1 POPs Basics

1 POP BASICS

1.1 INTRODUCTION

The term POP stands for *Particle OPerators*, and they operate on particle primitives. Particle primitives are geometric entities which have a collection of points – each of which contains their own attributes for such things as location, velocity, acceleration, etc. Various forces can act on these particles (Wind POP, Force POP, etc.) to cause a dynamic simulation that realistically models such phenomena as flocking, smoke, fire, and dust.

1.2 POPS VS PARTICLE SOP

Why would you use POPs instead of the Particle SOP? The Particle SOP can still be used to create particle systems – it is useful for systems that involve one source, a single collision object, and a single attractor. In the following cases, POPs provide more control than what the Particle SOP by itself can offer. Some of the features POPs offer are:

- Large number of local variables to represent particle attributes.
- Control over attribute manipulation and inheritance.
- Behaviours based on the occurrence of 'events'.
- Grouping of points based on rules or collisions.
- Support for per-particle rotation and instanced geometry.
- More powerful birthing options.
- Multiple collision objects/attractors.
- Ability to customise a particle system's behaviour with networks of POPs.
- Ability to write custom POPs with the HDK.

1.3 PARTICLE PRIMITIVES

Particle primitives are a fundamental geometry type which contain a collection of particles birthed from a single source. A POP network can contain several particle primitives, just as a SOP network can contain multiple geometry primitives.

For more information on particle primitives, see the *Geometry Types* section > *Particle Systems* p. 230).

1.4 COOKING

Particle systems are state dependent. Another way to look at this is that a POP merely takes in a geometry detail (which represents the current state of the particle system) and updates it to the current frame.

Because the state at the new frame depends on what happened in the past, POPs only cook in a forward direction. Playing the animation backwards won't update the geometry correctly. To see how the particle system is behaving, always play the animation in a forward direction.

Whilst the playback is occurring, you can interactively change POP parameters and the system will update accordingly. However, what you see is how the particles would behave if that parameter were changed at that point in time, not what it would look like if that parameter had been set that way all the time. To see just how the animation would play back, go to the first frame (usually frame 1) and play back the entire length of the animation.

Tip: Short-cut in the for getting to the first frame while in the Viewport: Click the *Reset* button in the Playbar, or use Alt on the keyframe button in the Playbar or Alt ← as a keyboard short-cut.

The POP SOP and the POP Editor Viewport are special because they know they are cooking POPs and send some special information to the POP network. This is accessible from both the POP SOP parameters, and from the View state parameters in the Viewport.

Note: The *opcook* scripting command doesn't know how to pass geometry into the POP network and therefore nothing will cook if you use it to try and force the cook with this command.

1.5 BUILT-IN MERGE SOP FOR POP OBJECT SPACE

POP Networks automatically use the SOP space of the object from which the POP is being cooked. For example, when you create a POP network, and you specify a geometry source from where the particles are birthed, the POP network will automatically assume the space, just as if the source geometry contained an Object merge SOP, since the Viewport doesn't have any space by default.

You can select the object whose space you would like to use in the POP Editor's Viewport > View state menu. See *View State – Select Transform Object* p. 4.

You can over-ride this automatic adaptation of object space by using the *Ignore Transform* button available in the appropriate POPs.

1.6 VECTOR VS COMPONENT DEFINITION

Vectors represent the direction and magnitude of a force. As particle systems inherently deal with reactions to forces, it is often necessary to define them in POPs. You can specify vectors POPs in one of two ways: Vector or Component form.

VECTORS DEFINITION

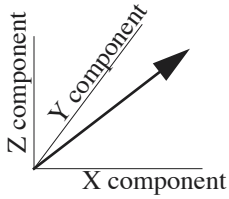
Vectors mean that you can use a mathematical expression such as:

```
vector3($VX, $VY, $VZ) * rotate(10,"X")
```

See the Expression Language > *Vector / Matrix Expressions* p. 12 chapter for a complete listing of vector functions. The advantage of using a mathematical expression for vector definition is that you can animate the vector using standard mathematical functions. In the above example for instance, the vector will rotate about the X axis 10 degrees. Most vector manipulations are easier to state as a vector expression. It would be difficult to express the rotation above in component form.

COMPONENT DEFINITION

When specifying vectors in component form, you basically define the vector as X, Y, and Z components as illustrated below:



Component form is useful when setting the vector to a constant value or when selectively changing only a few components.

1.7 PARTICLE CHARGE

When computing forces, some POPs use the charge to determine the direction of the force. The charge is used to multiply the computed force. If the resulting force is positive, the force will act in its intended direction. If the resulting force is negative, the force will act in the opposite direction.

For example, the Attractor POP computes a force that attracts the particle to the attractor. In this case, a positive force will move the particle towards the attractor. If the charge were negative, the negative force would move the particle away from the attractor, creating a repulsion force.

The Interact POP naturally computes a repulsive force so that particles are moved away from other particles. In this case, a positive force moves particles away while a negative force would move particles closer together.

You can set or alter the charge of a particle using the Property POP.

2 POPS AND THE VIEWPORT

2.1 WHAT THE POP EDITOR VIEWPORT DISPLAYS

POPs define how the particle system should behave. Unlike SOPs, by themselves, POPs don't contain any geometry. Instead, geometry must be passed into a POP network. The network will then cook and move particles according to how the POPs are connected.

The POP SOP (see *SOPs* section > *POP OP* p. 694) uses POP networks to cook its geometry data. This is how you would actually go about using POP networks to create geometry that can be used in your scene.

The Viewport in the POP Editor simulates how the POP SOP would cook the particle system. The parameters available in the View state of the POP Editor are similar to those found in the POP SOP (see: *SOPs > POP OP* p. 694).

2.2 VIEW STATE – SELECT TRANSFORM OBJECT

In the Viewport, you can select the geometry object from which the POP network will inherit it's space from by selecting a SOP network from the POP Editor's View state menu:

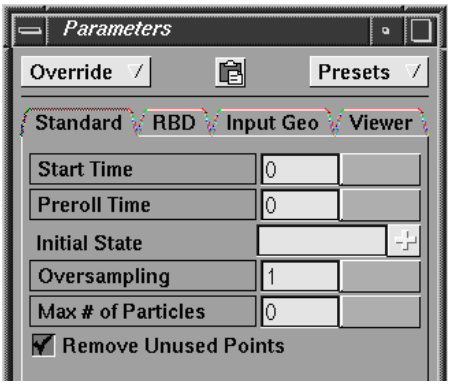
2.3 TRANSFORM OBJECT

When you create a POP network, you must specify a geometry source from where the particles are birthed. The POP network will automatically assume the coordinate space of the geometry in that object, just as if the source geometry contained an Object merge SOP. You select which object – whose space you would like to use – in the POP Editor's Viewport from this menu.

2.4 SELECT STATE

This will allow selections to be made in the Viewport but it is "read only" in that the points can't be moved. The selections are possible so that you can selectively view which points to inspect in the geometry spreadsheet (select from the pop-up Info at the bottom of the Viewport > *Geometry Spreadsheet)*.

2.5 POP VIEWER STATE PARAMETERS – STANDARD PAGE



Note: All of these parameters are actually the parameters of the POPNET (inside of which the POPs live, just like SOPs live inside Objects) being viewed. Thus these parameters can also be changed by going to the parameter dialog of the POPNET being viewed.

START TIME *timestart*

The start time is the time at which the POP simulation starts cooking. Before this, the particle system will be empty. The default is *Tstart* (frame one).

PREROLL TIME *timepreroll*

How many seconds of the POP simulation to bypass, after the reset time is reached. For example, if you put the number *33* into this field (and reset is at *Tstart*), frame one will show the simulation that was at a time of 33 seconds. In other words, the first thirty-two seconds have been bypassed, and the time at thirty-three seconds is shifted to frame one. The first thirty-two seconds must still be calculated in order to compute the status of the points, so you will notice some delay upon reset.

INITIAL STATE

Instead of starting from an empty particle system, you can use a geometry file to specify the initial state for the simulation. Create a geometry file by using a Geometry Output OP, and specifying a POP SOP as its output.

OVERSAMPLING

Oversampling is how many times in between frames to cook the simulation. For example, a value of 1 means to cook once per frame. A value of 2 means to cook at frame 1, frame 1.5, frame 2, etc.

Access Oversampling in the *POPnet > Parameters*, or in the POP SOP.

Note: The Sampling Rate in the POP Editor is not associated in any way with the oversampling in the SOP. The SOP's oversampling is the only one used when rendering, or for display of the SOP's data. The POP Viewport oversampling is only for the POP editor (there may be multiple SOPs referencing the same POP network with different oversampling rates).

MAX # OF PARTICLES

Specifies the maximum number of particles allowed to exist in the system.

REMOVE UNUSED POINTS

Removes all unused points from the input geometry. When points are removed, memory is conserved. On the other hand, re-using points saves the time needed to purge the points from memory.

2.6 POPS VIEWER STATE PARAMETERS – RBD PAGE

The RBD solver requires a number of new parameters to specify how to cook the network. These parameters can be found on the Rigid Body Dynamics page of the Particle CHOP, the POP SOP, and the POP Viewer parameter dialog. The following sections describe each of the new parameters.

REST THRESHOLD

This parameter is used to specify the relative velocity between objects that is considered as resting contact by the RBD solver. Resting contact is treated differently from collision contacts, which is why this division is necessary. Usually the default value should be used here unless your objects are very large.

CONTACT TOLERANCE

This is the distance between two objects at which the RBD solver considers them to be touching. If this value is made large, objects will collide and bounce in obviously unrealistic ways (i.e. before they actually contact). If this value is made small, you should increase the *Max Time Splits* value which will allow more accurate simulation of collisions but will also slow the simulation.

MAX TIME SPLITS

This parameter specifies how finely the RBD solver is allowed to split up a time step. The solver splits time steps to find the precise moment at which two object collide. The larger the number of time splits you allow the more accurately collision can be simulated, but also the longer the simulation may take. This parameter should take into account the maximum velocity at which particles will be moving in the simulation, the Contact Tolerance value, and the size of each time step (which is one frame time divided by the Oversampling parameter on the Standard page).

CONSTRAINT TIME

This parameter specifies the approximate time (in seconds) that the RBD solver can take to satisfy an unsatisfied constraint (see the Constraint POP for information about setting up constraints). The smaller this value is, the more rigidly constraints behave, but the larger the forces are that get applied to the objects. The larger this value is, the less likely you are to produce an unstable situation in your simulation.

SOLVER TYPE

Choose from: *Euler (fastest, but least accurate)*, *Midpoint*, and *Runge-Kutta (slowest, but most accurate)*.

2.7 POP VIEWER STATE PARAMETERS – INPUT GEO PAGE

OBJECT N / SOP N

Specify the Input geometry objects here by specifying their *Object* and *SOP* from the pop-up menus provided. These let you set the context geometry to use for the POP network.

CONTEXT GEOMETRY

All POP networks can have 'Context Geometry'. These context geometries are set in the parameters of the POPNET, the Particle CHOP, or the POP OP. In the POP OP they can also be set by wiring OPs into the inputs of the POP OP. These context geometries can be referenced by the following POPs: Source POP, Collision POP, SoftBody POP, Attractor POP, Creep POP (All of which have the parameter – *Geometry Source > Use Context Geometry)*. So, instead of hard-coding a specific piece of source geometry or collision geometry, a single POP network can be used by several POP OPs to generate several different outputs, depending on the geometry input into the POP OP.

context geometry - example

The main point of 'Context Geometry' is that a POP network can be used as a macro instead of a dedicated effect. For example, suppose you want to have a piece of geometry emit particles from it's points, and then have those particles fly over to the points of some other piece of geometry. You create a Source POP that references an object, an Attractor POP that references some other object, then you put a POP SOP into an object somewhere and you're done. You don't even provide any inputs into the POP SOP. Now suppose you have another pair of objects where you want to have this same effect. Without Context Geometry you would have to create a whole new POP network that was identical to the first, except it referenced different objects in the source and attractor POPs. Suppose you have 50 of these pairs of objects where you want to use this effect - that's 50 POP networks.

Enter Context Geometry. Change the Source POP to use Context Geometry 1, and the Attractor POP to use Context Geometry 2. In your POP SOP, connect the source geometry to input 1 and the attractor geometry to input 2. But now you want to do this same thing with 50 pairs of objects. You still need 50 POP SOPs. Each one takes as input the pair of geometries you want to work with. But you only need one POP network. This is a tremendous advantage if you ever want to make a change to the POP Network.

In the POP Viewport, you go to the 'Operation Parameters' dialog, and in the 'Input Geo'. On this page, you specify the Object and SOP to use for each of your four Context Geometries. So, for this example, when you're designing your POP network, you would set the 'Object 1' and 'SOP 1' to point to the source geometry. You

would set the 'Object 2' and 'SOP 2' to point to the destination Attractor geometry. Now in the POP Viewport, the Source POP will behave exactly as if you had entered these Object and SOP values into its parameters. Even templating geometry works.

2.8 POP VIEWER STATE PARAMETERS – VIEWER PAGE

TRANSFORM OBJECT

When you create a POP network, you must specify a geometry source from where the particles are birthed. The POP network will automatically assume the coordinate space of the geometry in that object, just as if the source geometry contained an Object merge SOP. You select which object – whose space you would like to use – from this menu (this is the same menu in the POP Viewport > Select state > sub-icons *Transform Object*).

CACHE SIZE

Specifies the size of the cache – which will hold the result of the pop simulation for faster subsequent playback. A 0 turns off caching, -1 sets infinite caching, and any other value sets the maximum number of frames to cache.

Note that to get an up-to-date and accurate particle simulation, you should reset the simulation with the yellow *Recook simulation from Start* button at the top of the POP Viewport.

3 PARTICLE ATTRIBUTES AND LOCAL VARIABLES

3.1 INTRODUCTION

Many POPs have access to variables that represent the particle's attributes. Particle attributes are stored as point attributes. Some attributes are added by certain POPs while other attributes are inherited from the birthing source.

Following, is a list of all available local variables, what they represent and which POPs add those attributes. Variables marked with a * are only present if they are added by one of the POPs. Variables marked with a + are only present if they were inherited from the birth source.

If an attempt is made to refer to a variable before the attribute has been added, a default value will be substituted. The POP tile warning will provide information on what the default value is.

For a general discussion of attributes, see *Geometry Types > Attributes* p. 233.

To get at these attributes from SOPs, you should use the *point()* function. See *Expression Language > SOP-Specific Expression Functions* p. 24.

3.2 LOCAL VARIABLES

Local Variable	Description	POP Which Adds Attribute
AGE	Age	
ATTRACT *	Attractor Point	Property POP
AX AY AZ	Acceleration	
BBX BBY BBZ	Bounding Box	
CA *	Alpha	Color POP
CR CG CB *	Diffuse Color	Color POP
CHARGE *	Charge	Property POP
DEAD	Point is dead	
DIST *	Distance from point to collision	Collision POP Limit POP
DRAG *	Drag	Property POP
FOLLOW *	Leader to follow	Property POP
GEN *	Generation	Source POP
HCR HCG HCB *	Color for last collision	Collision POP
HITID *	ID for last collision	Collision POP Limit POP
HITTIME *	Time for last collision	Collision POP
HMAPU HMAPV *	Map UV for last collision	Collision POP Limit POP

Local Variable	Description	Attrib added by...
HNX HNY HNZ *	Normal for last collision	Collision/Limit POP
HTX HTY HTZ *	Last collision's Hit Position	Collision POP Limit POP
HU HV *	Prim UV for last collision	Collision POP
ID *	ID Number	Source POP
ITER	Processing iteration number	
JUSTHIT	Point has just collided	
LIFE	Percent of life used	
LIFESPAN	Expected life of particle	
MAPU MAPV MAPW+	Texture Coordinates	
MASS *	Mass	Property POP
NPT	Total Number of Points	
NGRP	Total number of points in group	
NUMHIT *	Number of times a particle has collided	Collision POP Limit POP
NX NY NZ +	Normal	
ORIGIN *	Original Source point was birthed from	Source POP
PARENT *	Parent's ID Number	Source POP
PSCALE *	Scale	Property POP
PT	Point Number	
PVX PVY PVZ *	Previous Velocity	Up Vector POP
RESTX RESTY RESTZ	Rest Position X/Y/Z	
ROTA *	Rotation Angle	Rotation POP
ROTX ROTY ROTZ *	Rotation Axis	Rotation POP
SLIDING	The sliding state of the particle.	State POP
SPEED	Absolute speed of part.	Location, Rigidbody, Source, Softbody, Split POP
SPEEDMAX *	Speed maximum	SpeedLimit POP
SPEEDMIN *	Speed minimum	SpeedLimit POP
STOPPED	Point is stopped	
SUPPOS / SUPPVEL SUPPUP / SUPPAGE	Suppress default Position, Velocity, Up-vector, and Aging rules.	Suppress Rule POP
SUPPROT / SUPPANGVEL	1 if particle suppressing its default rotation rule / angular velocity rule	Suppress Rule POP
STUCK	1 if particle is "Stuck" to a collision object	State POP

TIMEINC	Time Increment	
TX TY TZ	Position	
UPX UPY UPZ *	Up vector	Up Vector POP
VX VY VZ	Velocity	

RBD VARIABLES

Local Variable	**Description**	**POP Which Adds Attribute**
COMX, COMY, COMZ	Centre of Mass X, Y, Z.	Property POP
BOUNCE	Bounce coefficient; energy lost due to collision.	Property POP
FSTATIC / FDYNAMIC	Amount of Static and Dynamic friction.	Property POP
RBODY	Particle is a rigid body.	State POP
ACTIVE	Particle is an active rigid body.	State POP
OVERRIDE	Particle instance transform should be overriden.	State POP

4 POP EVENTS

4.1 INTRODUCTION

In the world of POPs, events occur when "something" happens. That something can be the collision of a particle with some geometry or a particle passing a certain threshold or anything else that can be detected about a particle. When the event occurs, it can be used to trigger certain POPs to activate which allows for very interesting behaviours.

