Houdini 16 Particles

Introduction

This physics portion of Houdini is called dynamics. There's a ton of different things you can do with dynamics, but this text is specifically about particles. You can think of particles as points/objects with no volume. When bound to sprites, they can be used to to simulate things like snow, rain, smoke, dust, etc…

Like all dynamics, particles have friction, bounce, and velocity attributes. Maybe mass/density attributes as well?

The shorthand for dynamics is DOPs (Dynamic OPerators). Particles seem to work under the dynamics portion of Houdini, but I've also seen them get called their own thing: POPs (Particle OPerators)?

Scene Scale and Dynamics

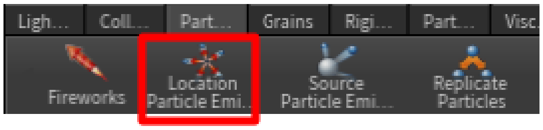
See the following section of the Houdini 16 Collisions document: Scene Scale, Dynamics Networks, and Dynamics Playback.

Not going to repeat those sections here -- all of them still apply.

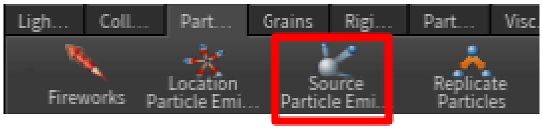
Particle Emitters

You can emit particles either from a point or from faces of a geometry.

To emit from a point, use Location Particle Emitter from the Particles shelf…



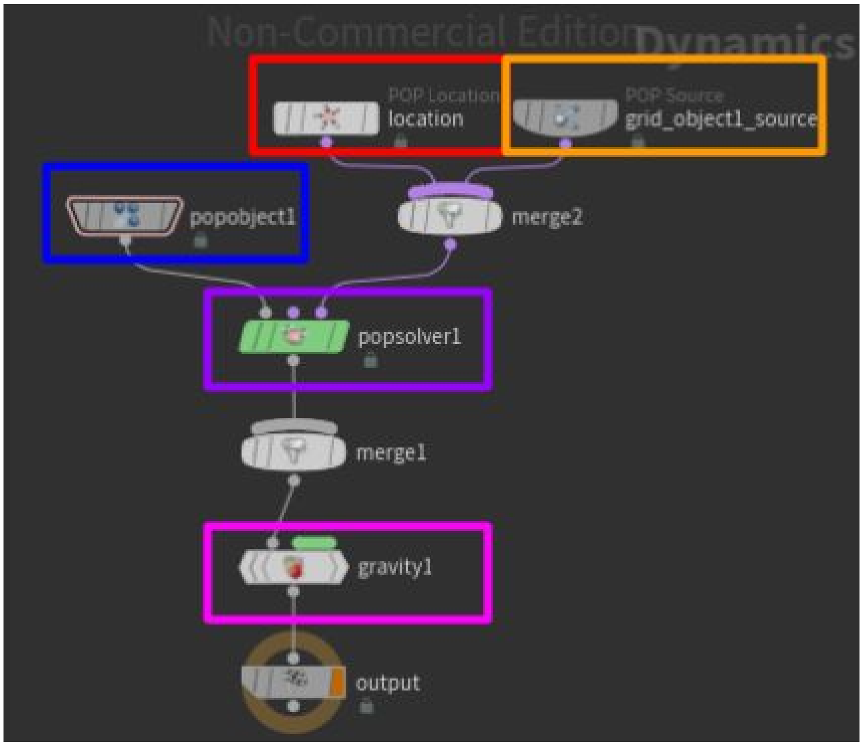
To emit from a set of polygons, select your geometry and use the Source Particle Emitter from the Particles shelf…



Regardless of which one you use, you'll get an AutoDopNetwork at /obj level.

> **NOTE**: If you don't remember, an AutoDopNetwork is a dynamics network that Houdini creates for you when you first use dynamics / adds to as you use more dynamics.

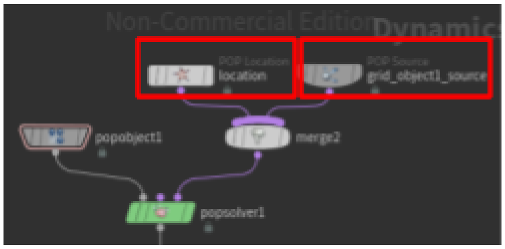
Inside the AutoDopNetwork, you'll see your particle emitters along with a popobject node being fed into a POP solver (Particle OPerator solver).



