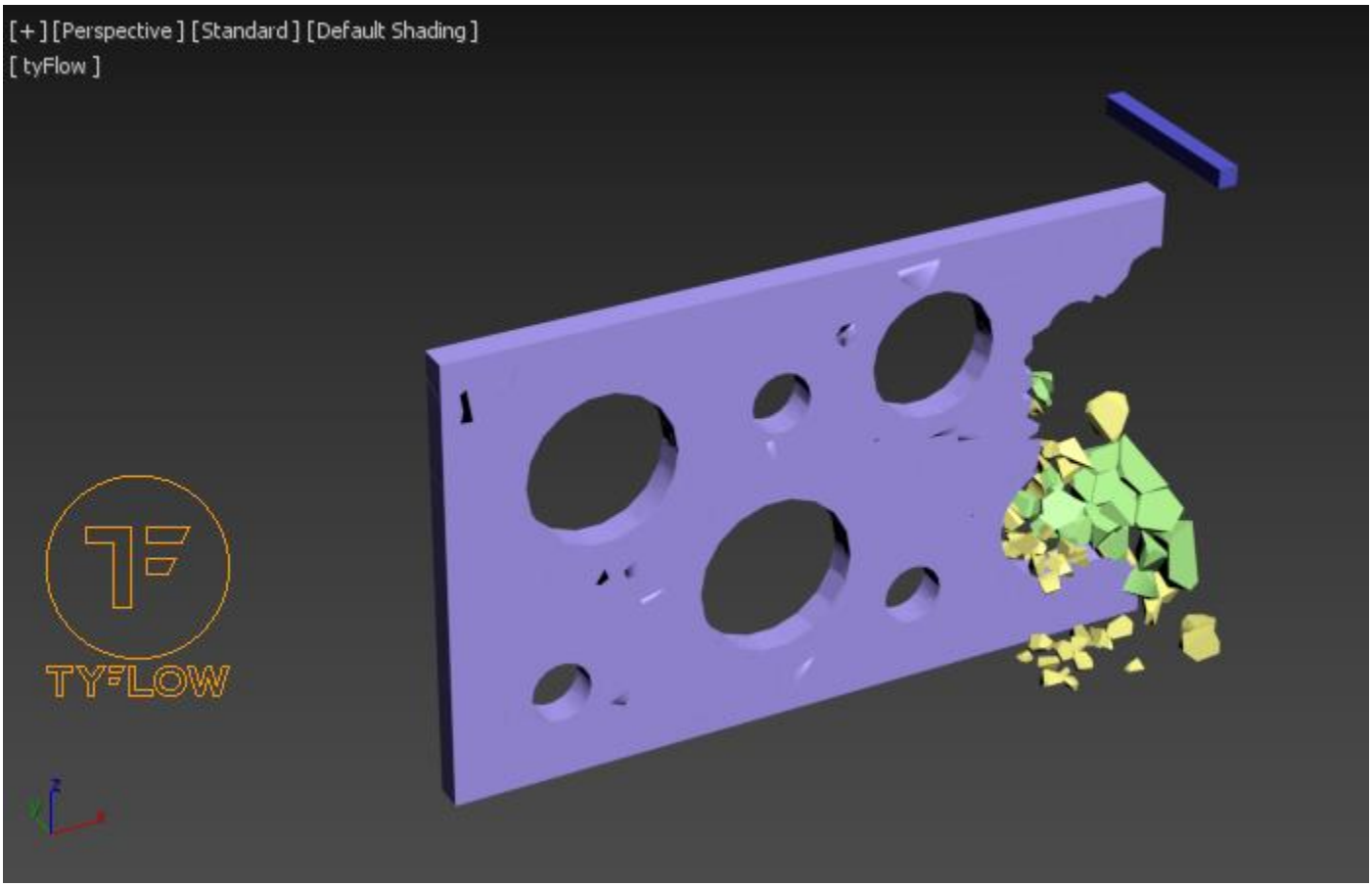
TyFlow Basics Tutorial

This tutorial takes a tyFlow example file (**tyFlow_bindSearch_001.max**), breaks it down and explains the reasons for the flow and shows how to re-create it from scratch.

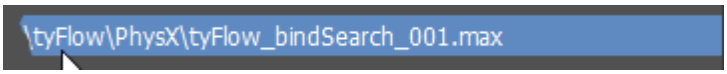
NOTE: A disclaimer, I am a beginner at **tyFlow**, that's the reason I am de-constructing the example file.

My understandings may not be perfect or the results may not be described a way a professional would. They are my understanding of what is going on based on analysis of what happens when I change parameters and (or) enable / disable operators to see the resulting changes. For me, this was a great way to learn and understand this flow.



Dissecting Bind Search Example File

This the example file I deconstructed, you can download the example files from the tyFlow Site.

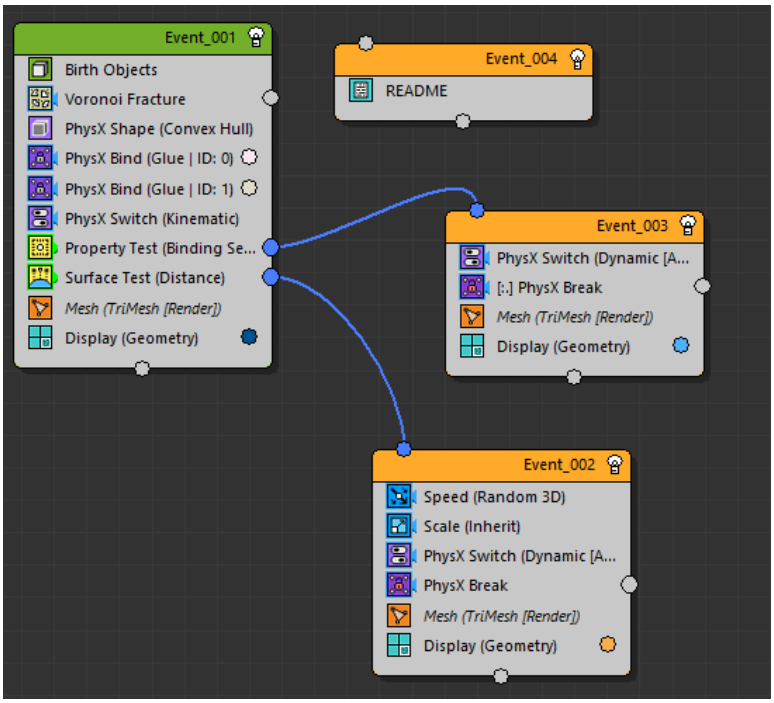


Creating the basic scene setup

In an effort to learn **tyFlow** I decided to totally reverse engineer the example file provided with **tyFlow** from the absolute beginning, this tutorial / deconstruction was not really made (Intended) to be shared as a tutorial, but I decided to get some screen grabs and write some notes, as I went along. These notes and this document were my attempt to help get me to understand the reasoning behind the selection of operators in this particle flow system. The document does walk through each step and as such can be used as a tutorial to re-create the final sample file created by Tyson.

The below image shows the final full tyFlow Particle system for this animation as per the sample file.

Nothing is assumed, in this tutorial, except fairly basic knowledge of 3ds max, we start the creation of the scene from an empty new file.



The text from the **readme** included in Tysons example file:

In this example we using a binding search Property Test to check to see if each particle's network of PhysX bindings is connected to the ground plane. Ground bindings are marked with ID=1, so if a particle's network of bindings doesn't ever reach a binding with ID=1, we know it's not connected to the ground any more.

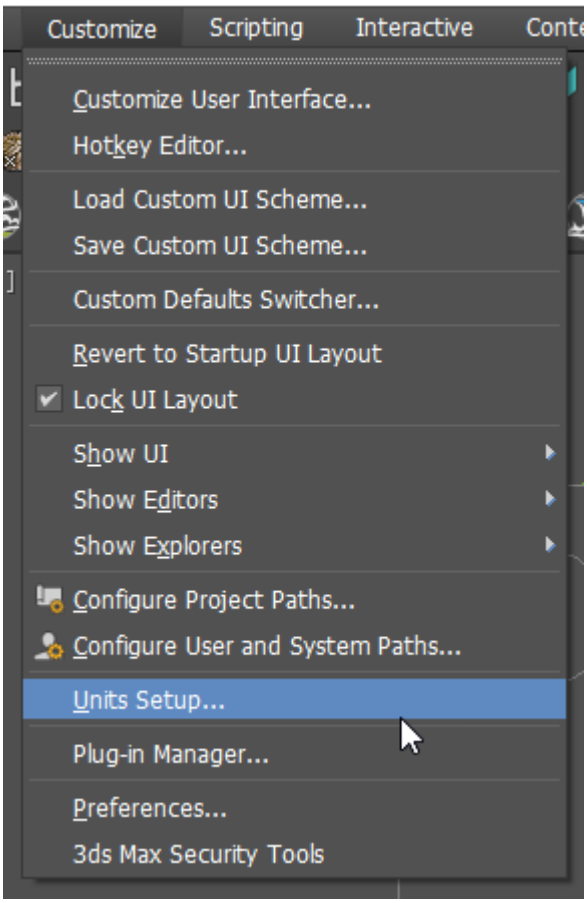
When a particle's binding network does not reach any bindings attached to the ground, it is sent on to another event where it is switched from kinematic to dynamic.

Using this method, we can prevent the existence of floating chunks in our simulation, even though by default all our chunks are kinematic and will not fall. Without the bind search Property Test, our simulation would result in many floating chunks as the pieces below them fall away.

Note: the binding search algorithm is multi-threaded, but still slow when a lot of particles are being processed. For that reason, we use the frame skipping setting in the Property Test's timing rollout to our advantage. It is set to only evaluate the operator once every few frames. This will speed up the simulation overall, and the difference in results is negligible, since it's not necessary to do the binding search every single step of the simulation in order to get a believable result.

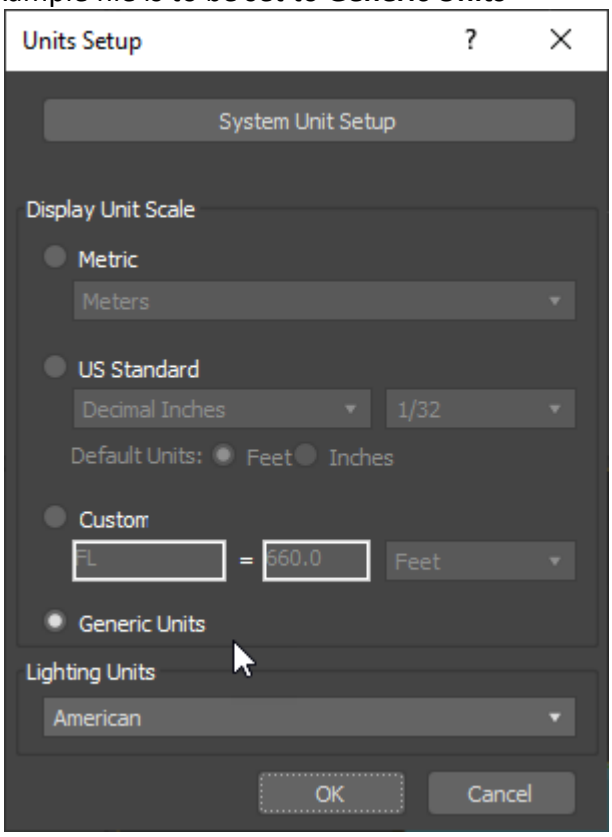
UNITS SETUP

To make sure we get the same results, max should be set to the same units as the example file:



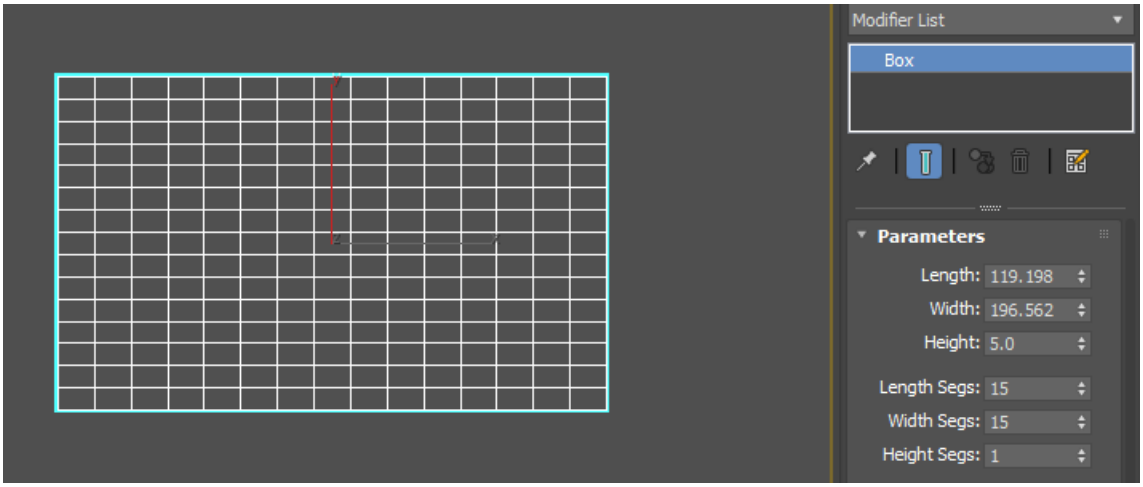
This can be changed in the units setup, under the customize menu.

The example file is to be set to **Generic Units**



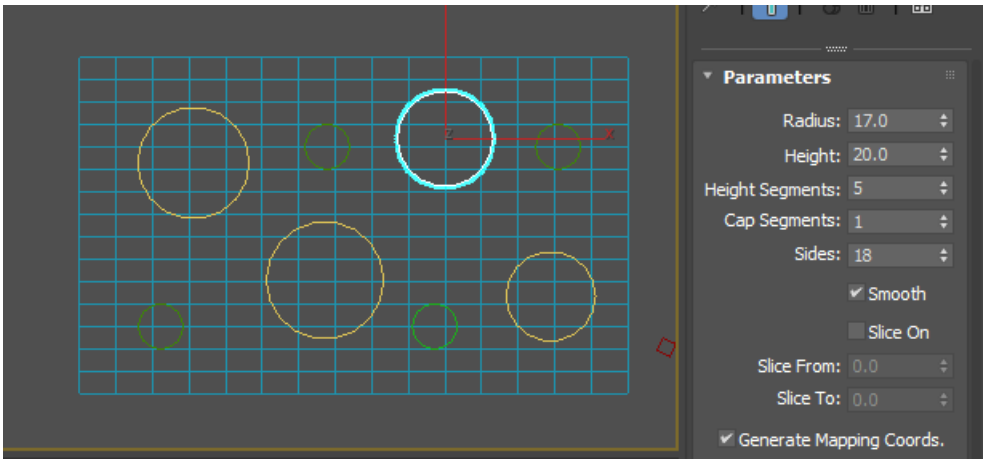
Make a Box

In the front viewport, I made a box (L: 120 W: 197 H: 5 will be fine)



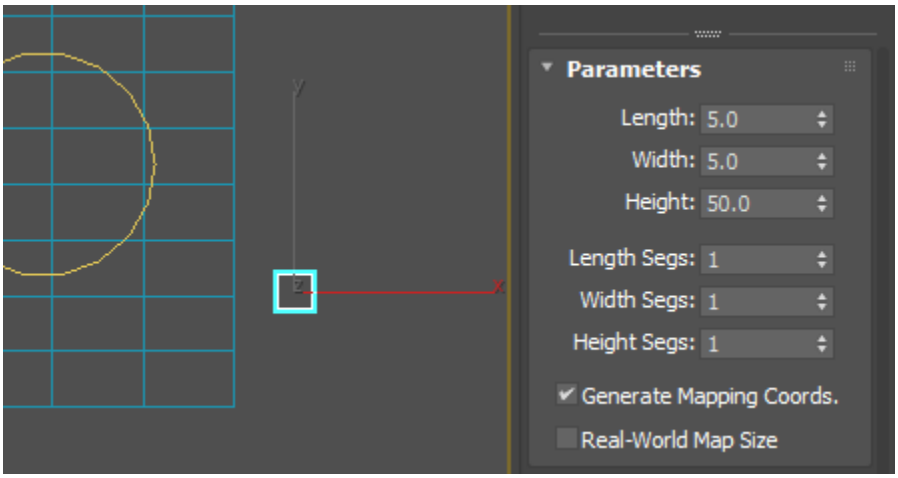
Make the Cylinders

Then in order to make our box with holes cut out we need to make the cylinders, again I did this in the front viewport. I used a mixture of create cylinder and copy paste, changing the radius to suit.