It is possible for particles to start birthing when a collision occurs. The particle color can change because other particles have died. In these examples, the collision and the death of a particle causes the event to be generated. These events then trigger the Source or Color POPs to activate.

4.2 GENERATING EVENTS

The POPs that produce events are:

Collision/Limit POP	An event is generated when a collision occurs.
Event POP	An event is generated when the expression rule evaluates to >0 for any of its input particles.

Events are identified by a string name. Whenever the POP's criteria for producing events is met, the named event is generated. The POP info button (on the POP tile) gives information on which events are currently occurring.

4.3 ACTIVATING POPS AND THE POPEVENT() FUNCTION

Most POPs have an *Activation* field. This parameter acts as a flag which indicates if a POP should be active or not. The POP will only be active when the field evaluates to greater than zero.

For example, a logical expression like:

```
$F == 5
```

means the POP will only be activated and act on its inputs for frame 5, since during that frame, it is a true condition and will return a boolean '1' value (which is >0, and thus activates the POP).

POPEVENT()

In addition to the regular expression functions, there is a special function available within POPs to detect events. The function *popevent(name)* returns 1 when the event is occurring and 0 otherwise. This function should only be used within POPs. Using this function, a POP can be activated only when an event occurs.

4.4 EXAMPLE

This is an example of how to generate particles when a collision occurs.

1. Launch Houdini and open a Network Editor pane to edit *SOPs*.

2. We want to edit the geometry in *geo1* (rename the *logo* object), so Alt click on *geo1* to take you into its SOPs. By default *geo1* contains a File and Xform SOPS – Delete these.

3. We need something to birth from as well as a collision object. Add a Circle SOP, which will be used as a birthing source. Also add a Grid SOP, orient it on the *ZX plane*, and move it down a bit by setting *Centre Y* to: -1.0 . The circle primitive in the Circle sop has only one point in it, so it will birth a single particle.

4. Switch to a POP Editor pane, and place a Source POP in the Network Editor. Set the birth source Object to: *geo1* ; set the SOP to: *circle1* .

5. We only want one particle to birth at frame 1, so use:
$F == 1 as the *Activation* expression (make sure *Expressions* is enabled).

6. Change the *Birth Mode* (*Birth* page) to *Birth all on activation* so all of the particles will be birthed whenever the Activation is enabled (whenever $F equals 1).

7. Set the *Birth Rate* to: 1.

8. Finally, give the particle some velocity. Do this in the *Attributes* page, and set the *Initial Velocity* to: *Set initial velocity.* Set Velocity Y to: -2, and reduce the *Variance* to: 0, 0, 0.

9. Collide these particles with the grid by appending a Collision POP. Set the Collision Object to: *geo1*, SOP: *grid1* .

10. Give the event generated by particles colliding with this grid a name by entering "birth" in the *Collision Event* field (*Behaviour* page).

11. Set the Display and Cook flags (blue and violet respectively) to be the *collision1* POP.

12. Now *Play* the animation by clicking on the single-frame advance until you reach frame 15. You should see a single particle moving downwards, hit the grid and die. At frame 15, the info button (on the POP tile) on the Collision POP should tell you that the birth event has occurred.

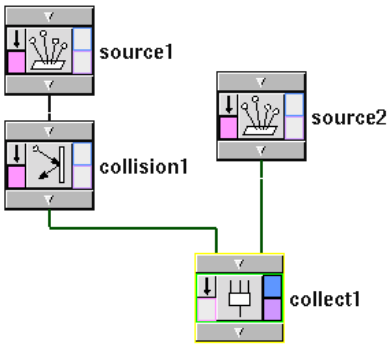
13. Now advance one frame to frame 16. The info button now indicates that no events are occurring. This is because the event has already occurred and ended.

14. We want new particles to be birthed when the first particle collides with geo1. To do so, place another Source POP – source2. Set the birth Object to: *geo1* and the SOP to: *grid1* .

15. In the *Activation* field, enter the expression: *popevent("birth")* .

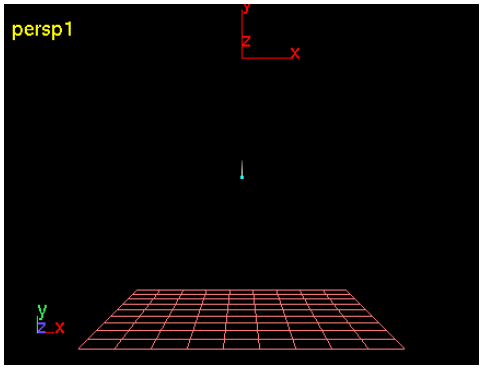
16. Set the *Birth Mode* to: Birth all on activation (*Birth* page).

17. Append a Collect POP to the *collision1* POP; wire source2 as a second input into the Collect POP. Set the Display and Cook flags on the Collect POP, and the set the Template flag on the *collision1*. Your POP network should look like this:



18. Turn on the Viewport's Point display (Viewport ✣ > Viewport options) in order to see all the particles (note: overrides Display POP).

19. Move back to frame1, and *Play* the animation. You see a single particle (generated by source1) moving downwards. When it collides with the grid, it dies, and every point on the grid births a particle (the "birth" event triggered source2).



When the particle collides with the grid, each point on the grid will birth a particle.

5 POP TIPS

5.1 GENERAL TIPS

- The Up vector is transformed so that it always points in a direction perpendicular to the velocity. To do this, the previous velocity must be known. When using up vectors, make sure the Source POP sets some sort of initial velocity so that Houdini will know how to modify the up vector as soon as the particle's velocity changes direction.

- For very long animations, if your particles suddenly die, remember that the default lifespan is 100 seconds or 3000 frames if there are 30 frames/second.

- When splitting particles, remember that the default Inherit Attributes mask is *, which means that the source particle's acceleration will also be inherited. However, lifespan, age and velocity are not inherited.

- Setting the drag attribute in the Property POP only sets how much drag the particle will have. To actually see the effects of drag, a Drag POP is needed.

- If a particle is moving because its position is being animated explicitly, it actually has no velocity as far as Houdini is concerned. Its velocity vector is (0, 0, 0) regardless of the fact that it is actually moving. This is important to consider when using the Follow POP because the leader's velocity is used to determine how the particles follow. If the leader has no velocity, the ambient speed should be used to determine how fast the particles try to follow.

- Even if a source is moving, it still has no velocity attribute. Use the Trail SOP to add a velocity attribute to the source geometry.

- For complex particle animations, don't use the *Real Time* option in the Playbar. Unlike geometry, cooking particle systems requires that every frame be cooked because the state of the particles is dependent on previous states. If the time is advancing quicker than the frames can be computed, the system will bog down. It is best to test particle animations not in real time.

  Particles that bounce off of collision geometry have their velocities modified with an applied acceleration. Remember this acceleration may be inherited if splitting particles from bounced particles.

5.2 TIPS ON COLLISION DETECTION (COLLISION & LIMIT POPS)

NON-MOVING GEOMETRY

- For collisions with geometry which is not moving, the user should leave *Geometry Moves With Time* unchecked. (The term "moving" is loosely used to also include geometry which is deforming, rotating in place, etc.).

- The die/stop/stick/continue behaviours should work well with *Collision Tolerance* left at 0. If some particles go through the geometry without colliding, a small tolerance value should normally help.

- For the bouncing behaviour, setting the tolerance may help reduce escaping particles, especially when the geometry is complicated, or particles are bouncing at or near a corner.

- The value set for the tolerance is a *Distance* measurement representing how much look-ahead the intersection algorithm should use beyond the particle's true motion. For instance, if the particle is moving at a speed of 30 units/sec in some direction, at a frame rate of 30 frames a second, then in one frame that particle should move 1 unit along its line of motion. A tolerance value of 0.01 would pretend that particle is really moving 1.01 units in that frame, so a surface 1.005 units away will be collided with on this frame, rather than the next one. (This resolves the problems that may arise if there was a surface really close to 1 unit away... the algorithm may or may not see that surface on this frame, and may or may not see it as a collision on the next frame).

- You can select *Bounce Accuracy*, which is the maximum number of bounces allowed within one frame.

MOVING GEOMETRY

- It is up to you to specify that the collision geometry is moving (i.e. the collision algorithm only changes if you check the *Geometry Moves With Time* parameter, regardless of whether the geometry is really moving or not). When that parameter is enabled, the *Oversampling* parameter becomes active and *Bounce Accuracy* inactive.

- *Oversampling* in the Collision POP is similar to the oversampling parameter for the whole POP network (check parameters for the POP network). The difference is that the one in the Collision POP only oversamples that POP, and doesn't force the whole POP network to be oversampled (which would unnecessarily slow things down). Oversampling is especially helpful if the geometry is moving/deforming very quickly.

- It is important to set the Collision Tolerance to something nonzero. Along with telling the intersection algorithm to look-ahead for collisions by the specified value, the tolerance also makes the intersection algorithm look around the particle (left, right, up, down, and back) for collisions within the specified tolerance. This is helpful for when the geometry moves towards the particle from behind, or from the side.

SETTLING

- Say you have a particle bouncing on a plane, with a gravity force acting down. If the gain normal on the bounce is set to be less than 1, the particle bounce height will diminish and eventually you'd expect the particle to settle on the ground. (Actually, you often need to go down to a gain normal around 0.6 or so before the bouncing diminishes properly). In order to save from having to calculate collisions for these settled particles on subsequent frames, and also in order to reduce the number of particles which fall through the surface instead of settling, the Collision POP tries to detect for such settled particles, and sets their state to *stopped*. Basically, this happens when the Collision POP detects that the bounces have diminished enough, and where there is an acceleration in a direction *perpendicular* to the collision surface which is forcing the particle back towards that surface.

CREEPING

- There are no special accomodations for creeping particles, which means they usually fall through. A particle is said to creep on a surface when its velocity on that surface is almost completely in a tangential direction to that surface. e.g. a particle sliding down an incline. These cases are not guaranteed to work.

ORDER OF COLLISION POPS IN A NETWORK

- When setting up the POP network, especially one with more than one Collision POP, the order of those collision POPs is quite important. Suppose you have Collision A followed by Collision B. (i.e. Collision A output goes into Collision B input). Now this network is cooked, and collisions have to be determined for a given particle. First, we check if there are any collisions with collision geometry A) Depending on the setting of *Bounce Accuracy* (for non-moving geometry), or O*versamples* (for moving geometry), there may be more than one bounce within the time frame. Say there are two bounces. Then when we get to Collision B, we check for collisions only in the motion remaining for that particle. That is, the particle has already travelled some distance to undergo the two collisions (and some time in the time frame has therefore elapsed). Now we only look for collisions starting at where Collision A left it off (which would be at the point of the second bounce) and going only for as much time as there remains within this frame (which would be less than a full frame of time because of the collisions that have already taken place).

- Special care must be taken if multiple Collision POPs whose geometries are close to each other. Because the particle won't even consider the collision geometry of Collision POPs below the current Collision POP, and thus it may appear to go through the surface even through it did the right thing. Looked at another way, this is sort of like setting a precedence/priority for the surfaces.

TYPES OF GEOMETRY ALLOWED FOR STICKING

- Particles can stick to triangles and quadrilaterals.

- Sticking is not supported for polygons with more than 4 sides.

LIMIT POP

The algorithms for collisions in the Limit POP have been changed, but should appear to work the same to the user. Note that the + *Limit* and - *Limit* parameters can be animated, which effectively means that the box or sphere can be made to move around and change in size. However, the collisions done in Limit POP assume the limit surfaces are *not* moving, so if moving boxes/spheres are desired, the user should really use a Collision POP instead.

- Since the Collision and Limit POPs used different intersection algorithms, you may want to try both. When the collision surface is either a sphere or a box/plane), because one may give better results than the other. The Limit POP is somewhat better when it comes to particles settling/creeping on the surface (at least when using a Box limit.

POINTS MISSING COLLISION PLANE

When the velocity vector is too small, it will miss the Collision Plane. One way to solve this, put a Velocity POP BEFORE Collision POP multiplying the velocity by 20, and then put another Velocity POP after the Collision POP dividing the velocity by 20.

Velocity POP 1: $VX*20, $VY*20, $VZ*20

Velocity POP 2: $VX/20, $VY/20, $VZ/20

5.3 FURTHER REFERENCE

See the other demos located in: *$HD/POPs/*

2 POPs – Particle OPerations

1 ACCELERATION POP

1.1 DESCRIPTION

The Acceleration POP can be used to explicitly set a particle's acceleration.

The acceleration can be evaluated either in component form or as a vector. See *Vector vs Component Definition* p. 2 for a discussion on differences between component and vector definition.

The difference between velocity and acceleration is that velocity is the *rate of travel*, whereas acceleration is the *change in rate of travel*.

1.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

APPLY MOTION

Specifies what types of points to apply the acceleration to:

to all points	Apply the acceleration to all of the particles.
to moving points	Apply the acceleration only to those particles which are in motion. This means that stopped and dying particles will not be affected.

ACCELERATION *ax ay az*

The acceleration of the specified particles is set to the values entered here.

1.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

1.4 SEE ALSO

- *Event POP* p. 41
- *Limit POP* p. 65
- *SpeedLimit POP* p. 110
- *State POP* p. 117
- *Velocity POP* p. 132

2 AGE POP

2.1 DESCRIPTION

There are default rules applied to all particles that will age the particle by the amount of time that passes in between cooks. This POP allows the particle to be aged differently from this default rule. However, note that the default rules will still be applied unless they are turned off with the Suppress Rule POP.

2.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

AGE *age*

If enabled, enter the Age you want to assign to the particles here (in seconds).
The current age can be obtained from the $AGE variable. For example, to age the particle by the time in between cooking, use: *$AGE + $TIMEINC* .

LIFESPAN *lifespan*

If enabled, enter the total expected Life you want the particles to have (in seconds). Use the $LIFESPAN variable if you want to base the new lifespan on the existing value.

2.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

2.4 SEE ALSO

- *Suppress Rule POP* p. 124

3 ATTRACTOR POP

3.1 DESCRIPTION

The Attractor POP applies forces on the particles as determined by the referenced Attractor SOP. A SOP can be used as an attractor if it has forces added to it. This can be done by using a Force SOP for metaballs or a Point SOP for face geometry types.

3.2 PARAMETERS – ATTRACTOR PAGE

ACTIVATION *(activate)*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

ATTRACTOR USE

This menu determines how the attractor points affect particles:

All points	All points affect each particle.
Single point per particle	Only one point affects each particle. Normally, all attractor points affect each particle. You can get some interesting effects by enabling this option in order to make each particle assigned to one attractor only. When this option is selected, the particles will head for one attractor each, in the same sequential order of the attractor geometry's points.

Note: The *Property POP* p. 80 can be used to add the attract attribute which specifies which attractor point a particle should be assigned to, rather than using the points in sequence.

GEOMETRY SOURCE

When using *Parameter Values*, this specifies where the POP should get the Object and SOP to stick to, otherwise it allows you to choose which *Context Geometry* to use.

context geometry

All POP networks can have 'Context Geometry'. These context geometries are set in the parameters of the POPNET, the Particle CHOP, or the POP OP. In the POP OP they can also be set by wiring OPs into the inputs of the POP OP. These context geometries can be referenced by the following POPs: Source POP, Collision POP,

SoftBody POP, Attractor POP, Creep POP (All of which have the parameter – *Geometry Source > Use Context Geometry)*. So, instead of hard-coding a specific piece of source geometry or collision geometry, a single POP network can be used by several POP OPs to generate several different outputs, depending on the geometry input into the POP OP.

See *Context Geometry - Example* p. 7.

OBJECT / SOP

Specify the SOP to use as the attractor.

IGNORE TRANSFORM OBJECT

By default, the geometry will automatically be transformed into the space of the object using this POP network. This toggle will cause the geometry not to be transformed.

STOP AT ATTRACTOR

Particles come to a complete stop when they reach the attractor. Note that the particle must come reasonably close to the attractor, or else it will not stop.

SCALE

Scales the entire attractor force acting on the particle.

SHOW ATTRACTOR RADIUS IN GUIDE

Show the radius of affect of each attractor point in the viewport as guide geometry.

SHOW ACCELERATION IN GUIDE

Show the applied acceleration as guide geometry defaults.

3.3 PARAMETERS – DEFAULTS PAGE

IGNORE CHARGE

Ignore the particle's charge in computations. The attractor by definition applies a force that moves the particle towards the attractor point. When charge is taken into consideration, a negative charge will reverse the direction of the force, making the particle move away from the attractor. The magnitude of the charge will also scale the applied force. For a discussion on particle charge, see *Particle Charge* p. 3.

OVERRIDE CHARGE

Override the charge attribute with the value specified below.

CHARGE /charge

When the Override Charge parameter is enabled, this is the charge that is used. For a discussion on particle charge, see *Particle Charge* p. 3.

IGNORE MASS

Ignore particle mass in computations. When mass is used, the applied force is divided by the particle's mass to compute an acceleration. Ignoring the mass means that the force is applied directly as an acceleration.

When mass is used, the particles will accelerate at different rates if the same force is applied. Heavier particles will move more slowly. When mass is ignored, all particles will have the same acceleration applied, regardless of what their mass is.

OVERRIDE MASS

Override the Mass attribute with the value specified below.

MASS /mass

When the *Override Mass* parameter is enabled, this is the mass that is used.

