



Next Steps: Houdini Procedural Modeling

M04: Creating Buildings Part 2 of 2

Ari Danesh
ari@sidefx.com

**SIDE EFFECTS
SOFTWARE**



Quick Review of Module 03

**SIDE EFFECTS
SOFTWARE**

Quick Review of Module 03

Create Digital Asset for Window

Create Foot Prints for Buildings

Create a shell (polyextrude) for Building

Clip extruded Building into three sections

- ▶ First Floor

- ▶ Floors

- ▶ Parapet

UV First Floor and Create Group for later Material

**SIDE EFFECTS
SOFTWARE**

Quick Review of Module 03 (cont.)

Take the Shell of the floors and feed it into a ForEach

Clip each floor individually

Clip the individual floor vertically in half

- ▶ The points will be used for vertical placement of the windows
- ▶ Delete corner points so windows will not overhang corners of building
- ▶ Resample the points to add more points for window placement

Create a Grid the same size and orientation as the window

- ▶ Copy the grid to each point of window placement
- ▶ Use the Hole SOP to cut out holes for window placement

Quick Review of Module 03 (cont.)

Object Merge the Window Object into the ForEach

- ▶ Copy each window into the holes

Jump back up to the Building level

PolyExtrude out the Parapet

Merge the different parts of the building

Apply Materials

Main Focus of Today

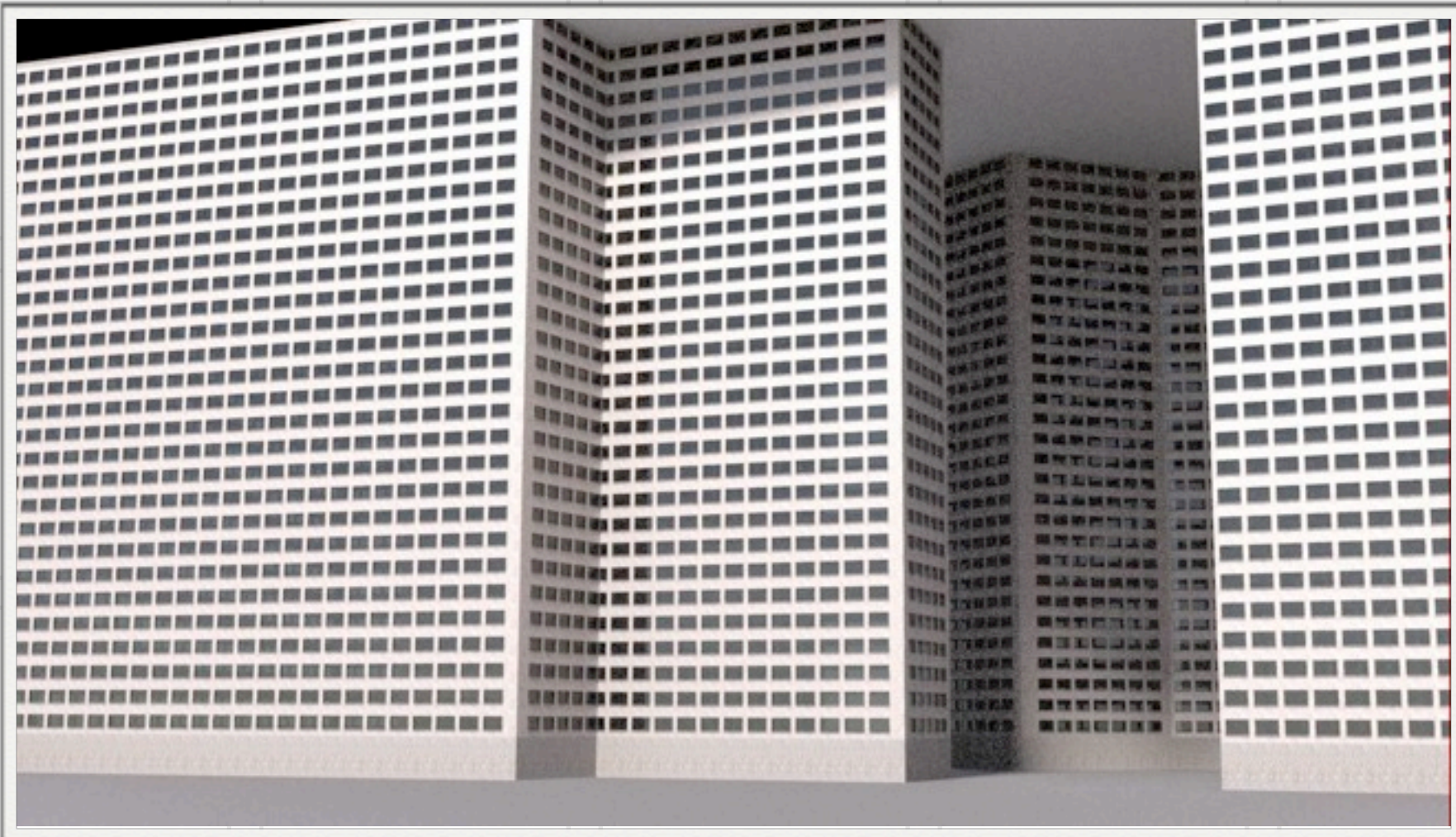
**Given any input building profile with the correct attributes,
construct a building on those profile(s)**

We will create those attributes today

**Once we have a profile with embedded attributes we should be
able to make an HDA that creates a building from the profile**

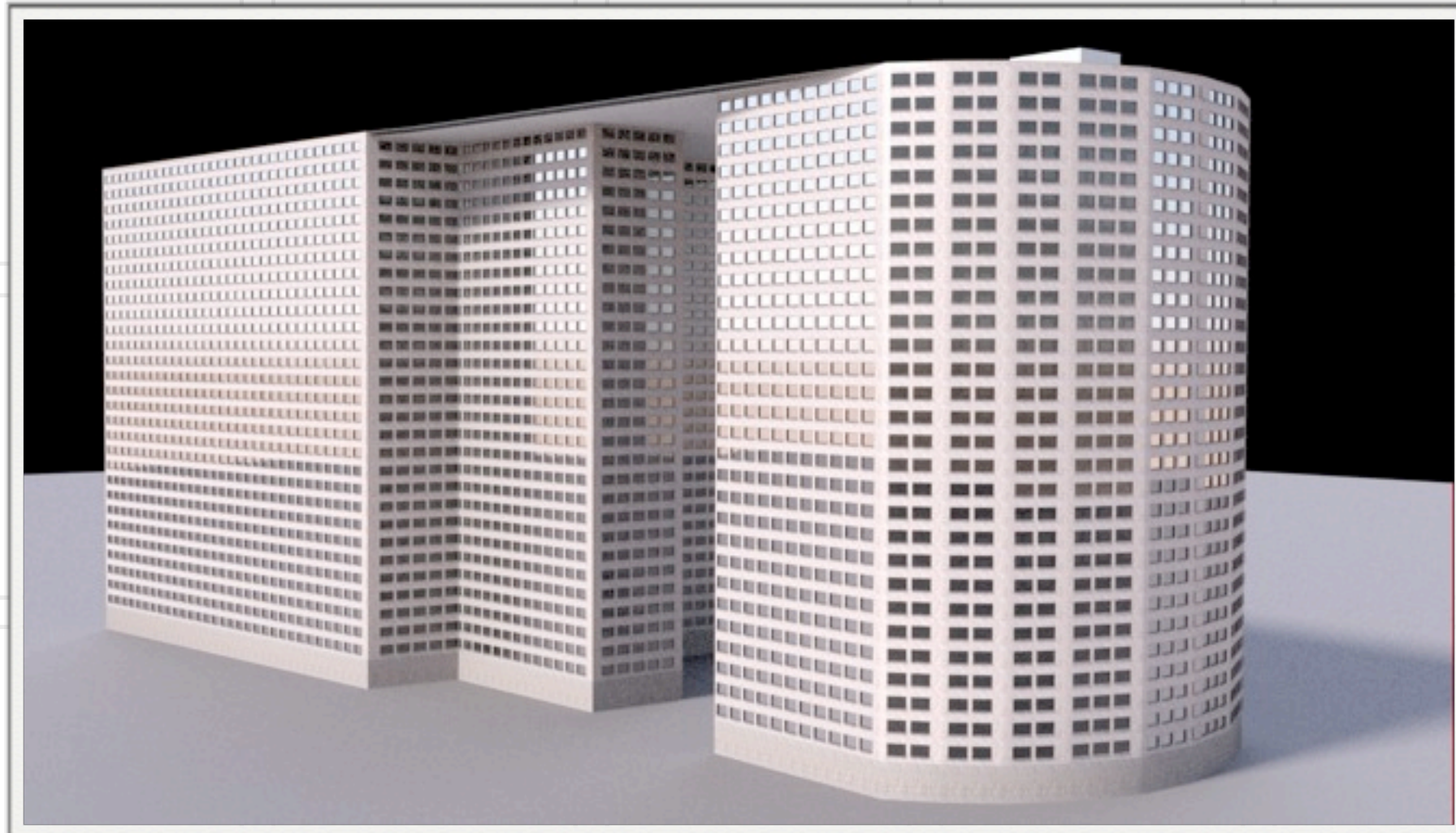
**SIDE EFFECTS
SOFTWARE**

Goal of this Module



- ▶ For a flat profile make primitive attributes to create a building
- ▶ Build a VOP network to add attributes if they do not already exist
- ▶ Create a Visualizer for the attributes either as display attributes or, or as simple geometry.
- ▶ Be able to modify the primitive attributes
- ▶ Wrap up the different tools into digital assets

Goal of this Module (cont.)



- ▶ Create a ForEach that will take any number of curves and create individual Buildings
- ▶ Create a Toggle for LOD
- ▶ Allow LOD to be automatically done with distance to camera
- ▶ Learn about the new 12.5 PointWrangle SOP

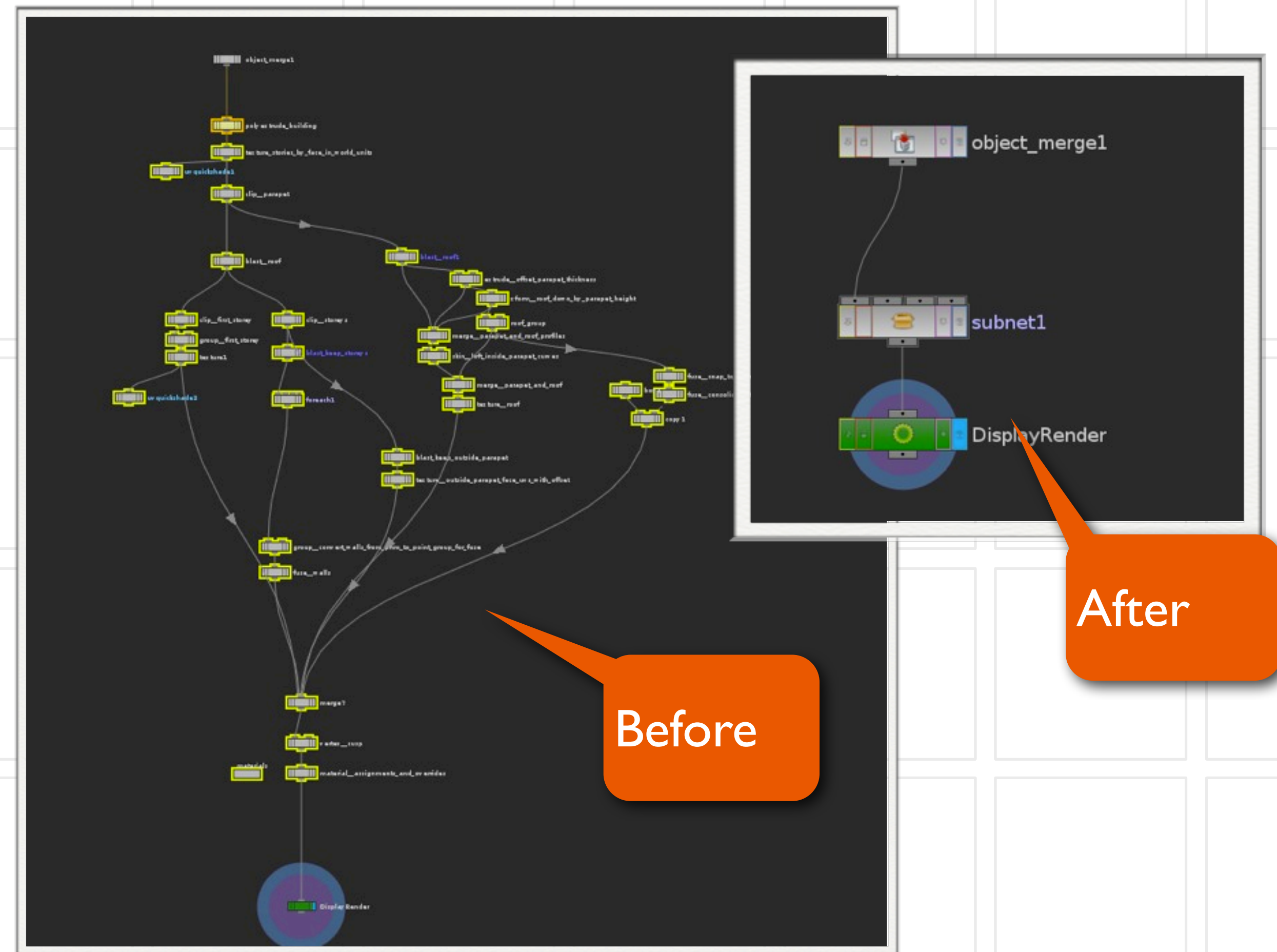


Creating a Building Generator Asset

**SIDE EFFECTS
SOFTWARE**

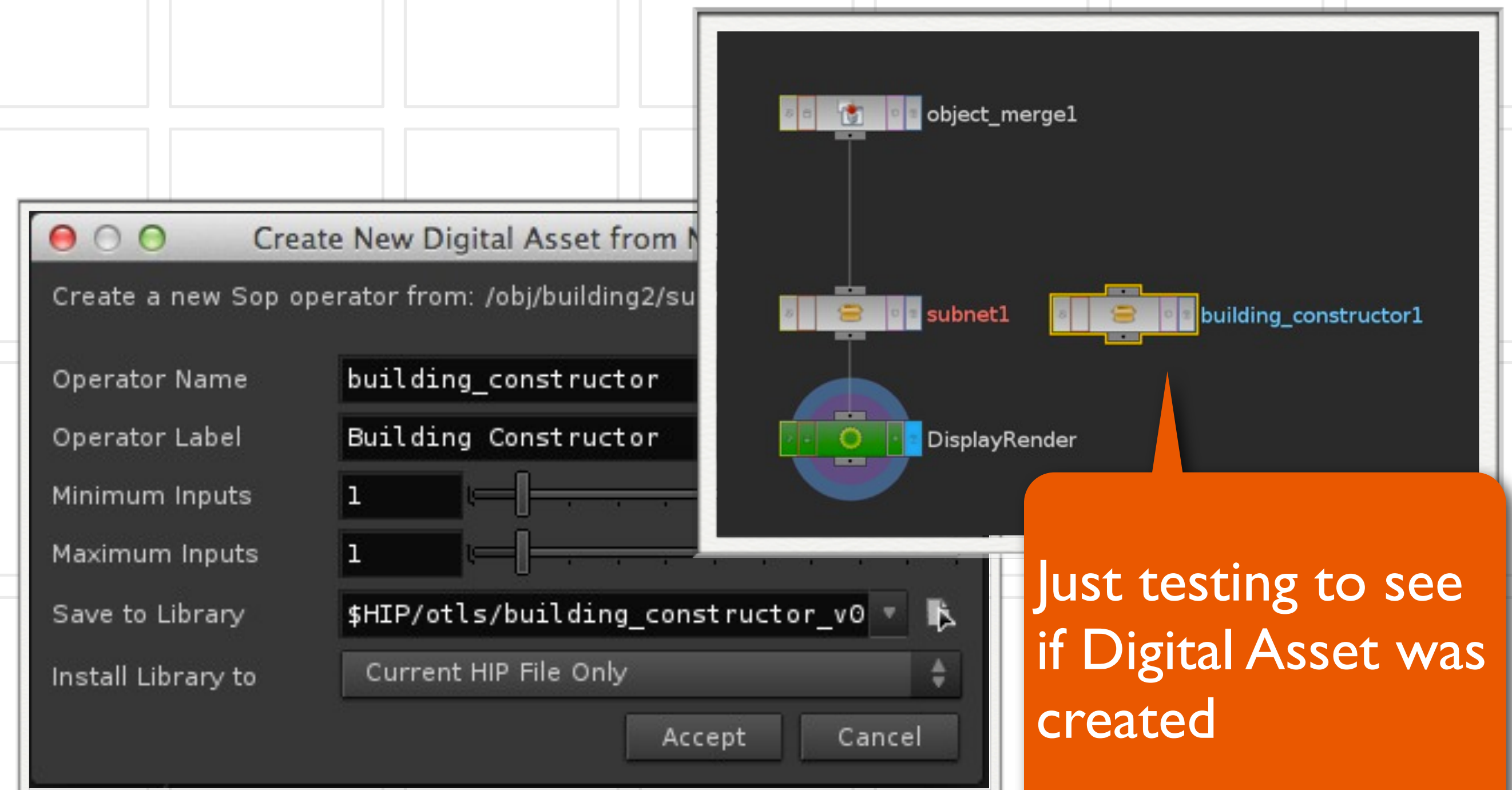
Converting the Building Network into the Building Generator Asset

- ▶ Dive into the Building Geometry
- ▶ The Object Merge will be the Input into our Asset
- ▶ The Display Render will be the Output of our Asset
- ▶ Highlight all the other nodes and make it into a subnet

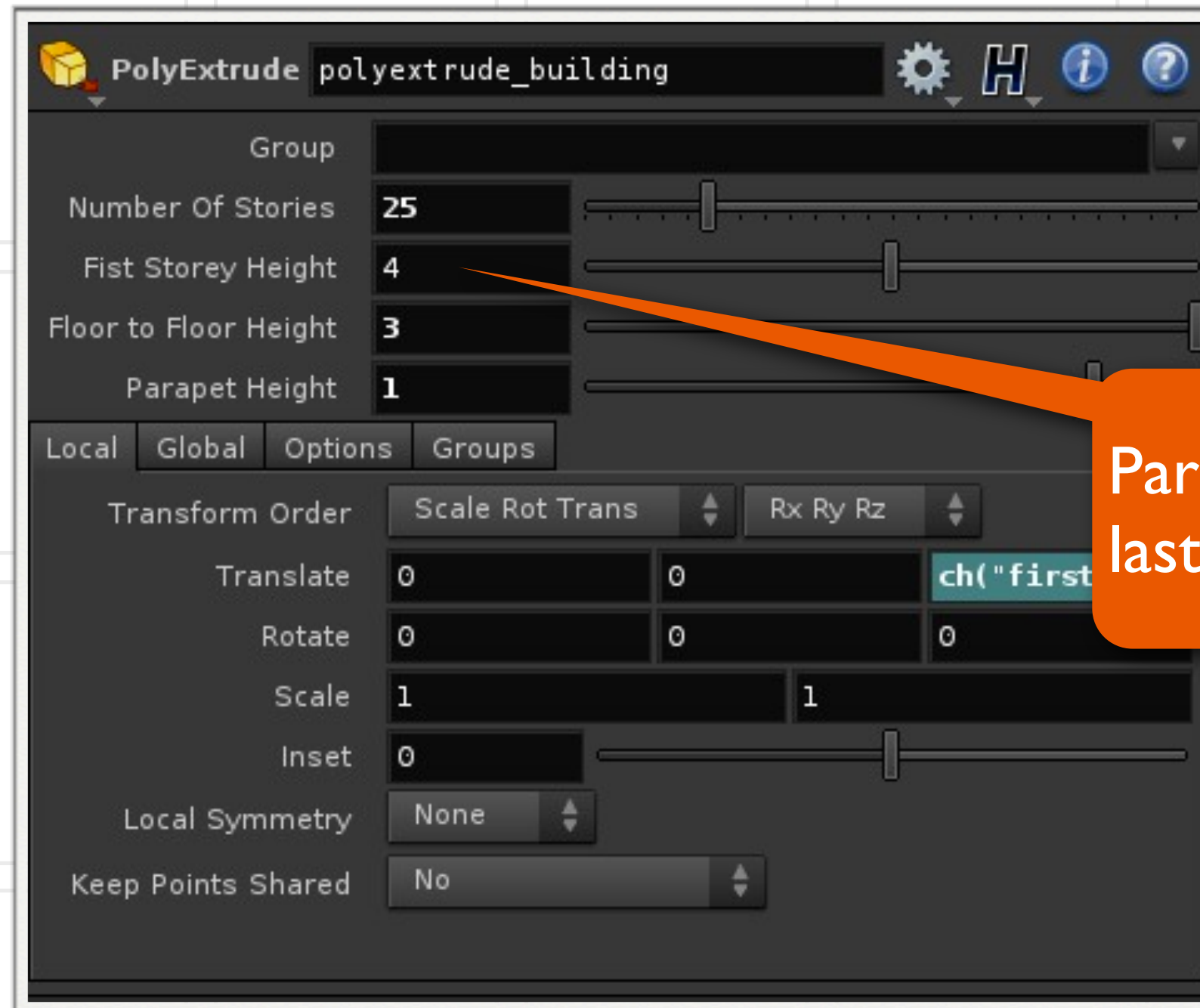


Converting the Building Network into the Building Generator Asset (cont.)

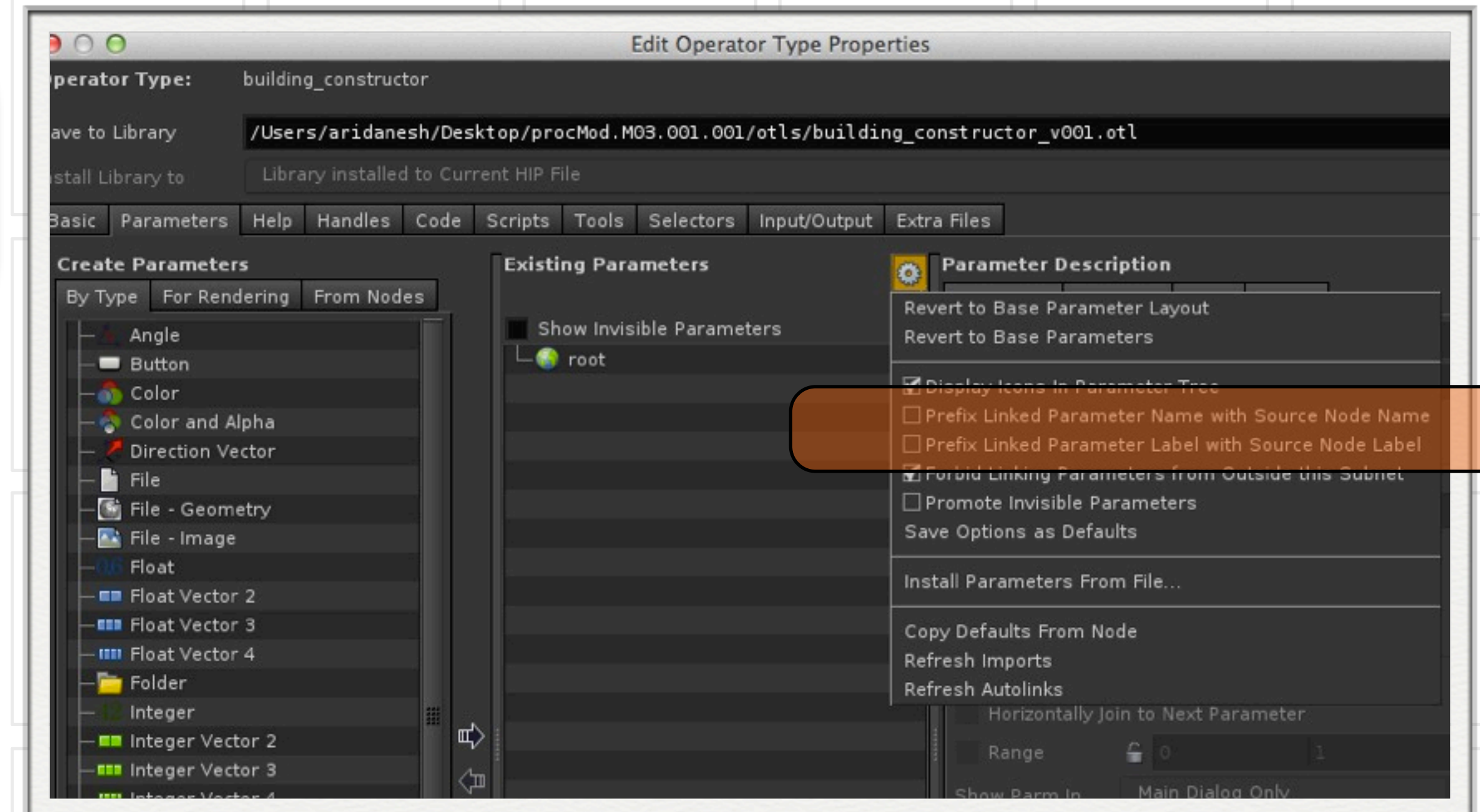
- ▶ Now let's make the subnetwork a Digital Asset
- ▶ Right Click on the SubNetwork and "Create Digital Asset"
- ▶ Name - building_constructor
- ▶ Label - Building Constructor
- ▶ Save to Library \$HIP/otls



The Parameters We Created in the PolyExtrude Node



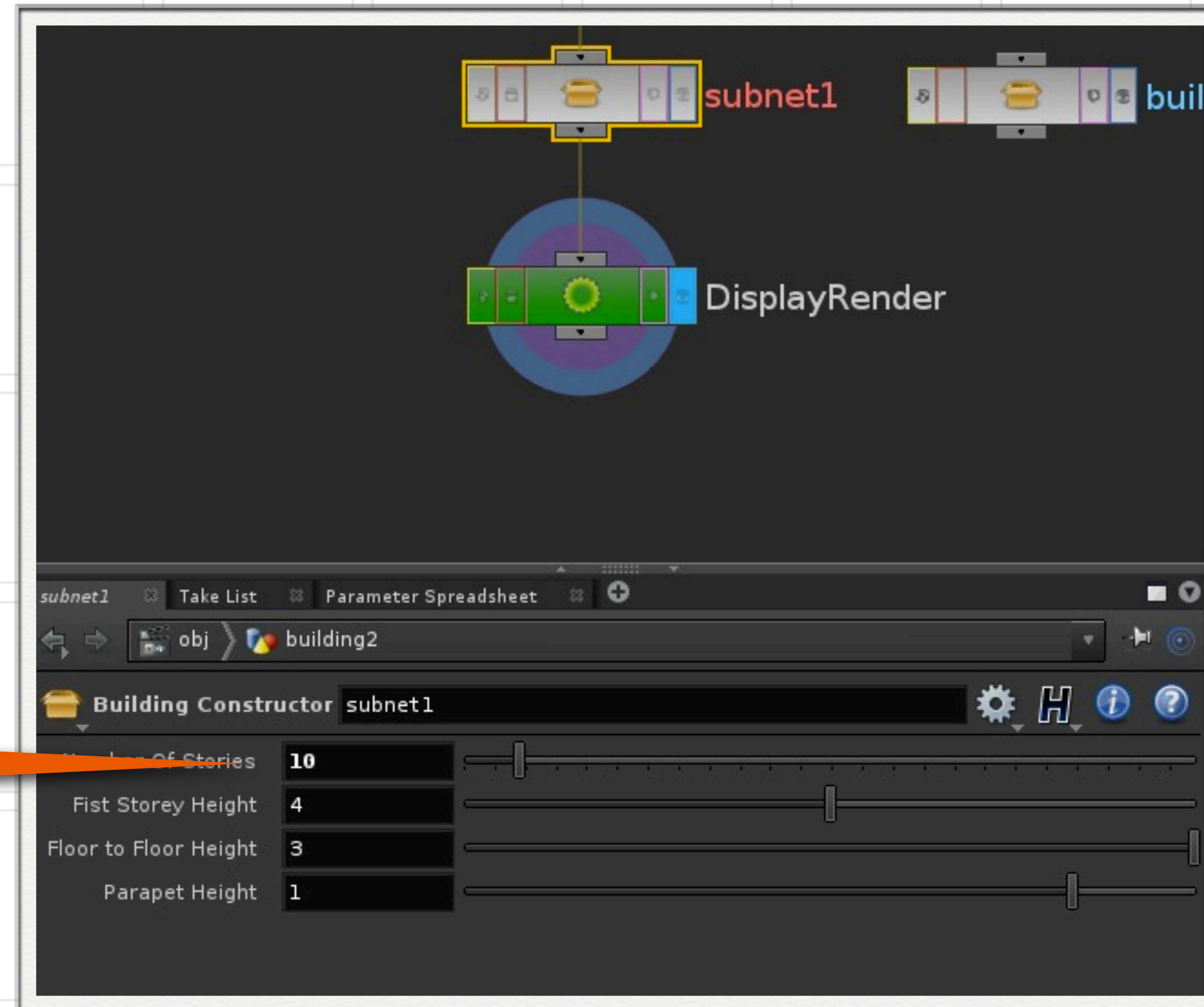
Parameters created last week



- ▶ Last Week we created several parameters inside the polyextrude node.
- ▶ We need to bring them up to the Asset Level
- ▶ Before we do that turn off in the Type Properties panel - Prefix Linked Parameter Name, Prefix Linked Parameter Label

