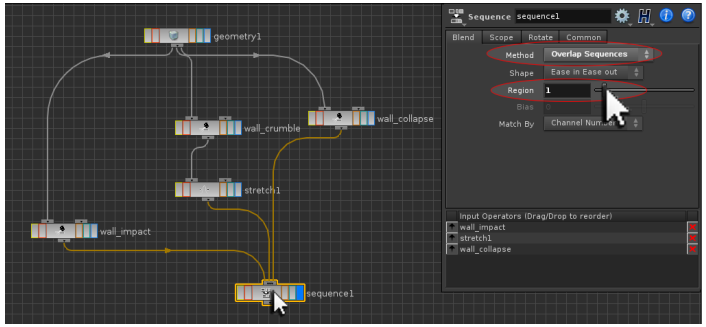
With the dynamic animation converted into a series of clips, the second clip where the wall begins to crumble can be stretched. **Append** to the **second Trim CHOP** a **Stretch CHOP**. In its **parameters** specify:

Interpolation	**Cubic**
Length Scale	**8**

This will stretch out the length of the second clip to create a time freeze just for this section of animation.



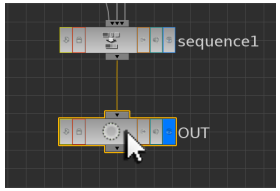
All three of the network branches can now be fed into a **Sequence CHOP** to restore the animation sequence.



In the **parameters** for the **Sequence CHOP** specify:

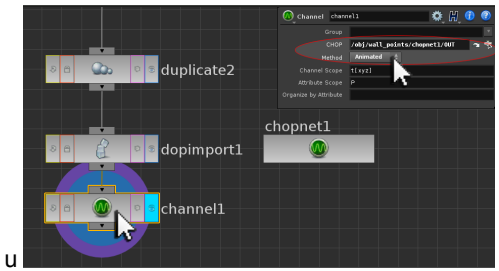
Method	**Overlap Sequences**
Region	**1**

This will recombine the clips by overlapping them slightly by 1 second.



As a final step, append a **Null CHOP** to the network and rename it to **OUT**.

This retimed animation sequence can now be reassigned to the wall simulation at **Geometry Level** by using a **Channel SOP**.

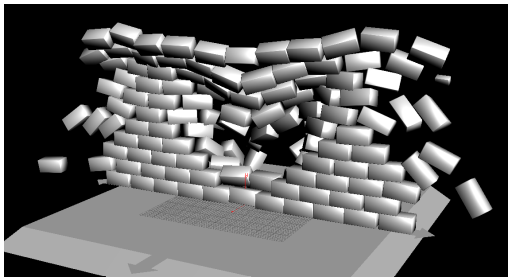


u

Append to the **DOP Import SOP**, a **Channel SOP**. In its **parameters** specify:

CHOP	**/obj/wall_points/chopnet1/OUT**
Method	**Animated**

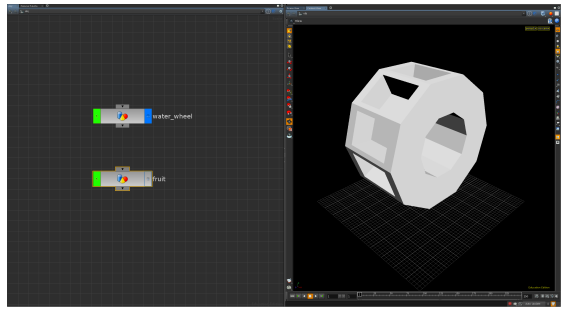
When **PLAY** is pressed, a time freeze occurs just as the wall begins to crumble.



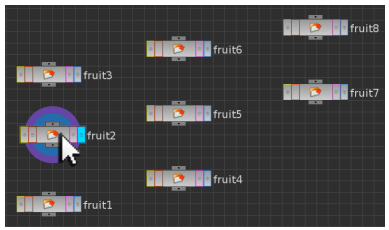
See file **H15_rbd_wall_timefreeze.hipnc**

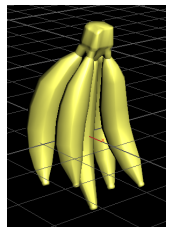
WATER WHEEL FRUIT CANNON

This example demonstrates how Flip Fluids and RBDs can interact, as well as how to constrain dynamic objects in space, and how to utilise specific creation frames for dynamic objects. Open the scene **fruit_cannon_begin.hipnc**.



This scene contains a **water_wheel geometry object** and a currently **hidden fruit geometry object**. Inside the fruit object are **8 File SOPs** each reading in a different type of fruit.