3.4 PARAMETERS – NOISE PAGE

The force applied by the attractor can be modulated by noise. The noise will change the strength of the force but will not change the direction it is applied.

SEED /seed

Specify a Seed for random number generator used by turbulence.

TURBULENCE /turb

The number of iterations to add fractional noise.

ROUGHNESS /rough

Scale of noise added with each iteration. When the turbulence is greater than 0, this parameter controls how much higher frequency terms will contribute to the noise.

EXPONENT /atten

The value entered here determines the Noise attenuation. The resulting noise will be attenuated to the exponent value given. Higher numbers in this field will tend to "attenuate" the noise, making low values lower and high values higher. This has the visual effect of localizing the noise and making it "sharper".

FREQUENCY */freqx /freqy /freqz*

Spatial frequency of noise field. Increased frequency values mean that the noise patters will bunch closer together.

AMPLITUDE */amp*

Maximum value of noise field.

OFFSET */offsetx /offsety /offsetz*

Amount to shift noise.

NOISE TYPE

The noise type to generate. There are four different noise types: *Hermite*, *Improved Hermite*, *Sparse Convolution*, and *Alligator*. Hermite and Improved Hermite are fastest. The Hermite method uses splines to interpolate values in the noise field. The Improved Hermite method uses a more linear interpolation. These methods contain artifacts at certain positions in the noise field. Sparse Convolution is computationally more expensive, but will produce noise fields free of artifacts present in the Hermite methods. Alligator provides a much different look than the other noise types.

3.5 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

3.6 SEE ALSO

- Geometry > *Force OP* p. 584 (you can specify attributes here).
- Geometry > *Point OP* p. 667 (you can specify attributes here).
- *Property POP* p. 80

4 ATTRIBUTE POP

4.1 DESCRIPTION

The Attribute POP allow you to add custom attributes.

The attribute can be of a float, integer or vector type. If the *Local Variable* name is not specified, the attribute name (must be all uppercase) will be used. After adding a user attribute, the local variable can be used anywhere in POPs where local variables are allowed.

4.2 PARAMETERS

ACTIVATION *(activate)*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

NAME

Name of the attribute.

LOCAL VARIABLE

Name of the local variable.

TYPE

The type of variable: Float, Integer or Vector.

SIZE

Number of elements in the attribute.

DEFAULT

Default attribute value.

VALUE *(value1 value2 value3)*

Value to set attribute to.

4.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

4.4 ATTRIBUTE TROUBLESHOOTING

PROBLEM

Tried to use a new attribute, but couldn't change it, only create it.

SOLUTION

The local variables work, but because Houdini allows the creation of attribute arrays, the local variable name ends up being the attribute name followed by an index (even if the attribute created is not an array). So, if you create an attribute *foo*, the local variable will be *FOO1* .

PROBLEM

One might suppose the only way to set a custom attribute downstream is with another Attribute POP. Is there an easier way?

SOLUTION

The Attribute POP allows you to both create and reset attributes. Since we know that an attribute called *foo* maps to *$FOO1*, you can just recreate *$FOO1* with another Attribute POP.

5 COLLECT POP

5.1 DESCRIPTION

The Collect POP performs a merge of the inputs with the exception that the input primitives are not merged into one large particle system, but rather all input primitives remain distinct.

A particle system is stored as a primitive data type in Houdini. Complex simulations may contain many particle primitives. For example, in a fireworks display, the particles shooting up can be one primitive. The explosions are another primitive. Sparkles can make up yet another particle primitive. However, when combined the three primitives will look like one complete simulation.

The Collect POP essentially allows you to group particle primitives together. After sequencing the inputs together, all of the descendent POPs will act on all of the primitives represented in the Collect POP's inputs.

5.2 PARAMETERS

POP INPUTS LISTED IN THE PARAMETER AREA

You can reorder and delete inputs to the Collect POP by clicking on the green up-arrows on the left, or clicking the red "X" buttons to the right.

5.3 SEE ALSO

Ref > Interface > Input Switcher p. 216.

6 COLLISION POP

6.1 DESCRIPTION

The Collision POP performs collision detection between the particles and the referenced SOP geometry.

Use the *Attributes* page to add attributes that store information about the particle's last collision. With the exception of the *distance* attribute, these attributes are not set unless a collision occurs.

6.2 PARAMETERS – OBJECT PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

GEOMETRY SOURCE

Allows you to choose between using the *Parameter Values* or *Context Geometry.*

context geometry

All POP networks can have 'Context Geometry'. These context geometries are set in the parameters of the POPNET, the Particle CHOP, or the POP OP. In the POP OP they can also be set by wiring OPs into the inputs of the POP OP. These context geometries can be referenced by the following POPs: Source POP, Collision POP, SoftBody POP, Attractor POP, Creep POP (All of which have the parameter – *Geometry Source > Use Context Geometry)*. So, instead of hard-coding a specific piece of source geometry or collision geometry, a single POP network can be used by several POP OPs to generate several different outputs, depending on the geometry input into the POP OP.

See *Context Geometry - Example* p. 7.

OBJECT / SOP

Specify the SOP to use as the collision object.

IGNORE TRANSFORM OBJECT

By default, the geometry will automatically be transformed into the space of the object using this POP network. This toggle will cause the geometry not to be transformed.

GEOMETRY MOVES WITH TIME

When enabled, the Collision POP will use the object transforms in its calculations in order to improve collision detection. However, this requires much more computation.

COLLISION TOLERANCE

If set to greater than 0, the Collision POP treats each particle as if it were a sphere of the given diameter, instead of a point in space. This decreases the accuracy of collisions, but greatly improves the chance of catching a collision due to moving or deforming geometry.

6.3 PARAMETERS – BEHAVIOUR PAGE

BEHAVIOUR

Die on Collision	Causes particles to die on collision.
Bounce on Collision	Particles will bounce on collision according to the *Gain Tangent* / *Gain Normal* parameters.
Stop on Collision	The particles will stop on collision, but not lose any of their attributes such as velocity. If the collision object continues to move, the particles will remain stationary and will not follow the collision object.
Stick on Collision	The particles will stick to the collision geometry. If the collision object continues to move, the particles will move with it. This option only works for Mesh, NURBS, and Bezier geometry types.
Continue on Course	Particles will continue as if nothing happened.

COLLISION EVENT

Enter a name for an event to generate when any point collides.

COLLISION GROUP

Group to contain collided points after a collision event. This group will only contain points that collided during that frame.

USE AS HIT ID

How to set the Hit ID attribute:

index	Enter an integer in the Hit Index field which will be used as a unique ID to identify the collision.
index + primitive number	The Hit ID will be set by adding the *Hit Index* field

with the primitive number of the primitive that the particle collided with. For example, if the *Hit Index* is set to 100 and the particle collided with primitive 5, the Hit ID is set to 105. Since primitive numbers are numbered from 0, you can use staggered hit indices to indicate exactly which primitive is hit when there are multiple collision objects. For example, collision object 1 uses hit index 100, collision object 2 uses hit index 200, etc.

HIT INDEX */hitindex*

Enter an integer number here as referenced by *Use as Hit ID*.

OVERSAMPLING */sample*

When the collision geometry is deforming over time, it is possible for the particle to pass through the collision geometry. What happens is that between frames, the collision geometry jumps past the particle. The sampling parameter will adjust the number of time samples to test for collision. This will help with collisions against deforming geometry but note that the operation is expensive. At least 1 sample must be taken.

BOUNCE ACCURACY */bounce*

Because particles are calculated using floating-point frame numbers, it is possible that fast moving particles may collide with an object more than once within the given frame. This parameter determines the maximum number of inter-frame bounces to perform.

GAIN TANGENT / GAIN NORMAL */gaintan /gainnml*

Friction parameters which can be regarded as energy loss upon collision. The first parameter affects the energy loss (gain) perpendicular to the surface. 0 means all energy (velocity) is lost, 1 means no energy is lost perpendicular to surface. The second parameter is the energy gain tangent to the surface.

- 1 and 1 means nothing is lost or gained.
- 0.1 and 1 cause the particles to strike the surface and dribble along it, like rain atop a roof.
- 1 and 0 makes them bounce perpendicular to the surface, no matter what angle they came in at.
- -1 and 1 makes the particle bounce back from whence it came.
- Gains greater than 1 cause energy gain (like a pinball machine bumper).

6.4 PARAMETERS – ATTRIBUTES PAGE

ADD ATTRIBUTE...

This page is used to add attributes to a particle when a collision occurs. The attributes you can add to a particle are:

Add Num Hit Attr	Number of times particle has collided.
Add Hit ID Attr	Hit ID as specified by *Use as Hit ID*.
Add Hit Time Attr	Time of collision
Add Hit Position Attr	Where collision occurred in local space.
Add Hit Position UV Attr	Where collision occurred in parametric UV coordinates.
Add Hit Normal Attr	Normal where collision occurred
Add Hit Diffuse Color Attr	Diffuse color where collision occurred
Add Hit Texture UV Attr	Texture UV where collision occurred
Add Distance Attr	Distance from current position to collision

6.5 ATTRIBUTE NAMES AND LOCAL VARIABLES

Following, are the attribute names associated with each attribute as well as the local variable to access it.

Parameter	Attribute	Local Variable
Add Num Hit Attr	numhit	$NUMHIT
Add Hit ID Attr	hitid	$HITID
Add Hit Time Attr	hittime	$HITTIME
Add Hit Pos Attr	hitpos	$HTX $HTY $HTZ
Add Hit Pos UV Attr	hitposuv	$HU $HV
Add Hit Normal Attr	hitN	$HNX $HNY $HNZ
Add Hit Diffuse Color	hitCd	$HCR $HCG $HCB
Add Hit Texture UV	hituv	$HMAPU $HMAPV $HMAPW
Add Distance Attr	dist	$DIST

6.6 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

Note that only the *Hit Index*, *Gain Tangent* and *Gain Normal* parameters can make use of the local variables.

6.7 SEE ALSO

- *Limit POP* p. 65
- *Softlimit POP* p. 100
- *State POP* p. 117

7 COLOR POP

7.1 DESCRIPTION

The Color POP can be used to change a particle's diffuse color or alpha. There are several mutually exclusive ways to set the color and alpha – by specifying the value directly; via a ramp file; or from a COP.

The POP will add the color (Cd) or alpha (Alpha) attribute if it doesn't exist already.

7.2 PARAMETERS – COLOR PAGE

ACTIVATION *(activate)*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

COLOR PAGE RADIO BUTTONS

Pass	Do not set the color (pass through).
Param	Set the diffuse color of the particles as specified below.
Ramp	Set the diffuse color of the particles using a ramp.
COP	Set the diffuse color of the particles by specifying COP.

COLOR PARAMETERS

diffuse color *(/diffr /diffg /diffb)*

Enter a diffuse color for the particle here as RGB values.

RAMP PARAMETERS

ramp hsv / alpha

As an alternative to specifying the diffuse color via explicit RGB values, you can retrieve a color using the color ramp supplied here. For information on ramps, see the: *Reference Manual > Interface > Color Ramp* p. 217.

lookup *(/colorrampu)*

Because a ramp file contains a range of color values, you need to specify an index to lookup one specific color within the ramp. Specify that index value here. It is a normalized (range = 0 - 1.0) value across the range of the ramp.

The default expression: *$F/$NFRAMES* means the color will change across the range of the ramp over the number of frames in the animation.

COP PARAMETERS

cop / cop network

This parameter allows you to specify a COP Network / COP from which to retrieve the diffuse color for the particles.

u / v */colorcopu /colorcopv*

Because a COP can contain an image using many colors, you need to specify a location within the COP image from where to obtain the color. U and V denote a percentage range (range = 0 - 1.0) along the X and Y axes of the COP image from which to obtain this color.

7.3 PARAMETERS – ALPHA PAGE

ALPHA

Pass	Do not set the alpha (pass through).
Param	Set alpha of the particles as specified below.
Ramp	Set alpha of the particles using a ramp.
COP	Set alpha of the particles by specifying COP.
Speed	Set alpha of the particles according to particle speed.

PARAM PARAMETERS

alpha */alpha*

Enter the Point alpha here. There are many local variables that can be used to set the alpha. For example, to make the particle become more transparent with time, use something like 1 - $LIFE.

RAMP PARAMETERS

ramp / ramp file

As an alternative to specifying the alpha explicitly, you can retrieve the alpha from a ramp file when this parameter is enabled, and by specifying the ramp file here. For information on ramps, see the: *Reference Manual > Interface > Color Ramp* p. 217.

lookup */alpharampu*

Because a ramp file contains a range of alpha values, you need to specify an index to lookup one specific alpha value within the ramp file. Specify that index value here. It is a normalized (range = 0 - 1.0) value across the range of the ramp.

COP PARAMETERS

cop / cop network

Enabling this parameter allows you to specify a COP Network / COP from which to retrieve the alpha for the particles.

u / v */alphacopu /alphacopv*

Because a COP can contain an image using many regions of differing alpha, you need to specify a location within the COP image from where to obtain the alpha. U and V denote a percentage range (range = 0 - 1.0) along the X and Y axes of the COP image from which to obtain the alpha.

ALPHA PARAMETERS

alpha speed

Change point alpha according to speed.

alpha 0 / 1 speed */alpha0speed /alpha1speed*

These two parameters denote the speed at which alpha for the particle is 0 and 1, respectively. The defaults are 100 and 0.

7.4 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

8 CREEP POP

The Creep POP allows you to stick particles to the surface of a piece of geometry. It also allows you to control the U/V position of the particles on the geometry so they can be made to creep across the surface.

The Creep POP lets you change the following attributes:
pstate, *posprim*, *posuv* .

8.1 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Subset of points to act on.

STUCK *stuck*

Values of 1 or more indicate that the particle should be stuck.

GEOMETRY SOURCE

When using *Parameter Values*, this specifies where the POP should get the Object and SOP to stick to, otherwise it allows you to choose which *Context Geometry* to use.

context geometry

All POP networks can have 'Context Geometry'. These context geometries are set in the parameters of the POPNET, the Particle CHOP, or the POP OP. In the POP OP they can also be set by wiring OPs into the inputs of the POP OP. These context geometries can be referenced by the following POPs: Source POP, Collision POP, SoftBody POP, Attractor POP, Creep POP (All of which have the parameter – *Geometry Source > Use Context Geometry)*. So, instead of hard-coding a specific piece of source geometry or collision geometry, a single POP network can be used by several POP OPs to generate several different outputs, depending on the geometry input into the POP OP.

See *Context Geometry - Example* p. 7.

OBJECT / SOP

Object and SOP containing the SOP to stick particles to.

PRIM NUMBER */posprim*

The primitive within the SOP to stick to.

PRIM U/V */posuv1 /posuv2*

U/V position of particle on primitive.

8.2 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9.

9 DRAG POP

9.1 DESCRIPTION

The Drag POP applies drag forces on the particles. Drag is applied in the opposite direction to the velocity. The drag applied is proportional to the particle's speed and drag coefficient.

This POP modifies the following attributes: *accel* .

9.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

APPLY TO LINEAR VELOCITY

Apply drag to the linear velocity of the particles (for RBD).

APPLY TO ANGULAR VELOCITY

Apply drag to the angular velocity of the particles (for RBD).

SCALE *scale*

Scales the entire drag force acting on the particle.

IGNORE MASS

Ignore particle mass in computations. When mass is used, the applied force is divided by the particle's mass to compute an acceleration. Ignoring the mass means that the force is applied directly as an acceleration.

When mass is used, the particles will accelerate at different rates if the same force is applied. Heavier particles will move more slowly. When mass is ignored, all particles will have the same acceleration applied, regardless of what their mass is.

OVERRIDE MASS

Override the Mass attribute with the value specified below.

MASS */mass*

When the *Override Mass* parameter is enabled, this is the mass that is used.

OVERRIDE DRAG */drag*

Override the Particle attribute with the value specified below.

DRAG */drag*

When the Override Drag parameter is enabled, this is the drag that is used.

9.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

9.4 SEE ALSO

- *Property POP* p. 80
- *Wind POP* p. 134.

10 EVENT POP

10.1 DESCRIPTION

The Event POP generates events based on rules. If the rule evaluates to a non-zero value for any particle in the inputs, the event is generated. A rule is any standard Houdini expression. There are many local variables that can be used in the expression.

The generation of an event can be used to trigger operations in other POPs through the Activation field and the *popevent()* function.

For more information on events, see *POP Events* p. 12.

10.2 PARAMETERS

ACTIVATION *(activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering one or more point groups. This subset defines which group of particles to apply the rule to.

EVENT NAME

Enter the name of event to generate in this field. Several named events will be generated if more than one name is entered.

This event is used to trigger action within other POPs via the *Activation* parameter in the desired POP.

PREDEFINED RULES

This menu contains a list of commonly used predefined rules. Among them are rules which generate an event if one of the following has occurred:

- Any particle is dead ($DEAD == 1)
- Any particle is Stopped ($STOPPED == 1)
- Any particle just Collided ($JUSTHIT == 1)
- Any particle has Collided ($NUMHIT > 0)

customizing predefined rules

The list of predefined rules can be customized by copying the file:

```
$HFS/houdini/POPEventRules
```

into your local */houdini* directory (i.e. *$HOME/houdini/*) and modifying the file.

RULE

If this rule evaluates to >0 for any particle in the inputs, the event is generated.

There are many local variables that can be used in the rule.
For example,

```
$TX > 10 && $TX < 15
```

generates the event if any of the input particles has its x position between 10 and 15.

For a list of local variables, see *Particle Attributes and Local Variables* p. 9.

