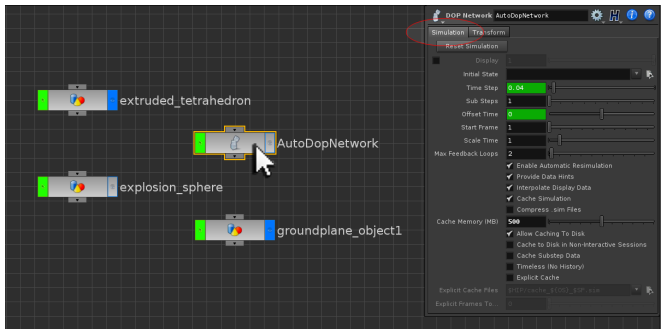
A GENERAL OVERVIEW OF DYNAMICS

A **DOP Network** is a container for dynamic operators (**DOPs**), and is either created automatically when using the Shelves (the AutoDopNetwork) or can be created by hand at any level of Houdini (the DOP Network). Each **DOP Network** node also has its own **upper level container parameters** that can also be modified.



SIMULATION FRAMES AND SIMULATION TIME

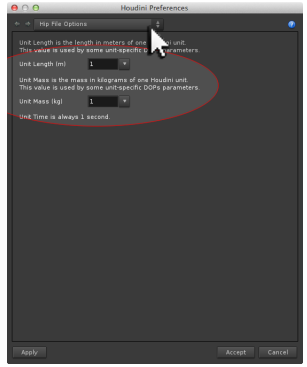
Each **DOP network is its own internal environment** contained within Houdini. **Playback time** can therefore be measured relative to the DOP Network. As a result, Global variables **$F** and **$T** are replaced by **$SF** (**simulation frame**) or **$ST** (**simulation time**) instead.

DOP SOLVERS

Currently there are **DOP Solvers** for calculating dynamic phenomena such as **cloth**, **particles**, **fire**, **smoke**, **hair**, and **fluid**. **All DOP Solvers** are capable of interacting with each other by means of the **Merge DOP** allowing for highly complex dynamic simulations to be produced. **Merged DOP Level node chains are calculated left to right**, and if incorrectly ordered or configured may generate spurious results. If **Dynamic Networks** are created using the **Shelves** rather than by hand, much of the initial setup becomes automated resulting in cleaner networks being generated.

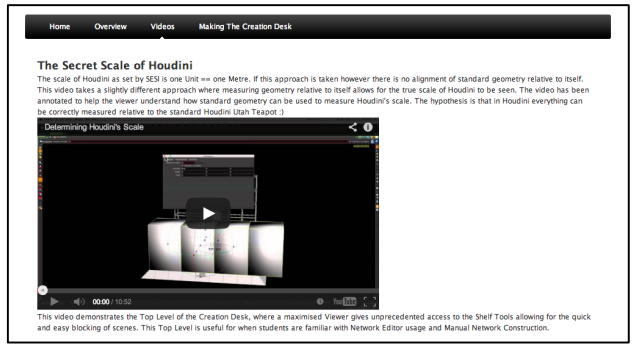
HOUDINI UNITS AND DYNAMICS

For accurate simulation work, the **Unit Length** and **Unit Mass** values of the scene need to be set. These can be found under the main **Edit menu > Preferences > Hip File Options**.



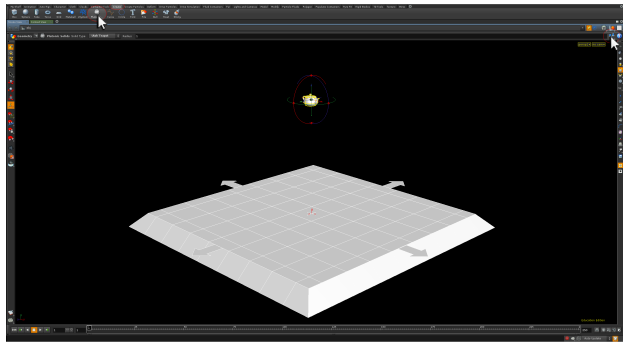
While scale in 3D animations systems is an abstract concept, in Houdini there is a secret scale that can be worked with. **See video 'The Secret Scale of Houdini'**.

http://houdinicreationdesk.ipage.com/videos.html

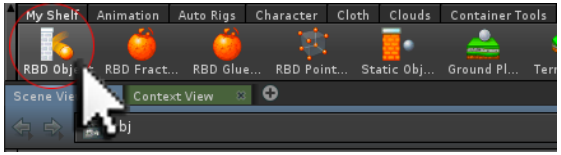


RIGID BODIES

In a **new Houdini scene**, **maximize** the **Viewer** and **unstow** the Shelves. Make sure that the **Viewer** is set to a **Scene View** and that the **Tool Options button** is set to **Create at Object Level.** From the **Rigid Bodies Shelf**, activate the **Ground Plane button**.



From the **Create Shelf** activate the Platonic Solids button, and use the Viewer parameters to set it to a **Utah Teapot** with a **Radius of 1**. Translate the teapot up so it is **10 units** above the ground plane.



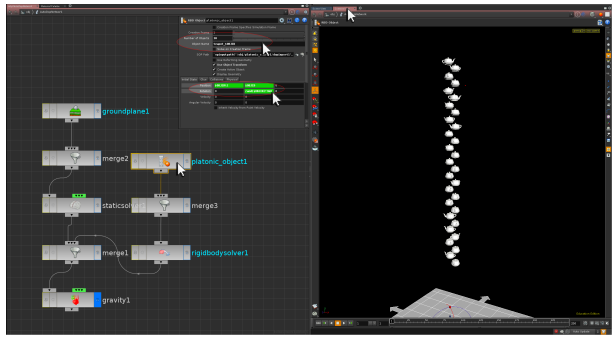
With the **teapot** still **selected**, **activate** the **RBD Object** button from the **Rigid Bodies Shelf**. When **PLAY** is pressed, the teapot falls onto the ground plane.

NOTE: When playing DOP simulations for the first time, ensure **Real Time Playback** is **deactivated**.

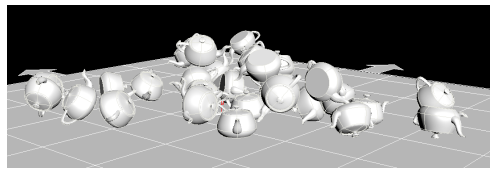
GENERATING MULTIPLE TEAPOTS

Head inside the **AutoDopNetwork** that has been automatically created at Object Level. In the **parameters** for the **RBD Object DOP** specify:

Number of Objects	**30**		
Object Name	**teapot_$OBJID**		
Initial State >			
Position	**$OBJID%2**	**$OBJID**	**0**
Rotation	**0**	**rand($OBJID)*360**	**0**

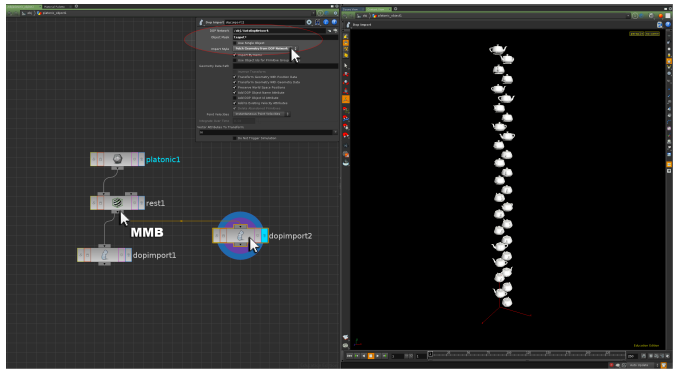


This will create a stack of 30 teapots in the **Context Viewer**. The **$OBJID variable** is a DOP Level variable returning an individual number unique to each dynamics object created. When **PLAY** is pressed, the stack of teapots fall onto the ground plane.



NOTE: This stack of teapots is not yet present in the Object Level Scene View, as this type of custom modification to the RBD dynamic teapot is not anticipated by the Shelf Tools in the nodes being automatically generated.

Head to the **Geometry Level** of the **platonic_object**. Here **MMB append** a **DOP Import SOP** as a new network branch from the automatically generated Rest Position SOP.



