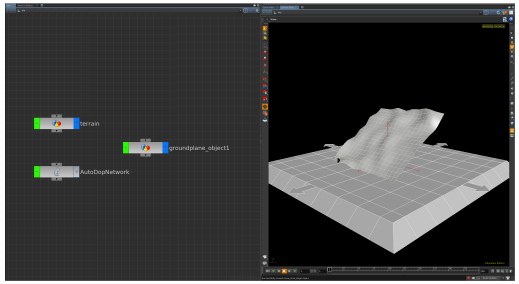
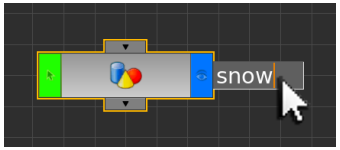
SIMPLE AVALANCHE

Open the scene **snow_stick_begin.hipnc**. This scene contains a piece of terrain geometry configured as a static terrain object at DOP Level. A ground plane has also been added. This example will look at generating FLIP Fluids on the terrain and sticking specific parts of the fluid to the terrain to create a simple avalanche effect.



CONFIGURING THE FLIP FLUIDS SNOW

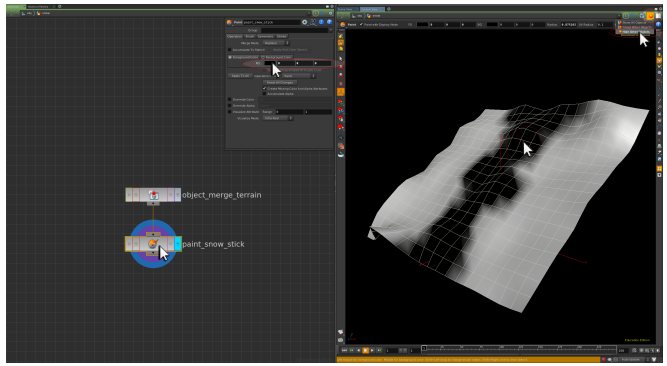
At Object Level, create a **new geometry object** and **rename it** to **snow**.



Inside this object, **delete** the **default File SOP**, and in its place create an **Object Merge SOP**. In the **parameters** for the **Object Merge SOP** specify:

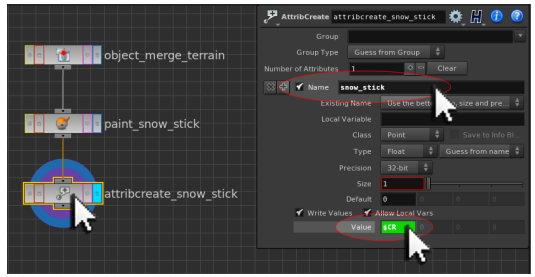
Transform	**Into This Object**
Object 1	**/obj/terrain/TERRAIN**

In the **Context View**, **hide** the **other scene geometry**, and **append** a **Paint SOP** to the Object Merge SOP. **Paint** a **black streak down the centre of the terrain**. When the fluid is created, the white areas of the terrain can be used to control where the snow will stick; the black areas will control where the snow will slide.



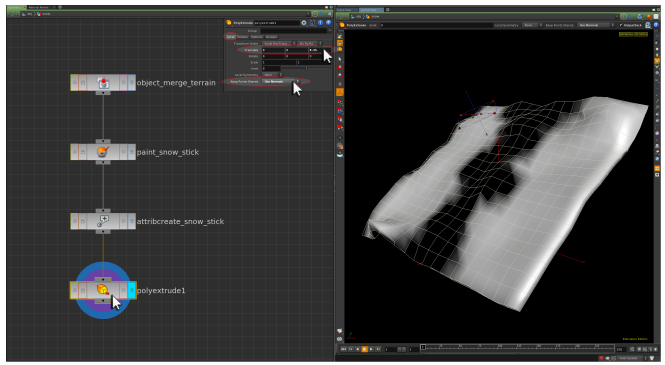
To the Paint SOP, **append** an **Attribute Create SOP**. In its **parameters** specify:

Name	**snow_stick**
Value	**$CR**

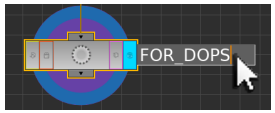


To the Attribute Create SOP **append** a **PolyExtrude SOP**. In its **parameters**, specify:

Local >			
Translate	**0**	**0**	**0.25**
Keep Points Shared	**Use Normals**		
Options >			
☑	**Output Back**		

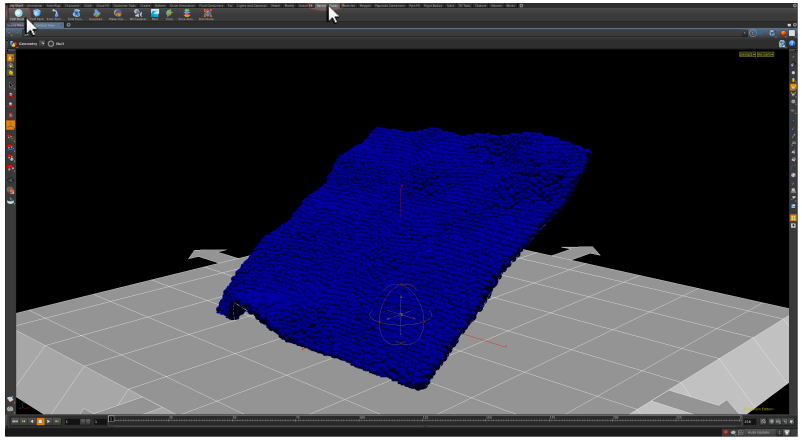


This will create volume to the painted terrain geometry that can be converted into FLIP Fluids. As a final step append a Null SOP and rename it to FOR_DOPS.

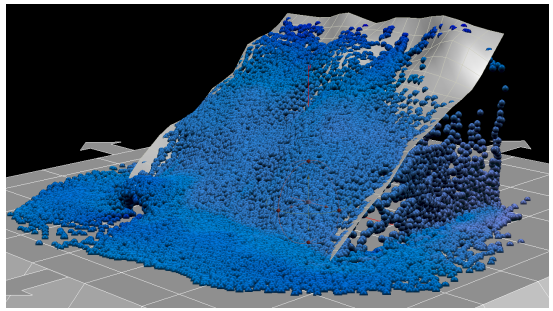


See file snow_stick_stage1.hipnc

Return to **Object Level**, and with the **snow object selected activate the Shelves**. From the **Particle Fluids Shelf LMB** the **FLIP Fluid from Object** Shelf Button.



This will convert the extruded snow geometry into a FLIP Fluid Particle System. When **PLAY** is pressed, the fluid particles uniformly slide down the terrain geometry onto the ground plane.

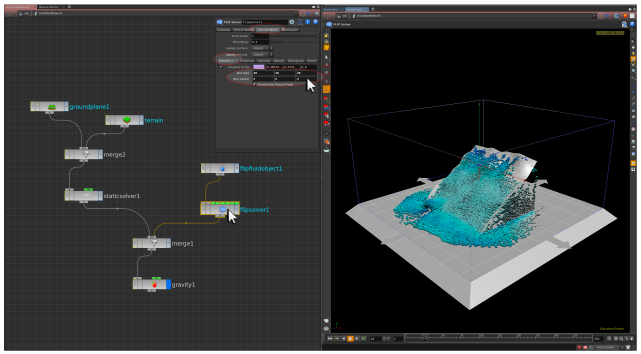


MAKING THE PARTICLES STICK

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

Box Size	**20**	**10**	**20**
Box Center	**0**	**5**	**0**

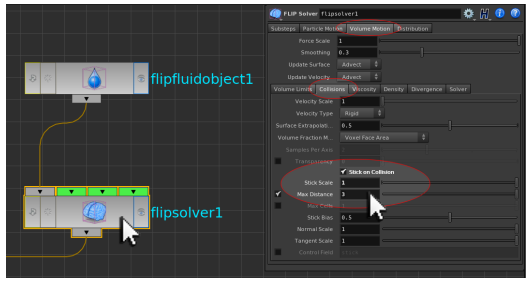
This will localize the bounding regions of the FLIP Fluids container to surround the immediate scene.

