Resource

USD Reference

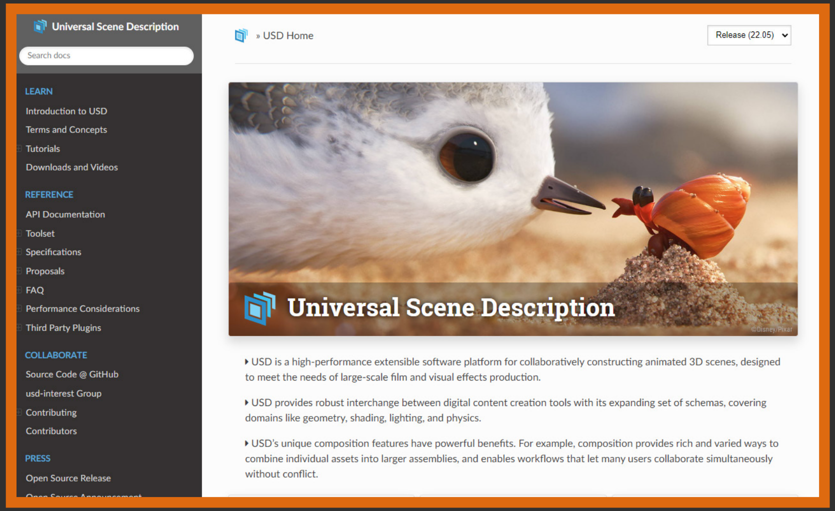
USD/Solaris

Houdini Insight

Introduction / Resource Links

This guide is designed to help you as you begin to work in USD. It will help you become familiar with the interface of Solaris, bring you up to speed on terminology, and begin to give you an understanding of how USD is being created behind-the-scenes.

Enjoy!



USD is a high-performance extensible software platform for collaboratively constructing animated 3D scenes, designed to meet the needs of large-scale film and visual effects production.

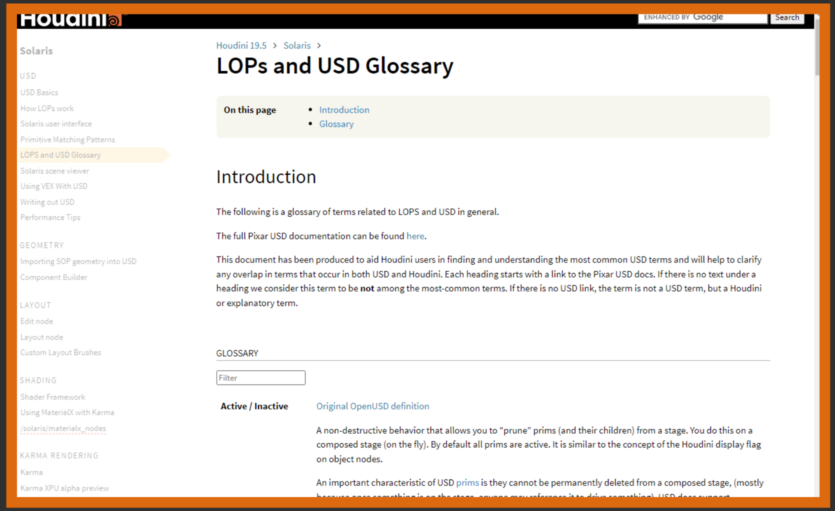
USD provides robust interchange between digital content creation tools with its expanding set of schemas, covering domains like geometry, shading, lighting, and physics.

USD's unique composition features have powerful benefits. For example, composition provides rich and varied ways to combine individual assets into larger assemblies, and enables workflows that let many users collaborate simultaneously without conflict.

Pixar USD Documentation

Link

This is the most in-depth resource on all of the technical details of USD. Here you can find terms & concepts, API documentation, source code links, and more. This page is on the Pixar website, and is maintained by the official development team.



LOPs and USD Glossary

Introduction

The following is a glossary of terms related to LOPS and USD in general.

The full Pixar USD documentation can be found here.