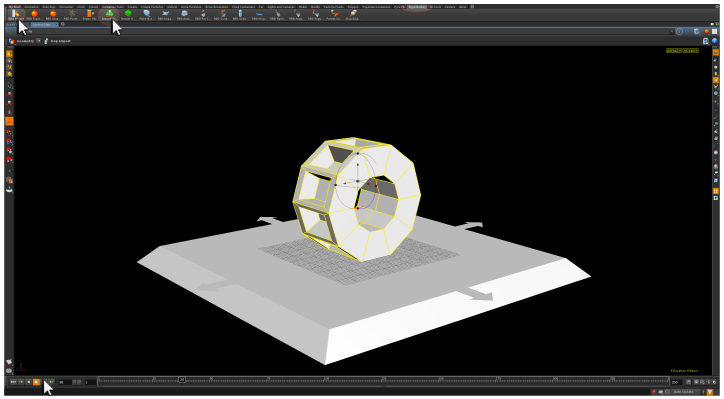




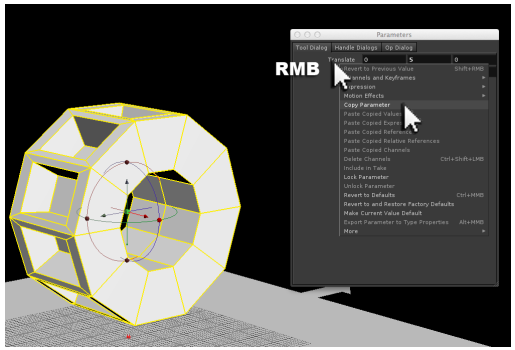
In this example, both fruit and water will be fired at the water_wheel causing it to spin.

CONFIGURING THE WHEEL

At **Object Level**, **maximise** the **Viewer** and **un-stow** the **Shelves**. From the **Rigid Bodies Shelf**, **LMB** on the **Ground Plane** button.



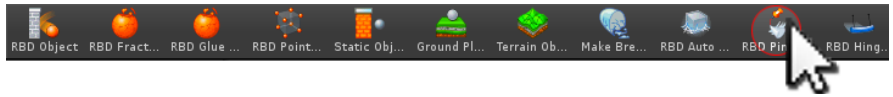
With the **water_wheel** object **selected**, press the **RBD Object button**. When **PLAY** is pressed, the water_wheel falls and lands on the ground plane.



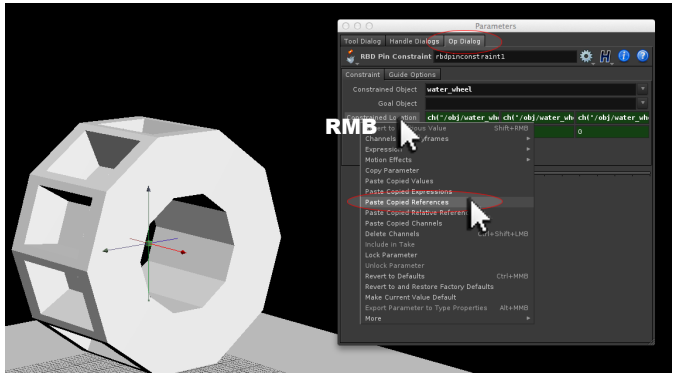
With the **water_wheel** object still **selected** in the **Viewer**, press **p** to activate a **floating Parameters Window**, and from the **Tool Dialog section**, **RMB copy the Translate parameter**.

ASSIGNING CONSTRAINTS

With the water_wheel object still selected, activate the **RBD Pin Constraint button** from the **Rigid Bodies Shelf**.



The **Help Prompt** will ask for the **Dynamic Object** for the Pin Constraint to be selected. Press **ENTER** to accept the selected water_wheel object. The **Help Prompt** will then ask for a location for the Pin Constraint. Again, press **ENTER** to **position** the **Pin Constraint** at the **scene origin**. As the Viewer will still be at DOP Level, the **parameters** of the **RBD Pin Constraint DOP** can be accessed by pressing **p** with the **mouse over the Viewer**.

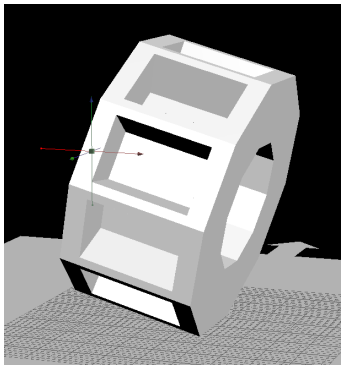


Under the **Op Dialog** section of the **parameters**, **RMB** on the **Constrained Location parameter**, and **Paste** the **Copied References**. This will automatically position the RBD Pin Constraint at the same position as the water_wheel.

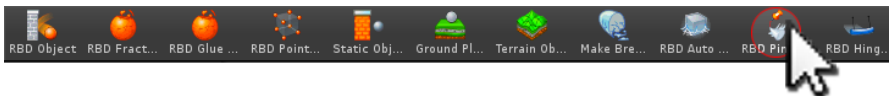
Edit the **Constrained Location x** parameter to include a position offset:

Constrained Location (x) **ch("/obj/water_wheel/tx")-3**

This will move the position of the constraint slightly to one side of the wheel causing it to swing and hit the ground plane.