10.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

11 FAN POP

11.1 DESCRIPTION

The Fan POP applies forces on the particles as if a cone-shaped fan were acting upon it. The fan has an origin and applies forces radially from its centre.

11.2 PARAMETERS – FAN PAGE

ACTIVATION */activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

ANGLE */angle*

The number of degrees about the Direction vector for which the fan has it's full intensity.

DELTA */delta*

The angle specified here determines the area outside the *Fan Angle* in which the intensity drops from it's full effect to zero.

ROLLOFF */rolloff*

This parameter specifies the rate of dropoff within the Delta Angle.

MAXIMUM DISTANCE */maxdistance*

A particle farther away than this distance will not be affected by the fan.
A maximum distance of 0 means that all particles are affected.

STRENGTH */strength*

Within the Cone Angle, this is the force applied to each particle. It is scaled from this value to zero within the delta angle. Strength is applied according to the following formula:

$$F = \frac{S}{D^2}$$

Where:
F = Force
S = Strength
D = Distance from origin of Fan

SCALE /scale

Scales the entire fan force acting on the particle.

SHOW FAN IN GUIDE

Show the fan in the viewport as guide geometry.

guide scale

Used to scale the fan guide geometry to a size appropriate to your simulation.

ADD ACCELERATION IN GUIDE

Show the acceleration applied to each particle by the Fan POP as guide geometry.

11.3 PARAMETERS – LOCATION PAGE

REFERENCE

Select from the pop-up menu to determine the Transform Space in which to interpret direction. The options are:

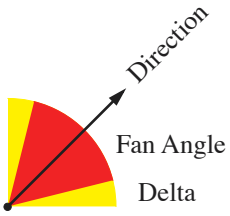
World Space	The Origin is interpreted as a point in world space.
This Object	The Origin is interpreted as a point in the transform object this POP is being cooked from (see *Built-in Merge SOP for POP Object Space* p. 2).
Referenced Object	The Origin is interpreted as a point in the space of the referenced object. In this way, the fan can be attached to a scene object.

OBJECT

Specify the object to use when the above parameter is set to *Referenced Object*.

ORIGIN /originx /originy /originz

Specify the origin of the fan in the space defined by the Reference parameter.



DIRECTION /dirx /diry /dir

Specifies the direction the fan is pointing. The direction is interpreted in the Reference space, and is an XYZ vector.

11.4 PARAMETERS – DEFAULTS PAGE

IGNORE MASS

Ignore the particle's mass in computations.

OVERRIDE MASS

Override the Mass attribute with the value specified below.

MASS *(mass*

When the *Override Mass* parameter is enabled, this is the mass that is used.

11.5 PARAMETERS – NOISE PAGE

The force applied by the fan can be modulated by noise. The fan will always apply the force radially from its origin. The noise will modify the strength of the fan.

Note: For a complete description of the *Noise* page parameters, see the entry: *Parameters – Noise Page* p. 24.

Seed	*/seed* Seed for random turbulence generator.
Turbulence	*/turb* Number of iterations to add fractional noise.
Roughness	*/rough* Scale of noise added with each iteration.
Exponent	*/atten* Noise attenuation.
Frequency	*/freqx /freqy /freqz* Spatial frequency of noise field.
Amplitude	*/amp* Maximum value of noise field.
Offset	*/offsetx /offsety /offsetz* Amount to shift noise.
Noise Type	Type of noise: *Hermite*, *Improved Hermite*, *Sparse Convolution*, and *Alligator.*

11.6 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

Note that only Strength, Scale, Mass, and the Noise parameters make use of the local variables.

11.7 SEE ALSO

- *Attractor POP* p. 22
- *Force POP* p. 51
- *Wind POP* p. 134

12 FOLLOW POP

12.1 DESCRIPTION

The Follow POP applies forces to the particles so that they follow some sort of target, or *Leader*. By copying or instancing a piece of animating geometry to each particle, it is easy to create rendered flocks using this POP.

The second input is used to specify the Leader. If a second input isn't specified, you must define the *Leader Group* in the *Leader* page.

12.2 PARAMETERS – FOLLOW PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on / off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

BEHAVIOUR

Select the type of following behaviour from the pop-up menu:

Normal Follow	Force applied so that for particles of unit mass, the velocity for the next frame is towards the specified position, with a speed equal to that of what it follows.
Accelerating Follow	Force applied is a unit force towards what the particle is following.

MINIMUM DISTANCE *mindistance*

This option only applies if Behaviour is set to *Normal Follow.*
Particles closer than this will be brought rapidly to a stop.

MAXIMUM DISTANCE *maxdistance*

This option only applies if Behaviour is set to *Normal Follow.*
If non-zero, particles further than this distance from followed point will *not* follow.

AMBIENT SPEED */ambspeed*

This option only applies if Behaviour is set to *Normal Follow.*
If the speed of the leader is less than this, the supplied value is used.

The *Ambient Speed* parameter can be used to imitate a normal follow behaviour on leaders which have no velocity. It can also be used to have followers keep moving even if the leader has stopped.

ATTRACT DISTANCE */attdistance*

This option only applies if Behaviour is set to *Accelerating Follow.* If the distance to the followed point is less than this, the force is repulsive; otherwise, it is attractive. This is useful for slowing particles down as they approach the leader.

PREDICT INTERCEPT

Applies only to *Normal Follow Behaviour.* If enabled, the particle will try to predict the leader's direction for the next frame and head it off.

STOP AT LEADER

Particle comes to a complete stop when it reaches the Leader. Note that the particle must come reasonably close to the leader, or else it will not stop.

SCALE */scale*

Scales the entire force acting on the particle.

For normal follow behaviour, it is recommended that you use *Speed Scale* to either speed up or slow down the particles that are doing the following. The applied acceleration will change the velocity direction, so modifying the *Scale* instead of the *Speed Scale* actually changes the particle's direction instead of just the speed.

SPEED SCALE */sscale*

Multiplier of followed object's speed to use (normal follow only).

For normal follow behaviour, only values between 0 and 1 for the *Speed Scale* parameter are useful. The net effect of values less than 1 is a delay in how long it takes the follow particles to achieve the desired velocity. Longer delays and rapidly changing velocities of leaders reduce the follow effect. Scales less than 0 in this Behaviour are not all that useful.

12.3 PARAMETERS – LEADER PAGE

LEADER GROUP

Specify a group to act as the Leader here.

LEADER

Particle Spatial Center — The center of the box bounding of the leaders.

Particle Center of Mass — The centre of mass of the leaders.

Particle Density Center — The centre of mass of the leaders, with all leaders having equal mass.

Individual Particles — Each following particle is assigned one leader to follow. The *Property POP* p. 80 can be used to add the follow attribute which specifies which leader to follow. If this attribute doesn't exist, each follower is assigned a leader in the order that the leaders appear (i.e. if there are 2 leaders, particle 1 – leader 1, particle 2 – leader 2, particle 3 – leader 1, etc.).

Note: The *Property POP* p. 80 can be used to add the *FOLLOW* attribute which specifies which leader to follow. If this attribute doesn't exist, each follower is assigned a leader in the order that the leaders appear (i.e. if there are 2 leaders: particle 1 - leader 1, particle 2 - leader 2, particle 3 - leader 1, etc.).

FOLLOW INDEX

How to interpret the follow index:

Particle ID — The follow index corresponds to the particle ID of the leader.

Particle Order — The follow index corresponds to the order in which the leaders are in the list. If the follow index attribute is not set, the default behaviour is to assign leaders in this order.

12.4 PARAMETERS – DEFAULTS PAGE

IGNORE MASS

Ignore the particle's mass in computations.

OVERRIDE MASS / MASS */mass*

Override particle mass with the value specified.

12.5 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

12.6 SEE ALSO

- *Interact POP* p. 59.

13 FORCE POP

13.1 DESCRIPTION

The Force POP applies a global directional force on the particles (e.g. gravity).

WIND POP VS FORCE POP

So just what is the difference between the Wind POP and the Force POP?

The application of external force directly affects a particles' acceleration, the rate of which is determined by the mass (F = Mass × Acceleration). Wind is an additional force, but one that is velocity sensitive. If a particle is already travelling at wind velocity, then it shouldn't receive any extra force from it. This implies a maximum velocity when using Wind on its own.

An increase in mass impedes acceleration for a given constant force. Drag is a force opposing the direction of motion which is velocity sensitive, i.e. the larger the velocity, the greater the effect of drag. Its useful for limiting the velocity of particles.

13.2 PARAMETERS – FORCE PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

FORCE OBJECT

Name of Force Object to override POP parameter values.

FORCE *forcex forcey forcez*

Magnitude of the Force to apply.

SCALE *scale*

Scales the entire force acting on the particle.

IGNORE MASS

Ignore the particle's mass in computations.

OVERRIDE MASS / MASS */mass*

Override particle mass with the value specified.

13.3 PARAMETERS – NOISE PAGE

The force applied by the pop can be modulated by noise. The noise is added to the global force so its direction and strength are modified.

Note: For a complete description of the *Noise* page parameters, see the entry: *Parameters – Noise Page* p. 24.

Seed	*/seed* Seed for random turbulence generator.
Turbulence	*/turb* Number of iterations to add fractional noise.
Roughness	*/rough* Scale of noise added with each iteration.
Exponent	*/atten* Noise attenuation.
Frequency	*/freqx /freqy /freqz* Spatial frequency of noise field.
Amplitude	*/ampx /ampy /ampz* Maximum value of noise field.
Offset	*/offsetx /offsety /offsetz* Amount to shift noise.
Noise Type	Type of noise: *Hermite*, *Improved Hermite*, *Sparse Convolution*, and *Alligator*.

13.4 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

13.5 SEE ALSO

- *Attractor POP* p. 22
- *Fan POP* p. 43
- *Wind POP* p. 134

14 GROUP POP

14.1 DESCRIPTION

The Group POP generates groups of particles according to various criteria. These groups can then be used to apply POPs selectively to specific particles.

Note that to choose particles based on a particular criteria, the Enable check button for that option must be on.

You can also create a group based on the instance attribute particles.

14.2 PARAMETERS – CREATE PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

GROUP NAME

Name of new group to contain chosen points.

PRESERVE GROUP

Append particles to group instead of clearing group first.

RULE – ENABLE

Choose based on rule. For example, when a particle is dead, collided, etc.

predefined rules

This menu contains several predefined rules which simply enter the following expressions into the *Rule* field:

- Particle is Dead ($DEAD == 1)
- Particle is Stopped ($STOPPED == 1)
- Particle is just Collided ($JUSTHIT == 1)
- Particle has Collided ($NUMHIT > 0)

customizing predefined rules

The list of predefined rules can be customized by copying the file:

```
$HFS/houdini/POPKillRules
```

into your local */houdini* directory (i.e. *$HOME/houdini/*) and modifying the file.

rule

If this rule evaluates to 1, the point will be chosen for inclusion in the group. There are many local variables that can be used in the rule. For example:

```
$AGE > 1 && $AGE < 3
```

selects all particles whose age is between 1 and 3 seconds.

```
$ORIGIN == 1
```

selects all particles whose origin ID is 1.

For a list of local variables, see *Particle Attributes and Local Variables* p. 9.

BOUNDING – ENABLE

Choose based on point being in bounding volume. Particles inside the bounding volume are added to the group.

bounding type

Type of bounding volume to use. The available options are:

- Bounding Box
- Bounding Sphere
- Bounding Object
- Bounding Metaball

center */tx /ty /tz*

Center of bounding volume.

size */sizex /sizey /sizez*

Size of bounding volume.

object / sop

Object / SOP containing geometry to use as bounding volume.

ignore transform object

By default, the geometry will automatically be transformed into the space of the object using this POP network. Enabling this parameter causes the geometry not to be transformed.

density minimum */densitymin*

When selecting based on a metaball volume, points whose density is below this minimum threshold will not be chosen.

GENERATOR – ENABLE

Choose based on which generator created the particles.

generators

Particles created by the generators specified will be chosen.
More than one generator can be specified.

RANDOM – ENABLE

Place particles randomly into *n* groups.

of groups *[/multi]*

Number of groups to create.
The groups names will be based on the group parameter.

For example, if the Group name is "Flock" and 5 groups are requested, the POP will create groups Flock1, Flock2, Flock3, Flock4 and Flock5 and the particles will be randomly distributed into the 5 new groups.

INSTANCE – ENABLE

Allows you create a group based on the instance attribute particles.

instances

Enter names of object instances here.

14.3 PARAMETERS – COMBINE PAGE

This page allows you to combine different groups based on specific boolean operators and combination methods.

The following operations are possible:

Negation; *Union*; *Intersect*; *Exclusive Or*; *Subtract*

- In the first field, enter a new group, or assign the result to an existing group.
- The button in the centre column takes the negation of the group to the right of it.
- The third field contains the Name of combined group.

14.4 PARAMETERS – SELECT PAGE

This page allows a group to be the selection in the Viewport.

SELECT GROUP(S) IN DETAIL

Use select group as the selection in the detail. Note that any selections already made will be replaced by this selection.

SELECT GROUP

Group to select.

14.5 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

15 INSTANCE POP

15.1 DESCRIPTION

The Instance POP instances geometry on a particle.

The render flag output of the object is instanced on the particle. Note that the geometry will only appear when rendered with mantra or RenderMan. The geometry will not be shown in the viewport.

Note: When you render the geometry, you can not use the "View: xxx" render commands to render instanced geometry. Use one of the regular mantra output drivers displayed in the Render icon's pop-up menu which are set in the Output Editor.

The template flag shows the instanced geometry as guide geometry in the Viewport.

15.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

PARTICLE GEO (PARTICLE INSTANCING)

The geometry object specified here will be "instanced" to each particle. Rather than making copies of the geometry at each particle, this provides a light-weight (less memory usage) method of adding complexity to the scene.

This parameter allows you the most efficient way to animate many copies of geometry with a particle system. Your geometry will be instanced to each of the particles in the system without consuming additional memory for each of the copies.

The particle axis controls how the instanced geometry will be oriented on the particle. The axis is determined by the particle's up vector (Up Vector POP) and applied rotation (Rotation POP).

ORIENTATION

How the instanced geometry will be oriented relative to the particle. Note that any rotation specified by the Rotation POP will be applied on top of this orientation.

Particle axes (vel and up vector)
: If there is no applied rotation, the instanced geometry will have its Z axis oriented along the particle's velocity and its Y axis will point toward the up vector. This is useful when the geometry should always face the direction the particle is travelling.

Fixed axis
: With no additional rotation applied, the instanced geometry will always be aligned with a fixed axis. This is useful when performing collisions. For example, if a particle bounces off a surface, the instanced geometry's orientation should remain the same and shouldn't flip around because the velocity has suddenly changed.

15.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

15.4 SEE ALSO

- *Rotation POP* p. 95
- *Up-Vector POP* p. 130
- Ref > Objects > Geometry Object > *Point Instancing* p. 283
- Ref > Geometry > *Point OP* p. 667

16 INTERACT POP

16.1 DESCRIPTION

The Interact POP is used to apply inter-particle forces. This allows particles to be attracted and/or repelled by other particles. The roll-off function for the force is a half-cosine stretched over the radius of effect.

When two particle interact, both of their charges are multiplied to scale the applied forces. This allows some particles to attract while others repel. By default, a positive force is a repulsive force.

Keep in mind that if you are using *Particle Intersects Effect*, then one particle may be within the effect radius of another, but not vice versa. If both particles do happen to be within the effect radius of each other, then a particle's influence on the other particle is the same as the relative force applied back to it.

16.2 PARAMETERS – INFLUENCE PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

PARTICLE RADIUS

The radius of the particle.

Use Particle Scale / Instance Size
: Radius is that of a sphere centered at the origin, large enough to envelop the instanced geometry object, scaled by the absolute value of the particle scale.

Use Supplied Radius
: Use the supplied radius.

SCALE *pscale*

Multiplier for particle radius.

RADIUS *pradius*

Radius to use for particle radius when set to "Use Supplied Radius".

EFFECT RADIUS

The extent of the effect from the particle's surface. It is best to think of the *Effect Radius* as a buffer extended from the particle "sphere" which is determined by Particle Radius.

Use Particle Radius Use the particle radius.

Use Particle Scale / Instance Size
The radius is that of a sphere centered at the origin, large enough to envelop the instanced geometry object, scaled by the absolute value of the particle scale.

Use Supplied Radius Use the supplied radius.

SCALE *escale*

Multiplier for effect radius.

RADIUS *eradius*

Radius to use for effect radius when set to "Use Supplied Radius".

16.3 PARAMETERS – BEHAVIOUR PAGE

OVERLAP BEHAVIOUR

This menu determines what happens when two particles overlap:

No Force No force is applied.

Random Direction The force is applied in a random direction.

INFLUENCE TYPE

How the effect radius is used:

Particle Intersects Effect Particle exerts force when another particle enters the shell of influence.

Effect Intersects Effect Particle exerts force when another particle's shell of influence enters its own shell of influence.

CHARGE BEHAVIOUR

Determines which charge will affect the force:

Use both charges If particle A affects particle B, then the charge of both A and B are used to scale the applied force.

Use charge of affecting particle
If particle A affects particle B, then the charge of A will be used to scale the applied force.

Use charge of particle being affected
If particle A affects particle B, then the charge of A will be used to scale the applied force.

ROLLOFF EXPONENT */exponent*

Exponent to apply to the roll-off function. Smaller values increase the effect at any given distance with the radius.

FORCE MULTIPLIER */multiplierx /multipliery /multiplierz*

Multipliers for each component of the force. This is useful to remove a particular component (i.e. set Muliplier -Z to 0 to not apply forces in the Z direction).

16.4 PARAMETERS – DEFAULTS PAGE

IGNORE CHARGE

Ignore the particle charge.