This document has been produced to aid Houdini users in finding and understanding the most common USD terms and will help to clarify any overlap in terms that occur in both USD and Houdini. Each heading starts with a link to the Pixar USD docs. If there is no text under a heading we consider this term to be **not** among the most-common terms. If there is no USD link, the term is not a USD term, but a Houdini or explanatory term.

GLOSSARY

Active / Inactive Original OpenUSD definition

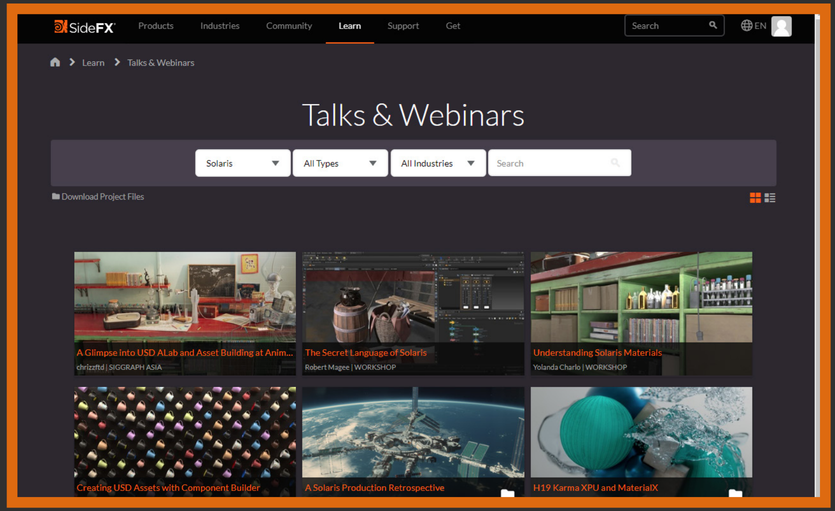
A non-destructive behavior that allows you to "prune" prims (and their children) from a stage. You do this on a composed stage (on the fly). By default all prims are active. It is similar to the concept of the Houdini display flag on object nodes.

An important characteristic of USD prims is they cannot be permanently deleted from a composed stage, (mostly

SideFX LOPs & USD Glossary

Link

Here you can find the terms & concepts as they relate directly to Houdini and Solaris. This page is within the Houdini Online Documentation, so you can find links to many other topics in Solaris from this page.


SideFX
Products
Industries
Community
Learn
Support
Get
Search
EN
Learn > Talks & Webinars
Talks & Webinars
Solaris
All Types
All Industries
Search
Download Project Files
A Glimpse into USD ALab and Asset Building at Anim...
chrizzftd | SIGGRAPH ASIA
The Secret Language of Solaris
Robert Magee | WORKSHOP
Understanding Solaris Materials
Yolanda Charlo | WORKSHOP
Creating USD Assets with Component Builder
A Solaris Production Retrospective
H19 Karma XPU and MaterialX

SideFX Talks & Webinars

Link

This link brings you to our page for all official SideFX talks and webinars. If you use the first dropdown on the top, you can filter by "Solaris" to get just the relevant items on Solaris or USD.

Important Terminology

Solaris

Marketing term for the Houdini scene building, lighting, layout, and pipeline toolset. It's where we can render through Hydra delegates.

Stage

Name of the Houdini context in which we use LOP nodes, and is also the USD term for the root of our scene.

LOPs

(lighting operators)

Name for the group of nodes/tools that are used to work with USD in Solaris.

USD

"Universal Scene Description"

USD is a scene description format that is created and maintained by Pixar. It enables workflows that allow simulataneous collaboration without conflict, provides robust interchange between DCCs, and is designed to meet the needs of large-scale film and VFX production.

USD Primitive

Almost every object that is added to our USD scene is considered a "primitive". This can include objects such as: lights, geometry, cameras, shaders, and render settings. In short, any "thing" that we add into our scene.

USD Formats / Extensions

Formats

Binary (a.k.a. USD Crate)

These files yield faster file reads and smaller file sizes, but are not human readable. They are best used in large scenes with high numbers of geometric objects.

ASCII

These files yield slower file reads and larger file sizes, but come with the benefit of being human readable. They could be used for small scenes or assets, and are helpful to debug issues.