Here's what the DOP network setup means…

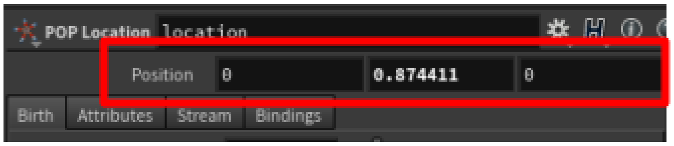
- POP Location controls particles coming out of a Location Particle Emitter
- POP Source controls particles coming out of a Source Particle Emitter
- POP Object makes the particles being generated interactable with other stuff in the dynamics -- e.g. collision with rigid bodies
- POP Solver performs the physics computations for the generated particles
- Gravity is what applies gravity (part of any AutoDopNetwork, not specific to particles)

The Location Particle Emitter and the Source Particle Emitter both have the same set of properties for how particles are generated. To access them, traverse into the AutoDopNetwork and select the node for whichever emitter it was you created (either POP Location or POP Source)...



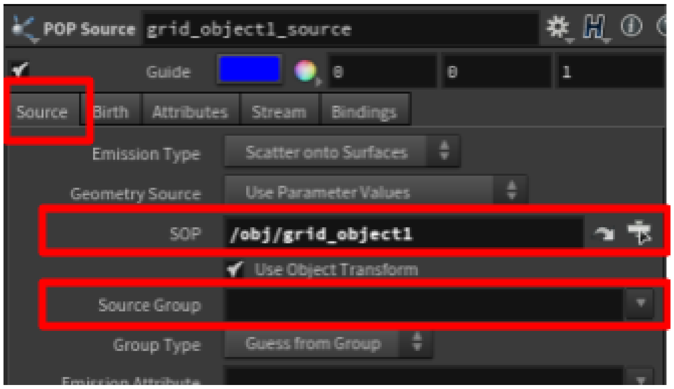
Location

If you used a Location Particle Emitter (POP Location), you can control which point the particles are generated from via the Position property…


POP Location location

Position 0 0.874411 0

If you used a Source Particle Emitter (POP Source), you can control which faces the particles are generated from via the Source tab...


POP Source grid_object1_source

✓ Guide 0 0 1

Emission Type: Scatter onto Surfaces

Geometry Source: Use Parameter Values

SOP: /obj/grid_object1

✓ Use Object Transform

Source Group:

Group Type: Guess from Group

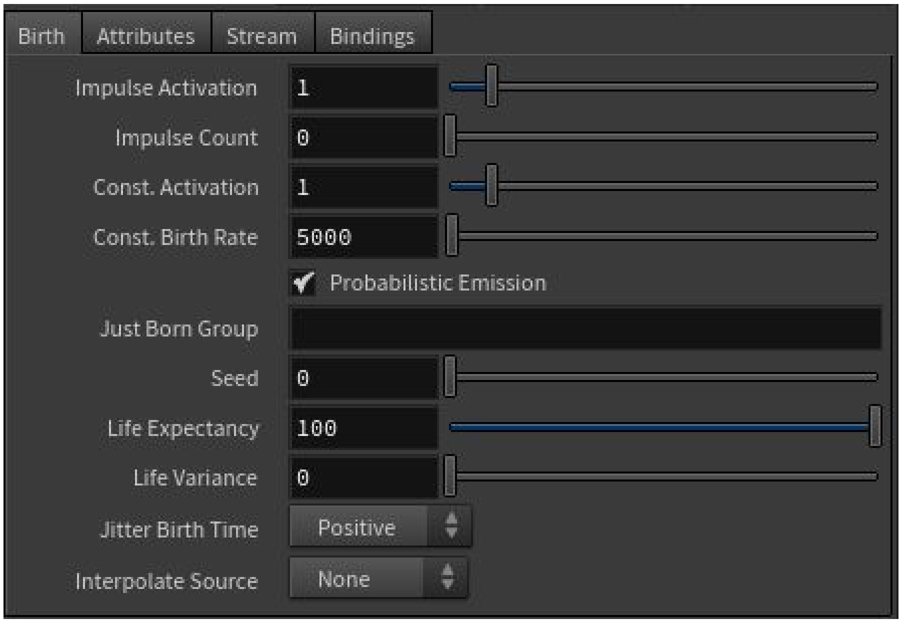
Emission Attribute

SOP is a reference to the geometry object, and Source Group is a reference to a group in that geometry (if empty selects the entire geometry).