OVERRIDE CHARGE / CHARGE */charge*

Override the particle charge. If the *Override Charge* parameter is enabled, it will use the value supplied here instead.

IGNORE MASS

Ignore particle mass.

OVERRIDE MASS / MASS */mass*

If the *Override Mass* parameter is enabled, it will use the value supplied in Mass instead.

16.5 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

16.6 SEE ALSO

- *Follow POP* p. 47
- *Property POP* p. 80

17 KILL POP

17.1 DESCRIPTION

The Kill POP kills particles based on rules. If the rule evaluates to a non-zero value for any particle in the inputs, the particle is killed. An event can be generated if any particles are killed. In addition, all killed particles can be added to a point group.

17.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

PREDEFINED RULES

This menu contains a list of commonly used predefined rules.
Among them are rules which will kill a particle that meets these criteria:

Menu	Rule
Kill all particles	1
Kill stopped particles	$STOPPED == 1
Kill just collided particles	$JUSTHIT == 1
Kill any collided particles	$NUMHIT > 0

customizing predefined rules

The list of predefined rules can be customized by copying the file:

```
$HFS/houdini/POPKillRules
```

into your local */houdini* directory (i.e. *$HOME/houdini/*) and modifying the file.

RULE

If this rule evaluates to 1, the particle will be killed.
There are many local variables that can be used in the rule. For example:

```
$ID == 50
```

kills the particle with the ID equal to 50.

For a list of local variables, see *Particle Attributes and Local Variables* p. 9.

EVENT

Enter a name for an event to generate when any particle is killed as a result of this POP.

GROUP

Specify the Group to contain killed points here.

17.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

17.4 SEE ALSO

- *State POP* p. 117

18 LIMIT POP

18.1 DESCRIPTION

The Limit POP determines when the particles have reached a boundary limit.

18.2 PARAMETERS – LIMITS PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

TYPE

This pop-up menu determines the Type of boundary limit. The choices are:

Box The +/- ranges specify six limit planes.

Sphere The +/- ranges specify the extents of a sphere.

+ LIMIT / - LIMIT */limitposx, y, z /limitnegx, y, z*

These parameters define the X, Y, and Z boundaries within which particles may exist. Particles that hit the boundary are dealt with as specified in the *Behaviour* page.

18.3 PARAMETERS – BEHAVIOUR PAGE

BEHAVIOUR

Die on Collision Causes particles to die on collision.

Bounce on Collision Particles will bounce on collision according to the *Gain Tangent* / *Gain Normal* parameters.

Stop on Collision The particles will stop on collision, but not lose any of their attributes such as velocity. If the collision object continues to move, the particles will remain stationary and will not follow the collision object.

Continue on Course Particles will continue as if nothing happened.

COLLISION EVENT

Enter a name for an event to generate when any point collides.

COLLISION GROUP

Group to contain collided points after a collision event. This group will only contain points that collided during that frame.

HIT INDEX */hitindex*

Enter an integer number here to be used as the Hit ID.

BOUNCE ACCURACY */bounce*

Because particles are calculated using floating-point frame numbers, it is possible for fast moving particles to collide with an object more than once within the given frame. This parameter determines the maximum number of inter-frame bounces to perform.

GAIN TANGENT / GAIN NORMAL */gaintan /gainnml*

Friction parameters which can be regarded as energy loss upon collision. The first parameter affects the energy loss (gain) perpendicular to the surface. 0 means all energy (velocity) is lost, 1 means no energy is lost perpendicular to surface. The second parameter is the energy gain tangent to the surface.

- 1 and 1 means nothing is lost or gained.
- 0.1 and 1 cause the particles to strike the surface and dribble along it, like rain atop a roof.
- 1 and 0 makes them bounce perpendicular to the surface, no matter what angle they came in at.
- -1 and 1 makes the particle bounce back from whence it came.
- Gains greater than 1 cause energy gain (like a pinball machine bumper).

18.4 PARAMETERS – ATTRIBUTES PAGE

ADD NUM HIT ATTR

The *Num Hit* attribute records the number of times the particle has collided.

ADD HIT ID ATTR

This adds a Hit ID to the particle. This is useful for tracking which object particles have collided with a unique ID number.

ADD HIT TIME ATTR

Enabling this parameter adds an attribute to the particle which records the Time of collision.

ADD HIT POSITION ATTR

Adds an attribute which records where the collision occurred.

ADD HIT NORMAL ATTR

Adds an attribute which stores the surface normal where the collision occurred.

ADD DISTANCE ATTR

Adds an attribute which computes the distance from the current position to where the collision occurred.

18.5 ATTRIBUTE NAMES AND LOCAL VARIABLES

Following, are the attribute names associated with each attribute as well as the local variable to access it.

Parameter	Attribute	Local Variable
Add Num Hit Attr	numhit	$NUMHIT
Add Hit ID Attr	hitid	$HITID
Add Hit Time Attr	hittime	$HITTIME
Add Hit Normal Attr	hitN	$HNX $HNY $HNZ
Add Distance Attr	dist	$DIST

LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

Note that only Hit Index, Gain Tangent and Gain Normal can make use of the local variables.

18.6 SEE ALSO

- *Collision POP* p. 29
- *Softlimit POP* p. 100

19 LINEAGE POP

19.1 DESCRIPTION

The Lineage POP is used to change attributes associated with the particle's geneology or family tree. These characteristics are normally set for you when you birth or split particles, but you can override the values with this POP.

A particle can have the following genealogical attribtues:

Name	Attribute	Local Variable	Description
ID	id	$ID	A unique number attached to the particle at birth. Like a Social Security Number.
Parent	parent	$PARENT	The ID number of the parent that the particle split from.
Origin	origin	$ORIGIN	This is the original source from where the particle and its ancestors were born.
Generation	gen	$GEN	How many generations of splitting have occurred since the Origin.

** These attributes are available when added by the Source POP.*

19.2 PARAMETERS

ACTIVATION

The *Activation* field turns the entire POP on and off. If the contents of the field evaluate to 1, the POP will be turned on. The POP will also be on if the field is left empty. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

ID *id*

Enter a new unique ID identifier. This is like changing the particle's social security number. This should be an integer value.

ORIGIN */origin*

Enter a new particle Origin ID identifier. Altering this attribute will make the particle seem like it was originally born from a different piece of geometry.

PARENT */parent*

Enter a new particle parent ID. This is like saying the particle had different parents than it actually did.

GENERATION */gen*

How many times particle has split. Altering this attribute will not change how old the particle is, but will change how many generations old a particle is. For example, a third generation particle (i.e. a grandchild) can be made into a second generation particle (i.e. a direct child).

19.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

19.4 SEE ALSO

- *Source POP (Generator)* p. 102
- *Split POP (Generator)* p. 111
- *Stream POP (Generator)* p. 119

20 LOCATION POP

20.1 DESCRIPTION

The Location POP births particles from a location in space.

20.2 PARAMETERS – LOCATION PAGE

*Local variables are usable in parameters denoted with a *.*

SYSTEM

Cartesian Specify the location with X, Y and Z coordinates.

Cylindrical Specify the location with Radius, Theta and Z coordinates (where theta is the angle between the X and Y axes).

Polar Specify the location with Radius, Theta and Phi coordinates (where Phi is the angle between the X and Z axes).

COORDINATES * */loc*

Enter birth location here, according to the type of coordinates you have choosen.

20.3 PARAMETERS – BIRTH PAGE

IMPULSE ACTIVATION */impulseactivate*

Turns impulse birthing on and off.
Impulse birthing results in all particles being born at once.

IMPULSE BIRTH RATE */impulserate*

Number of particles to birth per frame.
Birth rate interpreted as exact number of particles to birth.

CONSTANT ACTIVATION */constantactivate*

Turns constant birthing on and off. This type of birthing results in continuous birthing while the POP is active as determined by the *Constant Activation* field. Birth rate is in particles-per-second.

CONSTANT BIRTH RATE */constantrate*

How many particles to birth per second.

BIRTH GROUP

If you would like the birthed points to be defined within a group, specify a group to contain birthed points here.

PRESERVE GROUP

Add particles to group instead of clearing group first.

LIFE EXPECTANCY */life*

How long each particle will live, in seconds. The default is 100 seconds, which is just over a minute and a half. Use small values such as one or two to see the effect.

LIFE VARIANCE */lifevar*

Variance of a particle's life expectancy in seconds. If life expectancy is one second, and the variance is zero seconds, each particle will live exactly one second. If variance is set to *0.5*, then some particles will live only a half second, while others will live a second and a half. The rest will live some time in-between. This randomness gives a more natural look to the particle births.

20.4 PARAMETERS – ATTRIBUTES PAGE

*Local variables are usable in parameters denoted with a *.*

VELOCITY * */velx /vely /velz*

Used to set or add to velocity.

VARIANCE * */varx /vary /varz*

Variance of set or added velocity.

ELLIPSOIDICAL VARIANCE

Allows you to birth particles with random directions that are evenly distributed in an ellipsoid (whose size is determined by *Variance*) about the base velocity, rather than a cube.

ADD ID / GENERATION / ORIGIN / SPEED ATTRIBUTE(S)

These parameters allow you to add the following attributes:
ID and Parent, Generation, Origin, and Speed to the particles.

You can set the *Origin Index* using the parameter (below).

This speed attribute is calculated automatically and is equal to:
sqrt(VX^2+VY^2+VZ^2) . It can be referenced through the local variable SPEED.

ORIGIN INDEX * */originindex*

Index to be used when adding the *Origin* attribute.

20.5 LOCAL VARIABLES

BBX BBY BBZ	Bounding box.
TX TY TZ	Particle position.

21 LOOKAT POP

21.1 DESCRIPTION

The Lookat POP applies a rotation on the particle's axis so that it's Z axis faces a given direction.

This is useful for doing things like sprites. It ensures that the textured faces will always be oriented towards the camera.

21.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

LOOKAT MODE

How the lookat will work:

Lookat plane Look at plane whose normal is defined by the direction.

Lookat point Look directly at the point specified.

REFERENCE

Select from the pop-up menu to determine the Transform Space in which to interpret direction. The options are:

World Space The specified direction is interpreted as a vector in world space.

This Object The specified direction is interpreted as a vector in the transform object this POP is being cooked from. (see *Built-in Merge SOP for POP Object Space* p. 2)

Referenced Object The specified direction is interpreted as a vector in the space of the referenced object. In this way, particles can be made to look in the same direction as any arbitrary object – like a camera.

OBJECT

Specify the object to use when the above parameter is set to *Referenced Object*.

DIRECTION */dirx /diry /dirz*

Specifies a vector for the lookat direction. It is interpreted in the Reference space.

Example: The vector's origin is at 0, 0, 0 in the Reference space.
A direction of (0, 0, 1) specifies a vector pointing down the Z axis.

21.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

21.4 SEE ALSO

- *Instance POP* p. 57
- *Rotation POP* p. 95
- *Up-Vector POP* p. 130

22 ORBIT POP

22.1 DESCRIPTION

The Orbit POP applies forces to the particles such that they orbit about a center. To help visualize the effect of this operator, turn on the template flag to view the acceleration guides.

This POP modifies the following attribute: *accel*.

22.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

OVERRIDE AXIS

Override for the orbit axis attribute.

AXIS *axisx axisy aaxisz*

Normal to the orbit plane.

OVERRIDE RADIUS

Override for the orbit radius attribute.

RADIUS *radius*

Radius of orbit in units.

OVERRIDE SPEED

Override for the orbit speed attribute.

SPEED *speed*

Speed of the orbit in RPM.

SCALE */scale*

Scaling factor that controls how fast particles move into their proper orbit.

22.3 PARAMETERS – CENTER PAGE

CENTER GROUP

Subset of points to act as centers.

CENTER TYPE

Specifies what is used for the centre of the orbit:

Particle Spatial Center	The center of the box bounding the center particles.
Particle Center of Mass	The center of mass of the center particles.
Particle Density Center	Corresponds to the center of mass with all particles having equal mass.
Center Point	Center Point co-ordinates as specified.
Individual Particles	Each orbitting particle is assigned one center particle to orbit around.

OVERRIDE CENTER POINT

Override for the orbit center attribute

CENTER POINT */cntrlocx /cntrlocy /cntrlocz*

Point to orbit around.

ORBIT INDEX

How to interpret the orbit index:

Particle ID	The orbit index corresponds to the particle ID of the center particle.
Particle Order	The orbit index corresponds to the order in which the center particles are in the list. If the orbit index attribute is not set, the default behaviour is to assign center particles in this order.

22.4 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

22.5 EXAMPLES

See demos in: *$HD/OPspecific/POPs/Orbit/*

22.6 SEE ALSO

- *Property POP* p. 80

23 POSITION POP

23.1 DESCRIPTION

The Position POP can be used to explicitly set a particle's position. The position can be evaluated either in component form or as a vector.

There are default rules applied to all particles that will move them from frame to frame. This POP allows the particle to be positioned directly. However, note that the default rules will still be applied unless they are turned off with the Suppress Rule POP.

23.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

APPLY MOTION

Specifies what types of points to apply the position to:

to all points	Apply the position to all of the particles.
to moving points	Apply the position only to those particles which are in motion. This means that stopped and dying particles will not be affected.

PARAM OPTIONS

position */tx /ty /tz*

Select from either *Vector* (a formula) or *Component* (X, Y, Z) definition, and enter the particle position in the following edit field(s). See *Vector vs Component Definition* p. 2 for a discussion on differences between component and vector definition.

The current position can be obtained from the $TX, $TY and $TZ variables and used within your definition. For example, to move the particle by its velocity scaled by the Time Increment (since velocity is distance-per-second), use the following *Vector* expression:

```
vector3($TX, $TY, $TZ) + $TIMEINC * vector3($VX, $VY, $VZ)
```

PATH OPTIONS

object / sop

Path makes the particles follow a path (like the Path Object in the Object Editor). Specify the *Object* and *SOP* for the object you want to be the path object.

ignore transform object

Transform geometry to transform object space.

position */pathpos*

Position along the path in the range of [0..1], where 0 = starting point, and 1 = end point. Higher/lower values allow multiple trips along the path.

end option

Cycle/reverse at the end of the path.

REFERENCE OPTIONS

object

Object gives a position in the object space of a specific object.
Specify the object from which the position should be taken here.

offset type / position */tx /ty /tz*

Select from either *Vector* (a formula) or *Component* (X, Y, Z) definition, and enter the particle position in the following edit field(s). The positions will be offset by the values in the Position parameter of the POP. See *Vector vs Component Definition* p. 2 for a discussion on differences between component and vector definition.

23.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

23.4 SEE ALSO

- *State POP* p. 117
- *Suppress Rule POP* p. 124
- *Velocity POP* p. 132

24 PROPERTY POP

24.1 DESCRIPTION

The Property POP can be used to change a particle's physical properties (mass, drag or charge, which attractor point to move towards), orbit, and other properties. By itself, this POP will not affect particle behaviour. The attributes are only useful when used in combination with some of the modifier POPs like Force, Wind, and Drag. The POP will add the attribute if it doesn't exist already.

This POP modifies the following attributes: *attract*, *charge*, *drag*, *follow*, *mass*, *oaxis*, *ocenter*, *ocntrid*, *oradius*, *ospeed* and *pscale*.

24.2 PARAMETERS – PHYSICAL PAGE

ACTIVATION *(activate)*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

MASS *(mass)*

Enter values for the particle's Mass.
Mass scales the force according to the following formula:

$$F = M \times A$$

Where:
F = Force
M = Mass
A = Acceleration

CENTER OF MASS

Relative position of particle center of mass. Specifies the location of the objects center of mass. This value is specified as a displacement in object space from the origin (0,0,0) of the object. The rigid body particle representing an object is always placed at the origin of the object, which may not be the real center of the geometry. In this case you would specify a center of mass so that the particle behaves as if it's center of mass is at the center of the geometry.

BOUNCE

Bounce coefficient of the object. When two objects collide, the product of the bounce coefficients of the objects determines the energy lost to the collision (and thus the velocities of the object after the collision).

DYNAMIC FRICTION

Coefficient of dynamic friction. When two objects are in contact and sliding across each other, the product of their dynamic friction values determines the force of friction acting against the sliding motion.

STATIC FRICTION

Coefficient of static friction. When two objects are in contact and relatively at rest, the product of their static friction values determines the amount of force required to induce a relative (sliding) motion of the objects.

24.3 PARAMETERS – ORBIT PAGE

These parameters work in conjunction with the *Orbit POP* p. 75.

AXIS */oaxisx /oaxisy /oaxisz*

The normal to the orbit plane.

CENTRE */ocenterx /ocentery /ocenterz*

The center of the orbit.

RADIUS *oradius*

The radius of the orbit in units.

SPEED */ospeed*

The speed of the orbit in RPM (rotation per minute).

ORBIT INDEX */ocntrid*

The ID of the point in the "Center Group" to be used as the center of orbit.

24.4 PARAMETERS – MISC PAGE

FOLLOW INDEX */follow*

Which particle to follow.

ATTRACTOR POINT */attract*

Attractor point to move towards.

CHARGE */charge*

Enter values for the particle's ± Charge.
See *Particle Charge* p. 3 for an explanation of the effect of charge.

SCALE */scale*

Scale applies a uniform scale to the instanced particle geometry.

DRAG */drag*

Enter values for the particle's drag.
Drag applies a force according to the following formula:

$$F = D \times (W-V)$$

F = Force
D = Drag
W = Wind
V = Velocity

24.5 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

In addition to the standard local variables, the following are available in this POP only:

FOLLOW *	Leader to follow.
ATTRACT *	Attractor point.
COMX, COMY, COMZ	Centre of Mass X, Y, Z.
BOUNCE	Bounce coefficient – energy lost due to collision.
FSTATIC / FDYNAMIC	Amount of Static and Dynamic friction.