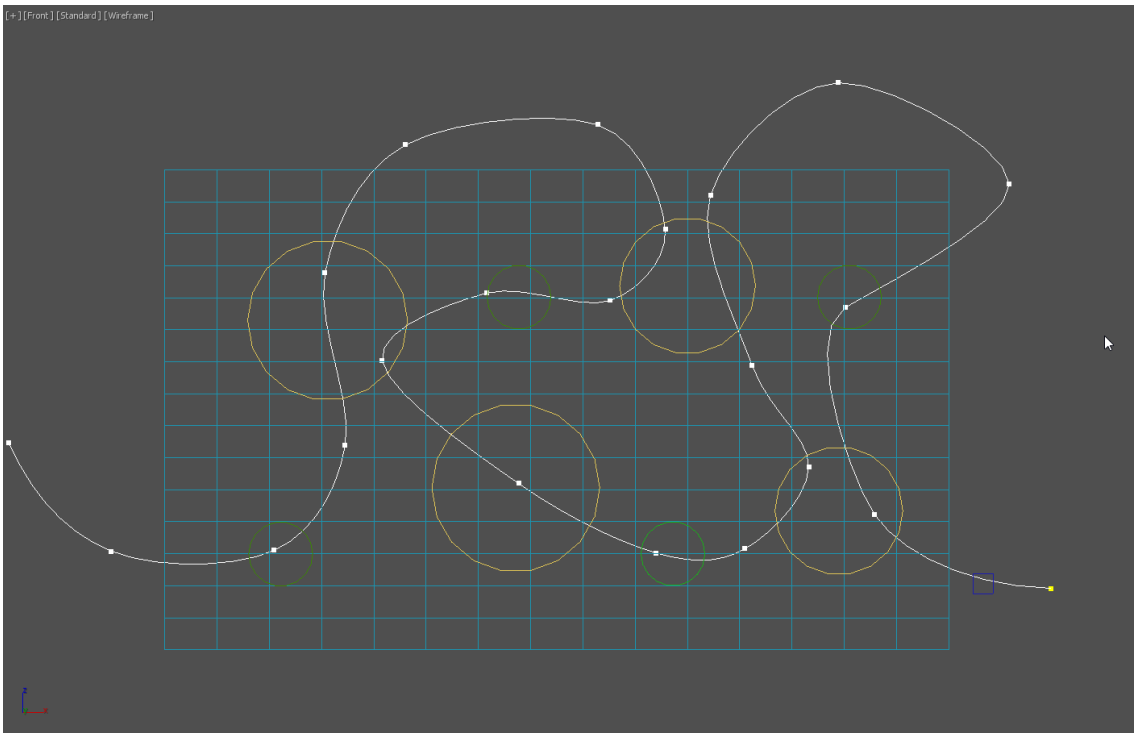
Make the Breaker Box (Bar)

Then I made the box, that is going to act as the breaker bar slicing through our cheese holed larger box.



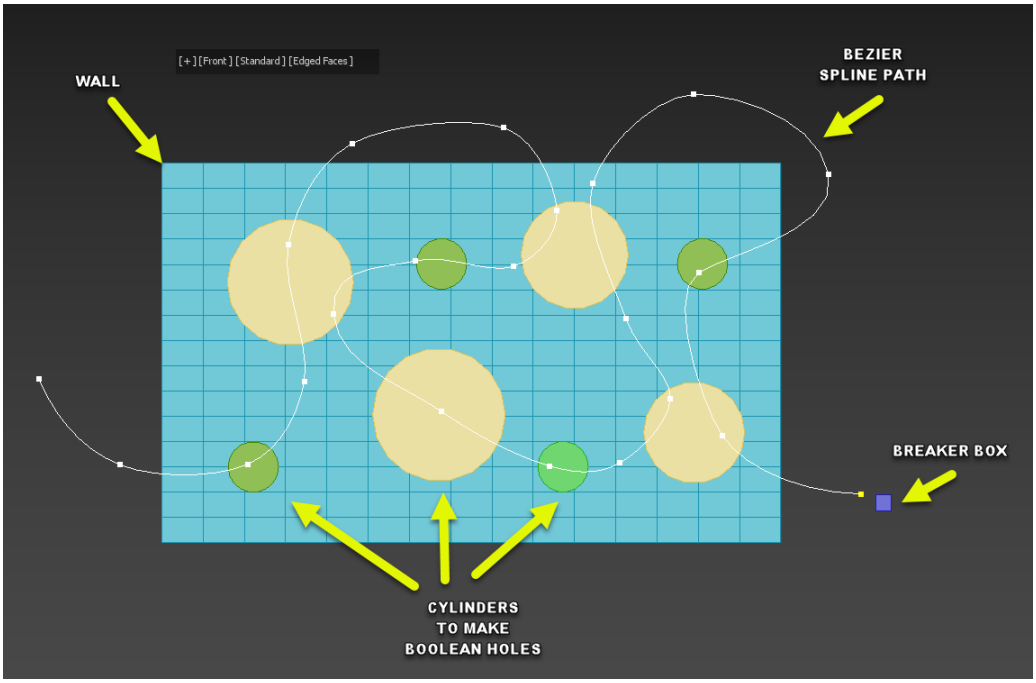
Draw a Spline that cuts through the holes

Then I drew a Bezier spline, trying to make the path follow Tysons original path using Bezier curves, you can roughly see below, where I clicked each point of the spline curve. I then went back in and modified the curve and turned some of the vertexes into Bezier points and adjusted the smoothness.



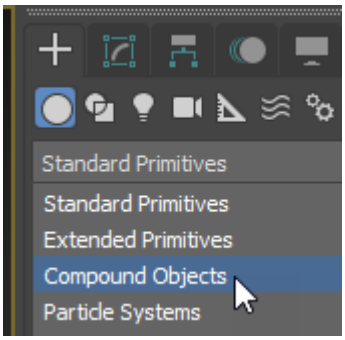
All the elements of our animation

Below, we have the breakdown of our scene, two boxes and eight cylinders and one spline path.

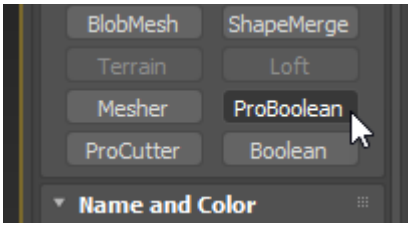


Create the compound Boolean wall cutouts

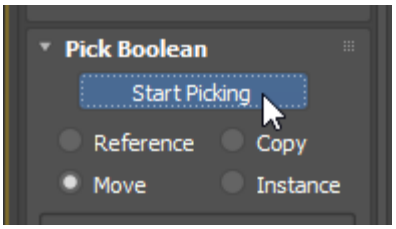
Now, we need to turn our wall/box into a cheese like box with lots of holes, so we need to make a compound object from our box and cylinders.



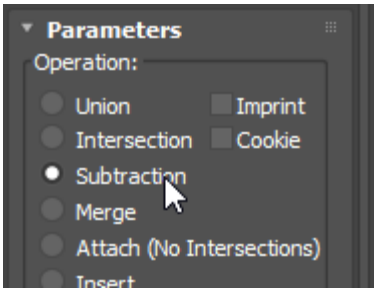
In the create panel, we select create compound object.



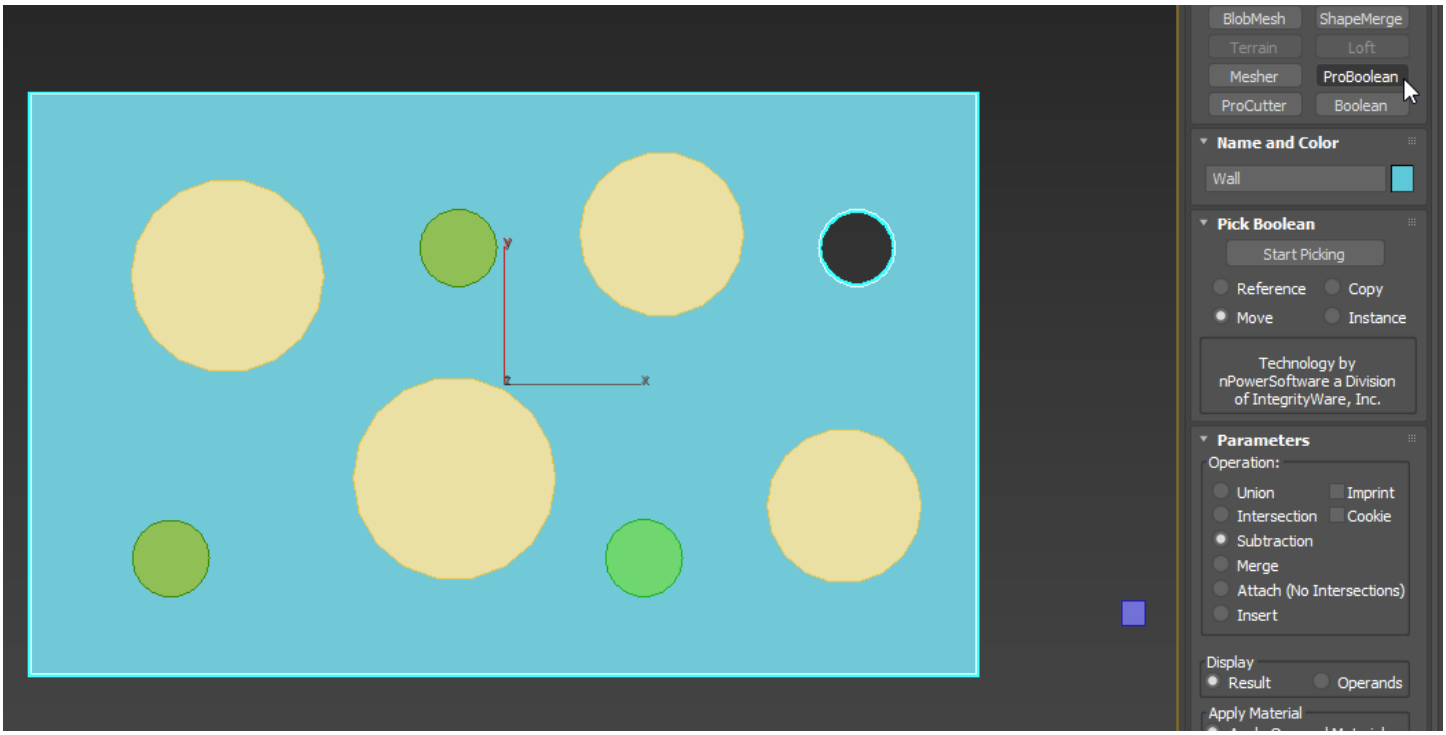
Then I used proBoolean to make my boolean object, I had the wall object selected already.



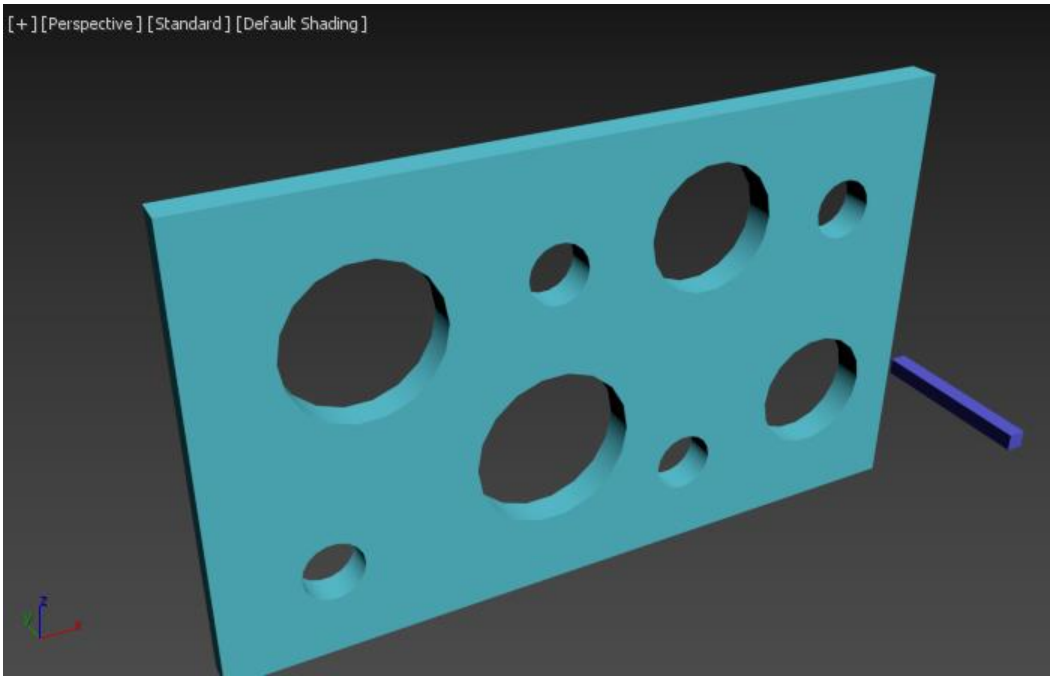
Then start selecting the objects (cylinders) I want to subtract from the wall mesh



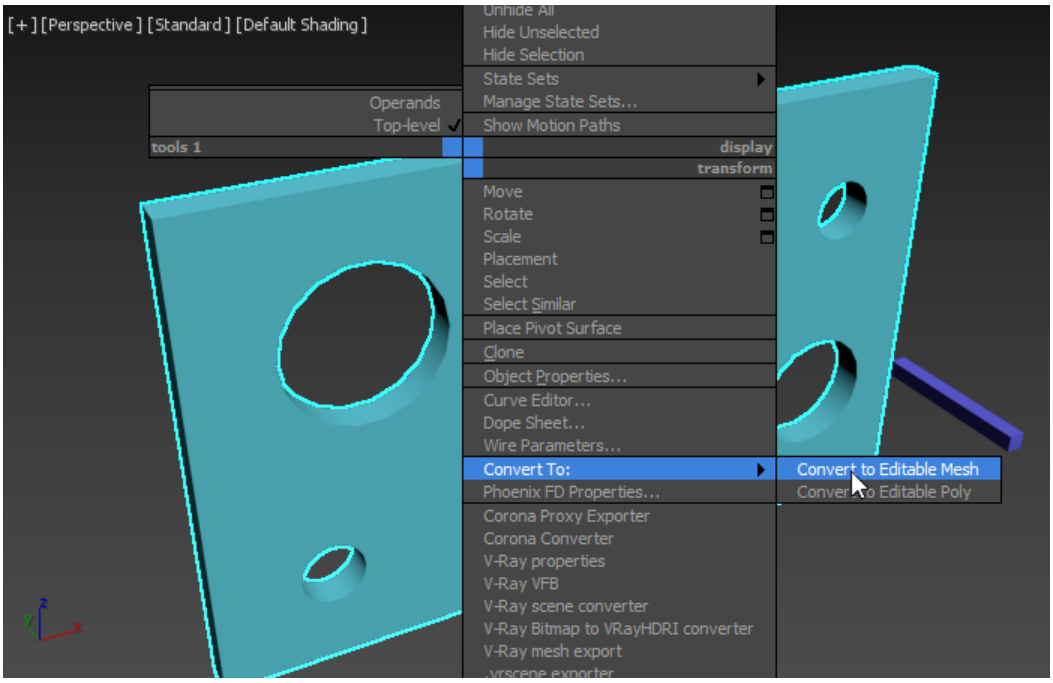
Select all cylinders until you have your final Boolean object



We will end up with a wall (box) that looks like this:

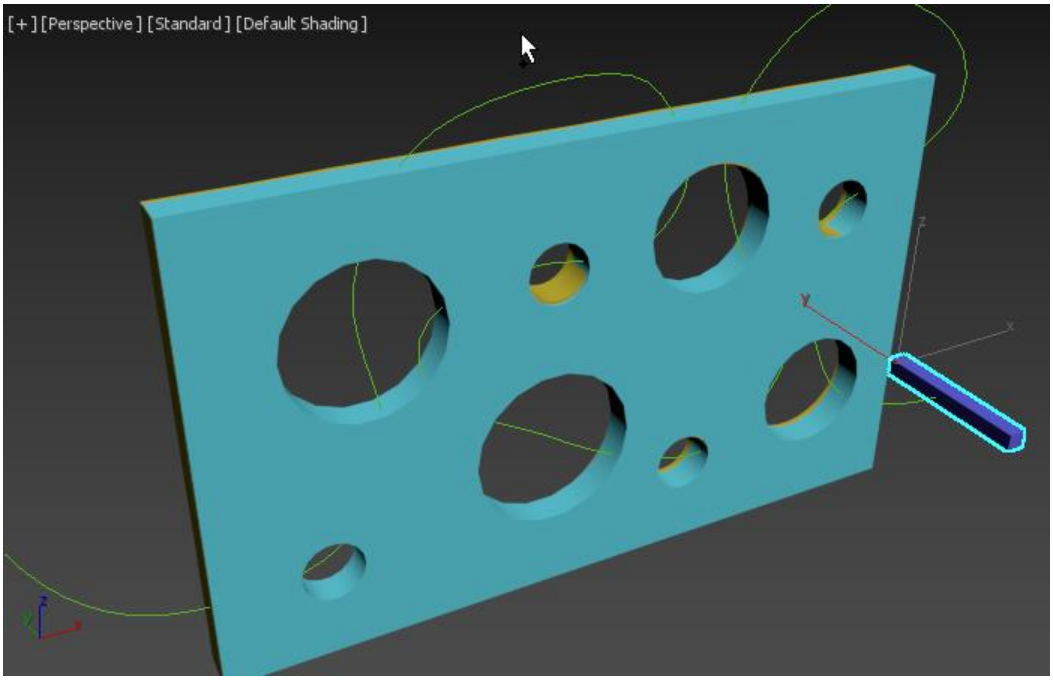


I then converted my compound object into an Editable Mesh object, by right clicking and select the menu option.



