Control:

* Main

Checks that only one node is selected.

If there is only one node selected, instantiate a MegascansAsset object with it – errors will be raised during initialization and caught in Main. A display message in main is called with the error text and asking the user to try again, and everything is aborted (SystemExit raised).

From initialization you know:

**The megascans asset subnet path**

**The megascans asset folder path on disk**

**The megascans asset (technical) name**

**A list of all of the files in megascan asset**

Get nodes

Asset Geometry:

**Asset geometry node, file node, transform node**

Asset Material:

A asset\_material class (represents asset material), subclasses are Redshift, Mantra Etc.

The init of this finds the respective thing needed

From there, the UI is called from main

UI is given the Megascans Object (so can show things like the path on disk etc.), it asks what the user wants, and then passes that to “execute custom lod and baking” on the Megascans Object.

“execute custom lod and baking” creates the fix subnet node, and everything necessary to carry out the job. The Megascans Object returns during “execute custom lod and baking” at any time it’s finished (displaying a message and destroying the fix subnet if there’s a problem e.g. high poly not found)

\*TODO: the above example should be handled in init

Currently, it is unaware if it has been run before – that is, what’s it’s running now, every variable should be an attribute.

I can create a bunch of classes for each renderer type (and store their map name and node setup dicts), and it’s init’s job is to find the needed nodes for the job

Methods like “add to megascans material node setup” and “update megascans\_material reader nodes export­ paths” (i.e. any interaction with megascans material node) could be replaced with hook methods in the class with renders

* Replace get nodes with two things: get\_asset\_geometry\_nodes and get\_asset\_material\_object
* UI will need to be updated to deal with the asset\_material object

Little thing:

Make a helper function to “get information about chosenres” and “get information about lowres” etc. (there is common code)

**Regarding the classes Megascans Material Classes**

A AbstractClass and classes are Redshift, Mantra etc.

The correct class is instantiated based on the logic in get\_asset\_material\_object

it stores rs material builder, redshift material and anything else relevant

‘name’ attribute will store the name of the renderer

‘**Configure megascans material node’** is a method which is given the Megascans Object and does any configuring necessary

‘get\_node\_to\_add\_reader\_nodes’ for redshift, will return rs\_material\_builder

(this will be called be the appropriate things)

**Make custom lod**

* fixsubnet, polyreduce percentage, highpoly name

**Configure megascans material node**

* Self.asset\_geometry\_node, and self.transform\_node (this has redshift parameters, so should be control by Redshift object)

**Cook event handler one & two**

* self.map\_name\_and\_reader\_node\_dict, self.chosen\_bake\_resolution\_str, self.rs\_material\_builder\_node, self.chosenres\_map\_name\_and\_export\_paths\_dict

**bake\_custom\_maps\_and\_update\_reader\_nodes\_accordingly**

* subnet\_node, polyreduce\_percentage\_float, chosen\_bake\_resolution\_str, use\_temp\_resolution\_bool, maps\_to\_bake\_dict
* self.megascans\_asset\_name
* self.megascans\_asset\_folder\_path
* self.map\_name\_and\_reader\_node\_dict
* self.rs\_material\_builder\_node

**execute custom lod and baking**

* polyreduce\_percentage\_float
* maps\_to\_bake\_dict
* chosen\_bake\_resolution\_str
* use\_temp\_resolution\_bool
* fix\_subnet\_node
* map\_name\_and\_node\_setup\_dict
* map\_name\_and\_reader\_node\_dict
* highpoly\_name
* customlod\_path

This tool lets you create a custom LOD and bake maps for a Megascans Asset in Houdini - at the click of a few buttons. Simply select the Megascans Asset’s subnetwork, and click on this shelftool.

You can select the poly-reduce percentage that you want your custom LOD to be, and you can select what resolution you want your baked maps to be. Notably, this tool was created to fix the shoddy Megascan’s LOD and displacement maps.

Optionally, for Redshift, it has the option to automatically create nodes in the Redshift Material Builder for maps that have been baked. It also has the option to temporarily bake out low-res maps while higher resolution maps are being baked.

One thing the tool does is enable tessellation, displacement, and set displacement scale in the Asset Material (which is something Bridge/Livelink forgets to do). This is done every single time the tool is run, which is epic.

Details:

How does it accomplish this? Lots of code. Each time you use the tool, it creates a new subnetwork inside the Megascans Asset subnet called “Megascans\_Custom\_LOD\_and\_Baking\_Subnet”, this holds all node networks required to create a custom LOD and bake maps. Inside, you’ll be able to see a PDG network that shows render progress for baking.

Where does it save what has been baked out? In the same folder with all the other maps for that Megascans Asset.

What naming convention do the newly baked out maps use? It uses the same naming conventions as the other megascan maps, with some alterations. For example, Megascans labels like so: siEoZ\_8K\_Displacement (“siEoZ” refers to the technical-side asset name that Megascans uses). However, as you’ll know maps are baked out based on a Highpoly Source and an LOD, so all of the maps baked out with this tool have the percentage of the LOD that was used to create them e.g. “siEoZ\_1K\_LOD50.0\_custom\_baking\_Vd”. Note that “Vd” corresponds to “Vector Displacement” - this is the naming convention that Houdini uses.

Can I use this tool multiple times on the same Megascans Asset? Sure thing, the tool has no awareness of if it has been run before. Do note that if an LOD with a polyreduce percentage already exists on disk (either from being baked out with this tool, or having the same naming convention to this tool), it’ll use that instead of baking out a brand new one – which is epic.