Chapter 4 - Crowd Agents

This page contains chapter 4 of the Houdini to 3ds Max Alembic Workflow tutorial, covering the export of Crowd Agents.

Overview

Crowd agents in Houdini are represented by what is called a **Packed Agent Primitive**. Packed Agents are a hierarchical structure, a *container* of sorts, that stores the rig, geometry and the animation clips that go with your model. This data is used to create what is called the *definition* of your agent. Furthermore, each **instance** of an agent can have a different set of shapes to randomize the appearance of your crowd - this is referred to as *layers*.

Houdini stores this data on disk and represents each **instance** of an agent in a simulation as a *single point* with a bunch of attributes that govern its appearance and behavior. At render time, Mantra reads those attributes and pulls the necessary information to render your agent with the correct layer /shapes applied to it.

Unfortunately, to our knowledge, the Alembic format does not provide this functionality. When you export a crowd simulation as Alembic, each agent instance is basically baked into the file, producing rather large amounts of data on disk.

The best option for rendering Houdini Crowds with V-Ray in 3ds Max is exporting them to a **VRayScene** using **V-Ray for Houdini** and loading them into 3ds Max.

You can download the project's files from here:

Download Project Files (260MBs)

Export

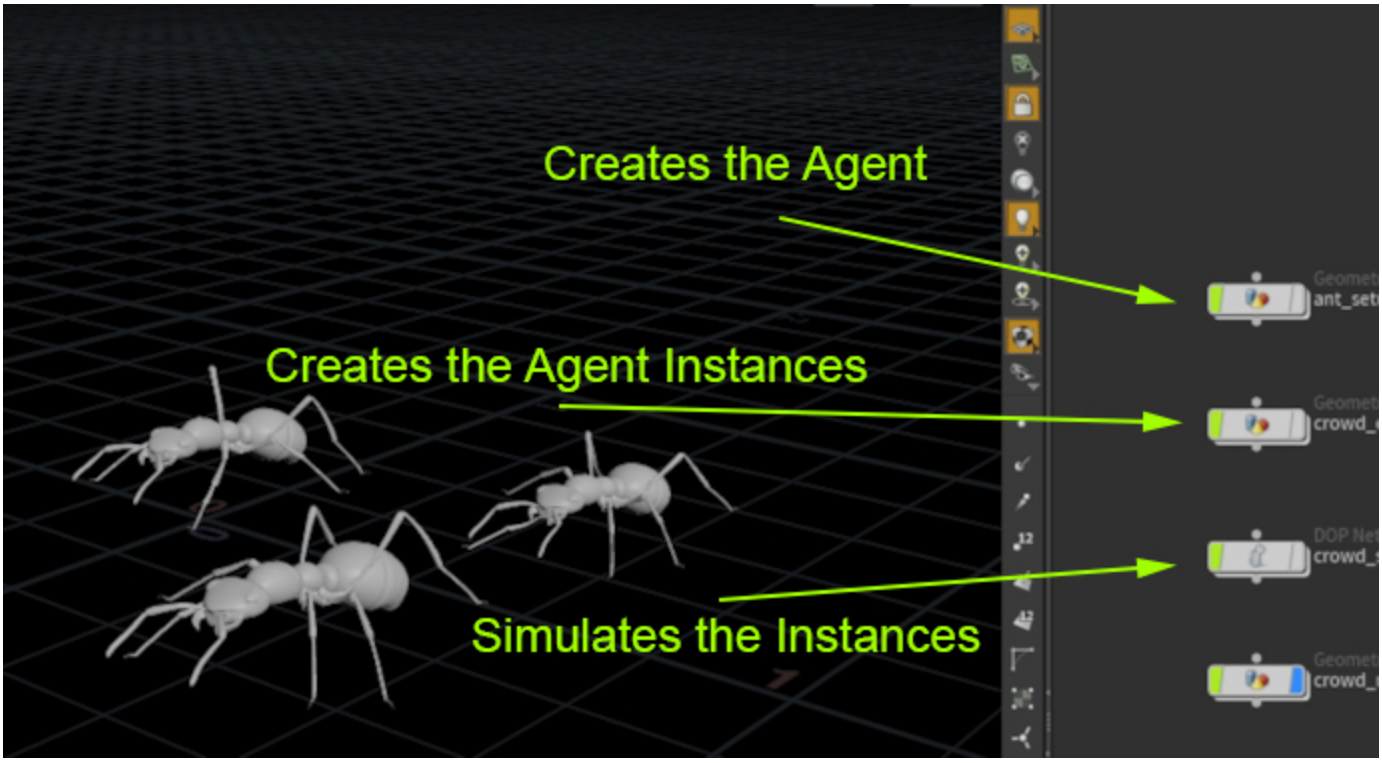
We have provided a project file called **chapter_04_ant_agent.hip.**

It contains a single agent in the **ant_setup** geometry node, with a single walk cycle clip and no additional layers.