24.6 SEE ALSO

- *Attractor POP* p. 22
- *Drag POP* p. 39
- *Fan POP* p. 43
- *Follow POP* p. 47
- *Force POP* p. 51
- *Instance POP* p. 57
- *Interact POP* p. 59
- *Orbit POP* p. 75
- *Wind POP* p. 134

25 PROXIMITY POP

The Proximity POP can be used keep track of the other particles in the proximity of the given particles. This POP modifies the following attributes: *nearest*, *nearestdist* and *numproximity.*

25.1 PARAMETERS

ACTIVATION */activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Subset of points to act on.

ADD NEAREST NEIGHBOUR ATTRIBUTE

The id of nearest particle.

ADD NEAREST DISTANCE ATTRIBUTE

Distance to nearest particle

ADD NUMBER IN PROXIMITY ATTRIBUTE */radius*

Number of particles within proximity.

CREATE PROXIMITY PARTICLE GROUPS

Create point groups of particles that fall within a given radius of source particles

create groups for

Particles to use to create proximity groups

group name root

Root name of groups to be created. The group names will be made by appending the ID or point number of the source particle.

INCLUDE SELF IN PROXIMITY GROUPS

Specifies whether or not to include the source particle in the proximity group.

PROXIMITY RADIUS */proximityradius*

Radius within which other particles are considered in proximity.

USE POINT NUMBERS INSTEAD OF IDS

This parameter also tells the '*nearest*' attribute to contain the point number of the nearest particle instead of the ID, and create the proximity group names.

Tip: This is useful within the *poppoint()* expression function.
For example: *poppoint(poppoint($PT, "nearest", 0), "foo", 0)*

25.2 LOCAL VARIABLES

The standard local variables are usable in this POP. These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

Note these additions:

NEAREST	Nearest neighbour.
NEARDIST	Distance to nearest neighbour.
NUMPROXIMITY	Number or particles within a given radius.

26 RENDER POP

26.1 DESCRIPTION

Use the Render POP to change the particle's rendering characteristics. It allows you to specify the geometry used to render particles.

26.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

PARTICLE TYPE

Though particles are displayed in the Viewport as streaks with a greater brightness at the head of the particle, this parameter controls how the particle system should be rendered:

Spheres	Each particle is rendered as a sphere.
Disks	Each particle is rendered as an oriented disk.
Lines	Each particle is rendered as a line connecting the start and end points between the particle's position on subsequent frames.
Tubes	Each particle is rendered as an open tube.
Capped Tubes	Each particle is rendered as a capped tube.
Rounded Tubes	Each particle is rendered as a rounded tube.

Tip: If you are outputting to RenderMan, set your particle type to *Render as Disk* for RiPoint and *Render as Line* for RiCurve.

PARTICLE SIZE *prsize*

How big the particles are. For spheres, disk and tubes this will be the radius.

PARTICLE BLUR *prblur*

This determines how long the particles will appear when rendered. For end to end connectivity, this should be set to *1/$FPS*. However, it is possible to stretch or shrink the particles by respectively increasing or decreasing this value. The view will update to display these changes.

SPHERE NORMAL

When rendering as disks, fake the surface normal to approximate a sphere.

renderer support

Both *mantra* and RenderMan support some, but not all of the above render types:

Primitive Type	mantra	RenderMan
Sphere	•	•
Disks	(as spheres)	•
Lines	(as tubes)	•
Tubes	•	•
Capped Tubes	•	•
Rounded Tubes	•	•

This table is only in regards to particle renderings. When something doesn't match, the particles will still be rendered, but not necessarily what you'd expect.

Note: When rendering disks with *mantra*, the sphere normal is always used.

27 RBD CONSTRAIN POP

27.1 DESCRIPTION

The RBD Constrain POP applies forces to rigid body particles to maintain a constraint, such as keeping a particle at a fixed location in space, or keeping one particle at a fixed location relative to another particle.

27.2 PARAMETERS

ACTIVATION *activate*

Turns the POP on and off.

FORCE OBJECT

Name of Force Object to override POP parameter values.

GROUP A

Particles to use as the first object in the constraint.

OBJECT A

Use particle with this value for its instance attribute as the first object in the constraint.

GROUP B

Particles to use as the second object in the constraint.

OBJECT B

Use particle with this value for its instance attribute as the second object in the constraint.

CONSTRAINT TYPE

The type of constraint to apply.

POINT TO NAIL

nail point */nailptx /mailpty /nailptz*

Point in space to keep the particle.

object point */nailobjx /nailobjy /nailobjz*

Offset from particle centre of mass to keep located at the nail point.

POINT TO POINT

object a point */ptobjax /ptobjay /ptobjaz*

Offset from object A center of mass to connect to object B.

object b point */ptobjbx /ptobjby /ptobjbz*

Offset from object B center of mass to connect to object A

ROTATION

rotation vector */rotvecx /rotvecy /rotvecz*

Vector in world space to align with object vector.

object vector */rotobjx /rotobjy /rotobjz*

Vector in object space to align with rotation vector.

27.3 LOCAL VARIABLES

The standard local variables are usable in this POP. These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

28 RIGID BODY (RBD CREATE) POP

28.1 DESCRIPTION

The key to creating rigid body simulations in Houdini is the RigidBody POP. This POP lets you create rigid body particles, and set relevant properties for these particles. It can also take input from another POP to convert existing particles into rigid body particles. A rigid body particle differs from other particles in several ways.

- A rigid body particle has a number of attributes which other particles may or may not have. These attributes are use by the rigid body solver (such as torque, static and dynamic friction, center of mass, instance).

- Rigid body particles can perform particle-particle collision detection using the instance attribute of the particle to determine the geometry to use for the collision detection. This collision detection changes the angular and linear velocities of the colliding particles to simulate collisions between ideal rigid bodies, taking into account bounce, relative masses and inertial tensors to create realistic motions.

- Rigid body particles are controlled by a different solver than other particles. The rigid body solver does not allow the suppression of rules (using the Suppress POP). This is because to generate realistic motion and avoid object interpenetration, the solver must be allowed to control all aspects of the particle's motion.

  Rigid body particles, however, do react to subsequent POPs the same way as other particles. If you append a Force POP to a network that contains both rigid body and normal particles, both types of particles will respond to the force in the same way (assuming similar masses on the particles).

PASSIVE RIGID BODIES

If a rigid body is active, or it is passive and has less than infinite mass (an object is considered to have infinite mass if its mass is greater than 1e+38, or its mass is exactly 0.0), its motion is controlled by the RBD solver. The RigidBody POP creates the particle and sets the initial velocity, angular velocity, rotation and position, and then doesn't affect the particle after that. If however the rigid body is passive (the Object's Active parameter is 0, or the particle Active state was set to 0 with the State POP) and has a mass greater than 1e+38 (or a mass equal to 0.0), the RigidBody POP treats it differently. The RBD Solver in this case does not move or change the velocity of the particle. The RigidBody POP looks at the keyframed positions and rotations of the object and sets the particle attributes appropriately at each frame. Thus you can have rigid body particles that are actually keyframe animated, but will cause reactions in other rigid body particles. Pairs of keyframed rigid body particles are not tested for intersection because neither object would be affected by the collision.

28.2 PARAMETERS – BIRTH PAGE

GENERATION OBJECTS

This parameter contains a list of objects. For each object in the list, a single particle is birthed at the origin of the object. The particle instance attribute is set to the object that generated it.

BIRTH TIME

Time at which to birth particles for the Generation Objects.

BIRTH GROUP

Group to contain birthed points

PRESERVE GROUP

Add particles to group instead of clearing group first

LIFE EXPECTANCY */life*

How long the particle will live in seconds.

LIFE VARIANCE */lifevar*

Variance of life expectancy in seconds.

28.3 PARAMETERS – RIGID BODY PAGE

ACTIVATION */activate*

Turns the POP on and off.

COLLISION EVENT

Event triggered by rigid body collisions

COLLISION GROUP

Group of colliding rigid body particles

PRESERVE COLLISION GROUP

Do not remove entries from collsion group after each frame

EXCLUSIVE GROUPS

Names of groups to be used as RBD particle collision groups

EXCLUSIVE GROUPS

The rigid body solver provides a very flexible mechanism for specifying which rigid body particles should be tested for collision. Simply specify the names of existing point groups in this field. All pairs of rigid body particles that share membership in one or more of these point groups are tested for collision. Group names entered in this field are global and apply to all rigid body particles in the POP network, not just the particles generated by this RigidBody POP. If no exclusive groups are set in the network, all rigid body particles do collision detection with all other rigid body particles.

MAKE INPUT PARTICLES RIGID BODIES

If this check box is off, input particles to this POP are not affected in any way. This ability is useful if you simply want to specify collision groups without actually changing any particles. The parameters of the first page (Generation Objects) are still created if this parameter is off.

SOURCE GROUP

If "Make Input Particles Rigid Bodies" is on, this parameter specifies the group of particles to operate on.

PARTICLE GEO

If "Make Input Particles Rigid Bodies" is on, this parameter specifies the value that should be assigned to the instance attribute of input particles, and thus which object should be used to inherit attributes for the particles.

28.4 PARAMETERS – ATTRIBUTES PAGE

The Attributes page lets you specify which attributes should be inherited from the particle instance object. The initial position and velocity attributes are only inherited on the first frame of the simulation. All other inherited attributes are updated from the instance geometry at each frame. Thus parameters that are animated at the object level will be animated at the particle level as well. See the section on New Object Parameters below for detailed descriptions of all these new parameters.

Inherit Object Translation Set particle P to object T.

Inherit Object Rotation Set particle rot to object R.

Inherit Object Center of Mass
Set particle com (centre of mass) to object com.

Inherit Object Velocity Set particle v to object velocity.

Inherit Object Angular Velocity
Set particle w to object angvel.

Inherit Object Mass Set particle mass to object mass.

Inherit Object Bounce Set particle bounce to object bounce.

Inherit Object Dynamic Friction
Set particle fdynamic to object fdynamic.

Inherit Object Static Friction
Set particle fstatic to object fstatic.

28.5 PARAMETERS – ADD ATTRIBUTES PAGE

Add ID Attributes Add ID and parent attributes.

Add Speed Attribute Add speed attribute (length of velocity vector).

Add Num Hit Attribute Number of times collided.

Add Hit Time Attribute Time of collision.

Add Hit Position Attribute Where collision occurred.

Add Hit Normal Attribute Normal where collision occurred.

Add Hit Force Attribute Force of collision.

29 RBD UNIFY POP

The RBDUnify POP is used to unite multiple RBD particles into one unified particle. RBD particles that are unified in this way are no longer treated independently by the RBD solver. Instead, information about the sub-particles is controlled by the unified particle.

29.1 PARAMATERS – BIRTH PAGE

ACTIVATION */activate*

Turns the POP on and off.

SOURCE GROUP

Subset of points to unify.

BIRTH GROUP

Group to contain birthed points.

PRESERVE BIRTH GROUP

Add particles to group instead of clearing group first.

COLLISION EVENT

Event triggered by rigid body collisions

COLLISION GROUP

Group of colliding rigid body particles

PRESERVE COLLISION GROUP

Do not remove entries from collsion group after each frame

MERGE INPUT PARTICLE GROUPS

Make unified particle a member of any group that any sub-particle belongs to.

29.2 PARAMETERS – ADD ATTRIBUTES PAGE

Add ID Attributes	Add ID and parent attributes.
Add Speed Attribute	Add speed attribute (length of velocity vector).
Add Num Hit Attribute	Number of times collided.
Add Hit Time Attribute	Time of collision.
Add Hit Position Attribute	Where collision occurred.
Add Hit Normal Attribute	Normal where collision occurred.
Add Hit Force Attribute	Force of collision.

30 ROTATION POP

30.1 DESCRIPTION

The Rotation POP can be used to apply an arbitrary rotation to the particle frame of reference. This means that you can add a different rotation to the geometry attached to each particle.

The particle's velocity and up vector define the frame of reference on the particle. The rotation is applied to this reference and defines how instanced geometry will be oriented on the particle. The rotation axis can be evaluated either in component form or as a vector.

Tip: The difference between the Up Vector and Rotation is that if the particle is visualized as an aeroplane, the up-vector would be the of roll / bank on the plane. The Rotation on the other hand would correspond to a person inside the plane turning their head to look in a different direction.

30.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

APPLY MOTION

Specifies what types of points to apply the rotation to:

to all points	Apply the rotation to all of the particles.
to moving points	Apply the rotation only to those particles which are in motion. This means that stopped and dying particles will not be affected.

ANGLE *rotangle*

Specify the Angle to rotate.

PREDEFINED AXES

Stuffs values into the *Axis* parameter if vector mode is enabled. The menu contains a list of commonly used predefined axes.

AXIS */rotx /roty /rotz*

Manually specify the Axis about which to rotate here.

30.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

30.4 SEE ALSO

- *Instance POP* p. 57
- *Lookat POP* p. 73
- *State POP* p. 117
- *Up-Vector POP* p. 130

31 SOFTBODY POP

31.1 DESCRIPTION

The Softbody POP performs soft body deformations similar to the Spring SOP. The edges of the geometry act as "springs" which act to pull the points back toward their original shape.

31.2 PARAMETERS – SOURCE PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

GEOMETRY SOURCE

When using *Parameter Values*, this specifies where the POP should get the Object and SOP to stick to, otherwise it allows you to choose which *Context Geometry* to use.

context geometry

All POP networks can have 'Context Geometry'. These context geometries are set in the parameters of the POPNET, the Particle CHOP, or the POP OP. In the POP OP they can also be set by wiring OPs into the inputs of the POP OP. These context geometries can be referenced by the following POPs: Source POP, Collision POP, SoftBody POP, Attractor POP, Creep POP (All of which have the parameter – *Geometry Source > Use Context Geometry)*. So, instead of hard-coding a specific piece of source geometry or collision geometry, a single POP network can be used by several POP OPs to generate several different outputs, depending on the geometry input into the POP OP.

See *Context Geometry - Example* p. 7.

OBJECT / SOP

Object and SOP containing geometry to use as softbody.

IGNORE TRANSFORM OBJECT

Transform geometry to Transform Object space

FIXED POINTS

Points to be consisdered stationary

HIDE PARTICLE SYSTEM

Make the particle system that controls the softbody deformation hidden (i.e. does not show in scene).

31.3 PARAMETERS – SPRING PAGE

SPRING BEHAVIOR

How the springs will behave:

Hooke's Law	Use Hooke's law: Force = Displacement × Spring constant
Normalize Displacement	Like Hooke's law except displacement is normalized to the original length of the spring (behavior used in Houdini 2.5).
No Springs	No springs will be used. Spring forces will not be applied between points. This behaviour is similar to the *Modify Source Geometry* option available in the particle SOP.

SPRING CONSTANT *(/springk)*

Stiffness of the spring.

INITIAL TENSION *(/tension)*

Initial spring tension before deformations.

31.4 PARAMETERS – ATTRIBUTES PAGE

Create Local Variables	Create local variables corresponding to inherited attributes.
Add ID Attributes	Add ID and parent attributes.
Add Generation Attribute	Add generation attribute.
Add Origin Attribute	Add origin attribute.
Add Speed Attribute	Add speed attribute.

USE AS ORIGIN

How to set the origin.

ORIGIN INDEX */originindex*

Index as used in origin.

31.5 NOTES

- When this POP is first activated, the geometry at that time is used. If the SOP changes thereafter, those changes will not be reflected in the geometry produced by the POP network.
- The POP creates a new particle system that will move the points of the geometry. To make this particle system invisible to the scene, enable the *Hide Particle System* parameter. In the info pop-up, the particle system will be denoted as "virtual".

31.6 SEE ALSO

- *Ref > Geometry > Spring SOP*

32 SOFTLIMIT POP

32.1 DESCRIPTION

The SoftLimit POP creates a spongy boundry surface off of which particles can bounce. This makes it easier to get 'nice' non-jerky behaviour than from the Limit and Collision POPs.

32.2 PARAMETERS

ACTIVATION */activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

TYPE

Shape of object to bounce on.

+ LIMIT */limitposx /limitposy /limitposz*

Positive object value.

- LIMIT */limitnegx /limitnegy /limitnegz*

Negative object value.

BEHAVIOUR

Whether the sponge will try to keep particles inside itself, or keep them outside of itself.

FORCE */force*

The constant force that should be applied in the shortest direction out of the sponge.

DAMPENING */vscale*

The *Dampen* parameter basically controls the maximum recovery speed (and hence the maximum speed at which particle can exit at) and how violently it reacts to fast moving particles.

The force is reduced by the component of the particle's velocity traveling out of the sponge. It is increased by the component travelling into the sponge. This is the scale to adjust that computation. The total force is never negative.

IGNORE MASS

Ignore particle mass.

OVERRIDE MASS

Override particle mass.

MASS *(mass*

Mass to use for override.

32.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

32.4 SEE ALSO

- *Collision POP* p. 29
- *Limit POP* p. 65

33 SOURCE POP (GENERATOR)

33.1 DESCRIPTION

Particles originate from geometry. The Source POP births particles from the specified SOP geometry. This allows a direct relationship between the geometry and the origins of the particles to be established. The Source POP is a generator POP and will add a new particle primitive to the geometry detail.

Note: POPs automatically adapt to the SOP-space of the object it is cooking from – just as if it had a built-in Object Merge SOP. In order to over-ride this behaviour, enable the *Ignore Transform Object* parameter.

