

Houdini 16 Fluids

[Introduction](#)

[Create Fluids](#)

[Create](#)

[Emit](#)

[Fill/Sculpt](#)

[Sandbox \(FLIP Tank\)](#)

[Destroy Fluids](#)

[Bounding Box Collision](#)

[Particle TTL \(time-to-live\)](#)

[Sink](#)

[Fluid Bounding Box](#)

[Bounding Box Size](#)

[Bounding Box Collision Behaviour](#)

[Fluid Collisions with RBDs](#)

[Feedback Scale, Density, and Collision/Particle Separation](#)

[Apply Artificial Force to RBD Object](#)

[Embed In Fluid Tool](#)

[Fluid Forces](#)

[Particle Forces](#)

[Volume Forces](#)

[Fluid Accuracy](#)

[Particles](#)

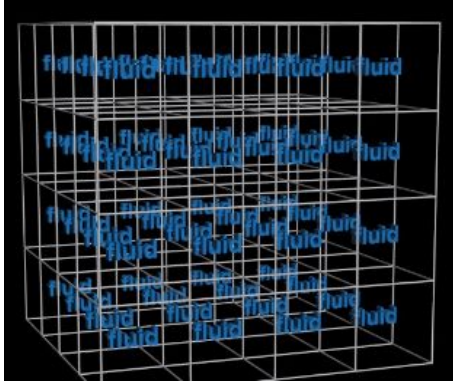
[Collision Geometry](#)

[Fluid Viscosity](#)

Introduction

Houdini's fluid model is called FLIP (FLuid Implicit Particle). It's a hybrid of 2 different toolsets. In the old days of Houdini, artists were given 2 ways of creating fluids: voxels are particles.

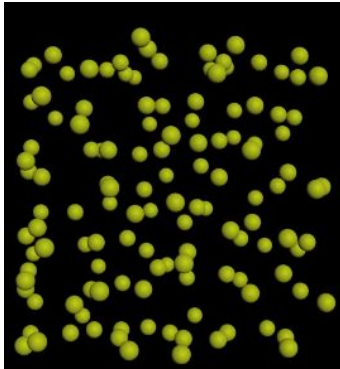
Voxels (also known as volume grid / volume box) were essentially a 3D stack of boxes where each box would contain information about the fluid: direction, speed, pressure, surface shape, etc...



Although this produced good/predictable results for a very long time, it had some problems...

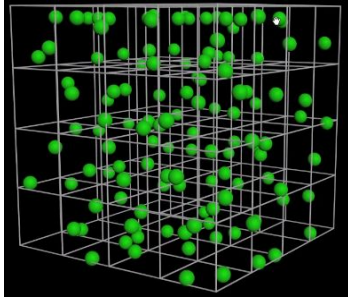
- if your fluid had to go somewhere outside of the box, it couldn't -- it would just disappear.
- if your fluid's voxel resolution was too low, certain parts could disappear at random.
- it was inefficient in that you would have to know the largest possible volume your fluid would consume before hand -- even if you were using a small piece of that volume in the majority of your frames, calculations for the entire volume would have to be done every frame.

Particles were essentially a cloud of points that would make up a fluid.



Unlike voxels, they weren't confined to a certain volume (they could go anywhere) and they were fast to calculate. But, they were also very unpredictable in that they could explode or otherwise act erratically for no apparent reason + they required the extra step of having to generate a surface.

FLIP is a hybrid of both the particle and voxel systems. FLIP essentially calculates each particle's movement inside a voxel, and the overall voxel shape is defined by where the particles are defined in space.



So essentially FLIP is just the volume system and the particle system talking back-and-forth to each other. One is dependent on the other.

NOTE: The only particle thing to note here is that you can apply both volume forces and particle forces to your fluids. This is described in the forces section.

Create Fluids

The various ways to create fluids are documented below. Keep in mind that when you do create a fluid, a few things end up happening...

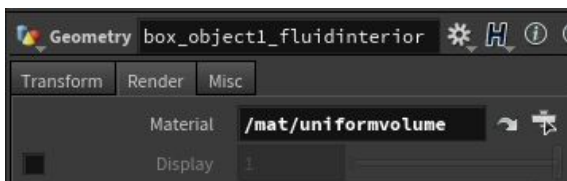
- AutoDopNetwork will get created (if it doesn't exist already).



- *_fluid geometry will get created -- show you your fluid particles.

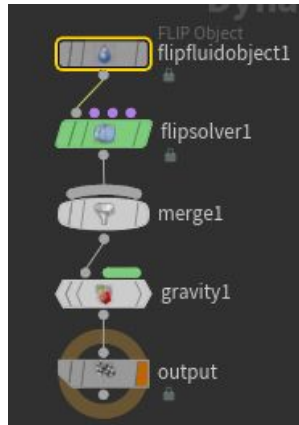


- *_fluidinterior geometry will get created -- used for rendering the final fluid material.



- Any original geometry you used as part of the creation may have stuff added to it to make it work with the fluid stuff in the AutoDopNetwork.

The inside of your AutoDopNetwork will have a portion added to it for FLIP fluids...

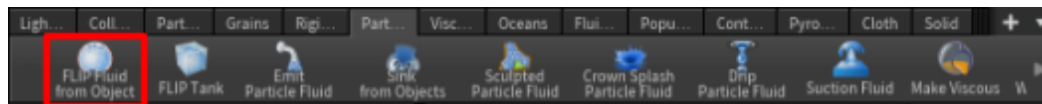


The important thing with this is that FLIP fluids will be confined to a certain bounding box. If they go past the extent of the bounding box they will either disappear or collide (depends on your settings).

See the Fluid Bounding Box section for more information.

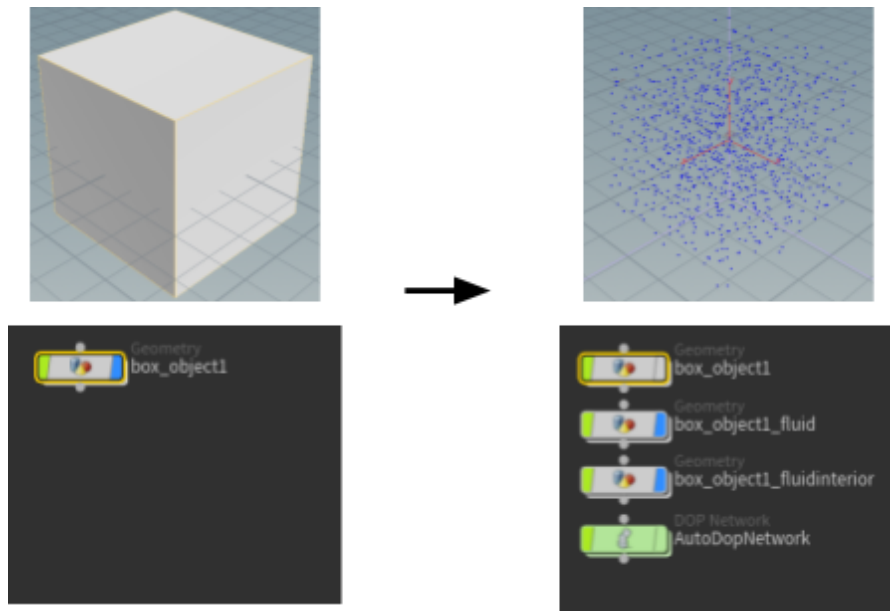
Create

You can convert pretty much any shape into a finite amount of fluid via FLIP Fluid from Object in the Particle Fluids shelf...



If you don't already have an object selected when you do this, it'll ask you to select one and press Enter (this is done inside the scene view). Once it does the conversion, your object will

get hidden and in its place will be the particles for your fluid.

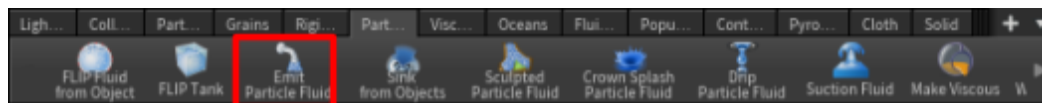


Notice that the original geometry (box_object1) gets hidden once the conversion is complete. If you wanted to expand or reshape or otherwise manipulate the shape, you need to do it on this hidden original geometry.

NOTE: Unsure what fluid and fluidinterior are for? Check the parent section.

Emit

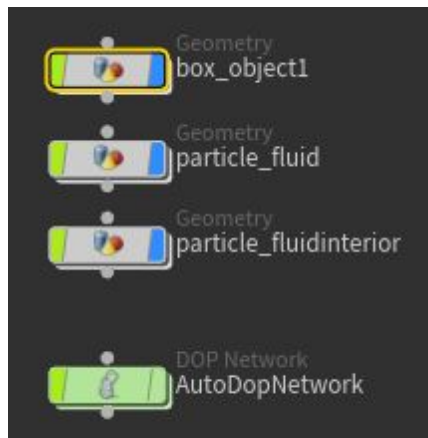
Just like how you could emit particles from some shape, you can also emit fluid from some shape. You can do this using the Emit Particle Fluid item in the Particle Fluids shelf...