In the **parameters** for the **DOP Import SOP** specify:

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

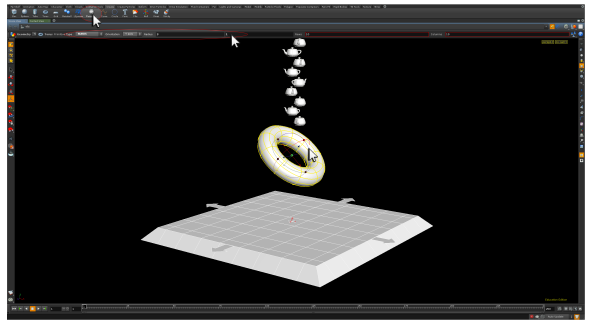
This will import the teapot stack, rather than the single original teapot into the Geometry Level for rendering purposes.

STATIC OBJECTS

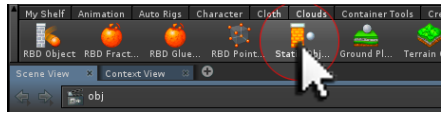
Static objects are Rigid Body objects that do not respond to dynamic forces such as gravity. As a result they can be used as collision events for dynamically moving geometry.

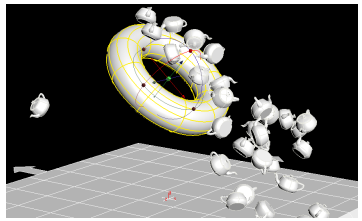
Maximise the **Scene View** and from the **Create Shelf CTRL + LMB** click on the **Torus button**. This will create a torus at the scene origin. In the Scene View **parameters** for the **Torus SOP** specify:

Primitive Type	**NURBS**	
Radius	**3**	**1**



Translate up and **rotate** the **torus** so it sits **between** the **ground plane** and the **teapots**. With the **torus** still **selected**, activate the **Static Object** button from the **Rigid Bodies Shelf**.





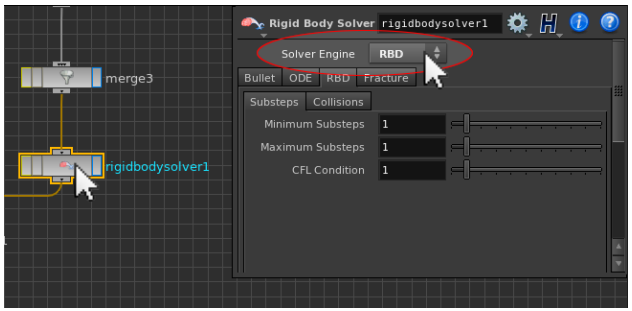
When **PLAY** is pressed, the teapots slide over the torus as if it were a solid object with no centre hole, and land on the ground plane. The reason for this error is the default Collision Geometry currently activated on the torus object.

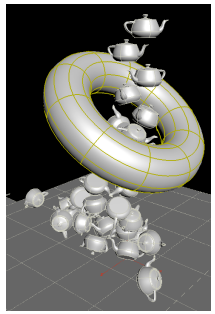
BULLET VS RBD SOLVER ENGINES

The **default Solver Engine** for **Rigid Body Solver** in Houdini is the **Bullet Engine**. This engine can calculate a large number of Rigid Body objects with very low processing overheads. **Setting** the **torus Static Object DOP > Collisions > Bullet Data > Geometry Representation parameter** to **Concave** will allow for objects with holes to be properly processed by the Bullet Engine Solver. The Bullet Engine can sometimes have difficulties processing complex simulations and concave geometry properly and also does not support scaling at Object Level.

An alternative to the Bullet Engine is the **RBD Solver Engine**. This was the default RBD engine in Houdini before the arrival of the (Open Source) Bullet Engine. While this engine allows for more complex RBD simulations and concave object shapes to be accurately processed, it is slower to calculate than the Bullet Engine.

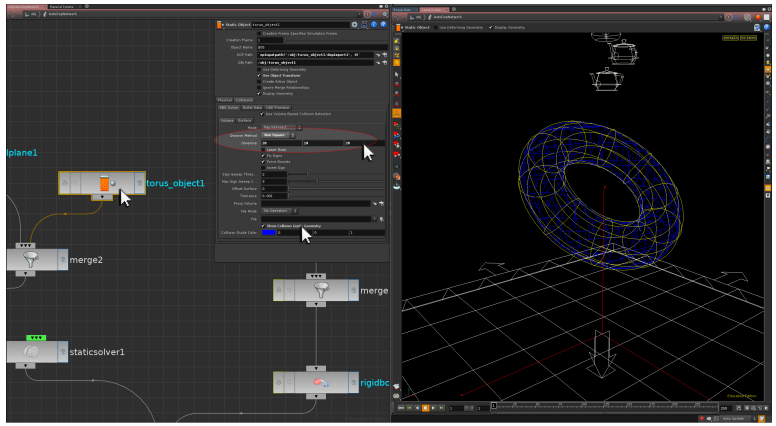
In the **parameters** for the **Rigid Body Solver DOP**, **switch** the **Solver Engine** from **Bullet** to **RBD**.





When **PLAY** is pressed, the teapots fall through the torus as expected. This is due to the default Collision Geometry for the RBD Engine anticipating complex concave geometry.

Collision geometry, no matter what Solver Engine is chosen, **should always be examined and adjusted** for **optimal simulation processing**.



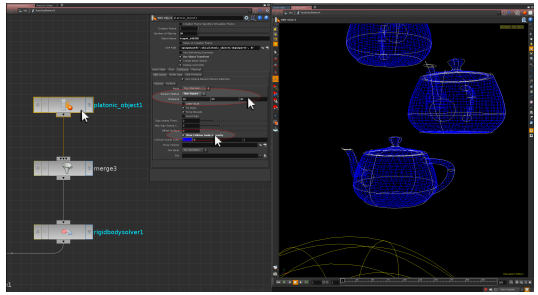
In the case of the **static torus object**, the following values can be specified for the **RBD Collision Geometry**:

Division Method	**Non Square**		
Divisions	**20**	**10**	**20**

This will still retain the overall shape of the torus; however there will now be less internal topology to the collision shape.

In the case of the **teapot's RBD collision geometry**, the following values will improve the collision shape to also better encompass the handle and the spout:

Division Method	**Non Square**		
Divisions	**30**	**30**	**30**



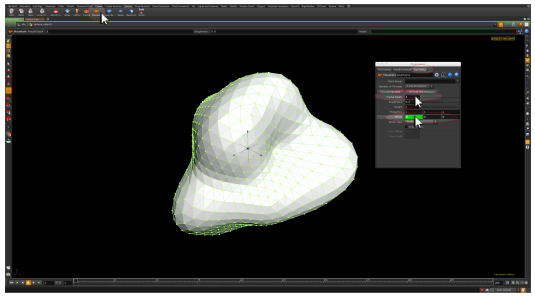
See file H15_RBD_teapots_stage1.hipnc

ANIMATED RIGID BODIES

DOPs can also import **animated geometry** as dynamic objects. This includes both **transformations** and **surface deformation** animation. Back at **Object Level**, create a **Sphere Object** from the **Create Shelf**, and specify in the Viewer **parameters**:

Primitive Type	**Polygon**
Frequency	**10**

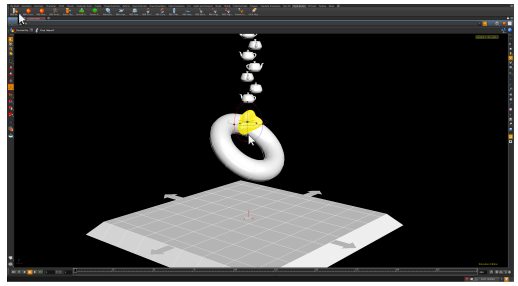
With the **sphere** still **selected** in the Viewer, activate the **Mountain button** from the **Deform Shelf**. This will add a Mountain SOP to the sphere object.