The ant agent is instanced using the crowdsource SOP in the **crowd_create** geometry node.

The **crowd_sim** DOP Network contains the crowd simulation for the ants.

Finally, the data pulled back into SOPs using a **DOP Import** node inside **crowd_render**.

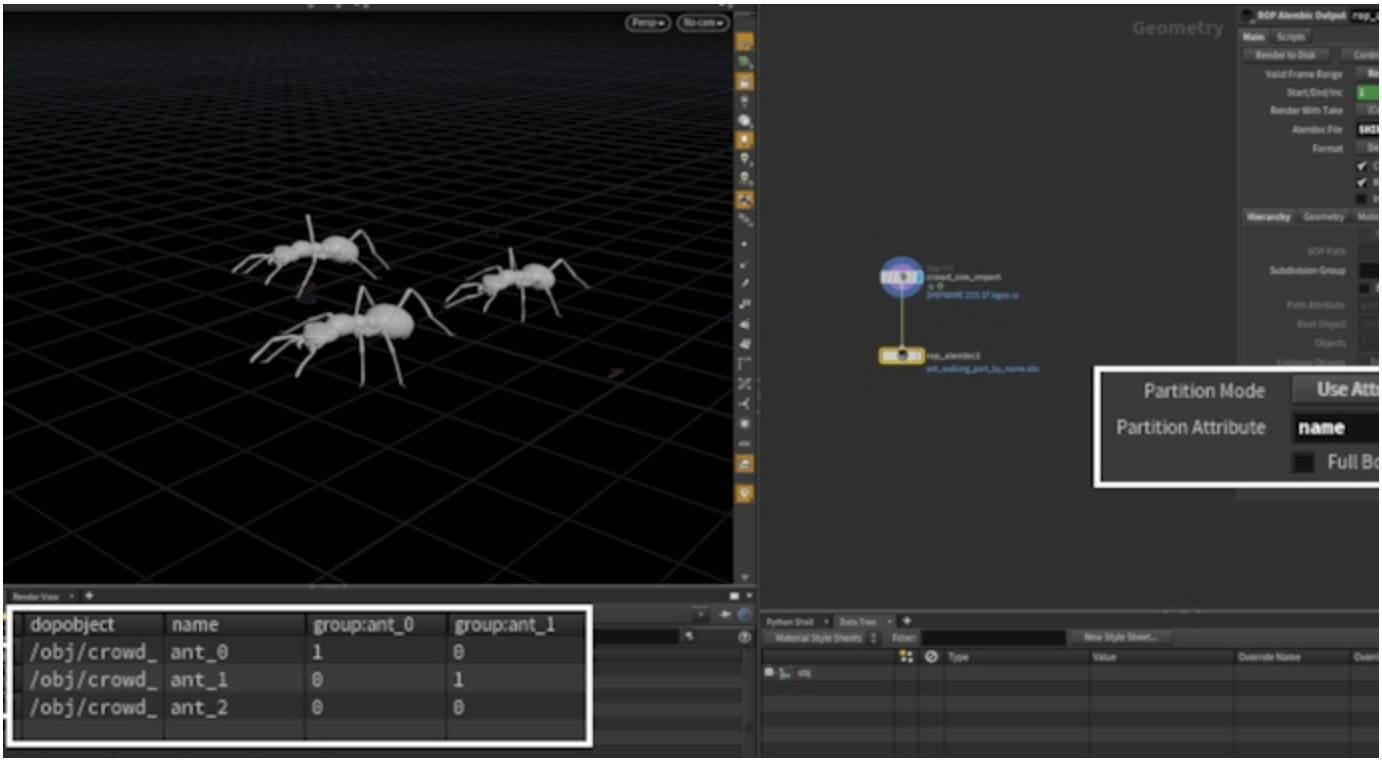


Once you pull the agent instances out of the DOP Network, they come back into SOPs as **Packed** Agent Primitives.

If you are not interested in partitioning the export based on the agent layers or other primitive intrinsic attribute, you can simply drop a **ROP Alembic Output**, set the **Partition Mode** to **Use Attribute Value** and set **@name** as the **Partition Attribute**. This exports **each** agent **instance** as a **separate s hape** inside the alembic file.