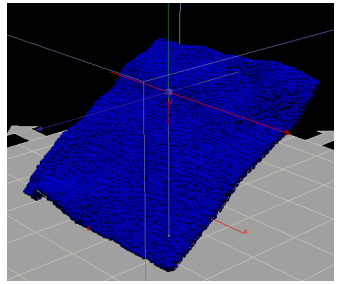


Under the **Volume Motion > Collisions** section of the **parameters** specify:

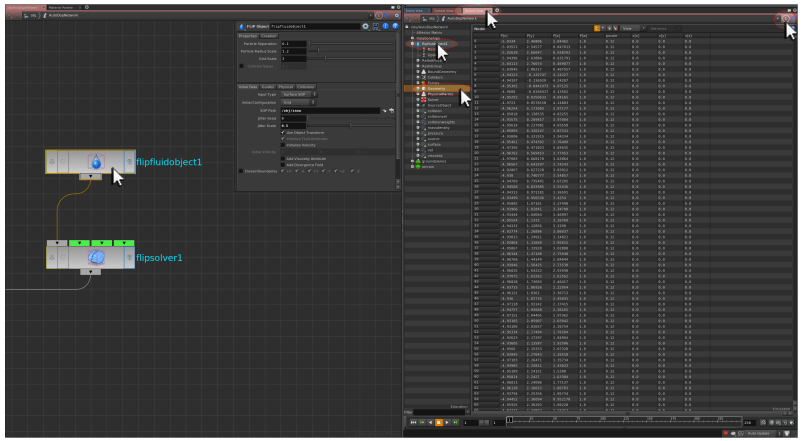
☑	**Stick on Collision**
Stick Scale	**1**
Max Distance	**3**

When **PLAY** is pressed, the **FLIP Fluid particles stick** to the terrain.





By default, **FLIP Fluid particles will not inherit** any **custom attributes** assigned to the source geometry. Activating a **Details View** over the **Viewer**, and setting its **Pane Link button to 1** can verify this.

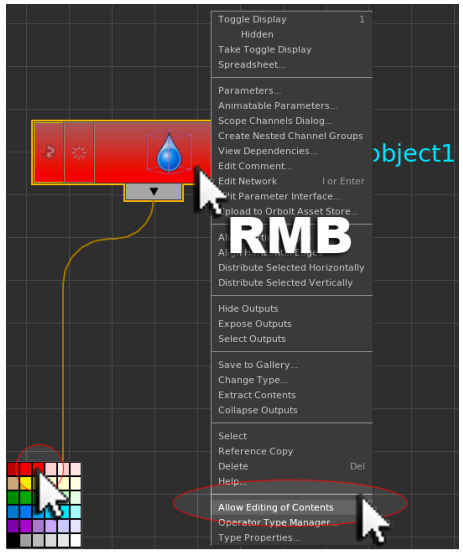


From the **/obj/AutoDopNetwork listing** activate the **flipfluidobject1/Geometry** listing to see the particle attributes. Only **Position**, **PScale**, and **Velocity** are created as FLIP Fluid particle attributes.

MINING DIGITAL ASSETS

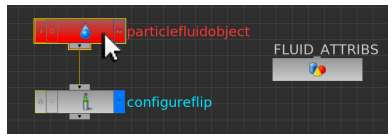
Custom Attributes can be **added to FLIP Fluid Particles**, by modifying an inner network of the **flipfluidobject1 FLIP Object DOP**. Somewhere inside this Digital Asset, the snow geometry is converted to particles, and at this part of the inner network, custom attributes can be assigned to the particles.

With the **mouse over** the **Network Editor**, **press c** to **activate** the **Color Swatch**. With the **flipfluidobject1 DOP selected**, **colour the node red** (to indicate it has been internally modified) and **RMB** on the node and choose **Allow Editing of Contents…**

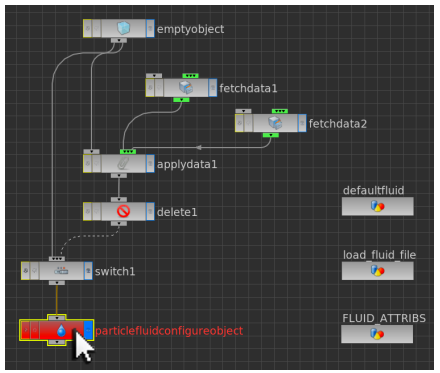


NOTE: Extreme care should be taken when modifying inner contents of a Digital Assets created by Shelf Tools, as this may break interaction with other shelf tools.

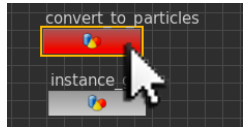
Go inside the (now unlocked) **flipfluidobject1 node** and **repeat this process** to **unlock and colourise** the inner **Particle Fluid Object DOP**.



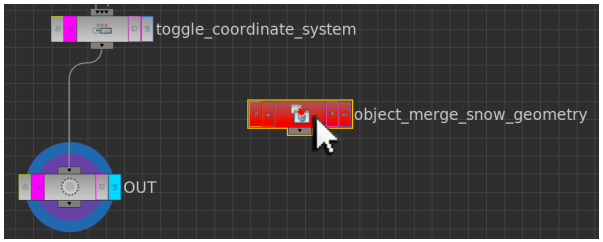
Go inside the (now unlocked) **Particle Fluid Object DOP node** and **repeat this process** to **unlock and colourise** the inner **Particle Fluid Configure Object DOP**.



Go inside the (now unlocked) **Particle Fluid Configure Object DOP node** and **colourise** the inner **convert_to_particles SOP Network**.



Inside the (now colourised) **convert_to_particles SOP Network**, create an **Object Merge SOP** near the end node of the network chain. Again **colour it red** to help **indicate** that it is **a custom addition** to the inner Digital Asset networks.



In the **parameters** of this **Object Merge SOP** specify:

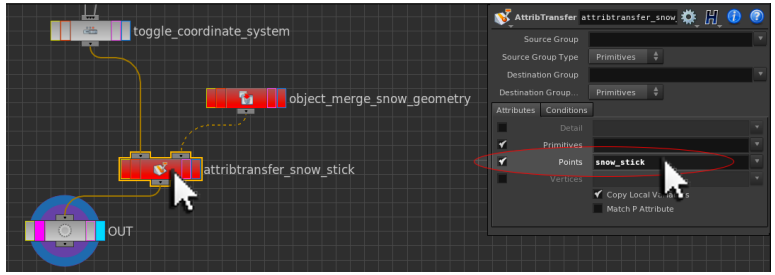
Transform	**Into This Object**
Object 1	**/obj/snow/FOR_DOPS**

This will read in the snow geometry so that its snow_stick attribute can be transferred onto the FLIP Fluids particle system before they are processed by DOPs.

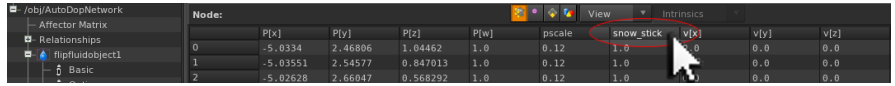
Append an **Attribute Transfer SOP** to the **toggle_coordinate_system Switch SOP** and in its **parameters** specify:

Attributes >

☑ **Points**	**snow_stick**



This will **transfer** the **snow_stick point attribute** onto the **fluid particle system**. When the **Network Editor** is returned back to the main **DOP Level**, this attribute addition can be confirmed in the **Details View**.