Rewind the simulation, and from the **Rigid Bodies Shelf** tools activate a second **RBD Pin Constraint**.

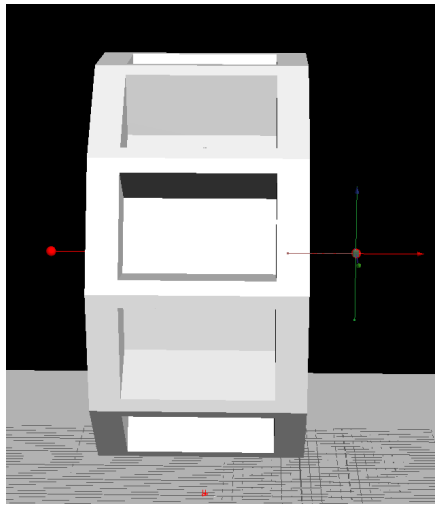


As before, the **Help Prompt** will ask for the **Dynamic Object** for the Pin Constraint to be selected. Select the water_wheel object and press ENTER. The **Help Prompt** will then ask for a location for the Pin Constraint. Again, press **ENTER** to **position** the **Pin Constraint** at the **scene origin**.

Activate the **parameters** for this **second RBD Pin Constraint**, and again, **paste the Copied References** into the **Constrained Location parameter**, this time specify a Constrained Location x parameter of:

Constrained Location (x) **ch("/obj/water_wheel/tx")+3**

This will create a spindle allowing for the water wheel to turn.



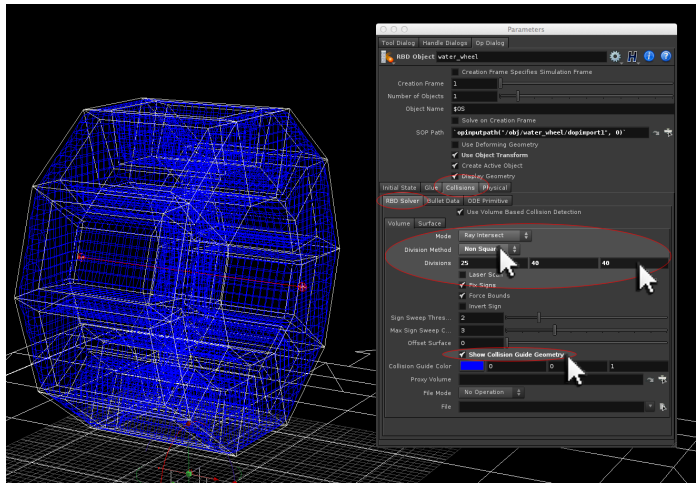
CONFIGURING THE WATER WHEEL

Still at **DOP Level**, **select** the **dynamic water_wheel** and press **ENTER** with the mouse over the viewer. **Activate** the **parameters** of the **water_wheel RBD Object DOP** by pressing **p** with the mouse over the Viewer.

Under the **Collisions > RBD Solver** section of the **parameters**, activate the **Show Collision Guide Geometry option** and specify:

Division Method	**Non Square**		
Divisions	**25**	**40**	**40**

This will improve the collision accuracy of the water wheel.



With this set, **deactivate** the **Show Collision Guide Geometry option**.

Under the **Physical** section of the **parameters** specify:

■	**Compute Mass**
Mass	**100**

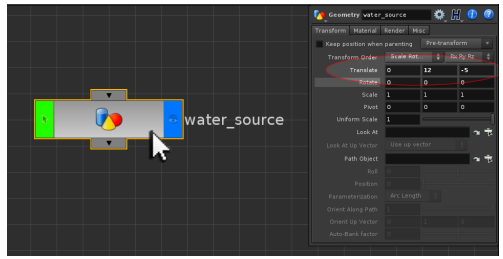
This will make the water wheel light enough to respond correctly to any objects that hit it and cause it to turn. As a final step, set the **Solver Engine** of the **Rigid Body Solver** to **RBD**.

See file fruit_cannon_stage1.hipnc

CREATING A SIMPLE WATER SYSTEM

Return back to Object Level, and create a new piece of geometry called water_source. In its Translate parameter specify:

Translate	**0**	**12**	**-5**

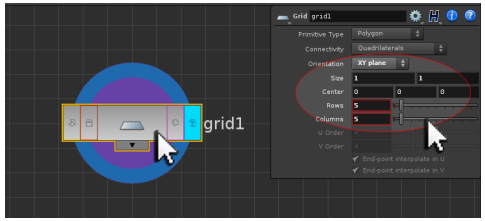


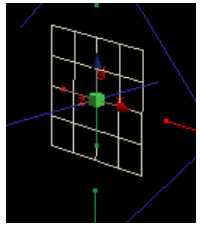
This will position the water_source object above and behind the water wheel. This can be used as a source for generating some **FLIP Particle Fluid**.