Extensions

Each extension has rules for what type of format can be saved within it.

.usd	.usda	.usdc
Binary/ ASCII	ASCII	Binary

.usd Flexibility

In the case of the .usd extension, it is saved as Binary ("Crate") by default. If you would like to save your .usd file as ASCII you need to append this string to the end of the file path in the USD ROP:

":SDF_FORMAT_ARGS:format=usda"

UI / Scene Graph Tree Pane

The Scene Graph Tree pane shows the stage/layers in the output of the selected LOP node. (It does not follow the display flag). This makes it easy to click around the network seeing how nodes affect the contents of the stage.

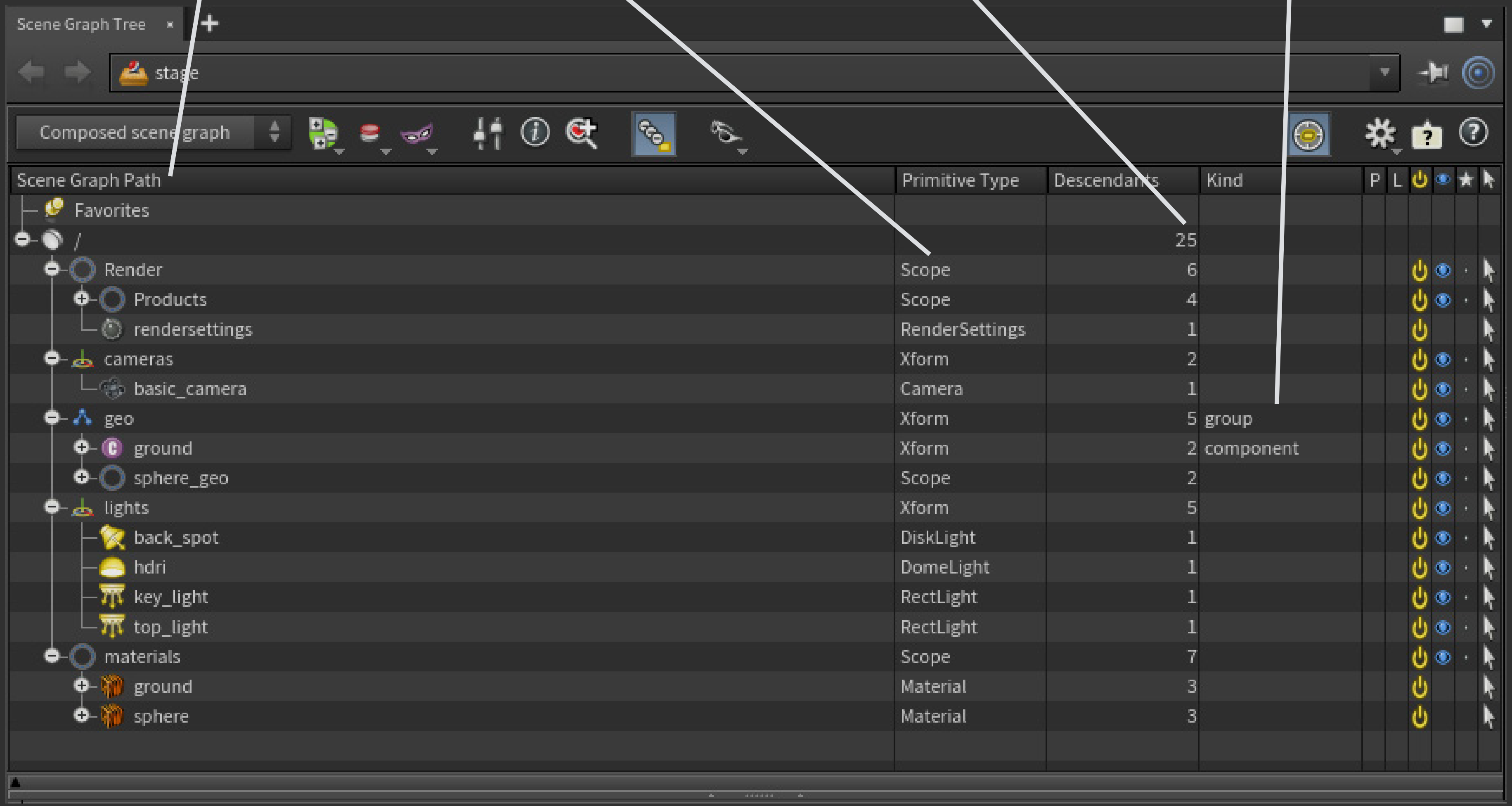
Interactive changes you make in this pane (such as activating/deactivating prims, showing/hiding prims, and solo-ing) apply to the view override layer. Changes in this layer are not saved to disk in USD files or rendered.