> **NOTE**: By default it looks like particles are generated at random across the geo. You can try fiddling with the Emission Type dropdown to change this.

Creation/Destruction

You can control when/where particles are created and destroyed via the Birth tab…



Impulse Activation 1

Impulse Count 0

Const. Activation 1

Const. Birth Rate 5000

☑ Probabilistic Emission

Just Born Group

Seed 0

Life Expectancy 100

Life Variance 0

Jitter Birth Time Positive

Interpolate Source None

Impulse Act/Count

* Activation is a on/off value (0 off, anything else = 1) that determines if this feature is enabled.
* Count is the number of particles to generate per frame.

NOTE: It says each time the node cooks… nodes cook per frame, right?

NOTE: This CAN be used along side Const. Activation / Birth Rate

Const Act/Birth Rate

* Activation is a on/off value (0 off, anything else = 1) that determines if this feature is enabled.
* Count is the number of particles to generate per second.

NOTE: It doesn't generate all the nodes at the start of each second. Instead it generates the relevant amount at each frame. So if our playback was set to 24 FPS and our count was set to 5000, each frame we'd get 5000/24 particles.

NOTE: This CAN be used along side Impulse Activation / Count

Seed Seed for random number generator used for determining where particles show up.

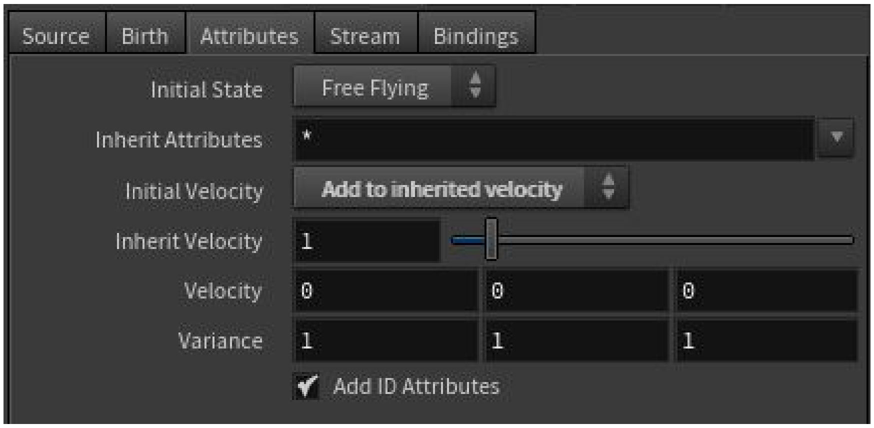
Life Expectancy How long each particle lives (in seconds).

Life Variance How much to vary the Life Expectancy value.

For example, if this was set to 5, the life expectancy would have a random number added to it ranging from -5 to 5.

Initial State

You can control the initial state of your particles via the Attributes tab…



Initial State: Free Flying

Inherit Attributes: *

Initial Velocity: Add to inherited velocity

Inherit Velocity: 1

Velocity: 0 0 0

Variance: 1 1 1

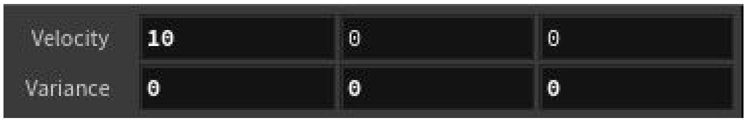
☑ Add ID Attributes

Velocity The initial velocity of the particles being generated

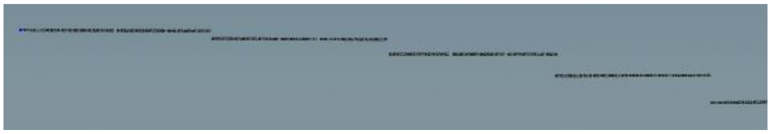
Variance How much to vary the velocity value.