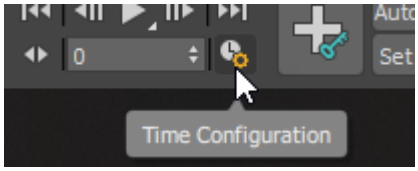
So, now our scene contains three objects:

- The cheese like wall with holes
- The breaker bar (box) that will cut through our wall
- And the spline we will use as the path for the breaker bar.



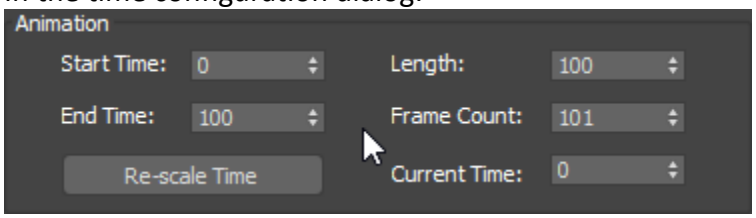
Adjusting the Animation Time Length

Note: If you started with a new blank scene to create the whole animation from scratch, then the default time length would have been 0-100 frames, this needs to be adjusted to 250 frames to match sample file.

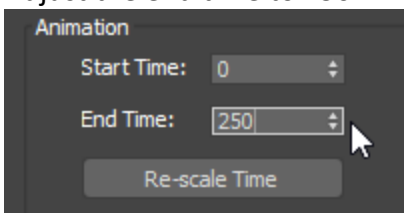


In the bottom toolbar, click the time configuration icon.

In the time configuration dialog:



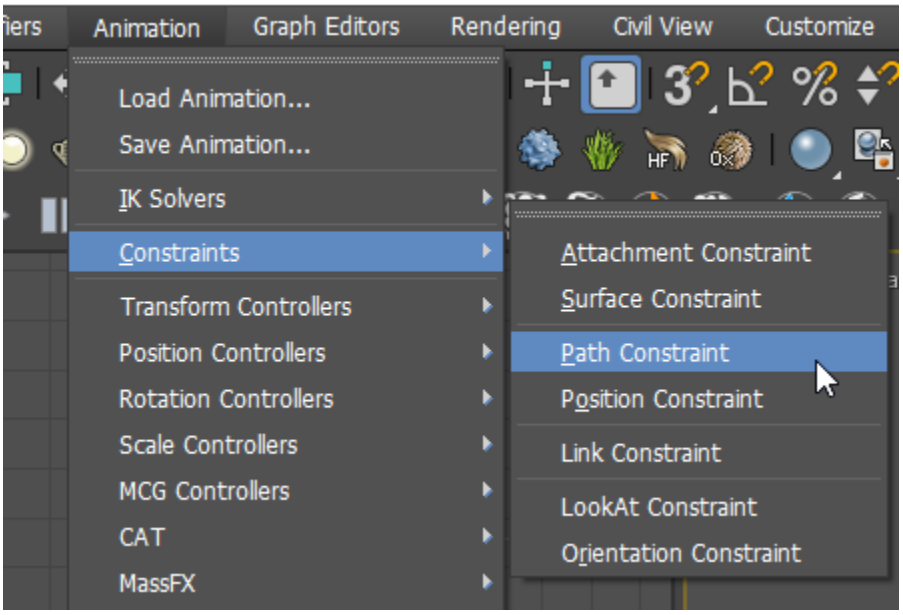
Adjust the end time to 250



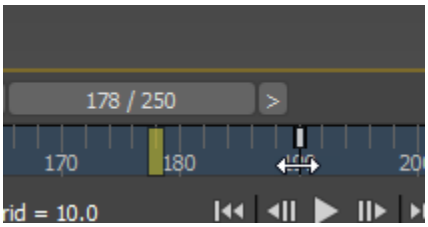
Now our scene has the basic objects and time configuration to match the sample file. The next step is to animate our breaker bar along the spline path before we introduce the particle system to our file.

Creating the Breaker Bar Animation

Next we need to animate the breaker bar (box) to move along our spline path and cut through all our holes. With the breaker bar selected, go to the menu **Animation / Constraints / Path Constraint**

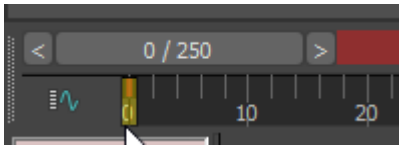


By default, it will animate it across the total frames in the scene (in this case 250), we will move the right animation key to frame 190 to match the original sample file. So now the breaker box will start at the beginning of the spline at frame 1 and finish at the end of the spline at frame 190.

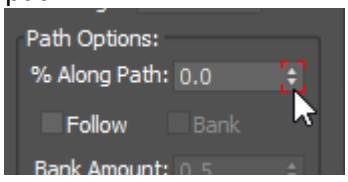


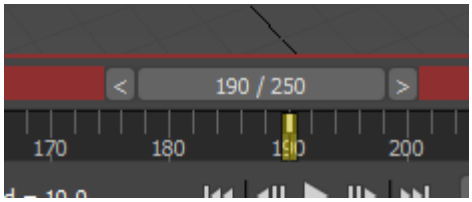
If you need to adjust the start and end positions of the breaker box along the spline, you can do it by selecting the box and going back to the menu **Animation / Constraints / Path Constraint**

In the Path Parameters rollout that appears you can adjust the % along path.

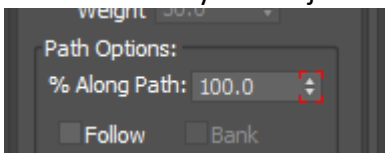


At frame 0, the breaker bar, should be at the beginning (right) side of the wall, if it's not, go to frame 0 and enable auto key and adjust the % along path



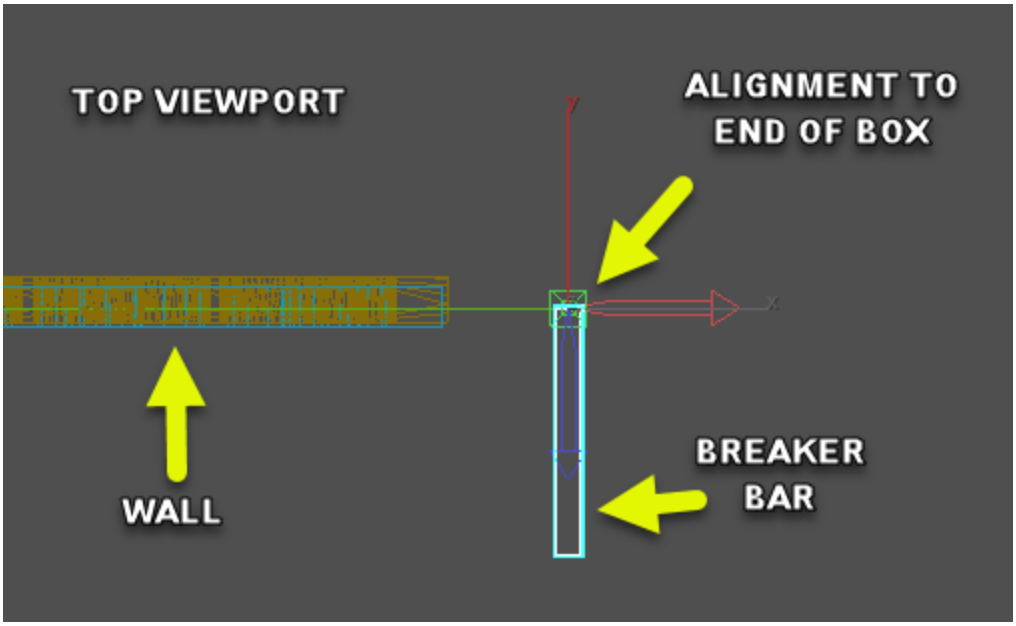


At frame 190, the breaker bar, should be at the end (left) side of the wall, if it's not, go to frame 190 and enable auto key and adjust the % along path

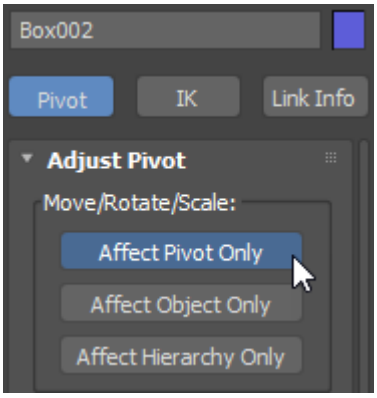


Adjusting the position of the breaker bar on the spline path

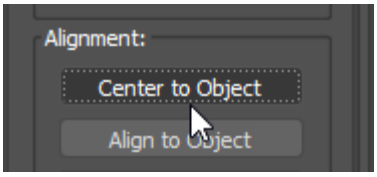
When I added the animation path constraint to my breaker bar box, the box aligned to the end of my box as shown below:



I needed to adjust the pivot point to be the middle of the breaker bar so it is positioned correctly in relation to the wall and centered on the spline path:



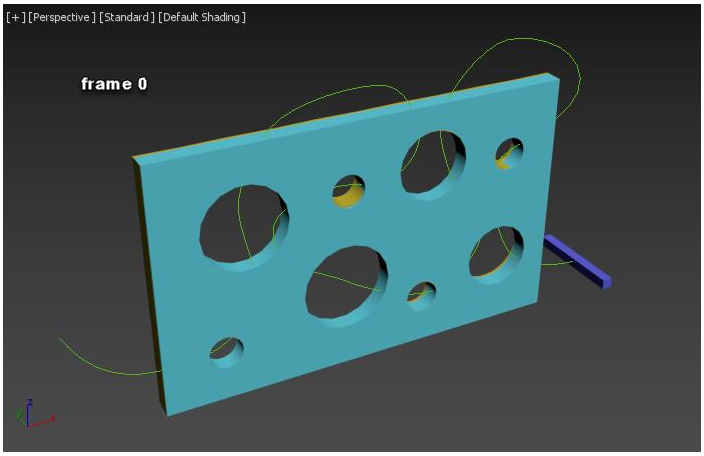
With the breaker box selected, go to the **Hierarchy Tab** and select **Pivot / Affect Pivot Only**, then select **center to object**:

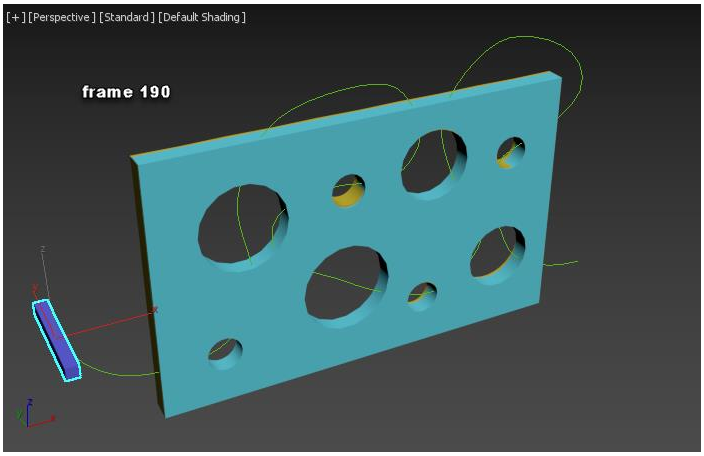


Deselect Affect Pivot Only, before exiting the Hierarchy Panel

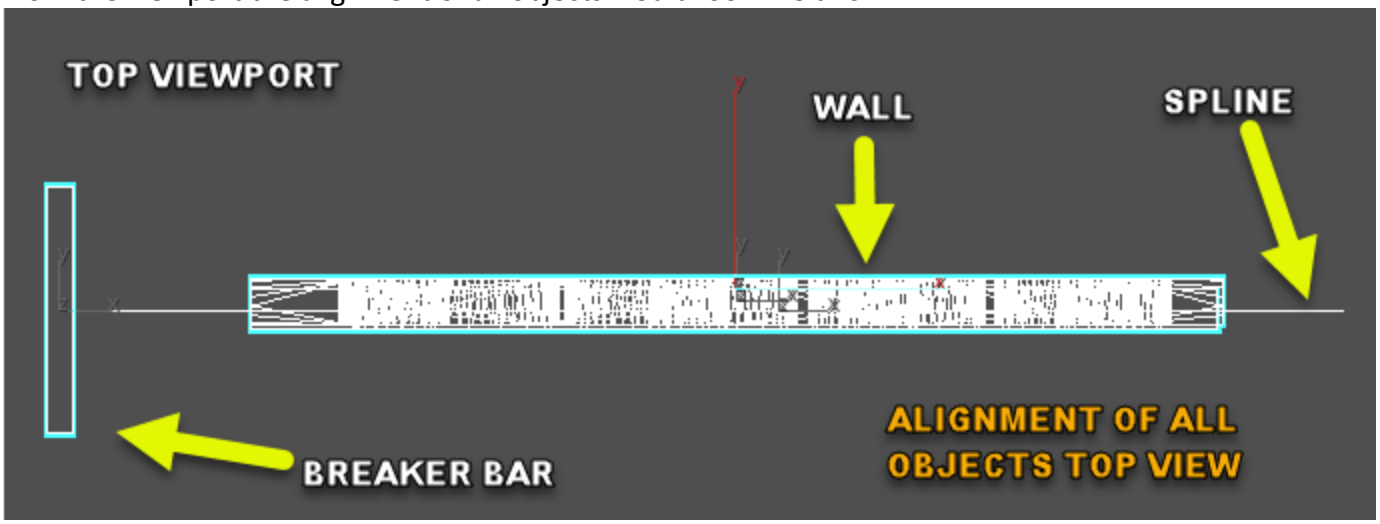
Final Object Layout before tyFlow

We now have everything ready to be able start our particle animation. The start file should look like this.