Displays the scene graph path of each primitive (hierarchy)

Displays each primitive's "type"

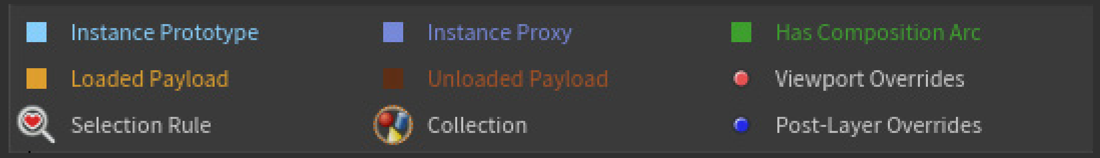
Displays the number of child primitives

Displays each primitive's "kind"



Scene Graph Path	Primitive Type	Descendants	Kind
Favorites			
/		25	
Render	Scope	6	
Products	Scope	4	
rendersettings	RenderSettings	1	
cameras	Xform	2	
basic_camera	Camera	1	
geo	Xform	5	group
ground	Xform	2	component
sphere_geo	Scope	2	
lights	Xform	5	
back_spot	DiskLight	1	
hdri	DomeLight	1	
key_light	RectLight	1	
top_light	RectLight	1	
materials	Scope	7	
ground	Material	3	
sphere	Material	3	

COLOR / ICON LEGEND



🟦 Instance Prototype	🟪 Instance Proxy	🟩 Has Composition Arc
🟧 Loaded Payload	🟫 Unloaded Payload	● Viewport Overrides
Selection Rule	Collection	● Post-Layer Overrides

TOOLBAR BUTTONS

This menu contains items for saving/restoring the expansion state of the scene graph tree. The expansion state is saved to the LOP Network node, and so to the .hip file.

This menu contains items for saving/restoring override states. Viewport overrides are the draw mode, active, display, and solo columns of the scene graph tree, which affect the display of the stage in the viewport. Like the expansion states, the override states are saved with the LOP Network.

This menu provides high level control over the USD population mask, payload loading, and layer muting features.

Opens the parameter dialog for the current LOP Network. This button basically exists to provide access to the parameter dialog for the /stage LOP Network, which is normally not accessible through the UI in a way that would allow opening its parameter dialog.

Open a pinned info window for the current LOP node.

Opens a dialog to create a new saved selection rule.

When switched on, expands the scene graph tree to show the currently selected primitive.

Opens a drop-down dialog to toggle which primitives are visible in the scene graph tree.

When switched on, the scene graph and primitive selection follows the current node.

Opens a drop-down dialog to toggle which columns are shown in the scene graph tree.