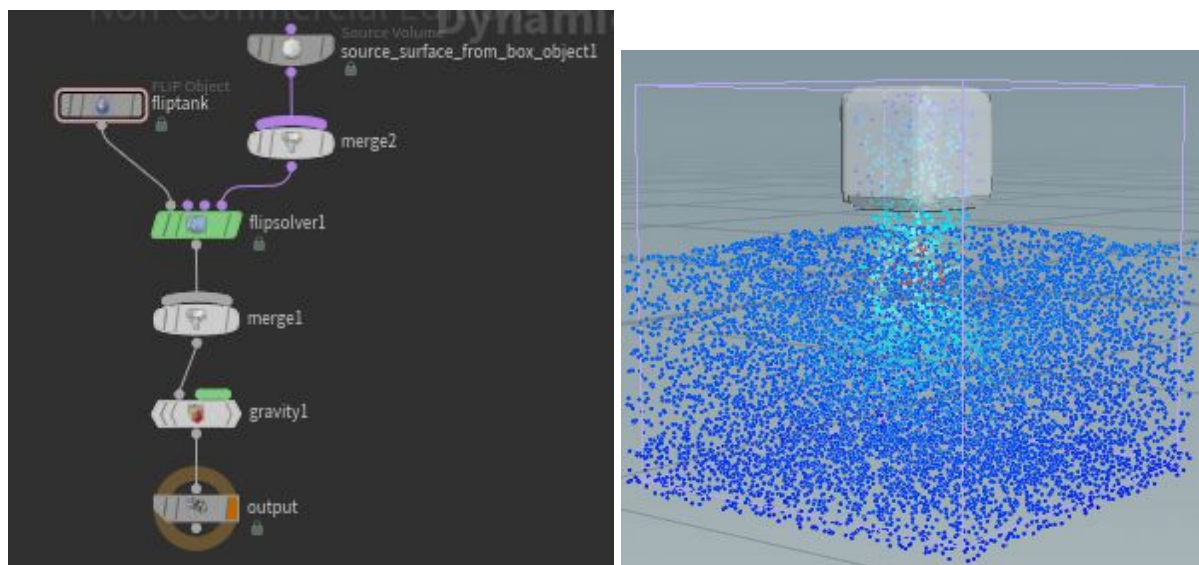
If you don't already have an object selected when you do this, it'll ask you to select one and press Enter (this is done inside the scene view).

It'll then ask you to select the fluid object you want to emit into (if any). For this, go into your AutoDopNetwork and select a FLIP object that feeds into the solver that you want your

generated particles to go into, then go back into the scene view and hit Enter.

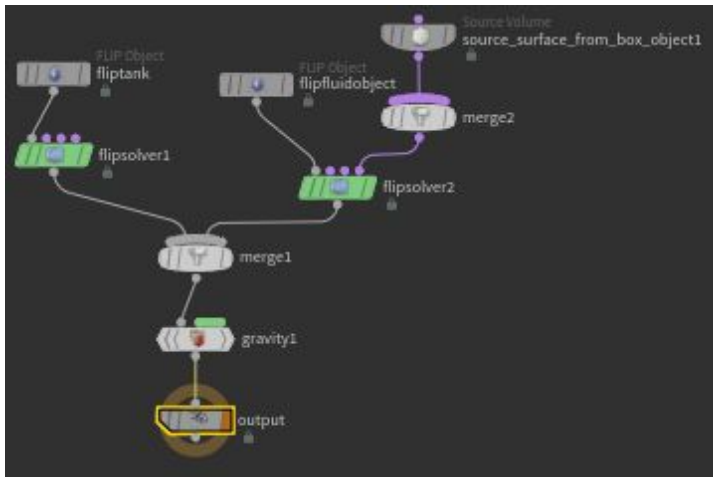


NOTE: Unsure what fluid and fluidinterior are for? Check the parent section.



NOTE: If you don't want it to have a destination, don't select anything and hit Enter. The AutoDopNetwork will create a brand new solver and merge it with any existing flipsolvers. When this happens, the water particles from the different flipsolvers won't

end up interacting with each other -- they'll just go through as if it was thin air.

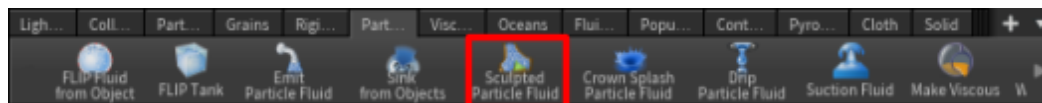


If you want to produce more water particles at once, you'll have to scale up the object you're emitting from.

NOTE: This particular way of creating water lets you set a TTL on the particles. See the Fluid Destruction section for more information.

Fill/Sculpt

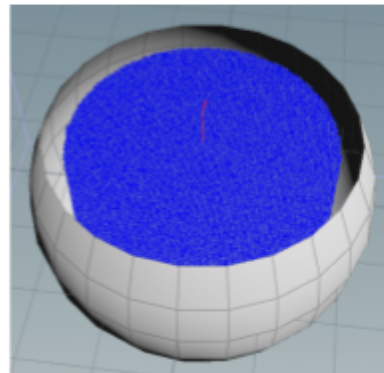
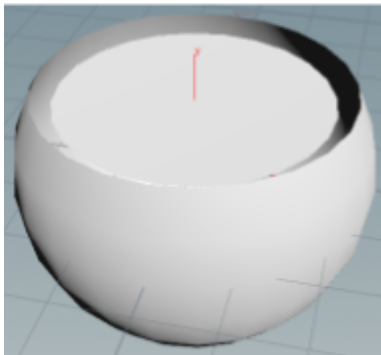
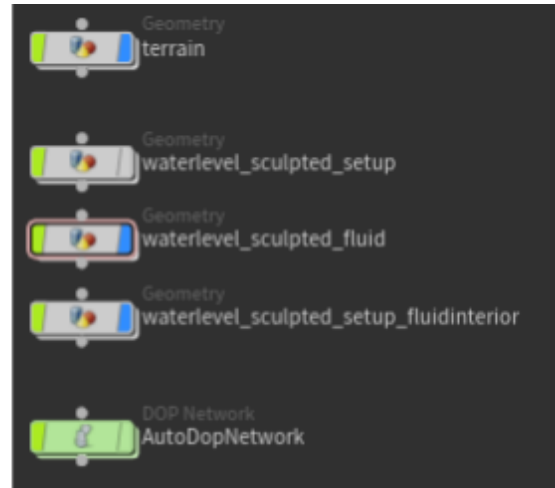
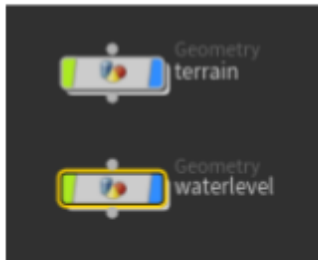
You can make it so that you have a specific volume that gets filled with water in the very beginning (without having to spend frames filling it up with an emitter or anything like that). This is done via Sculpted Particle Fluid in the Particle Fluids shelf...



Make sure you have nothing selected when you click this shelf item. When you do click it, you'll prompted in the scene view to make 3 selections (pressing Enter after each selection)

- object that represents how high the water should be
- object that represents the terrain
- object(s) that represents obstructions the water can hit

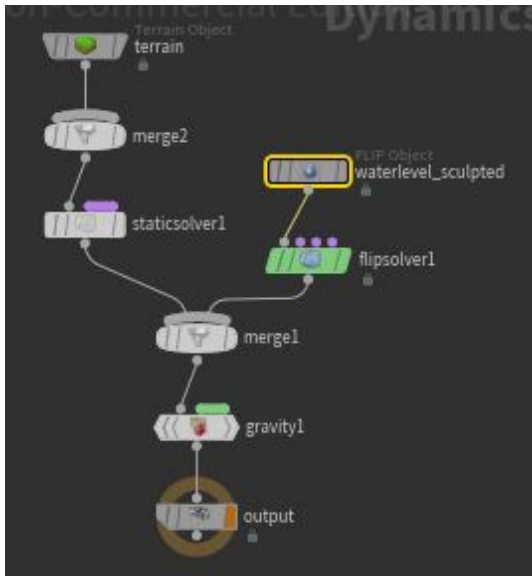
NOTE: Be careful with what you select as a terrain object. It determines what direction it should try filling in by the normals on that object. So if you notice water filling on the outside of your terrain instead of the inside, you need to reverse the normals. You can do this by shoving a Reverse node into your geometry...



NOTE: Unsure what fluid and fluidinterior are for? Check the parent section.

A few things happen once you do this...

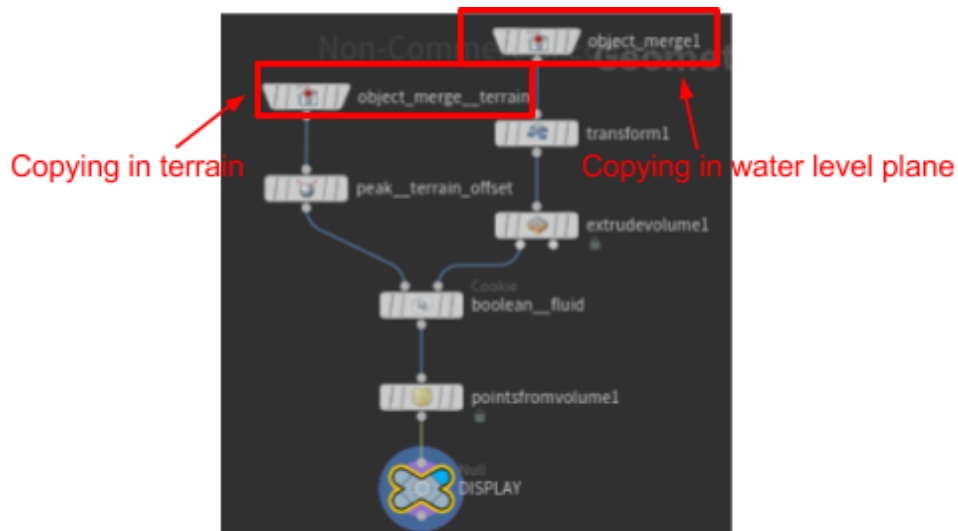
1. your terrain will get a Terrain collision object associated with it (as if you went under the Collisions shelf and chose Terrain).