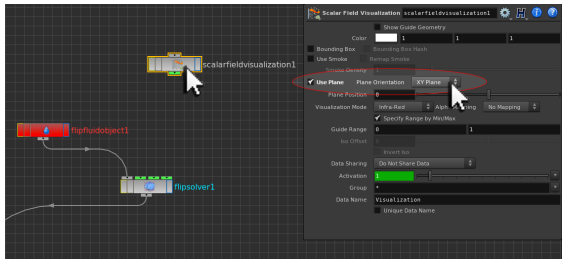
See file snow_stick_stage2.hipnc

CREATING A STICKINESS CONTROL FIELD

While the assigned **Stick on Collision activation** for the **FLIP Fluid particles creates a uniform sticking effect**, it is now possible (**with the addition of the snow_stick attribute),** to control the stickiness of the particles using a **Control Field**.

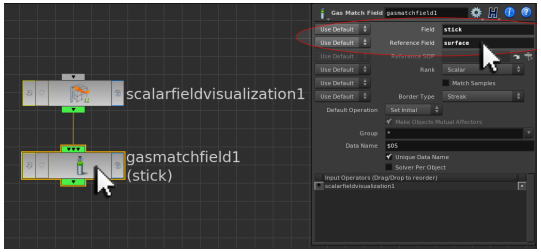
A **Control Field** is an **attribute** assigned to an **entire Volume Container** rather than individual particles. In order to use the **snow_stick particle attribute** to stick the fluid particles to the terrain, **it can be converted** into a **Control Field** affecting the entire Volume Container.

At **DOP Level**, alongside the FLIP Solver DOP **create a Scalar Field Visualization DOP**. This node will allow for examination of the Control Field when it has been configured. **Activate** its **Use Plane parameter** tick box.



Append to the **Scalar Field Visualization DOP** a **Gas Match Field DOP**. This will initialize the Control Field. In its **parameters** specify:

Field	**stick**	(the control field name)
Reference Field	**surface**	(an existing field used as comparison)

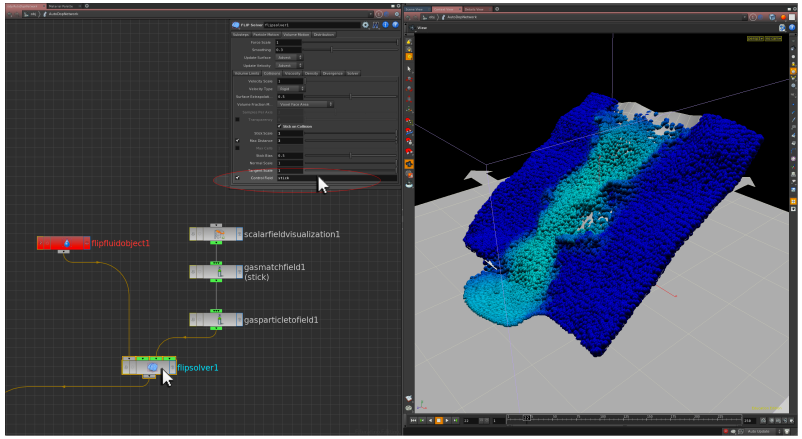


NOTE: The **FLIP Solver** already **anticipates** the **stick Field name**; however an arbitrary name could also be used.

Append to the **Gas Match Field DOP**, a **Gas Particle to Field DOP**. This will transfer the particle attributes over to the initialized Control Field. In its **parameters** specify:

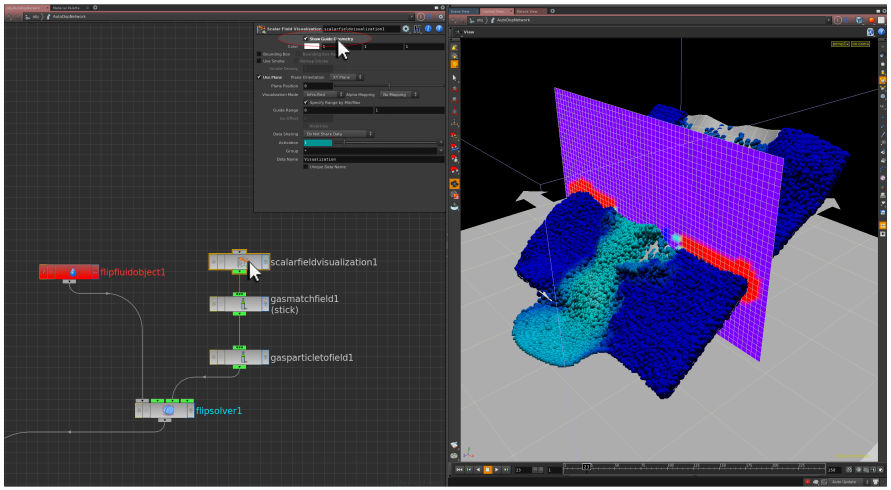
Destination Field	**stick**
Particles	**Geometry**
Attribute	**snow_stick**

As a final step, **wire the output** of the **Gas Particle to Field DOP** into the **third input** of the **FLIP Solver DOP** (which controls the Volume Velocity of the FLIP Fluids).



When the **FLIP Solver > Volume Motion > Collisions > Control Field parameter tick box** is activated, only the painted range of particles slide when **PLAY** is pressed.

The **snow_stick Control Field** can also be **visualized** by activating the **Show Guide Geometry tick box** of the **Scalar Field Visualization DOP** and pressing **PLAY**.



See file snow_stick_end.hipnc

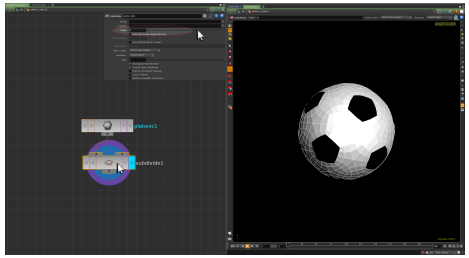
DENTING A FOOTBALL

While converting geometry to a stiff Cloth Object can create object denting, more bespoke denting control can be given to a scene by using the **SOP Solver Mechanism** instead. The **SOP Solver Mechanism** is **a DOP Level implementation** of the **Solver SOP**, where dynamic history can be recorded on objects.

In a **new Houdini scene**, create a **Ground Plane** from the **Rigid Bodies Shelf**. From the **Create Shelf**, activate a **Platonic Solids object** and set its **type** to **Soccer Ball**.

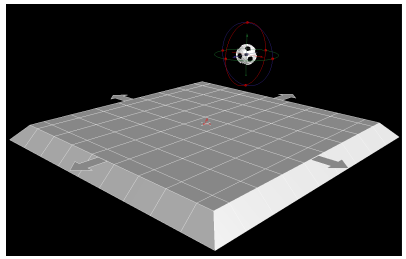
At the **Geometry Level** for the Soccer Ball, append a **Subdivide SOP** specifying in its **parameters**:

Depth **2**



At **Object Level**, position the Soccer Ball:

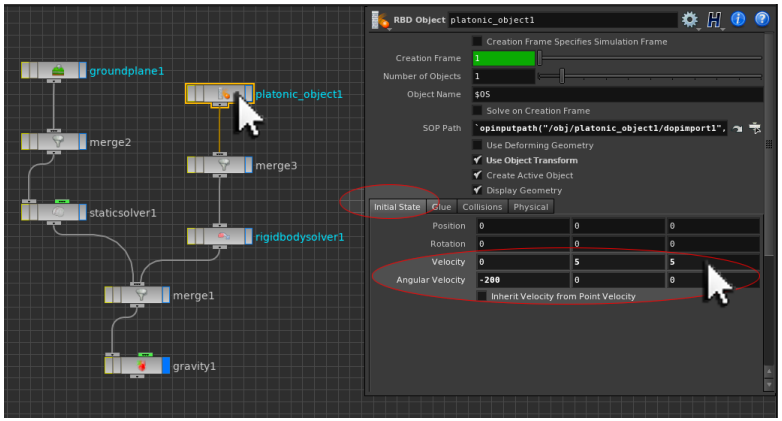
Translate **0** **5** **-5**



From the **Rigid Bodies Shelf**, **initialise** the **soccer ball** as a **RBD Object**.