33.2 PARAMETERS – SOURCE PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

EMISSION TYPE

This menu establishes from where in the source geometry the particles should be birthed:

Points (ordered) — The particles will birth from all the existing points in the source geometry. Particles are birthed in the order that they are numbered. If the geometry consists only of a primitive (e.g. primitive sphere), then only one point of geometry will exist for particles to birth from. If the source geometry is a NURBS curve or surface, the CVs will be used.

Points (random) — This setting works the same as *ordered*, with the exception that the points are birthed in a random order rather than in the order in which they are numbered. This would be similar to running the geometry through a Sort SOP and selecting *Random* - which is no longer necessary with this option. Note that this random order does not change over time. That is, once the sequence is used once, it will birth in the same order during the next sequence.

Prim center (ordered) — Birth particles from the center of the primitives in the SOP. This is useful when used in combination with the *Rotate to Template* parameter in the Primitive SOP.

Prim Center (random)	Same as the ordered version, but goes throught the source primitives in a random order.
Prim. Center (attribute)	See 'Emission – Attribute', below.
Edges (ordered)	Particles are birthed from both spline and polygon edges. The birth points are along continuous intervals on the curve, not just from CVs or points.
Edges (random)	Same as the ordered version, but goes throught the source primitives in a random order.
Edges Center (attribute)	See 'Emission – Attribute', below.
Surfaces (ordered)	Will birth particles from a UV spline surface or a mesh type. If there is no surface but only a spline curve, then particles are birthed from the curve. If polygons are used, points will only be birthed from the polygon edges.
Surfaces (random)	Same as the ordered version, but goes throught the source primitives in a random order.
Surfaces (attribute)	See 'Emission – Attribute', below.
Volume	This will birth particles from within a closed geometric volume. The behaviour is undefined if the geometry is not closed.
Metaballs	Particles are birthed from within a metaball volume according to the *Density Threshold* and *Density Minimum* settings.

Note: If you have a trimmed geometry surface, the *From surfaces* option will birth particles only from the visible portion of the trimmed surface. If you want particles to be birthed from the entire surface, specify the SOP before the trimming action occurs.

emission – attribute

These emission types are similar to the existing types except they use the 'Distribution Attribute' from each primitive as the probability distribution. Primitives with a value <= 0 are not birthed from. For the remaining primitives, for each particle to be birthed, the probability that the primitive is chosen is equal to that primitive's attribute value divided by the sum of all the primitives' values. Combined with the Measure SOP, this allows birthing based on the primitives' areas.

GEOMETRY SOURCE

When using *Parameter Values*, this specifies where the POP should get the Object and SOP to stick to, otherwise it allows you to choose which *Context Geometry* to use.

context geometry

All POP networks can have 'Context Geometry'. These context geometries are set in the parameters of the POPNET, the Particle CHOP, or the POP OP. In the POP OP they can also be set by wiring OPs into the inputs of the POP OP. These context geometries can be referenced by the following POPs: Source POP, Collision POP, SoftBody POP, Attractor POP, Creep POP (All of which have the parameter – *Geometry Source > Use Context Geometry)*. So, instead of hard-coding a specific piece of source geometry or collision geometry, a single POP network can be used by several POP OPs to generate several different outputs, depending on the geometry input into the POP OP.

See *Context Geometry - Example* p. 7.

OBJECT / SOP

Specify the Object / SOP containing the geometry here.

SOURCE GROUP

If the specified sop contains a point or primitive group, you can use it here to birth only from that group.

IGNORE TRANSFORM OBJECT

Normally particles automatically adapt to the Object space of the geometry from which the particles are birthed. You can override this behaviour by enabling this option.

DENSITY THRESHOLD */threshold*

This, and the next option only apply if *Emission Type* is set to *Metaball*.

Metaballs are fields of effect with a density at their centre, and falling to a density of zero at their edges. When birthing particles from metaballs, this POP will continue sampling points until the summed density exceeds this threshold.

For example, suppose the threshold density is set to 0.8. The first point sampled has a density of 0.3. The second has density 0.4. The sum is now 0.7. The third point has density 0.15. This pushes the sum to 0.85, exceeding the threshold. This third point is then used as the source position for a birth. For metaballs with a negative weight, the absolute value of the density is used to compute the sum.

Setting the threshold high tends to cause births to occur near the denser areas but you will still get births elsewhere. The end result is a nice falloff instead of a hard edge where the threshold would be.

DENSITY MINIMUM */densitymin*

When birthing from metaballs, points with densities less than or equal to this value are not used to birth particles. For metaballs with a negative weight, the absolute value of the density is used so this parameter behaves more like a density maximum.

33.3 PARAMETERS – BIRTH PAGE

IMPULSE ACTIVATION */impulseactivate*

Turns impulse birthing on and off.
Impulse birthing results in all particles being born at once.

IMPULSE BIRTH RATE */impulserate*

Number of particles to birth per frame.
Birth rate interpreted as exact number of particles to birth.

CONSTANT ACTIVATION */constantactivate*

Turns constant birthing on and off. This type of birthing results in continuous birthing while the POP is active as determined by the *Constant Activation* field. Birth rate is in particles-per-second.

CONSTANT BIRTH RATE */constantrate*

How many particles to birth per second.

BIRTH PROBABILITY */probmin /probmax*

How many particles to birth per source point. Fractional values can be used. For example, 4.5 means birth at least 4 points and have a 50% probability that a fifth will be born. When birthing from instanced geometry, fractional values will be rounded down because only integer numbers of particles can be birthed per source point.

BIRTH GROUP

If you would like the birthed points to be defined within a group, specify a group to contain birthed points here.

LIFE EXPECTANCY */life*

How long each particle will live, in seconds. The default is 100 seconds, which is just over a minute and a half. Use small values such as one or two to see the effect.

LIFE VARIANCE */lifevar*

Variance of a particle's life expectancy in seconds. If life expectancy is one second, and the variance is zero seconds, each particle will live exactly one second. If variance is set to *0.5*, then some particles will live only a half second, while others will live a second and a half. The rest will live some time in-between. This randomness gives a more natural look to the particle births.

ACCURATE BIRTHS

This parameter simulates the Particle SOP's behaviour of having *Jitter Births* and *Accurate Moves* both enabled. It tells the Source POP to recook the source geometry each time a new particle is going to be birthed. Thus it can slow the POP considerably, but if the source geometry is very fast moving or deforming, it can produce a much more satisfactory distribution of particles.

33.4 PARAMETERS – ATTRIBUTES PAGE

INHERIT ATTRIBUTES

Defines which attributes to inherit from geometry source.
By default, all attributes are inherited (*). You can also specify a pattern.
For example:

```
v Cd ^up
```

to specify that particles should inherit: velocity, diffuse color but not up vector.

When birthing from edges / surfaces, floating point and vector attributes are interpolated. When birthing from volumes or metaballs, the attribute defaults are used.

CREATE LOCAL VARIABLES

Creates local variables corresponding to inherited attributes.

INITIAL VELOCITY

How to set the initial velocity:

Use Inherited Velocity	Particles will use the velocity attribute or normal of the geometry from which they are birthed.
Add to Inherited Velocity	Particles will use the velocity attribute or normal of the geometry from which they are birthed + the amount specified in *Velocity* below.
Set Initial Velocity	Sets the velocity of the particles to the value specified in *Velocity* below.

INHERIT VELOCITY *inheritvel*

How much of inherited velocity to use. This allows you to scale the inherited velocity. For example, entering a value of 0.5 means that the particles' inherited velocity will be scaled by half.

VELOCITY *vel*

Used to set or add to a particle's velocity.

VARIANCE /var

By adding a variance to the velocity, you can randomise the velocities generated.

ELLIPSOIDICAL VARIANCE

Allows you to birth particles with random directions that are evenly distributed in an ellipsoid (whose size is determined by *Variance*) about the base velocity, rather than a cube.

33.5 PARAMETERS – ADD ATTRIBUTES

ADD ID / GENERATION / ORIGIN / SPEED ATTRIBUTES

Add ID and parent / generation / origin, and speed attributes.

attributes added

Parameter	Attribute	Local Variable
Add ID Attr	id, parent	$ID, $PARENT
Add Generation Attr	gen	$GEN
Add Origin Attr	origin	$ORIGIN
Add Speed Attr	speed	$SPEED

Example: When checked, the *speed* attribute is calculated and is equal to: *sqrt(VX^2+VY^2+VZ^2)* . It can be referenced through the local variable $SPEED.

USE AS ORIGIN

How to set the origin attribute.

index — Enter an integer in the Origin Index field which will be used as a unique ID to identify where the particle originated from.

index + geo num — The origin will be set by adding the Origin Index to the geometry number. For point emissions, the geometry number is the point number. For edge and surface emissions, the geometry number is the primitive number. For metaball or volume emissions, there is no geometry number and 0 is used. Using this option, it is possible to determine exactly which part of the geometry a particle was birthed from.

ORIGIN INDEX /originindex

Enter an integer number here as referenced by *Use as Origin*.

33.6 LOCAL VARIABLES

BBX BBY BBZ	Bounding box.
CA	Alpha
CR CG CB	Diffuse color
DIST	Distance to collision.
DRAG	Drag.
ITER	Processing iteration.
MAPU MAPV MAPW	Texture coordinates.
MASS	Mass.
NPT	Number of points in detail.
NGRP	Number of group entries.
NPRIM	Number of primitives in detail.
PSCALE	Scale.
NX NY NZ	Normal.
P	Source point/primitive number.
U V	Source UV position.
SPEED	Absolute speed of particle.
TIMEINC	Frame time increment.
TX TY TZ	Position.
UPX UPY UPZ	Up vector.
VX VY VZ	Velocity.
WEIGHT	Point weight.

These variables refer to the source geometry's properties.

Note: Only the *Birth Probability*, *Life Expectancy*, *Life Variance*, *Inherit Velocity*, *Velocity*, *Variance* and *Origin Index* parameters make use of the local variables.

33.7 SEE ALSO

- *Split POP (Generator)* p. 111
- *Stream POP (Generator)* p. 119

34 SPEEDLIMIT POP

34.1 DESCRIPTION

The SpeedLimit POP can be used to set minimum and maximum speed limits. Each particle can have its own set of imposed limits. If the limit values are negative, then no restrictions are imposed on the particle's speed. This POP will only set the speed-limit attributes on the particle. The limits are not imposed until the particle is moved during post frame processing with its default velocity rule (see *Suppress Rule POP* p. 124). The POP will add the attribute if it doesn't exist already.

34.2 PARAMETERS

ACTIVATION *(activate)*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

SPEEDLIMIT *(speedlimitmin speedlimitmax)*

Set the minimum / maximum speed of the particle here.

34.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

34.4 SEE ALSO

- *Suppress Rule POP* p. 124

35 SPLIT POP (GENERATOR)

35.1 DESCRIPTION

The Split POP births particles from other particles. It is a generator POP and will add a new particle primitive to the geometry detail.

35.2 PARAMETERS – SOURCE PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

EMISSION TYPE

This option is similar to the options provided in the Source POP > *Emission Type* p. 102 except the points/edges/surfaces refer to instanced geometry. Select from the menu to establish from where the particles should be birthed:

From particle Birth from particle position.

Points (ordered) of Instance

The particles will birth from all the existing points in the source geometry. Particles are birthed in the order that they are numbered. If the geometry consists only of a primitive (e.g. primitive sphere), then only one point of geometry will exist for particles to birth from. If the source geometry is a NURBS curve or surface, the CVs will be used.

Points (random) of Instance

This setting works the same as *ordered*, with the exception that the points are birthed in a random order rather than in the order in which they are numbered. This would be similar to running the geometry through a Sort SOP and selecting *Random* - which is no longer necessary with this option. Note that this random order does not change over time. That is, once the sequence is used once, it will birth in the same order during the next sequence.

Prim. center (ordered) of Instance
Birth particles from the center of the primitives in the sop. This is useful when used in combination with the Rotate to Template parameter in the Primitive SOP.

Prim. center (random) of Instance
Same as the ordered version, but goes throught the source primitives in a random order.

Edges (ordered) of Instance Particles are birthed from both spline and polygon edges. The birth points are along continuous intervals on the curve, not just from CVs or points.

Edges (random) of Instance Same as the ordered version, but goes throught the source primitives in a random order.

Surfaces (ordered) of Instance
Will birth particles from a UV spline surface or a mesh type. If there is no surface but only a spline curve, then particles are birthed from the curve. If polygons are used, points will only be birthed from the polygon edges.

Surfaces (ordered) of Instance
Same as the ordered version, but goes throught the source primitives in a random order.

Volume of Instance
This will birth particles from within a closed geometric volume. The behaviour is undefined if the geometry is not closed.

Metaballs of Instance Particles are birthed from within a metaball volume according to the *Density Threshold* and *Density Minimum* settings.

Note: If you have a trimmed geometry surface, the *from surfaces* option will birth particles only from the visible portion of the trimmed surface. If you want particles to be birthed from the entire surface, specify the SOP before the trimming action occurs.

DENSITY THRESHOLD */threshold*

This, and the next option only apply if *Emission Type* is set to *Metaball*.

Metaballs are fields of effect with a density at their centre, and falling to a density of zero at their edges. When birthing particles from metaballs, this POP will continue sampling points until the summed density exceeds this threshold.

For example, suppose the threshold density is set to 0.8. The first point sampled has a density of 0.3. The second has density 0.4. The sum is now 0.7. The third point has density 0.15. This pushes the sum to 0.85, exceeding the threshold. This third point is then used as the source position for a birth. For metaballs with a negative weight, the absolute value of the density is used to compute the sum.

Setting the threshold high tends to cause births to occur near the denser areas but you will still get births elsewhere. The end result is a nice falloff instead of a hard edge where the threshold would be.

DENSITY MINIMUM */densitymin*

When birthing from metaballs, points with densities less than or equal to this value are not used to birth particles. For metaballs with a negative weight, the absolute value of the density is used so this parameter behaves more like a density maximum.

35.3 PARAMETERS – BIRTH PAGE

BIRTH PROBABILITY */probmin /probmax*

How many particles to birth per source point. Fractional values can be used. For example, 4.5 means birth at least 4 points and have a 50% probability that a fifth will be born. When birthing from instanced geometry, fractional values will be rounded down because only integer numbers of particles can be birthed per source point.

BIRTH GROUP

If you would like the birthed points to be defined within a group, specify a group to contain birthed points here.

LIFE EXPECTANCY */life*

How long each particle will live, in seconds. The default is 100 seconds, which is just over a minute and a half. Use small values such as one or two to see the effect.

LIFE VARIANCE */lifevar*

Variance of a particle's life expectancy in seconds. If life expectancy is one second, and the variance is zero seconds, each particle will live exactly one second. If variance is set to *0.5*, then some particles will live only a half second, while others will live a second and a half. The rest will live some time in-between. This randomness gives a more natural look to the particle births.

KILL ORIGINAL PARTICLE

After birthing, kill original particle. This removes the source particle from the primitive it belongs to. This allows you to "transfer" particles from one primitive to another.

35.4 PARAMETERS – ATTRIBUTES PAGE

INHERIT ATTRIBUTES

Defines which attributes to inherit from geometry source.
By default, all attributes are inherited (*). You can also specify a pattern.
For example:

```
v Cd ^up
```

to specify that particles should inherit: velocity, diffuse color but not up vector.

When birthing from edges or surfaces, floating point and vector attributes are interpolated. When birthing from a volume or metaballs, the attribute defaults are used.

INITIAL VELOCITY

How to set the initial velocity:

Use Inherited Velocity	Particles will use the velocity attribute or normal of the geometry from which they are birthed.
Add to Inherited Velocity	Particles will use the velocity attribute or normal of the geometry from which they are birthed + the amount specified in *Velocity* below.
Set Initial Velocity	Sets the velocity of the particles to the value specified in *Velocity* below.

INHERIT VELOCITY *inheritvel*

How much of inherited velocity to use. This allows you to scale the inherited velocity. For example, entering a value of 0.5 means that the particles' inherited velocity will be scaled by half.

VELOCITY *vel*

Used to set or add to a particle's velocity.

VARIANCE *var*

By adding a variance to the velocity, you can randomise the velocities generated.

ELLIPSOIDICAL VARIANCE

Allows you to birth particles with random directions that are evenly distributed in an ellipsoid (whose size is determined by *Variance*) about the base velocity, rather than a cube.

USE AS ORIGIN

When the emission type is *From Particle*, the origin of the new particle is inherited from its parent. When birthing from instanced geometry, this parameter tells how to set the origin attribute.

The Origin Index can be composed of three different parts, or all added together: *index*, *geo num* and *parent*.

index	An integer entered in the Origin Index field.
geo num	For point emissions, the geometry number is the point number. For edge and surface emissions, the geometry number is the primitive number. For metaball or volume emissions, there is no geometry number and 0 is used.
parent	The origin value of the parent particle.

ORIGIN INDEX *(originindex)*

Enter an integer number here as referenced by *Use as Origin*.

35.5 LOCAL VARIABLES

The local variables available for this POP are:

BBX BBY BBZ	Bounding box.
CA	Alpha
CR CG CB	Diffuse color
DIST	Distance to collision.
DRAG	Drag.
ITER	Processing iteration.
MAPU MAPV MAPW	Texture coordinates.
MASS	Mass.
NPT	Number of points in detail.
NGRP	Number of group entries.
NPRIM	Number of primitives in detail.
PSCALE	Scale.
NX NY NZ	Normal.
P	Source point/primitive number.

U V	Source UV position.
SPEED	Absolute speed of particle.
TIMEINC	Frame time increment.
TX TY TZ	Position.
UPX UPY UPZ	Up vector.
VX VY VZ	Velocity.
WEIGHT	Point weight.

