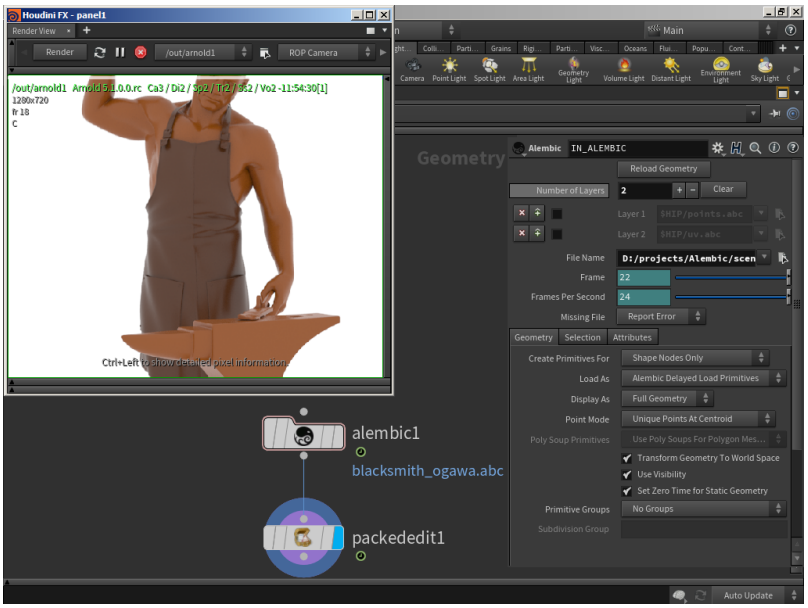
Alembic



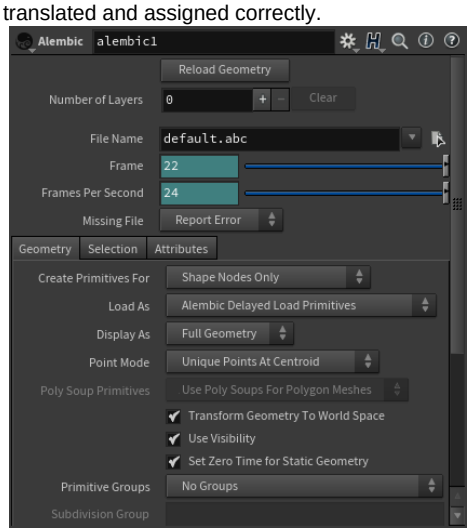
A procedural node that is capable of reading Alembic files. HtoA supports Alembic (.abc) files.

> Information about common Arnold settings can be found here.

Alembic Material Translation

If an asset that has its materials applied when exported as an Alembic, it can be brought back in as a packed Alembic and the material assignment will still work. You will need to enable Export Referenced Materials on the ROP (this is disabled by default). It works by examining the Alembic file for the *shop_materialpath* at translation time.

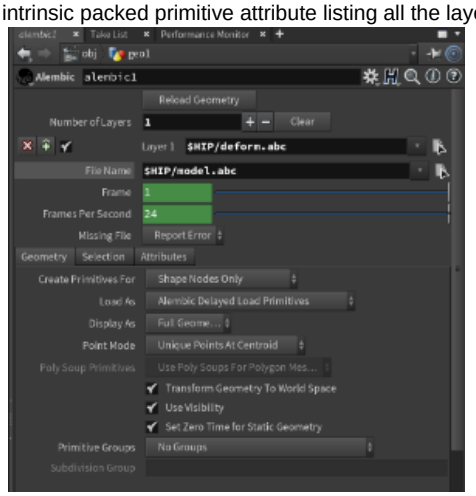
It's also possible to create unified meshes that contiain a *shop_materialpath* attribute that varies per primitive, and these materials will still be translated and assigned correctly.



Layers

Adds extra files that can be used to override properties in the Alembic file. When you load Alembic layers, it creates the abc_filenames

intrinsic packed primitive attribute listing all the layers and the base filename. These are passed to the Alembic procedural.



An extra alembic layer called deform.abc has been added to override the point positions in the base layer model.abc

Material Attribute

Defines which *alembic* property in the file contains the material names, which if present will be used to create the per polygon `shidxs` arrays on the `polymesh` node.