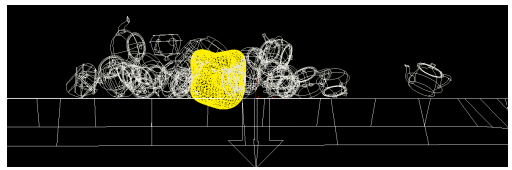
Press **ESC** and **ENTER** with the **mouse over the Viewer** to ensure the **Mountain SOP** is activated as the **tool mode**, and press **p** to bring up a **floating parameters window**. Under the **Op Dialog** section of the **parameters** specify:

Fractal Depth	**1**		
Offset	**$T**	**0**	**0**

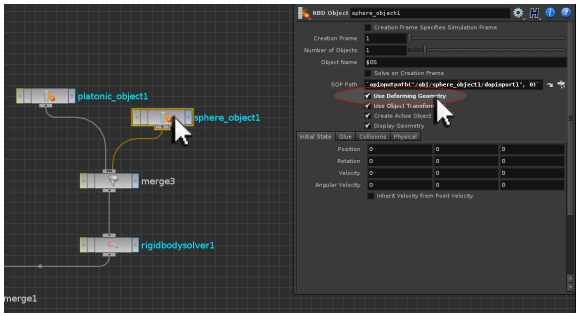
This will create an animated deforming surface for the sphere. At **Object Level** transform the sphere up so that it is between the **torus** and the **teapots**.



With the **sphere selected**, activate the **RBD Object button** from the **Rigid Bodies Shelf**. This will import the sphere into the Dynamics simulation network. When **PLAY** is pressed, the animated sphere appears to behave correctly however upon closer examination, the sphere penetrates the ground plane. DOPs must be explicitly told about this deforming surface, before a correct simulation playback can occur.

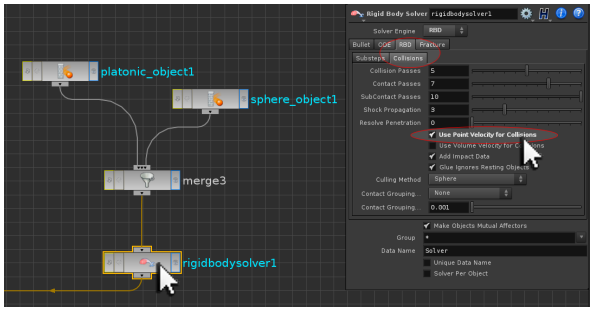


ACTIVATING DYNAMIC SURFACES

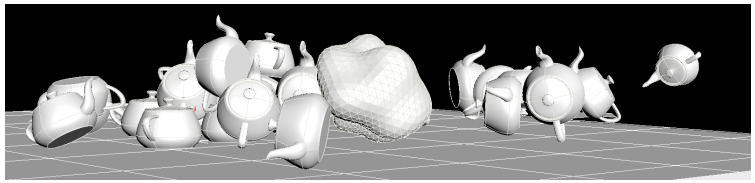


At **DOP Level**, locate the **RBD Object DOP** importing in the sphere. In its **parameters**, activate the **Use Deforming Geometry** option. This will tell DOPs to anticipate surface and transformation animation on this dynamic object, rather than using the initial static shape of the sphere for dynamics.

Locate the **Rigid Body Solver DOP**, and in the **RBD > Collision parameters** activate the **Use Point Velocity for Collisions** option. The Rigid Body Solver is responsible for calculating rigid body interactions, and with this option set, will now check each point on the sphere when calculating collisions.

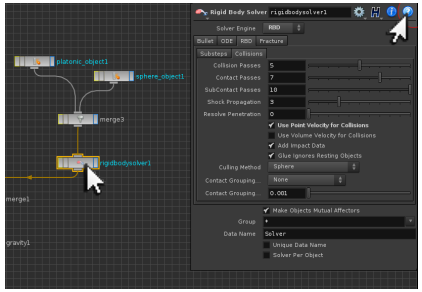


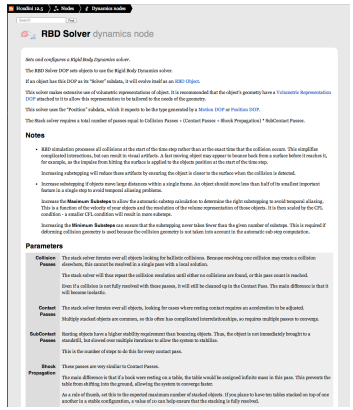
When **PLAY** is pressed, the teapots and the spheres all collide correctly. The animated surface of the sphere also causes it to roll around the ground plane under its own momentum.



When a simulation has been configured it is important to check its visual success. An example error could be where two teapot handles become intersected for no reason. Any **simulation errors** can be **corrected** by examining the **associated Solver Engine** of the dynamics system.

As this is a **Rigid Bodies simulation** using the **RBD Engine**, examination of these parameters is required. The **Help Card** for this solver can be accessed by activating the **Help Card** for the **Rigid Body Solver DOP**, and then locating the **help card text link** to the **RBD Solver Help Page**.

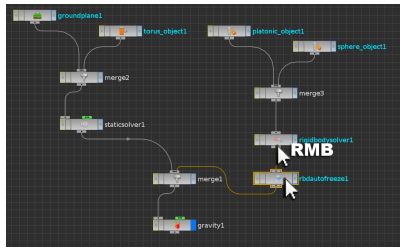




This **Help Card** will explicity list the **RBD Solver parameters** that can correct simulation errors. When reviewing the Help Cards for dynamics solvers, it is beneficial to relate as closely as possible any visual error with the described error types listed in the Help Card. This can prevent the inadvertent adjustment of non-relevant parameters, which in turn can slow simulation time. In this example the teapots are landing and stacking on top of each other. When the parameters section of RBD Solver Help Card is examined, the **Shock Propagation parameter** description relates directly to the stacking of geometry. **Increasing** the **Shock Propagation parameter** value to **5** can therefore improve the visual success of any stacking based simulations. Other key RBD Solver parameters for improving simulation success include **Minimum** and **Maximum Substeps** (ie how many in-between frames a simulation is calculated). Setting these to **Minimum Substeps of 5**, and **Maximum Substeps of 10** will improve the accuracy of the simulation, but at a cost to the overall simulation time.

AUTO FREEZING RESTING DYNAMIC OBJECTS

In the DOPs Network, **RMB** append a **RBD AutoFreeze DOP** to the **Rigid Body Solver DOP**.

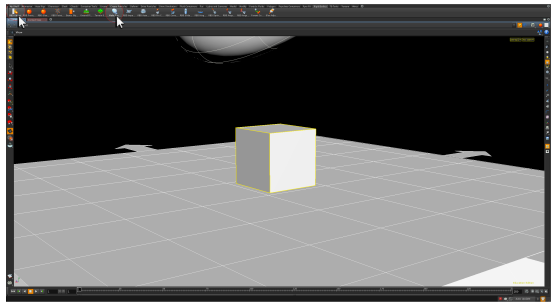


As the teapots come to rest after impacting with the ground plane, this node will tell DOPs to no longer to calculate them as part of the simulation. This in turn will allow the simulation processing to focus solely on the animated sphere calculation.

See file H15_RBD_teapots_stage2.hipnc

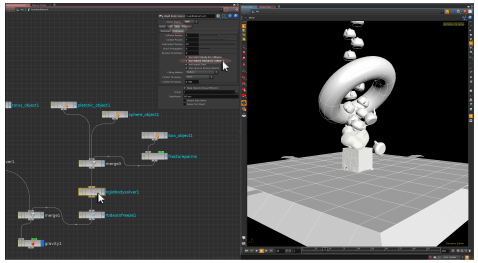
AUTOMATED BREAKING OBJECTS

At **Object Level**, create a **Box Object**, and set the **Size** of the Box SOP as **2**, **2**, **2**. Position the Box Object so that it sits on the ground directly underneath the Torus Object. With the Box Object selected, activate the **Rigid Bodies Shelf**, and **LMB** choose **RBD Object button** followed by **Make Breakable button**. Press **ENTER** to confirm the **rbd box** as the **breakable object**.



When **PLAY** is pressed, the box begins to break as soon as the deforming sphere impacts on it. Examination of the scene reveals however that the impact debris created is penetrating and going through the ground plane.