Drag and Drop Parameters to the Type Properties Panel

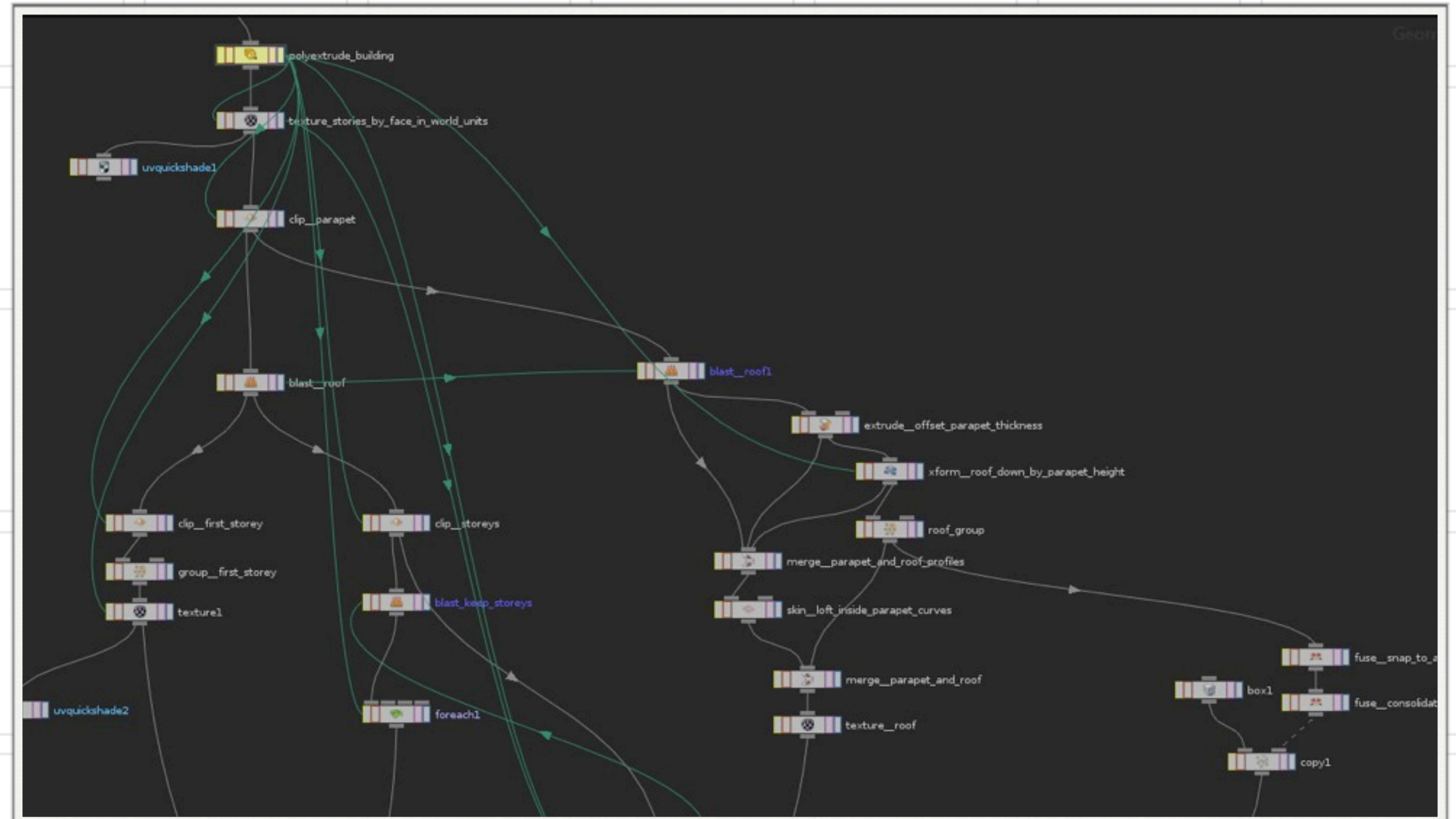
Drag and Drop the Parameters from the PolyExtrude to the Type Properties Panel and you will see this result



**SIDE EFFECTS
SOFTWARE**

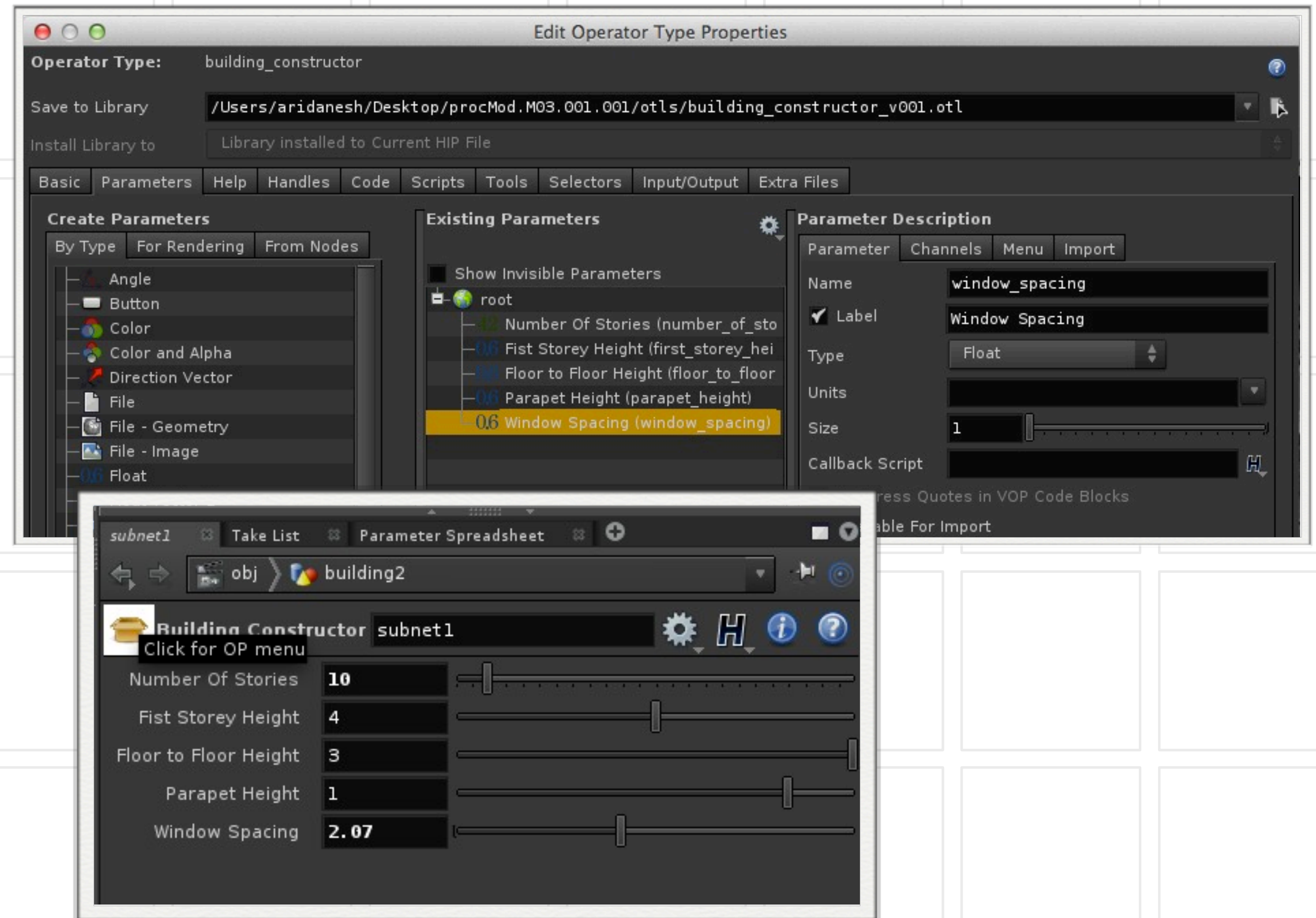
Making Sure We Got All the Parameters by Checking Dependencies

- ▶ Dive back into the digital asset
- ▶ Hover your mouse over the network view and press “d”
- ▶ This will open the display options for the network
- ▶ Click on the Network View Tab and then the Dependency Sub Tab
- ▶ Select the top three toggles
- ▶ Green Wires - Direct Links
- ▶ Red Wires - Indirect Links



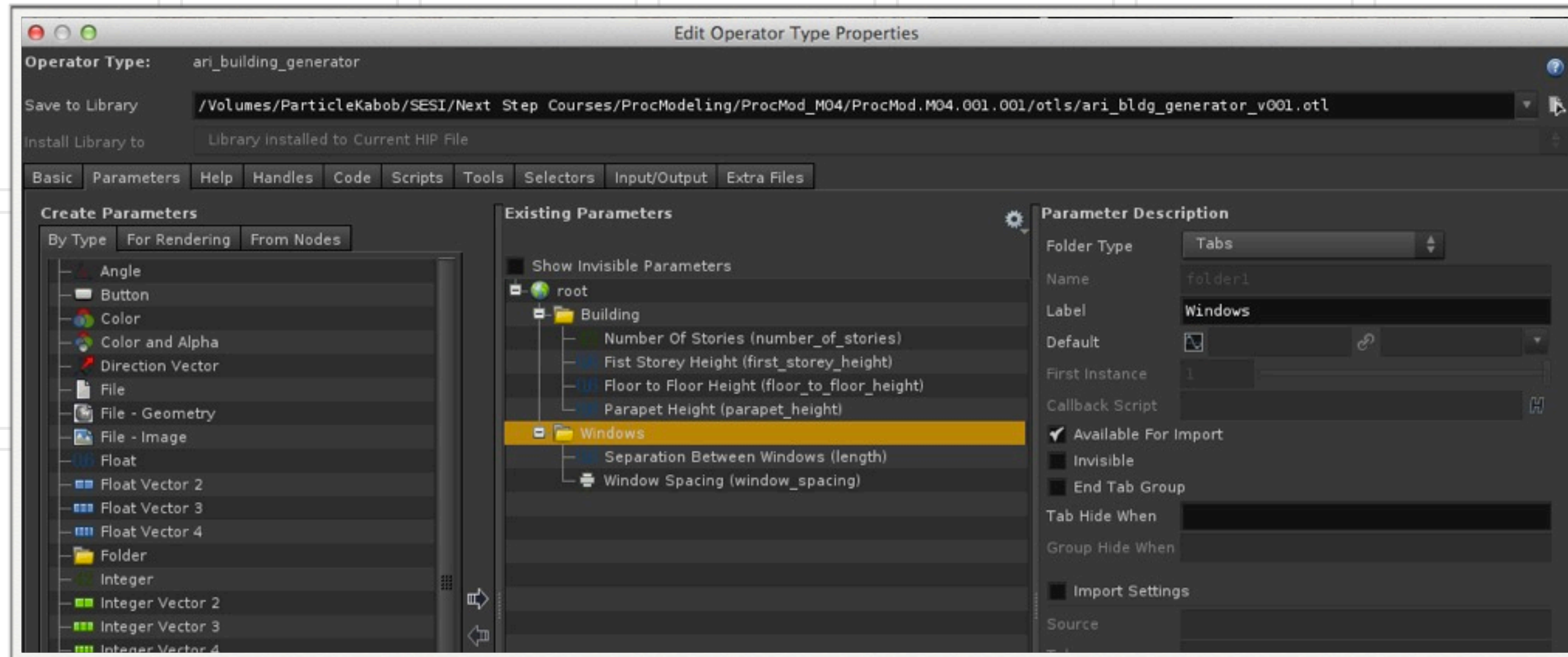
One More Parameter to Raise Up

- ▶ Dive into the For Loop
- ▶ In the Resample Node “resample_window_locations” drag and drop the length parameters into the Type Properties panel
- ▶ Give it a name - window_spacing
- ▶ Give it a label - Window Spacing

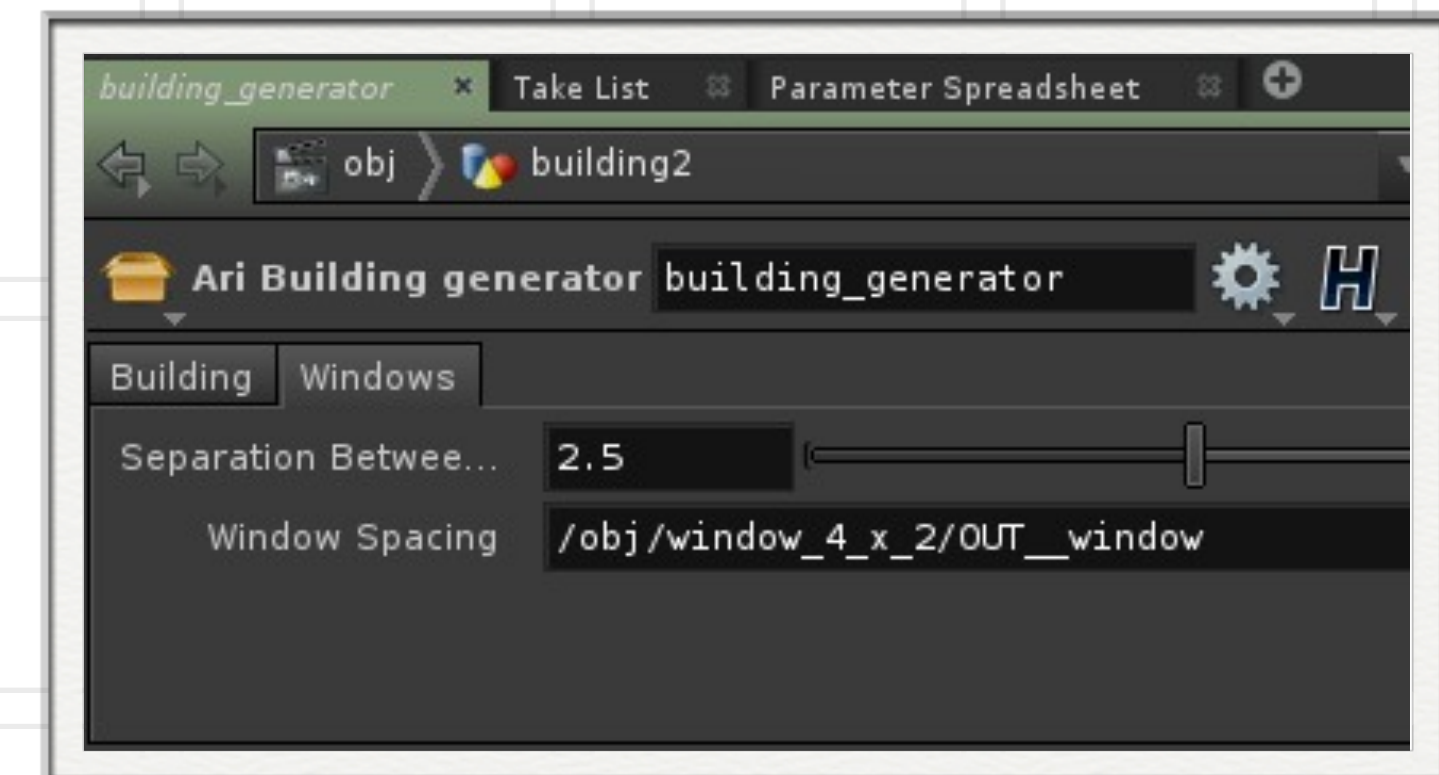
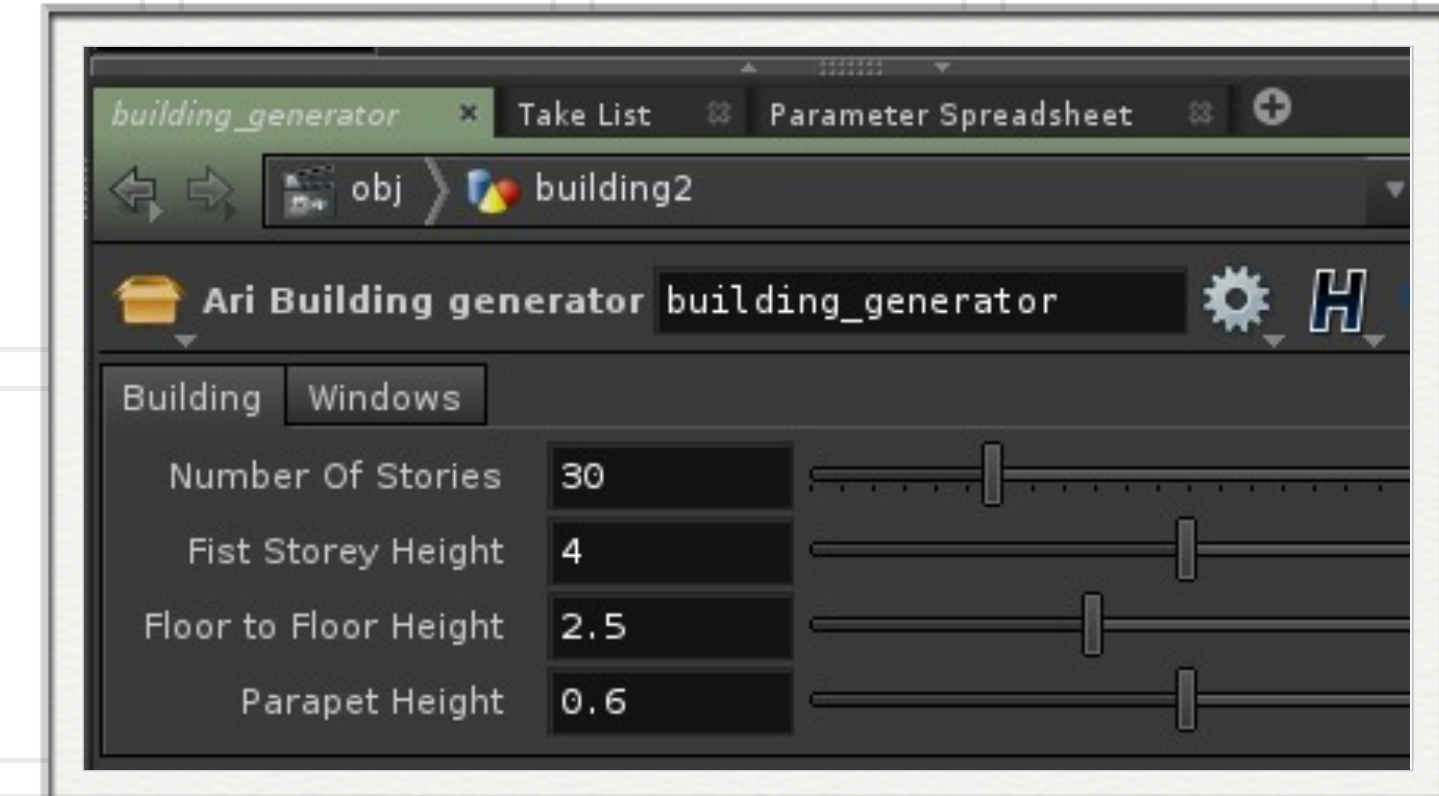


**SIDE EFFECTS
SOFTWARE**

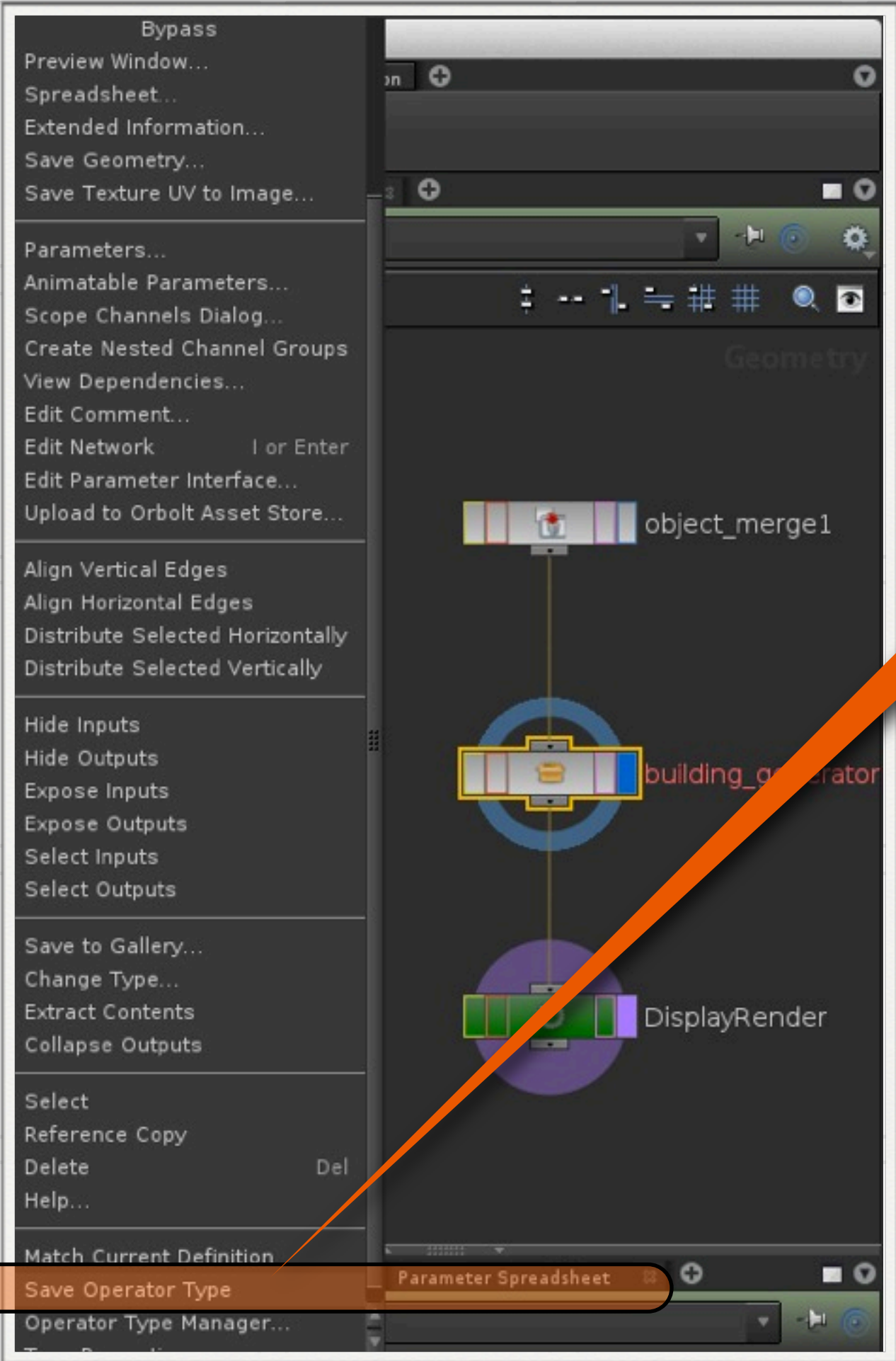
Clean Up the User Interface



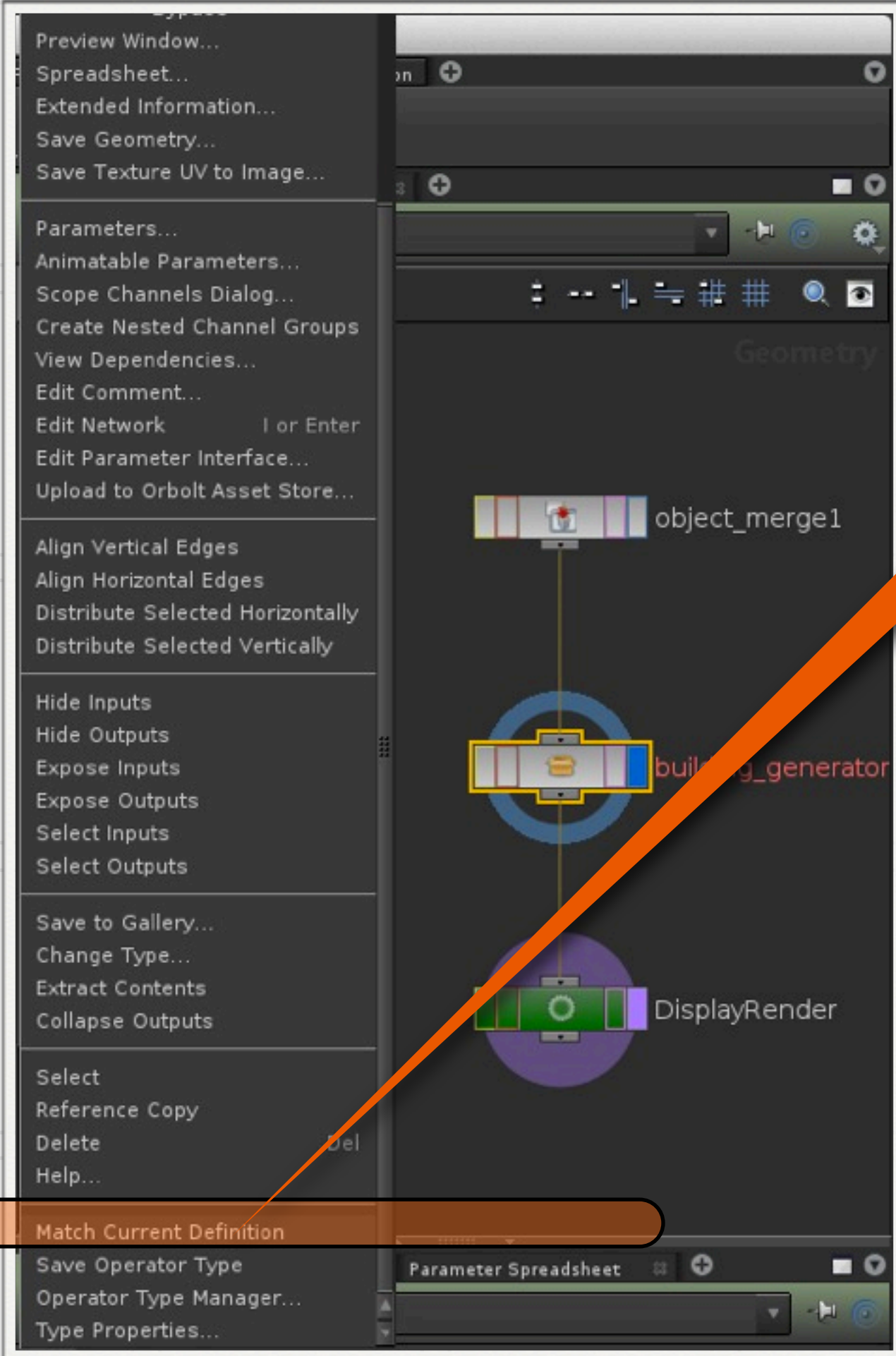
- ▶ Add Two folders to the User Interface (Building & Windows)
- ▶ Place the appropriate parameters in the folders
- ▶ Click Accept



Save and Match Current Definition



Save Operator Type...



then Match Current Definition



Making the Attributes for the Profiles

Now that the Building Generator is Done it is Time to Turn Our Attention to the Profiles

**SIDE EFFECTS
SOFTWARE**

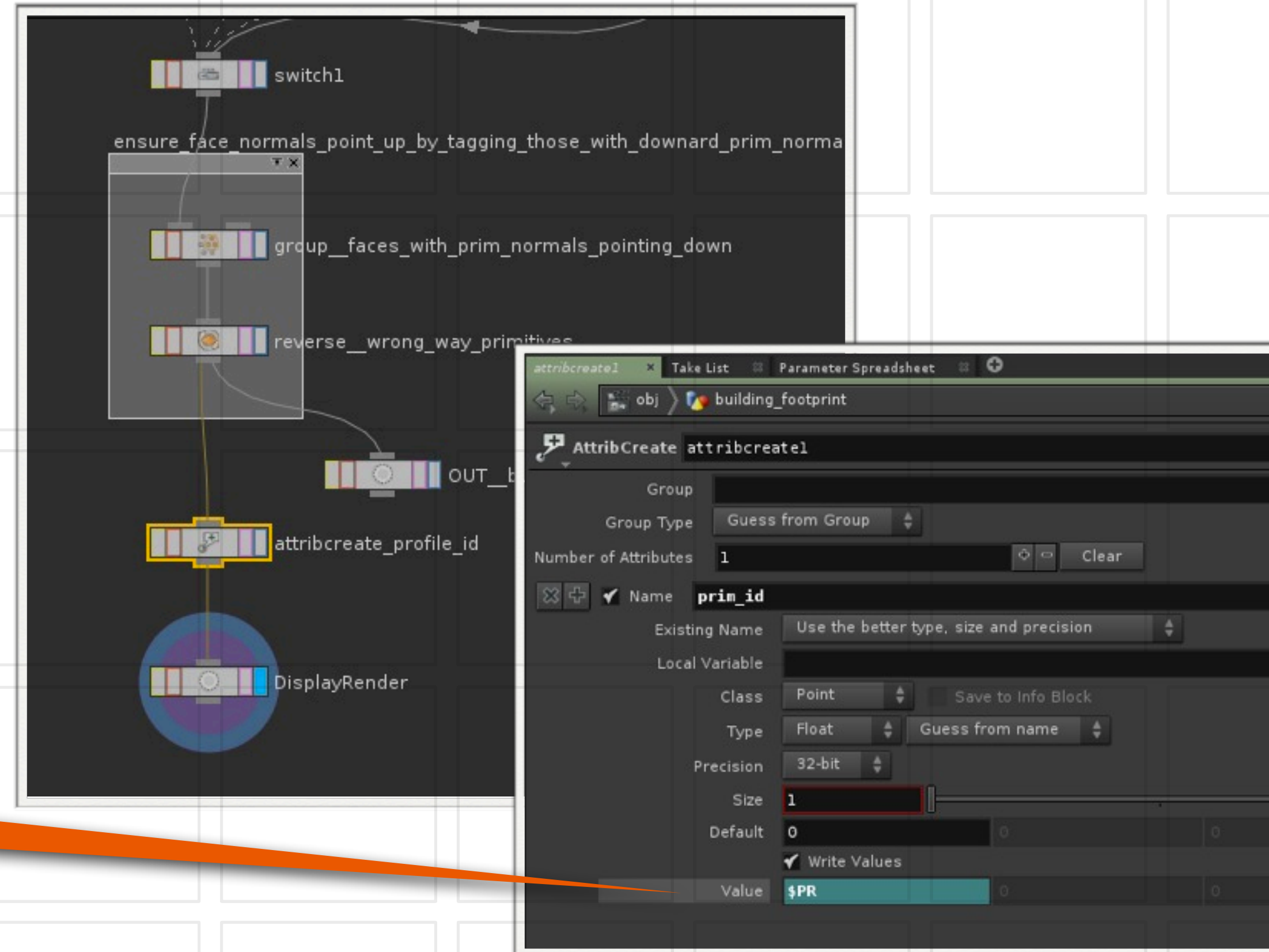
What Attributes Do We Want to Create?