2. your obstructions get static objects associated with them (as if you went under Collisions shelf and chose Static).

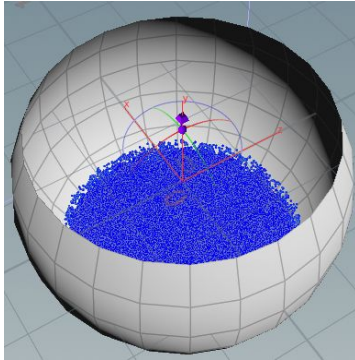
(no picture here because I didn't choose any, but this would be getting fed into the staticsolver along with the terrain)

3. your water level geometry node gets renamed with a “_sculpted_setup” suffix and stuff gets added to it such that it generates particles in the terrain shape up the water level you specified.



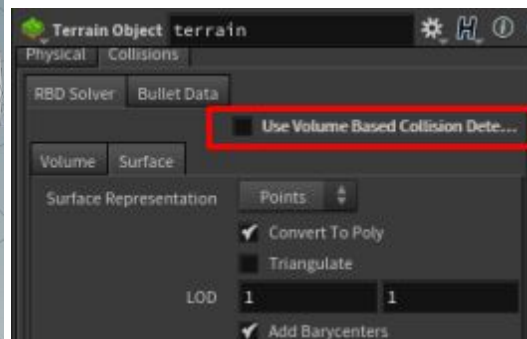
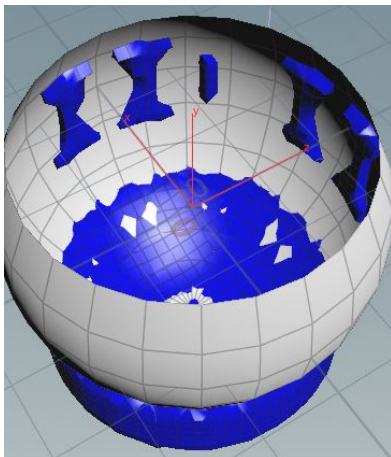
There are lots of problems here...

The end result of the example above was that the water would not stay in the basin. It seems that the particles would seep and gradually get stamped down (or disappear?) as the simulation progressed.



NOTE It almost seems as if the water is going down to some invisible basin underneath the sphere. See the picture of the volume based collision geometry below.

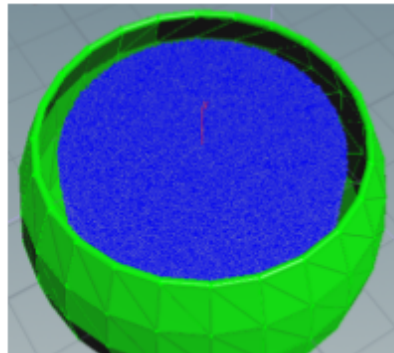
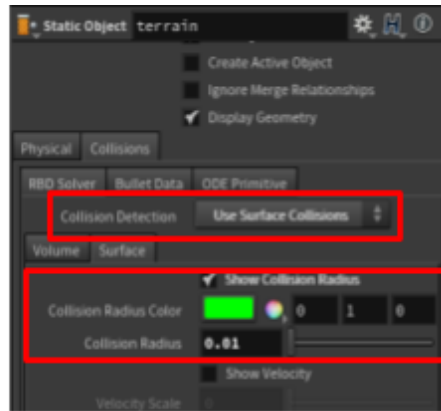
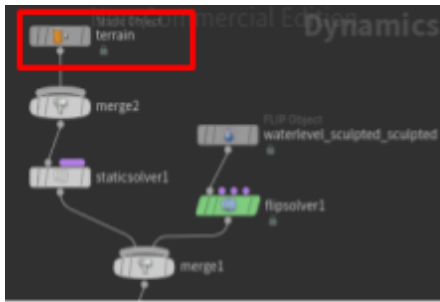
The collision geometry generated on the terrain in the example above is extremely messed up. It uses volume based collision by default and does a terrible job of creating the collision geometry. Switching to surface based collision seems to help only sometimes. The problem is that you don't have the option to set Collision Radius on a Terrain object like you do with a Static Object, which means in certain cases seeping of water between the seams still happens.



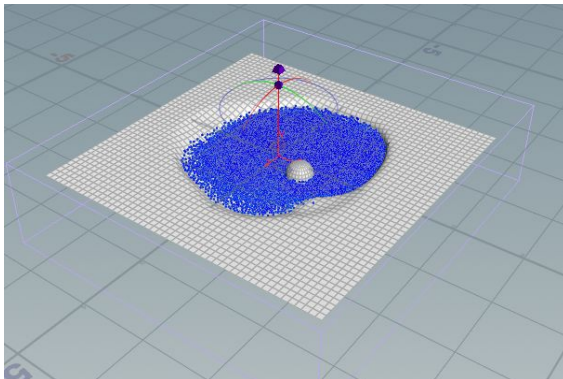
NOTE: Adding thickness to the terrain geometry (polyextrude node) did nothing.

NOTE: One potential workaround I found to the collision issue was to manually remove the terrain object from the AutoDopNetwork (including whatever was added to the Geometry node to get it in there) and place it back in as a Static object. That way you'd get access to the extra options for Surface collisions. BUT, the problem here is that the fluid particles just disappear as gravity pulls them down and they collide -- it's as if they've hit the bottom of the fluid bounding box. I don't know how to fix this but there's

probably a way.



NOTE: Okay, so after continued experimentation, it looks like that it may be this is expecting terrain-like geometry. That is, if I can a normal grid and sculpt some grooves into it, then use another grid for the water-level, everything works perfectly fine. Surface collision mode on the terrain object also seems to work well. I don't know why my original cut-off circle geometry didn't work.



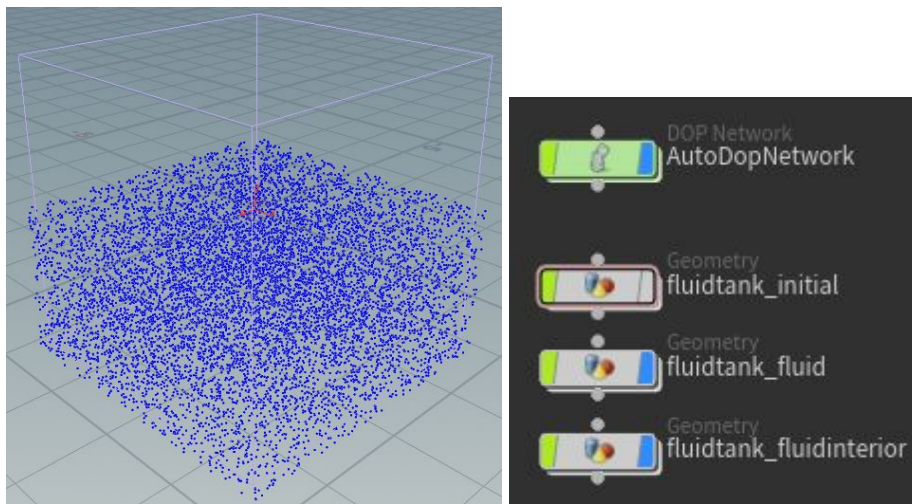
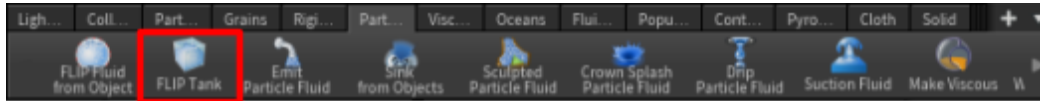
For best results you should give your terrain/static objects some depth (make sure they aren't thin like paper)

Sandbox (FLIP Tank)

A FLIP tank is almost the same thing as creating a box and converting it to a fluid, except that...

- the bounding box for the solver will be linked to the bounds of the box object (this is called fluidtank_initial)
- the fluid will fill roughly 50% of the bounding box
- the fluid will collide with the bounding box instead of go through it

FLIP tanks are often used as a sandbox to play with water. You can do create a FLIP tank via the FLIP Tank item in the Particle Fluids shelf...



Destroy Fluids

There are a few ways to destroy fluid particles.

Bounding Box Collision

You can control what happens to your fluid particles when they hit the walls of your fluid solver's bounding box. To see how to do this, check out the Fluid Bounding Box section.

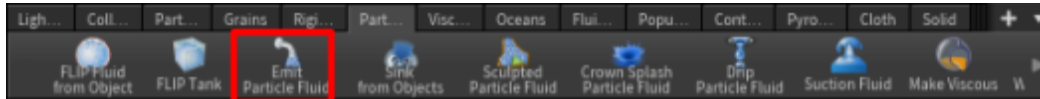
The default for whether your fluid disappears or not depends on the exact way your fluid was created.

For example, if you created your fluid from the FLIP Fluid Object item in the Particle Fluid shelf, the fluid particles disappear when they hit the bounding box walls. But, if you created your fluid using the FLIP Tank item under the same shelf, they'll collide instead of disappear.