From the viewport the alignment of all objects would look like this



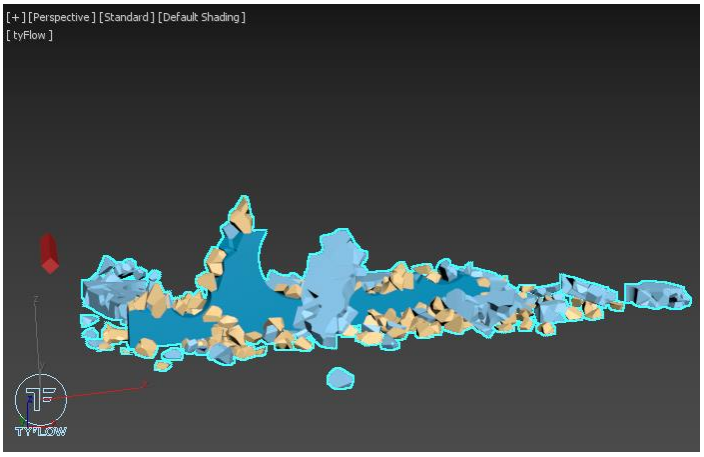
Let's add a particle system to this baby!

If your start file doesn't match the progress so far, load:

"**tyFlow bindSearch Ready For TyFlow.max**"

Creating a tyFlow System

Understanding Why Three Events

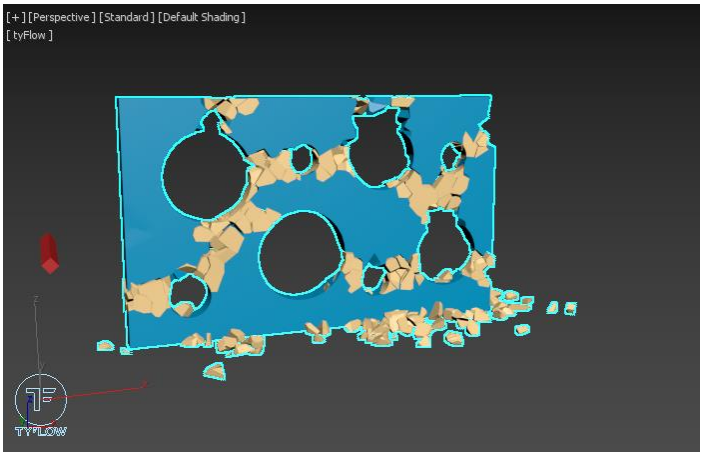


Final Result:

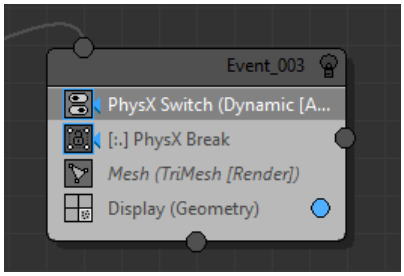
This is the final (250th) frame of the render sequence with the full particle system in tact as per original example file.

The wall is fully collapsed and we had two separate interactions with the wall.

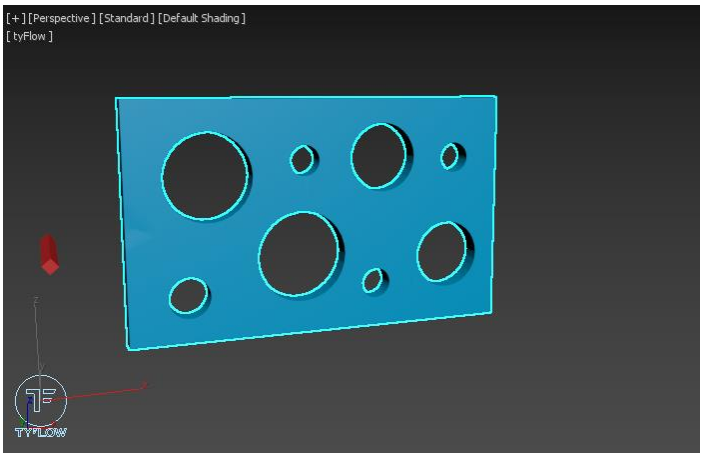
- **Event_002** tells the flow what to do with segments touched by the breaker bar
- **Event_003** tells the flow what to do with the segments left floating in the air, **Event_003** is dependent on **Event_002**



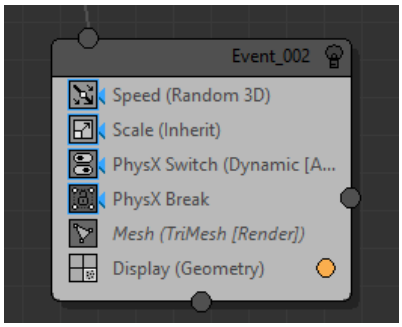
Result with **Event_003** disabled.



We can see wall segments suspended in mid air without **Event_003** being active.



Result with **Event_002** disabled



Event_002 is the event that handles the breaker bar touching segments of the wall, when this event is disabled, we never go to **Event_003** as there are no floating wall segments.

You can visualize the flow this way:

Event_001

- Create a copy of our wall with particles and a tyFlow Object
- Break the **tyFlow** copy of the wall into segments and turn it into a PhysX object
- Setup our wall for PhysX behavior and interaction with the breaker bar
- Apply Binding Operators to act as the glue to hold the wall segments together until something breaks that bind and to give us a way to test bind conditions
- Tell the wall to be Kinematic.
- Perform 2 tests and do something based on those test results

TEST 1 – SURFACE TEST

If this test is true, go to Event_002

TEST 2 – BINDING ID TEST

If this test is true, go to Event_003

Event_002

The breaker bar has touched sections / elements of our wall

- Add some speed to help the wall break down
- Scale the wall segments slightly smaller to help them break away and not get stuck
- Turn the wall back into dynamic object and be affected by the PhysX Solver
- Break all bindings so touched segments fall apart

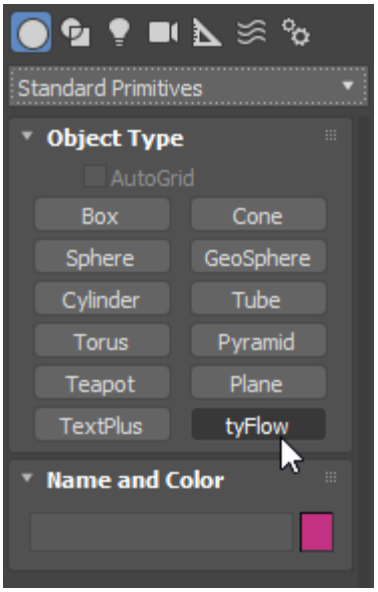
Event_003

There are floating wall segments, that shouldn't be hanging mid air

- Switch them from Kinematic to Dynamic Objects
- Break the binds that hold them together

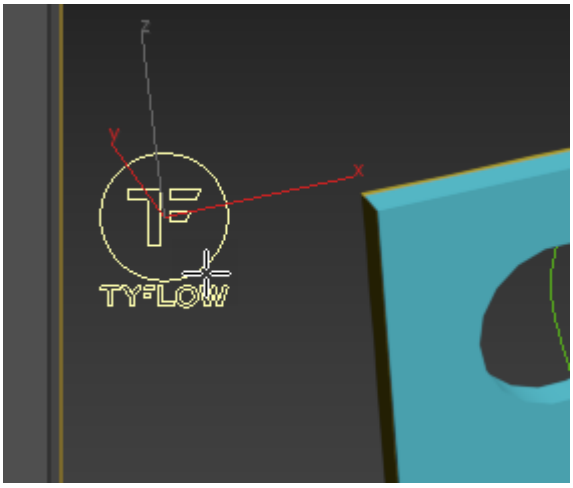
NOTE: if event 2 never happens, there would be no floating wall sections and no need for event 3

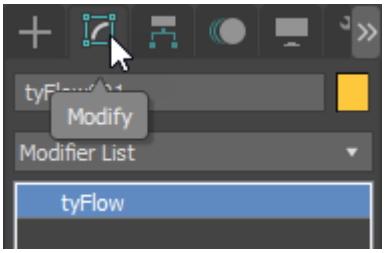
EVENT 001 – MAIN FLOW



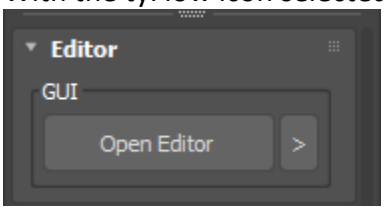
In the Orthographic Viewport, create a tyFlow node:

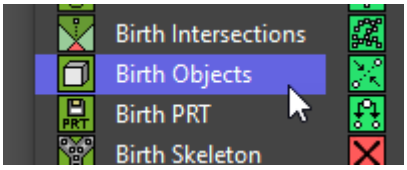
Go to the create panel, select tyFlow from the standard primitives, click and drag to create the icon



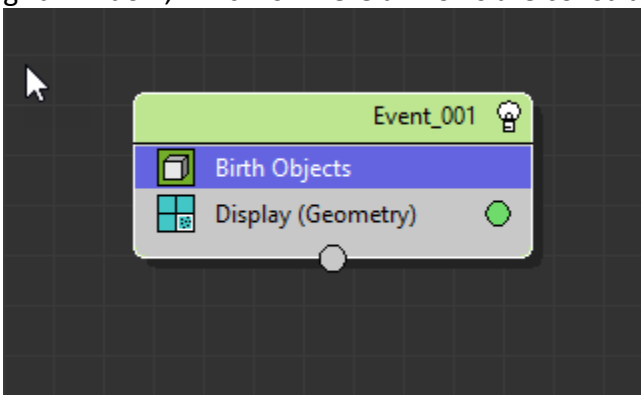


With the tyFlow icon selected, go to the modify panel and click open editor



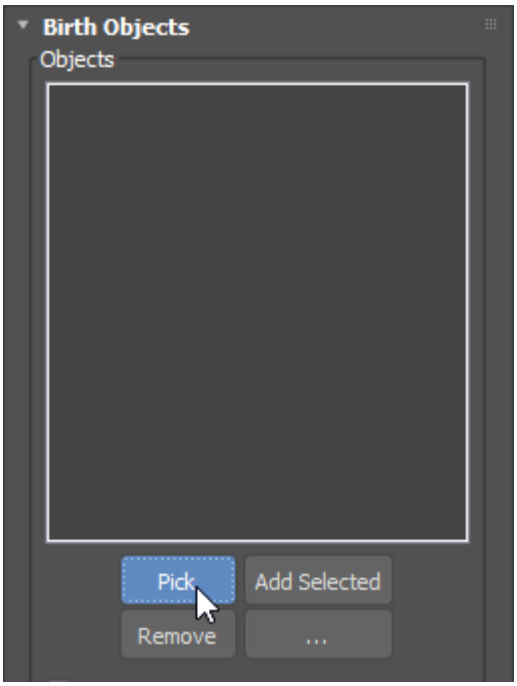


In the open editor window, click and drag the Birth Objects operator to the grid window, which is where all flows are constructed.

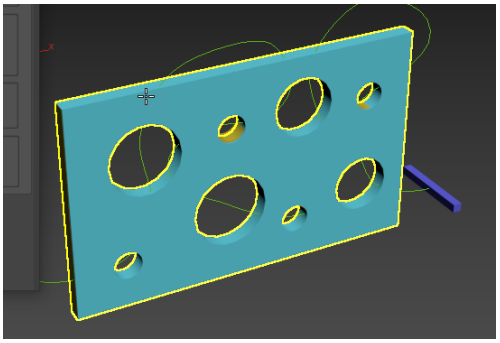


NOTE: the **Event_001** display color for this flow defaulted to green for me, later at the end of creating this **Event_001** flow I will change it to BLUE.

You can do this now if you want, just click on the colored circle and pick a blue color.

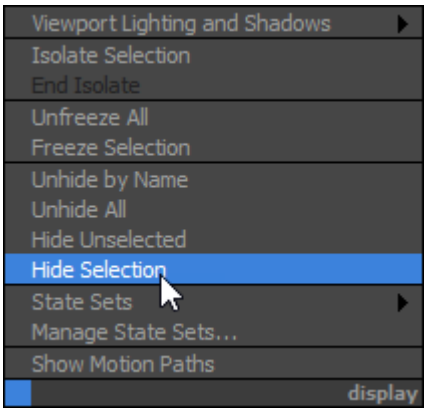


From the Birth Objects Rollout, click the pick button and select our wall object:



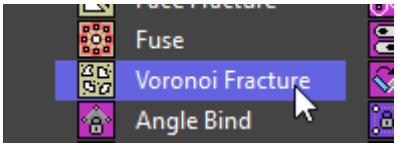
Our wall object, has essentially been cloned and will now be the source object for the particles, its part of the tyFlow object.

We can hide our original wall (box), right click on it in the viewport and select “Hide Selection”

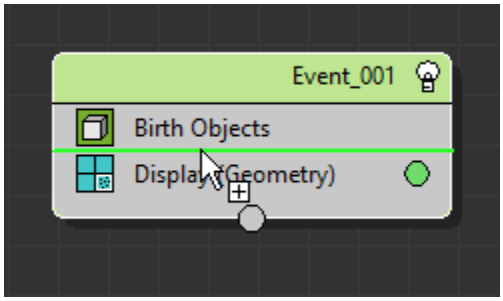


We can also do the same for the spline path:

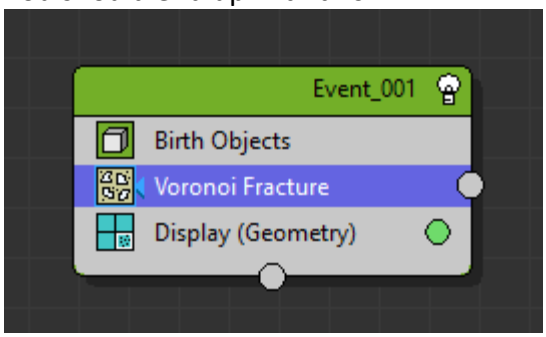
Adding Voronoi Fracture to our Flow



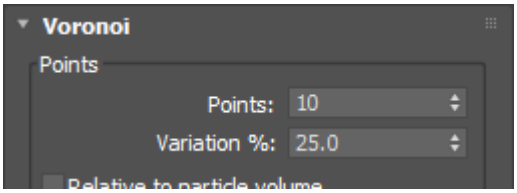
From the operator selection window choose Voronoi Fracture, drag this operator to your event 001 flow and drop below the birth operator.



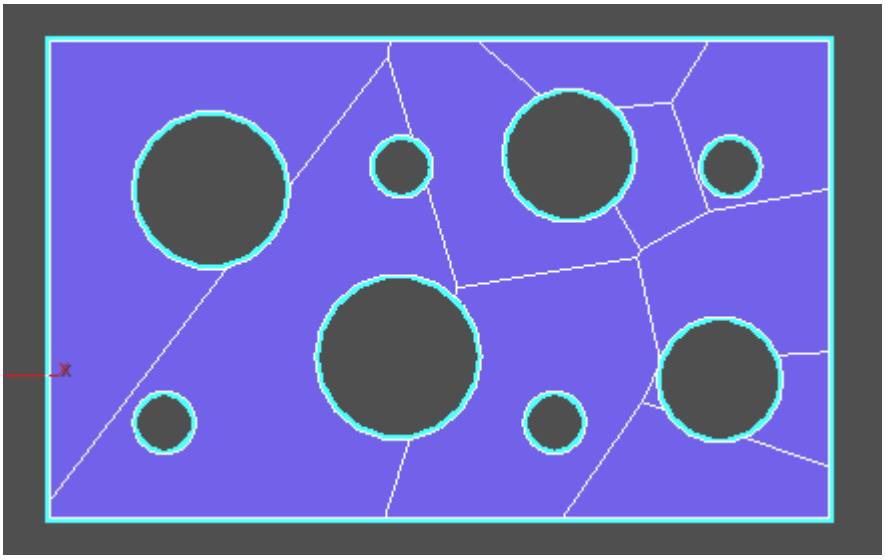
You should end up with this:

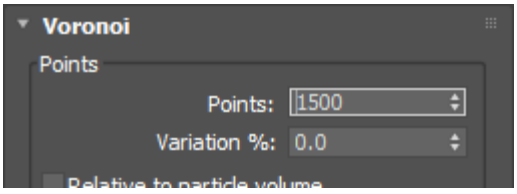


Now when we look at our wall (remember it's a tyFlow copy) you will see 10 fracture lines, assuming your tyFlow defaults to the same Voronoi settings as mine.



