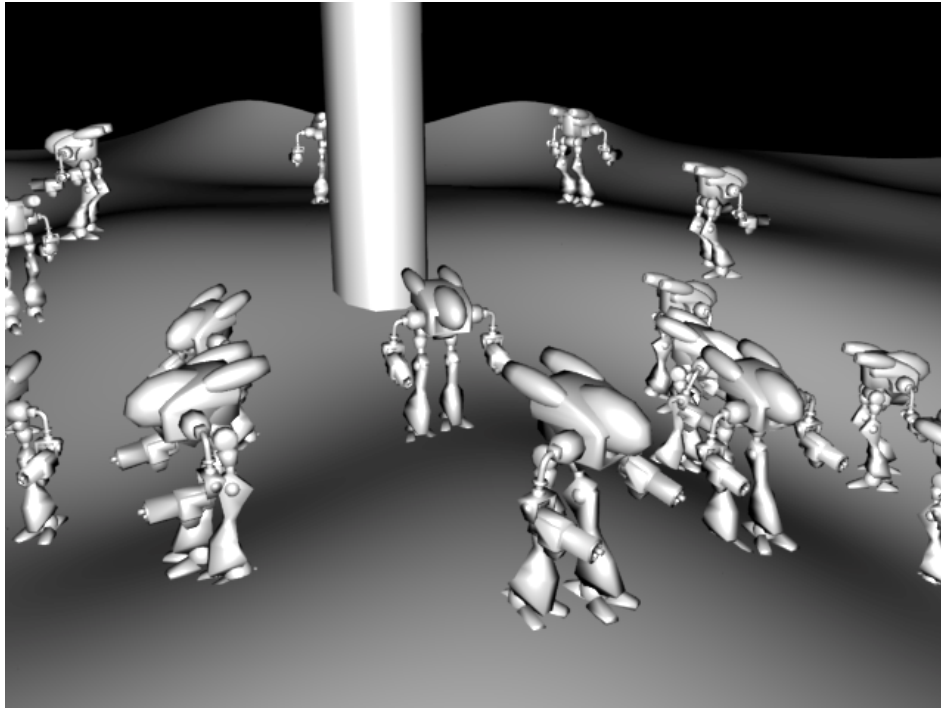


Workshop 07: Particle III - Virtual Crowd

Overview



One way to implement a virtual crowd system is to use particle. A virtual crowd system contains the following elements:

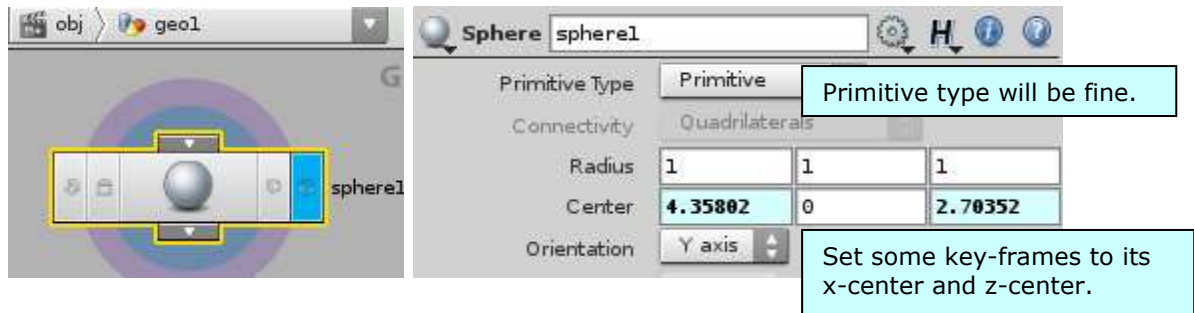
A "leader"	Usually the motion of the leader will be controlled by animator, i.e. can be key-framed.
"Followers"	<p>"Followers" will then follow the motion path of the leader. In particle implementation, they are a group of particles. The motion will be automatically generated by the software. Geometry will then be copied onto the particles.</p> <p>There are some additional requirements for a virtual crowd system:</p> <ul style="list-style-type: none"> ● Followers will roughly follow the leader's path; ● Followers will not crash into each other; ● Followers will avoid obstacles; ● Followers will align on a landscape, if the scene contains one. ● Different geometry can be copied onto the particles, based on some particle's attribute; say, the speed of a particle (fast speed particle uses geometry A; slow speed particle uses geometry B; etc). ● Followers may have minor variation among themselves. They should not look too regular.