These variables refer to the source particle's properties. It does not refer to the properties of the instanced geometry when birthing from instanced geometry.

Note: Only the *Birth Probability*, *Life Expectancy*, *Life Variance*, *Inherit Velocity*, *Velocity*, *Variance* and *Origin Index* parameters make use of the local variables.

35.6 SEE ALSO

- *Source POP (Generator)* p. 102
- *Stream POP (Generator)* p. 119

36 STATE POP

36.1 DESCRIPTION

The State POP can be used to set the particle's state information. This includes whether the particle is dying, is stopped, or has just collided. A non-zero value in a flag enables it.

The default state of a particle is zero, implying it's alive and healthy. A change in state can result from the particle being dead, stopped, stuck, or just hit. Although the various state attributes can be accessed via convenient local variables, the value of the state attribute is a bitwise OR operation of the various possibilities:

dead=1, stopped=2, stuck=4, justhit=8.

Thus, a dead and stopped particle will have a state value of 3. Most POPs will not process particles with non-zero states, as it would, for example, be inefficient to process dead particles.

36.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

DEAD *dead*

Boolean value which indicates whether the particle will be killed. A value of 1 means that the particle is dying and will be killed at the end of this frame.

STOPPED *stopped*

Boolean value which indicates whether the particle is stopped. A stopped particle is immobile and is also not susceptible to forces acting on it. Its position, velocity and axis will not change. A value of 1 means that the particle is stopped.

STUCK *stuck*

Allows you to set or unset the Stuck state of a particle. This is useful if you want particles to stick to a primitive for a while, and then get released.

SLIDING */sliding*

Allows you to set or unset the Sliding state of a particle. Without this, you could only know if a particle had collided, but you could not know if it was in the process of sliding after the collision. This would be useful, say, if you wanted to add friction to particles which were sliding across a collision surface.

JUST HIT */justhit*

Boolean value which indicates whether the particle just collided during this frame. This state acts like an impulse. It is 1 during the frame the particle collided and 0 otherwise.

36.3 PARAMETERS – RIGID BODY DYNAMICS PAGE

RIGID BODY */rbody*

Particle is a rigid body.

ACTIVE */active*

Particle is an active rigid body.

OVERRIDE */override*

Particle instance transform should be overridden.

36.4 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area. These additional RBD variables also apply:

RBODY	Particle is a rigid body.
ACTIVE	Particle is an active rigid body.
OVERRIDE	Particle instance transform should be overriden.

36.5 SEE ALSO

- *Collision POP* p. 29
- *Limit POP* p. 65

37 STREAM POP (GENERATOR)

37.1 DESCRIPTION

The Stream POP is used to birth particles where streams of other particles intersect. Birthing is done when the density of colliding particles is greater than or equal to some specified threshold. Spatial partitions are created and the threshold specifies how many particles need to be present in one of these partitions before birthing occurs.

The terms *partition* and *division* are interchangeable within this POP. One way to visualise this is: given a 3D cube which is 2 × 2 × 2 in dimension; and the partitions are set to 1 × 1 × 1, then you end up with 8 partitions. Because the particle is not birthed directly from geometry or existing particles, its attributes will contain the default values.

37.2 PARAMETERS – SOURCE PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

BIRTH POSITION

Where in the partition new particles are to be birthed:

Centre of Division Particles are birthed from the centre of the division.

Average Position of Particles
Particles are birthed from the average position of the particles in a division. This yields births where the particles look visually most dense.

Random Position between Particles
Particles are birthed at random positions in the division

DIVISION SIZE *sizex sizey sizez*

Size of each uniform partition. The divisions/partitions are uniform only in the sense that all are identical in shape. However, the X, Y, and Z division sizes need not be identical.

OFFSET OF GRID *offsetx offsety offsetz*

This offset is subtracted from each uniform partition in order to move it away from its default anchor. The default anchor is the origin, which makes this offset necessary in some instances.

For example, imagine a case where the high density part of one stream lies perfectly on a division boundary. These particles are then considered to be in only one of these partitions, and even if the other stream is dense enough in the other partition, and will be remarkably close, no particles would be birthed. This offset then becomes necessary in order to offset the partition so it encloses the stream.

ENABLE / STREAM / GROUP / KILL

These buttons allow the following functions:

Enable: Stream <n>	Stream <n> will be used.
Group: Stream <n>	Group to use for stream <n>
Threshold: Stream <n>	Number of stream <n> particles required in partition to birth (*/threshold<n>*).
Kill Original particle <n>	Kills the original particle <n> in that stream.

37.3 PARAMETERS – BIRTH PAGE

** = parameters where THRESH and DENSITY variables can be used.*

BIRTH PROBABILITY * */probmin /probmax*

How many particles to birth per partition. The Birth Probability is used as follows. The first field is the minimum probability of a birth in a partition with the minimum density. The second field is the maximum probability. So 1-1 will result in one particle being birthed, while 0.5-1.5 will result in anywhere from 0-2 particles being birthed.

Fractional values can be used. For example, 4.5 means birth at least 4 points and have a 50% probability that a fifth will be born.

BIRTH GROUP

If you would like the birthed points to be defined within a group, specify a group to contain birthed points here.

LIFE EXPECTANCY * */life*

How long each particle will live, in seconds. The default is 100 seconds, which is just over a minute and a half. Use small values such as one or two to see the effect.

LIFE VARIANCE * */lifevar*

Variance of a particle's life expectancy in seconds. If life expectancy is one second, and the variance is zero seconds, each particle will live exactly one second. If variance is set to *0.5*, then some particles will live only a half second, while others will live a second and a half. The rest will live some time in-between. This randomness gives a more natural look to the particle births.

37.4 PARAMETERS – ATTRIBUTES PAGE

INITIAL VELOCITY

How to set the initial velocity. The inherited velocity is considered to be the average of all the enabled particles in the given partition. Only particles in enabled streams are used.

Use Inherited Velocity	Particles will use the inherited velocity only.
Add to Inherited Velocity	Particles will use the inherited velocity + the amount specified in *Velocity* below.
Set Initial Velocity	Sets the velocity of the particles to the value specified in *Velocity* below.

INHERIT VELOCITY */inheritvel*

How much of inherited velocity to use. This allows you to scale the inherited velocity. For example, entering a value of 0.5 means that the particles' inherited velocity will be scaled by half.

VELOCITY */vel*

Used to set or add to a particle's velocity.

VARIANCE */var*

By adding a variance to the velocity, you can randomise the velocities generated.

ELLIPSOIDICAL VARIANCE

Allows you to birth particles with random directions that are evenly distributed in an ellipsoid (whose size is determined by *Variance*) about the base velocity, rather than a cube.

ORIGIN INDEX */originindex*

Enter an integer number here to be used as the origin.

37.5 LOCAL VARIABLES

DENSITY	Total number of particles from enabled streams contained in the partition.
THRESH	Sum of all enabled stream thresholds.
TIMEINC	Frame time increment.
TX TY TZ	Birth position.

The *Birth Probability*, *Life Expectancy*, *Life Variance*, *Inherit Velocity*, *Velocity*, *Variance* and *Origin Index* parameters make use of the local variables.

37.6 SEE ALSO

- *Source POP (Generator)* p. 102
- *Split POP (Generator)* p. 111

38 SUBNET POP

38.1 DESCRIPTION

The Sub-net POP is essentially a way of creating a macro to represent a collection of POPs as a single POP in a Network Editor. The Sub-net POP can contain an entire POP Network within it, stream-lining and simplifying your POP network both visually and conceptually.

Selecting *Edit Sub-Network...* from the OP's pop-up menu presents you with a new Network Editor with four sub-network inputs. These four inputs are connected directly to the four inputs on the Sub-net OP in your original network. Proceed by attaching OPs as required to these four sub-network inputs. The display OP will be wired back to the output connector of the Sub-net OP in your original OP network. To get back to the original OP network, go up a level (type U).

Please refer to the *Sub-Networks* p. 190 in the *Interface* section for a complete discussion and an example of how to use sub-networks.

Tip: Select several Operators that you want to make into a sub-network, and select *Collapse Selected* from the *POP's pop-up* menu to automatically create a sub-network out of them. You will see the selected Operators replaced by a single Subnetwork POP, and it will be properly rewired to contain the previously selected POPs.

38.2 PARAMETERS

INPUT #1 - #4 LABEL

These labels are displayed when you click with on one of the Sub-net POP's inputs. This is useful for remembering what the inputs are used for in your Sub-network.

39 SUPPRESS RULE POP

39.1 DESCRIPTION

The Suppress Rule POP is used to turn off the application of default particles rules. These rules performs default time stepping operations like updating particle positions and aging the particle. By not applying the default rules, it allows custom rules to be built using fundamental POPs like Position, Velocity, Up Vector or Age.

Tip: To motion blur particles, you need to create all the points you need at frame 0, and then 'release' them into the POP tree with a Suppress POP.

39.2 PARAMETERS – MOTION PAGE

SUPPRESS DEFAULT POSITION RULES

Do not apply default rule that updates position.

SUPPRESS DEFAULT VELOCITY RULES

Do not apply default rule that updates velocity.

SUPPRESS DEFAULT ROTATION RULES

Do not apply default rule that updates rotation.

SUPPRESS DEFAULT ANGULAR VELOCITY RULES

Do not apply default rule that updates angular velocity.

39.3 PARAMETERS – OTHER PAGE

SUPPRESS DEFAULT UP VECTOR RULES

Do not apply default rule that updates up vector.

SUPPRESS DEFAULT AGING RULES

Do not apply default rule that updates age.

SUPPRESS DEFAULT REAPING RULES

Do not apply default rule that reaps dead particles. Particles will be dead, but will not be removed from the system.

39.4 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

39.5 SEE ALSO

- *Age POP* p. 21
- *Position POP* p. 78
- *Up-Vector POP* p. 130
- *Velocity POP* p. 132

40 SWITCH POP

40.1 DESCRIPTION

The Switch POP switches between different inputs. The inputs are numbered consecutively from 0 to N-1 (where N is the number of inputs to the POP).

The Switch POP will use the POP which is specified in the input parameter (this can, of course, be animated).

This POP is useful for applying different POPs to a particle system as time passes. By animating the input selected as a function of time, a different set of POPs can be applied to the particles.

40.2 PARAMETERS

SELECT INPUT */input*

Input to select. This can be animated using a channel, or an expression can be used.

41 TORQUE POP

41.1 DESCRIPTION

The Torque POP applies a rotational force to the particles. To help visualize the effect of this operator, turn on the particle axes flag in the Viewport Display options. Torque is expressed as a force applied to a position relative to the particle.

This POP modifies the *torque* attribute.

41.2 PARAMETERS – TORQUE PAGE

ACTIVATION */activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

FORCE */forcex /forcey /forcez*

Magnitude of the Force.

POSITION */posx /posy /posz*

Position at which to apply force.

SCALE */scale*

Overall force scaling factor.

41.3 PARAMETERS – NOISE PAGE

Note: For a complete description of the *Noise* page parameters, see the entry: *Parameters – Noise Page* p. 24.

Seed */seed*
Seed for random turbulence generator.

Turbulence */turb*
Number of iterations to add fractional noise.

Roughness */rough*
Scale of noise added with each iteration.

Exponent	*/atten* Noise attenuation.
Frequency	*/freqx /freqy /freqz* Spatial frequency of noise field.
Amplitude	*/ampx /ampy /ampz* Maximum value of noise field.
Offset	*/offsetx /offsety /offsetz* Amount to shift noise.
Noise Type	Type of noise: *Hermite*, *Improved Hermite*, *Sparse Convolution*, and *Alligator*.

41.4 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

42 TRANSLATION POP

42.1 DESCRIPTION

The Translation POP is used to apply a translation to the particle frame of reference. The particle's velocity and up vector define the frame of reference on the particle. This translation displaces the frame of reference from the particle's position. The translation can be evaluated either in component form or as a vector.

42.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Subset of points to act on.

APPLY MOTION

Type of points to apply translation to.

TRANSLATION *transx transy transz*

The amount of translation along the X, Y, and Z axes.

42.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

42.4 SEE ALSO

- *Rotation POP* p. 95

43 UP-VECTOR POP

43.1 DESCRIPTION

The Up-vector POP can be used to define the particle frame of reference up vector. The velocity defines the Z direction. The up vector defines the Y direction and the cross product of the two define the X direction.

The particle frame of reference is used to orient geometry instanced on the particle. The up vector can be evaluated either in component form or as a vector.

Tip: The difference between the Up Vector and Rotation is that if the particle is visualized as an aeroplane, the up-vector would be the amount of roll / bank of the plane. The Rotation on the other hand would correspond to a person inside the plane turning their head to look in a different direction.

This POP can be used to provide an initial Up Vector for particles when they are born or to alter the Up Vector according to your own rules.

Note that there are default rules applied to all particles that will transform the Up Vector according to the particle's directional change from frame to frame. To turn off the application of the default rule, use the *Suppress Rule POP* p. 124.

Because the default rule needs to know what the previous velocity of the particle is, this POP also adds the previous velocity attribute. If the Up Vector is inherited from source geometry, the previous velocity attribute will automatically be added.

43.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

APPLY MOTION

Specifies what types of points to apply the up vector to:

to all points Apply the up vector to all of the particles.

to moving points Apply the up vector only to those particles which are in motion. This means that stopped and dying particles will not be affected.

UP VECTOR */upx /upy /upz*

Supply an Up vector here.

43.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

43.4 SEE ALSO

- *Rotation POP* p. 95
- *State POP* p. 117
- *Suppress Rule POP* p. 124

44 VELOCITY POP

44.1 DESCRIPTION

The Velocity POP can be used to explicitly set a particle's velocity.
The velocity can be evaluated either in component form or as a vector.

There are default rules applied to all particles that will change its velocity depending on the acceleration. This POP allows the particle velocity to be set directly. However, note that the default rules will still be applied unless they are turned off with the Suppress Rule POP.

The difference between velocity and acceleration is that velocity is the *rate of travel*, whereas acceleration is the *change in rate of travel*.

44.2 PARAMETERS

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

APPLY MOTION

Specifies what types of points to apply the position to:

to all points	Apply the velocity to all of the particles.
to moving points	Apply the velocity only to those particles which are in motion. This means that stopped and dying particles will not be affected.

VELOCITY *vx vy vz*

Enter the particle velocity here.

The current velocity can be obtained from the $VX, $VY and $VZ variables. For example, to change the particle's velocity by its acceleration (scaled by the time increment since acceleration = distance-per-second). Use (in vector form):

```
vector3($VX, $VY, $VZ) + $TIMEINC * vector3($AX, $AY, $AZ)
```

44.3 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

44.4 SEE ALSO

- *Position POP* p. 78
- *State POP* p. 117
- *Suppress Rule POP* p. 124

45 WIND POP

45.1 DESCRIPTION

The Wind POP applies wind forces on the particles. The acceleration due to wind is proportional to the given wind intensity and the particle's drag coefficient.

WIND POP VS FORCE POP

So just what is the difference between the Wind POP and the Force POP?

The application of external force directly affects a particles' acceleration, the rate of which is determined by the mass (F = Mass × Acceleration). Wind is an additional force, but one that is velocity sensitive. If a particle is already travelling at wind velocity, then it shouldn't receive any extra force from it. This implies a maximum velocity when using Wind on its own.

An increase in mass impedes acceleration for a given constant force. Drag is a force opposing the direction of motion which is velocity sensitive, i.e. the larger the velocity, the greater the effect of drag. Its useful for limiting the velocity of particles.

45.2 PARAMETERS – WIND PAGE

ACTIVATION *activate*

The *Activation* field turns the entire POP on and off. A POP is only active if this parameter evaluates to a value greater than 0. See *POP Events* p. 12 for an example.

SOURCE GROUP

Specify a subset of points by entering a group(s) here on which to act.

WIND *windx windy windz*

Wind magnitude acting on the particles. Using wind (and no other forces, such as turbulence (Noise page), the particles will not exceed the wind velocity.

SCALE *scale*

Scales the entire wind force acting on the particle.

IGNORE MASS

Ignore particle mass in computations. When mass is used, the applied force is divided by the particle's mass to compute an acceleration. Ignoring the mass means that the force is applied directly as an acceleration.

When mass is used, the particles will accelerate at different rates if the same force is applied. Heavier particles will move more slowly. When mass is ignored, all particles will have the same acceleration applied, regardless of what their mass is.

OVERRIDE MASS / MASS */mass*

Override particle mass with the value specified in *Mass*.

OVERRIDE DRAG / DRAG */drag*

Override particle drag with the value specified in *Drag*.

45.3 PARTICLES – NOISE PAGE

The wind applied by the POP can be modulated by noise. The noise is added to the wind so its direction and strength are modified.

Note: For a complete description of the *Noise* page parameters, see the entry: *Parameters – Noise Page* p. 24.

Seed	*/seed* Seed for random turbulence generator.
Turbulence	*/turb* Number of iterations to add fractional noise.
Roughness	*/rough* Scale of noise added with each iteration.
Exponent	*/atten* Noise attenuation.
Frequency	*/freqx /freqy /freqz* Spatial frequency of noise field.
Amplitude	*/ampx /ampy /ampz* Maximum value of noise field.
Offset	*/offsetx /offsety /offsetz* Amount to shift noise.
Noise Type	Type of noise: *Hermite*, *Improved Hermite*, *Sparse Convolution*, and *Alligator*.

45.4 LOCAL VARIABLES

These are listed in *Particle Attributes and Local Variables* p. 9. They are also listed in the pop-up help (click the ?) in the Parameters area.

45.5 SEE ALSO

- *Drag POP* p. 39
- *Property POP* p. 80