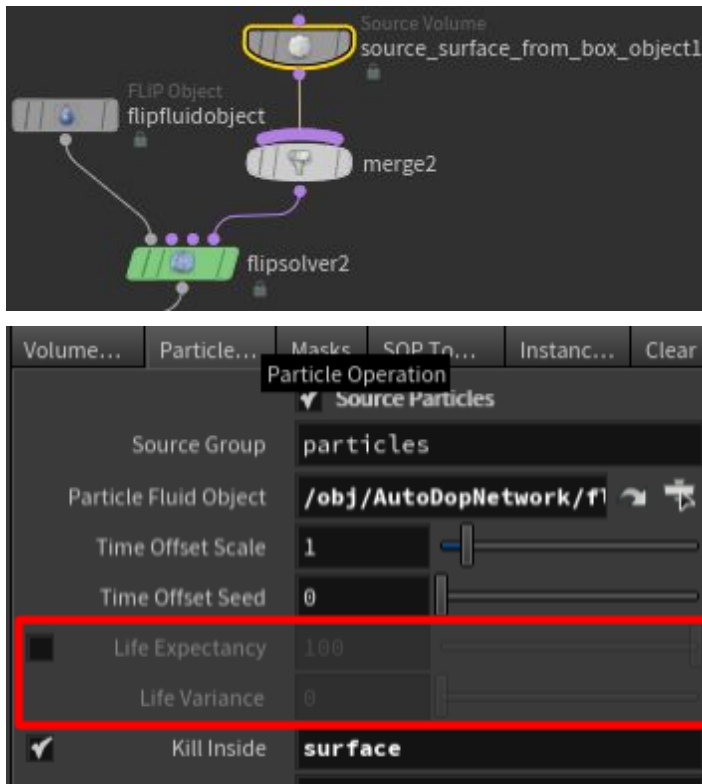
NOTE: It should be obvious why this is. FLIP Tank is meant to be a sandbox playground. What good is a water testing playground if there's no water.

Particle TTL (time-to-live)

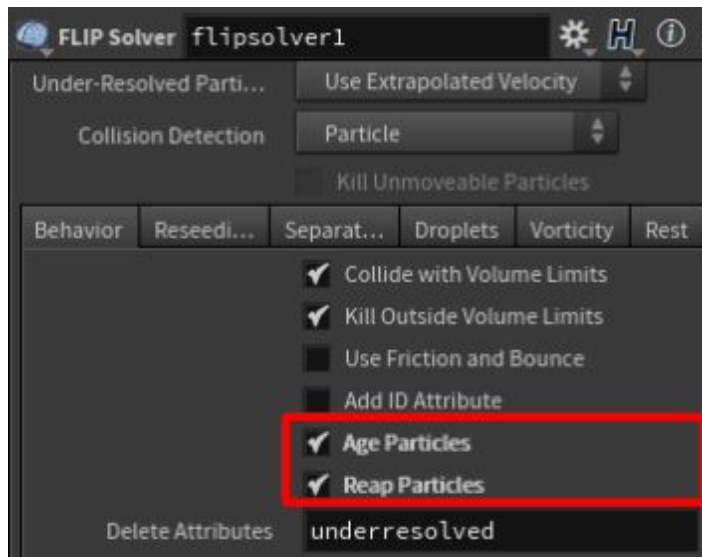
You can set TTLs on fluid particles generated from an emitter (if you chose the Emit Particle Fluid item in the Particle Fluids shelf)...



If you want to kill water particles after a certain amount of time (control lifespan), you can do so by going to the Source Volume node in your AutoDopNetwork and selecting the Particle Operation tab. Enable Life Expectancy and set the value to whatever you need to (value is in seconds).



Once you set this, you also need to go to the solver and specifically tell it to age particles and reap particles.



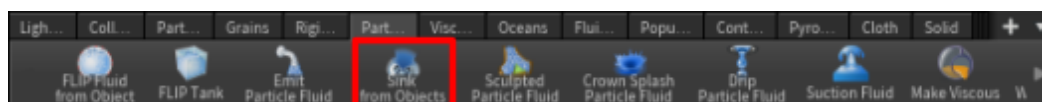
NOTE: If you set the solver to enable age+reap particles, technically you can add an age parameter to particles generated from anything? If you look at the geometry spreadsheet, you can see what extra attributes were added to each particle to get them to age...

		P[z]	age	dead	life
5980	79	0.438584	0.479167	0	0.5
5979	1	0.359167	0.479167	0	0.5
5978	29	0.473689	0.479167	0	0.5
5977	74	0.365334	0.479167	0	0.5
5976	52	0.329777	0.479167	0	0.5
5975	03	0.139516	0.479167	0	0.5

The life attribute seems to be the big one here. It tells the solver how long to keep the particle around. There doesn't seem to be a node available in the dynamics context to add an attribute to these particles, so it must be done via VEX or python? I don't know enough about Houdini to do that yet, but I think it's definitely doable.

Sink

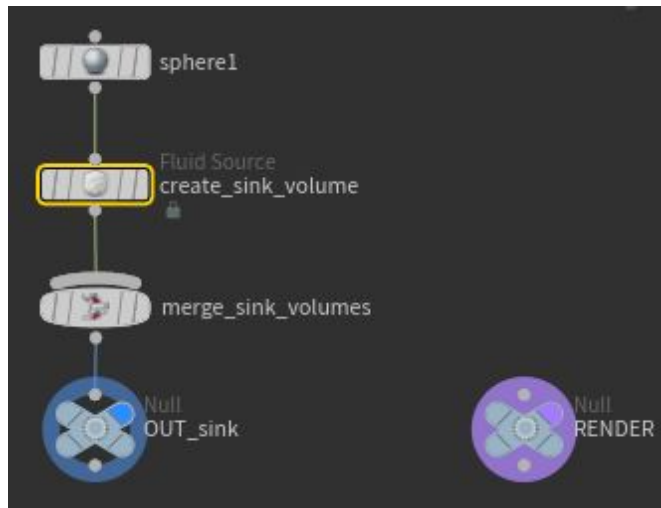
Just like how you could emit fluid particles from some shape, you can turn some shape into a sink for fluid particles. You can do this using the Sink from Objects item in the Particle Fluids shelf...



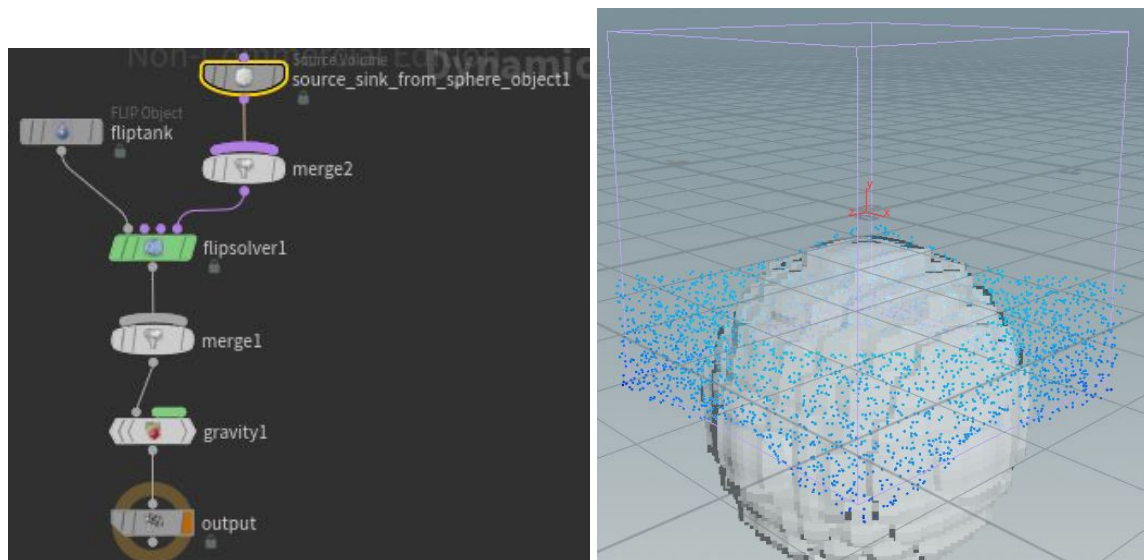
If you don't already have an object selected when you do this, it'll ask you to select one and press Enter (this is done inside the scene view).

It'll then ask you what “smoke object” you want your sink added to. This is probably a bug. What it's asking for is the fluid solver you want your sink added to. For this, go into your AutoDopNetwork and select a FLIP object that feeds into the solver that you want your fluid particles disappear from, then go back into the scene view and hit Enter.

You shouldn't get any new objects under your /obj context. But, the object you selected to be your sink will have some stuff added to it...



In addition to that, in your AutoDopNetwork, you'll find a new “Source Volume” node added to the flipsolver you chose. If you run your simulation, you'll notice that the water particles will start draining into the object.



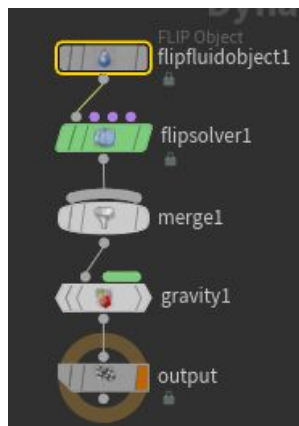
NOTE: If you don't see the particles draining, it may be that your sink object isn't big enough. Jack up the size.

NOTE: Notice that, just like for an emitter, a sink adds a Source Volume node. The difference (I guess) is how the node is set up. It's probably set to kill particles that enter it vs generate particles from it.

If you want to consume more water particles at once, you'll have to scale up your sink object.