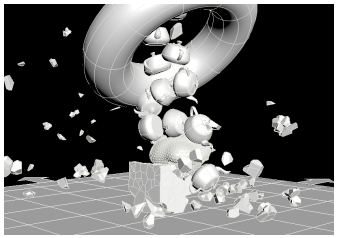
This again can be corrected by modifying the **parameters** of the **RBD Engine Rigid Body Solver**. Examination of the **Help Card** reveals that activating **Use Volume Velocity for Collisions** parameter will tell DOPs to **anticipate changes** in the **volume representation** and **topology** of the **RBD Objects being processed**. In the case of the **breakable box**, all of its **shatter chunks** have **differing volumes** to the original box shape, and are being continually recalculated over the course of the simulation as other shattering takes place. With the **Use Volume Velocity for Collisions parameter activated**, the box and its associated debris remain now above the ground plane.



A more **impressive impact explosion** can be created for the box by modifying the **Physical Parameters** of the **RBD Object DOP** reading it in. In the **Physical Parameters** specify:

Density	**100**
Bounce	**1.5**

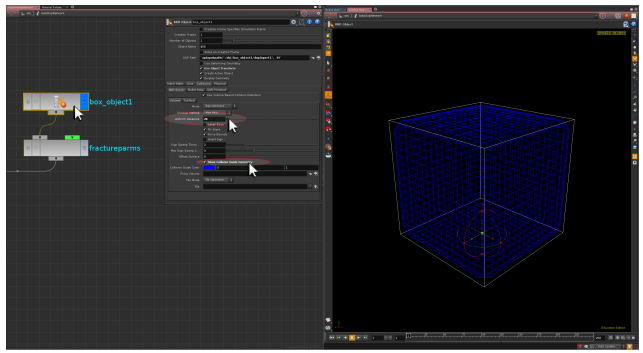
Now when **PLAY** is pressed, the impact explosion effect is much greater.



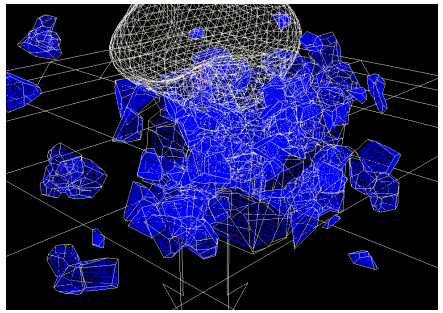
NOTE: Large decreases in density for an RBD Object can create geometry penetration problems. **Working in DOPs** is therefore a **cyclic** and **iterative process** of compensating for RBD values by adjusting the Substeps and Collisions parameters of the Rigid Body Solver DOP to help create the desired aesthetic. Understanding the Substeps and Collisions parameters of the Rigid Body Solver DOP (or indeed any other Solver DOP) is therefore key to efficient DOPs working.

COLLISION GEOMETRY AND BREAKABLE OBJECTS

Another consideration when working with breakable objects is the initial collision geometry of the object being broken. By default, its collision geometry settings will automatically be inherited by each shatter chunk being created by the **fractureparms** DOP node.



Reducing the collision geometry's **Uniform Divisions parameter** to **20** will reduce the topology of the breakable box, whilst still retaining its overall shape. Each resulting shatter chunk will also inherit this value.



See file H15_RBD_teapots_stage3.hipnc

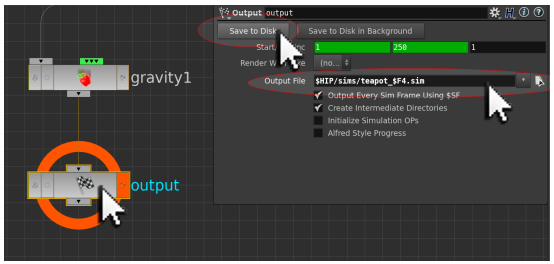
BAKING OUT SIMULATIONS

It is also good practice to **bake out simulations to disk**. Baking out simulations **can be done either** directly **inside the DOP Network** (using the **Output DOP**), or at **Outputs Level** of Houdini (using a **Dynamics ROP**). **Both methods** will generate a series of **.sim** files storing the dynamics simulation. A Dynamics ROP can also be called by a Render Farm to render simulations in a similar way to calling a Mantra ROP on the Render Farm.

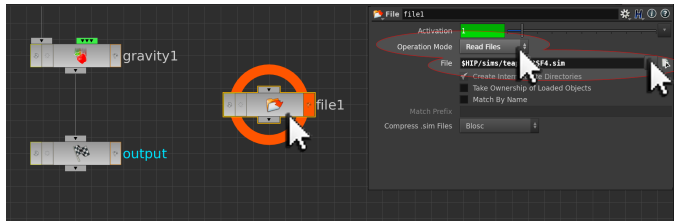
Both methods can also be called using the **hrender Shell Command** allowing for simulation baking to occur without Houdini being open and running.

A **.sim** file will store on a **per frame basis** the **simulation data** of a given dynamics network. **Generating .sim files** can either be done **after the configuration of a dynamics network** is complete, **or for more complex simulations, as the network is being developed**.

Baked simulations can also then drive further simulations, allowing for a layering up of simulations on top of each other, in order to create a visually dense and complex end result. To the **Gravity Force DOP**, **RMB** append a **ROP Output Driver**. In the **parameters** for the **ROP Output Driver**, specify a **Frame Range** and an **Output File Name** and **Location** for the resulting **.sim** files (for example **/transfer**).



When the .sim files for this scene have been rendered, they can be read back into DOPs using a **File DOP**. The **Operation Mode parameter** of **File DOP** must be set to **Read Files** for the simulation to be re-imported from disk.



NOTE: A **File DOP** can also **write out .sim files** in a similar way to the **ROP Output Driver DOP**. It is however better to **only use a File DOP for reading in** rather than writing out. This will help prevent accidental overwriting out of .sim files.

When the resulting simulation is read back into DOPs it can be fully checked for any errors, and if necessary the RBD Solver parameters can be adjusted and the simulation re-saved out to disk.

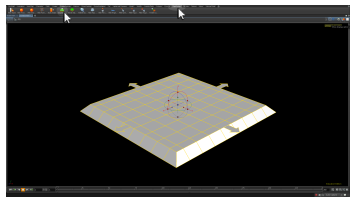
NOTE: Before re-baking the simulation, the **Display Flag** must be reset back to the **Output DOP** before it is activated.

When Dynamics simulations have been finalised, the **geometry** they affect can also be further baked out as a **series of .bgeo files**. This can be done using a **ROP Output Driver created at Geometry Level**. Doing this can help maximise computational and rendering efficiency, as the .bgeo files can have excess attributes removed before rendering and also do not have to store the same intensity of internal data that a series of .sim files has to store.

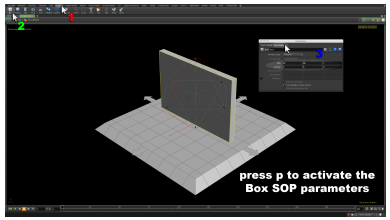
See file H15_RBD_teapots_end.hipnc

MAKE BREAKABLE & METABALLS

In a new Houdini scene, activate the Scene View and make sure the Viewer **Tool Options** button is set to **Create at Object Level**. From the **Rigid Bodies Shelf** activate the **Ground Plane** button. This will configure a Dynamics Network on which an exploding wall effect can be built.



From the **Create Shelf**, activate the **Box button**. With the **mouse over the Viewer**, press **ENTER** to confirm its creation at the scene origin. Press i to go inside the box object to its Geometry Level.

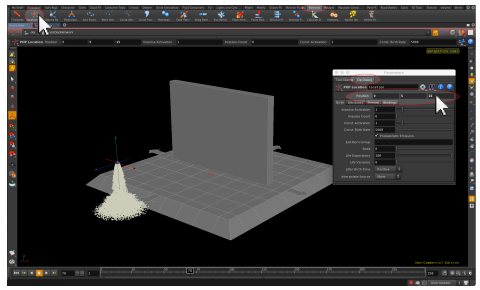


With the **mouse** over the **Viewer**, press **p** to activate the **parameters** for the **Box SOP**. Go to the **Op Dialog** section and specify:

Size	**15**	**10**	**1**
Centre	**0**	**5**	**0**

This will create a simple wall that can be used to test the breaking effect.

Press **u** with the **mouse over** the **Viewer** to go back up to **Object Level**. From the Create **Particles Shelf**, activate the **Location Particle Emitter** Button. Press **ENTER** to confirm the position of the emitter at the scene origin. Houdini will create a particles network inside the AutoDopNetwork.