Name	Type	Class
number_of_floors	int	point
floor_to_floor	float	point
parapet_height	float	point
window_spacing	float	point
window_corner_offset	float	point
profile_id	int	point
first_floor_height	float	point

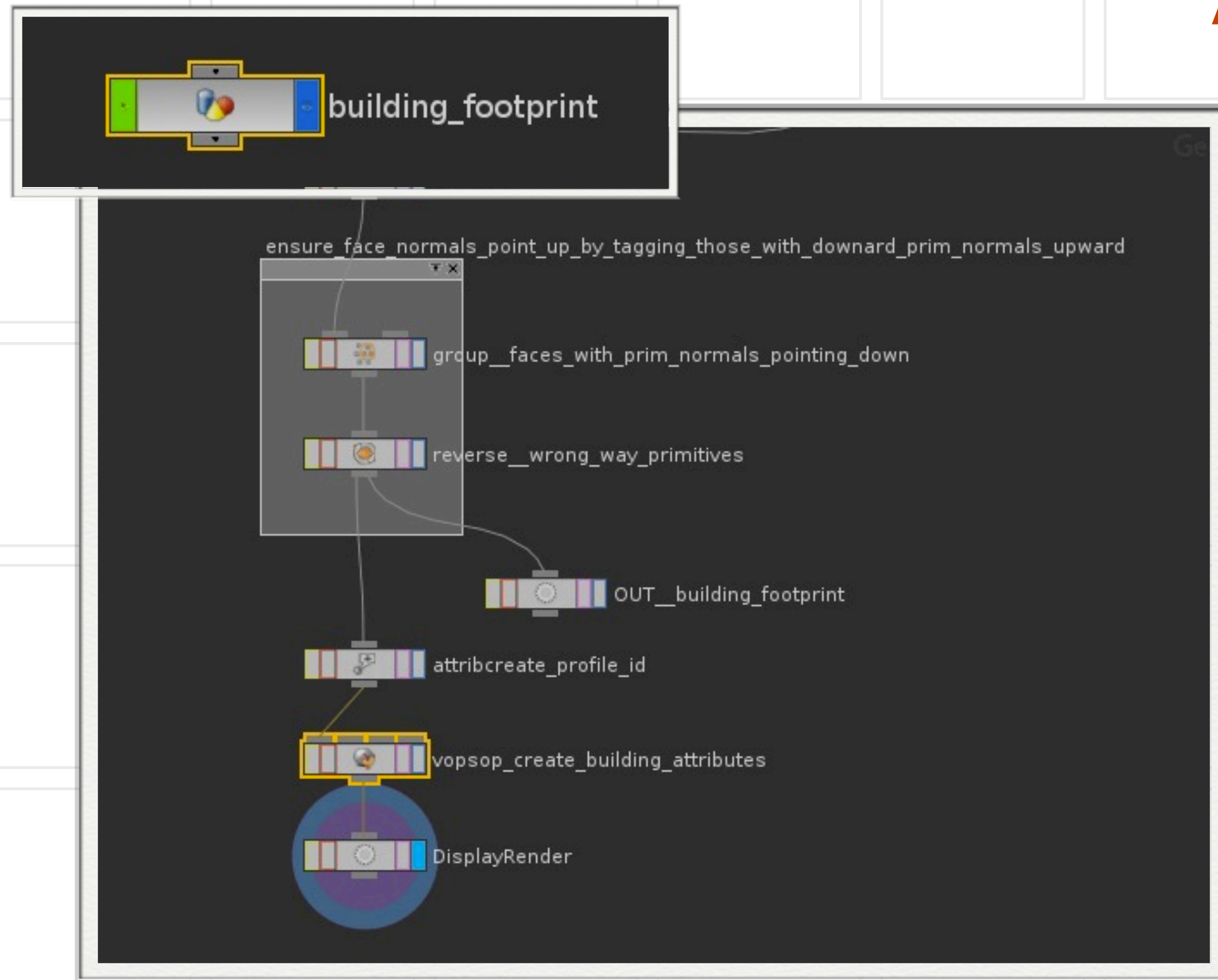
Create a Profile ID

- ▶ After the Reverse SOP append a Attribute Create
 - ▶ Name - profile_id
 - ▶ Class - Point
 - ▶ Size - 1
 - ▶ Value - \$PR

Every curve has a unique Prim ID so just transfer the ID to the points



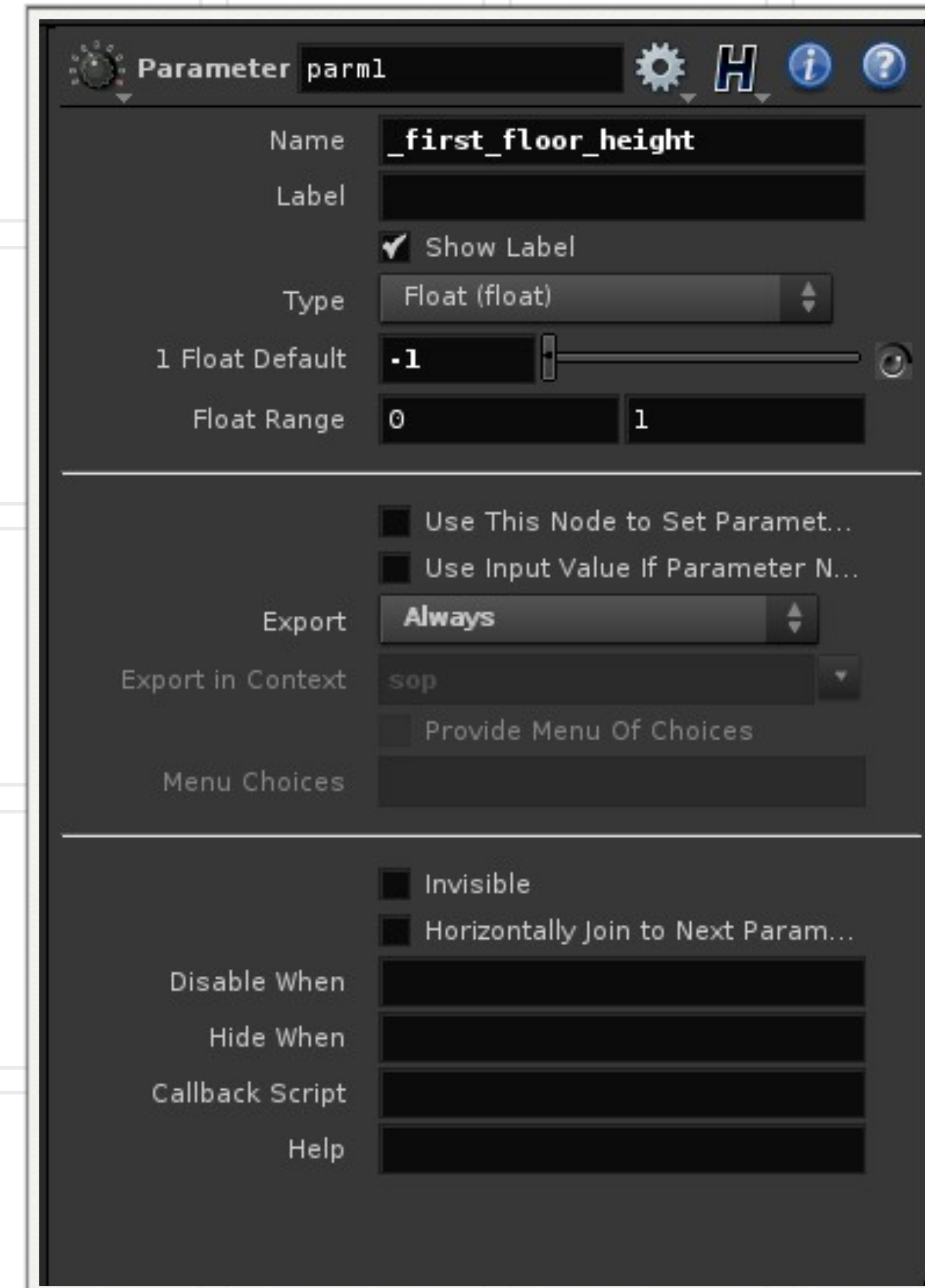
Adding a VOPSOP to the Network



- ▶ After the Attribute Create SOP
 - ▶ Append a VOPSOP
 - ▶ Name it - vopsop_create_building_attributes
 - ▶ Dive inside

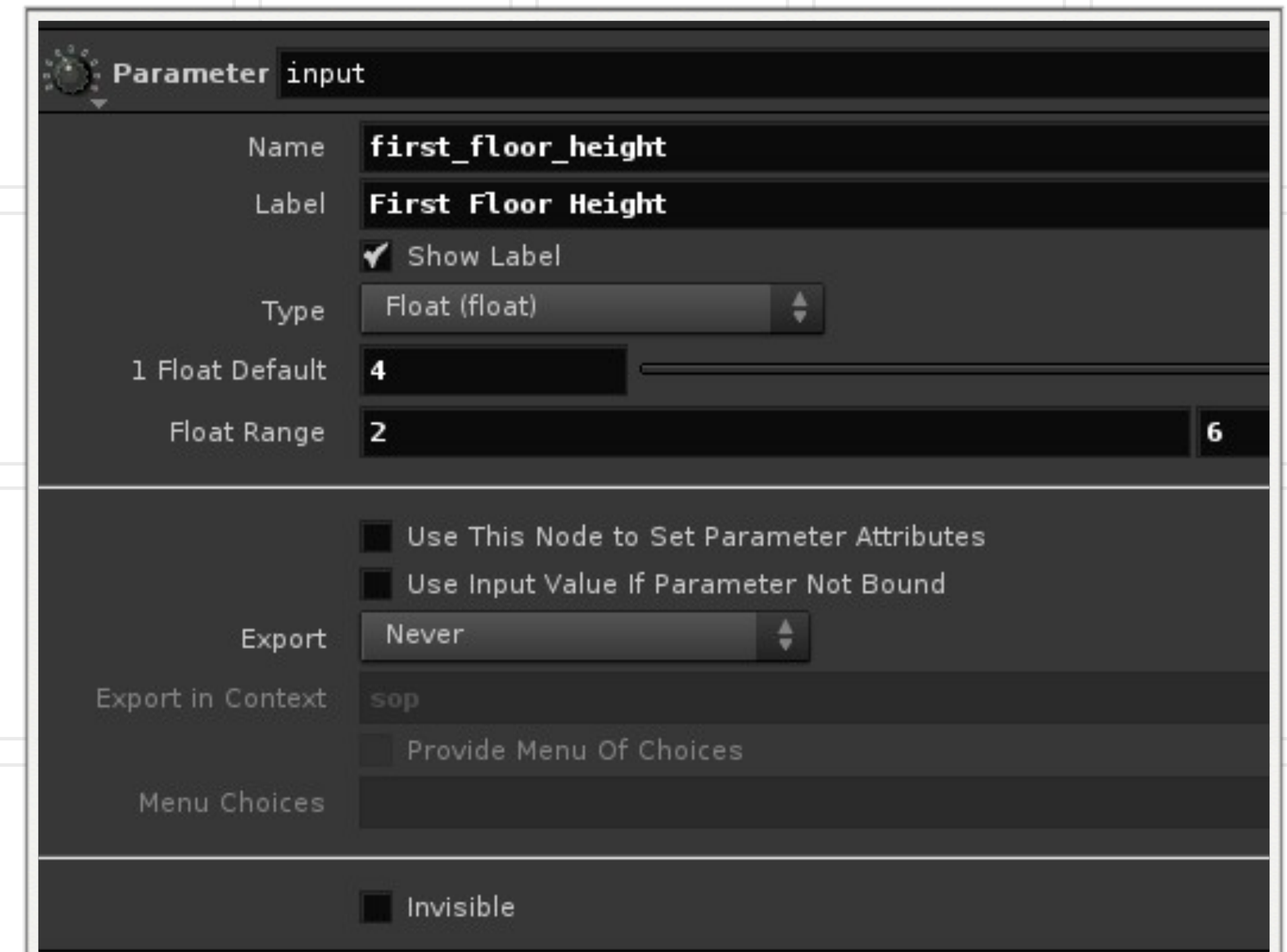
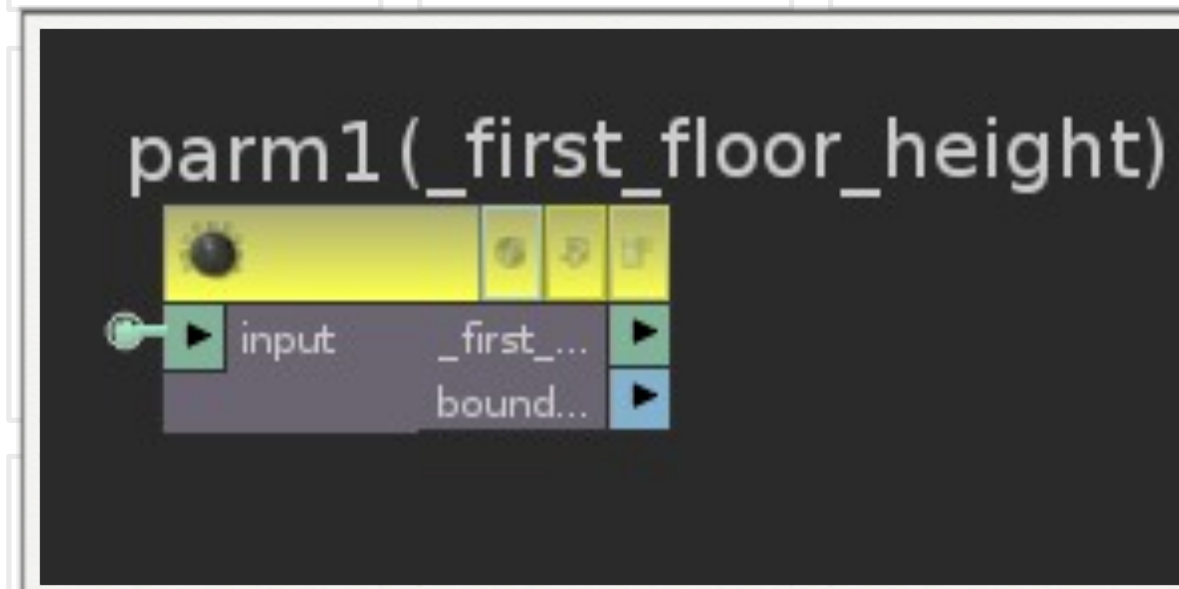
Now we Can Create Our Profile Attributes

- ▶ Add a Parameter - `_first_storey_height`
 - ▶ I put an underscore at the beginning of the name so I will not have name conflicts with the next step
- ▶ I set the default to “-1” which is a common naming convention to say the value has not been set yet
- ▶ Type - float
- ▶ Export - Always
- ▶ Invisible - Select



Now we Can Create Our Profile Attributes (cont.)

- ▶ Promote the Input Parameter and select the dongle
- ▶ This time I name it first_storey_height without the underscore and give it a label
- ▶ Give it a default value
- ▶ Give it a range



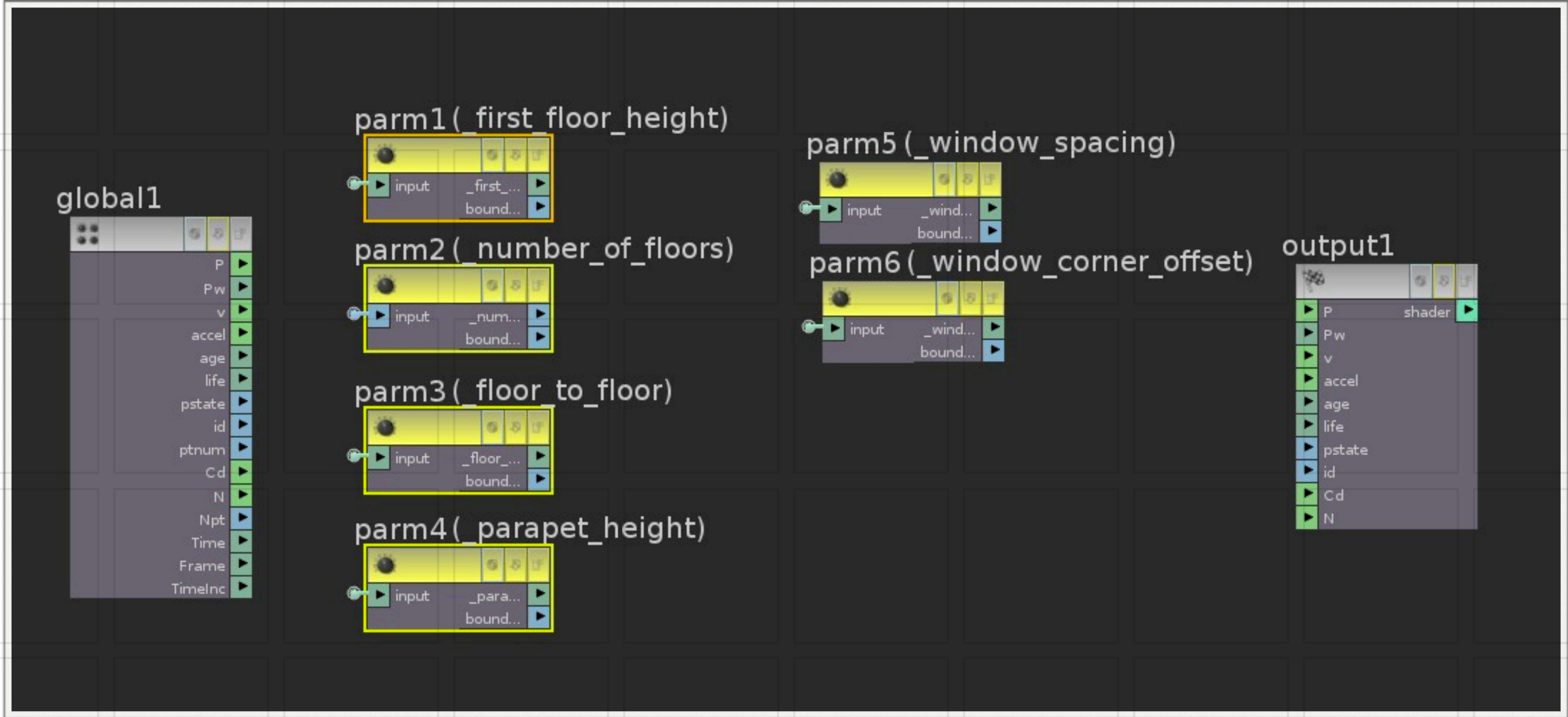
Now we Can Create Our Profile Attributes (cont.)

- ▶ Create Parameters following the same steps as we just did for number of stories

We already created `profile_id` at the SOP level

Name	Type	Class
number_of_floors	int	point
floor_to_floor	float	point
parapet_height	float	point
window_spacing	float	point
window_corner_offset	float	point
profile_id	int	point
first_floor_height	float	point

Parameters Defined



A Couple of Parameter Definitions

Export

Specifies whether the new parameter can be exported to other contexts (written to as well as read from). If set to **Always** or **When Input is Connected**, this operator gets an input. The value wired into this input is then assigned to the exported parameter. In a Surface network, exported parameters can be used to create deep rasters. In SOP and POP networks, the exported parameters create new geometry attributes.

Never

The parameter will not be exported.

Always

The parameter will be exported.

When Input is Connected

The parameter will only be exported if the node's input is ultimately connected to another VOP.

Invisible

Do not show the parameter in the shader's parameter interface. The parameter still exists and will be used to generate shader strings.

Creating Prim Attributes

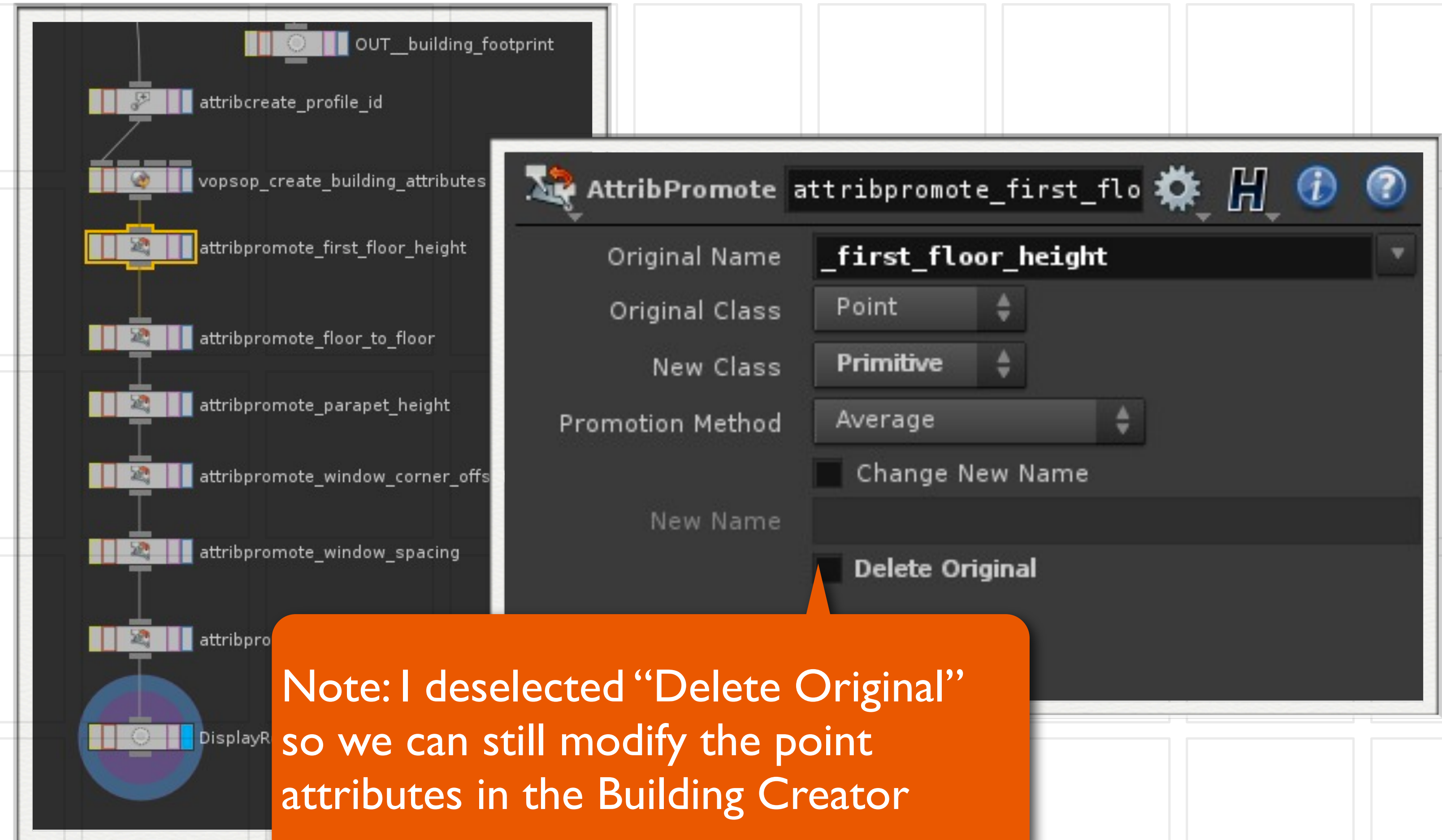
So far we have a bunch of **Point Attributes**. But what we really need are **Prim Attributes** with the same name

We need to promote all the **Point Attributes** We just created to **Prim Attributes**

▶ continued on next slide...

Now we Can Create Our Profile Attributes (Cont.)

- Need to promote all the point attributes to prim attributes



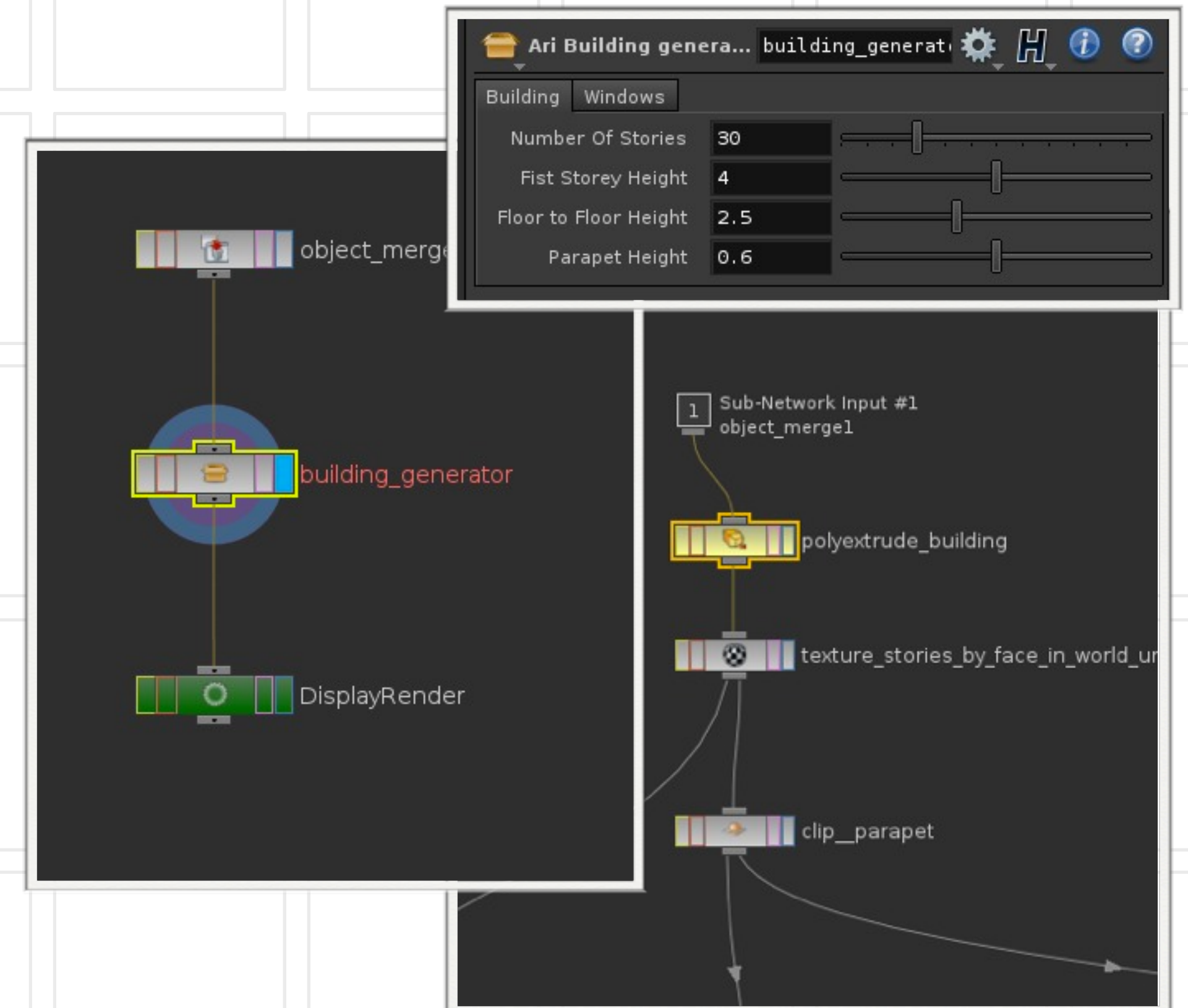


Testing Out the Building Profile Attributes

**SIDE EFFECTS
SOFTWARE**

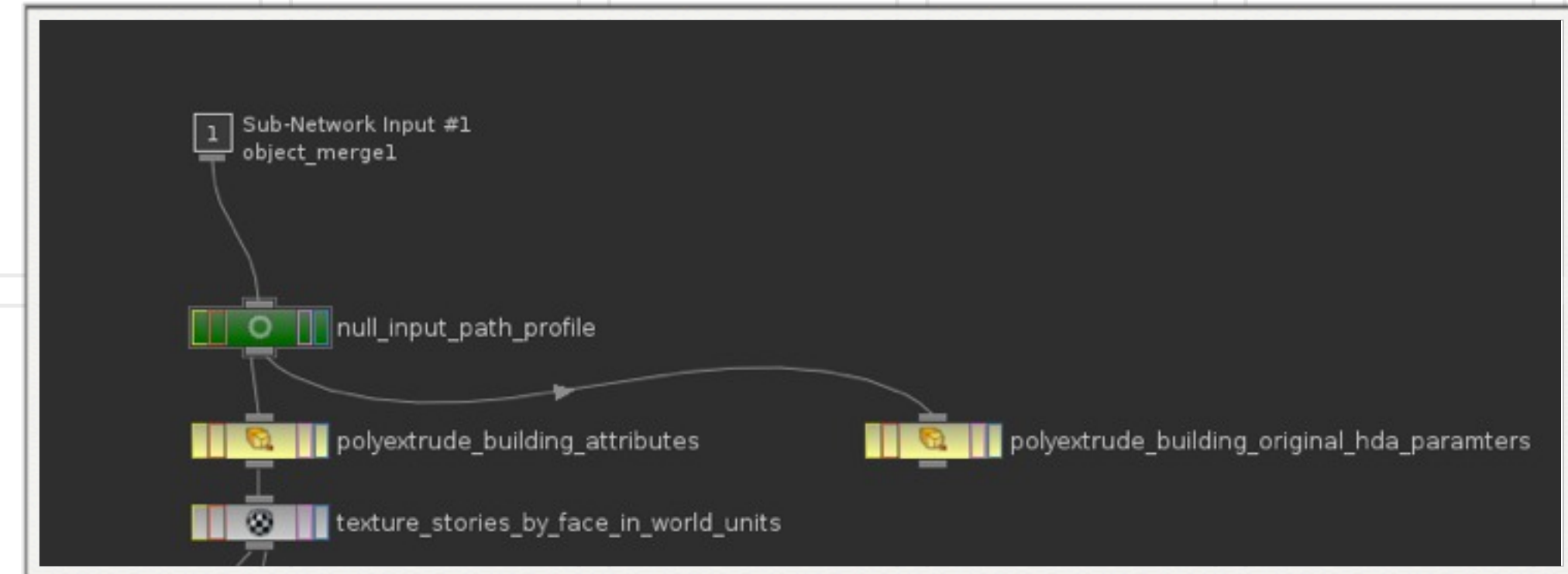
Where Do We Use the Attributes

- ▶ Up till now the artist has tweaked the parameters on the Building Generator Asset to Create the Building
- ▶ In turn the Building Generator Asset Parameters drive the PolyExtrude parameters inside the Asset
- ▶ Temporarily we need to create another PolyExtrude that is driven by the the Attributes attached to the Building Footprint curves.