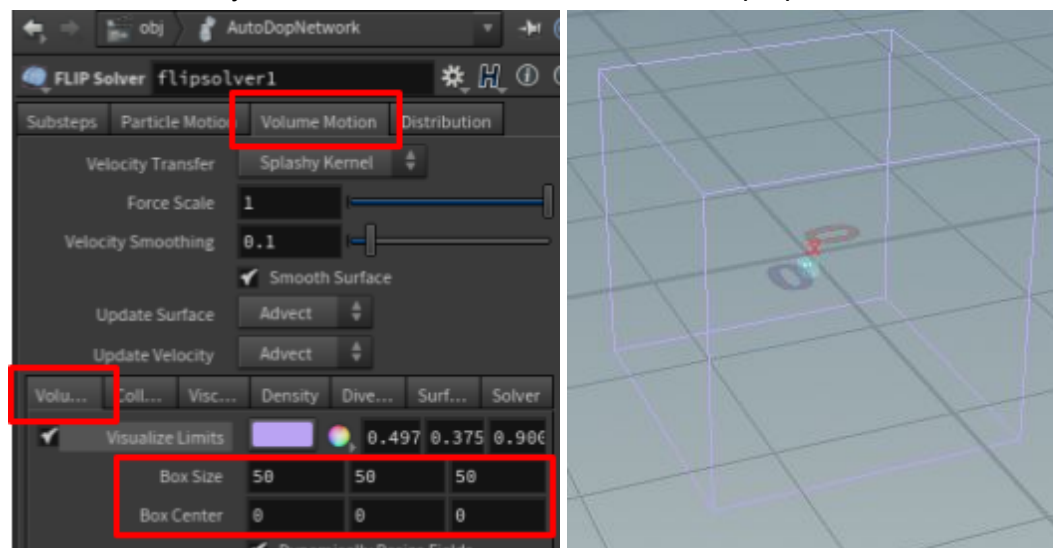
Fluid Bounding Box

The important thing with this is that FLIP fluids will be confined to a certain bounding box. If they go past the extent of the bounding box they will either disappear or collide (depends on your settings).

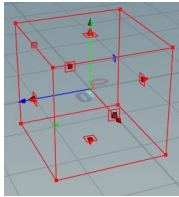


Bounding Box Size

To control the size of the bounding volume box, go to the flipsolver node. Under Volume Motion -> Volume Limits you'll find the Box Size and Box Center properties...

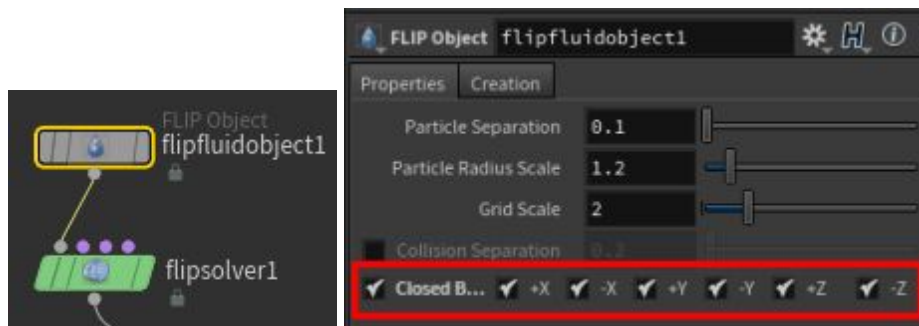


NOTE: You can also use the handle tool and change this directly in the scene view...

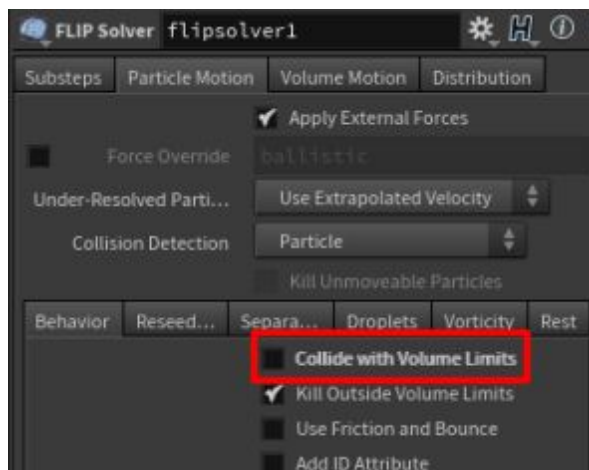


Bounding Box Collision Behaviour

To control whether particles collide or disappear when they hit the limits of the bounding box, you need to go to whatever FLIP object it is that's being fed into the solver. Under the Properties tab, you'll see an option called Closed Boundaries that'll let you choose which parts of the box you want to treat as closed (collides with) or open (disappears when hit).



If you choose to have them collide with the boundary, you can set it so that the force of the water doesn't get reflected back when it hits that boundary by unchecking the Collide with Volume Limits checkbox in the flipsolver (under Particle Motion -> Behaviour)...

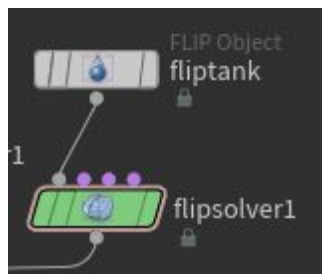


Fluid Collisions with RBDs

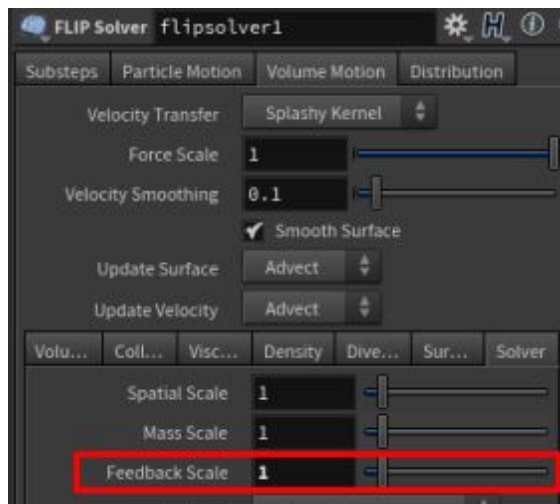
There doesn't seem to be a foolproof way of having FLIP fluids properly interact with collision objects. Ultimately that means that if you're trying to simulate an object floating on or inside the water, you're going to have a tough time. The following sections are an overview of the possible techniques to deal with this.

Feedback Scale, Density, and Collision/Particle Separation

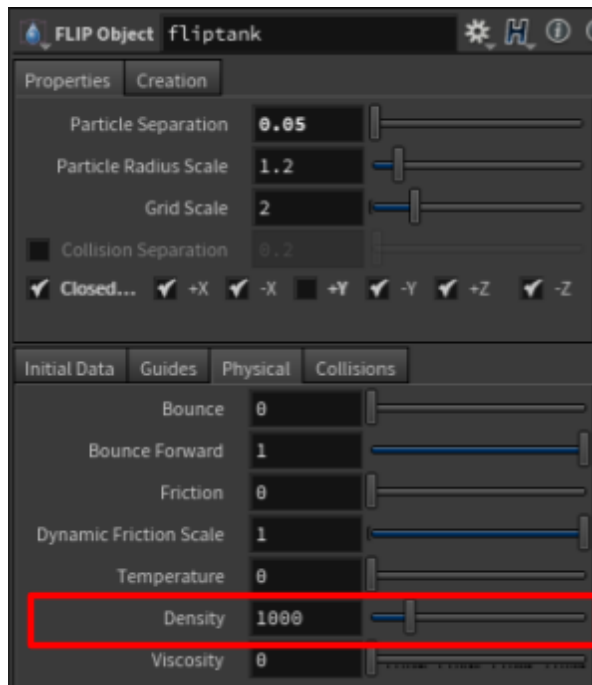
You can adjust how much 'feedback' the water gives to collision objects based on the Feedback Scale value in the flipsolver...



In your solver, you can find the Feedback Scale slider under Volume Motion -> Solver. The Feedback Scale goes from 0 to 1, where 0 is 0% and 1 is 100%. I noticed that somewhere up high (around 0.9 or 1.0) is a good value.



Once you do this, you'll need to potentially tweak the density of your water and the RBD objects you throw in them. You can tweak the density of your water by going to Physical tab of your flipobject. This density value will interact with the density of your RBD object (and the feedback scale to determine) to determine if your object should float or sink or whatever.

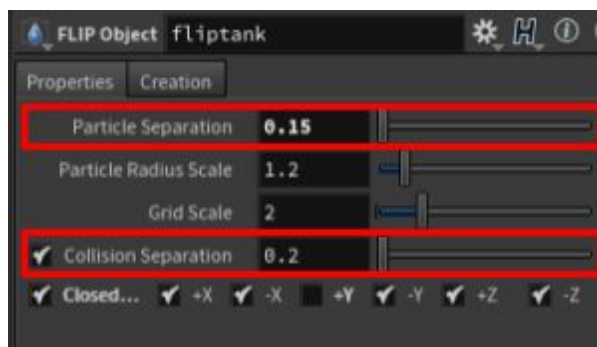


NOTE: You can find the density of your RBD object in the Properties tab of your RBD object. Remember that changing the density means that it'll your mass is being modified, which means that you'll potentially mess with other physics-related stuff.

Once you do this, you may notice highly erratic behaviour in your simulation with the way the RBD objects react to fluids. There are a couple of things you can tweak under the Properties tab of the flipobject to help with this.