Firstly, we will create a simple crowd system without obstacles and landscape. Then we will add obstacles. Lastly, we will add a landscape.

Step 1) Create a leader

We can use a sphere as the "leader"; so, let's set key-frame to a simple sphere (note: to make this exercise simple, try to keep the sphere lies on the plane $y=0$):

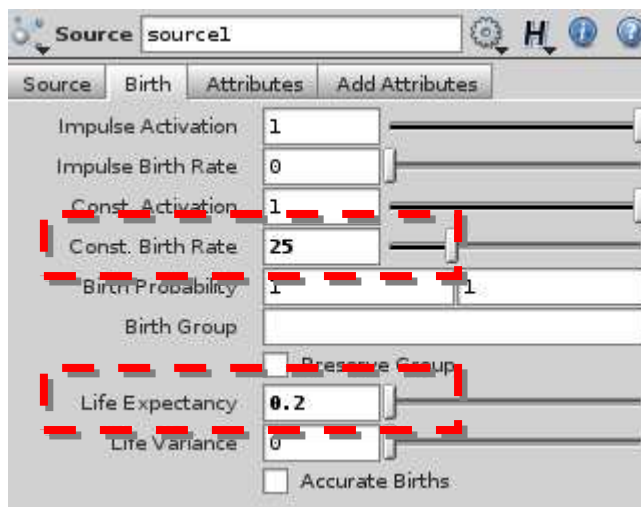
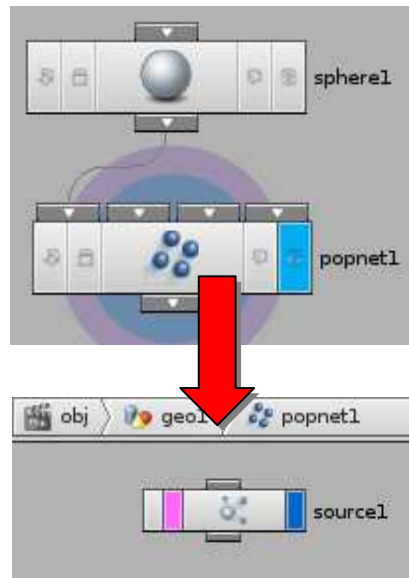
1. In OBJ level, create a **GEOMETRY**.
2. Go into the geometry, create a **SPHERE**.
3. Set key-frames in the x-center and z-center attributes of the sphere. (To set key-frames, basically we repeat the following steps: (1) choose a frame; (2) move the sphere; (3) Alt-left-click on the parameters we want to set a key at.)



In Houdini, leaders should also be particles. Therefore, we need to generate particles from the sphere:

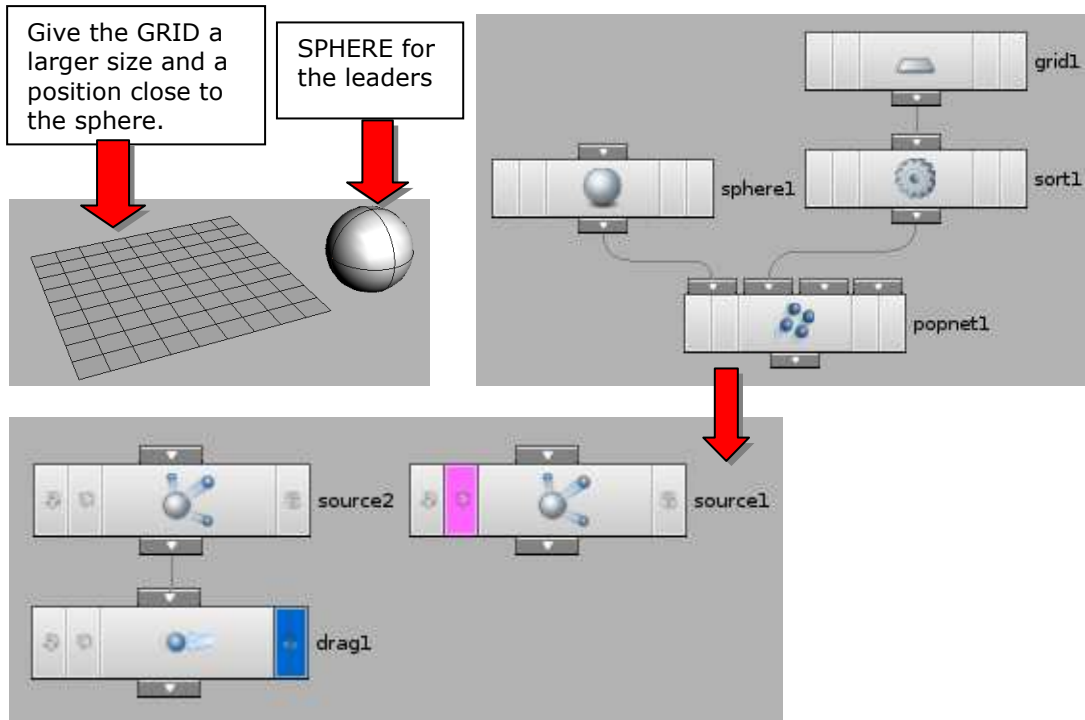
1. Connect the sphere to a **POP NETWORK**.
2. Go into the pop network, and create a **SOURCE**. Use the sphere as the source of particle (i.e. choose "Use First Context Geometry").