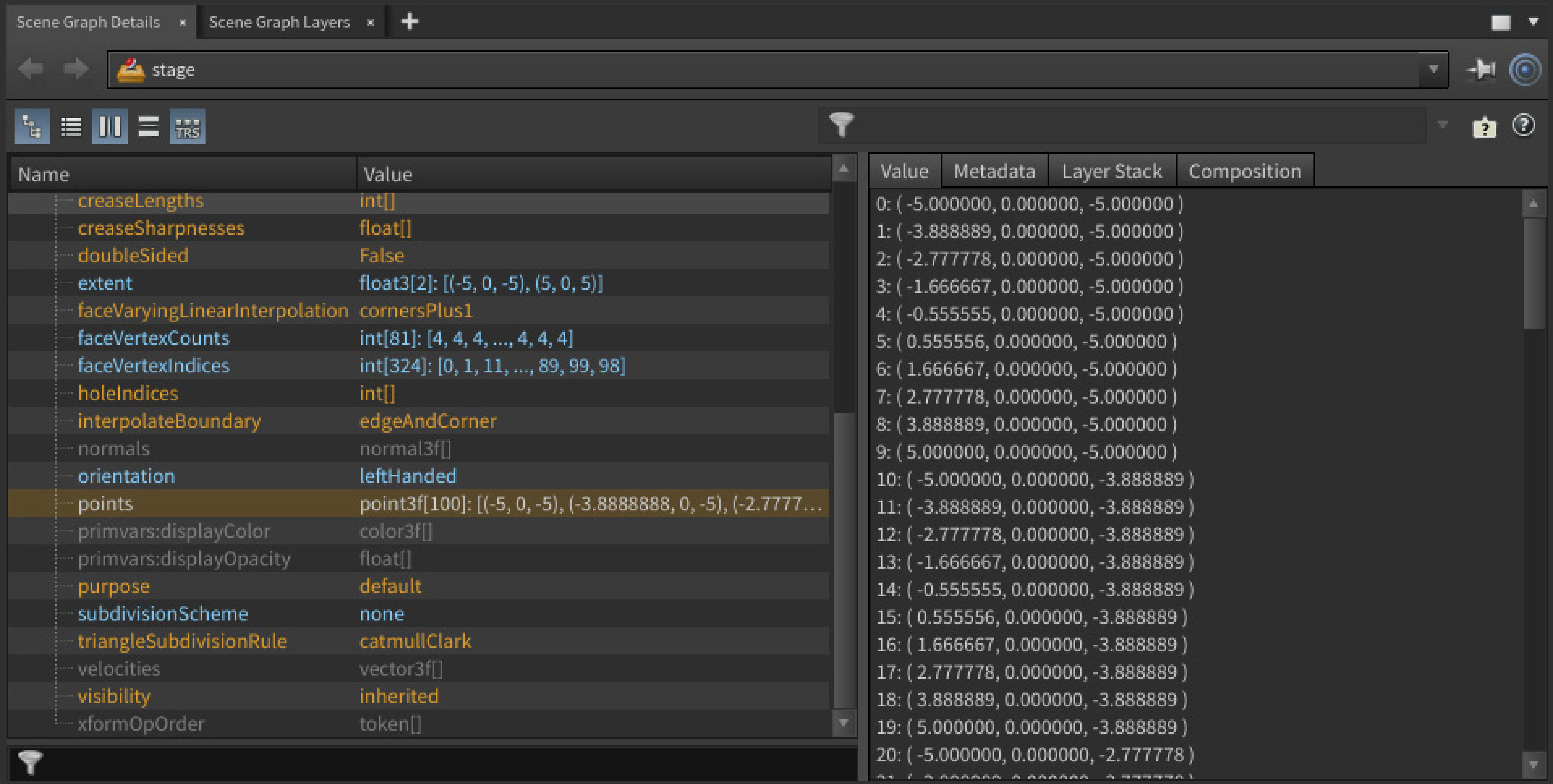
Opens the color/icon legend for the scene graph tree.

PRIMITIVE TOGGLES

⏻ Active	👁 Visible	👁 Visible (animated)	· Nothing is solo
· Deactivated	◡ Hidden	◡ Hidden (animated)	× another prim is solo
			★ This prim is solo

UI / Scene Graph Details Pane

The Scene Graph Details pane shows attributes, values, metadata, layer, and composition info. for the currently selected primitive in the Scene Graph Tree pane.