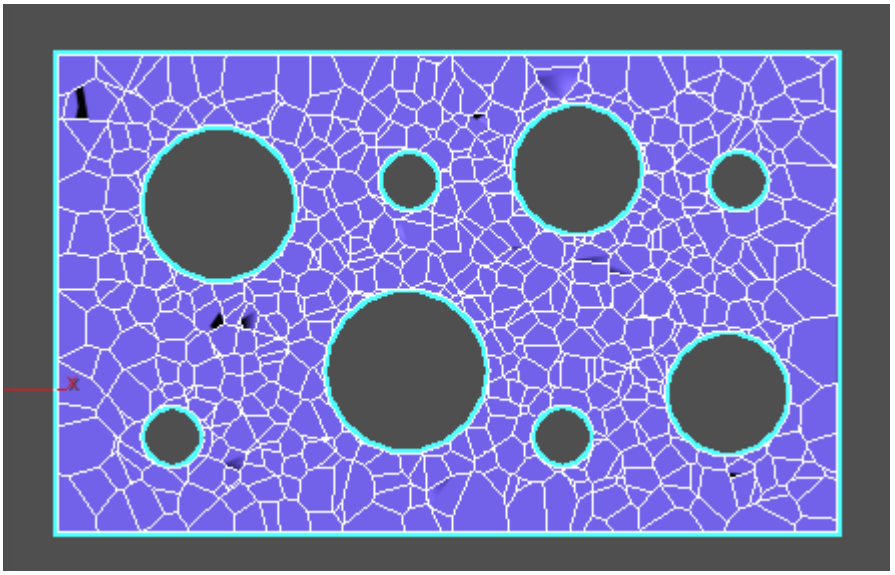
Creates this result:





We will change the **Points** and **Variation %** to match the settings in the sample file:

Creates this result:

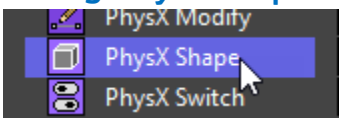


NOTE: (PER MANUAL)

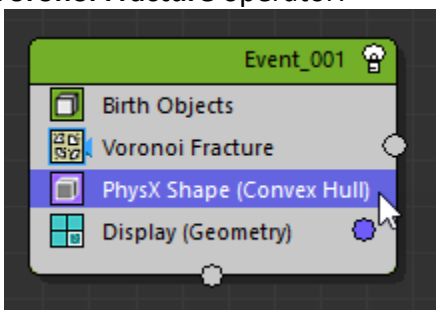
Points: the number of points used to create fractures. Roughly the number of resulting fracture meshes, depending on the location of the point cloud.

Variation %: the per-particle percentage of variation to apply.

Adding PhysX Shape to our Flow



We are now going to add the **Physx Shape** operator to our flow, below the **Voronoi Fracture** operator:



NOTE: (PER MANUAL)

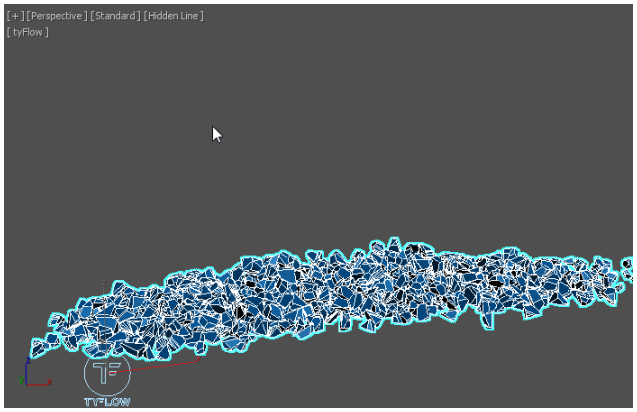
The PhysX Shape operator allows you to convert particles into PhysX rigidbodies.

Convex Hull: a convex hull encapsulating particle shape mesh vertices.

We are not changing any of the default parameters

Adding PhysX Binds to our Flow

Looking at the example file, I am trying to understand why two PhysX Bind Operators are added. Doing a test to understand the example system we can see the following:



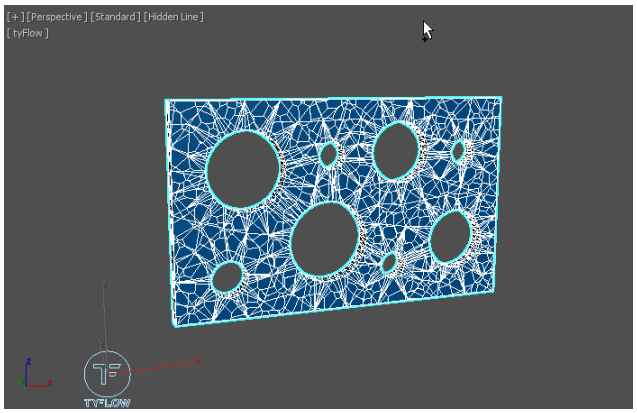
Original sample file, frame 40 of the animation, no bind operators applied

NOTE: (PER MANUAL)

The PhysX Bind operator can be used to create bindings between PhysX particles

INFO:

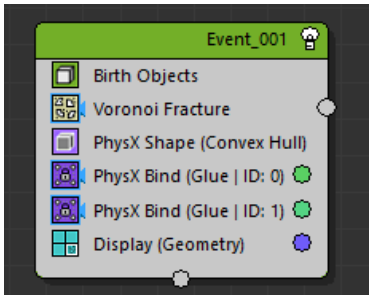
PhysX bindings are not solved by this operator, only created. PhysX bindings are solved by the global PhysX solver at the end of each simulation step. PhysX bindings created by this operator will persist between events.



Original sample file, frame 250, 1st bind operator applied glue ID 0

So, what we can see, is before the bind operator was applied the wall collapsed, even without our breaker bar actually being the cause, the bind operator holds the wall together.

The same result after 2nd bind operator applied glue ID 1, the wall stays together, there is further analysis of this below:



So, our flow should look like this at the current stage of creating the flow.

Notice the ID difference on the two bind operators. The Glue ID, 1 is 0, the other is 1

Understanding the PhysX Bind Operator

I deviate from the tutorial steps here to understand the two PhysX Bind Operators added:

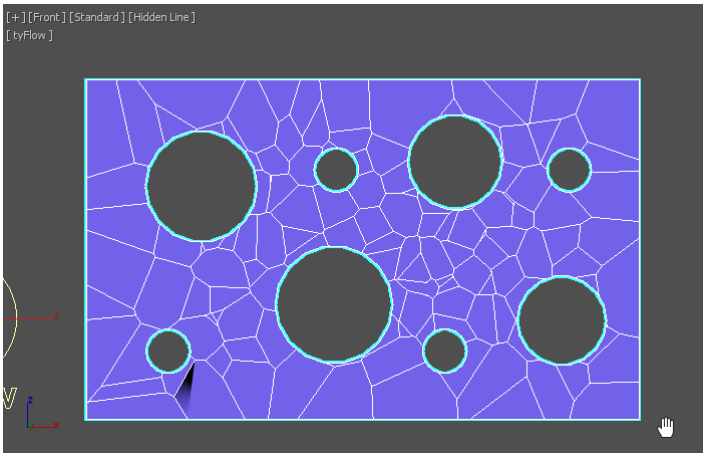
Breaking down the bind operator a bit further, we can do some tests and see the following:

For the purpose of this test I have changed the number of points in the Voronoi Fracture to 150 from 1500 to make it easier to see the connections and the effect the bind operation has on this flow. I have also changed PhysX Switches to be Inactive, so the broken away parts of the wall won't fall or move.

Event Colors (Display Geometry) :

Event_001 Blue
Event_002 Yellow
Event_003 Green

FRAME 0

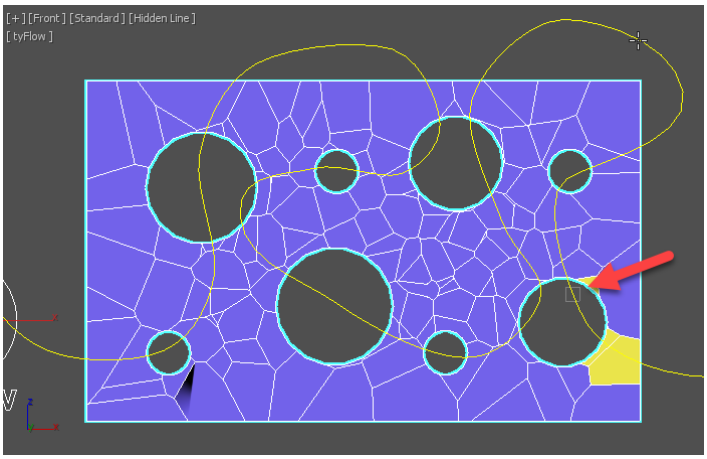


At frame 0, the whole wall is blue, indicating neither the surface test, or property test have been qualified as true.

The breaker bar is only just starting along the path, it's about to hit the top of the bottom right hole. The fractured elements touched by the breaker bar are turning yellow, indicating the surface test has passed them to **Event_002** (yellow)

The red arrows point to the breaker bar position along the path

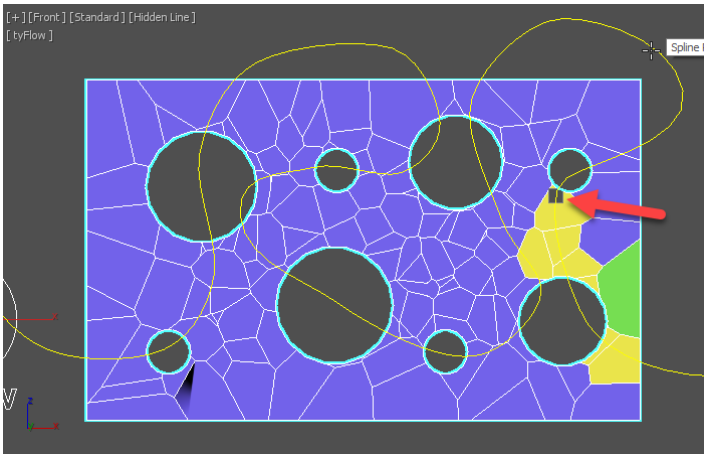
FRAME 14



The breaker bar is now about to hit the top right smaller hole. What we learn at this point is, the surface test operator is turning any fracture element that is touched by the breaker bar yellow and the orphaned section green.

Green is **Event_003**, so it tells us that based on the Property Test with binding search set to 1 that the green element is not in contact with the ground.

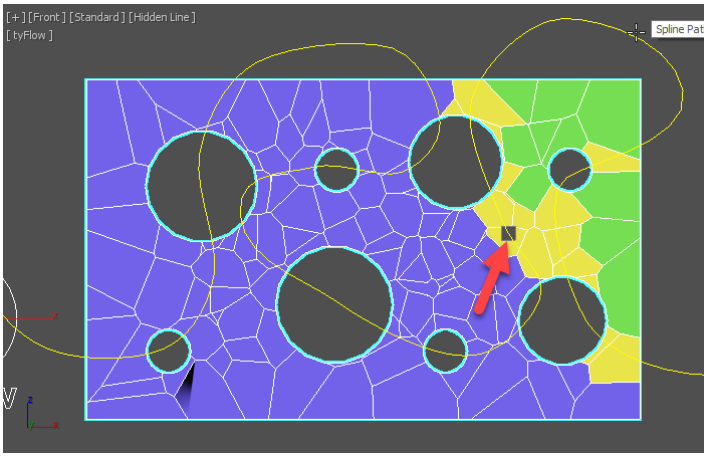
FRAME 22



Ok, so we get the drift:

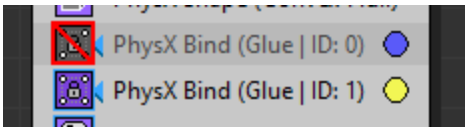
- Contact with the breaker bar equals yellow
- Orphaned, not in contact with the ground anymore equals green
- The blue, has not been hit by the bar and is still in contact with the ground

Note: the segments that turn yellow are selected based on the surface test distance, ours is set to 5. If we increase this size, more segments will turn yellow, even though the breaker bar didn't actually touch them. So, you need adjust the surface test distance for the effect you want.

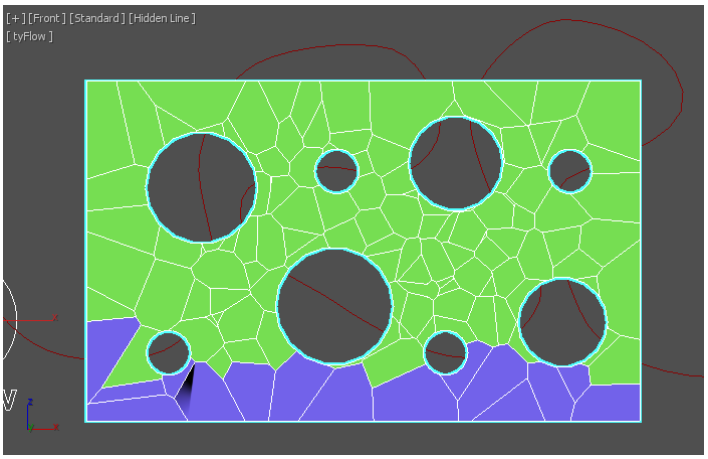


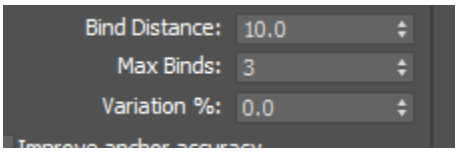
One final dissection of our example:

If we turn OFF the PhysX Bind (ID 0) operator, we get the result to the right

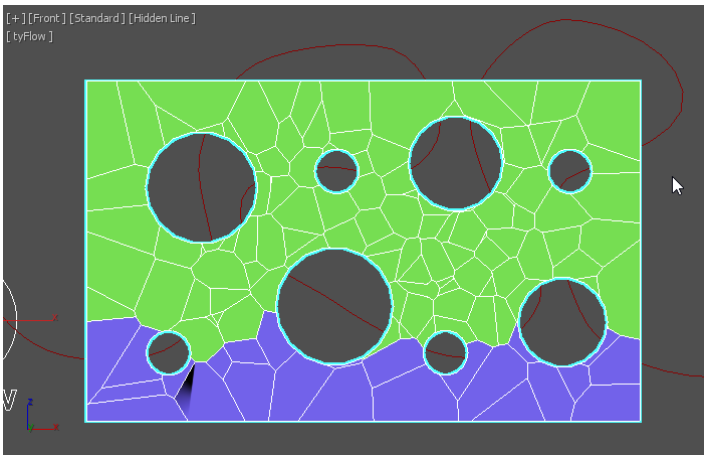


What this confirms is, that without the PhysX Bind (ID 0) operator, all elements out of range of the PhysX Bind (ID 1) are now identified as not touching the ground, the length / distance the blue extends is based on the bind distance in the rollout for our PhysX Bind (ID 1) operator. Ours was set to 10.





If I increase this to say 20, you will see more segments defined as having (ID 1), in our case being classed as touching the ground.



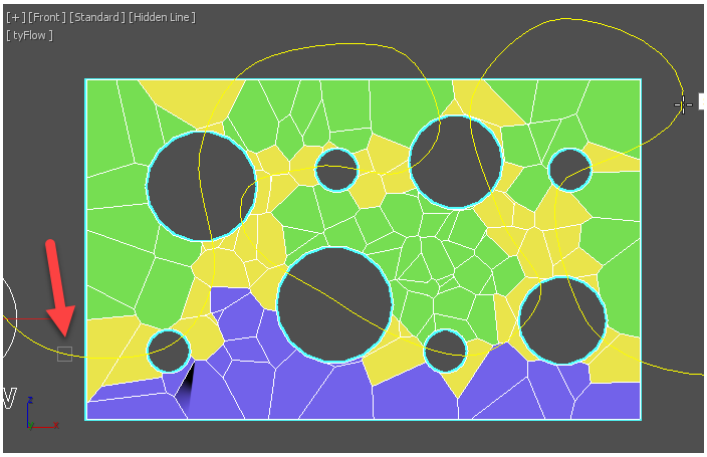
Bind is the glue that holds the segments together, by applying two binds, we are able to differentiate between touching the ground or not, and that was the purpose of this example file. With the bind search we can find those floating elements and do something with them.