At **DOP Level**, locate the **RBD Object DOP** reading in the soccer ball and in its parameters specify:

Initial State >			
Velocity	**0**	**5**	**5**
Angluar Velocity	**-200**	**0**	**0**
Physical >			
Bounce	**2**		

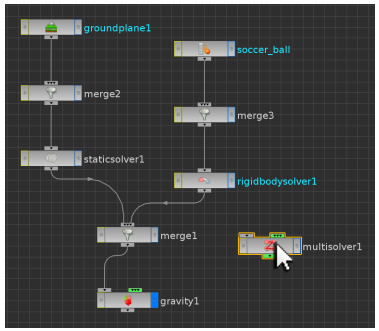


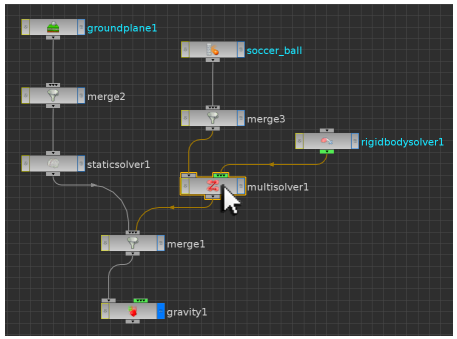
When **PLAY** is pressed the ball bounces across the ground plane.

THE MULTIPLE SOLVER

In order to create a SOP Solver Mechanism at DOP Level, a **Multiple Solver DOP** must be used. The **Multiple Solver DOP** allows for a **dynamic object** to be **processed** by **more than one solver**. **Combinations of solvers** can therefore be initiated on a single dynamic object. The **Multiple Solver DOP** is a **utility solver** similar to the **Blend Solver DOP** and the **Switch Solver DOP**.

Alongside the main network create a **Multiple Solver**. **Shake free** the **RBD Solver** from the network, and move the **Multiple Solver** into its place. Wire the RBD Solver into the **green data input** of the **Multiple Solver**.

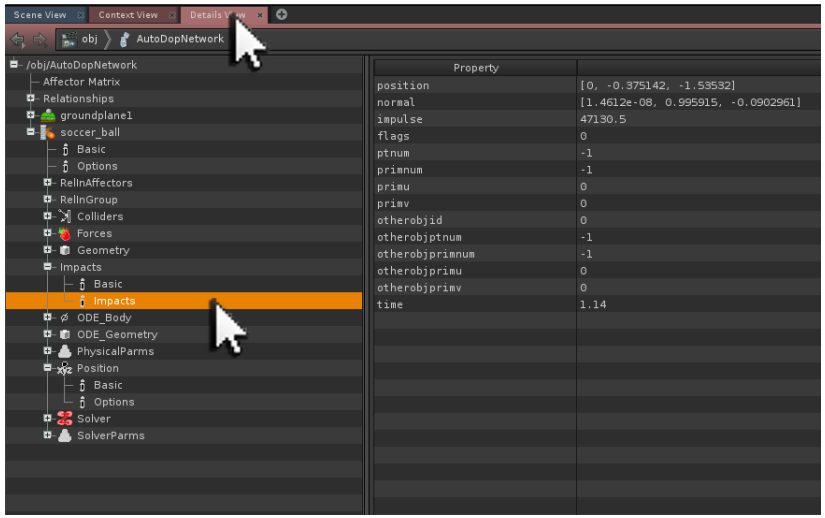




When **PLAY** is pressed the simulation runs as before; however the use of the Multiple Solver allows for additional solvers to be incorporated into the simulation of the soccer ball.

MODIFYING THE SOCCER BALL UPON IMPACT

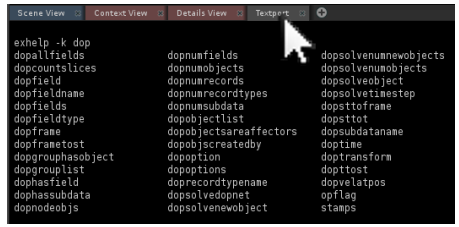
Play through the simulation until the ball first impacts with the floor. Over the **Viewer**, load up a **Details View** and set its **Pane Number** to **1**.



The **sub-data spreadsheet** for the simulation reveals an **Impacts** section that appears in the listing only on frame where the soccer ball impacts with the ground. The **Impacts section** contains a **position attribute** detailing the position of the impact in space.

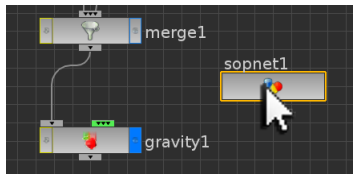
The actual visual impact frame may not quite match the frame number where the Impacts listing appears in the Details View. This is due to the **default settings** of the **Rigid Body Solver DOP**. This can however be refined at a later stage.

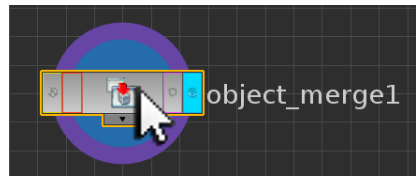
In order to extract data from the sub-data spreadsheet and use it elsewhere in the Houdini envornment, a number of **DOP specific expressions** can be called. A full list of these expressions can be seen by activating a **Textport** and typing the command **exhelp –k dop**.



The **dopfield expression** can be used to **extract** the correct **subdata**. **Typing exhelp dopfield** into the **Textport** will bring up the **Help Card** for this **expression**.

Alongside the main DOP network, create a **SOP Network**. Inside it create an **Object Merge SOP**.

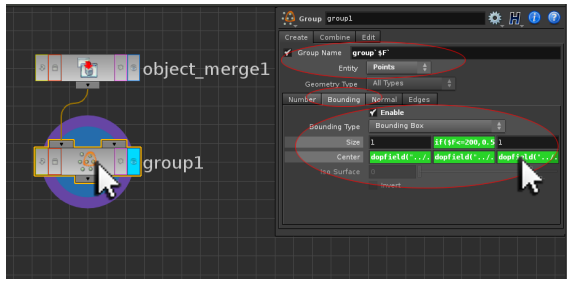




In the **parameters** for the **Object Merge SOP** specify:

Transform	**Into this Object**
Object 1	**/obj/AutoDopNetwork:platonic_object1/Geometry**

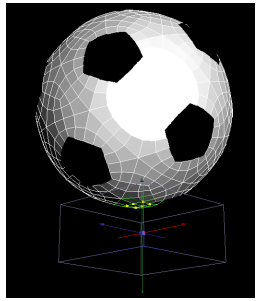
This will read only the dynamic soccer ball geometry from the DOP Network.