Name	Value
creaseLengths	int[]
creaseSharpnesses	float[]
doubleSided	False
extent	float3[2]: [(-5, 0, -5), (5, 0, 5)]
faceVaryingLinearInterpolation	cornersPlus1
faceVertexCounts	int[81]: [4, 4, 4, ..., 4, 4, 4]
faceVertexIndices	int[324]: [0, 1, 11, ..., 89, 99, 98]
holeIndices	int[]
interpolateBoundary	edgeAndCorner
normals	normal3f[]
orientation	leftHanded
points	point3f[100]: [(-5, 0, -5), (-3.8888888, 0, -5), (-2.7777…
primvars:displayColor	color3f[]
primvars:displayOpacity	float[]
purpose	default
subdivisionScheme	none
triangleSubdivisionRule	catmullClark
velocities	vector3f[]
visibility	inherited
xformOpOrder	token[]

Value | Metadata | Layer Stack | Composition

```
0: ( -5.000000, 0.000000, -5.000000 )
1: ( -3.888889, 0.000000, -5.000000 )
2: ( -2.777778, 0.000000, -5.000000 )
3: ( -1.666667, 0.000000, -5.000000 )
4: ( -0.555555, 0.000000, -5.000000 )
5: ( 0.555556, 0.000000, -5.000000 )
6: ( 1.666667, 0.000000, -5.000000 )
7: ( 2.777778, 0.000000, -5.000000 )
8: ( 3.888889, 0.000000, -5.000000 )
9: ( 5.000000, 0.000000, -5.000000 )
10: ( -5.000000, 0.000000, -3.888889 )
11: ( -3.888889, 0.000000, -3.888889 )
12: ( -2.777778, 0.000000, -3.888889 )
13: ( -1.666667, 0.000000, -3.888889 )
14: ( -0.555555, 0.000000, -3.888889 )
15: ( 0.555556, 0.000000, -3.888889 )
16: ( 1.666667, 0.000000, -3.888889 )
17: ( 2.777778, 0.000000, -3.888889 )
18: ( 3.888889, 0.000000, -3.888889 )
19: ( 5.000000, 0.000000, -3.888889 )
20: ( -5.000000, 0.000000, -2.777778 )
```

TOOLBAR BUTTONS

Tree View

Show prims and attributes as a tree where the attributes of each selected prim are listed as children of the prim.

List View

Show prims and attributes as a spreadsheet, where the selected prims are rows, and attributes on the prims are columns.

Side-by-side

Show the primary and secondary subpanes side-by-side.

Top-bottom

Show the primary and secondary subpanes above-and-below.

TRS Decompose Transforms

In the spreadsheet view, show Translate, Rotate, and Scale as separate columns.

Filter

Type in this field to filter attributes when the pane is displayed as list view.

ATTRIBUTE COLOR LEGEND

■ No Value	■ Fallback Value	■ Default Value
■ Time Samples	■ Value Clips	■ Relationship
■ Computed Value		

SECONDARY PANE TABS

Value

Displays a list of values. For attributes that are unique for each elements there is an index specified before the value. (“###:”)

Metadata

A tree view of the selected attribute’s metadata. Some of the values show what type of data (float, vec, etc.), what geometry “level” the attribute is stored on, or a line from the USD documentation.

Layer Stack

A table view of the layers that hold an opinion relating to the selected attribute. Each row in the table relates to a USD layer, and the colums show the layer name and primitive path that holds the opinion.

Composition

A tree view that shows all of the composition related to the current selected primitive in the Scene Graph Tree. The values in this tab do not change when selecting attributes in the primary subpane.

Primitive Terminology

Leaf Primitive

A leaf primitive is the end of a branch of the scene graph tree. Another way of describing this is that it doesn't have any child primitives.

Primitive Types

Primitive types define what a primitive is. This definition gives the USD software a concrete idea of what this primitive will be used for. Some examples are: Mesh, Light, OpenVDB, Render Settings, and Point Instancer.

Primitive Kind

- a method for defining hierarchy on primitives

Assembly	• An important group • Usually a published asset or reference to one
Group	• Can contain components, groups, or assemblies
Component	• Cannot contain groups or assemblies • Only able to contain Subcomponents
Subcomponent	• An "important" primitive inside a component (e.g., a door pivot)

Kind Example

A hierarchy in the scene graph tree could look like the following:

```
/Models ← group
  /Characters ← group
    /Lady ← assembly
      /Skin ← component
        /mesh
      /Purse ← component
        /mesh
    /Dog ← assembly
      /Skin ← component
        /mesh
      /Collar ← component
        /mesh
  /Props ← group
```

Collections & Selection Rules

Collections

Collections are created as primitives in the scene graph in order for you to save them to disk in a .usd file, and then easily recall them later in your scene or by another artist/department. In this way, you can think of “Collections” as a USD version of Houdini “Groups”.