> **NOTE**: What about the Initial/Inherit fields? These fields are only available if you're using a Source Particle Emitter (if you're generating your source from geometry). I think what it's for is if the geometry you're generating particles on is also a dynamics object (e.g. rigid body), its velocity and stuff will be mixed into the velocity of the particles being generated.

For example, if you had a velocity of (10, 0, 0) and a variance of (0, 0, 0), your particles would shoot down the X-axis like a straight line…

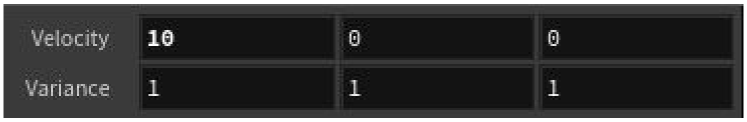


Velocity	10	0	0
Variance	0	0	0



> **NOTE**: Why is it dropping in steps instead of gradually? Probably because the FPS is higher than the physics iterations per second?

For example, if you had a velocity of (10, 0, 0) and a variance of (1, 1, 1), your particles would shoot down the X-axis like a straight line but each particle would vary its velocity in each axis by a random value from -1 to 1…



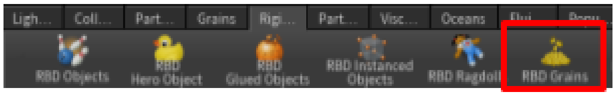
Velocity	10	0	0
Variance	1	1	1

Particle Collisions

Collisions w/ Dynamics

By default, collisions will interact with rigidbody objects and collision objects. They won't interact with each other though.

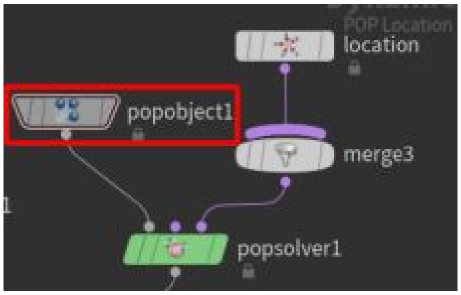
> **NOTE**: If you want particles to interact with each other, check out RBD Grains under the rigid body shelf…

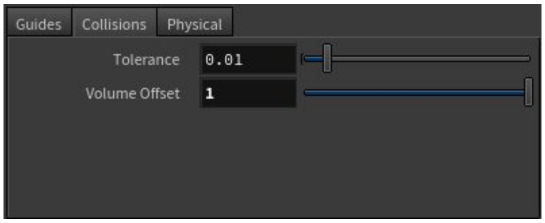


Particles don't seem to have a mass or volume associated with them. People on reddit say that the solver assigns some default mass/volume for each particle in order to calculate things like bounce and friction (I'm guessing mass/volume is required for these calculations).

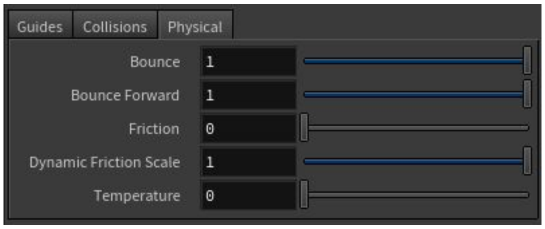
> **NOTE**: Remember that collision objects include: ground planes, static objects, etc.. -- basically anything from the Collisions shelf / anything being fed into a static solver.

The physical/collision details of the particles can be found under the popobject being fed into the particle solver in the AutoDopNetwork. If you select it, in the properties pane you'll find a Collisions tab and a Physical tab...





Tolerance 0.01
Volume Offset 1



Bounce 1
Bounce Forward 1
Friction 0
Dynamic Friction Scale 1
Temperature 0

Tolerance defines how close the particle has to be to the object to consider it collided? Since there is no volume for a particle (I think), it's calculated as a ray?