GEOMETRY TYPES FOR EMITTING FLUIDS

Inside the **water_source object**, **delete** the **File SOP** and create a **Grid SOP**. In its **parameters** specify:

Orientation	**XY Plane**	
Size	**1**	**1**
Rows	**5**	
Columns	**5**	



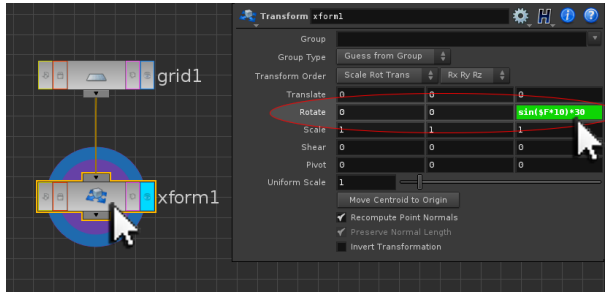


This will create a simple surface from which the FLIP Particle Fluid can emit from.

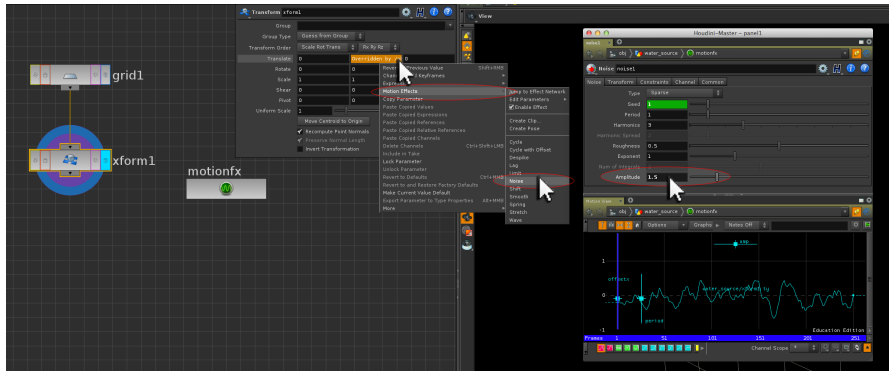
IMPORTANT NOTE: As this will be a continuous stream of FLIP Particle Fluid, it is better to use flat geometry as its source rather than geometry with volume (such as a Sphere SOP). Doing so helps prevent the generation of excess fluid.

As with all simulation work, **the source geometry should be doing something interesting** in order to create a visually dynamic result. Append to the **Grid SOP** a **Transform SOP**, and in its **parameters** specify:

Rotate	**0**	**0**	**sin($F*10)*30**



This will cause a rotational rocking movement to the grid. **Motion Effects** can also be added to the **Translate Y parameter** to give additional fluctuation to the movement of the grid.



RMB on the **Translate Y parameter**, and from the **Motion Effects** menu choose **Noise**. In the resulting **Motion Effects dialog window** specify:

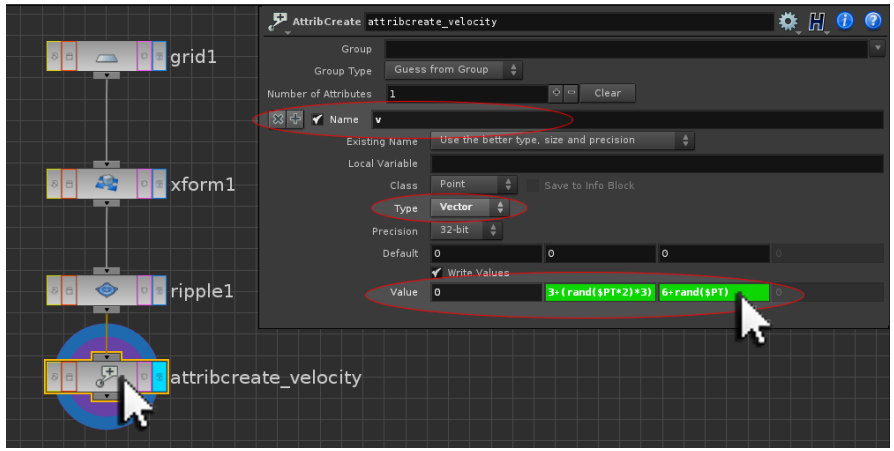
Amplitude	**1.5**

The surface of the grid can also be deformed, again in order to build variation into it as a source for the FLIP Particle Fluid. **Append** to the Transform SOP, a **Ripple SOP**. In its **parameters** specify:

Scale	**3**		
Wave Speed	**10**		
Random Seed	**$T**		
Up Direction	**0**	**0**	**1**

DEFINING FLIP PARTICLE FLUID VELOCITY

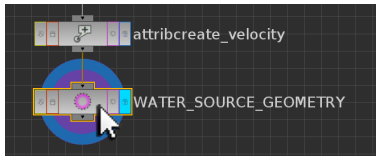
Before activating the water_source object as an emitter for FLIP Particle Fluids, it is also a good idea to assign a **velocity attribute** to the grid in order to control the direction and flow of the resulting fluid. Without such information, the fluid would simply pour onto the ground plane under the force of gravity, rather than being propelled forward over the water_wheel.