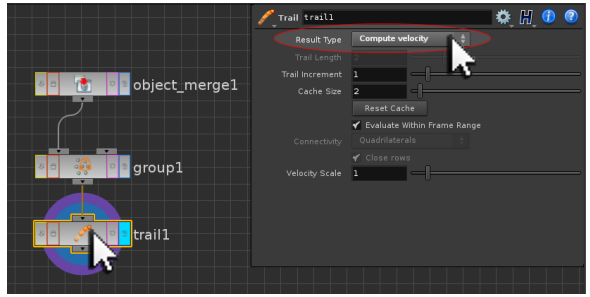
Append to the **Object Merge SOP** a **Group Geometry SOP**. In its **parameters** specify:

Group Name **group`$F`**
Entity **Points**
Bounding >
☑ **Enable**
Bounding Type **Bounding Box**
Size **1** **if($F<=125,0.5,0)** **1**
Centre
x
dopfield("../..","platonic_object1","Impacts","Impacts",0,"positionx")
y
dopfield("../..","platonic_object1","Impacts","Impacts",0,"positiony")
z
dopfield("../..","platonic_object1","Impacts","Impacts",0,"positionz")

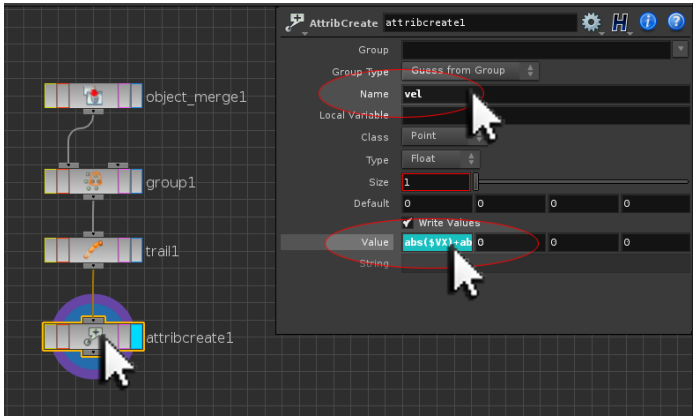
When **PLAY** is pressed, a bounding box grouping impact points on the soccer ball will jump into position whenever the ball impacts with the ground.



NOTE: A separate group is being formed for every impact, allowing for the geometry of the soccer ball to be modified at these moments. After 125 frames this automated grouping will stop due to the if() statement expression setting the size of the bounding box to zero.



Append to the **Group Geometry SOP** a **Trail SOP**, setting the **Result Type** parameter to **Compute Velocity**.



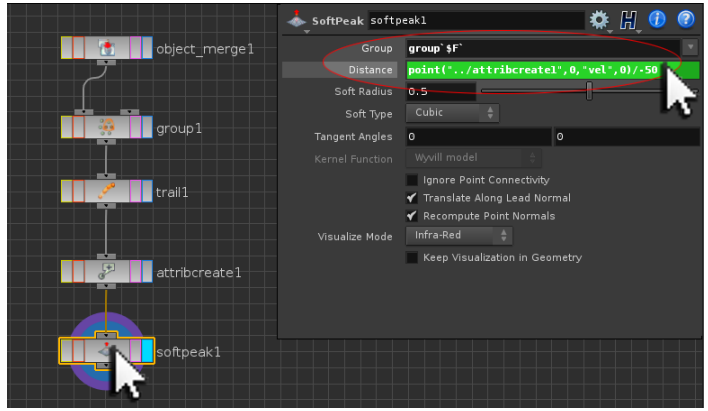
Append to the **Trail SOP** an **Attribute Create SOP**. This can be used to store combined velocity values as a custom attribute. In the **parameters** for the **Attribute Create SOP** specify:

Name	**vel**			
Value	**abs($VX)+abs($VY)+abs($VZ)**	**0**	**0**	**0**

This combined velocity value will be used to automatically determine the amount of denting that will take place.

THE SOFT PEAK SOP

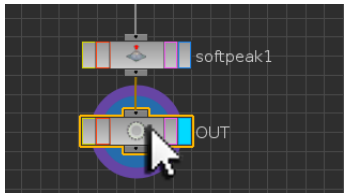
Denting of the soccer ball can be done by using a Soft Peak SOP. By default this operator will not accept custom attribute mappings (ie vel -> $VEL) to drive its parameters. Custom attribute values can however be accessed by using the point() expression. **Append** to the **Attribute Create SOP** a **Soft Peak SOP**.



In its **parameters** specify:

Group	**group`$F`**
Distance	**point("../attribcreate1",0,"vel",0)/-50**

Now the harder the impact is of the soccer ball on the ground plane, the more the Soft Peak SOP will dent the soccer ball.



As a final step, append a **Null SOP** to the end of the network chain, and rename it to **OUT**.

CONFIGURING THE SOP SOLVER

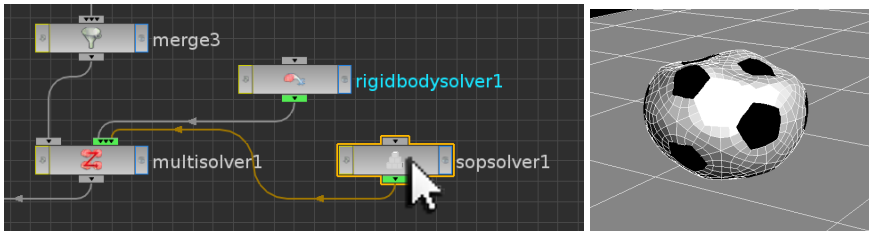
Return back to DOP Level, and create a SOP Solver. Wire it into the green data input of the Multiple Solver alongside the RBD Solver. As more than one solver is now being used for the soccer ball, a parameter on each solver must be activated to prevent data naming problems. For both the **Rigid Body Solver** and the **SOP Solver** specify:

☑ **Unique Data Name**

In the **parameters** for the **SOP Solver** specify:

☑ **Use External SOP**

SOP Path **/obj/AutoDopNetwork/sopnet1/OUT**

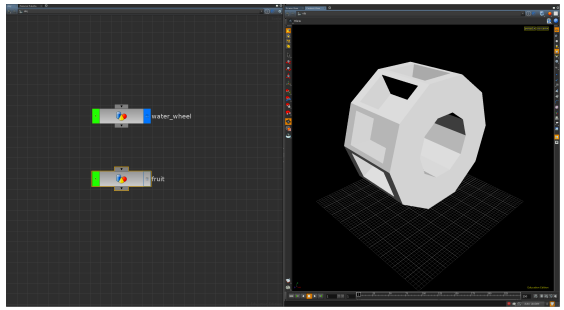


Now when **PLAY** is pressed and the simulation calculated, the soccer ball deforms (and remains deformed) with every impact.

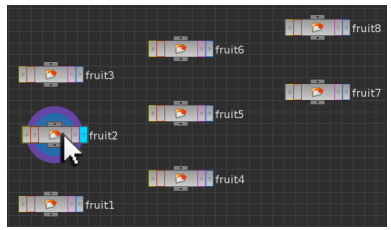
See file soccer_ball_sop_solver.hipnc

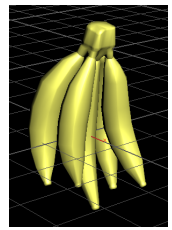
WATER WHEEL FRUIT CANNON

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.

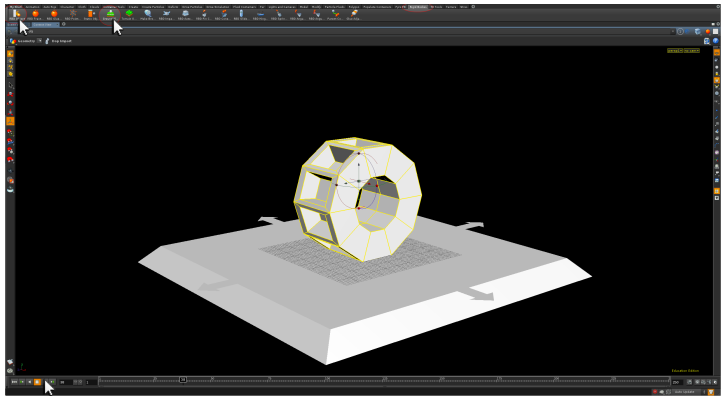




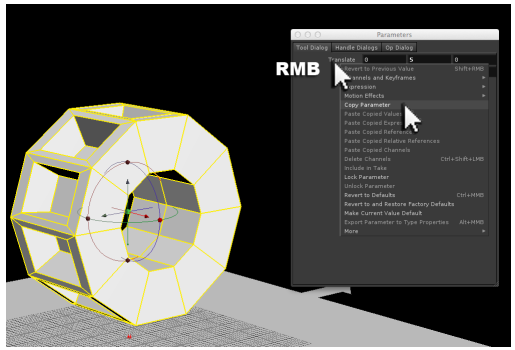
In this example, both fruit and water will be fired at the water_wheel causing it to spin.