As we want to keep the particle's position closer to the sphere's position, I would like to reduce the life of the particles. I also don't need so many particles (it will only slow down the machine), so I reduce the birth-rate, too:

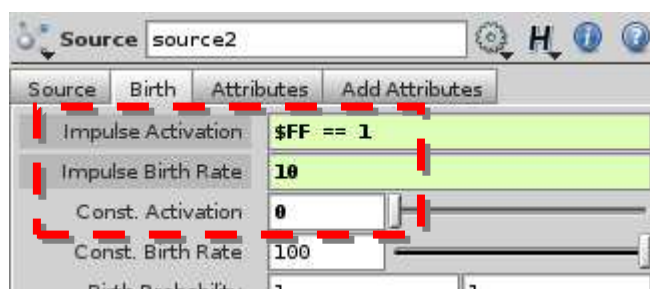


Step 2) Create the followers

We can use a **GRID** as the source of "follower" particles. Connect a GRID to a **SORT** to randomize its point number, and then connect the SORT to the POP NETWORK, and add one more **SOURCE** in the pop network. Adding a default **DRAG** to the followers is always a good idea:

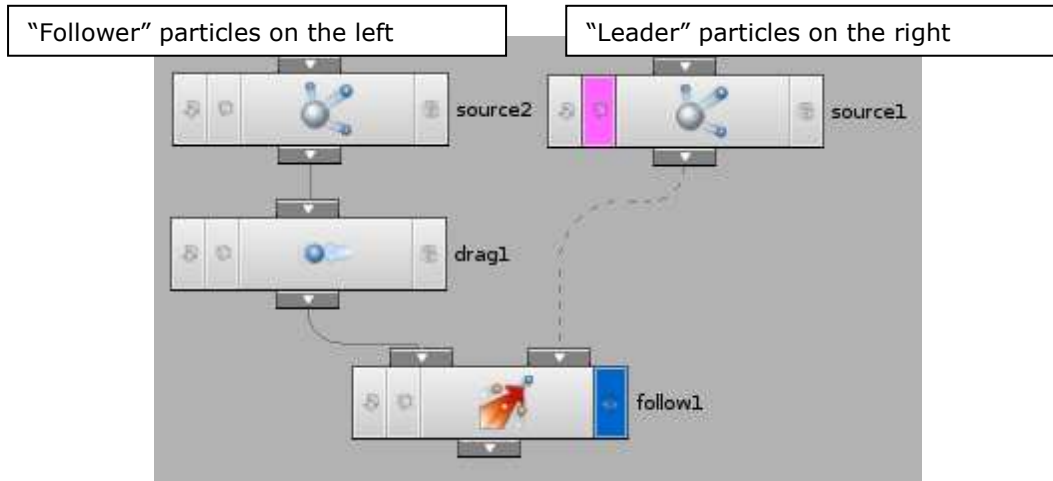


For followers, I want to generate fixed number of particles (say, 10 followers) at frame #1. Therefore, in the second SOURCE, I can use the trick mentioned before:



Step 3) FOLLOW

To ask a group of particles to follow another group of particle, we use **FOLLOW**:



There are some parameters of the FOLLOW that you may want to fine-tune:

The screenshot shows the 'Follow follow1' behavior settings panel. The 'Behavior' dropdown is set to 'Accelerating Follow'. The 'Attract Distance' is set to 1, and the 'Scale' is set to 4. Red dashed boxes highlight these settings, with callout boxes explaining their functions:

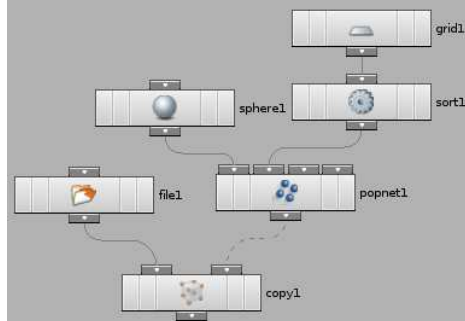
- Accelerating Follow:** "Accelerating Follow" means that leader particles will apply an attractive force to the followers.
- Attract Distance:** If the distance between leader and follower is closer than this value, the "attractive" force will become a "repulsive" force.
- Scale:** Magnitude of the force

Note that when the leader has stopped, there are some artifacts in the follower's motion. It is because the FOLLOW forces the followers to move, even though the leader has stopped. It can be solved by some ad hoc method, say, activate the FOLLOW only before frame #300:

The screenshot shows the 'Follow follow1' behavior settings panel with the 'Activation' field set to '\$FF < 300'. This configuration ensures that the follow behavior is only active for the first 300 frames of the animation.

Step 4) Copy geometry onto follower only

Now we can copy geometry onto the particles to see what the result so far is. If you need a model, you can use the file **bgeo\pmstandstill.bgeo**:



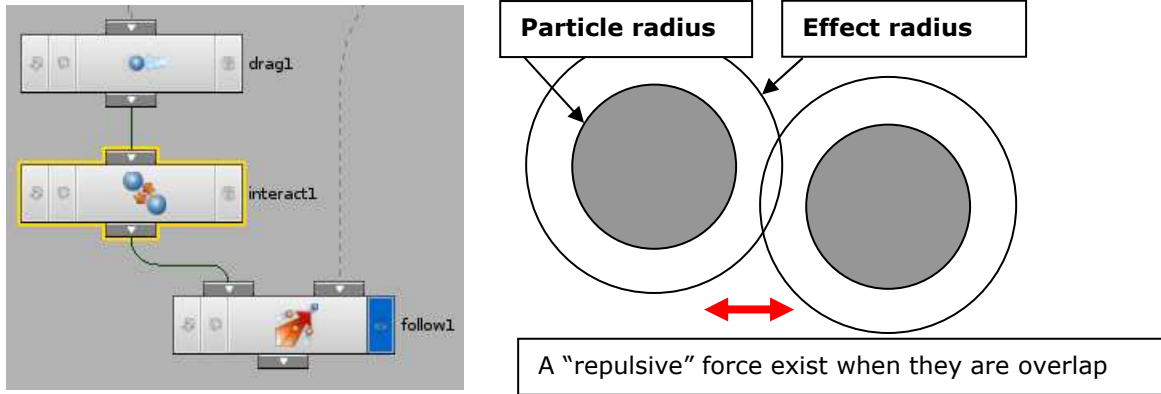
Note that we don't want to copy the geometry onto the "leader" particles. One solution is to group the follower particles using **GROUP**, and then apply **COPY** only onto that group:

The first screenshot shows the **Group** node configuration for **group2**. The **Group Name** is set to **follower**. The **Generators** list includes **source2**. A red dashed box highlights the **Group Name** and **Generators** fields. A callout box says "Give the group a meaningful name". Another callout box points to **source2** with the text "And choose the followers' SOURCE name". A red arrow points from **source2** in the **Generators** list to the **group2** node in the graph below.

The second screenshot shows the **Copy** node configuration for **copy1**. The **Template Group** is set to **follower**. A red dashed box highlights the **Template Group** field. A callout box says "COPY only be applied onto 'follower' group".