To the ouput of the Ripple SOP, append an **Attribute Create SOP**. In it **parameters** specify:

Name	**v**		
Type	**Vector**		
Value	**0**	**3+(rand($PT*2)*3)**	**8+rand($PT)**

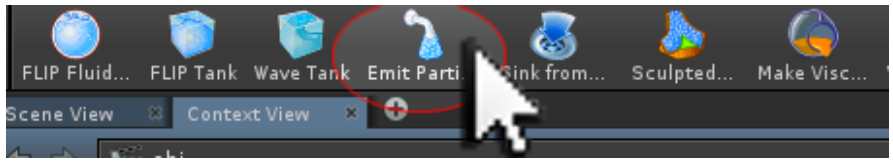
This will add per-point randomised velocity information in both the Y and Z axis. As a final step, append a **Null SOP** renaming it to **WATER_SOURCE_GEOMETRY**.



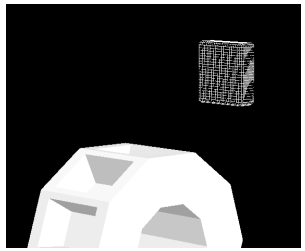
See file fruit_cannon_stage2.hipnc

ACTIVATING THE FLIP PARTICLE FLUID

Return to **Object Level** and **Maximise** the **Viewer** and un-stow the **Shelves**.

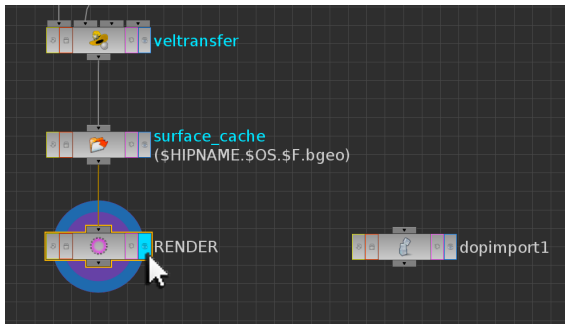


With the **water_source object selected**, **LMB** the **Emit Particle Fluid** button from the **Particle Fluids shelf**. When prompted to select the Fluid or Fluid Box, simply press **ENTER** to complete and configure the Fluid setup.

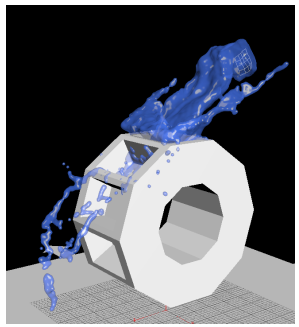


Initially the **water_source object** will appear as a volume mesh, and no fluid will appear in the Viewer when PLAY is pressed.

Inside the **water_source object**, reactivate the **Display and Render Flags** for the **WATER_SOURCE_GEOMETRY Null SOP**. This will allow the source grid to still be seen in the Viewer.



Similarly, go **inside** the automatically created **particle_fluid object** and set the **Display Flag** to the **RENDER Null SOP**. This will allow the fluid to be seen in the Viewer.



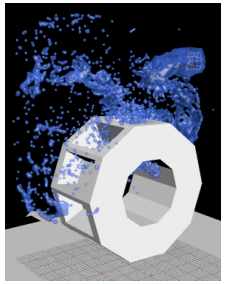
When **PLAY** is pressed, the fluid can be seen pouring over the water_wheel object; however currently it does not cause it to turn as anticipated. This is due to a default setting inside the FLIP Solver not to influence other dynamic objects.

Inside the **AutoDopNetwork**, locate the **FLIP Solver** and under the **Volume Motion > Solver** section of its **parameters** specify:

Feedback Scale **1**

NOTE: The **Cog Menu** of the **FLIP Solver** can be activated to set this adjustment as a **permanent default value** for all future FLIP Solver work.

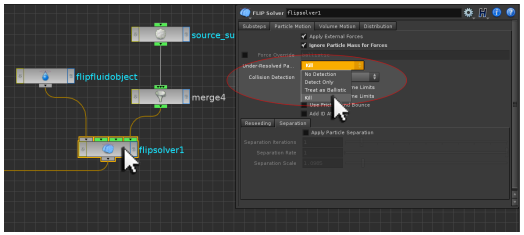
Now when **PLAY** is pressed, the water_wheel turns as expected under the force of the fluid impacting with it; however the fluid wildy explodes upon contact with the water_wheel.