Collections can be saved to the “/collections” primitive (default), or to any other primtive in the scene graph.

Selection Rules

Selection rules are similar to collections in that they describe groups of primitives in your scene. However, these are unique to Houdini and are only saved to the .hip file and can not be saved out to .usd files.

Primitive Matching Patterns[1]

Path Patterns

Match the primitive with the path `/Kitchen/Table/Leg1`:

```
/Kitchen/Table/Leg1
```

Match all children of `/Kitchen`:

```
/Kitchen/*
```

Match any children of `/Kitchen` whose names start with `Chair`:

```
/Kitchen/Chair*
```

Match any descendants of `/Kitchen` whose names are `Handle1,Handle2, Handle3, or Handle4`:

```
/Kitchen/**/Handle[1234]
```

Match any children of `/House` whose names are `Room` followed by a single character (such as `RoomA`):

```
/House/Room?
```

Collection Syntax

Match any prims specified by the `KeyLights` collection on the `/collections` primitive:

```
%KeyLights
```

Match any prims specified by the `KeyLights` collection on the `/House/LivingRoom` primitive:

```
/House/LivingRoom.collection:KeyLights
```

or

```
%/House/LivingRoom/KeyLights
```

Selection Rule Syntax

To match all primitives specified by the `/rules/Characters` selection rule:

```
%rule:/rules/Characters
```

or

```
%rule(/rules/Characters)
```

[1] For an in-depth reference of all primitive matching patterns please go to: sidefx.com/docs/houdini/solaris/pattern.html

Importing from SOPs

Scene Import

The Scene Import LOP is a quick way to bring the entire “/obj” context onto the stage.

We can use the *Objects*, *Force Objects*, or *Exclude Objects* parameters to include or exclude specific objects from importing onto the stage. There are a few “Scene Import” variations in the TAB menu that can quickly bring in all objects, cameras, lights, or materials.

SOP Import

The SOP Import LOP is a way to load geometry onto the stage from a specific location in SOPs.

Since this node requires a SOP path parameter to be set, it is only able to import geometry to the stage from one individual node. This method has limited support for material importing, and is primarily a geometry import method.

SOP Create

The SOP Create LOP is a way to create geometry directly on the stage.