Step 5) INTERACT

To avoid particles crash into each others, we can use the **INTERACT**. This OP will add an "effect radius" to each particle, and apply a "repulsive" force to particles when their effect radius overlaps:



There are some parameters of the INTERACT that you may want to fine-tune, for example:

You can scale the default particle radius, or to supply a radius for the particle.

You can control the size of "effect radius". The larger "effect radius" is, the more empty space between particles.

Smaller "Rolloff Exponent" makes the force drop slowly within the effect radius. Larger value makes the force drop quicker.

Magnitude of the "repulsive" force

If, for example, later on you want to copy a larger geometry onto the particles, you can give a larger value to the "particle radius".

NOTE that the job of an INTERACT is not to avoid collision. So if the particles are moving too fast, this INTERACT cannot guarantee that the particles are separated. In order to get a realistic virtual crowd system, you need to fine-tune different parameters of the whole pop network, say: the FOLLOW's force magnitude, the INTERACT's force magnitude, the "radius" of particle, etc.

Note also that after frame #300, my FOLLOW has been disabled. Therefore, the INTERACT become the ONLY force, which may also lead to some undesired result.

Exercise 1: Add obstacles using COLLISION and/or ATTRACTOR

Try to add a TUBE to the scene as an obstacle.

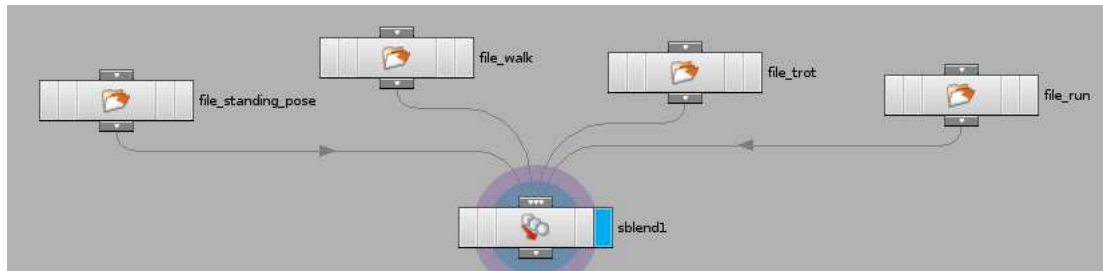
1. Create an obstacle (a **polygon TUBE**)
 2. Try to add a **COLLISION** to guarantee that the followers will not go through the TUBE. (Hint: the collision object can be even larger than the original TUBE.)
- You can also try to create a force near the location of the tube, and then add an **ATTRACTOR** in the pop network. This can “push away” the particles before they hit the obstacle. Try both **METABALL + FORCE** and a **POINT** force. (Which one is better?) Use negative value for “repulsive” force.
-

Step 6) Blending between motions

You are given the following files and animation sequences:

- bgeo/pmstandstill.bgeo
- bgeo/pmwalk0.bgeo ~ bgeo/pmwalk23.bgeo
- bgeo/pmtrot0.bgeo ~ bgeo/pmtrot23.bgeo
- bgeo/pmrun0.bgeo ~ bgeo/pmrun23.bgeo

We can use **SEQUENCE BLEND** to create a smooth transition between those animation sequences. First, use several **FILES** to read in those animation sequences. Then, connect them into a **SEQUENCE BLEND**:



Note that after read in the files, you have to replace:

```
$HIP/bgeo/run$F.bgeo
```

by:

```
$HIP/bgeo/run`$F%24`.bgeo
```

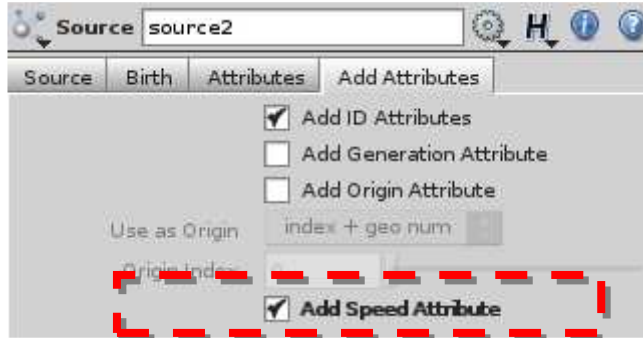
because the animation sequence has 24 frames only.