Alembic Packed Primitive Overrides

It is possible to override parameters per packed primitive using houdini primitive attributes. This is possible using the Arnold properties on the OBJ node, but this method will override all packed primitives in the geometry context. Using primitive attributes gives finer grained control over each shape in the alembic file.

> With floats use f@…, strings s@… and integers i@… For enumeration types, you can use s@ for the string or i@ for the index.

Polymesh Examples

Some more examples when the packed primitive is a mesh are:

```
i@ar_visibility = 255;
 i@ar_sidedness = 255;
 i@ar_receive_shadows = 1;
 i@ar_self_shadows = 1;
 i@ar_opaque = 1;
 i@ar_matte = 0;
 f@ar_disp_padding = 1.0;
 f@ar_disp_height = 0.5;
 s@ar_subdiv_type = "catclark";

 i@ar_subdiv_type = 0; // equivalent to s@ar_subdiv_type = "none";

 i@ar_subdiv_iterations = 2;
 i@ar_disp_autobump = 0;
 i@ar_autobump_visibility = 1;
```

> You can use "**`kick -info`**" on target node types to find out the list of parameters that can be used as primitive attribute overrides i.e. {polymesh, curves, points}.

When using an integer for an enum type, use "**`kick -info`**" on the parameter to determine the indices to use. For example:

```
$ kick -info polymesh.subdiv_type
node: polymesh
param: subdiv_type
type: ENUM
default: none
enum values: none catclark linear
```

and so that integer values {0,1,2} correspond to string values {"none", "catclark", "linear"} for this enum type.

So the two lines:

```
s@ar_subdiv_type = "catclark";
 i@ar_subdiv_type = 1;
```

are equivalent. Please note it's safer to use the string form in case the enum list changes in future Arnold releases.

Visibility

The Arnold parameter *visibility* can be modified using this technique, but you can't use the individual ray types such as *ar_visibility_diffuse_reflect* directly. To build the desired values from the bitfields, you would write for example:

```
i@visibility = 32; // i@ar_visibility_diffuse_reflect=1;
i@visibility = 4; // i@ar_visibility_diffuse_transmit=1;
i@visibility = 64; // i@ar_visibility_specular_reflect=1;
i@visibility = 8; // i@ar_visibility_specular_transmit=1;
```

You can also combine the flags by adding the bits for example:

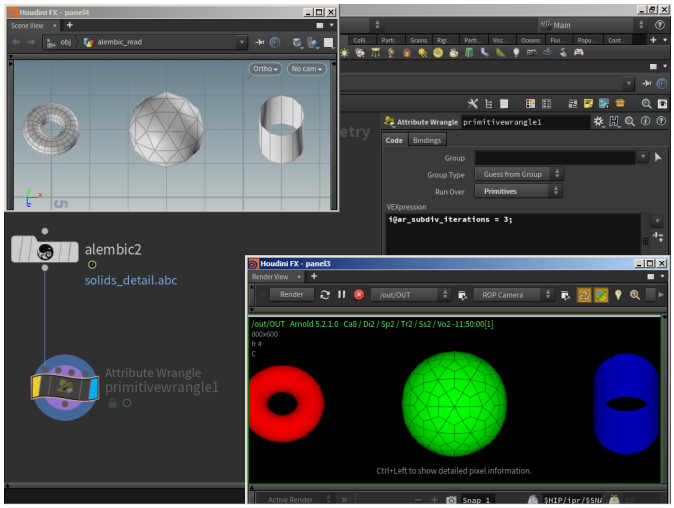
```
i@visibility = 36; // ar_visibility_diffuse_reflect and
ar_visibility_diffuse_transmit only
```

See https://trac.solidangle.com/arnoldpedia/chrome/site/Arnold-5.2.2.1/doc/api/group__ai__ray.html for all the ray type definitions in hex.

Example Scene

Below is an example of an Attribute Wrangle (primitive) node used to increase the number of subdivision iterations on a sphere within an Alembic file using the following VEX expression:

```
i@ar_subdiv_iterations = 3;
```



Subdivision of sphere increased to 3 using Attribute Wrangle -> Alembic

A scene file is available here.

Operator Graph

If you want to connect a *Procedural Operator* for *alembic* nodes, then pick the operator in the Arnold Properties for the OBJ node, under Arnold | Shapes | Procedurals | Operator Graph parameter. This will be applied to all of the Alembic *procedurals* created within the geometry context.