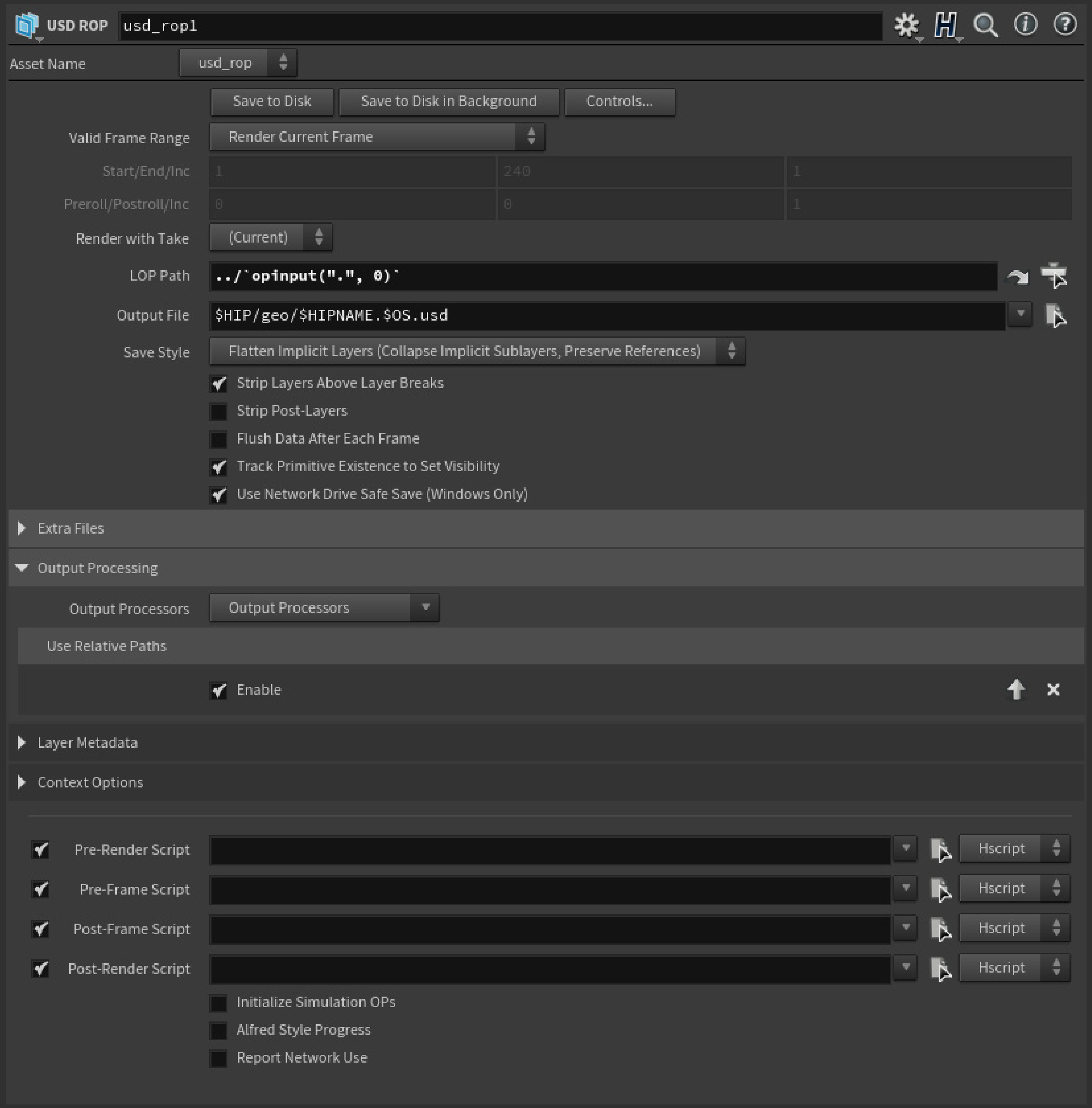
The node contains a SOP network internally and can be used to create anything you are able to in SOPs. Since it contains its own network, you do not need to go back to the Object context in order to modify your geometry. This can be useful for small setups that will only be used in Solaris/LOPs.

Saving USD files to disk

USD ROP

The USD ROP allows us to write out both static and animated USD files. It also gives us options as to how to save out our layers under the “Save Style” dropdown.

One incredibly powerful section of parameters are under “Output Processing”. These processors allow for a number of different operations to be done while writing the file. Some of these processors include: making all paths relative (set by default), forcing all explicit file path extensions to be the same, and adjusting the directory structure.



USD ROP usd_rop1

Asset Name: usd_rop

Save to Disk | Save to Disk in Background | Controls...

Valid Frame Range: Render Current Frame

Start/End/Inc: 1 | 240 | 1

Preroll/Postroll/Inc: 0 | 0 | 1

Render with Take: (Current)

LOP Path: ``../`opinput(".", 0)` ``

Output File: `$HIP/geo/$HIPNAME.$OS.usd`

Save Style: Flatten Implicit Layers (Collapse Implicit Sublayers, Preserve References)

- [✓] Strip Layers Above Layer Breaks
- [ ] Strip Post-Layers
- [ ] Flush Data After Each Frame
- [✓] Track Primitive Existence to Set Visibility
- [✓] Use Network Drive Safe Save (Windows Only)

▸ Extra Files

▾ Output Processing

Output Processors: Output Processors

Use Relative Paths

- [✓] Enable

▸ Layer Metadata

▸ Context Options

- [✓] Pre-Render Script — Hscript
- [✓] Pre-Frame Script — Hscript
- [✓] Post-Frame Script — Hscript
- [✓] Post-Render Script — Hscript
- [ ] Initialize Simulation OPs
- [ ] Alfred Style Progress
- [ ] Report Network Use