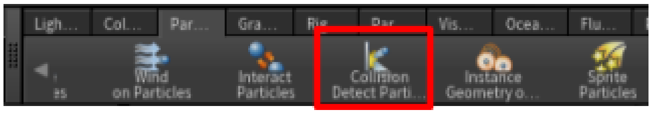
Volume Offset defines how much volumes are pushed out by a particle when they collide? Does volume here mean volumetrics (e.g. fog) or any object with a volume????

Bounce and Friction should be obvious. The rest I'm not too sure about.

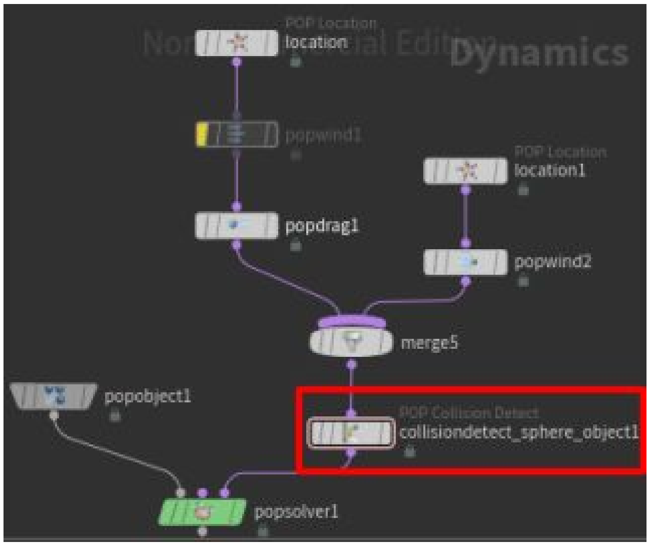
> **NOTE**: Popobject is where particles are generated. Check out the geometry spreadsheet when in the AutoDopNetwork and select popobject -> Geometry.

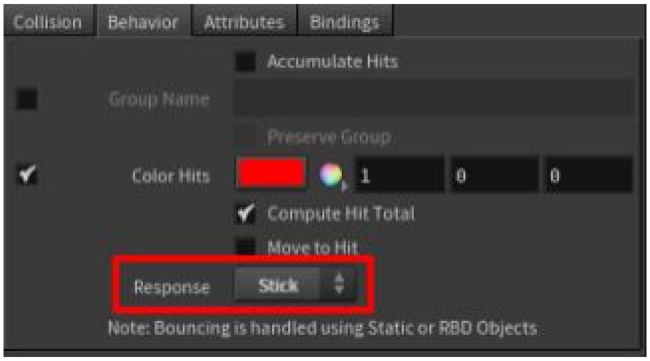
Collisions w/ Non-Dynamics

If you want to collide with a shape that isn't a dynamic (hasn't been added in as a rigid body or a static object or a ground plane or anything like that), you need to use the Collision Detect Particles item in the Particles shelf (it's near the end of the shelf)...



Once you do this, it adds POP Collision Detect node between your particles and the popsolver. In here you can adjust what happens on collision in the Collision tab via the Response dropdown (your options are slide/stick/die/stop -- all pretty self-explanatory).