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.



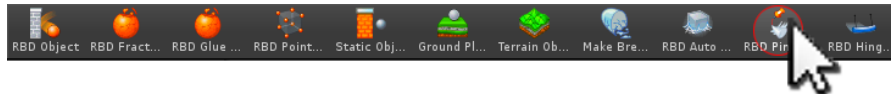
With the **water_wheel** object **selected**, press the **RBD Object button**. When **PLAY** is pressed, the water_wheel falls and lands on the ground plane.



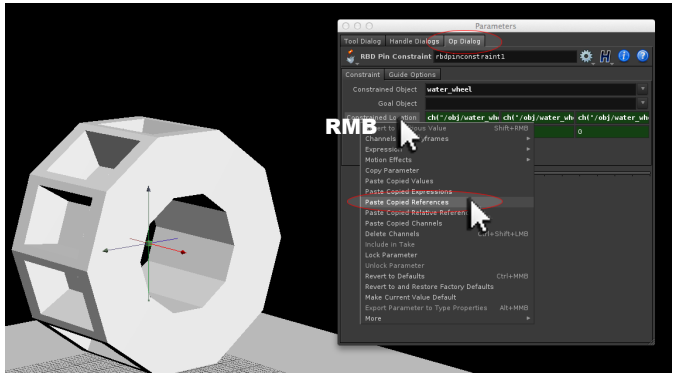
With the **water_wheel** object still **selected** in the **Viewer**, press **p** to activate a **floating Parameters Window**, and from the **Tool Dialog section**, **RMB copy the Translate parameter**.

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**.

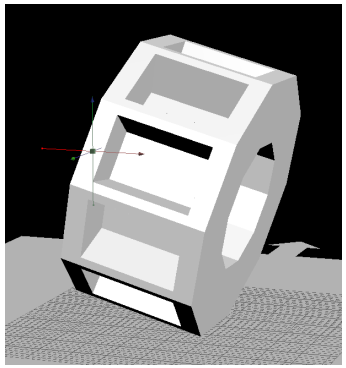


Under the **Op Dialog** section of the **parameters**, **RMB** on the **Constrained Location parameter**, and **Paste** the **Copied References**. This will automatically position the RBD Pin Constraint at the same position as the water_wheel.

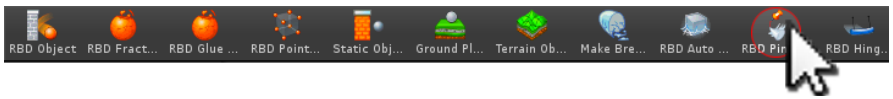
Edit the **Constrained Location x** parameter to include a position offset:

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**.

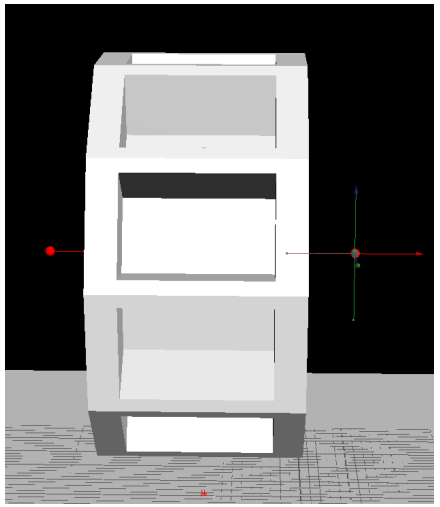


As before, the **Help Prompt** will ask for the **Dynamic Object** for the Pin Constraint to be selected. Select the water_wheel object and press ENTER. The **Help Prompt** will then ask for a location for the Pin Constraint. Again, press **ENTER** to **position** the **Pin Constraint** at the **scene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



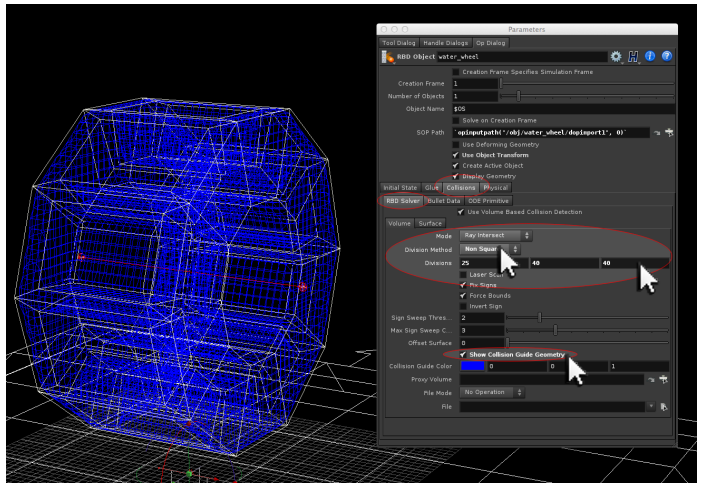
CONFIGURING THE WATER WHEEL

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**

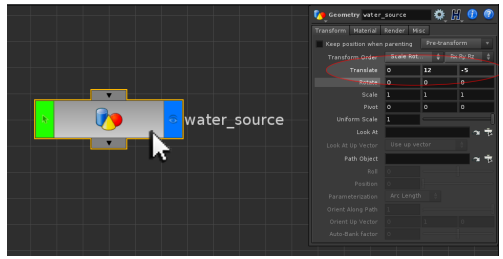
This will make the water wheel light enough to respond correctly to any objects that hit it and cause it to turn. As a final step, set the **Solver Engine** of the **Rigid Body Solver** to **RBD**.

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**

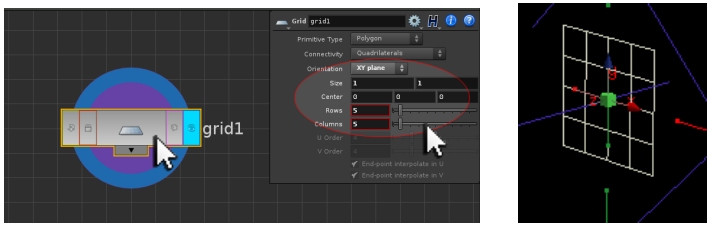


This will position the water_source object above and behind the water wheel. This can be used as a source for generating some **FLIP Particle Fluid**.

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**	

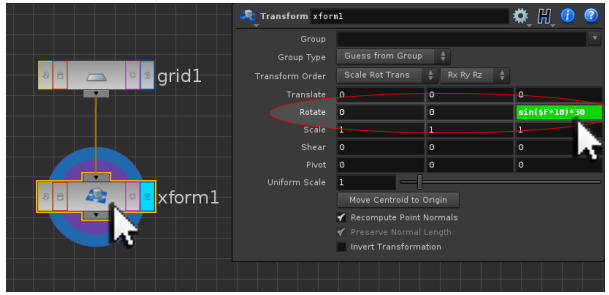


This will create a simple surface from which the FLIP Particle Fluid can emit from.

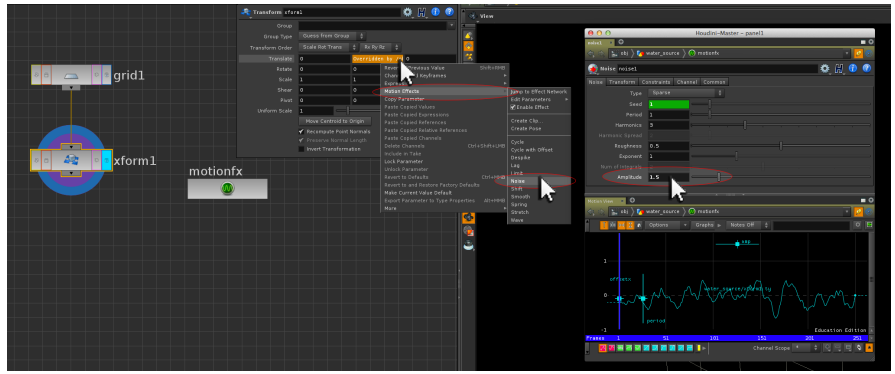
IMPORTANT NOTE: As this will be a continuous stream of FLIP Particle Fluid, it is better to use flat geometry as its source rather than geometry with volume (such as a Sphere SOP). Doing so helps prevent the generation of excess fluid.