Where Do We Use the Attributes (cont.)

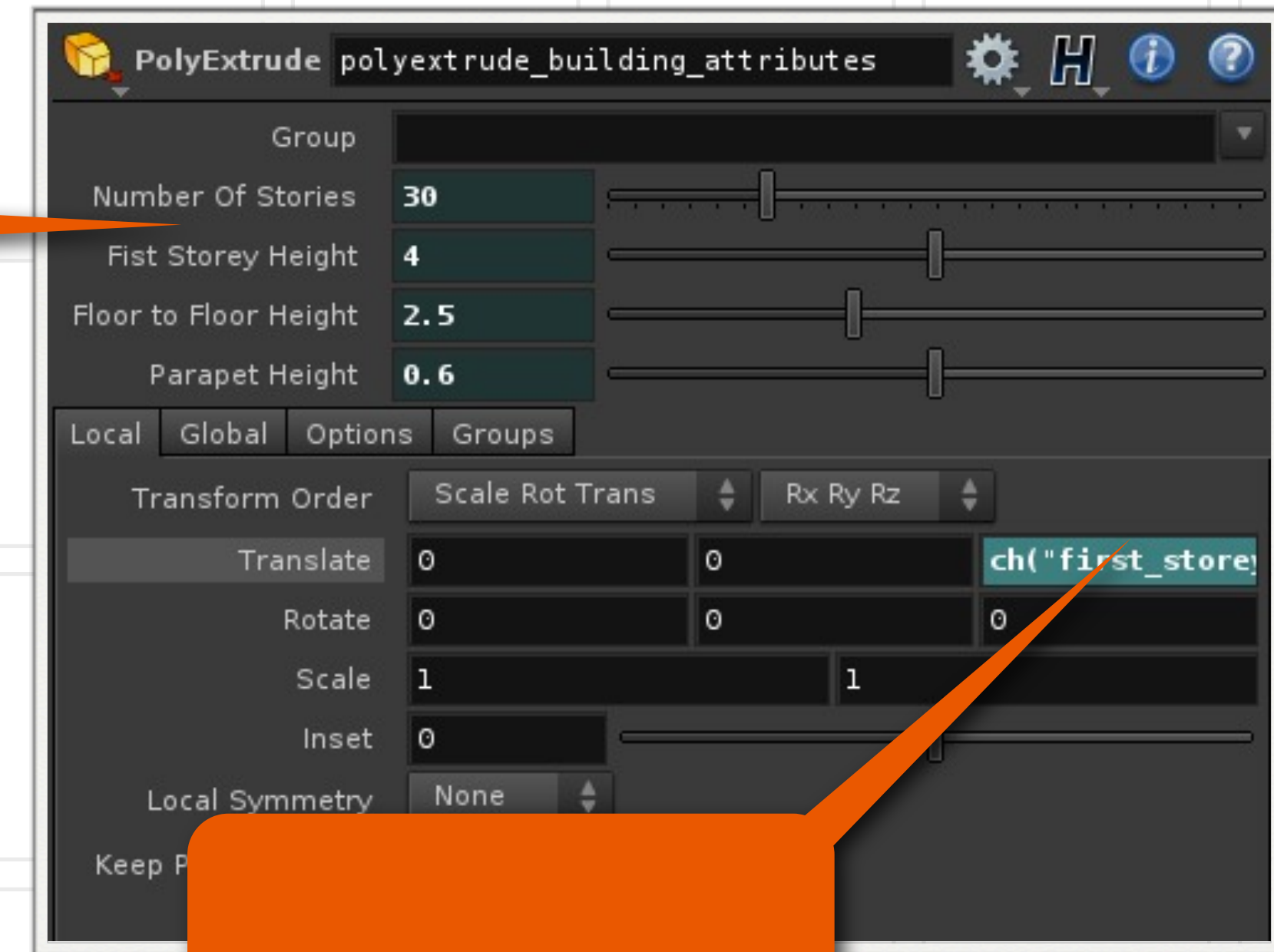
- ▶ Before the Polyextrude Node Create a NULL
 - ▶ Name it - null_input_path_profile
- ▶ Duplicate the PolyExtrude Node
 - ▶ Name the Original Node - polyextrude_building_attributes
 - ▶ We are doing this because if you remember all the other nodes in this network reference this node for values
 - ▶ Name the Duplicate Node - polyextrude_building_hda_parameters
 - ▶ This is the original node values - We need this to re-hook the original values after our experiment



Where Do We Use the Attributes (cont.)

Delete these
channels

- ▶ Delete the top channels channels in the polyextrude_building_attributes



Not this channel

We will use the Prim Expression to get at our attributes

- ▶ float prim (string surface_node, float prim_num, string attrib_name, float attrib_index)

- ▶ Returns the value of a primitive attribute.

- ▶ When given the "P" or "Pw" attribute, returns the centroid of the

- ▶ primitive.

- ▶ EXAMPLES

- ▶ `prim("/obj/geo1/facet1", 3, "P", 0)`

- ▶ Evaluates the X component of the centroid of primitive 3 in the

- ▶ specified surface node.

- ▶ `prim("/obj/geo1/facet1", 3, "Cd", 1)`

- ▶ Evaluate the green color of the "Cd" attribute of primitive 3.

PRIM EXPRESSIONS

This Node



polyextrude_building_attributes * Take List Parameter Spread

obj > building2 > building_generator

PolyExtrude polyextrude_building_attributes

Group	
Number Of Stories	<code>prim("../null_input_path_profile", 0, "_number_of_stories", 0)</code>
Fist Storey Height	<code>prim("../null_input_path_profile", 0, "_first_floor_height", 0)</code>
Floor to Floor Height	<code>prim("../null_input_path_profile", 0, "_floor_to_floor", 0)</code>
Parapet Height	<code>prim("../null_input_path_profile", 0, "_parapet_height", 0)</code>

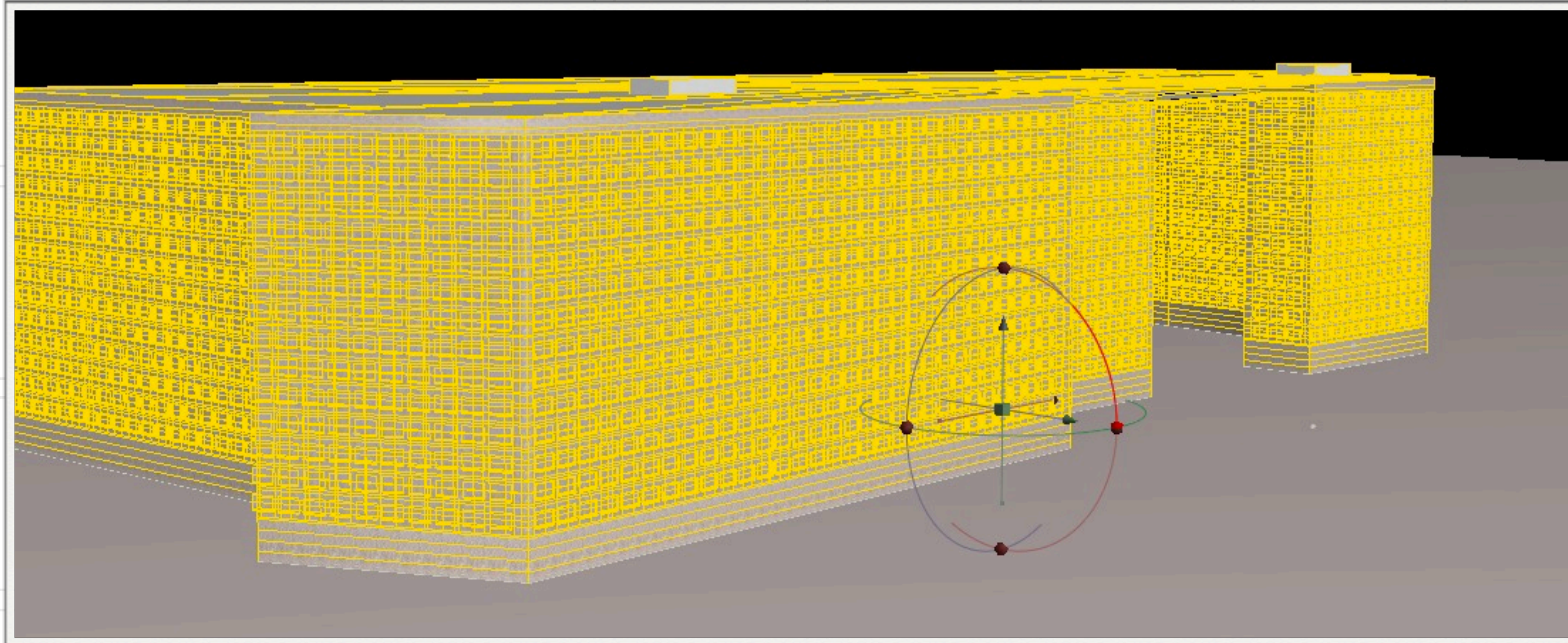
Local Global Options Groups

Transform Order Scale Rot Trans Rx Ry Rz

CORRECTION:
Should be
number_of_floors

**SIDE EFFECTS
SOFTWARE**

Test Your Network So Far...



Looking Good!

**SIDE EFFECTS
SOFTWARE**

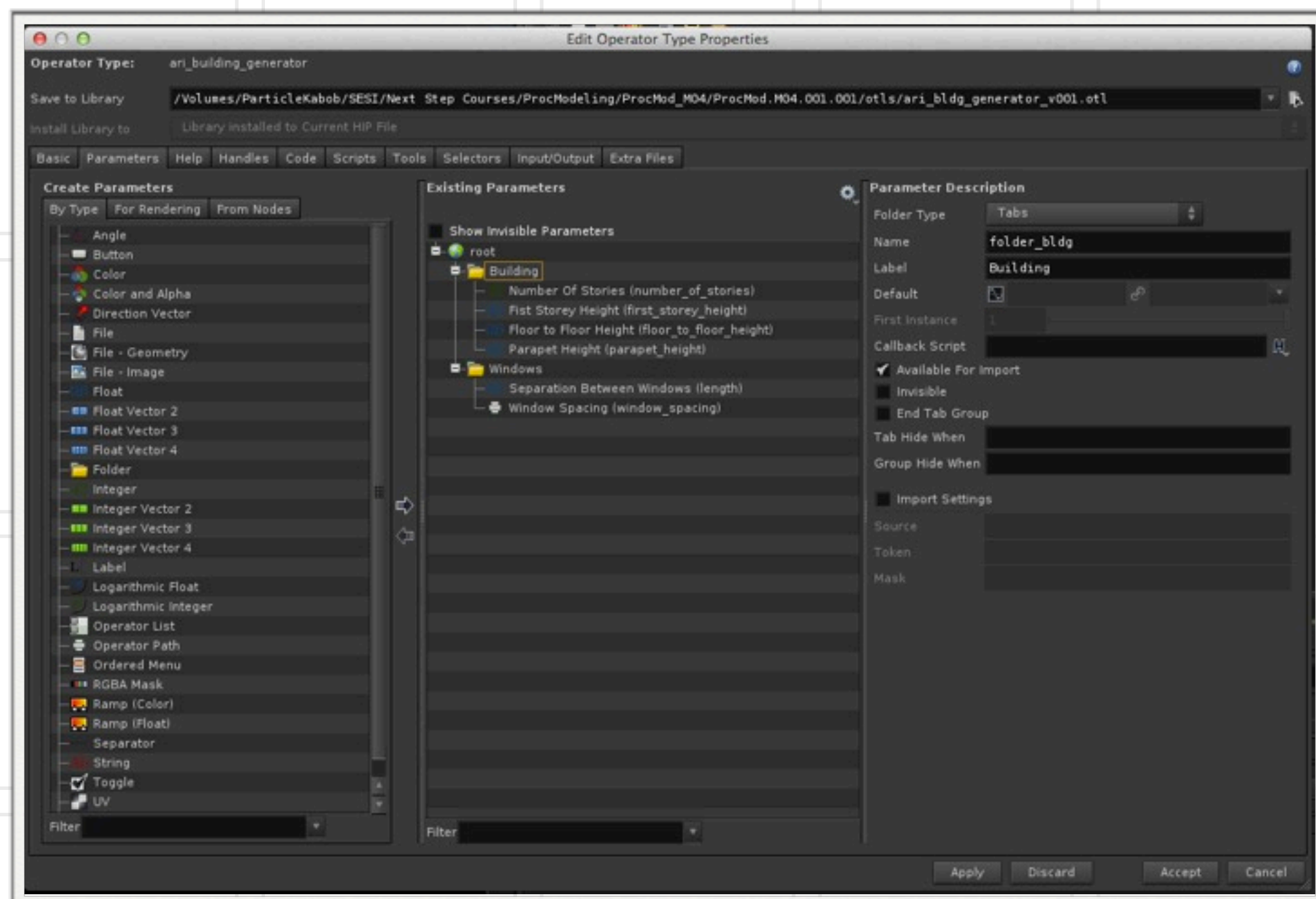


Allowing the Artist to Choose

Attributes in the Profile Curve or Parameters in the BuildingGenerator Asset

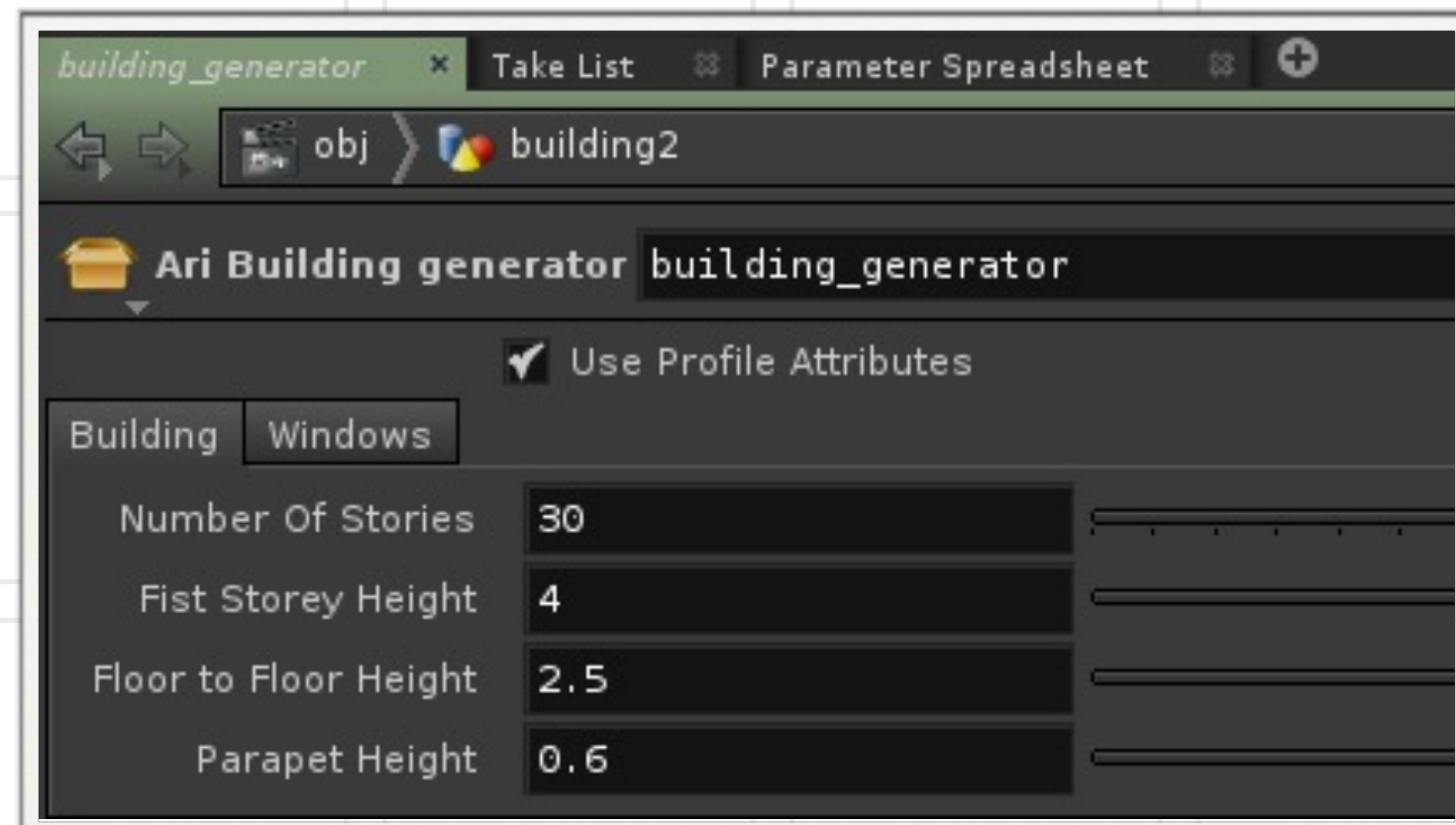
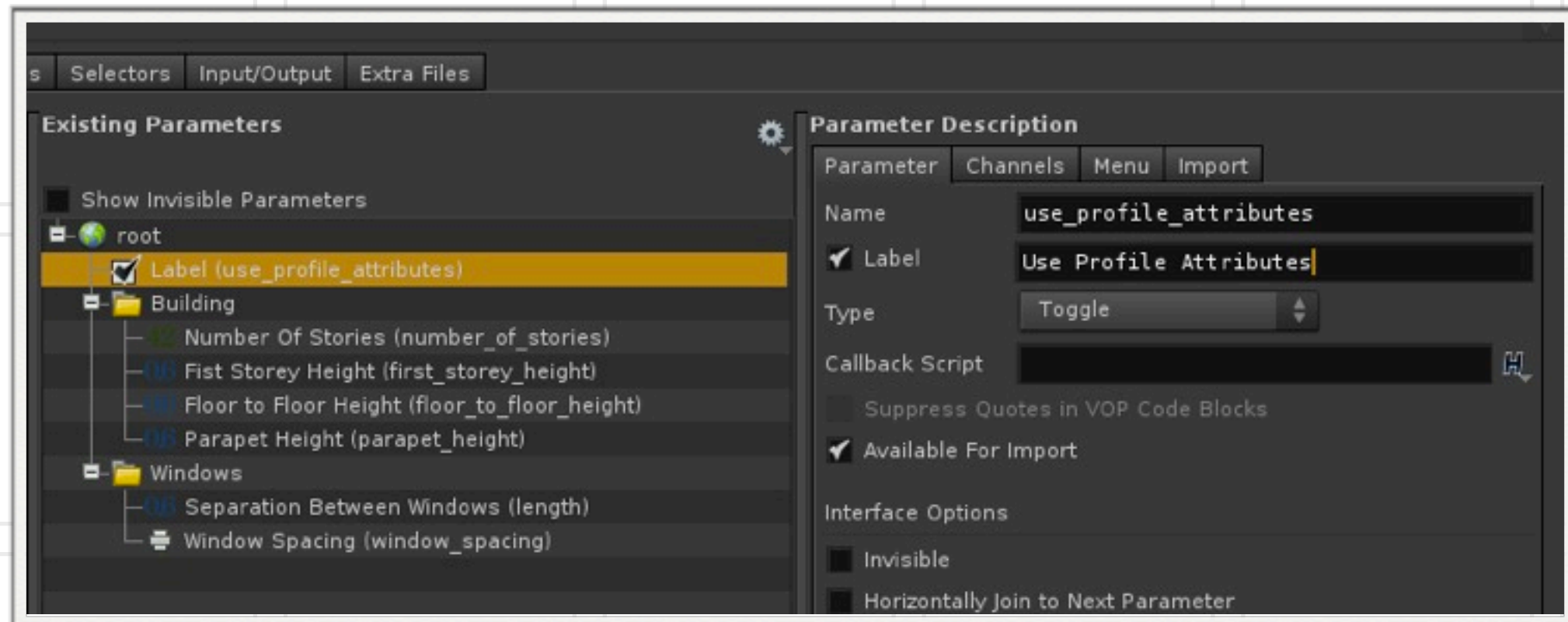
**SIDE EFFECTS
SOFTWARE**

Modifying the User Interface



- ▶ We are going to modify the User Interface of the Building Generator Asset
 - ▶ We are going to include a toggle to allow the user to choose to use either the built-in attributes of the profile curve or the parameters in the Digital Asset
- ▶ Right Click on the Building Generator and Select “Type Properties”
 - ▶ Remember if your Building Generator is Locked (Blue Text) right click the node and select “Allow Editing of Contents” first

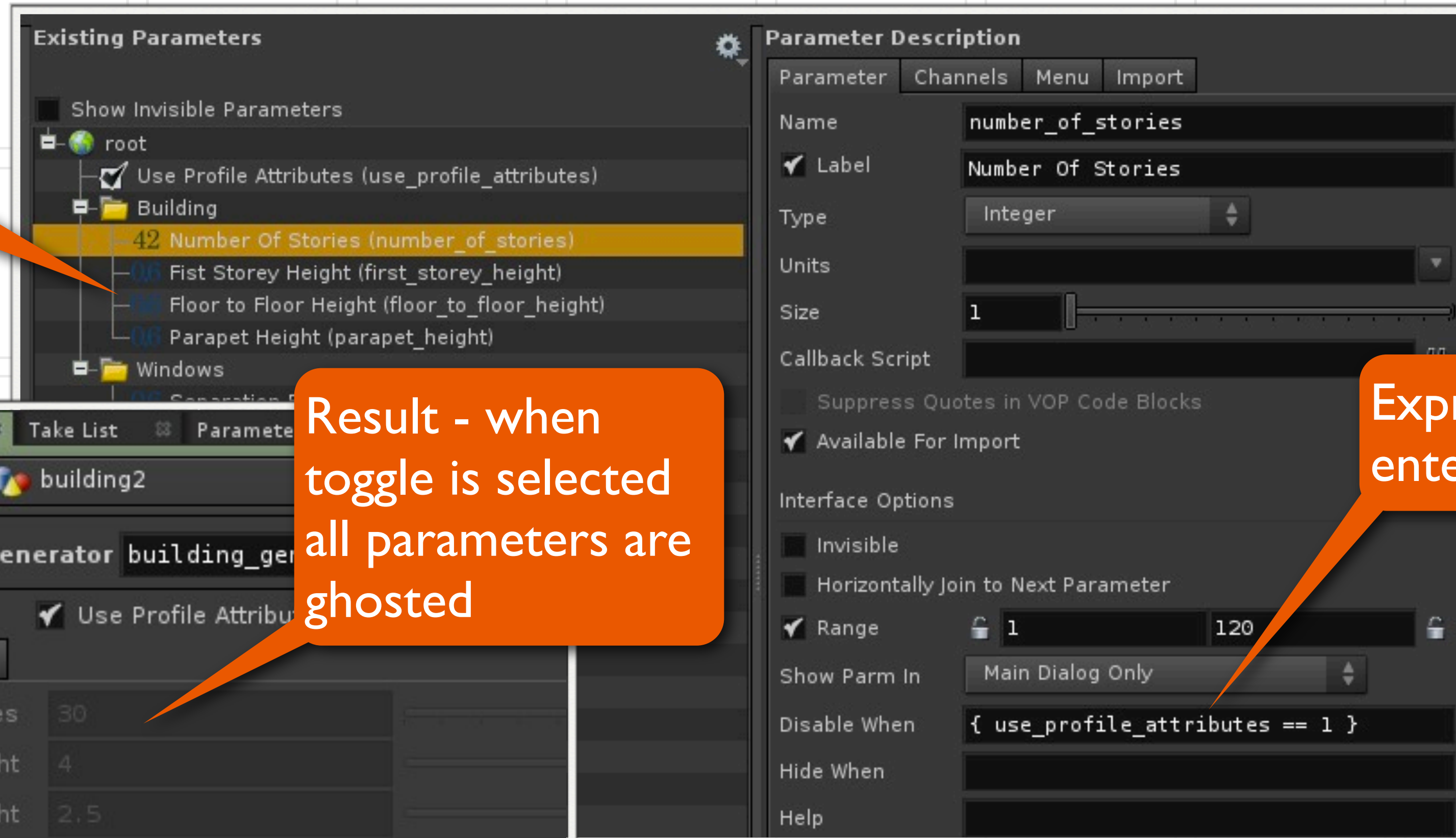
Adding the Toggle



- ▶ Add a Toggle to list of parameters and move it to the top
 - ▶ Name it - use_profile_attributes
 - ▶ Label - Use Profile Attributes
- ▶ Now we want to ghost the other parameters if “Use Profile Attributes” is selected
 - ▶ use the expression - { use_profile_attributes == 1 } on all the other parameters

Adding the Toggle (cont.)

Enter Expression
in all parameters
except toggle



Result - when
toggle is selected
all parameters are
ghosted

Expression
entered here

Modify Expression to PolyExtrude

Now we need to modify the expression on the polyextrude node to determine if the toggle is selected or not. If the toggle is selected the polyextrude should use the values from the profile curves if not it should use the values from the parameter on the Building Generator Asset

We will use the if statement - from exhelp in the textport

▶ continued on next slide...

Modify Expression to PolyExtrude (cont.)

`float if (float expression, float true_value, float false_value)`

Returns the value of the second or third argument depending on the truth of the first argument.

This is a function, which means all parameters to the function are evaluated.

So something like `if($F > 1, system('echo 1'), system('echo2'))`

will result in both system calls being run regardless of the result of the expression.

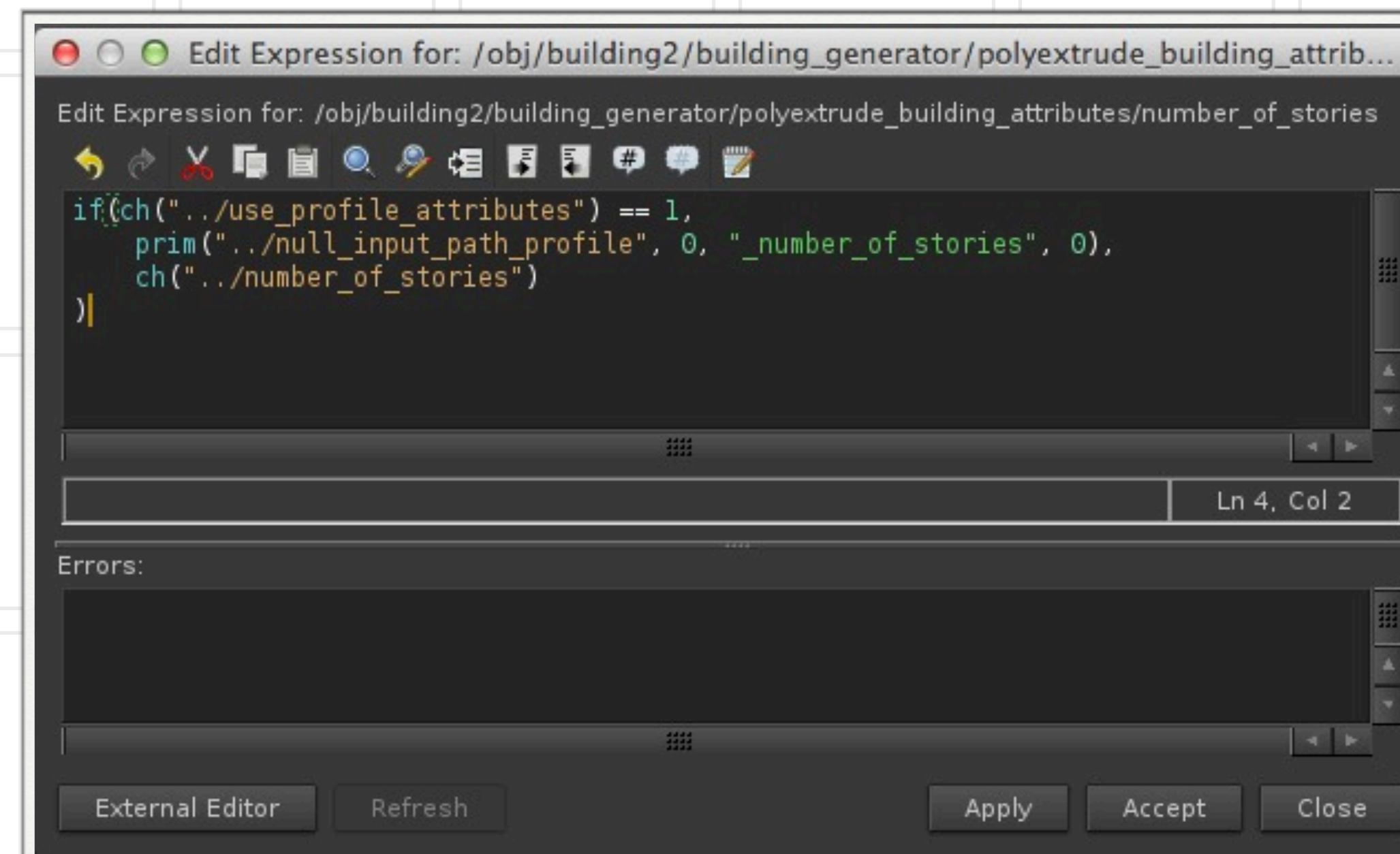
EXAMPLES

`if ($F<12, $F, 75)`

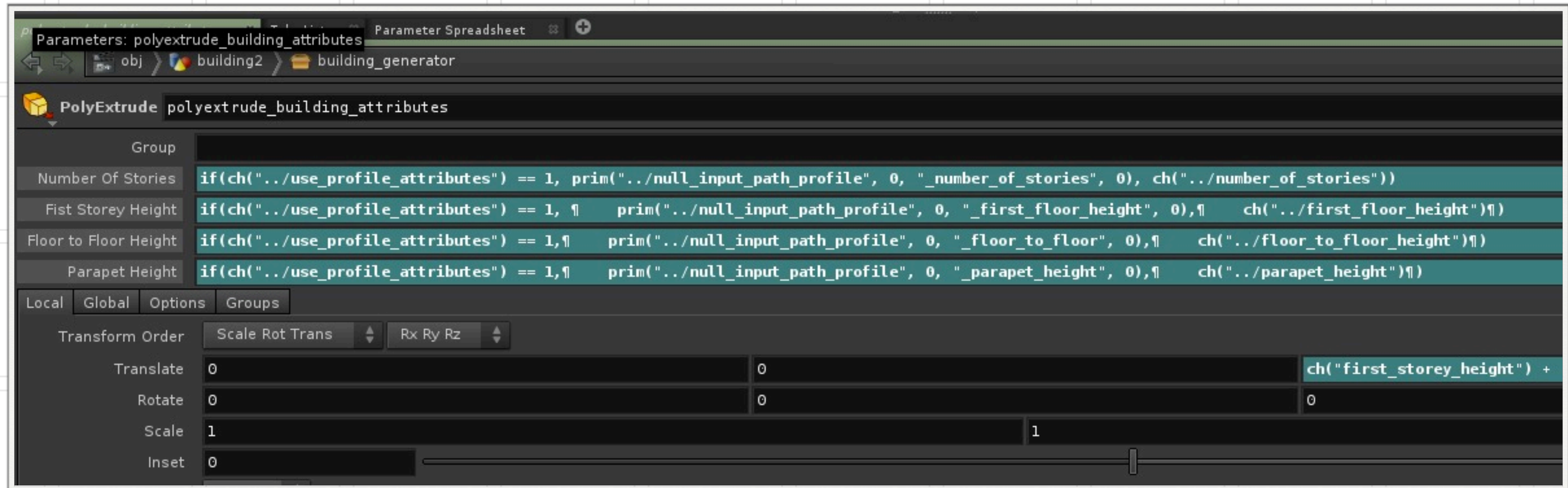
When the current frame number (\$F) is less than 12, returns the current frame number, otherwise returns 75.