SEQUENCE BLEND is similar to a SWITCH. The only difference is that the “Blend Factor” value of the SEQUENCE BLEND can have decimal place – to allow smooth transition between motions. Of course, you can key-frame the “Blend Factor” value, but a more interesting approach is to use the particle’s speed to control this value: if a particle stops, it should use the “standing_pose” file. If a particle is moving fast, it should use the “run” sequence. See the next step.

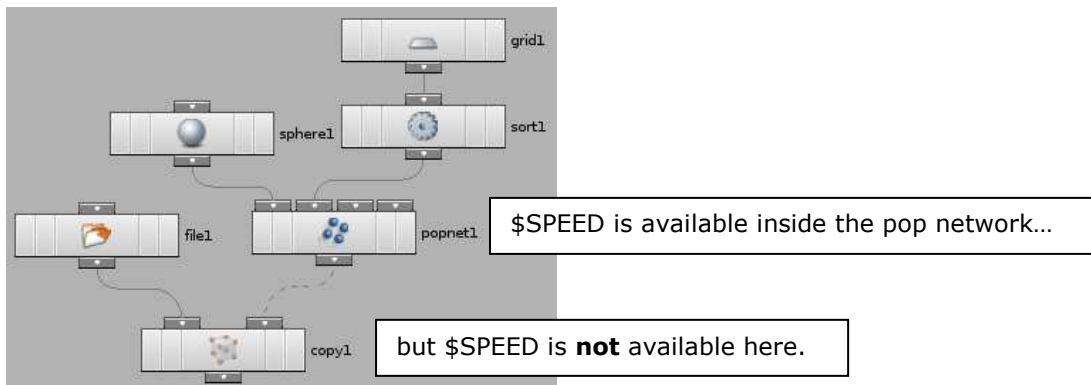
Step 7) Choose suitable motion based on particle speed

Now, suppose we want to use the particle’s speed to choose which animation sequence should be used. To get a particle’s speed, we can use the following tricks:

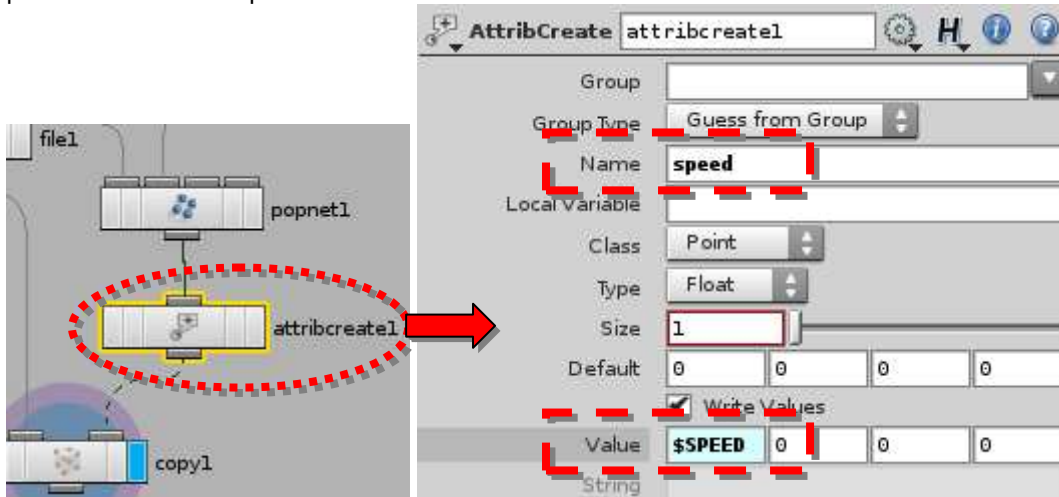
1) Turn on the “**Add Speed Attribute**” of the follower’s **SOURCE**, in order to add the attribute “speed” to the follower particles. Houdini will calculate the speed of the particles automatically:



2) However, this “speed” attribute is available inside the pop network (represented by internal variable \$SPEED), but it is not available in COPY:



To make the “speed” available in COPY, we can use an **ATTRIBCREATE** to “convert it from particle attribute to point attribute”:



Then, the variable \$SPEED, which is the speed of each particle, is available in COPY.