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:

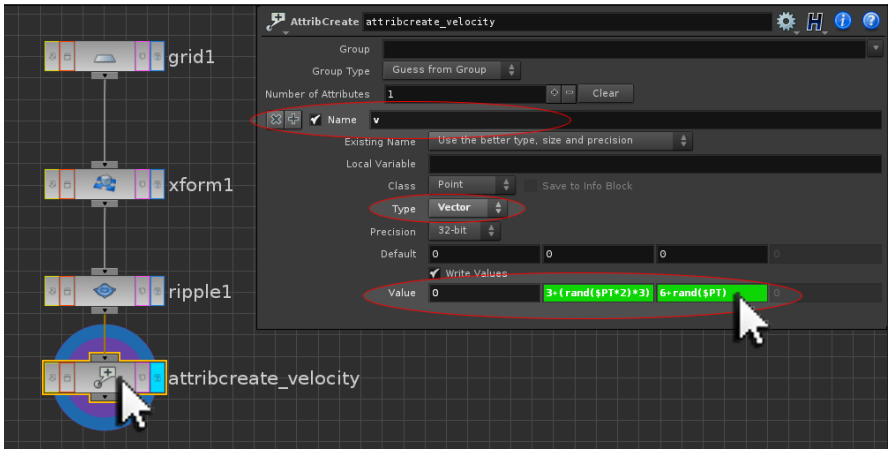
Amplitude	**1.5**

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**

DEFINING FLIP PARTICLE FLUID VELOCITY

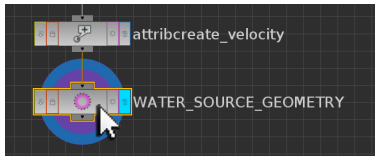
Before activating the water_source object as an emitter for FLIP Particle Fluids, it is also a good idea to assign a **velocity attribute** to the grid in order to control the direction and flow of the resulting fluid. Without such information, the fluid would simply pour onto the ground plane under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

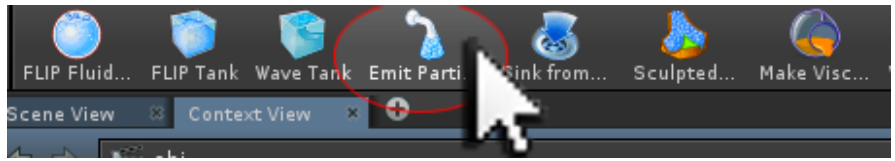
This will add per-point randomised velocity information in both the Y and Z axis. As a final step, append a **Null SOP** renaming it to **WATER_SOURCE_GEOMETRY**.



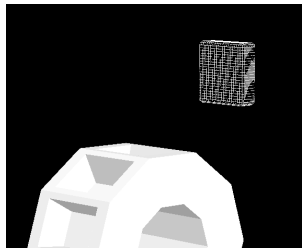
See file fruit_cannon_stage2.hipnc

ACTIVATING THE FLIP PARTICLE FLUID

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

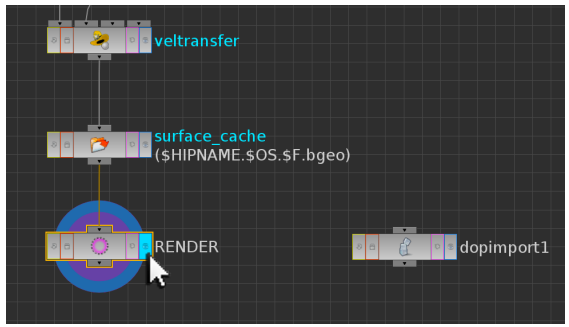


With the **water_source object selected**, **LMB** the **Emit Particle Fluid** button from the **Particle Fluids shelf**. When prompted to select the Fluid or Fluid Box, simply press **ENTER** to complete and configure the Fluid setup.

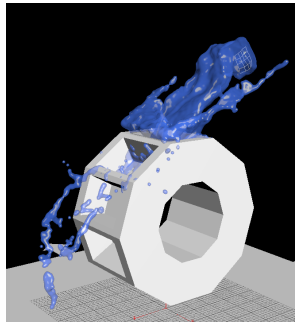


Initially the **water_source object** will appear as a volume mesh, and no fluid will appear in the Viewer when PLAY is pressed.

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.



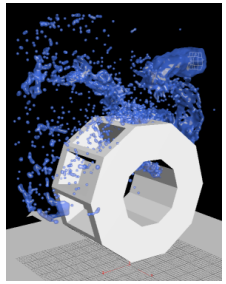
When **PLAY** is pressed, the fluid can be seen pouring over the water_wheel object; however currently it does not cause it to turn as anticipated. This is due to a default setting inside the FLIP Solver not to influence other dynamic objects.

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.

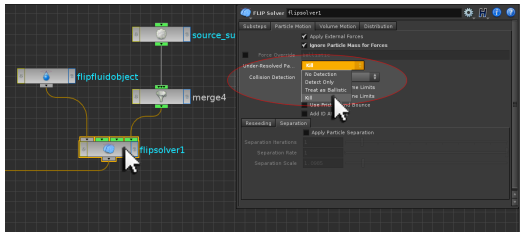
Now when **PLAY** is pressed, the water_wheel turns as expected under the force of the fluid impacting with it; however the fluid wildy explodes upon contact with the water_wheel.



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**.



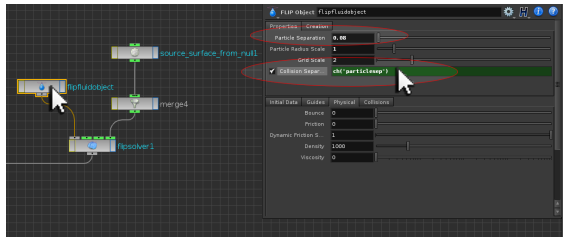
This will kill off any particles that the Flip Solver cannot resolve in terms of the simulation.

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:

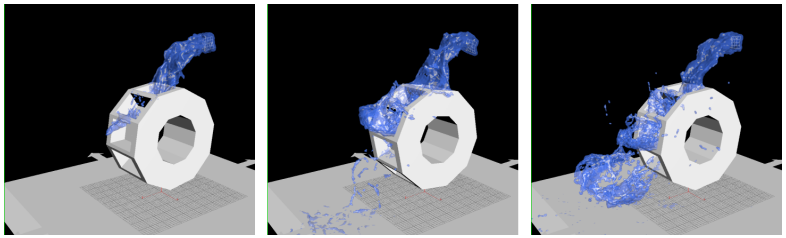
Particle Separation	**0.08**
Particle Radius Scale	**1**

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.