If you do want to partition the geometry further, you can drop an **Unpack SOP** and use the **Primitive Intrinsic attributes** in an **Attribute Wrangle** to **generate a custom partition attribute**.

For instance, appending the intrinsic Layer Name attribute to s@name allows you to isolate a subset of your agents that share only this specific layer. Using the shape name, layer name and agent name gives you access down to the individual shape level of each agent instance.



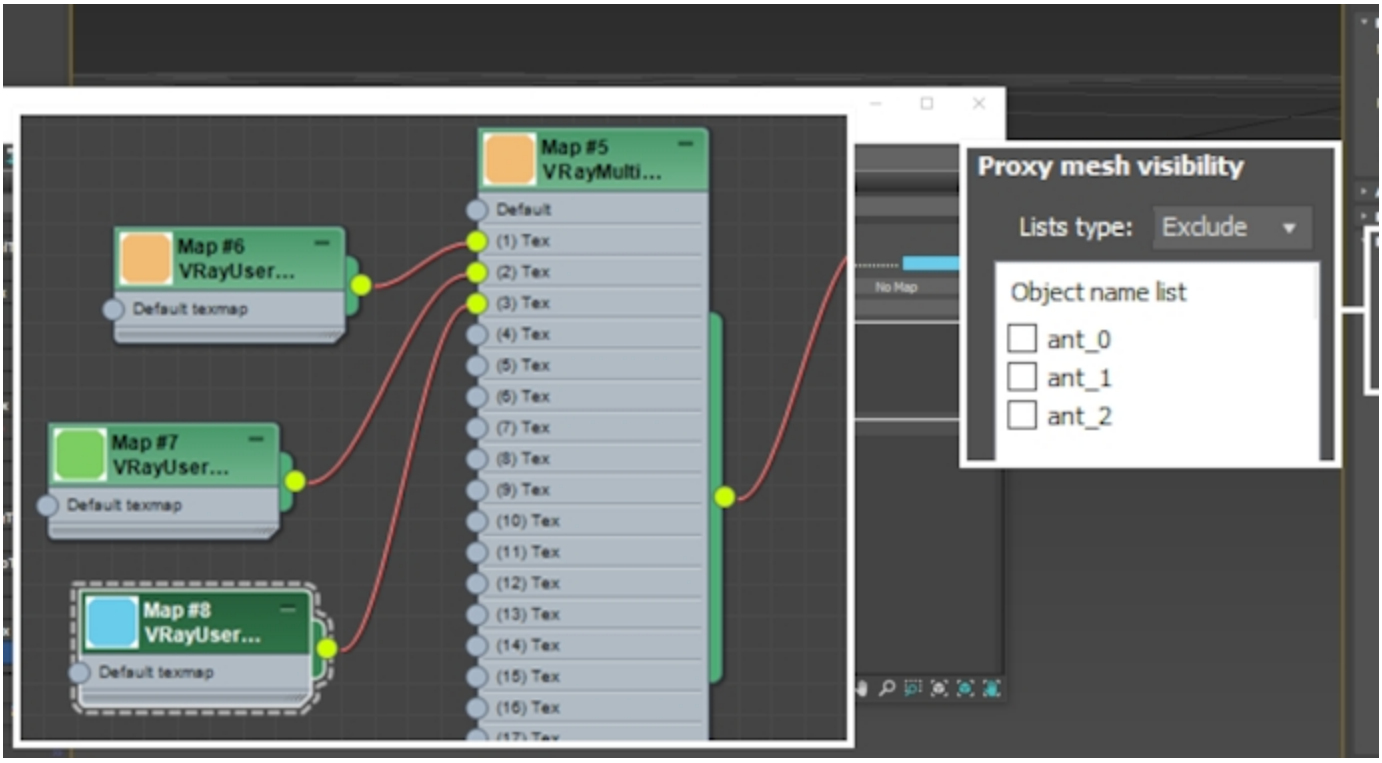
Import into 3ds Max

Back in 3ds Max, you can drop a **V-Ray Proxy** and load the exported Alembic file.

The partitioning should be respected and you should get individual access to each primitive using a **Multi/Sub-Object Material** or the **V-Ray MultiSubTexture**.

By default, the MultiSub texture works using the Face Material IDs but **you can set the Get ID from parameter to Random by Render ID**. This allows you to **randomize the textures** applied to the shapes inside the Alembic.

The same applies for the Multi/Sub-Object - as discussed earlier - you can use the Sub-Object material to apply separate individual materials to each shape.



Here are the 3 ants rendered in 3ds Max.