See file fruit_cannon_stage3.hipnc

PREVENTING FLIP FLUID EXPLOSIONS

FLIP Fluids explosions occur when there is a build up of fluid pressure especially when fluid particles get compressed into tight spaces of collision geometry. Adjusting settings in the **FLIP Solver DOP** and the **FLIP Object DOP** can help prevent such explosions. Under the **Particle Motion section** of the **parameters** for the **Flip Solver**, modify the **Under-Resolved Particles parameter** from **Use Extrapolated Velocity** to **Kill**.



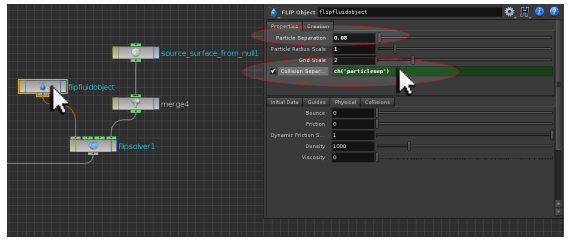
This will kill off any particles that the Flip Solver cannot resolve in terms of the simulation.

REFINING THE FLIP FLUID AESTHETIC

This **Particle Separation** parameter of the **FLIP Object DOP** controls the visual resolution of the fluid. The **smaller this value**, **the bigger the fluid system will appear**; however more particles are required to maintain a steady stream and calculation times will increase. The **larger this value** is, t**he smaller the fluid system will appear** and will generate less particles and less calculation time as a result. This parameter should therefore be adjusted to create an appropriate scale of the fluid relative to the scene being created. In the **parameters** for the **FLIP Fluid Object DOP** specify:

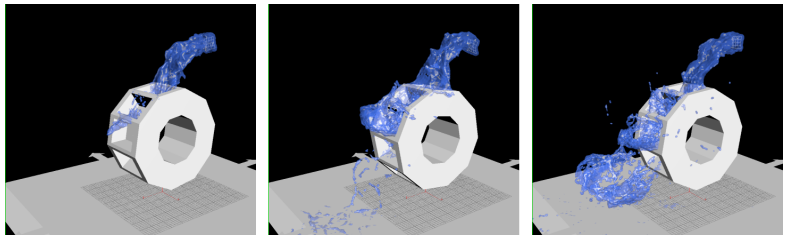
Particle Separation	**0.08**
Particle Radius Scale	**1**

This will reduce the size of the fluid particles, making its visual resolution appear bigger. The **Particle Radius Scale** controls the overall **volume** of the fluid.



On the **Flip Object DOP**, also **activate** the **Collision Separation parameter**, and **Channel Reference** the **Particle Separation parameter into it**. This will create a **higher resolution collision field** that can also help the FLIP Solver resolver the collisions more accurately.

When **PLAY** is pressed, the fluid is more controlled; however the **water_wheel** now appears **too responsive** to the impact of the fluid upon it. **Increasing** the **Physical > Mass** parameter of the **water_wheel RBD Object DOP** to **2000** will make it much heavier and more resistant to being moved by the fluid.



NOTE: Generating a Flip Book Render of the sequence is a useful way to be able to preview the simulation in real-time if Viewer based playback becomes too slow.

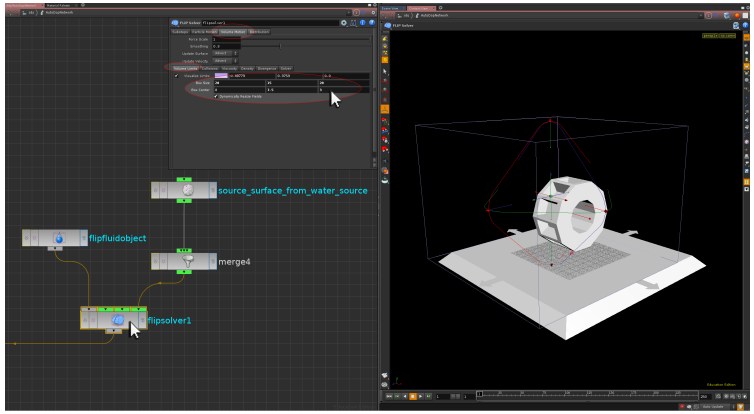
SETTING THE FLUID BOUNDING AREA

The bounding volume of FLIP Fluid network can also be adjusted relative to the scene being created. Only FLIP Fluids within this bounding area are calculated, with fluid particles being culled once they go outside this region.

NOTE: This adjustment should ultimately be done relative to the scene camera.