Example Expression

- ▶ So the expression will look like the following
 - ▶ `if(ch("../use_profile_attributes") == 1, prim("../null_input_path_profile", 0, "_number_of_floors", 0), ch("../number_of_floors"))`
- ▶ Using the Expression Editor we can break apart the line so it is more readable



Add the Expression for Each Parameter



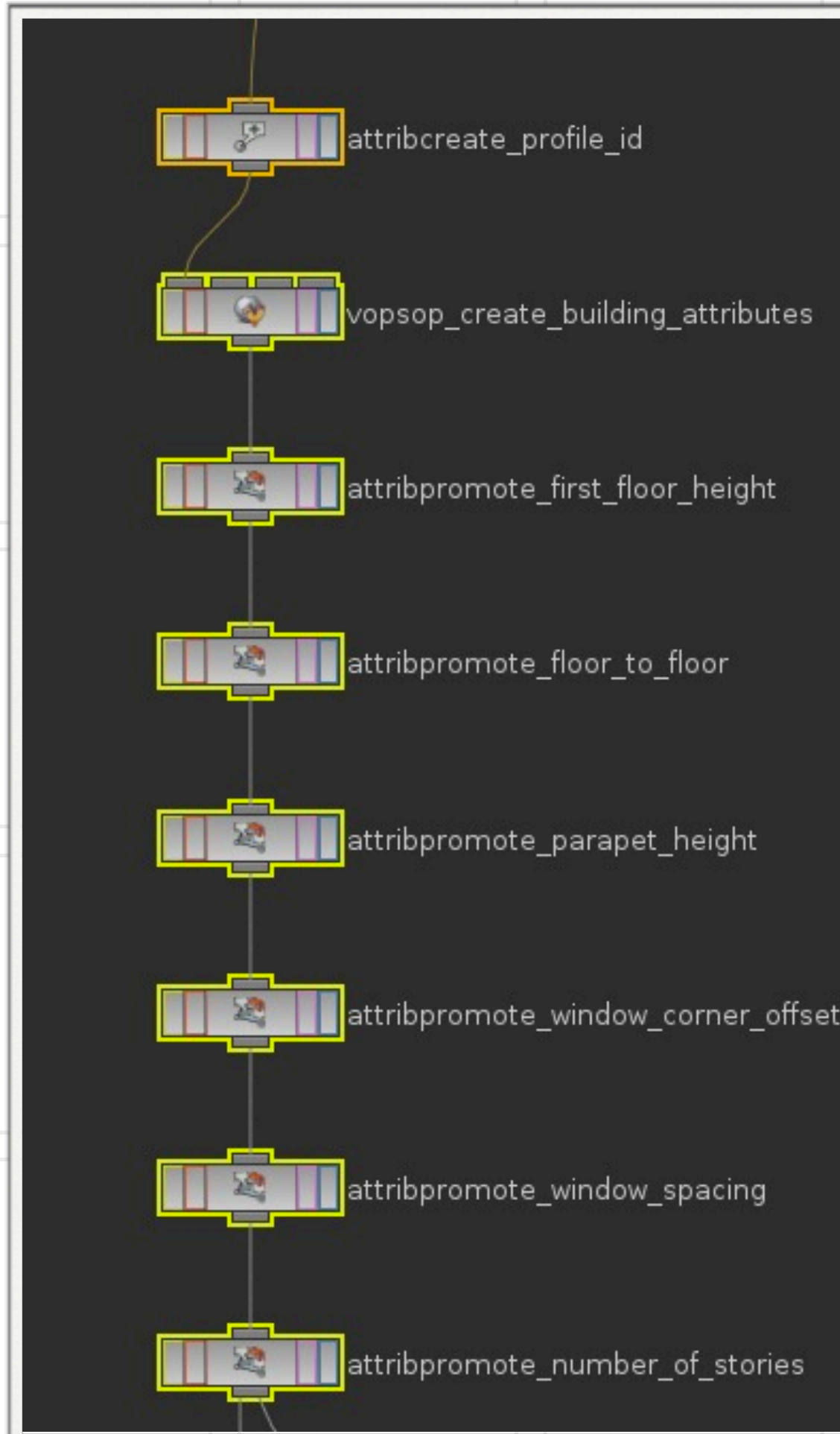


Making the Footprint Profile Digital Asset

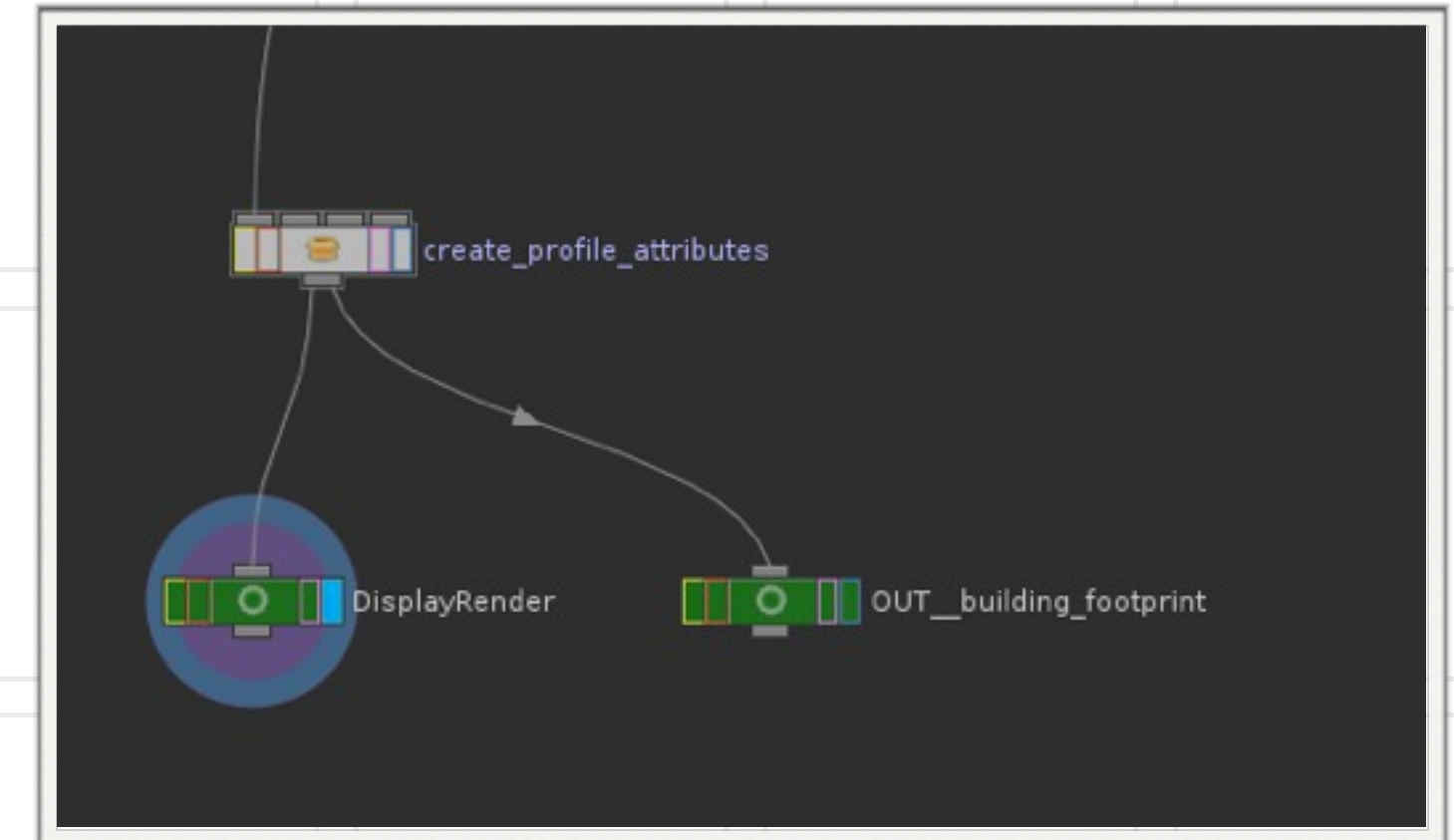
Time for More Cleanup

**SIDE EFFECTS
SOFTWARE**

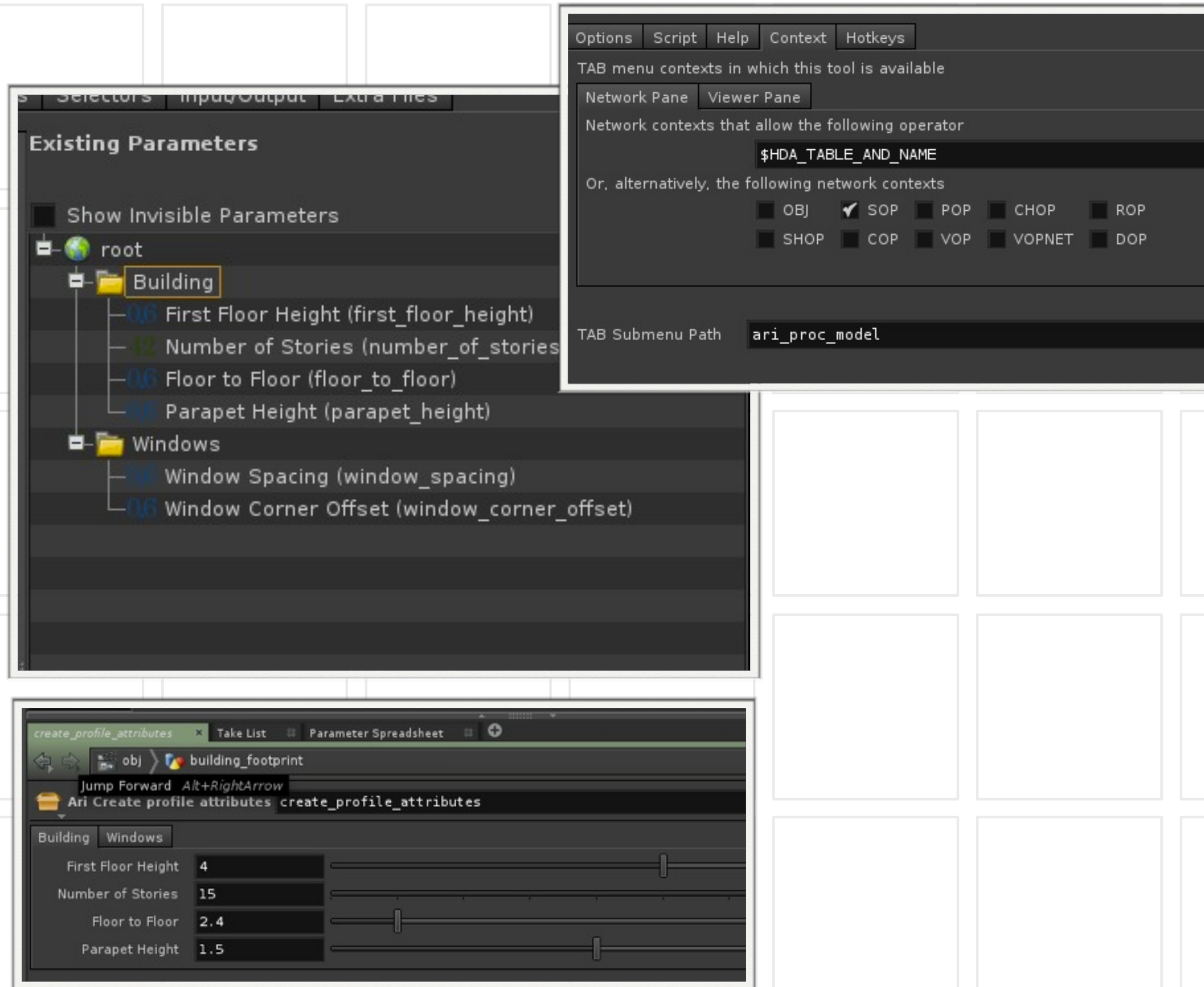
Convert the VOPSOP and Attribute Promotes to an Asset



- ▶ Go back up to the Obj level
- ▶ Dive into the building_footprint object
- ▶ Select everything from the `attrib_create_profile_id` to the VOPSOP and all the promote attributes
- ▶ Make them into a subnetwork and name it “`create_profile_attributes`”
- ▶ Convert the Subnet into a Digital Asset



Convert the VOPSOP and Attribute Promotes to an Asset (cont.)



- ▶ In the Type Properties Panel
 - ▶ Move the Subnets parameters into the type properties
 - ▶ Add two folders - One for Buildings and one for Windows
 - ▶ Move the appropriate parameters into the folders
- ▶ Select the Tools tab, Context Sub Tab
 - ▶ Network Context - SOP
 - ▶ Tab Sub Menu - Proc Modeling
- ▶ Click Accept



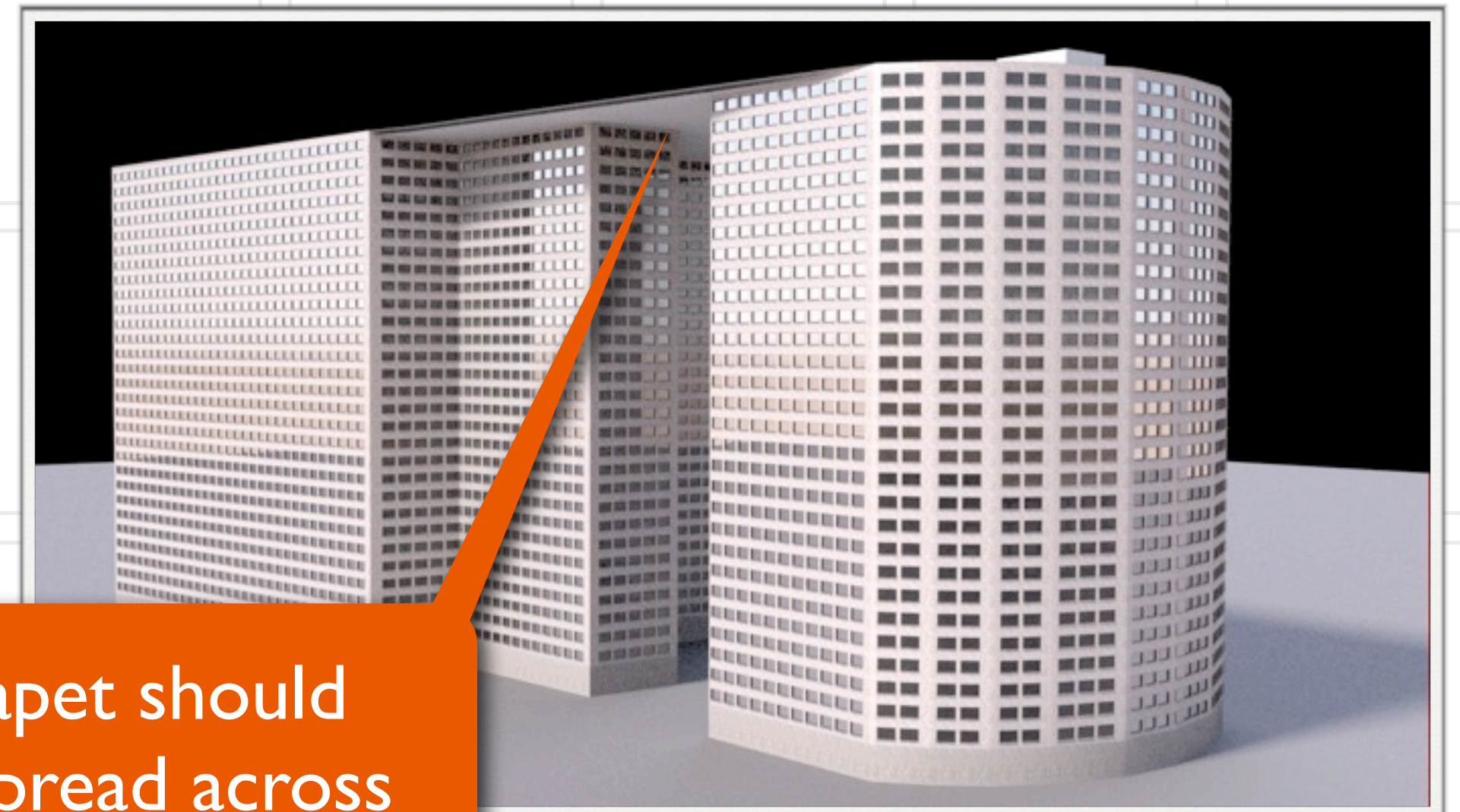
Taking Care of the Parapet Problem

**SIDE EFFECTS
SOFTWARE**

Parapet Problem

- ▶ If you remember from the last Module we had a problem with the Parapet
- ▶ All the building profiles were treated as one building and therefore the parapet was “smeared” across all the buildings
- ▶ Let us fix this now
 - ▶ We will do this by isolating each profile

The parapet should not be spread across all buildings



**SIDE EFFECTS
SOFTWARE**

Adding the Profile ID to the Attribute Profile Asset



Prim ID

- ▶ If you look at the Prim IDs of the curve you can see each one has a unique number. We can use that to our benefit

Creating Unique Profiles

Allow Editing of “create_profile_attributes”

Right Click the node and select “Type Properties”

- ▶ In the Parameter Editor add a String Field
- ▶ Name the string field - group_id
- ▶ Label - Group ID
- ▶ Click Accept

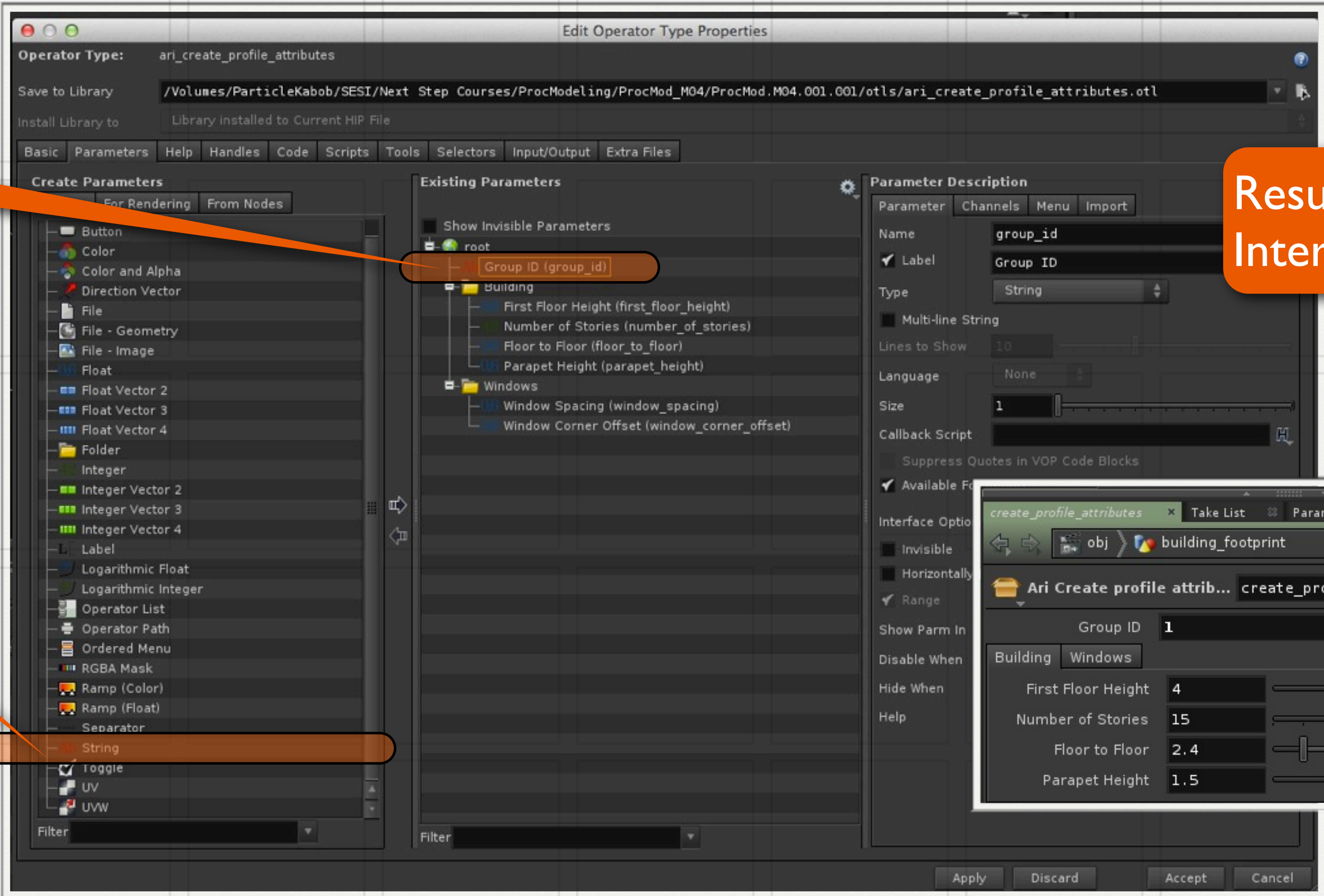
See photo on next slide...

Creating Unique Profiles (cont.)

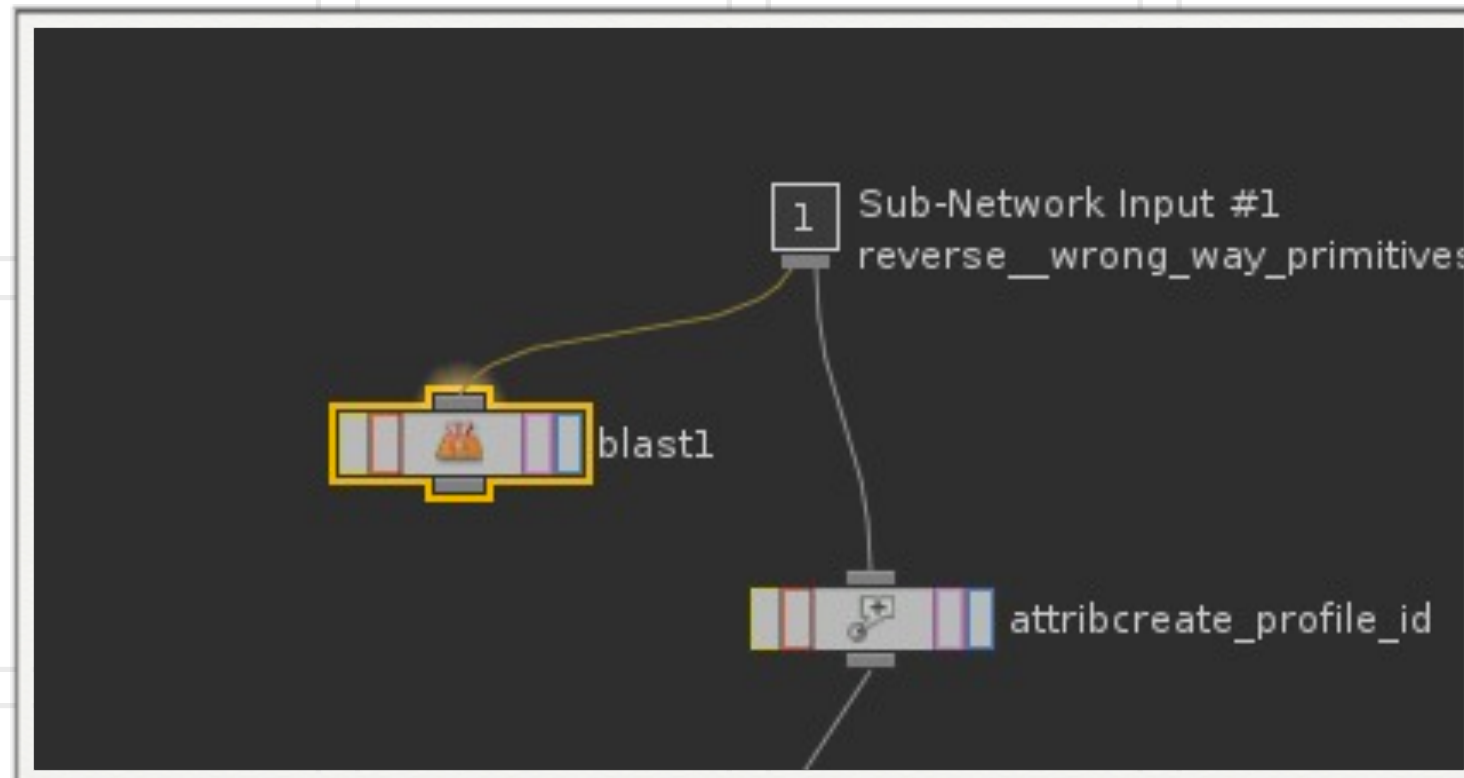
Place at top of user interface

String data type

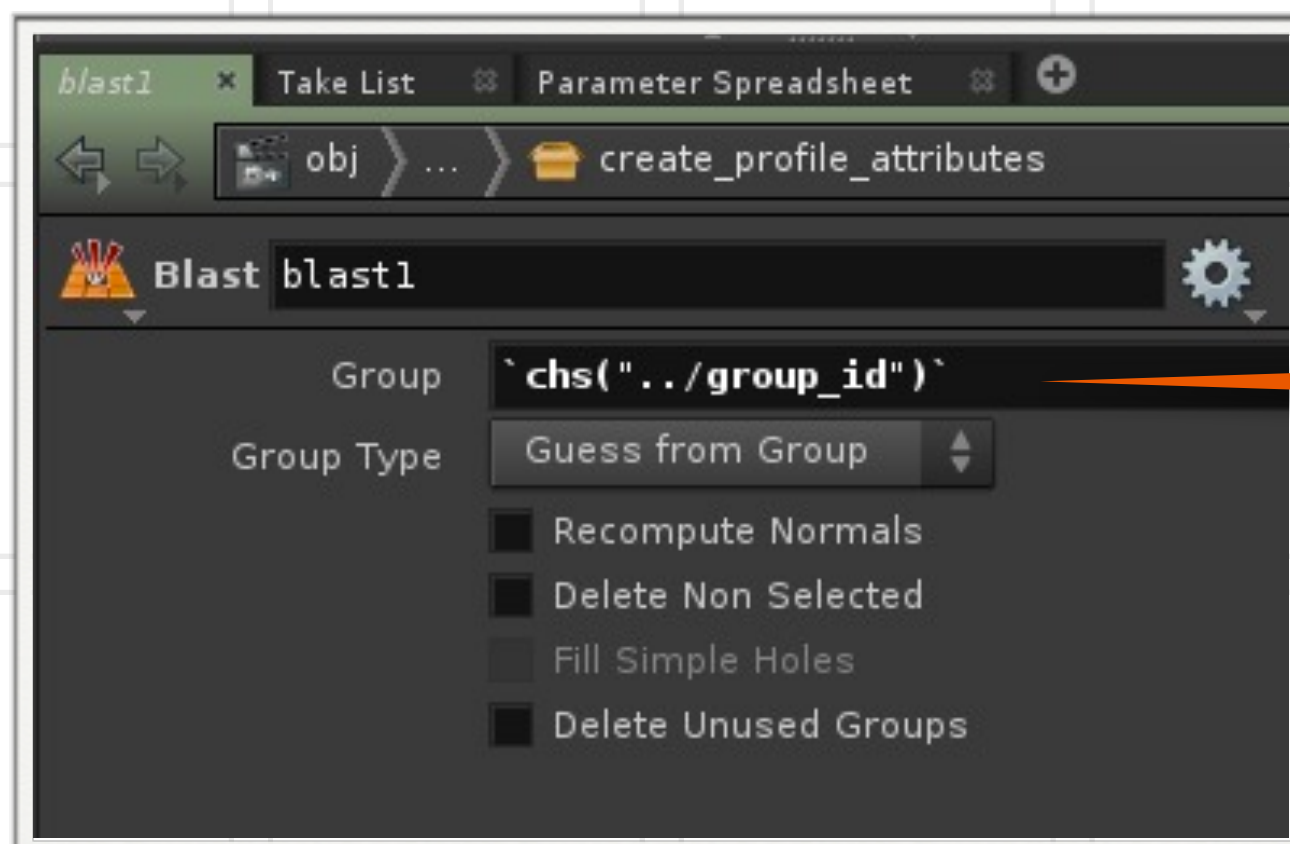
Resulting User Interface



Creating Unique Profiles (cont.)