☐ Accumulate Hits

☐ Group Name

☐ Preserve Group

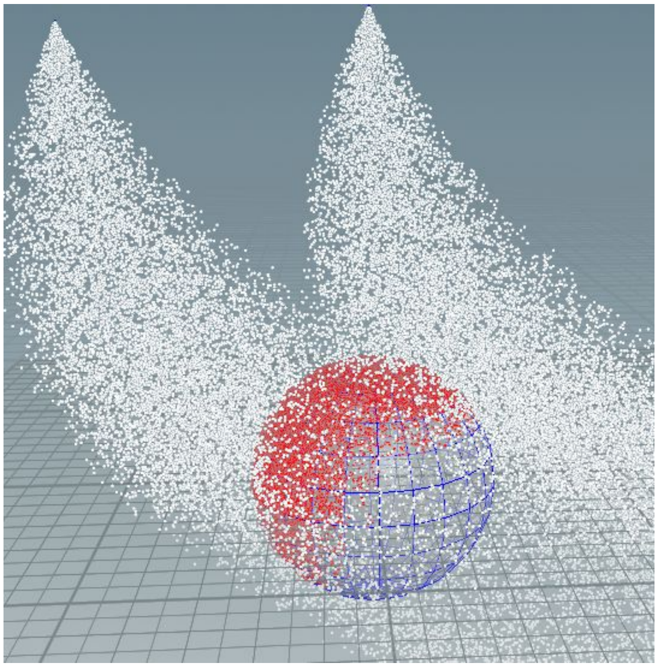
☑ Color Hits 🟥 1 0 0

☑ Compute Hit Total

☐ Move to Hit

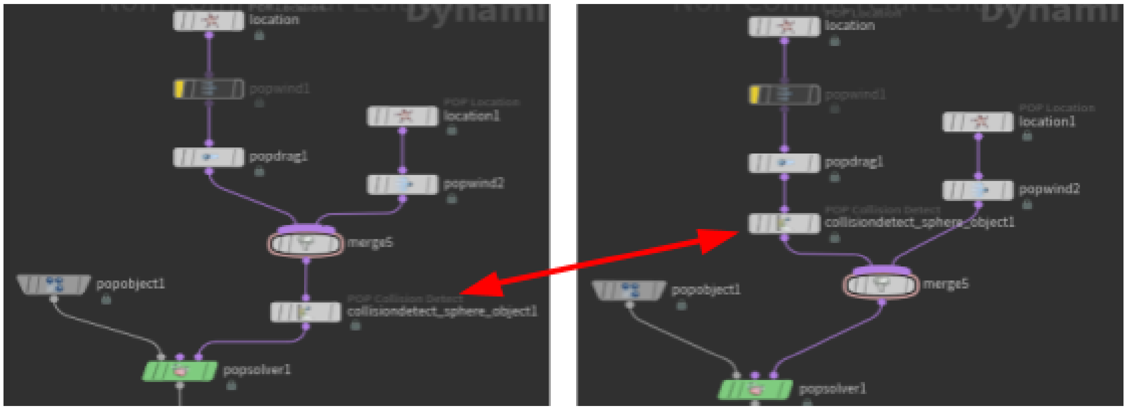
Response Stick

Note: Bouncing is handled using Static or RBD Objects



NOTE: The shape that the collision is being applied to is under the Collision tab.

NOTE: Notice where the POP Collision Detect node was generated. It's after the merge, which means all particles being fed into the popsolver will follow the collision detect 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).



Particle Forces

There 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.

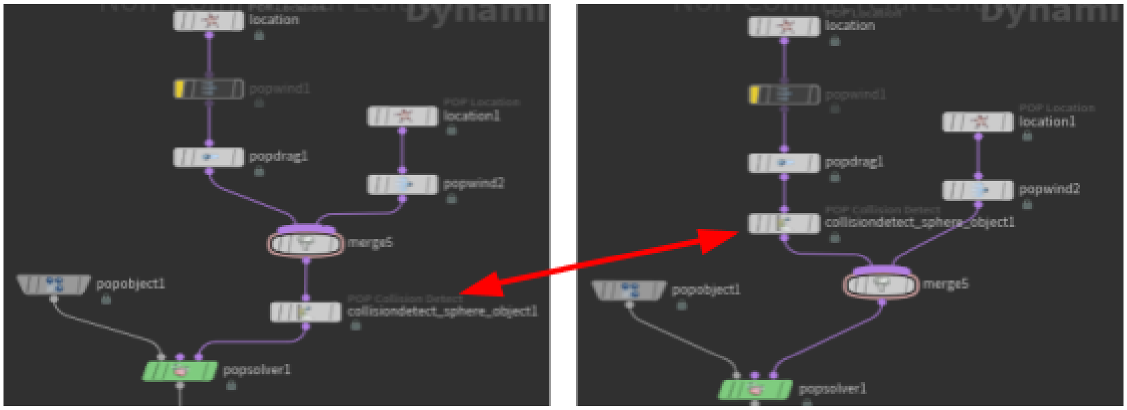
For example, rigid body objects are not affected by these objects. The particles may change direction as a result of the force and hit the rigid body differently (causing the rigid body to move differently), but the force itself won't move the rigid body.

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).