In this final screen capture, we take the breaker bar all the way to the end of the spline path it is following and we get the result on the right

Yellow – touched by the bar sent to **Event_002**

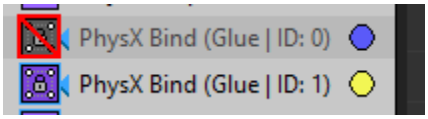
Green – not touched by the bar, and now defined as not touching ground, sent to **Event_003**

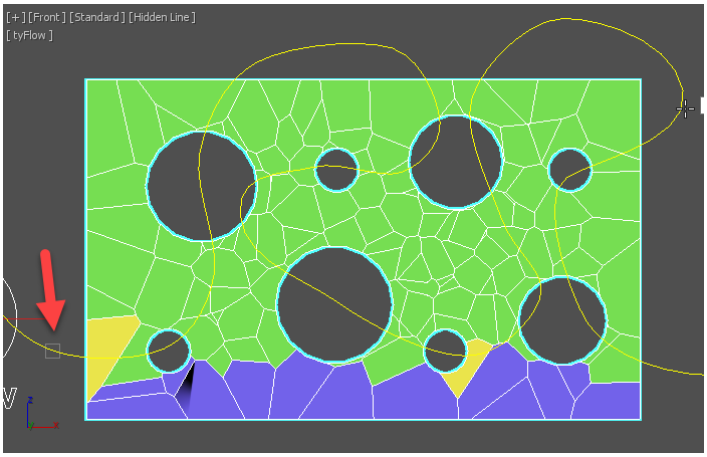
Blue – because of PhysX Bind Operator, defined as touching ground.



If we didn't have both bind operators, there would be hardly any yellows, so the 1st bind operator with ID set to 0 is critical for the surface detection to work and the second bind operator with ID set to 1 is critical to find floating wall segments. Both bind operators work in a relationship so we can have two separate test paths.

For completeness, a screen grab on the right, shows our wall with PhysX Bind (ID 0) disabled. Notice there are only two yellow segments.





Adding PhysX Switch to our Flow

NOTE: (PER MANUAL)
The PhysX Switch operator gives you control over how PhysX rigidbodies will be treated by the PhysX solver.

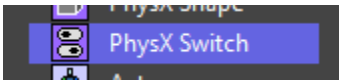
INFO:
Dynamic rigidbodies are directly affected by the PhysX solver. Collisions will cause them to move/rotate in response.

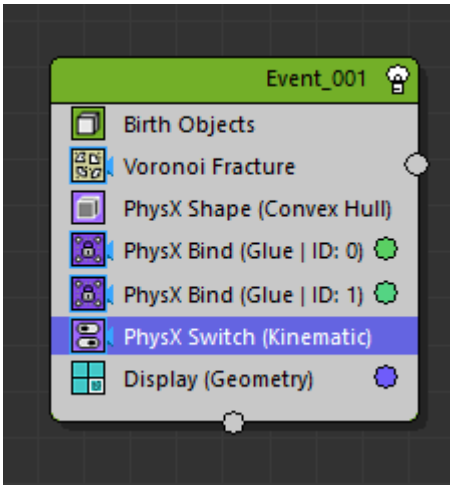
Kinematic rigidbodies are not directly affected by the PhysX solver. Dynamic rigidbodies will collide with them, but they themselves will not move in response.

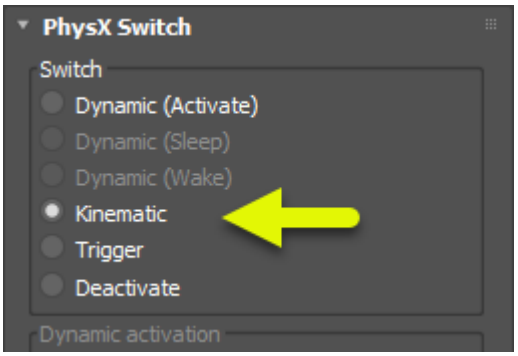
Trigger rigidbodies are not directly affected by the PhysX solver. Dynamic rigidbodies will register collision points with triggers rigidbodies, but dynamic rigidbodies with a matching trigger simulation group will not be affected by those collisions.

Deactivated rigidbodies are not processed by the PhysX solver at all.

Drag and drop the PhysX Switch operator below our two binding operators. Change the switch type to be Kinematic:

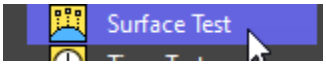




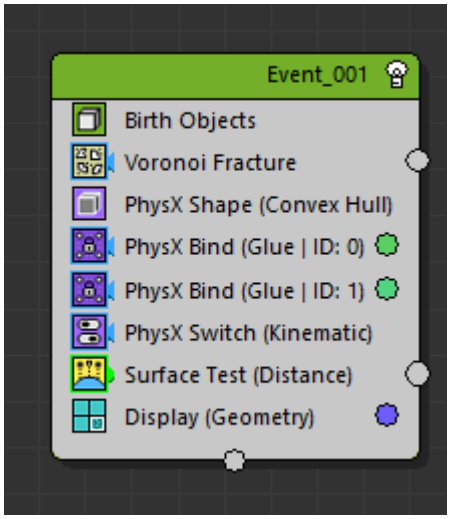


By setting **Event_001** to kinematic, we are telling the wall not to be affected by the PhysX solver. The wall will stay intact until we tell it otherwise with another PhysX Switch operator later in our flow.

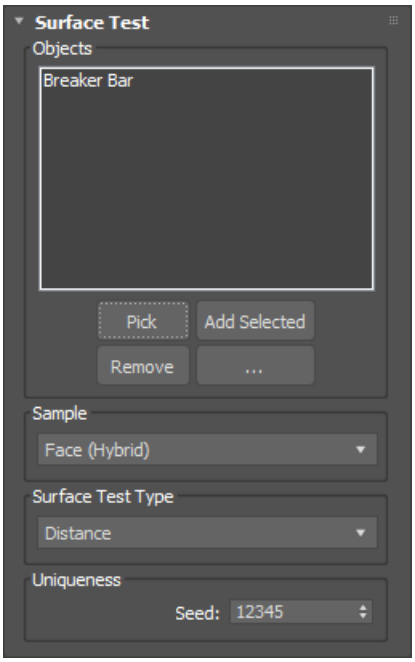
Adding a Surface Test to our Flow

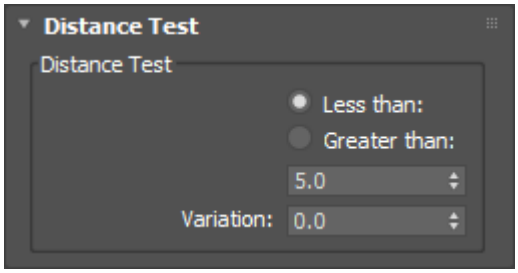


Left – our particle system at this point:



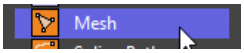
Below; pick our breaker bar box to be the surface test object

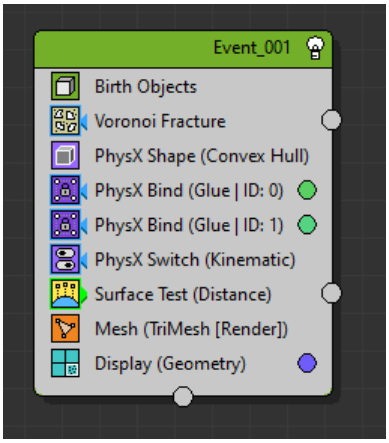




Set the distance test to be less than 5

Add a Mesh Render to our Flow





Our MAIN flow at this point is now complete.

The mesh operator allows the particles to be rendered.

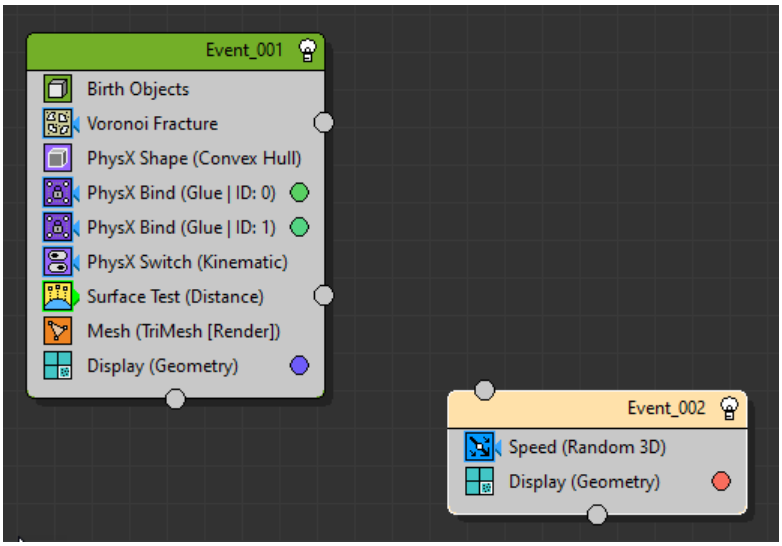
Note on Event_001

I have not added the Property Test operator at this time of the flow design, which is the only operator missing from **Event_001**. The reason I didn't, is because I am following the flow in its logical sequence, so I will add it when we make **Event_003**.

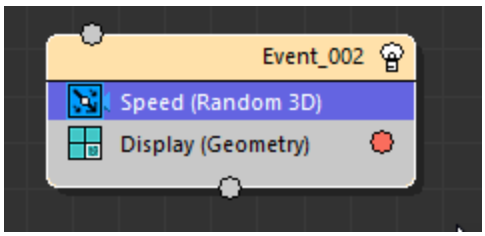
EVENT 002 – SURFACE TEST FLOW

Adding Speed test to EVENT 002

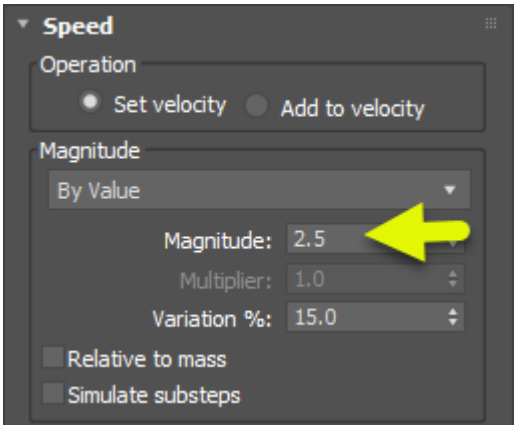
NOTE: (PER MANUAL)
The Speed operator allows you to assign velocities to particles. Velocity vectors are constructed by multiplying a direction vector by a magnitude.



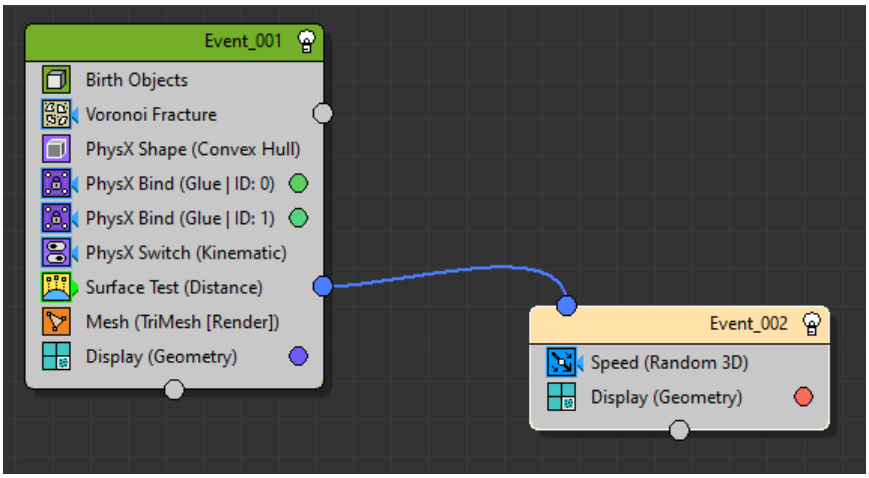
Click and drage the Speed operator to an empty space in the flow window, creating a new event as shown below:



Note: My **Event_002** color defaulted to red, I will change this to be YELLOW for **Event_002**



In the speed rollout for the operator, change the magnitude to 2.5

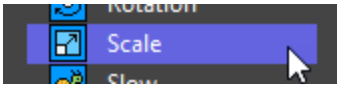


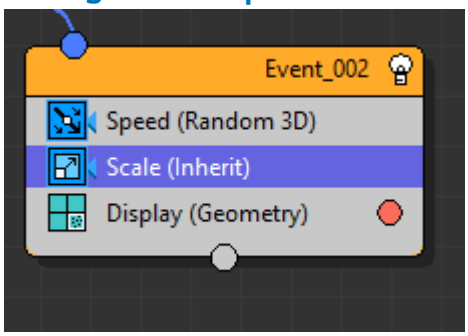
Connect the two events as shown

What we have just told the main flow is:

- Test the surface by distance of our breaker bar to our wall
- If the test condition is met, branch of to event 002
- Do things in event 002 as per that flow

Adding a scale operator to EVENT 002





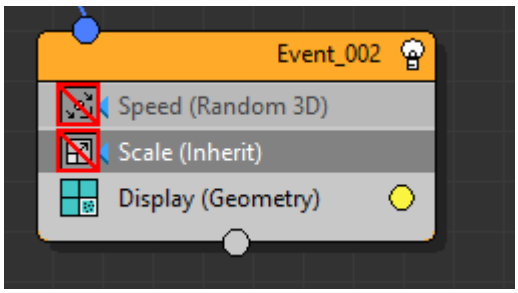
Event 002 – should look like the left image at this point:

Change the scale value of X/Y/Z to 95% in the rollout

Essentially we have made all the yellow segments just a tiny bit smaller opening gaps between each element.

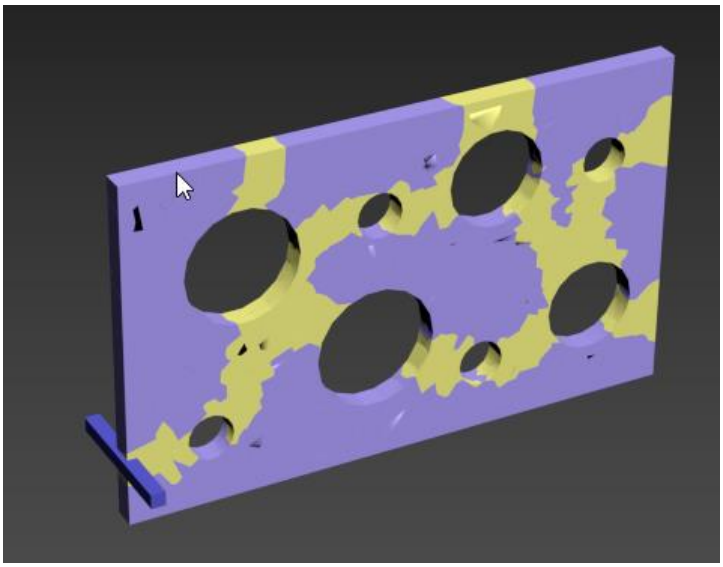
Lets try to understand what we have created so far:

Lets turn OFF the SPEED and SCALE operators in this event and run our animation:



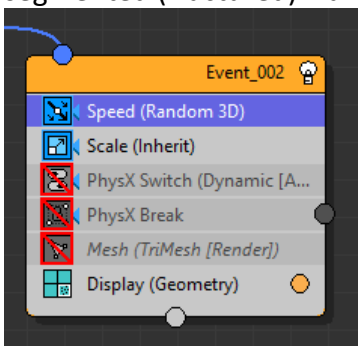
Notice, my display color is yellow for this EVENT 002 output.