- ▶ Dive into the “create_attribute_profile” asset
- ▶ Drop down a Blast SOP
 - ▶ Group = ``chs("../group_id")``

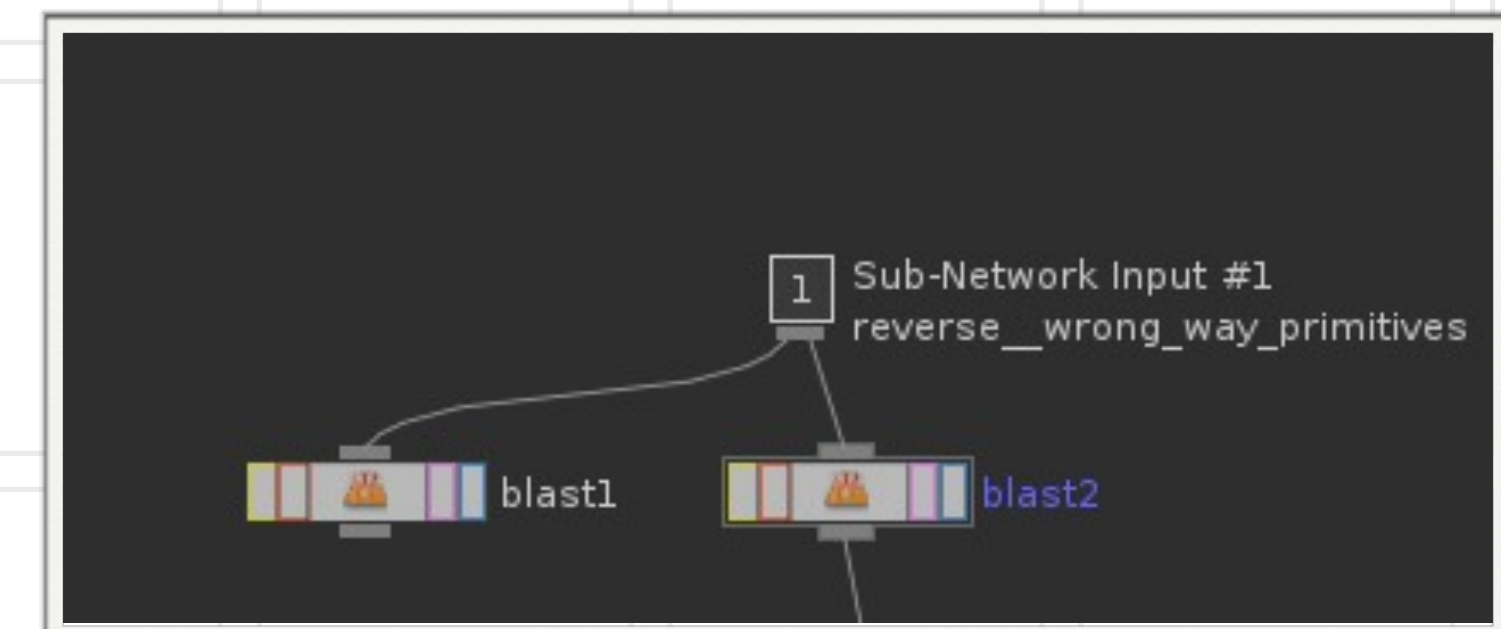
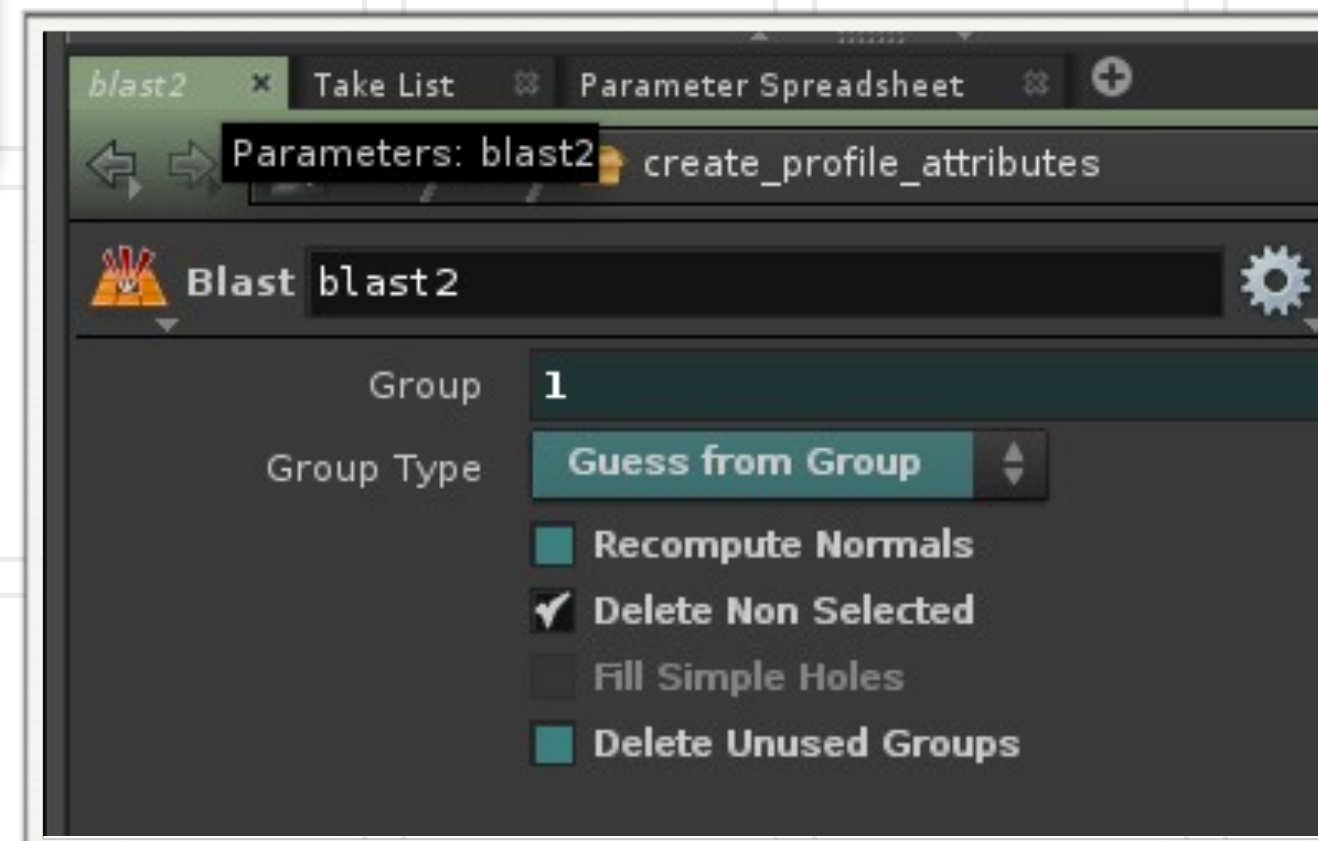


Notice the back ticks surrounding the expression. To evaluate an expression inside a Group parameter the back ticks are needed

Creating Unique Profiles (cont.)

- ▶ Create a reference copy of the Blast SOP by right clicking on the node and selecting “reference copy”
- ▶ Delete the channel “Delete Non Selected” from the reference copy and select it.
- ▶ Prepend the reference copy to the “attribute_create_profile_id”

Reference Copy



SIDE EFFECTS
SOFTWARE

What Are We Doing?

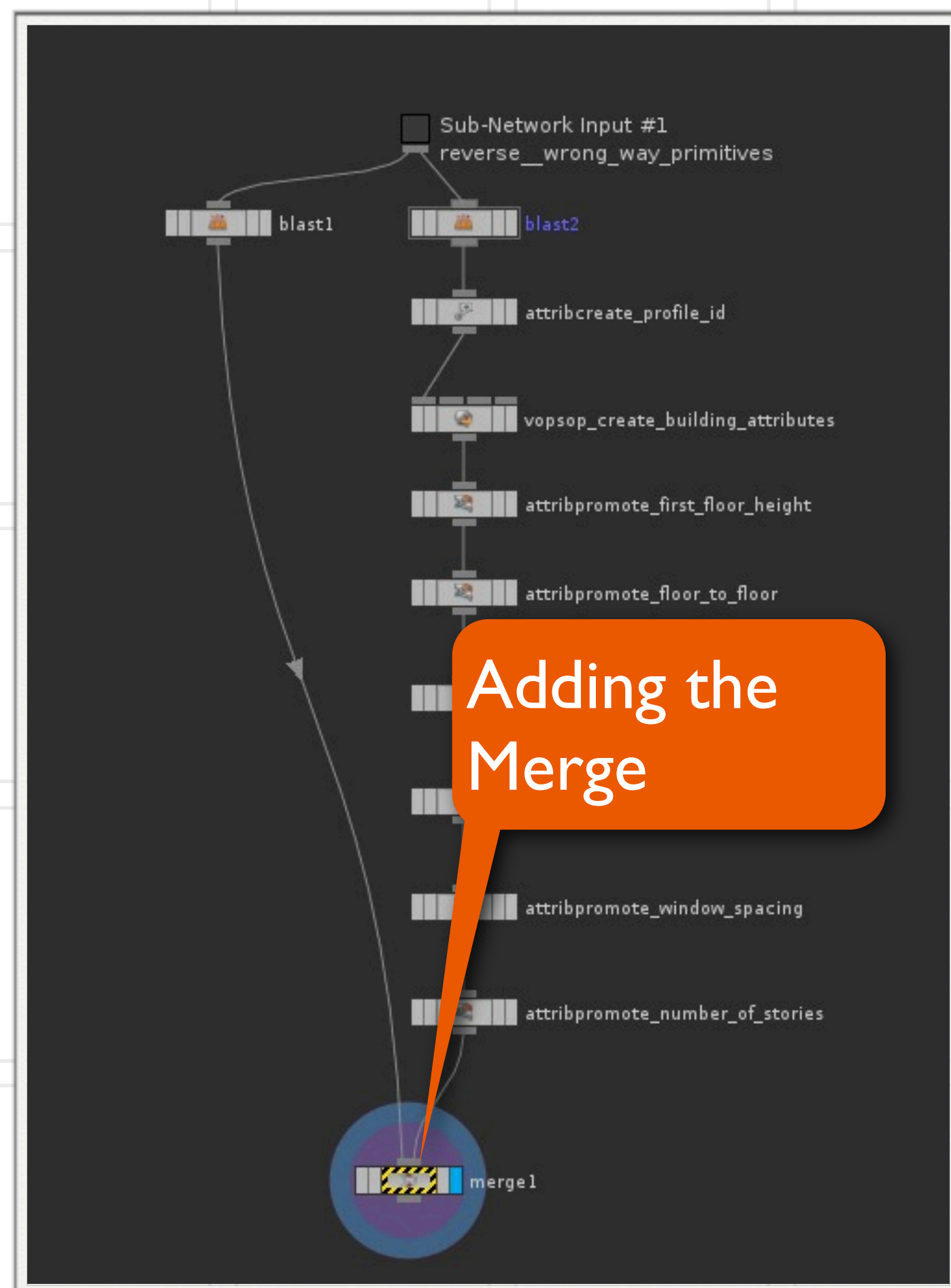
By creating two chains of Blast Nodes one deleting the selected primitive and one deleting the others we are making sure that we keep all the profiles for further use down the road

The Blast node deleting the other primitives will process and add the attributes specific to the profile (e.g., floor_to_floor, parapet_height)

The Blast node that contains all the the other profiles will have no processing done to it but will be passed on with a Merge SOP.

This way we can have a string of “create_profile_attributes” assets adding the attributes for each profile

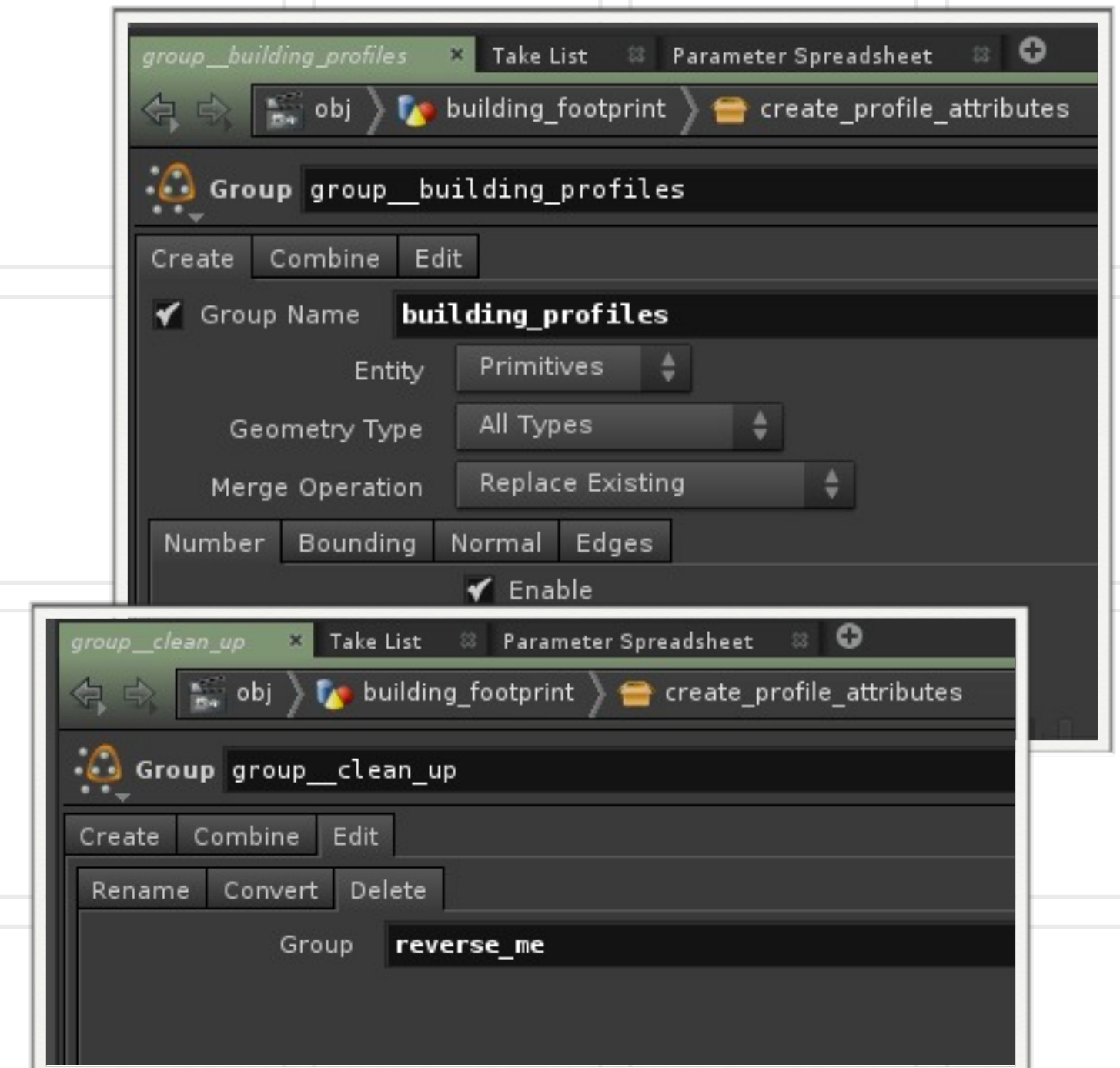
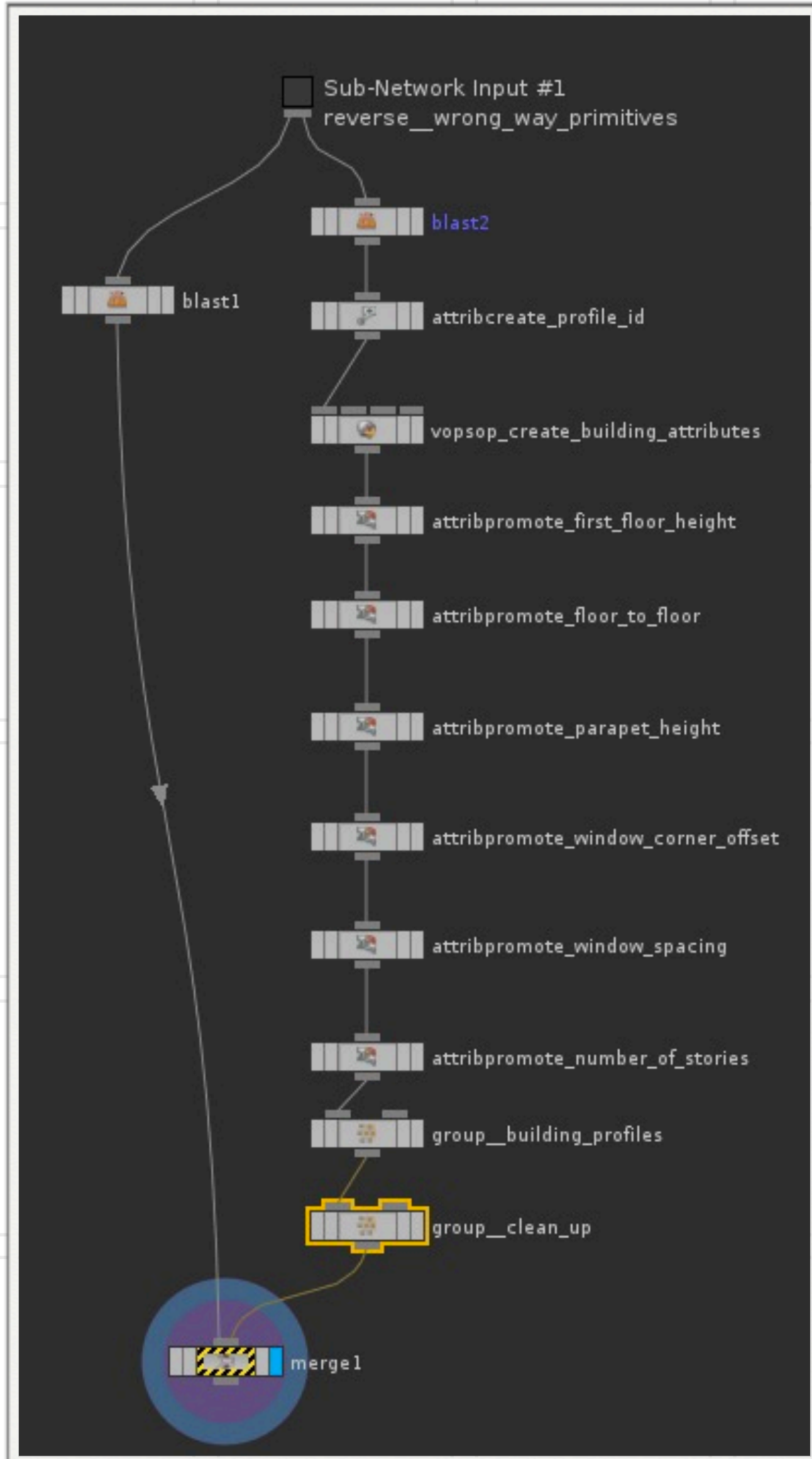
See network on following slide...



- ▶ Two more things...
- ▶ Append a Group SOP to the end of the right chain before the merge
 - ▶ name - building_profiles
 - ▶ type - primitive
- ▶ This will allow use to manipulate the profile as a whole in the Building Creator Asset
- ▶ continued on next slide...

Create_Profile_Attributes final

- ▶ It is always good to do house cleaning. We no longer need the “reverse_me” group so let us delete it Append another Group SOP to the end of the right chain before the merge
- ▶ select the “Edit tab”, select the “Delete” sub tab
- ▶ group - reverse_me
- ▶ Save the asset and match attributes





Level of Detail Generator

What is the Distance of the Building Footprint to Camera

**SIDE EFFECTS
SOFTWARE**

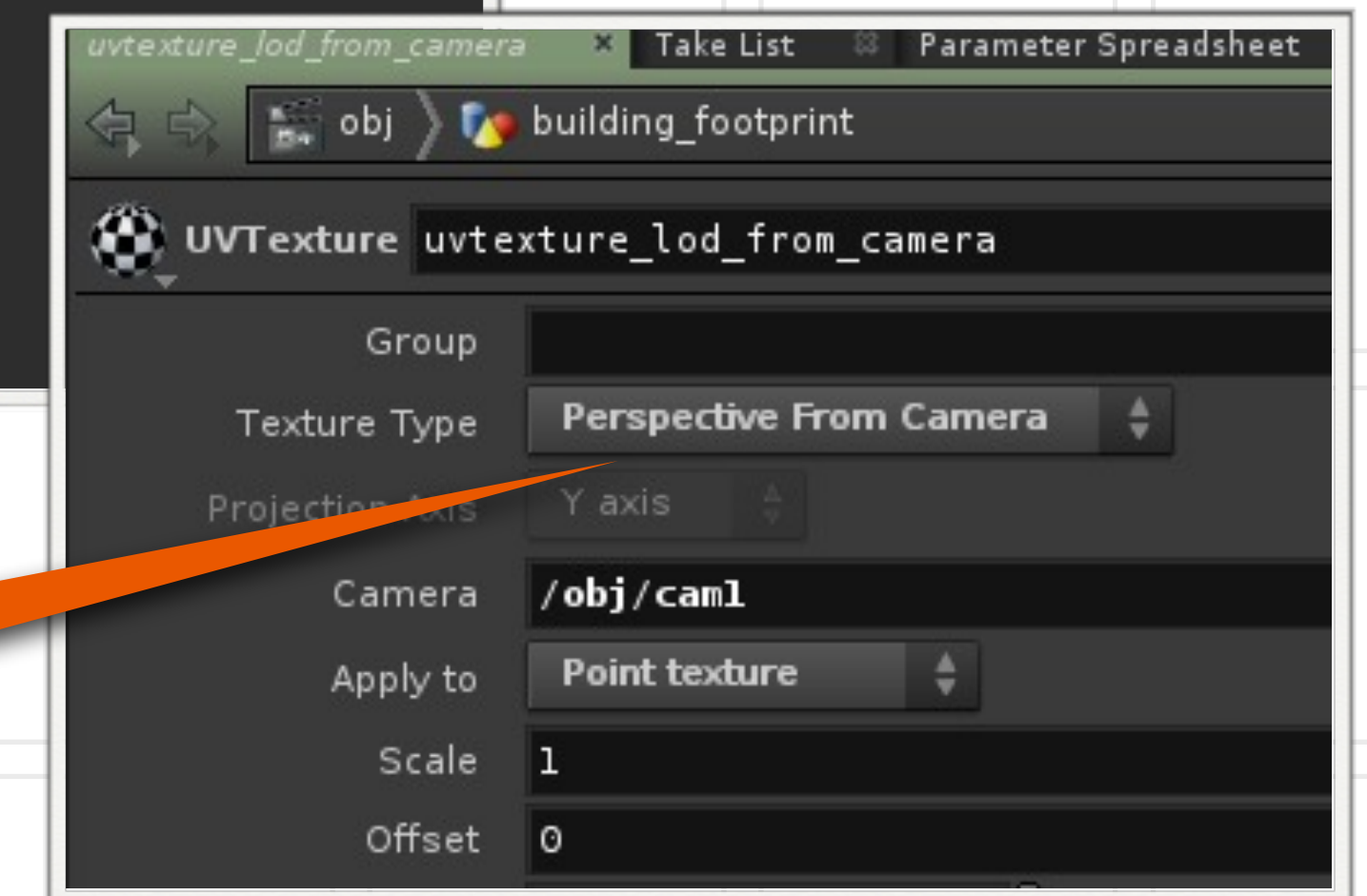
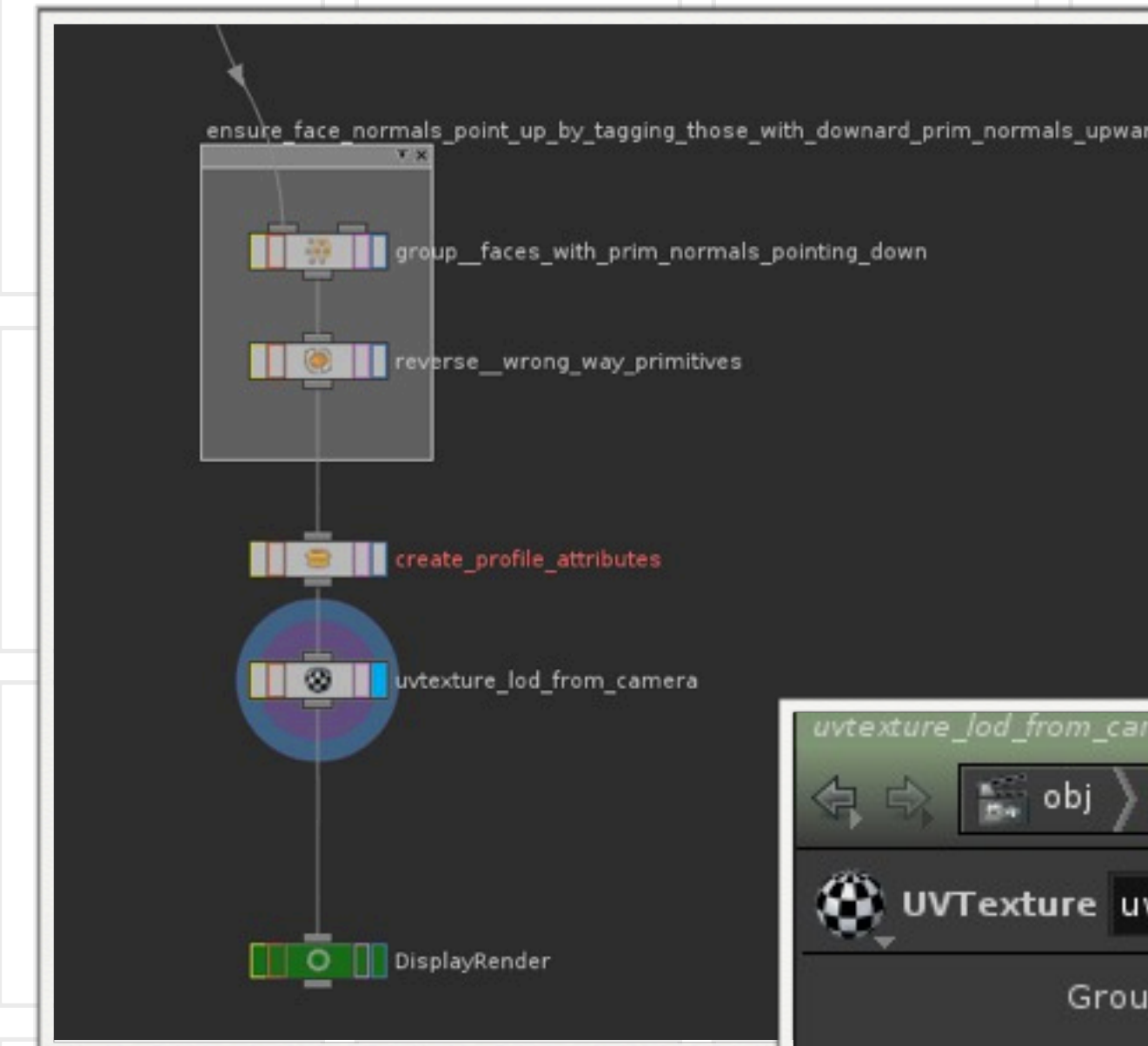
Building Level of Detail Generator

Maybe we want to automatically generate two levels of detail

- ▶ Hi-rez - Here we will build the building with all the windows if it is closer than a certain distance away from the camera. We want to add an attribute that says it is hi-rez
- ▶ Low-rez - Anything further we want to tag as low-rez

LOD Generator

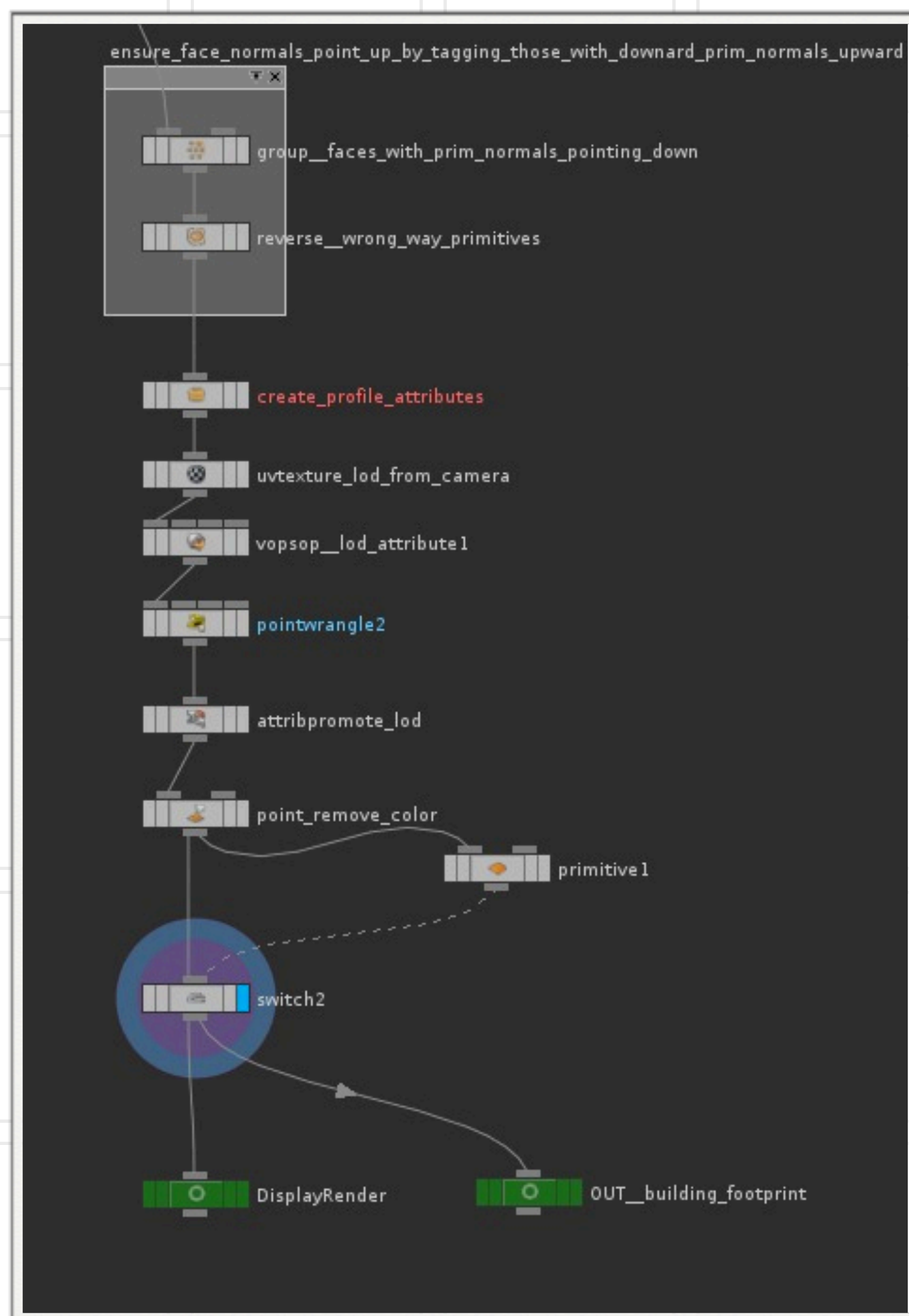
- ▶ Dive back into the Building Footprint Object if you are not there already
- ▶ Append a UVTexture SOP to the “create_profile_attributes” asset
 - ▶ Set Texture Type - Perspective from Camera
 - ▶ Set Camera - to any existing Camera
 - ▶ Apply to - Point Texture