Actually the explanation here: “convert it from particle attribute to point attribute” is not very precise – it involves a concept called “**attribute signature**”, which is too hard to be explained in this course. If you are interested in the detail, you can check the tutorial on Sidefx’s web-site:
http://www.sidefx.com/index.php?option=com_content&task=view&id=1500&Itemid=305

Exercise 2: Choose geometry based on particle's speed

1. Create a **stamp variable** in the **COPY**. The value of the variable should be equals to the particle speed.
2. Add a **stamp()** expression in **SEQUENCE BLEND**, to choose the suitable geometry based on particle's speed.

Additional exercise: each "follower" geometry should have variation – try to use different value to "offset" the frame number of the filename, so that different frame number should be used for different particle. In the order words, in the file name of those FILES, in stead of having this:

```
$HIP/bgeo/pmrun`$F%24`.bgeo
```

I want to have this:

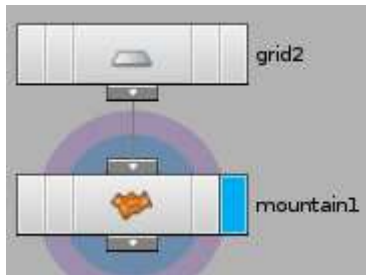
```
$HIP/bgeo/pmrun`($F + per-particle random offset value)%24`.bgeo
```

How to create this effect? (Hint: using stamp() again)

What is the most natural choice for the "per-particle random offset value"?

Step 8) Particles align on the ground

Lastly, suppose we want to add a landscape:



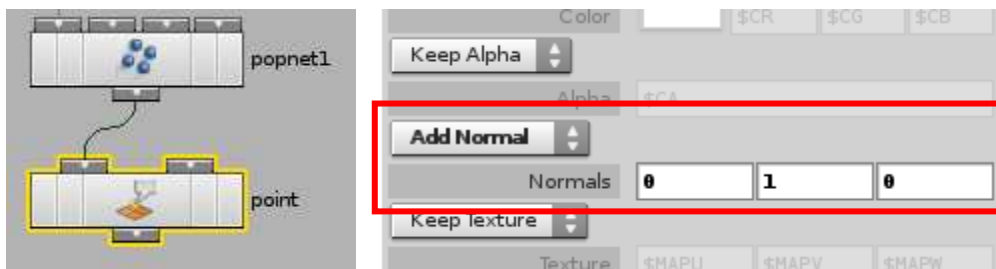
A large **GRID** with primitive type **NURBS**, and y-center = 5.

A **MOUNTAIN** with a larger "height" value

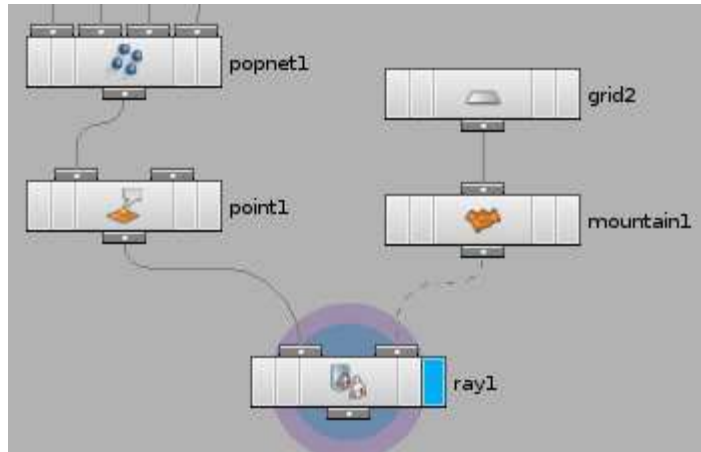
To force all the particles "landing" on a landscape is a little bit tricky. Of course, you can explicitly control the leader's position, to make it follow the up and down of a landscape. However, the follower's positions are not guaranteed.

In Houdini, we can use the **RAY** to **project and move all the particles onto the landscape**.