The result of doing turning off those two operators at frame 250.

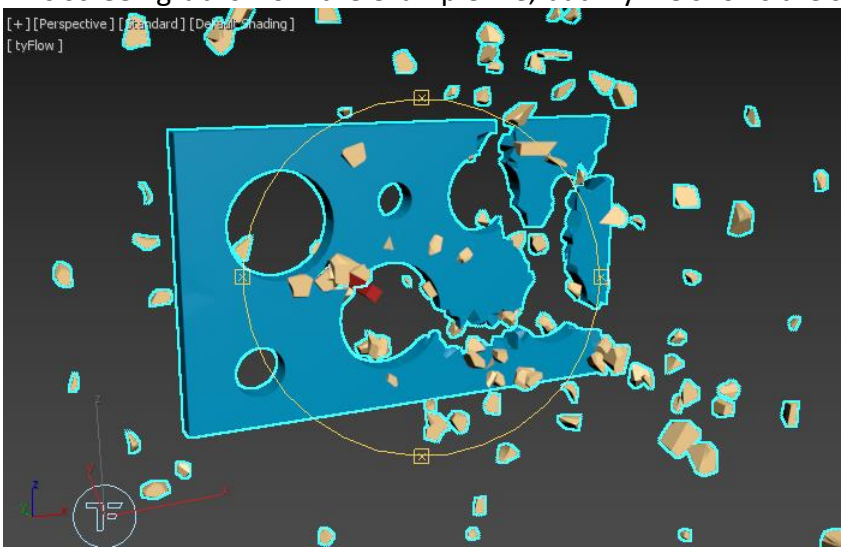


Notice that all the segments that are touched by the breaker bar, have turned yellow, indicating they were affected by our test condition and they are now being affected by EVENT 002

If we look at frame 100, with this branch still not finished, as far as adding operators, we will see particle mayhem, wherever the breaker bar touched the Voronoi segmented (fractured) wall



This screen grab is from the example file, but my file shows the same result:

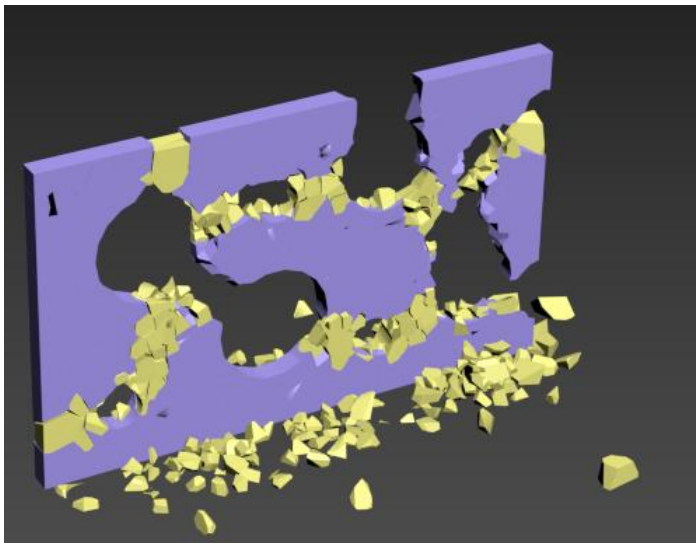


The way I interpret the particle system at this point is:

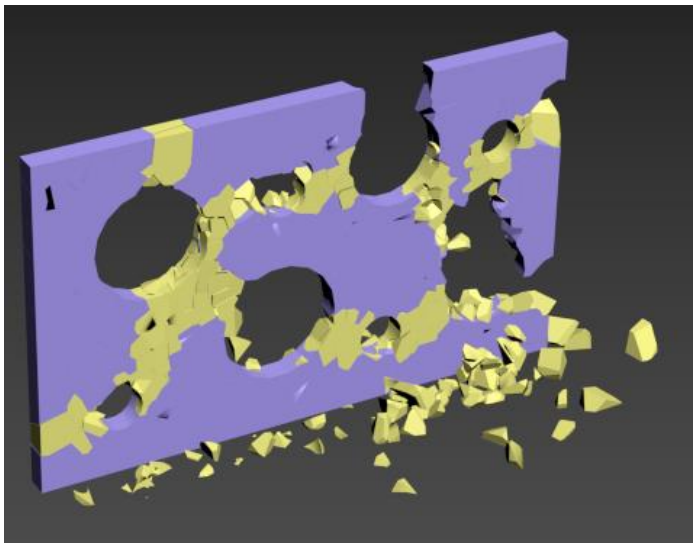
- EVENT 001 is all about setting up the wall for fracture, holding the fragments together until we get to our test conditions.
- The way I interpret the two test conditions is:
    - EVENT 002 is all about being hit by the breaker bar, and whatever we put in EVENT 002 is to control that part of the animation.
    - EVENT 003 has no initial affect on the animation, until things have happened in EVENT 002, and it seems the purpose of EVENT 003 is to handle the rest of the wall fracture after EVENT 002 has had its impact on the wall – ie: once parts of the wall are gone or impacted from the breaker bar, now do this to the remaining wall in EVENT 003

Lets compare frame 250 render output with the Scale Operator enabled versus disabled, so we can understand what the scaling of the wall segments did for us

ENABLED RESULT FRAME 250



DISABLED RESULT FRAME 250

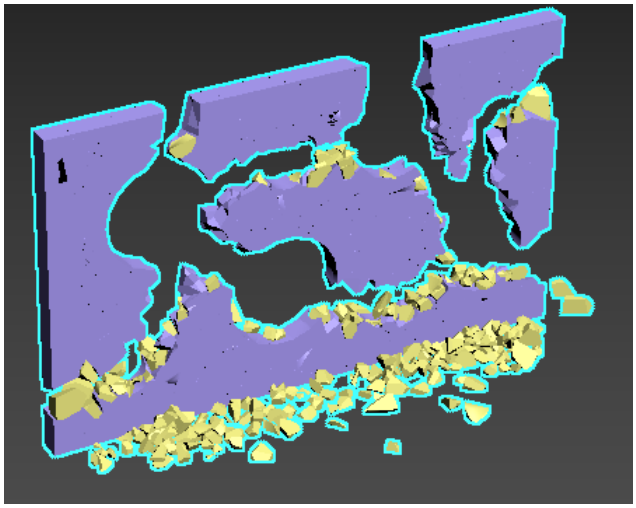


Comparing these two screen grabs, we can see that by decreasing the size by 5% to 95% of original size we got more movement and more parts of the wall fell to the ground creating more open gaps and instability in the wall above.

To finalize my understanding of the use of the scale operator in this animation, I set the scale to 75% on all three axis and rendered the result. I got what I expected would happen, by decreasing the size of the segments event smaller, as they are hit by the breaker bar, now virtually all the pieces fall to the ground. This helps establish the notion, that the purpose of scaling the segments, is to help the wall segments break away and help the wall collapse.

Because of the speed of the animation, you don't actually see this scaling, this is also, because the scaling doesn't actually happen until the breaker bar hits those segments of the wall.

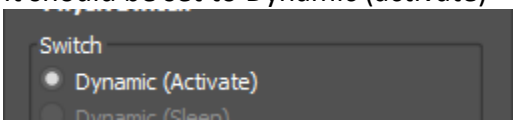
Frame 250 render output with scale set to 75%



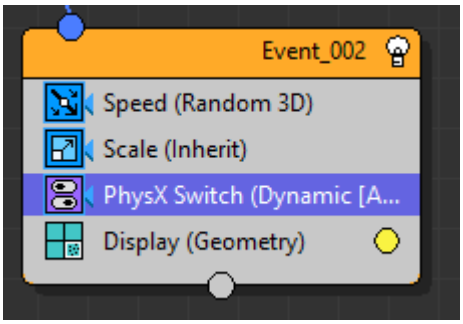
Adding PhysX Switch to EVENT 002

Click and drag a PhysX Switch to EVENT 002 below the Scale Operator.

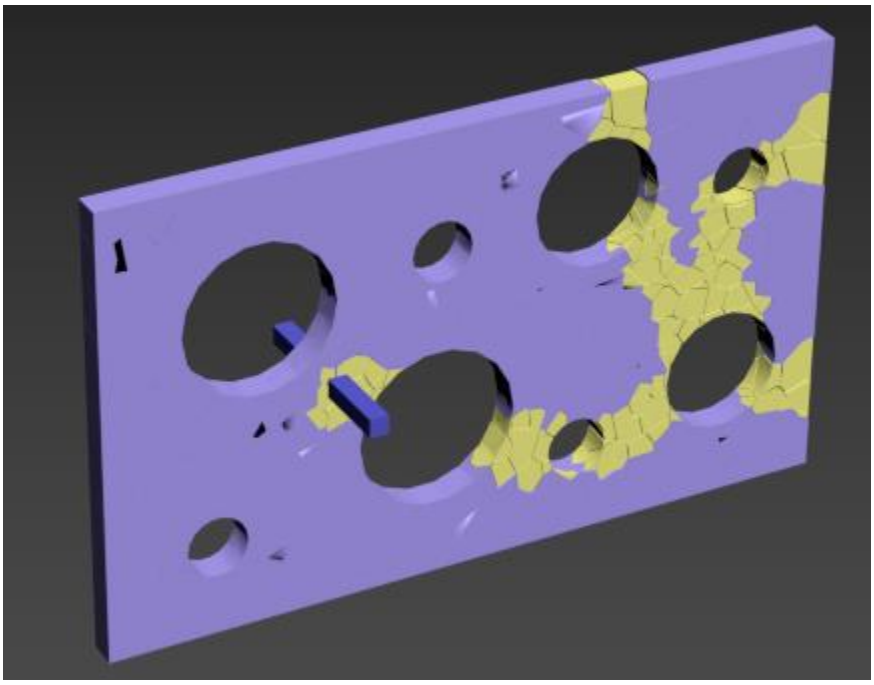
It should be set to Dynamic (activate)



Our EVENT 002 Flow should look like this:

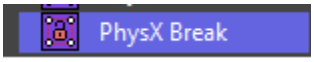


If we look at frame 100 now after adding the PhysX Switch, we will see a rather different result, than the mayhem before it was added.

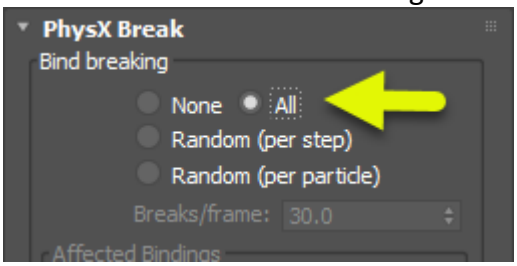


Adding PhysX Break to EVENT 002

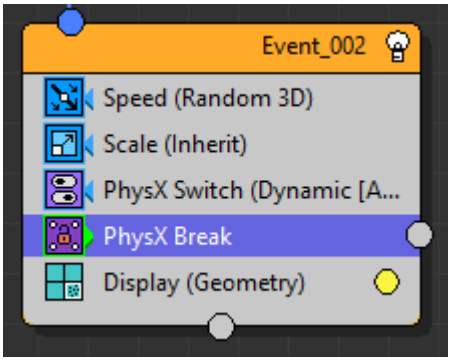
Click and drag a PhysX Break to EVENT 002 below the PhysX Switch Operator.



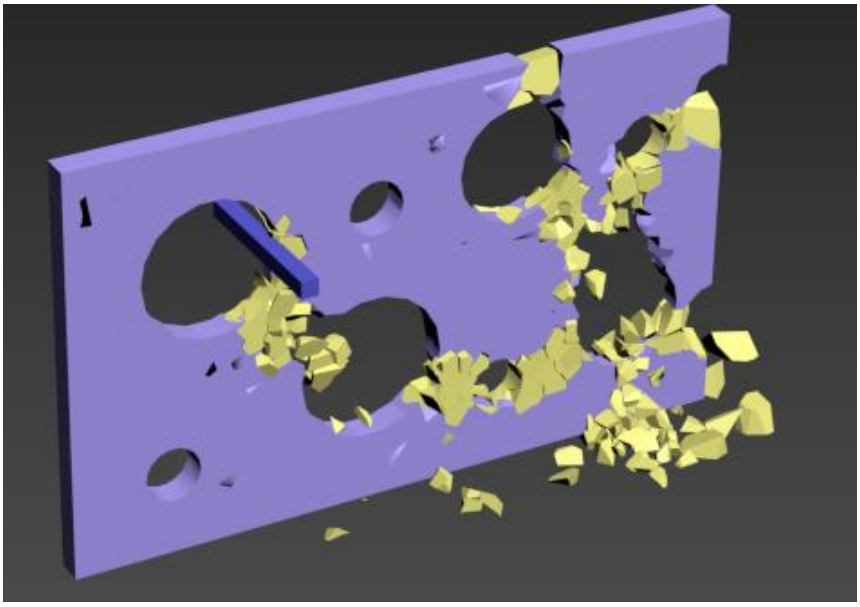
It should be set to Bind Breaking - ALL



And EVENT 002 particle flow should look like below:



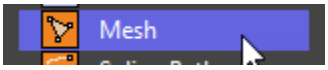
If we take another look at frame 100 after adding the PhysX Break operator to EVENT 002 we see the following:



So we can see that EVENT 002 purpose is to break out the segments of the wall hit by the breaker bar and we can see that the remaining wall remains in place, that is the part of the wall supposed to be handled by EVENT 003.

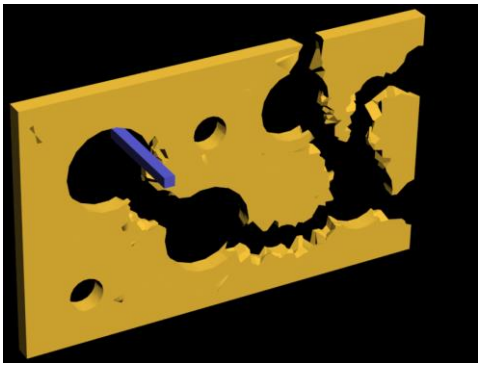
EVENT 002 handles the yellow segments while EVENT 003 handles the blue segments of the wall, EVENT 003 requires EVENT 002 to have any impact on the animation.

Add a Mesh Render to EVENT 002

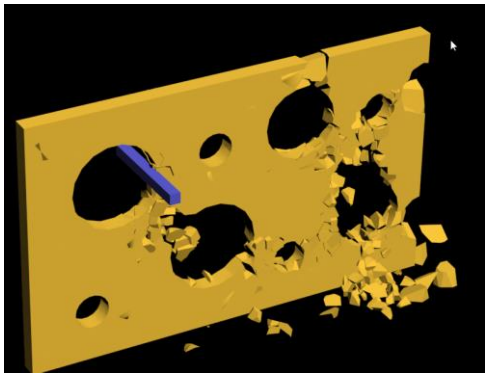


The mesh render operator allows us to render the particle system, without it, we would not see anything in the render output.

Without Mesh Render



With Mesh Render



EVENT 003 – BINDING SEARCH FLOW

At this point, we have a wall that has some segments hit by the breaker bar and those segments have fallen to the ground, what remains are the wall segments that were not hit by the breaker bar and most of those have stayed intact, some are even suspended in mid-air.