We will use Perspective from Camera to measure the distance the profile is from the Camera

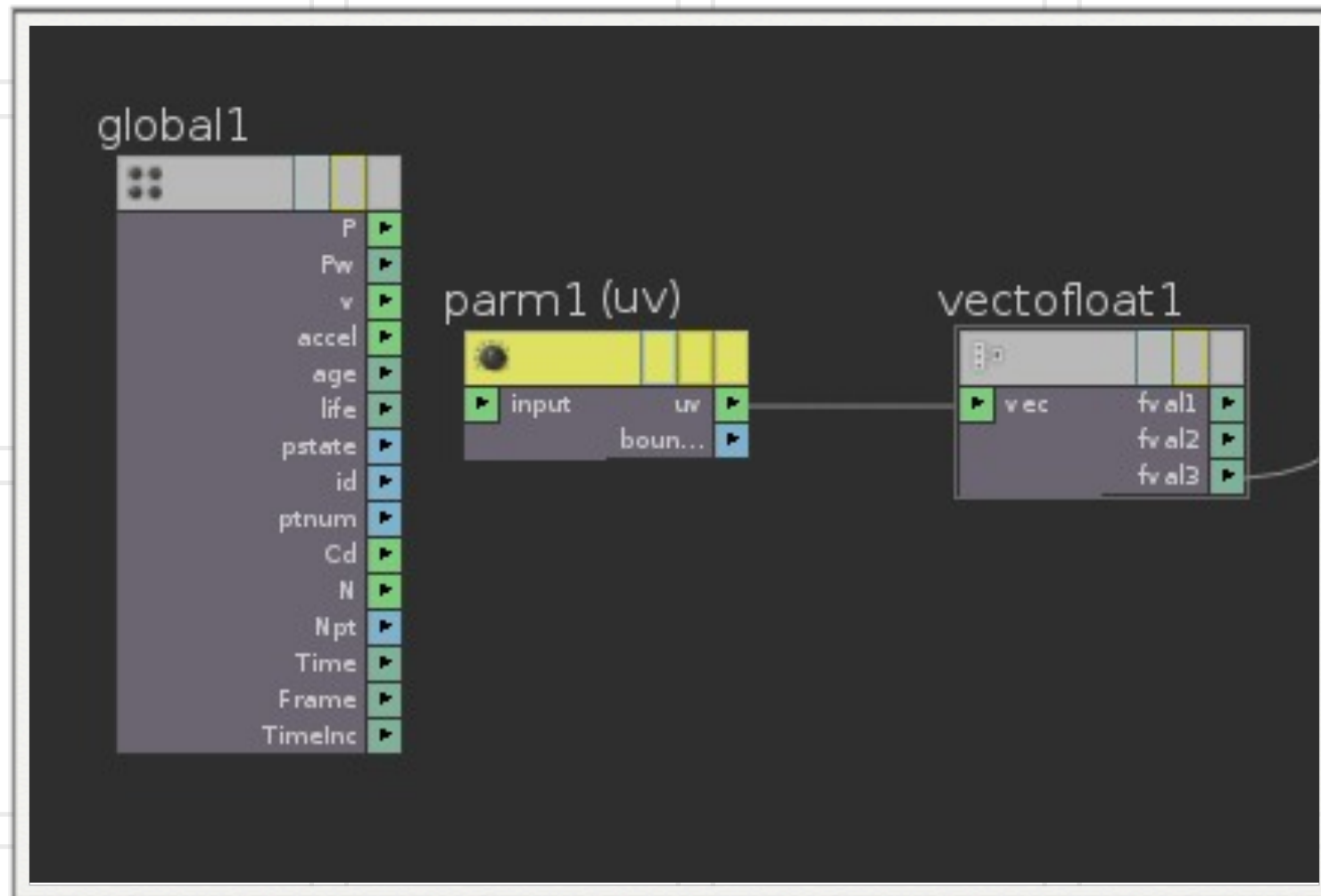
SIDE EFFECTS
SOFTWARE

LOD Generator Overview



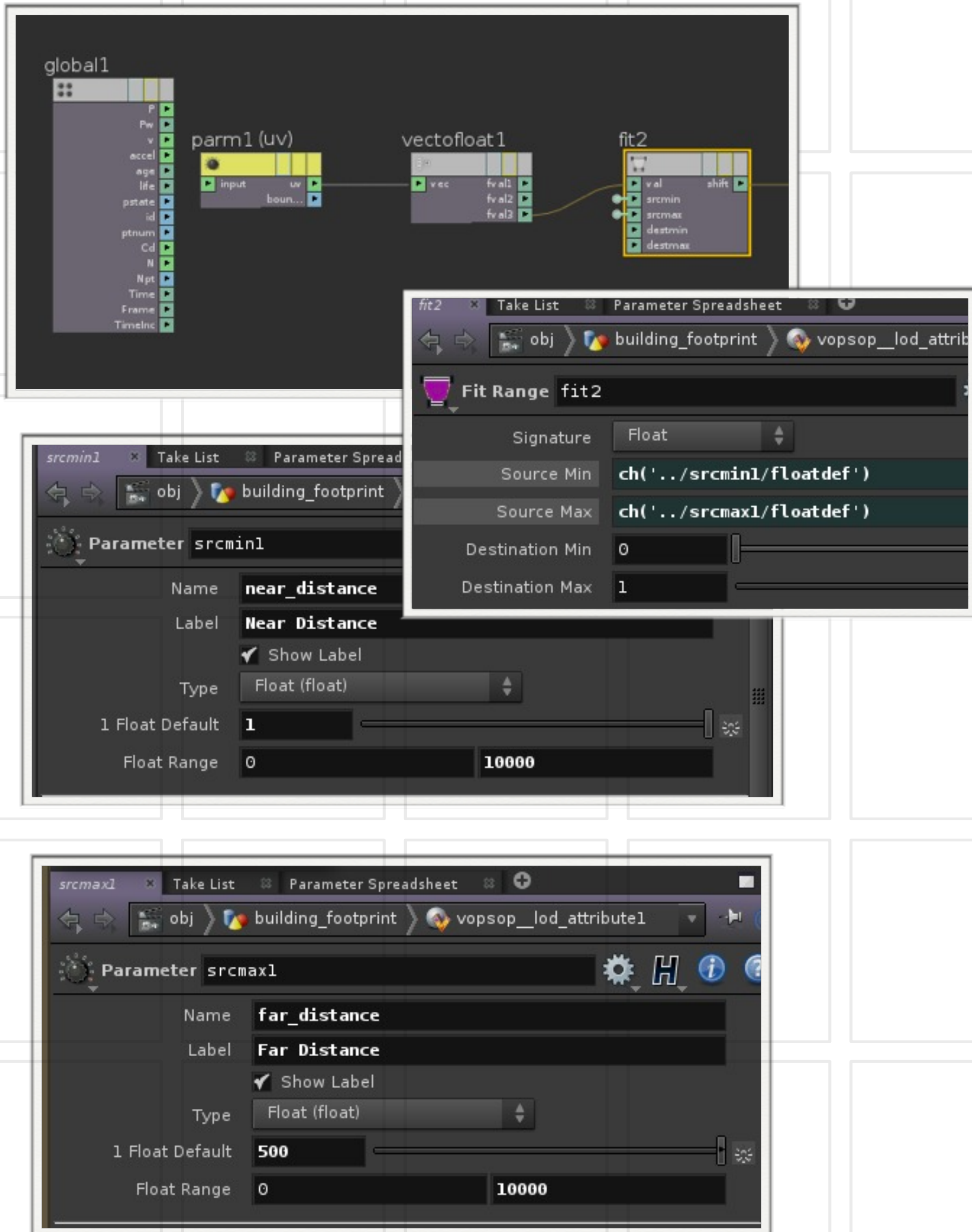
- ▶ Network we are going to create for LOD Creation on the left
- ▶ We will use a VOPSOP to convert the “w” space from the UVTexture into an attribute either Low_Rez or Hi-Rez depending on distance from Camera
- ▶ We will then use the new (H12.5) PointWrangle SOP to apply the LOD attribute to Point Color
- ▶ Then we will use the Primitive SOP to visualize the color
- ▶ Finally we will use a Switch to let the artist visualize LOD or not

LOD Generator (cont.)



- ▶ Append a VOPSOP to the UV Texture. Dive Inside
- ▶ Drop down a parameter node
 - ▶ name - uv
- ▶ Append a vector_to_float VOP
 - ▶ We only need the “w” component of the uv. The “w” component can be thought as the z-distance from the point to the camera

LOD Generator (cont.)



- ▶ Append a FitRange VOP from the “w” component of the uv to the value input of the FitRange
- ▶ Promote both SourceMin and Source max
- ▶ Select the dongle for SourceMin. In the parameters
 - ▶ name - near_distance
 - ▶ label - Near Distance
 - ▶ default 1, float range 0 - 10,000
- ▶ Select the dongle for SourceMax. In the parameters
 - ▶ name - far_distance
 - ▶ label - Far Distance
 - ▶ default 500, float range 0 - 10,000

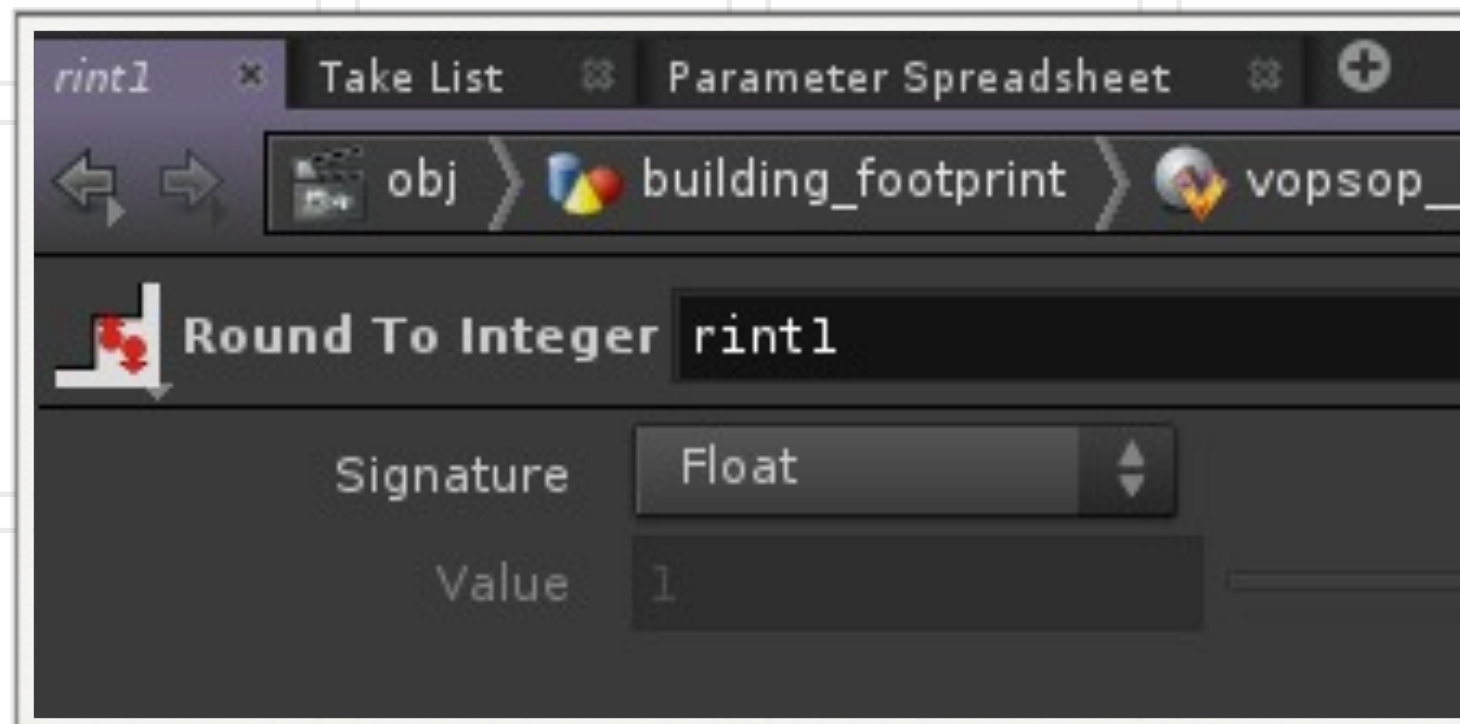
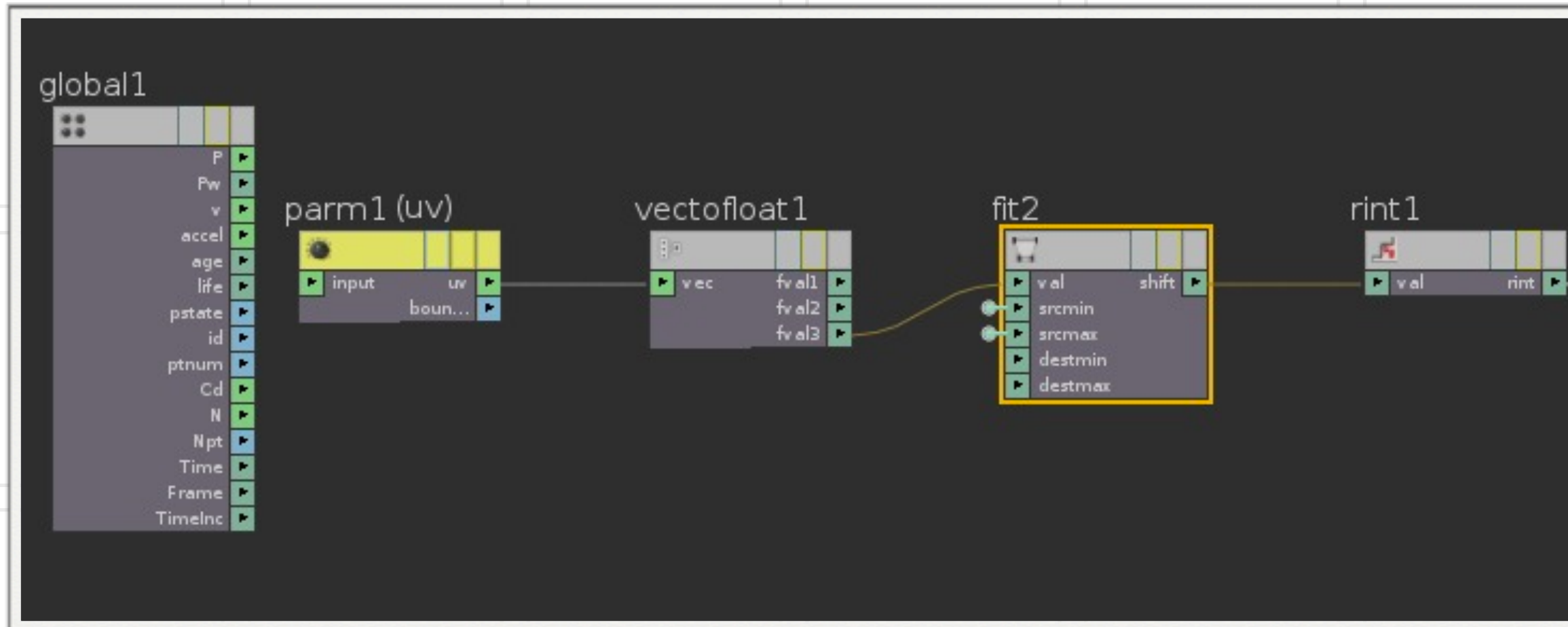
What Have We Just Done?

We took the real world coordinates of the distance from the camera to the object and fit it into a range of 0 to 1.

We allow the artist to input the min and max ranges he/she wants to evaluate for Low and Hi resolution

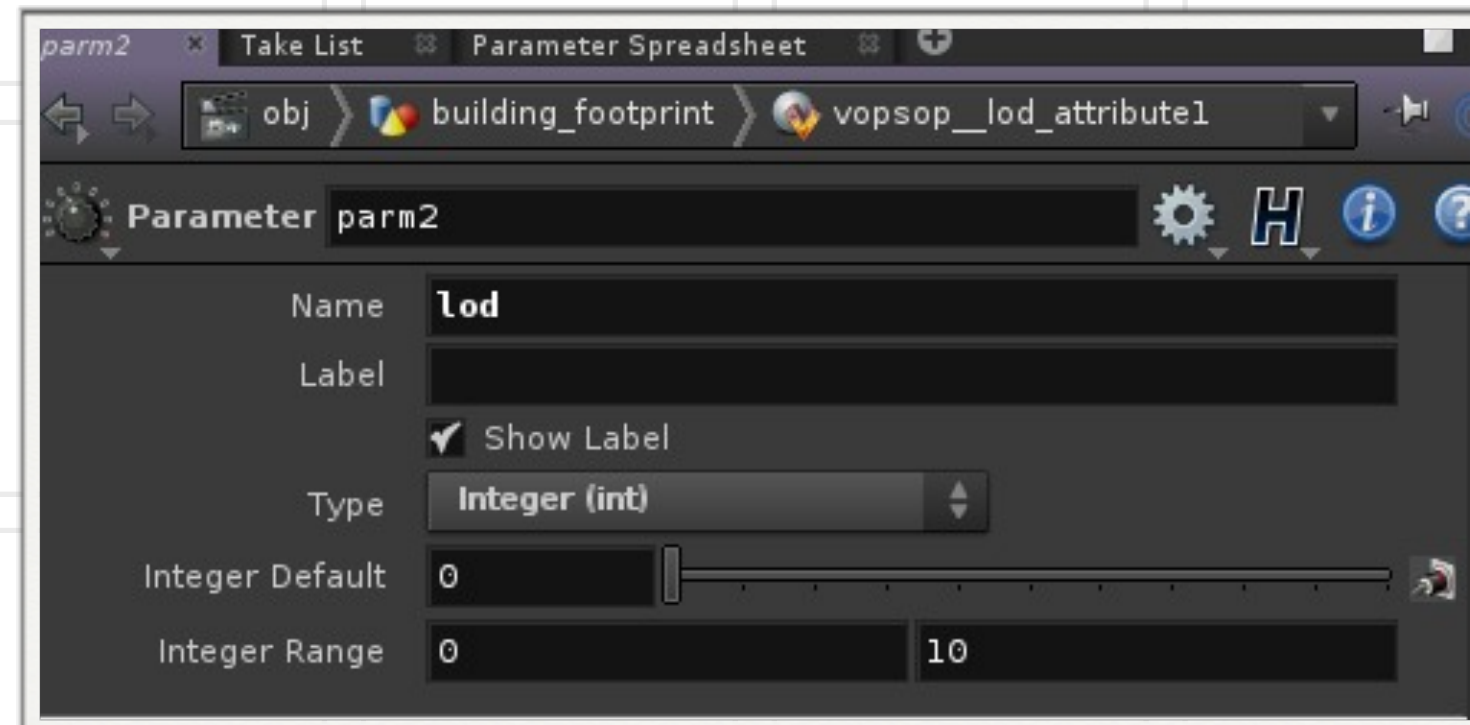
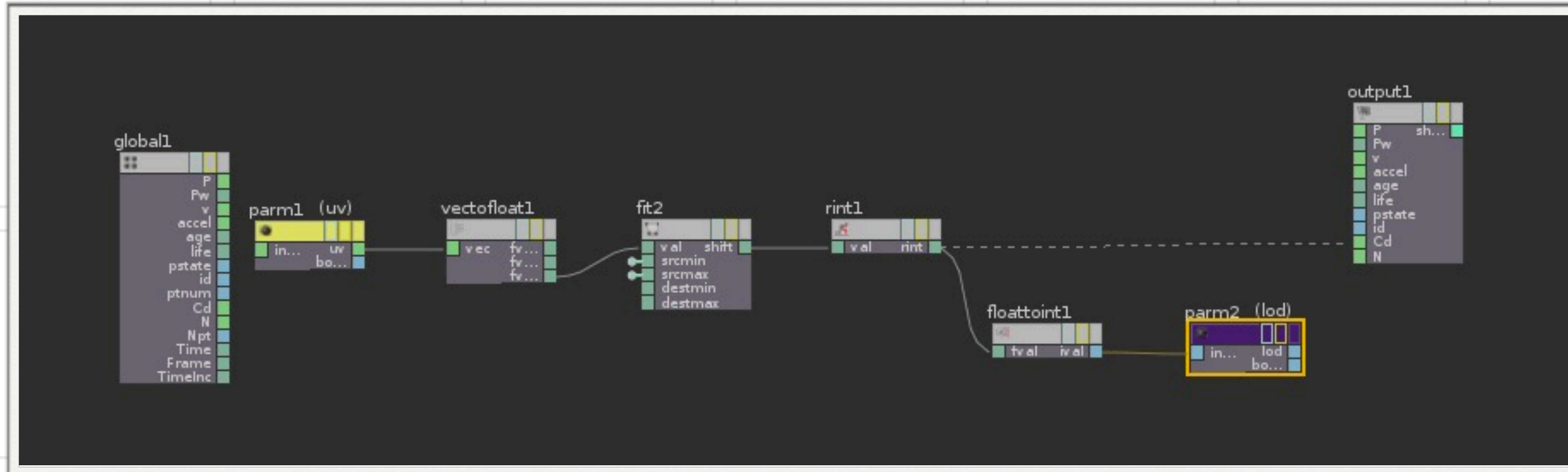
The next step will be to round the value to either 0 (High_Rez) or 1 (Low-Rez)

LOD Generator (cont.)



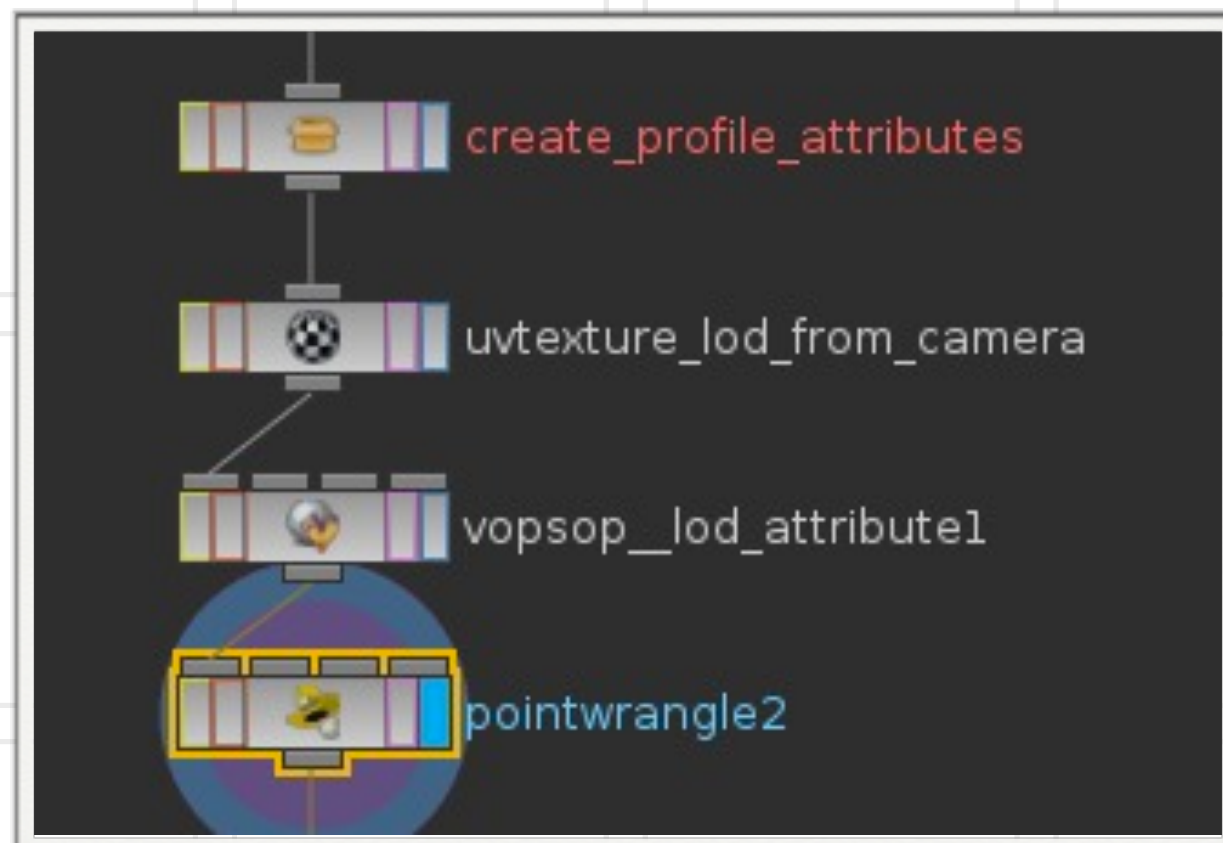
- ▶ Append a Round to Integer VOP to the fit. This will force the value to either 0 or 1
- ▶ Signature - float

Final VOPSOP for LOD

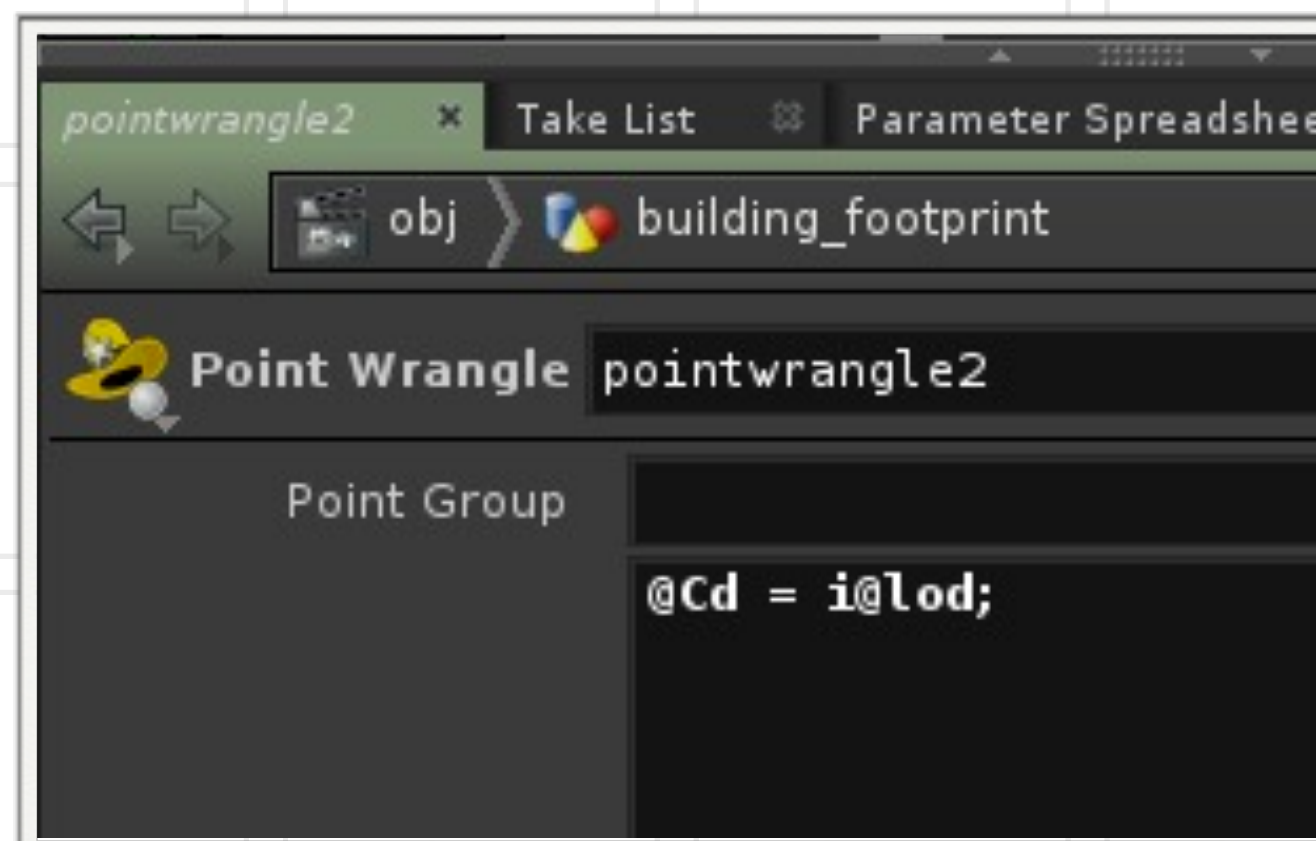


- ▶ Append a Float to Integer VOP to the rint
- ▶ Append a parameter to the Float to Integer
 - ▶ name - lod
 - ▶ type - integer
 - ▶ export - always

Append a Point Wrangle SOP



- ▶ Now the VOPSOP is complete append a PointWrangle to the VOPSOP
- ▶ `@Cd = i@lod;`
- ▶ Which means take the attribute called lod as an integer and apply it to the Cd (point color) attribute



What is the PointWrangle SOP?

Runs a VEX snippet to modify point attributes, including position.

This is a very powerful, low-level node that lets experts who are familiar with VEX tweak point attributes using code.

This node corresponds to the VOP SOP, but uses a textual VEX snippet instead of a VOP network.

For example, the following code loads the foo attribute as a vector and copies it to the P (position) attribute. You don't need to specify the type of the P attribute because it's one of the known attributes Houdini casts automatically.