Inside the **AutoDopNetwork**, select the **FLIP Solver DOP** and under the **Volume Motion > Volume Limits** section of the **parameters** specify:

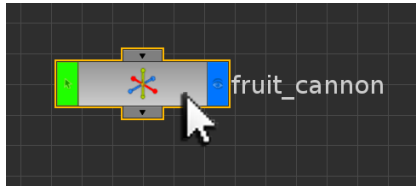
Box Size	**20**	**15**	**20**
Box Center	**0**	**7.5**	**3**

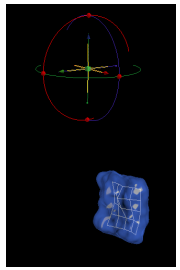


This will reduce the size of the bounding container of the fluids. **See file fruit_cannon_stage4.hipnc**

ADDING IN THE FRUIT

RBD Point Instancing can also be used to create a firing cannon effect with dynamic geometry, as well as simply instancing geometry onto point data. In this example a **Null Object** can be manually created as a RBD Point Object so it can fire out pieces of fruit over the FLIP Fluid simulation. At **Object Level**, create a **Null Object**, and rename it to **fruit_cannon**.

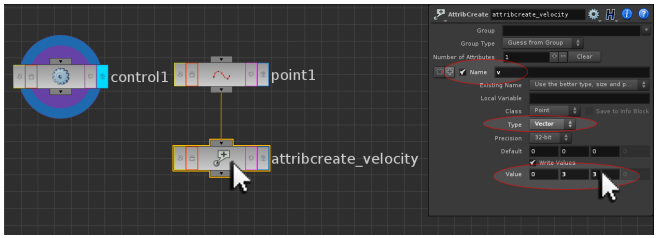




In the **parameters** for the Null object specify:

Translate	**0**	**15**	**-4**

This will position the Null object above but in front of the FLIP Fluid. At **Geometry Level**, append an **Attribute Create SOP** to the **Point SOP**.



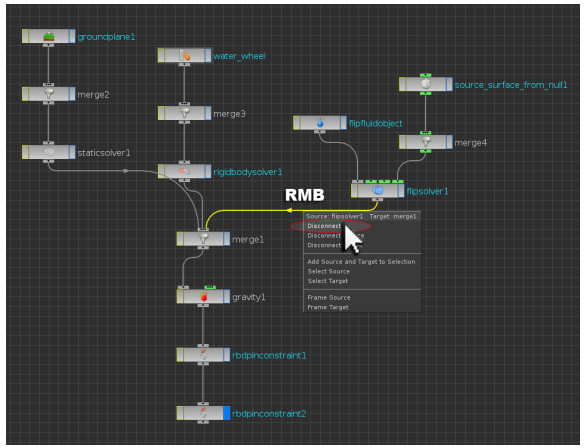
In its **parameters** specify:

Name	**v**		
Type	**Vector**		
Value	**0**	**3**	**3**

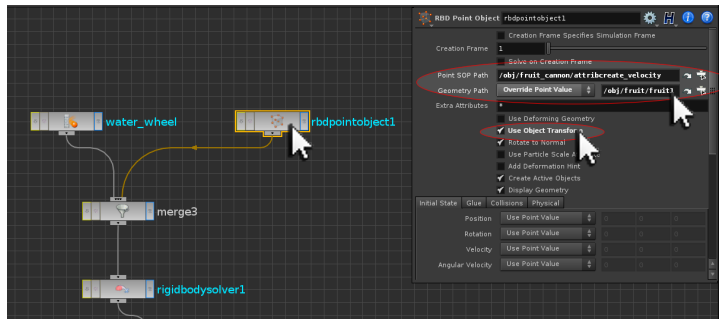
This will prime the fruit_cannon to fire out fruit over the fluid.

IMPORTANT NOTE: Null Objects are not recognised as geometry objects by the Shelf Tools; however can be called into DOPs by manually configuring the appropriate nodes.

Inside the **AutoDopNetwork**. **temporarily disconnect** the **output** of the **Flip Solver DOP** from the main network. This will save simulating the fluid whilst the fruit_cannon is being configured.



Alongside the water_wheel RBD Object DOP, create a **RBD Point Object DOP** and wire it into the **Merge DOP** that goes into the **Rigid Body Solver DOP**.



In the **parameters** for the **RBD Point Object DOP**, specify:

Point SOP Path **/obj/fruit_cannon/attribcreate_velocity**
Geometry Path **Override Point Value** **/obj/fruit/fruit1**
☑ **Use Object Transform**

When **PLAY** is pressed, a single kiwi fruit is fired out from the centre of the Null Object, hitting the water wheel as it falls.