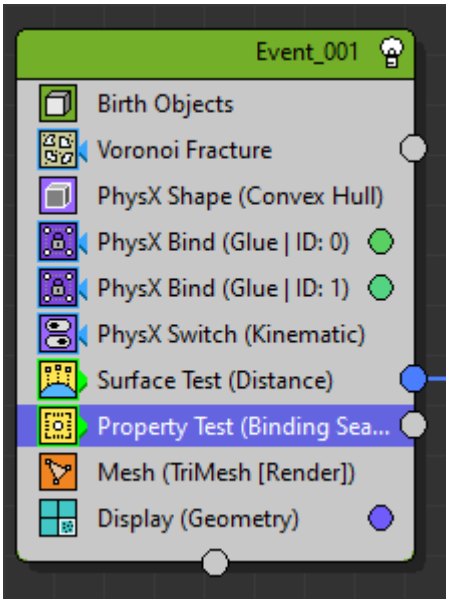
If I understand Tysons example file properly, we are going to test for those suspended sections of the wall with the Property Test Operator – set to test property binding

Adding Property Test Operator

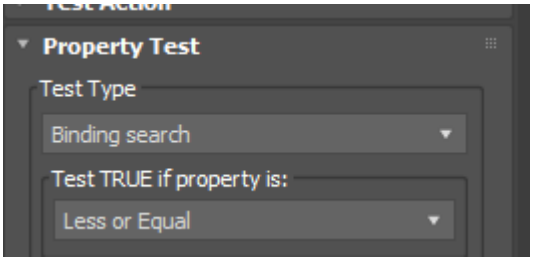
NOTE: The Property Test operator is added to Event_001 – it is the test that sends us to Event_003

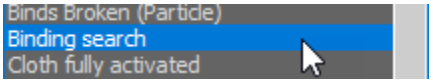
Click and drag the Property Test operator to our main event **Event_001**

The main event should now look like below image:

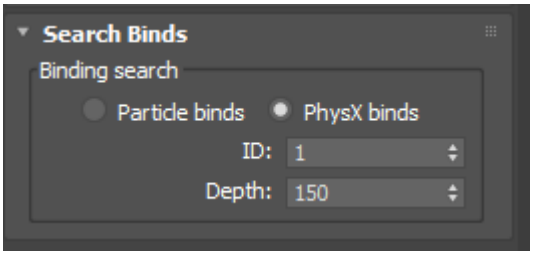


In the rollout for the Property Test Operator, set Test Type to Binding Search

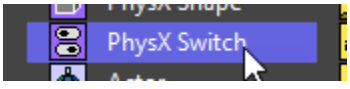




Also change the search binds rollout to PhysX Binds with an ID of 1

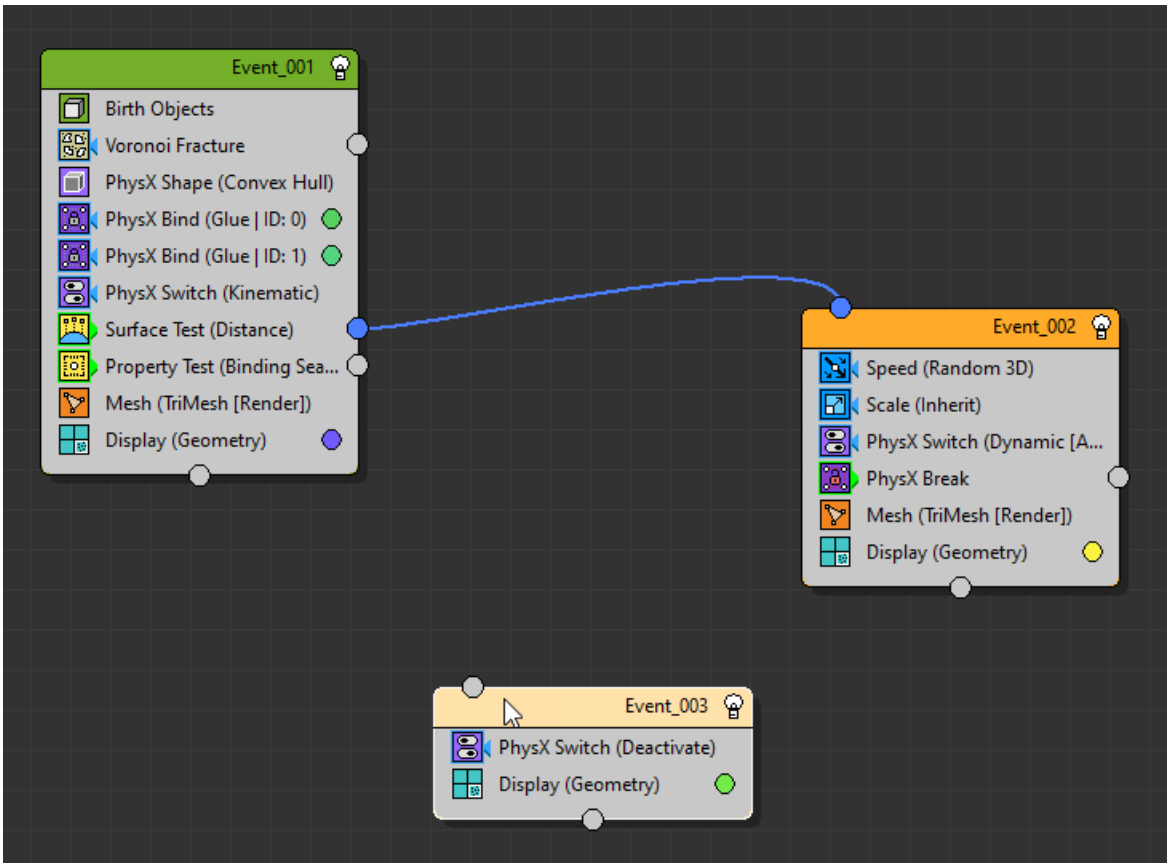


Adding PhysX Switch Operator



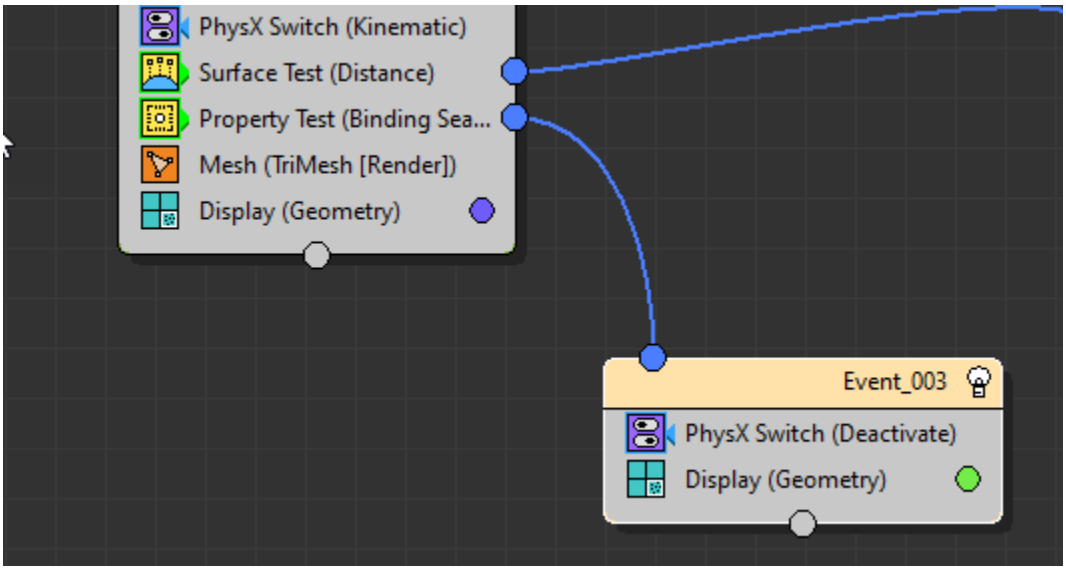
Drag a PhysX Switch operator to an empty section on the window creating a new **Event_003** flow

Our particle system should now look like this:

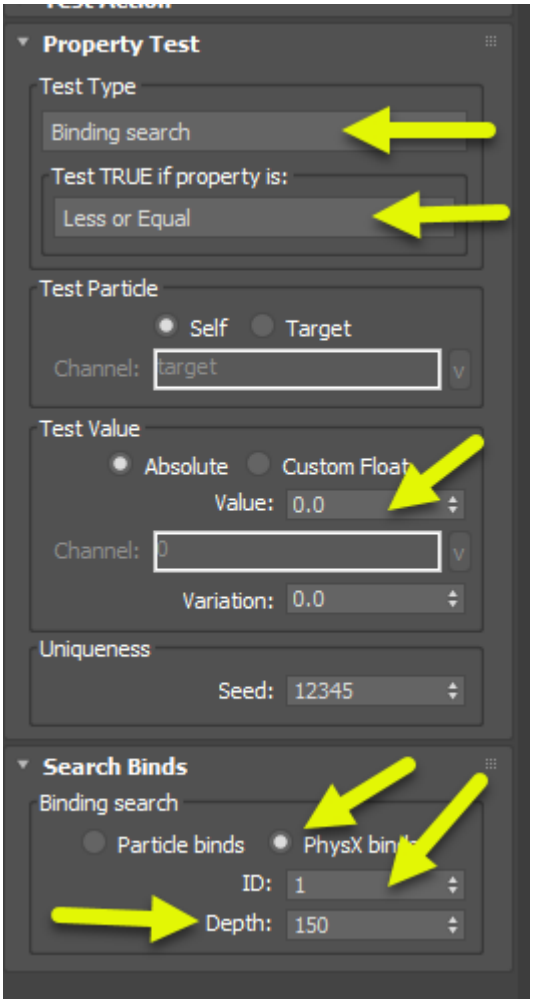


Note: My **Event_003** color defaulted to green, I will keep this color. If yours is not green, then change it to be GREEN for **Event_003**

Let's connect our Property Test output to **Event_003**



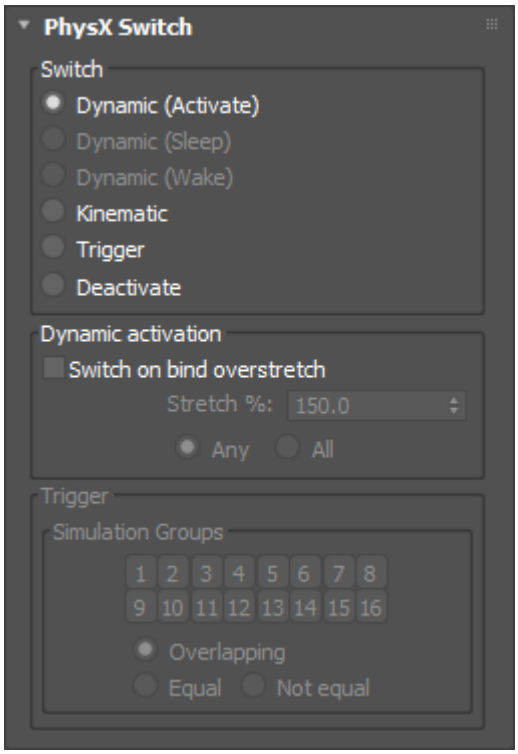
Setting Property Test Parameters

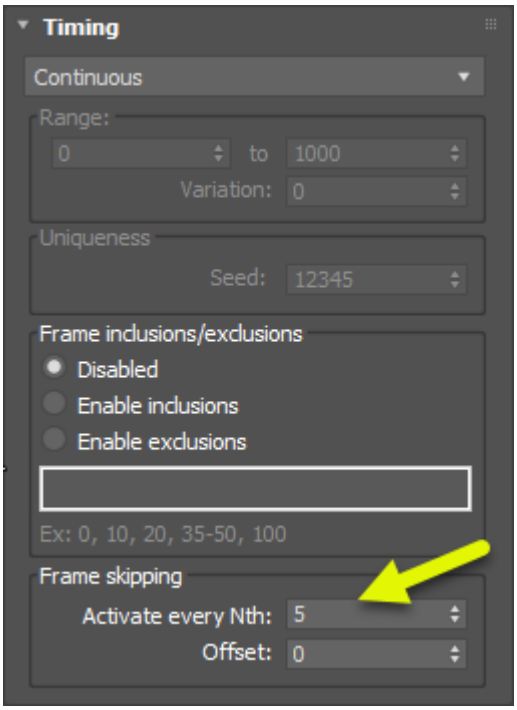


We need to check/set the parameters for the Property Test in **Event_001** to the following settings shown on the left.

Then we go to **Event_003** and change the setting for the PhysX Switch operator to Dynamic (activate), as shown below

NOTE (PER MANUAL):
Dynamic (Activate): particle PhysX shapes will be treated as dynamic rigidbodies. Affected rigidbodies will be activated.



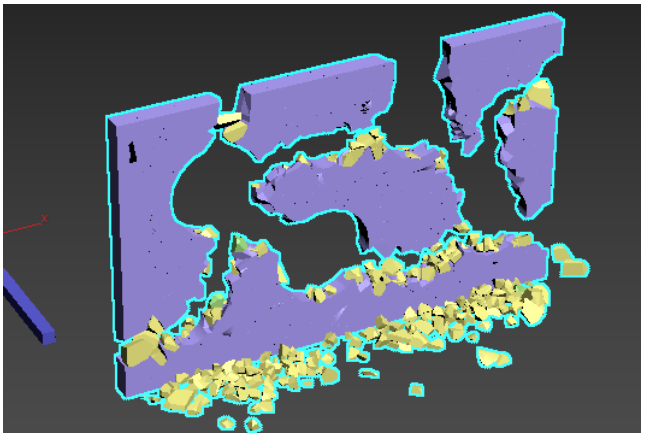


It was also noted in Tysons readme, that we could improve performance by skipping frames for the Property Test operator. He stated that it didn't need to happen every frame and it looks like he has set it to test every 5 frames.

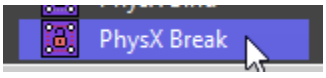
Go to the Timing Rollout in the Property Test operator for **Event_001** and change this timing value, as shown on the left.

I Interpret this to mean, we have tested for hanging wall (suspended) wall elements, so we are going to activate those elements and handle them with the next steps in our **Event_003** flow.

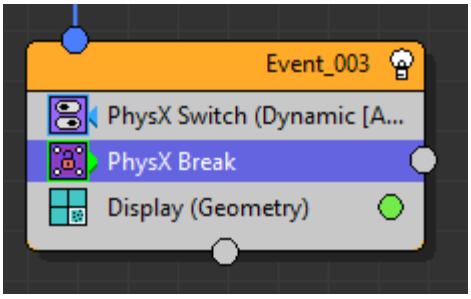
Looking at frame 250, we see the wall is still standing and some segments suspended in midair.



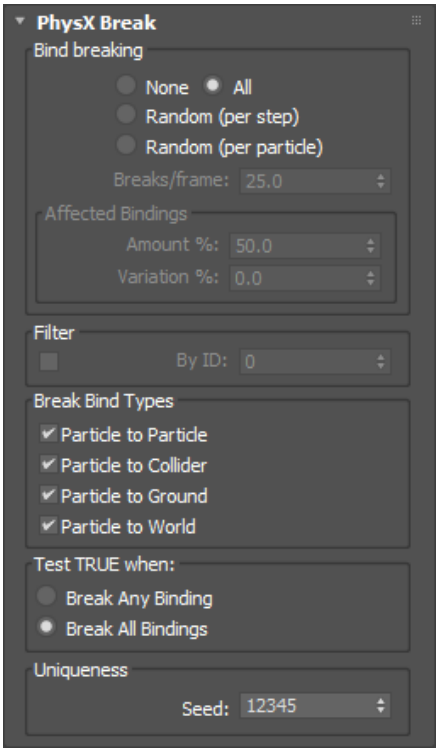
We are nearly there; the end is in sight. Let's add a PhysX Break Operator to **Event_003** after our PhysX Switch operator



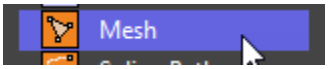
We should be looking like this for **Event_003**



Change the parameter in the rollout, as shown below

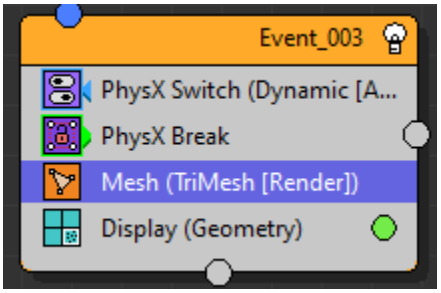


Adding Mesh Operator



The final operator to be added is the mesh operator which allows the particles to be rendered, it needs to be at the bottom of every flow you want to be rendered.

Event_003 should look like the screen grab below:



That's it!! – Its time to replay the whole animation and render it, if you want.

Have fun, hope this was helpful