Position the **location** of the **emitter** a short distance in front of the wall. **Specific coordinates** for this location can be set in the **Op Dialog** Parameters.

Coordinates	**0**	**5**	**15**

When **PLAY** is pressed, a burst of particles fall onto the ground plane.

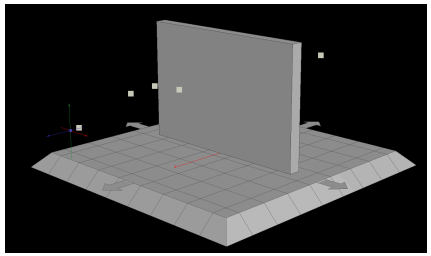
In the **Birth** section of the OP Dialog parameters specify:

Const. Activation	**$SF<=25**
Const. Birth Rate	**5**

In the **Attributes** section of the OP Dialog parameters specify:

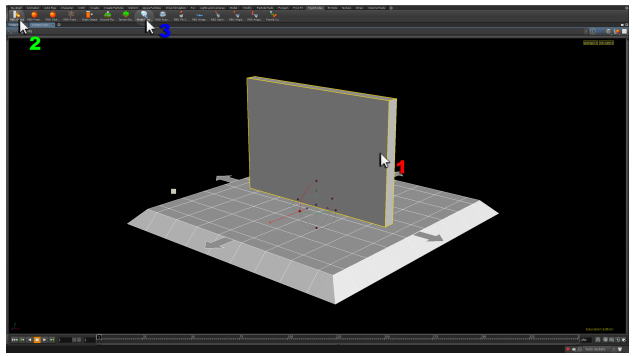
Velocity	**0**	**0**	**-20**
Variance	**10**	**10**	**10**

When **PLAY** is pressed, a short burst of particles will fire towards the wall, bouncing on the ground plane as they pass through it.

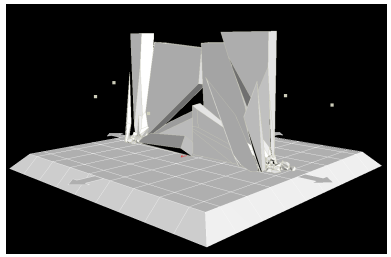


The display size of the particles can be increased by activating the Viewer's Display Options (**d** with the **mouse over** the **Viewer**).

Return to Object Level, select the **wall**, and from the **Rigid Bodies Shelf** press the **RBD Object Button** followed by the **Make Breakable** button. Press **ENTER** to confirm the dynamic wall as the DOP Object to make breakable.

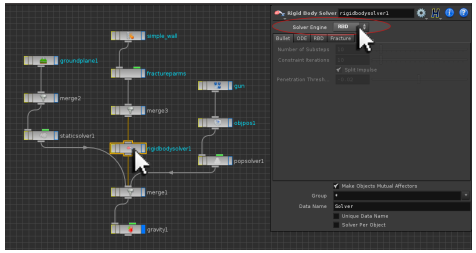


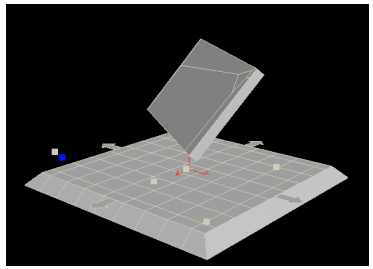
Initially the wall object will fracture into many pieces, and the particles will simply pass through the wall geometry. This is due to the default **Bullet Engine** being active on the **Rigid Body Solver DOP**.



Head inside the AutoDopNetwork, and locate the **Rigid Body Solver DOP**. In the **parameters** for the **Rigid Body Solver DOP** specify:

Solver Engine **RBD**

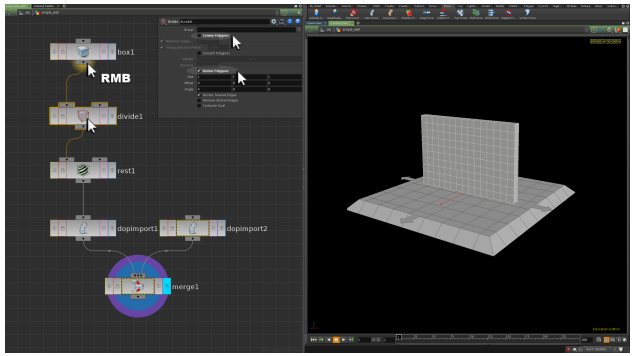




When **PLAY** is pressed again, the particles will fire and bounce off the wall, causing it to fly off the ground plane. The wall also still cracks as a result of its interaction with the ground plane. See **metaball_make_breakable_stage1.hipnc**

ACTIVATING THE PARTICLES TO BREAK THE WALL

Go inside the **box_object**, and **RMB** insert a **Divide SOP** after the **Box SOP**.



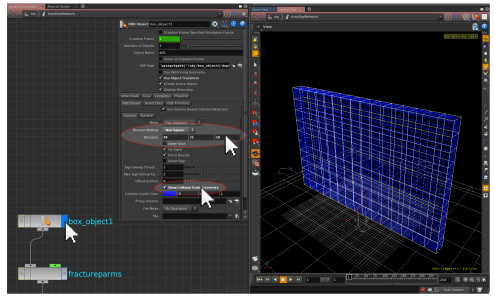
In the **parameters** of the **Divide SOP**, **deactivate Convex Polygons**, and **activate Bricker Polygons**. This will add additional geometry detail to the wall that in turn will aid its breaking aesthetic.

Inside the **AutoDopNetwork**, activate the **Display** Flag for the **box_object RBD Object DOP**. In the **parameters** activate the **Show Collision Guide Geometry** option, and specify:

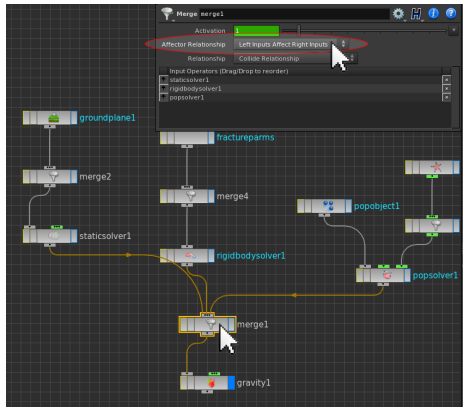
Division Method	**Non Square**		
Divisions	**50**	**30**	**10**

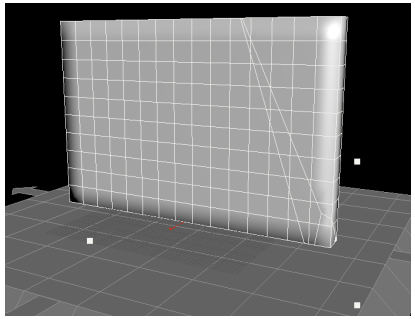
This will create suitable topology for the collision volume.

NOTE: Each breaking chunk will also inherit this Divisions setting.



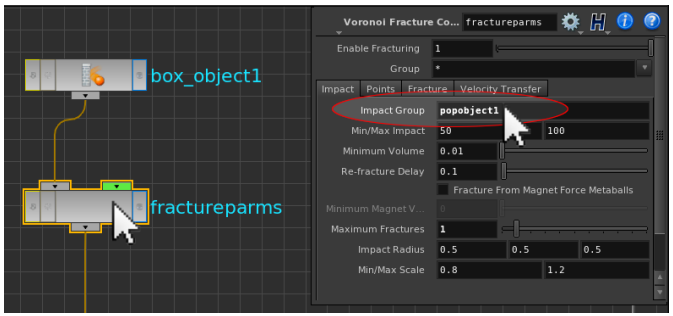
Deactivate the **Collision Geometry display**, and select the **Merge DOP** responsible for merging all of the dynamic solvers together. In its **parameters** set the **Affector Relationship** to **Left Inputs Affects Right Inputs**.





This will make the wall dominant to the particles hitting it, preventing it from flying away. When **PLAY** is pressed, the wall shatters as before from influence of the ground plane; however now the particles simply bounce off it.