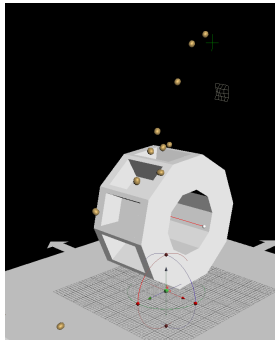
THE CREATION FRAME

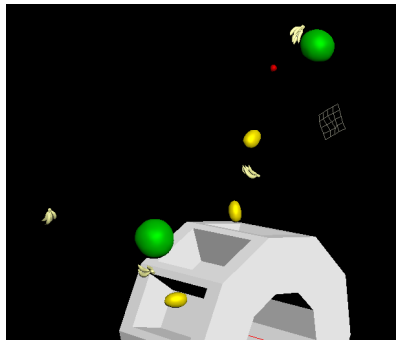
Dynamic Objects will be generated on the **Creation Frame** specified in their creation DOP. In the case of the RBD Point Object DOP, the default Creation Frame of 1 results in the DOP generating a single dynamic kiwi fruit at frame 1. Adding an expression into the Creation Frame parameter can modify this behaviour. In the **parameters** of the **RBD Point Object DOP** specify:

Creation Frame **if ($SF % 10 == 1, $SF, 0)**

When **PLAY** is pressed, a stream of kiwis fires from the Null Object on every 10th frame.

The **RBD Point DOP** can also be configured to pick a different piece of fruit whenever its Creation Frame is activated. Modifying the Geometry Path reading in the apple geometry can do this.

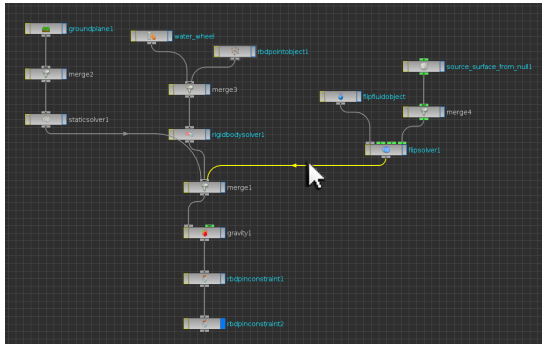


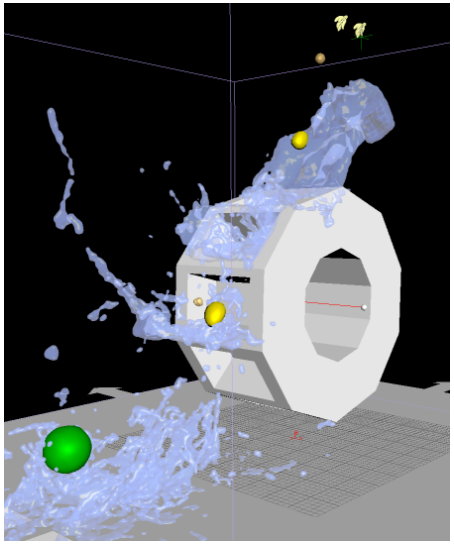


In the **parameters** for the **RBD Point Object DOP specify**:

Geometry Path **/obj/fruit/fruit`int(rand($SF)*8)+1`**

When **PLAY** is pressed, a stream of different fruit is fired from the Null Object every 10th frame. With the fruit cannon configured, the particle fluid water can be added back into the DOP Network.



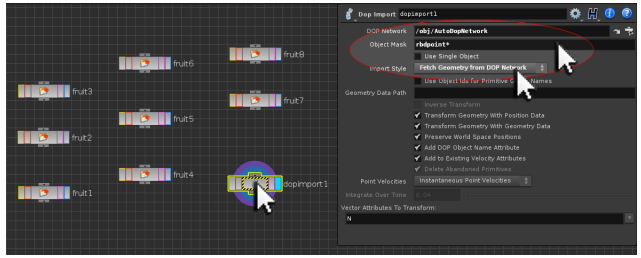


See file **fruit_cannon_stage5.hipnc**

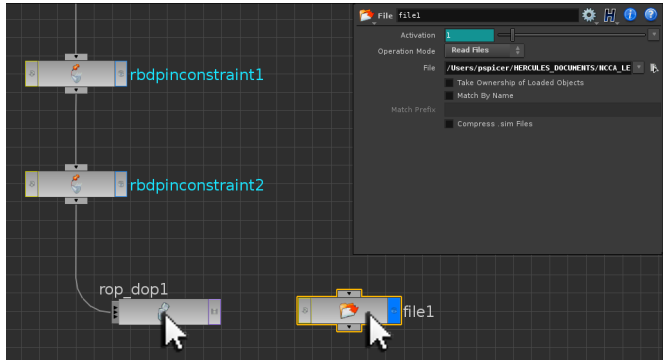
FURTHER REFINEMENTS

Return back to Object Level and examine the simulation. Currently the fruit does not appear at Object Level. **Activate** the **Display Flag** for the **Fruit Object**, and at is **Geometry Level**, create a **DOP Import SOP**. In its **parameters** specify:

DOP Network	**/obj/AutoDopNetwork**
Object Mask	**rbdpoint***
Import Style	**Fetch Geometry from DOP Network**



As a final step, the simulation can be rendered out at a series of .sim files at DOP Level in order to help speed up final rendering.



See file fruit_cannon_end.hipnc