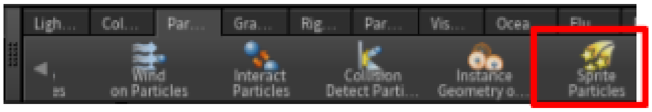


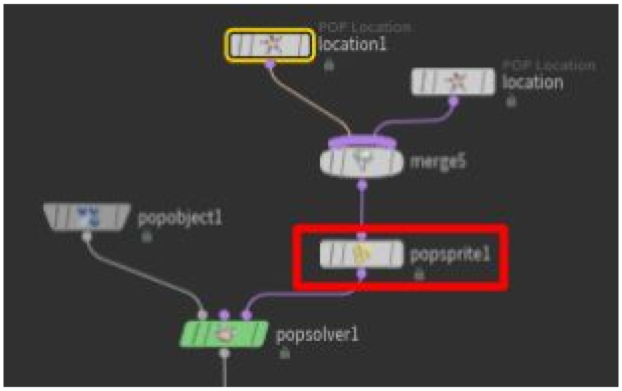
Particle Sprites

Sprites are textures that are attached to particles in billboard style.

> **NOTE**: Billboarding means that it's a flat 2d texture that always faces the camera. No matter how you orient the camera, you're always looking at the picture head-on. See https://gamedev.stackexchange.com/q/54871

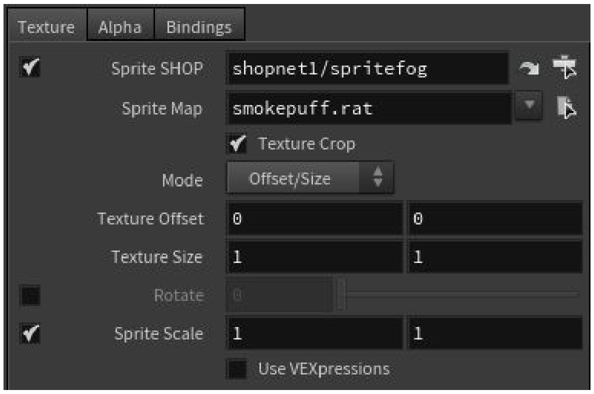
You can apply sprites to particles via Sprite Particles options of the Particles tab. Once you do, you'll get a POP Sprite node in your AutoDopNetwork.





> **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 POP Sprite node is also important. If it's after the final merge before the popsolver, it means all particles being fed into the popsolver will have sprites applied. If you only want sprites applied to particles emitted from certain emitters, move it so that it's in the path of those emitters (before the final merge).

The two important tabs in the popsprite node’s properties are Texture and Alpha. Texture lets you define the particulars of the texture displayed, while Alpha lets you control how the sprite fades away (or fades in)...



☑ Sprite SHOP: shopnet1/spritefog

Sprite Map: smokepuff.rat

☑ Texture Crop

Mode: Offset/Size

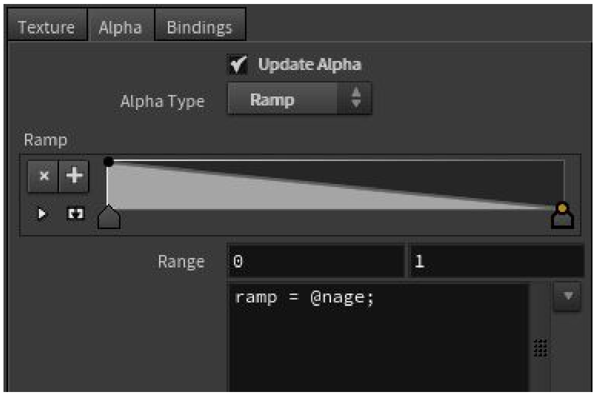
Texture Offset: 0 0

Texture Size: 1 1

□ Rotate: 0

☑ Sprite Scale: 1 1

□ Use VEXpressions



☑ Update Alpha

Alpha Type: Ramp

Range: 0 1

```
ramp = @nage;
```

Everything here should be pretty self-explanatory. Sprite SHOP/Map refers to the material to apply to the sprite. Texture Offset/Size is for things like tilesets. Rotate/Scale is for rotation and scale. Alpha Type controls how much alpha is applied at which point.

> **NOTE**: An important part of the lessons was using VEX to programmatically control some of these values. I’m going to skip this for now as it’ll probably be covered more in depth in later lessons.