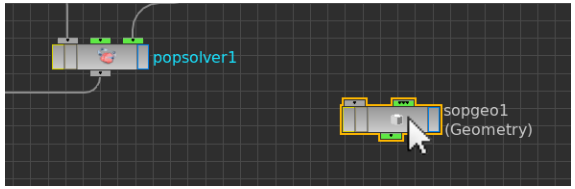
Select the **fractureparms** node beneath the **box_object RBD Object DOP**, and in its parameters change the Impact Group parameter from * (all the dynamic objects) to **popobject1**.



This will ensure only the particles create the fracturing effect. When **PLAY** is pressed, the particles bounce off the wall but no fracturing takes place.

ACTIVATING DYNAMIC PARTICLES AS MAGNET FORCES

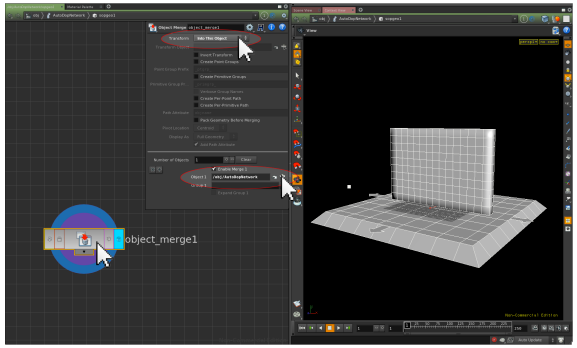
In order for the particles to fracture the wall, they must first be activated as metaball based magnet forces. This can be done directly inside the AutoDopNetwork.



In a new part of the **Network Editor**, create a **SOP Geometry DOP**. This node contains a SOP Level where dynamic geometry can be modified as the simulation runs.

Double LMB on the **SOP Geometry DOP** to go inside it. Here create an **Object Merge SOP** and in its **parameters** specify:

Object 1 **/obj/AutoDopNetwork**
Transform **Into this Object**

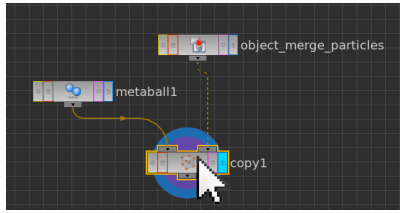


This will read the entire simulation into this SOP Level. If however the **Object 1 parameter** path is **modified** to:

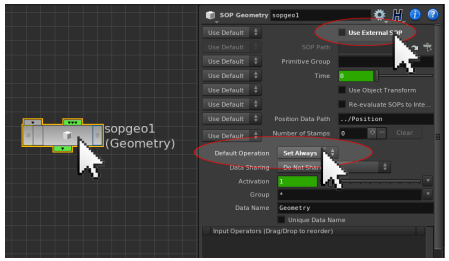
Object 1 **/obj/AutoDopNetwork:popobject1/Geometry**

Only the dynamic particles geometry will be read into this SOP Level instead.

NOTE: This **/obj/AutoDopNetwork:object_name/Geometry** syntax can be used to isolate any dynamic object individually.



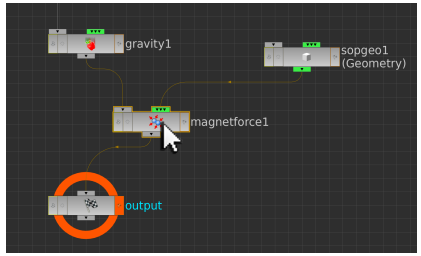
With the dynamic particles isolated in the Object Merge SOP, **copy a metaball** onto the particle system.

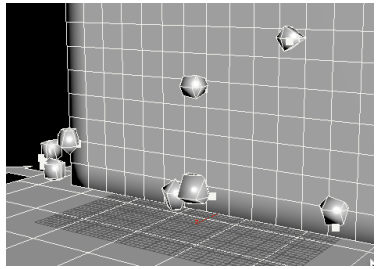


Press **u** to return back to the **AutoDopNetwork Level**, and in the **parameters** for the **SOP Geometry DOP deactivate** the **Use External SOP tick box** and set the **Default Operation** to **Set Always**.

This will force the SOP Geometry DOP to look at its internal SOP network rather than an external one, plus ensure the copying of the metaball onto the particles is evaluated every frame.

Append a **Magnet Force DOP** to the **Gravity DOP**, and wire the **SOP Geometry DOP** as its **second input**. When **PLAY** is pressed, the copied metaballs can be seen hitting the wall.

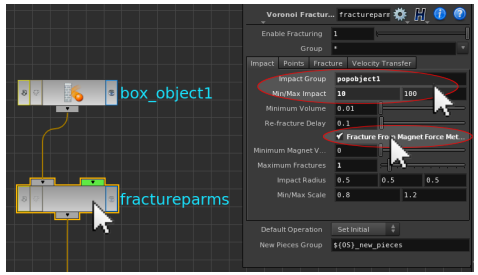


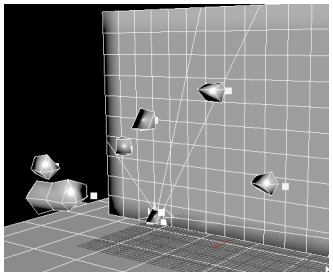


In the **fractureparms DOP parameters** specify:

Min/Max Impacts **10** **100**

☑ **Fracture From Magnet Force Metaballs**



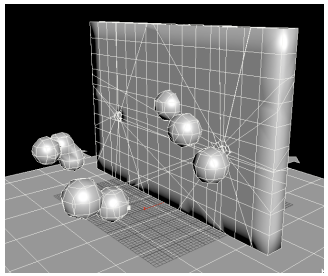


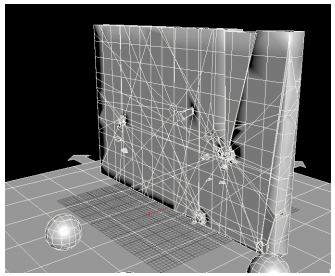
This will activate the metaballs for fracturing, and give a suitably low fracture impact value to achieve the effect. When **PLAY** is pressed, the metaballs impact with the wall as before; however only the first metaball hit causes the wall to crack.

In order to get the remainder of the metaballs to also crack the wall, increase the **Max Fractures** parameter of the **fractureparms DOP** to **8** (the number of metaball bullets).

NOTE: Ordinarily this Parameter will control how many times an object or is broken pieces will shatter upon further collision. In the context of a metaball impact setup, this appears to refer to the number of expected impacts an object is likely to receive.

As a final step, increase the **Scale Force parameter** of the **Magnet Force DOP** to **10**. This will amplify the force created by the metaballs fracturing the wall. When **PLAY** is pressed, the effect of the metaball shattering can be seen.





The simulation can now be checked for collision intersection errors. If any intersections occur, the Collision Geometry Divisions for the wall can be increased. The **Shock Propagation parameter** located in the **RBD Solver DOP** can also be increased to improve upon how the geometry stacks onto itself when the breaking chunks are formed. The **Use Volume Velocity for Collisions tick box** can also be activated so DOPs anticipates changes in the volume of the resulting fractured wall pieces.

See file **H13_metaball_make_breakable_stage2.hipnc**

COLOUR DRIVEN SHATTERING

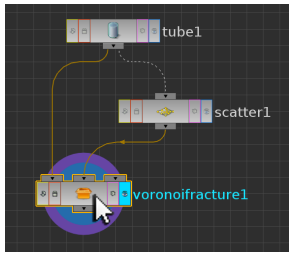
In a **new Houdini scene**, create a **Tube Object**, and inside at its Geometry Level specify in the **parameters** for the **Tube SOP**:

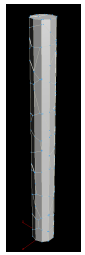
Primitive Type	**Polygon**		
☑	**End Caps**		
Center	**0**	**ch("height")/2**	**0**
Height	**20**		

Append a **Scatter SOP** to the Tube SOP, and specify in its **parameters**:

Force Total Count	**40**

As a new node, create a **Voronoi Fracture SOP**, and wire the **output** of the **Tube SOP** as the **first input**, and the **output** of the **Scatter SOP** as the **second input**. This will create fracture chunks across the tube.