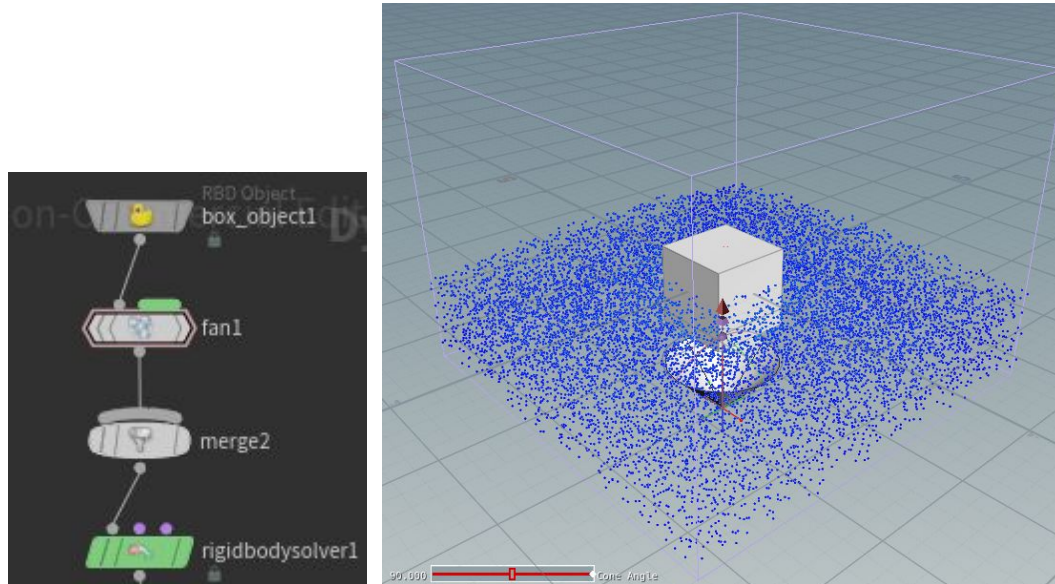
The first is the Particle Separation value. A low Particle Separation value of your flipobject means more particles, which means a more accurate simulation (but also one that's more computationally expensive).

The second is the Collision Separation value. I don't know exactly what this does, but enabling it seems to get rid of all the erratic behaviour without having to lower the Particle Separation value.



Apply Artificial Force to RBD Object

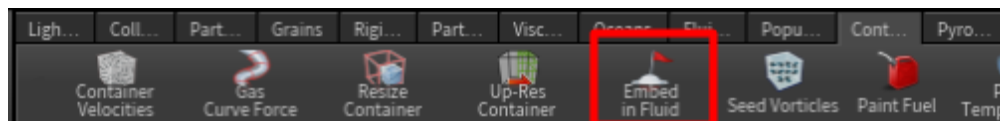
One option is to having your object interact with water would be to apply a force to the object to make it seem like the water is resisting it. For example, you can stick a Fan Force node after your RBD object to simulate buoyancy in the water. The water will still move around properly as the box goes up-and-down inside it, making it look realistic.



NOTE: The fan is beneath the box. It's being partially obstructed by the water particles. Remember the force that's being applied to the RBD does not have an effect on the water.

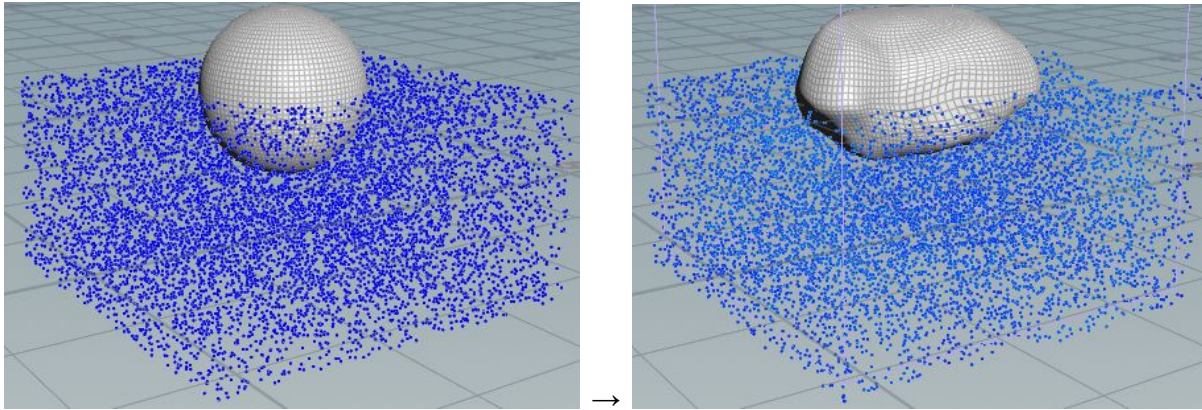
Embed In Fluid Tool

One way to get your objects to interact with water would be to use the Embed in Fluid item under the Container Tools shelf.



This is almost always NOT what you want. What it does is that it takes the POINTS of the object and embeds it alongside the FLIP fluid particles, causing your object to deform over time rather

than interact with the water...



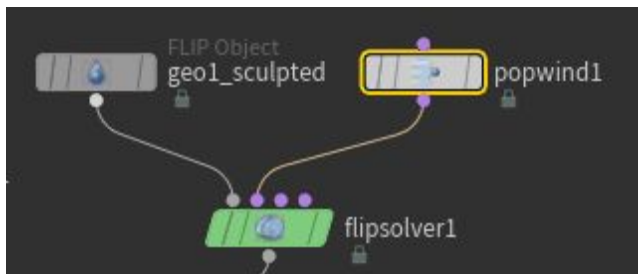
Nevertheless, if you want to use this...

1. make sure you have nothing selected
2. click the shelf item
3. select the object to embed and press Enter
(you'll be prompted in the scene view)
4. select the flipobject in your AutoDopNetwork and press Enter
(you'll be prompted in the scene view to do this)

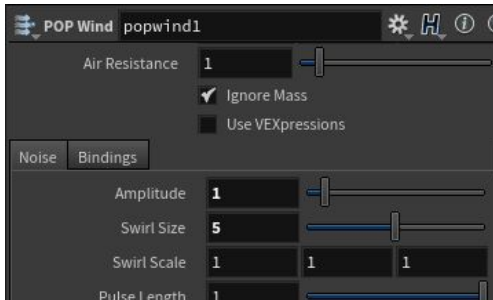
Fluid Forces

Particle Forces

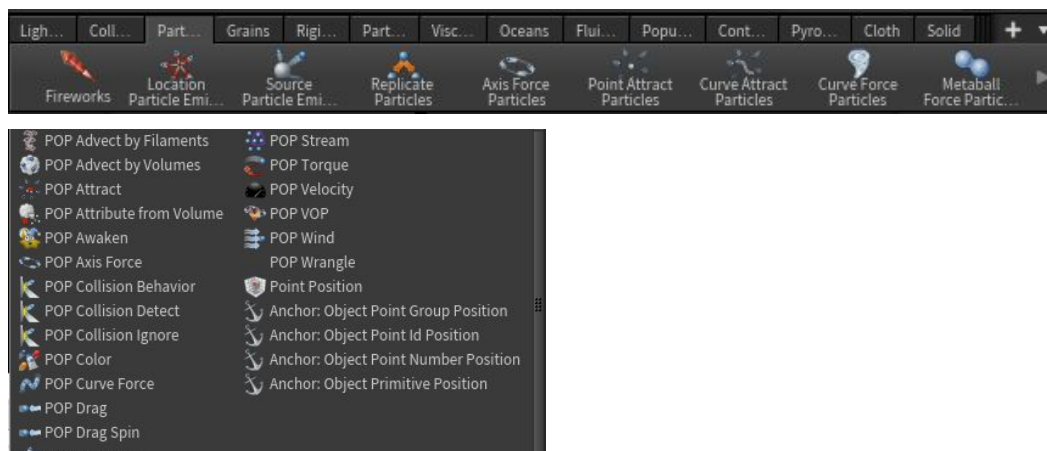
Remember that FLIP fluids are a combo of Houdini's old volume-based fluids and particle-based fluids. As such, you can apply any of the POP forces to FLIP fluids. These forces get plugged into the second input of a fluid solver (the Particle Velocity input)...



HINT: A common technique to create turbulent water is to connect a popwind and change up the noise amplitude and swirl...



The following is a copy of the forces section of the particles document. It gives an overview of the particle forces available. You can find these all under the Particles shelf or via the Tab menu search (search for “POP”)...



There are a ton of different forces you can apply to particles. It seems like these forces ONLY effect particles and nothing else.

NOTE: The gravity node is applied to particles as well. If you don't want it applied, you need to bypass it. Keep that in mind when applying these forces.

For example, rigid body objects are not affected by these objects. The particles may change direction as a result of the force and hit the rigid body differently (causing the rigid body to move differently), but the force itself won't move the rigid body.

Here are the types of forces you can apply...

