Advanced Katana

Katana Plugins

Katana provides APIs for creating plugins of various types, but the most commonly-needed plugins are:

- Create scene graph locations: Ops, or the deprecated ScenegraphGenerator plugins
- Inspect the scene graph: ScenegraphIterator plugins
- Modify attributes: Ops, or the deprecated AttributeModifier plugins
- Asset system support: Asset plugins

Katana ships with an Alembic plugin, and so it should be immediately useful. Please see the Katana documentation on how to write custom plugins. Katana also ships with some example plugins for asset management, etc. Many studios use common geometry formats, but many also have their own geometry cache formats that will require Op or ScenegraphGenerator plugins to be written for Katana, or else switch to Alembic to take advantage of the standard.

The `PrimitiveCreate` node can also generate various types of geometry that you can use to get started. If you want to have some fun, let Katana set up an interesting one for you by going to the python tab, and type exactly "i want a pony" without the quotes into the bottom text entry, select the text and hit ctrl+enter. Try it, as it will help you get familiar with the `PrimitiveCreate` node.

KtoA Provided Nodes

KtoA comes with additional nodes and ops that help create common scene setups

- ArnoldObjectSettings
- ArnoldInstanceSettings
- ArnoldUserData
- ArnoldOpenVDBVolume and ArnoldOpenVDBSurface
- ArnoldXGen
- ArnoldYeti

All of these operations can be done with Opscript, AttributeSet and other Katana nodes to create volumes and renderer procedurals manually, but the user must carefully craft these locations to get the attributes set properly. It is usually much easier and less error-prone to use KtoA's provided nodes. For more information on the attributes that KtoA recognizes, please see Katana Location Extensions for extensive details. Note that KtoA recognizes nearly all of the standard Katana parameters and attributes, so the Katana documentation (online help, Appendix G) applies equally to KtoA as it does to the rest of Katana.

KtoA Ops

Note that for each KtoA node, there is an associated python file and a DSO/DLL that implements the actual Op to get the bulk of the work done. By examining the python files, users can see how to create their own opscript that uses the KtoA Ops directly rather than be restricted to the interface provided by the KtoA node. As an example, one user wanted to pre-create locations and tag them with an attribute that, when found by his opscript, would generate XGen procedurals as children locations. The existing ArnoldXGen node was too limiting, but he made use of the `ArnoldXGenOp` directly to do the heavy lifting of creating the `renderer procedural` locations properly for use with XGen.

Glossary

- *node*: where actions take place modifying the scene in some way, and each node can touch any part of the scene; e.g., `CameraCreate, Merge, MaterialAssign,` etc.
- *node graph*: all of the nodes taken together, connected to each other to direct the flow and order of changes in the scene.
- *parameter*: a piece of data on a node telling the node how to operate; parameters may be organized hierarchically inside of a node; e.g., intensity assigned to a light that is being created, a CEL expression saying what scene graph locations to touch, etc.
- *scene graph*: all of the scene graph locations together, organized hierarchically, with special categories for cameras, geometry, lights, materials, and procedurals.
- *scene graph location*: a particular point in the scene graph that has a name, type, and various other attributes, which by convention will have a minimal set of attributes that match the type; e.g., `polymesh, subdmesh, light, camera, group, material, etc.`
- *attribute*: a piece of data on a scene graph location; attributes are organized hierarchically within the location, and may be arrays of substantial amounts of data, e.g., `material` (assigned material), P (vertex positions), N (surface normals), etc.
- *CEL*: a pattern that matches locations in the scene graph; this is used by nodes to decide what parts of the scene graph to operate on.
- *monitor*: the area displaying the results of renders.
- *viewer*: the area displaying a simple preview of the geometric data requested by the user.
- *args file*: an XML file describing how to present node parameters to the user.
- *procedural*: for renderers, this is a program/script that can be run during render time to load more data (so not all data is required to be given up-front). Katana itself interacts with renderers by inserting a Katana procedural into the render, but other procedurals can be inserted by Katana such as a hair instantiation procedural, a crowd instancer, a custom geometry format procedural, etc.
- *op*: an operation on a location in the scene graph; these are usually what actually does the work for nodes in the node graph.