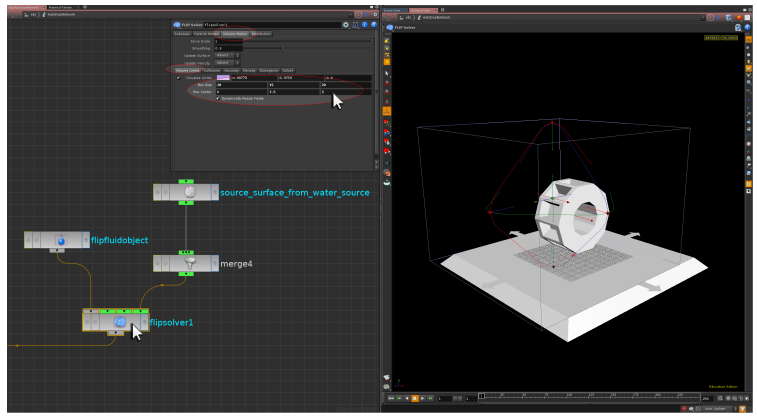
SETTING THE FLUID BOUNDING AREA

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:

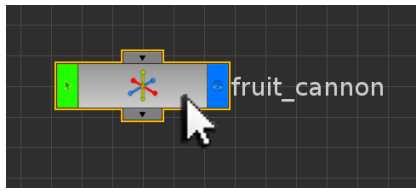
Box Size	**20**	**15**	**20**
Box Center	**0**	**7.5**	**3**

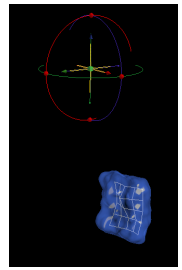


This will reduce the size of the bounding container of the fluids. **See file fruit_cannon_stage4.hipnc**

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**.

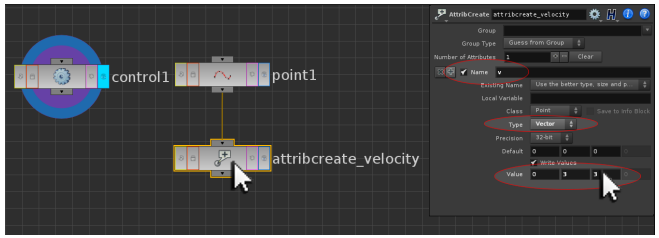




In the **parameters** for the Null object specify:

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**.



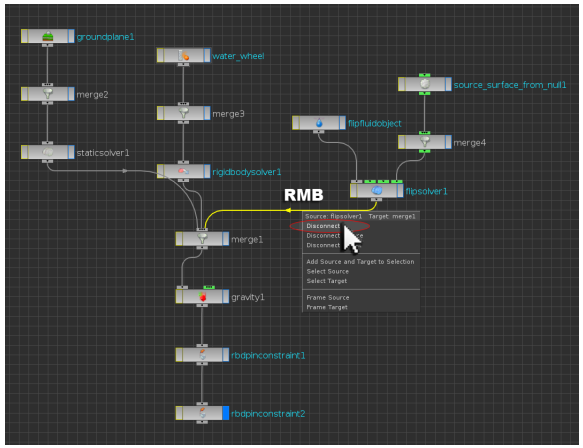
In its **parameters** specify:

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

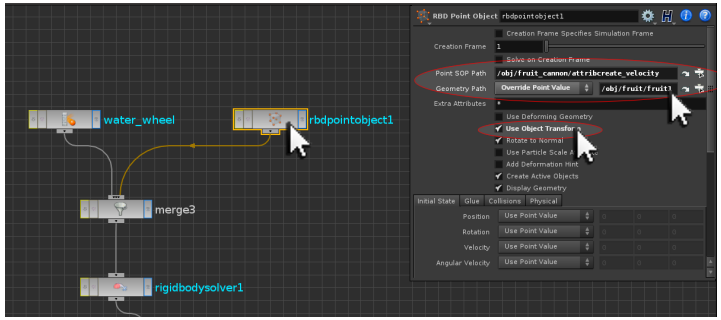
This will prime the fruit_cannon to fire out fruit over the fluid.

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.

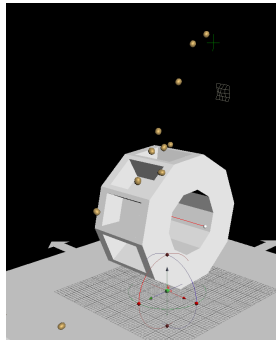
THE CREATION FRAME

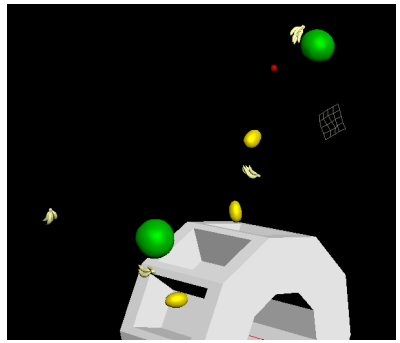
Dynamic Objects will be generated on the **Creation Frame** specified in their creation DOP. In the case of the RBD Point Object DOP, the default Creation Frame of 1 results in the DOP generating a single dynamic kiwi fruit at frame 1. Adding an expression into the Creation Frame parameter can modify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

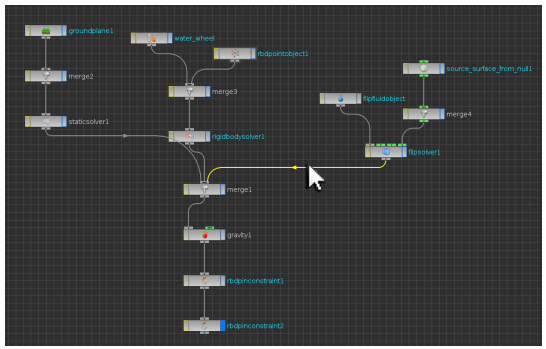


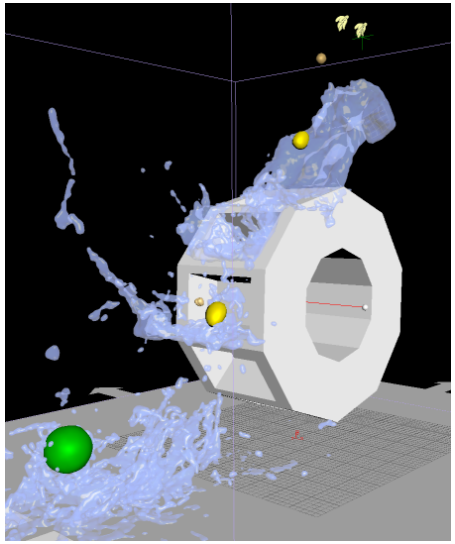


In the **parameters** for the **RBD Point Object DOP specify**:

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.



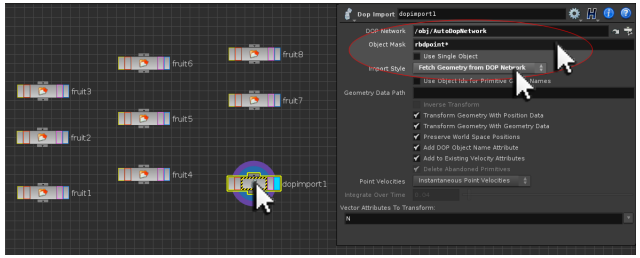


See file **fruit_cannon_stage5.hipnc**

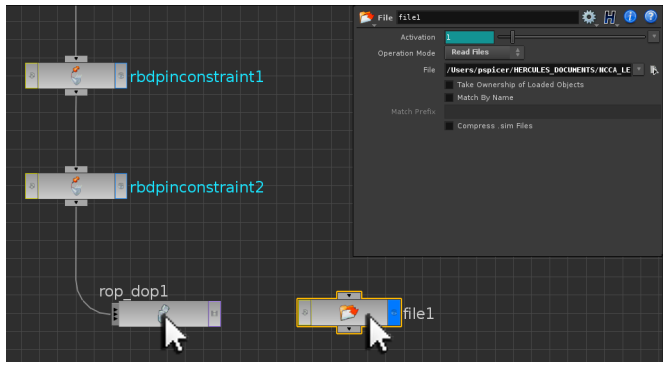
FURTHER REFINEMENTS

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

USEFUL DOP UTILITY – THE FETCH OBJECT – See file dop_trade_for_highres_geo.hipnc