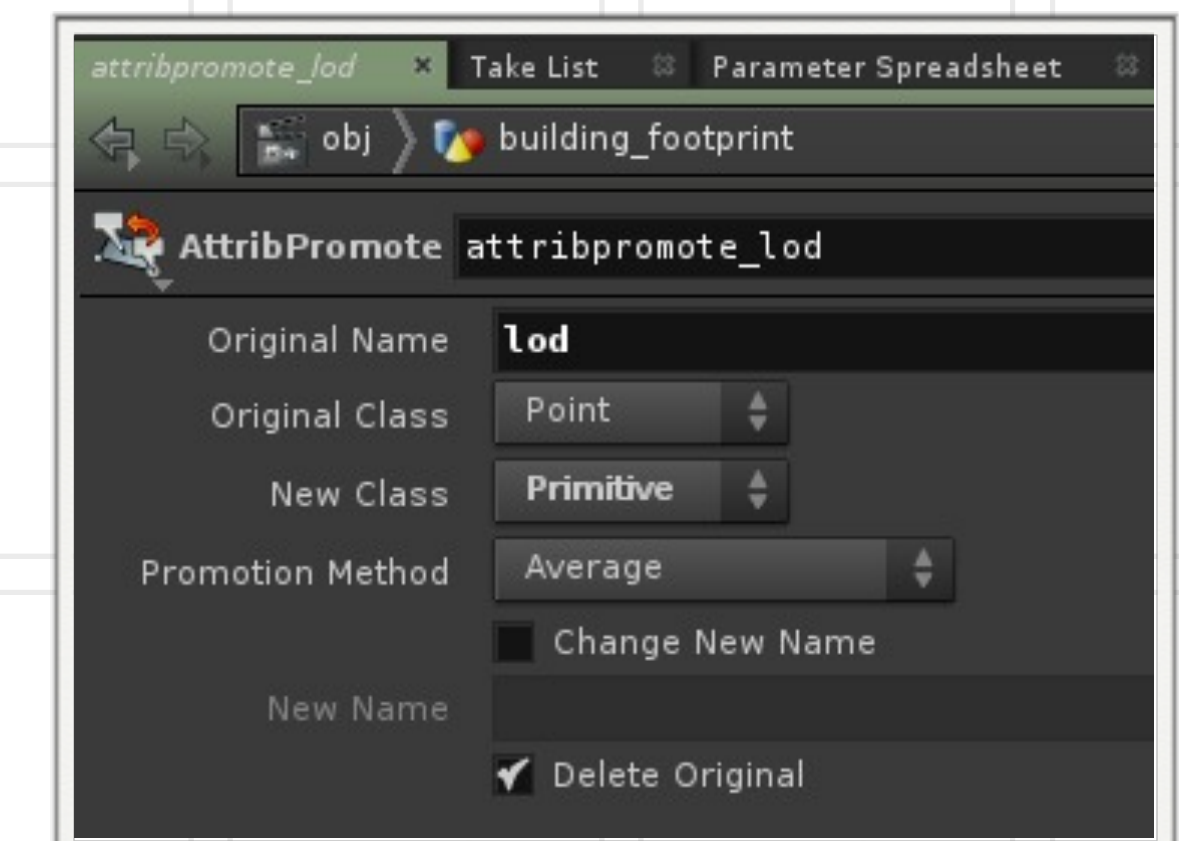
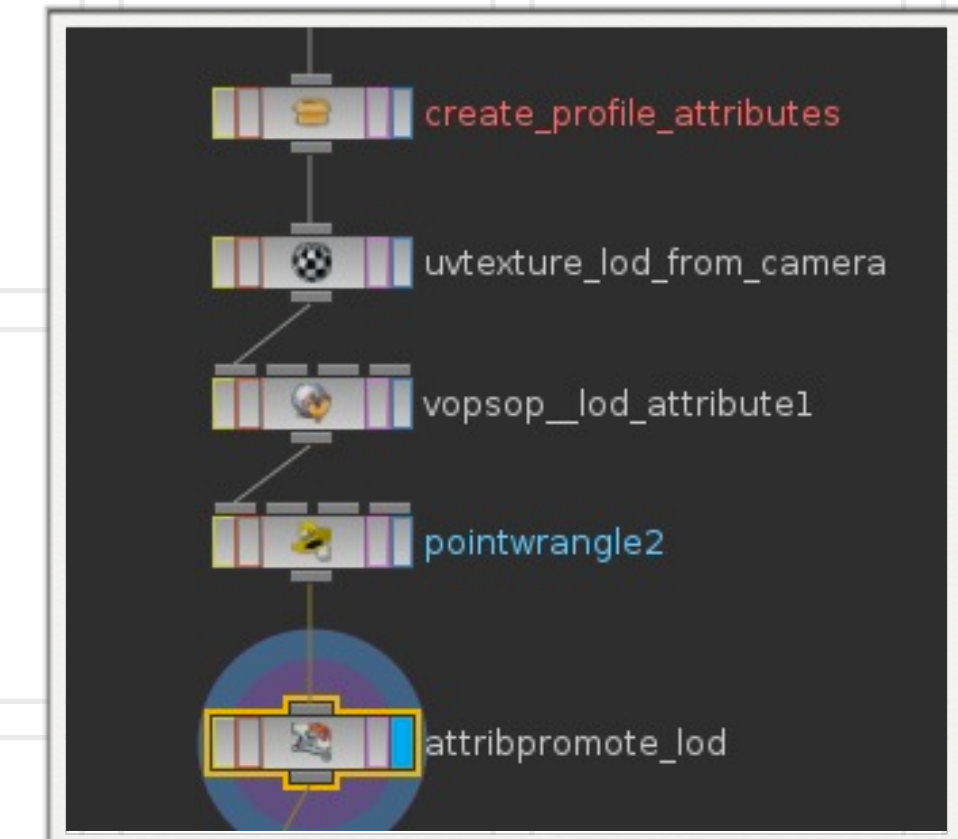
```
@P = v@foo;
```

The following code sets the x component of the Cd attribute to the value of the whitewater attribute. You don't need to specify the type of the Cd attribute because it's one of the known attributes. You don't need to specify the type of the whitewater attribute because it's a float and unknown attributes are cast as float automatically.

```
@Cd.x = @whitewater;
```

Append an Attribute Promote

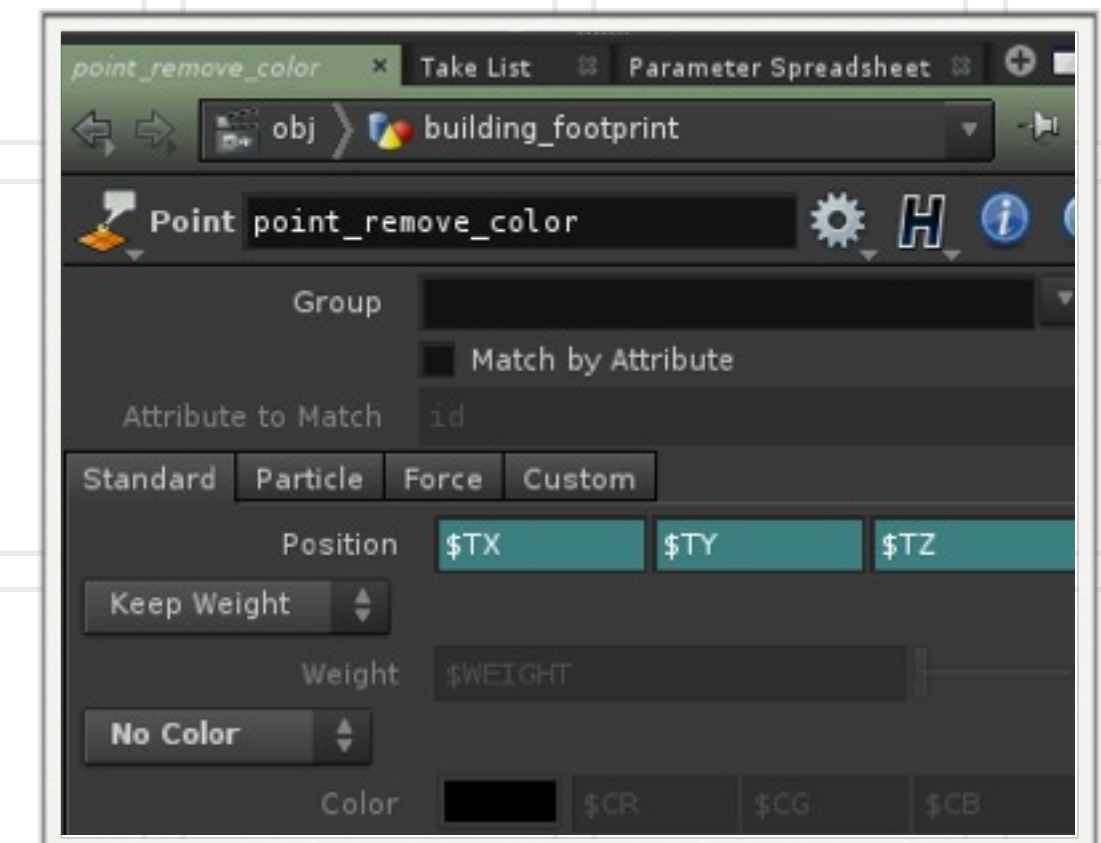
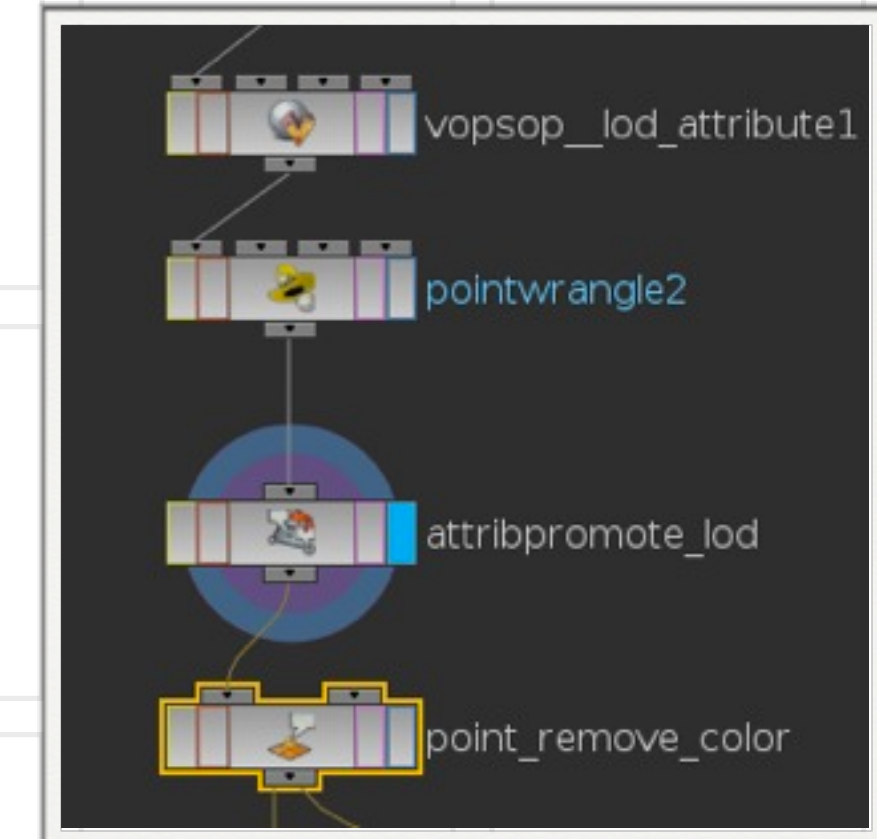
- ▶ Write now we have point attributes for LOD. But we do not want to build part of a building as low-rez and part as high-rez. Therefore we are going to promote the lod to a primitive. When done each profile will only have one prim LOD
- ▶ Original Name - lod
- ▶ Original Class - Point
- ▶ New Class - Primitive
- ▶ Promotion Method - Average
- ▶ Select - Delete Original (We do not need the point attribute)



**SIDE EFFECTS
SOFTWARE**

Let's Remove Point Color

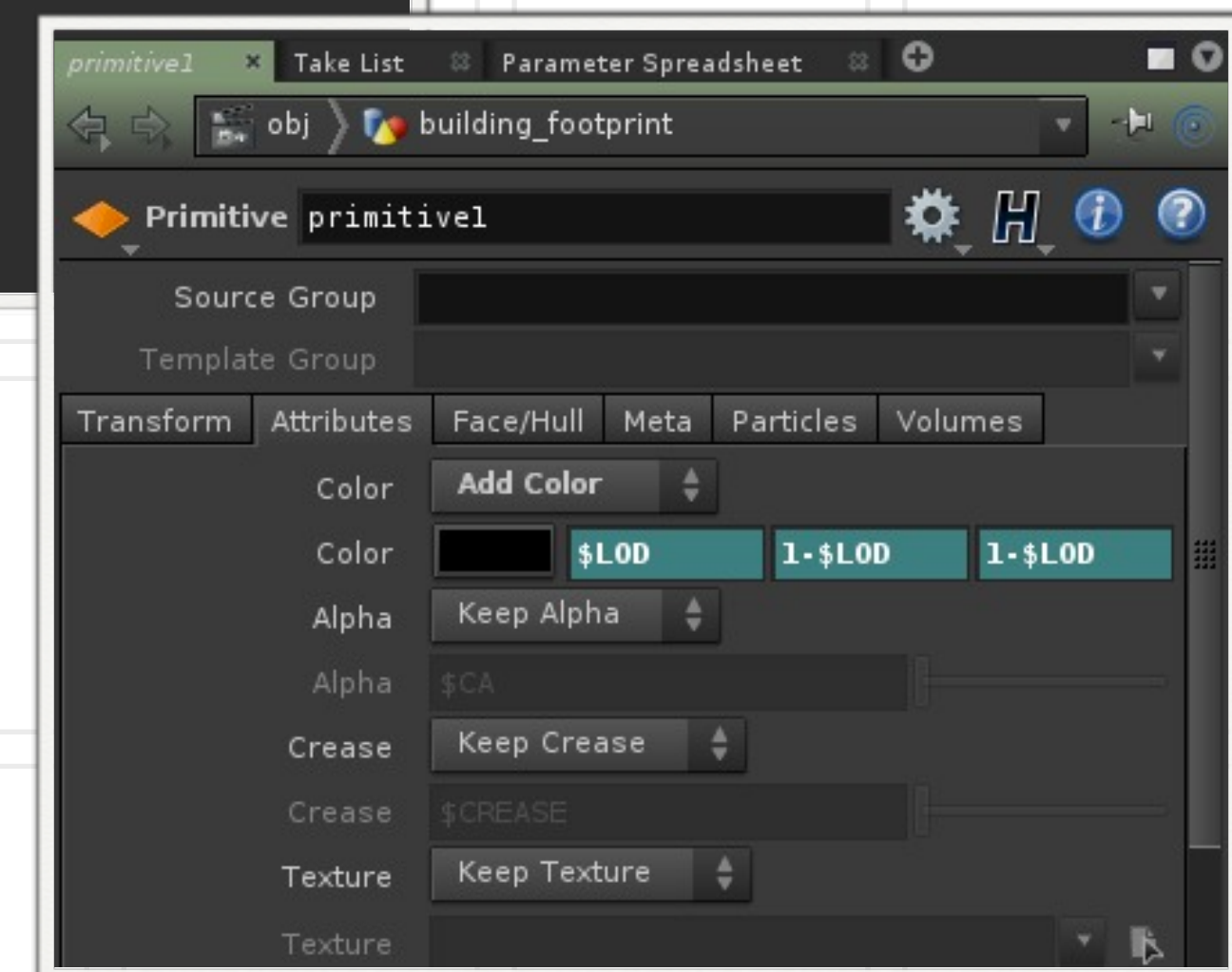
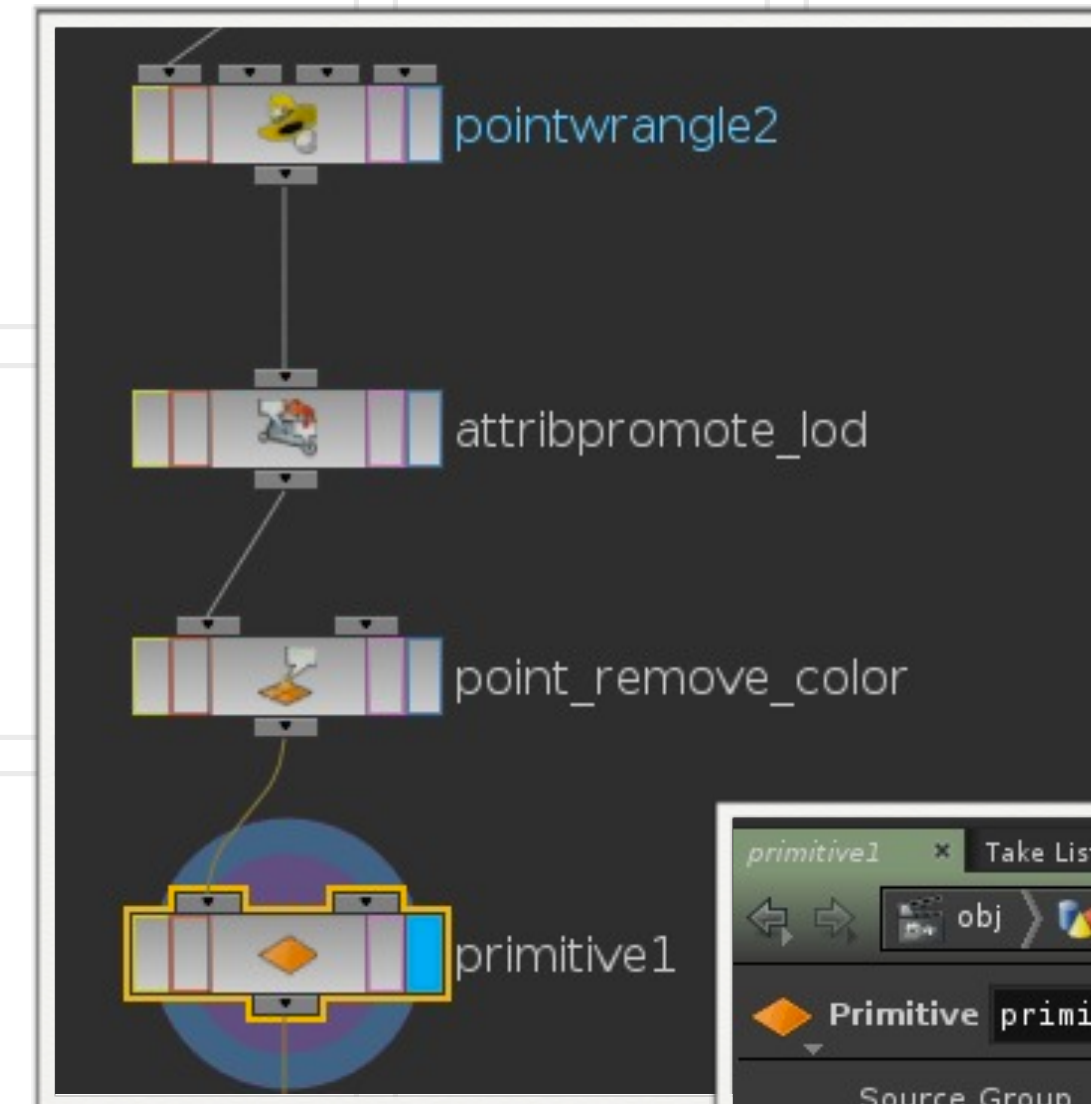
- ▶ We want to give the artist the choice to visualize the LOD or not
- ▶ Drop down a Point SOP
- ▶ In the Color Menu select “No Color”



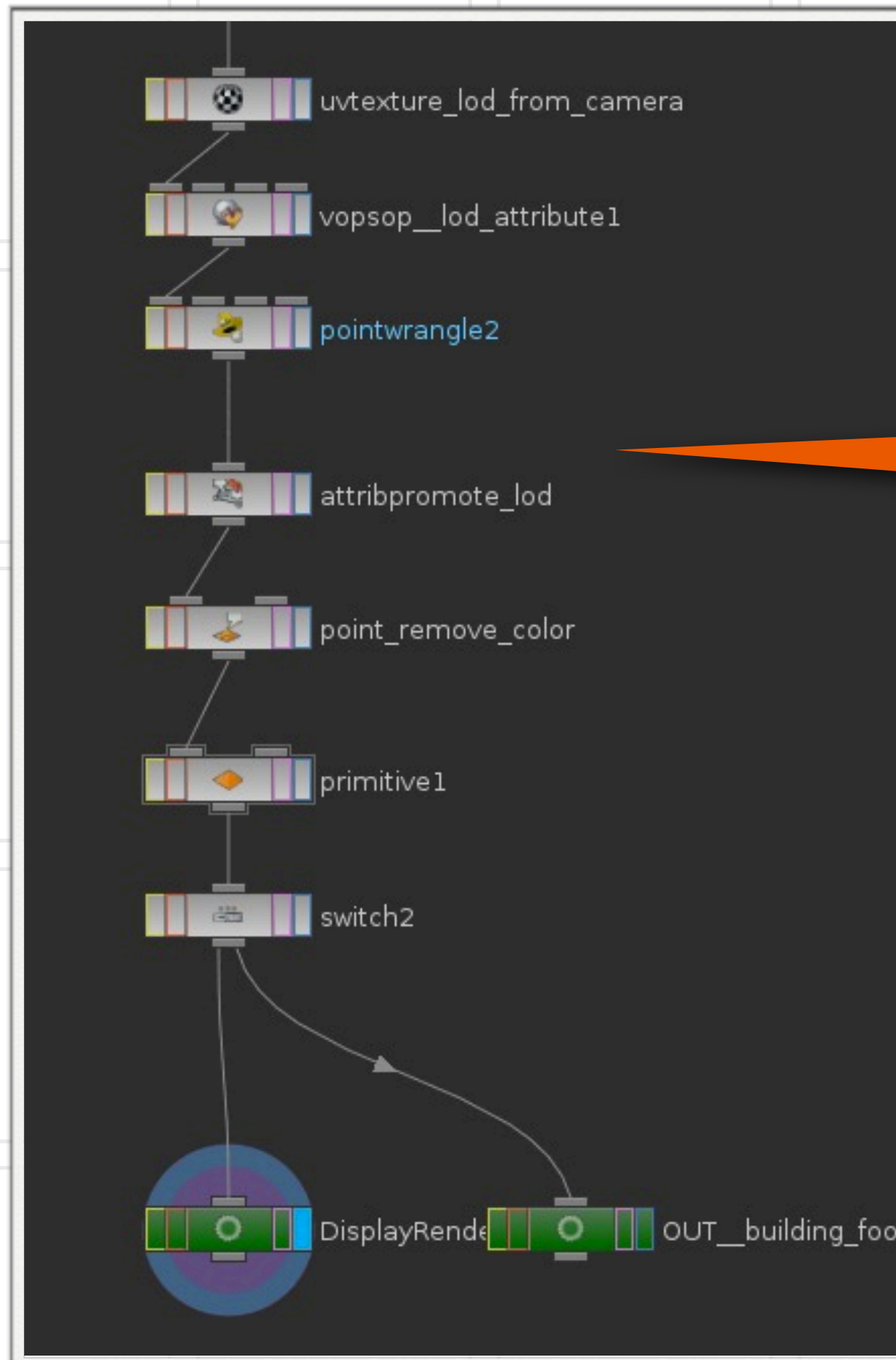
**SIDE EFFECTS
SOFTWARE**

Visualize the LOD

- ▶ Append a Primitive SOP
- ▶ For Color - Add Color
- ▶ Color (r,g,b) - \$LOD, 1-\$LOD, 1-\$LOD

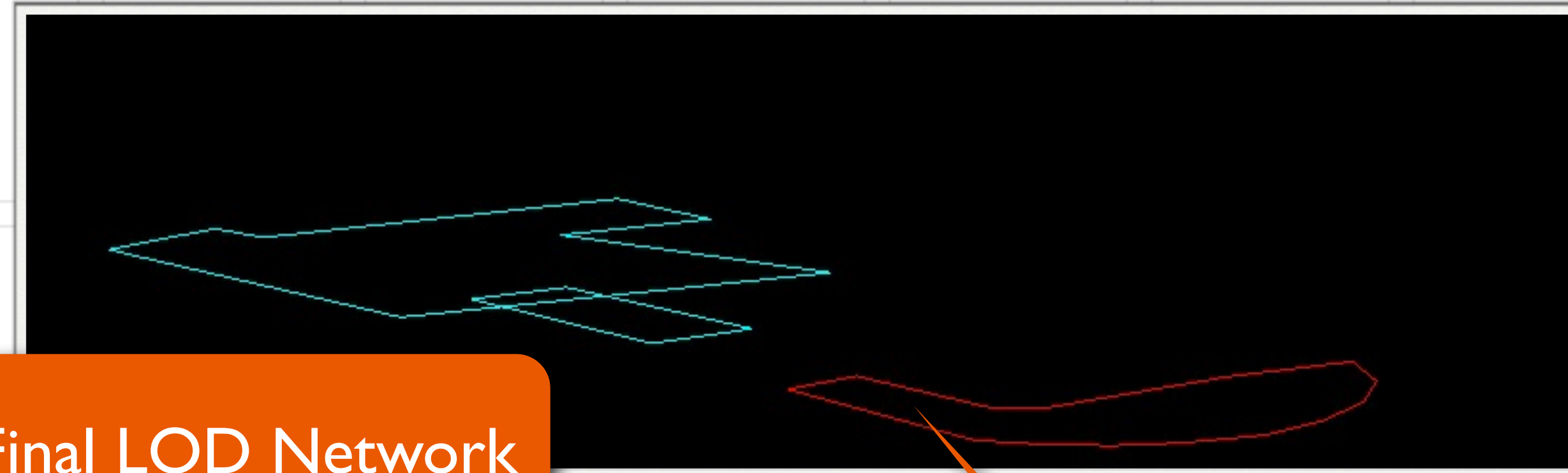


Append A Switch



Final LOD Network

- We want the Artist to decide if he/she wants to visualize the LOD so append a Switch with no color as the first entry and Prim color as the second entry



Color Visualization of LOD



Assignment - Wrap up the LOD Network into a Digital Asset

Do it yourself

**SIDE EFFECTS
SOFTWARE**



Modifying the Building Generator to Use LODs and new Create Profile Attribute Asset

**SIDE EFFECTS
SOFTWARE**

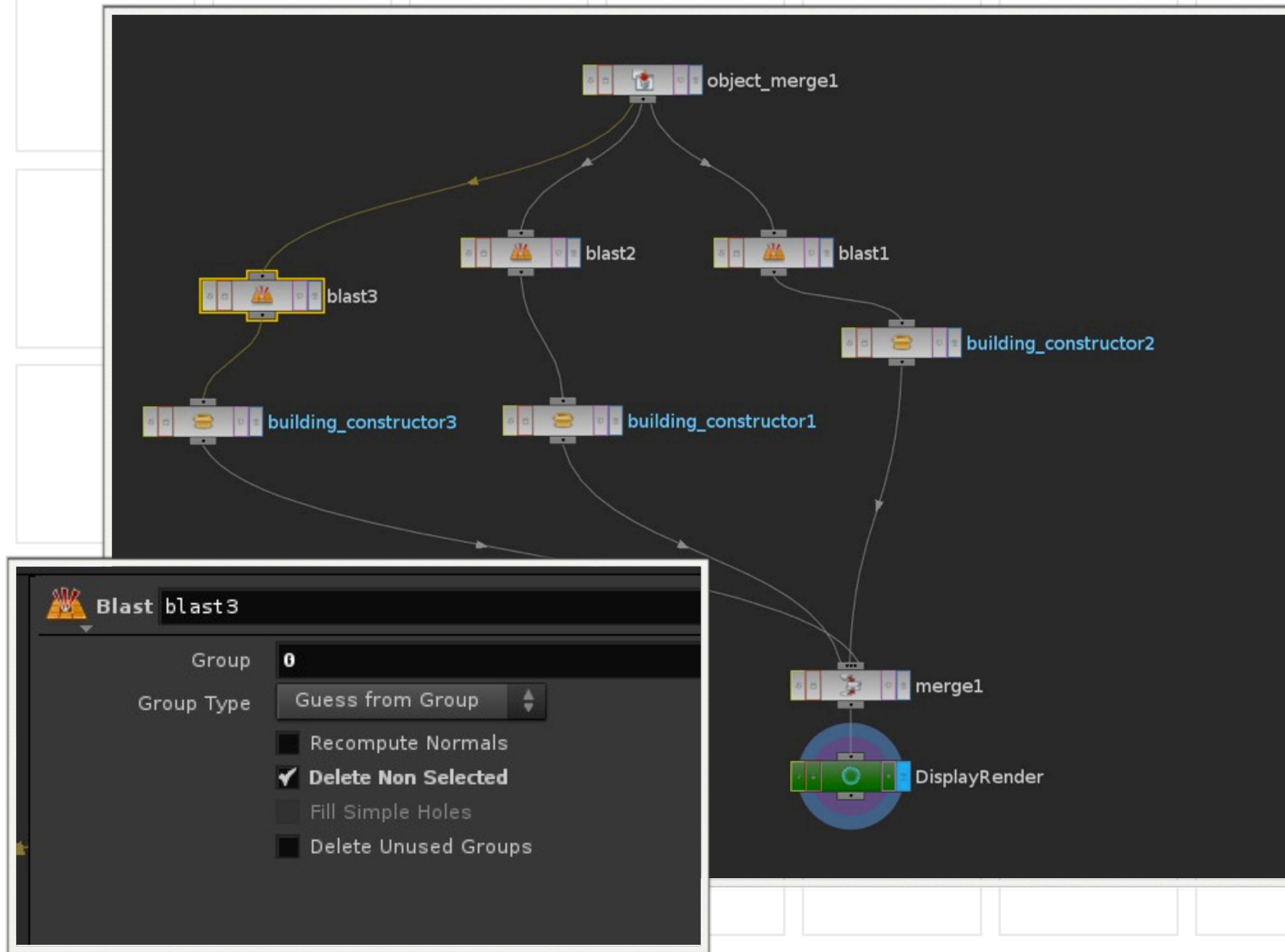
What is chsop() ?

string chsop (string channel)

Evaluates the parameter at the current time as a node path string.

The string is assumed to be a path to one or more nodes, node groups, or bundle names. If the nodes or groups are specified as relative paths, they are converted to absolute paths using the referenced node as the source for the relative paths.

Need to Modify Building Generator



- ▶ Need to Prepend the Building Constructor with a Blast. Put the Profile Number in the Group and select “Delete Non Selected”



End Module 04

**SIDE EFFECTS
SOFTWARE**