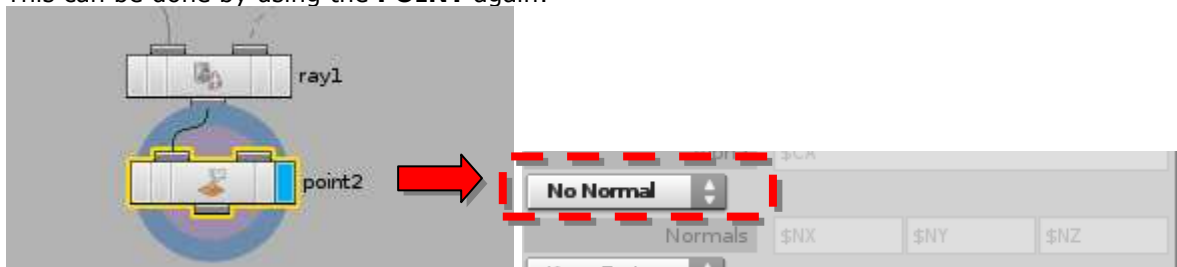
- 1) The RAY will project points along the point's normal. So our first step is: to assign $[0,1,0]$ (or $[0,-1,0]$, depends on your landscape's position) as the point normal to all the particles, using a **POINT**:



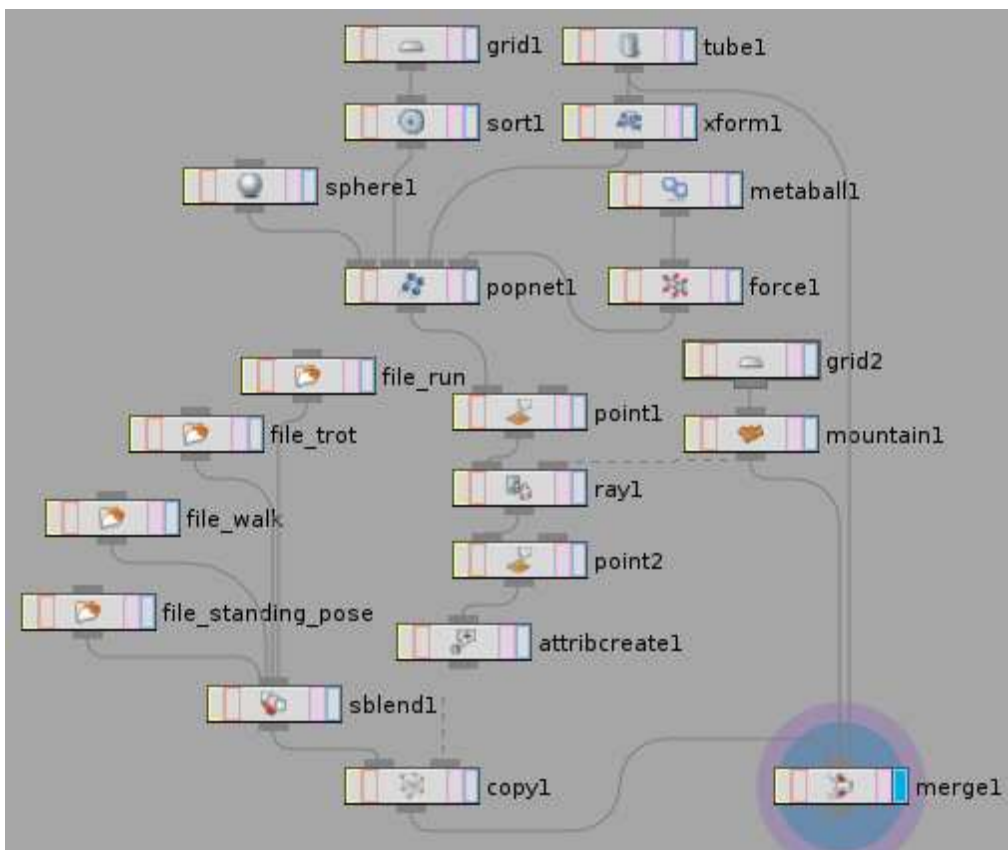
- 2) Next, we connect the **RAY**: the particles will then be projected and moved onto the landscape:



3) However, we need to use the particle's orientation to control the orientation of the copied geometry. Therefore, we need to remove the [0,1,0] point normal before copying. This can be done by using the **POINT** again:



4) Finally, we can copy geometry onto the particles. My network looks like this. Your network may have a little bit different.



**** Week 07 END ****