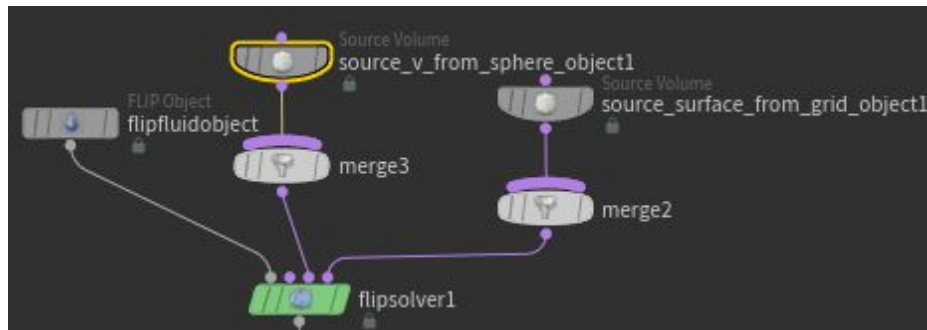
- Axis Force → for spinny things like hurricanes/turbines/dust devils/etc..
- Wind/Drag → applies a constant force to particles regardless of location
- Fan → Like Wind/Drag but applied to a certain area
- Force → something akin to gravity? How's this different than Wind/Drag?
- Flock → flocking behaviour for particles
- Curve Force → creates forces generated from a curve????
- Point/Curve Attract → attracts particles to point or curve

NOTE: See the section on Collision w/ Non-Dynamics... Just like how the placement of the POP Collision Detect node is important, the placement of your force node is also important. If it's after the final merge before the pop solver, it means all particles being fed into the popsolver will follow the force rules added by this node. If you only want these rules only applied to particles emitted from certain emitters, move it so that it's in the path of those emitters (before the final merge).



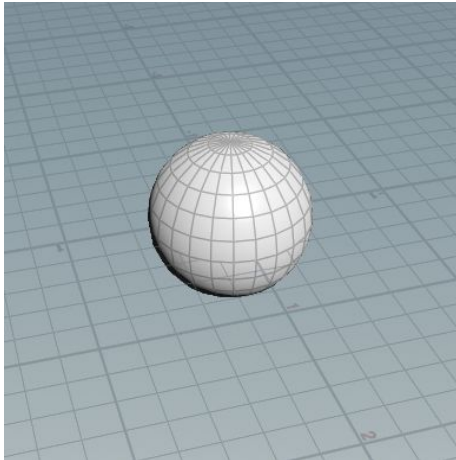
Volume Forces

Remember that FLIP fluids are a combo of Houdini's old volume-based fluids and particle-based fluids. As such, you can apply any of the volume forces to FLIP fluids. These forces get plugged into the third input of a fluid solver (the Volume Velocity input)...

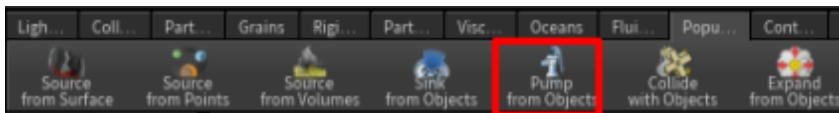


The easiest way to apply a volume force to an object is to use a Pump. You can do this by...

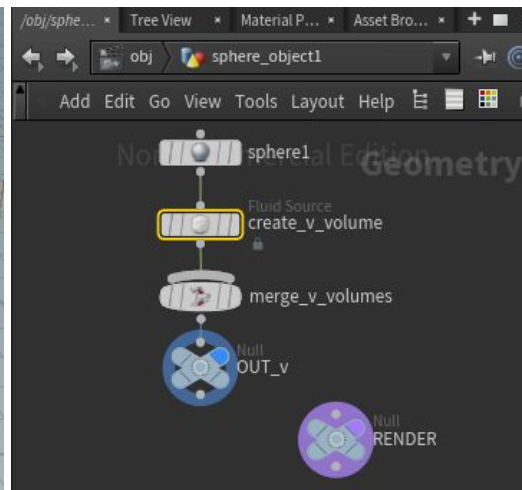
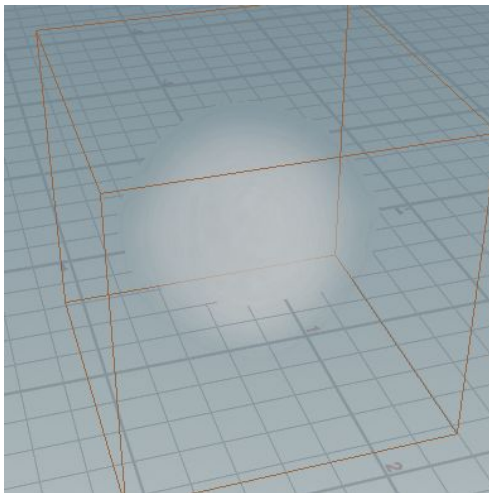
1. create an object in your scene (e.g. sphere)



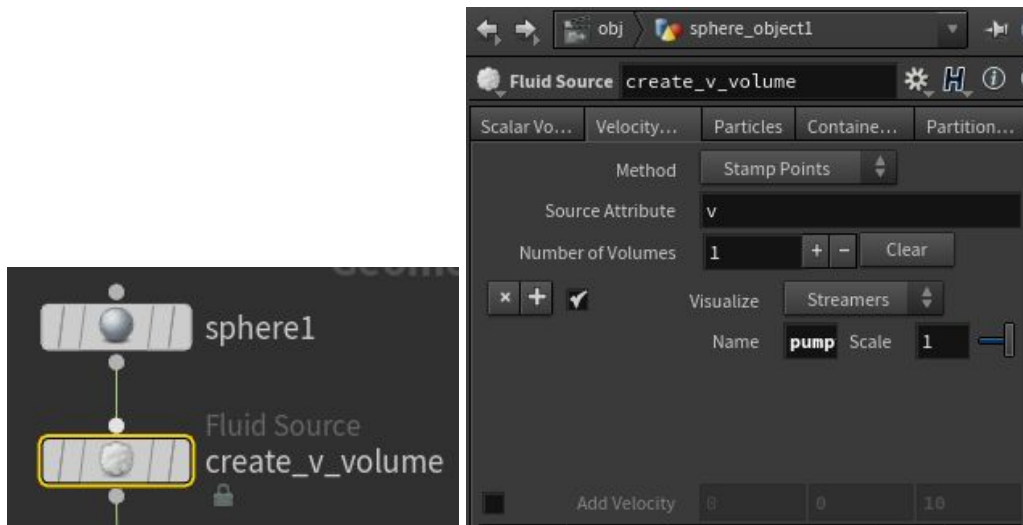
2. make sure nothing is selected
3. select the Pump from Object item in the Populate Containers tab



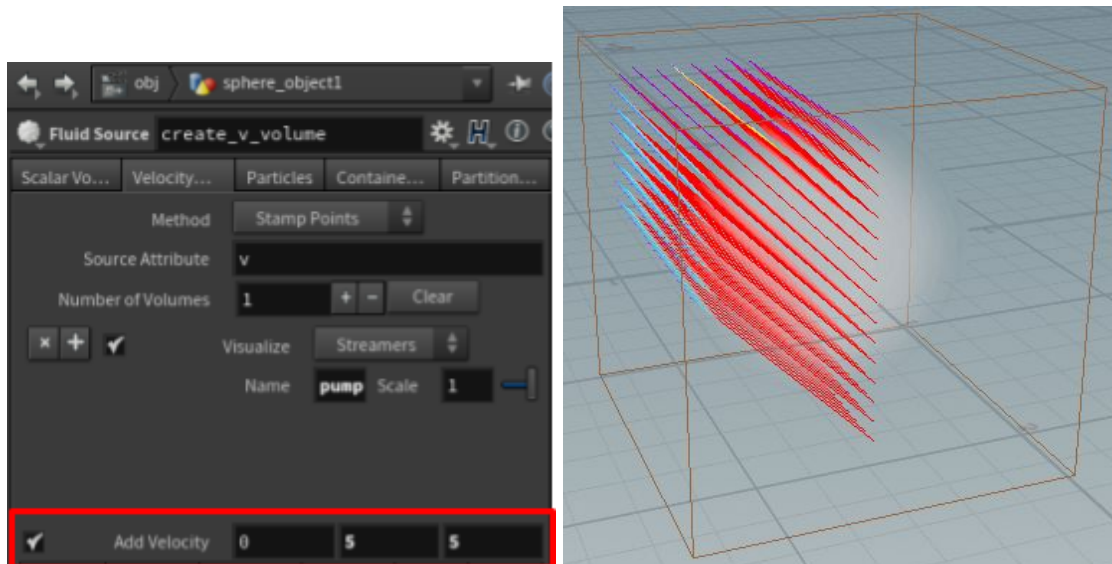
4. select your object and press Enter
(you'll get a prompt to do so in the scene view)
5. select the FLIP fluid object being fed into your flipsolver in the AutoDopNetwork and press Enter
(you'll get a prompt to do so in the scene view -- the prompt says gas instead of fluid, this is probably just a bug)
6. your object should show up as mist now -- dive into it
(this won't show up in the final render + you can hide it in your scene view if you want -- it will still function)



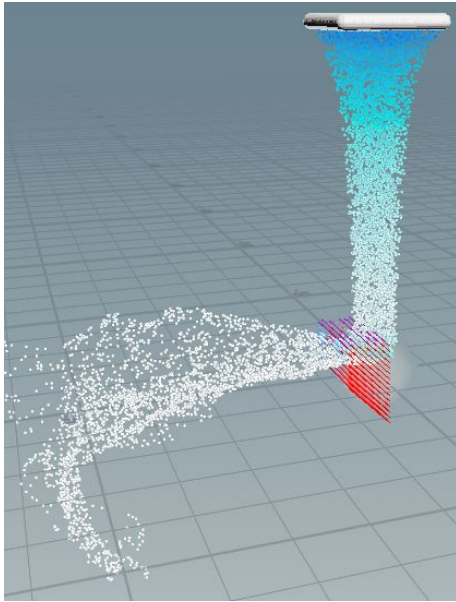
7. select the Fluid Source node and go to the Velocity Volume tab



8. enable Add Velocity and punch in a force vector
(red guides will show up in the scene view once you do this)



9. fluids coming into this area will now have this force applied

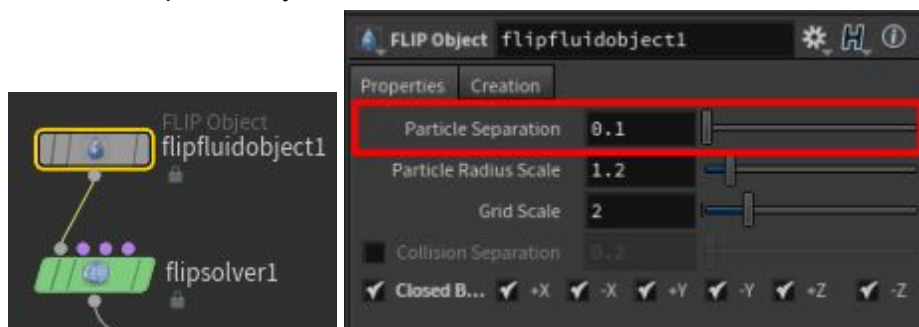


Fluid Accuracy

Particles

To control the accuracy of your FLIP fluid simulation, you can change the size of the particles that get generated. The more particles you have in your fluid simulation, the better it will look. It sounds like the more movement/collisions you have, the more particles you want.