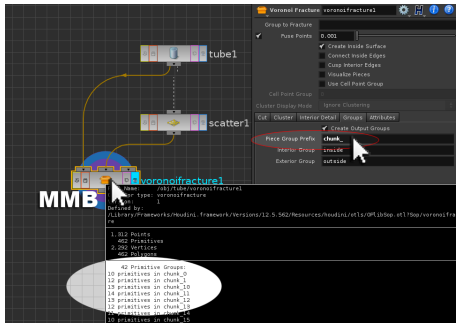
NOTE: By default, the Voronoi fracture pieces have a distinct CG aesthetic. Additional Points can be scattered over the tube before the Voronoi Fracture SOP to create more naturalistic fracture shapes. **See James Roberts Houdini Fracturing Master Class (/public/mapublic/HOUDINI_RESOURCES/MADE1213_Houdini_MasterClasses/JamesRoberts).**

In the **parameters** of the **Voronoi Fracture SOP** specify:

Groups >

Piece Group Prefix **chunk_**

This will assign a group called chunk_ to each individual shattered component of the Voronoi fractured tube. This can be verified by **MMB** on the **Voronoi Fracture SOP**.



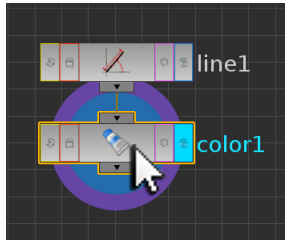
In a new part of the **Network Editor**, create a **Line SOP**. In its **parameters** specify:

Points **50**

Keyframe (**Alt + LMB**) the **Length parameter** so that on **Frame 1**, the Length parameter has a **value of 0**; and on **Frame 50** the Length parameter has a **value of 20**. If necessary **Scope** this **parameter** to ensure the animation curve is type **Bezier**. This will create an animated growth to the Line SOP.

To the output of the **Line SOP** append a **Color SOP**. In its **parameters** specify:

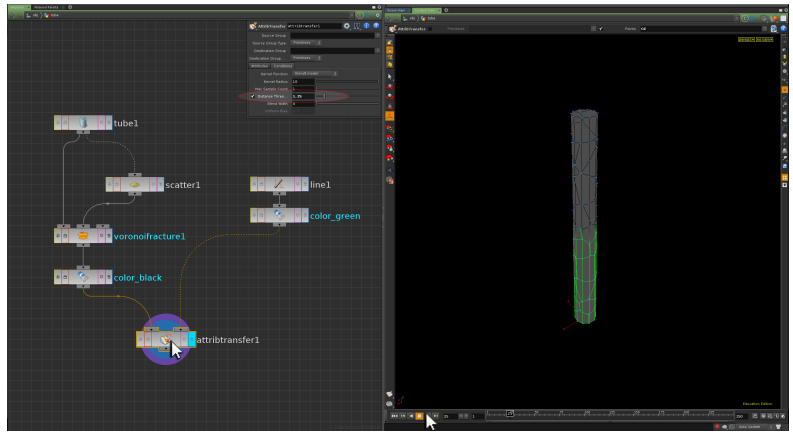
Color **0** **1** **0**



This will colour the animated line green, which in turn can be used to drive dynamic animation of the fracture chunks.

Append a **second Color SOP** to the **output** of the **Voronoi Fracture SOP**, this time setting its colour to **black**. Setting the **Viewer** to **Hidden Line Ghost** will allow the fractured tube geometry to still be seen.

Append to this **second Color SOP** an **Attribute Transfer SOP**, and wire the output of the **green Color SOP** as its **second input**.



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

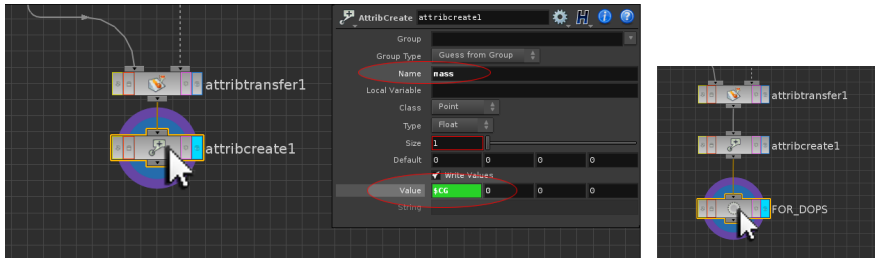
Attributes		
☐	**Primitives**	
☑	**Points**	**Cd**
Conditions		
☑	**Distance Threshold**	**1.25**

This will transfer the animated green line colour onto the fractured tube geometry.

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

Name	**mass**			
Value	**$CG**	**0**	**0**	**0**

This will store the animated colour as an attribute that can be called in DOPs to drive the breaking of the fractured tube.

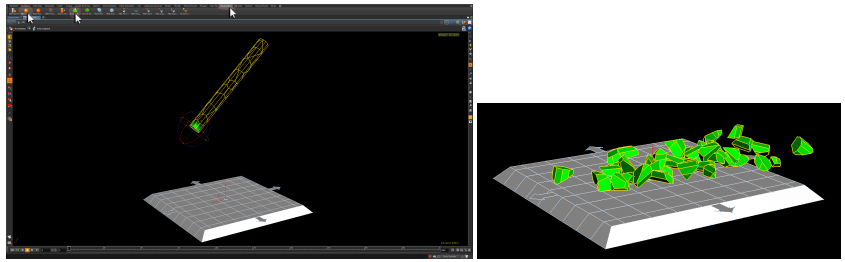


As a final step, append a **Null SOP** to the network chain, and **rename** it to **FOR_DOPS**.

See file H15_green_tube_dynamic_chunks_stage1.hipnc

At **Object Level**, **maximise** the **Viewer**, and activate the **Rigid Bodies Shelf**. From the **Rigid Bodies Shelf LMB** the **Ground Plane button**. This will initialise a Dynamics Network for the simulation.

Move the **Tube Object** so that it sits angled above the **Ground Plane**, and from the **Rigid Bodies Shelf LMB** the **RBD Fractured Object button**. When prompted by Houdini for a type of fractured object, select the **RBD Fractured Object button**. This will import the Tube Object into the Dynamics simulation in a format allowing for greater customisation than the default Packed Object option.

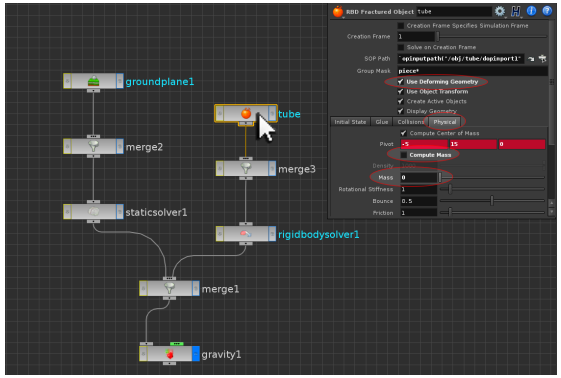


When **PLAY** is pressed, the tube lands on the Ground Plane and breaks apart on impact.

USING COLOUR TO DRIVE THE SIMULATION

Currently the tube falls uniformly to the ground plane and breaks apart under the influence of gravity. The animated colour of the tube can however be used to drive the breaking of the chunks and their response to forces.

At Object Level, go inside the **AutoDopNetwork** node, and locate the **RBD Fractured Object DOP** reading in the tube geometry.