To set the particle size, you need to go to whatever FLIP object it is that's being fed into the solver. Under the Properties tab, you'll see an option called Particle Separation. The smaller this is, the more particles you'll have.

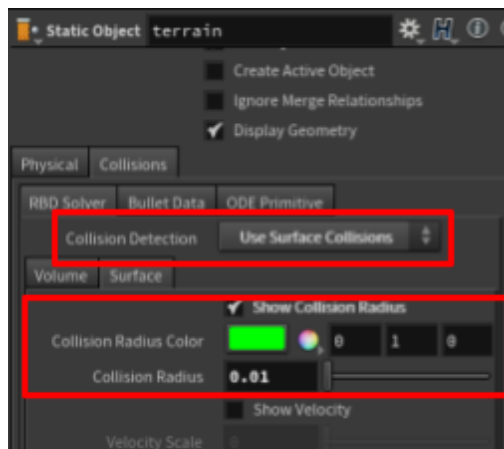


Obviously... the downside with doing this is that the more particles you have the longer it'll take to compute.

Collision Geometry

You'll notice a lot of times that your fluids will seep through objects instead of properly colliding with them. The reason for this is very likely because the objects your colliding with have bad collision geometry.

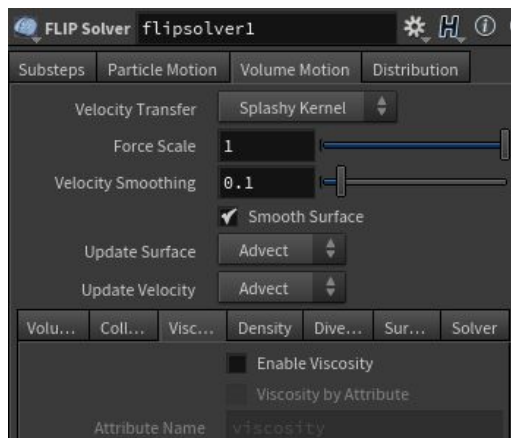
If you're dealing with rigid bodies or collision objects (e.g. ground object / terrain object / static object), you can use surface collisions instead of volume collisions to get better results. You can read the document on collisions for a detailed overview on this...



NOTE: One other option to use (if you want) is to try to make your geometry thicker, that way the volume-based collision may have an easier time generating correct collision geo.

Fluid Viscosity

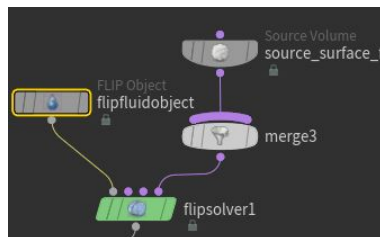
Enabling fluid viscosity manually isn't a simple process. The flipsolver has a Enable Viscosity checkbox under Volume Motion -> Viscosity, but enabling just that does nothing.



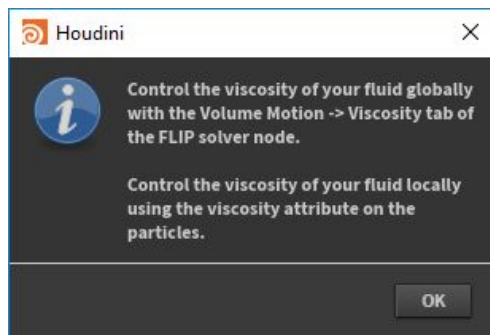
It turns out that there are a bunch of different places you have to enable viscosity to get it to actually function. There's a handy item under Particle Fluids shelf that does this for you called Make Viscous...



Make sure you have nothing selected and click it. In the scene view you'll get prompted to select the FLIP Object(s) to make viscous and press Enter. Go into your AutoDopNetwork, select them, and hit Enter.



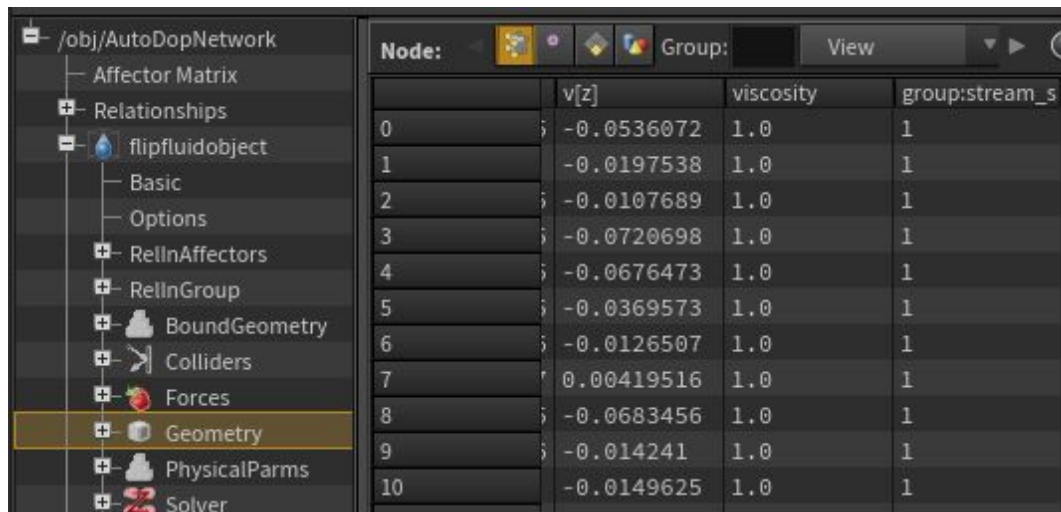
You'll get the following message box show up...



Essentially all this is saying is that there's 2 ways to set viscosity...

If you want to have different particles be different viscosities, you need to add to the viscosity attribute to your particles. If you look at the geometry spreadsheet, you'll see that the particles

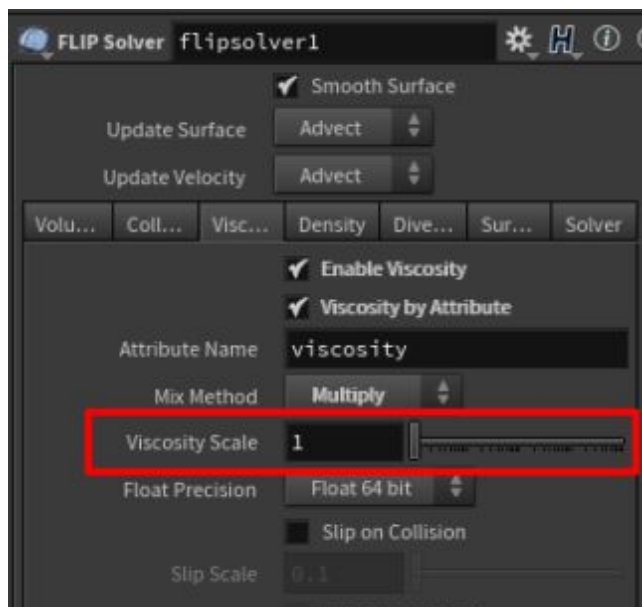
generated by your FLIP Object now each have a viscosity attribute of 1.0...



Node:	v[z]	viscosity	group:stream_s
0	-0.0536072	1.0	1
1	-0.0197538	1.0	1
2	-0.0107689	1.0	1
3	-0.0720698	1.0	1
4	-0.0676473	1.0	1
5	-0.0369573	1.0	1
6	-0.0126507	1.0	1
7	0.00419516	1.0	1
8	-0.0683456	1.0	1
9	-0.014241	1.0	1
10	-0.0149625	1.0	1

NOTE: There is no node in the dynamics context to add to an attribute. You'll have to do this via VEX, which I don't know how to do yet.

If you want all your fluids to have the same viscosity, you can use the Volume Motion -> Viscosity Tab of your flip solver...



The more you jack up the Viscosity Scale value, the more viscous your fluid will be. For example, something like 10000 would represent honey. You can get it high enough to represent something like bread dough.

NOTE: Remember from above that your fluid particles will have a viscosity of 1.0. Note the Mix Method -- what's happening is that your viscosity attribute is being multiplied by the Viscosity Scale value you set. If they're always defaulting to 1.0, you don't have to worry. If they're something else, you may want to use a different Mix Method.