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

☐ **Fracture By Name**

Group Mask **chunk***

☑ **Use Deforming Geometry**

Physical >

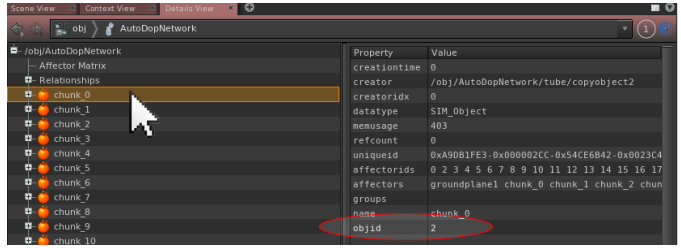
☐ **Compute Mass**

Mass **0**

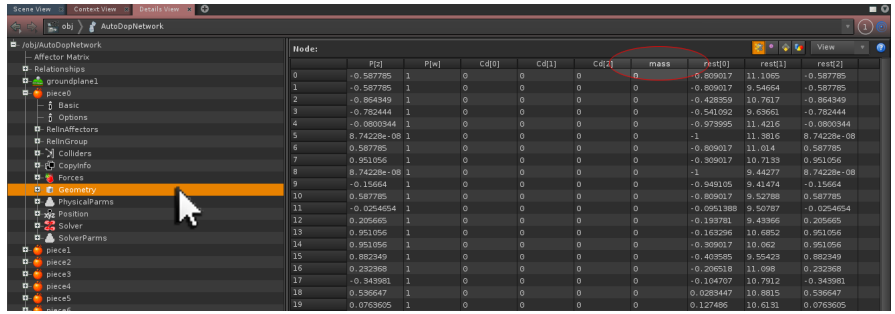
Activating the Use Deforming Geometry option will allow the animated colour information to become part of the DOPS calculation. Deactivating the Mass of the tube will stop the tube responding to DOP forces. Deactivating Fracture By Name will force the RBD Fractured Object DOP to read in each fracture chunk using the groups created by the Voronoi Fracture SOP. When **PLAY** is pressed, the tube now remains at its original location.

THE AUTODOPNETWORK DETAILS VIEW

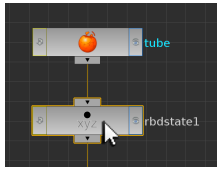
Over the **Viewer Pane**, activate a **Geometry Spreadsheet**, and set its **Pane Number** to **1** so that it returns information about the AutoDopNetwork.



Each chunk of the tube is represented by a **$OBJID** number. The $OBJID number for **chunk_0** is **2**, **chunk_1** is **3**, **chunk_2** is **4** etc. The $OBJID number for each chunk piece of the tube can be used to animate each chunk and their responses to forces individually.



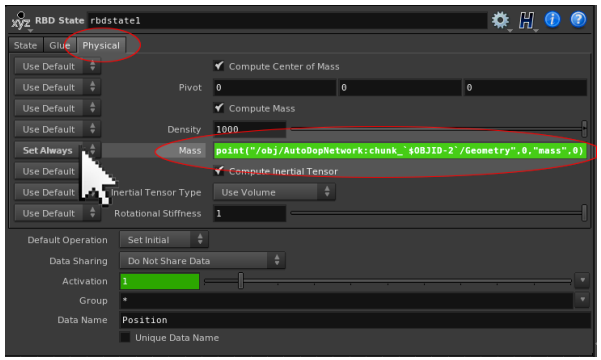
Each piece also has internal listings for all the attributes of the dynamic object. The **Geometry** listing for each piece in the **Details View** reveals the animated mass attribute created before the tube was imported into DOPs. When **PLAY** is pressed, this mass attribute for each piece updates, as each chunk turns green (due to the **Use Deforming Geometry** option activated on the **RBD Fractured Object DOP**). This animated mass information can be retrieved on a per chunk basis so that as each chunk turns green, it responds to dynamic forces.

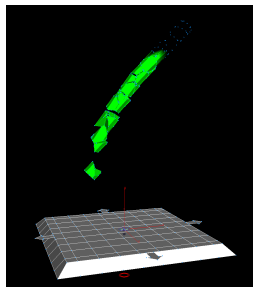


Append to the **RBD Fractured Object DOP**, a **RBD State DOP**. This operator can be used to animate and control RBD behaviours of the tube geometry over time (rather than simply their Initial State behaviours created by the RBD Fractured Object DOP).

Under the **Physical** section of the **parameters** for the RBD State DOP, activate the **Mass** parameter as **Set Always** and specify:

Mass **point("/obj/AutoDopNetwork:chunk_`$OBJID-2`/Geometry",0,"mass",0)**





This will activate the animated mass on each chunk individually, so as they turn green, they respond to dynamic forces. When **PLAY** is pressed, each chunk falls to the ground plane, as it turns green.

The animated mass value can also be utilised to drive other parameters on the RBD State DOP. Under the state section of the parameters, specify:

Set Always
Angular Velocity **if(ch("mass")>0,rand($OBJID)*500,0)** **0** **0**

When **PLAY** is pressed, each chunk now has rotational velocity assigned to it creating a more tumbling momentum when each chunk breaks off. When the chunks hit the ground however, they continue to roll around because of this constant application of Angular Velocity being generated by the RBD State DOP. Deactivating the RBD State DOP after the tube has turned green can rectify this.

In the **parameters** for the **RBD State DOP**, **RMB** on the **Activation parameter** and choose **Delete Channels** from the resulting menu. In its place specify:

Activation **if($SF>=50,0,1)**

This will deactivate the RBD State DOP after 50 frames. When **PLAY** is pressed, the chunks tumble rotate as before; however no come to a standstill after they land on the ground plane.

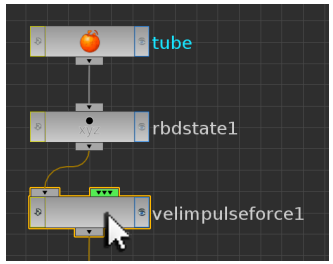
NOTE: $SF (Simulation Frame) is being used rather than $F due to DOPs being a contained environment separate from the main scene.

NOTE: Deactivation of the RBD State DOP after 50 frames does not affect the animated mass of the tube chunks.

ADDING VELOCITY

As a final step velocity can be added to the chunks to move them in a specific direction. If this is done in the RBD State DOP, again constant velocity would be applied to the chunks and turn off after 50 frames. This visually would result in all the chunks appearing to come under the influence of gravity at the same time. A more subtle application of velocity to the chunks can be achieved by using a **Velocity Impulse Force DOP**.

To the **RBD State DOP append** a **Velocity Impulse Force DOP**.



In its **parameters** specify:

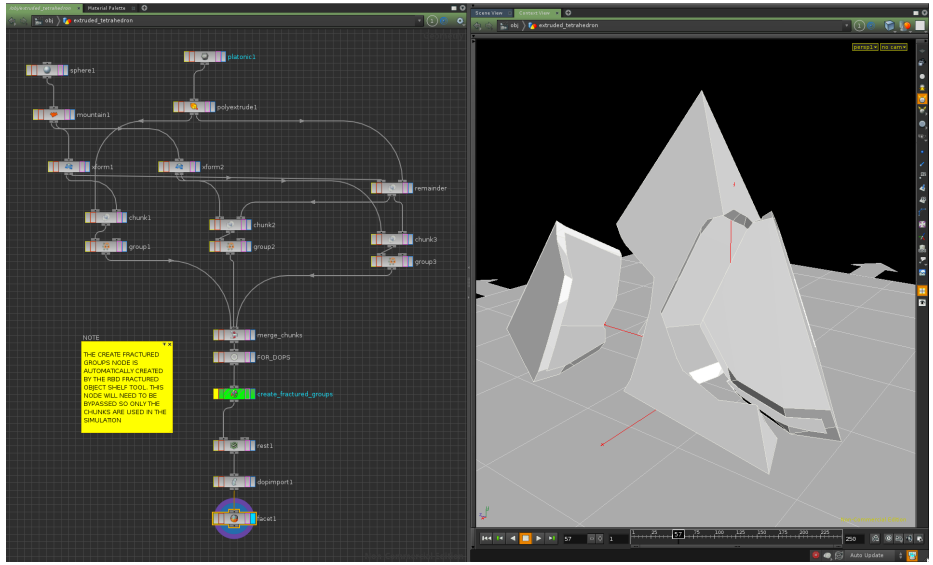
Velocity Change	**0**	**rand($OBJID)/2**	**rand($OBJID*3)/3**
Activation	**if($SF>=50,0,1)**		

When **PLAY** is pressed, each chunk now is propelled slightly in the Y & Z Axis; however falls more naturally under the influence of gravity and comes to a rest when all chunk forces have been exhausted.

See file H15_green_tube_dynamic_chunks_end.hipnc

USEFUL UTILITIES – THE COOKIE SOP

The **Cookie SOP** can be used to generate customised shapes for geometry fractures. This can be useful when more refined bespoke shapes are required (for example shards of glass) than can be generated by the regular automated geometry fracturing tools.



See file simple_rbd_cookie_explosion.hipnc