HOUDINI FOUNDATIONS

CREATE A TASK WORKFLOW TO BUILD CITIES

To manage pipeline workflows, you can turn to Task Operators (TOPs) which are built using a technology called the Procedural Dependency Graph (PDG). Workflows created using PDG use TOP nodes that generate work items that distribute tasks to either your local computer or a larger compute farm.

The TOP network lets you determine the dependencies between different work items and how they are contributing to the final output. This information is easy to visualize in the node graph which can be used to define how you want data to flow through your network. TOPs lets you build workflows that can be used to automate, analyze and scale your pipeline.

In this lesson, you will use TOP nodes to take a city map, create buildings for each city block and then grow this system to handle more complex buildings and larger city maps. Houdini artists will probably know how to do this in SOPs, but by using TOPs, you can learn the PDG workflow while creating a system that can be easily scaled to distribute multiple tasks to an external compute farm to process in parallel.

NOTE: *This lesson uses Image Magick - make sure this app is installed on your computer.*

LESSON GOAL

- *To create a TOP (Task Operator) Network to build up a procedural city and then render it out.*

WHAT YOU WILL LEARN

- How to convert a city map image into geometry
- How to set up a TOP network to save out the city blocks
- How to create Geometry in TOPS to build buildings on each city block.
- How to create a city core to make some buildings higher than others
- How to wedge the cityscape to try different locations for the city core
- How to wedge the use of different map images
- How to use TOPs to render out the city and compare wedges using an image mosaic

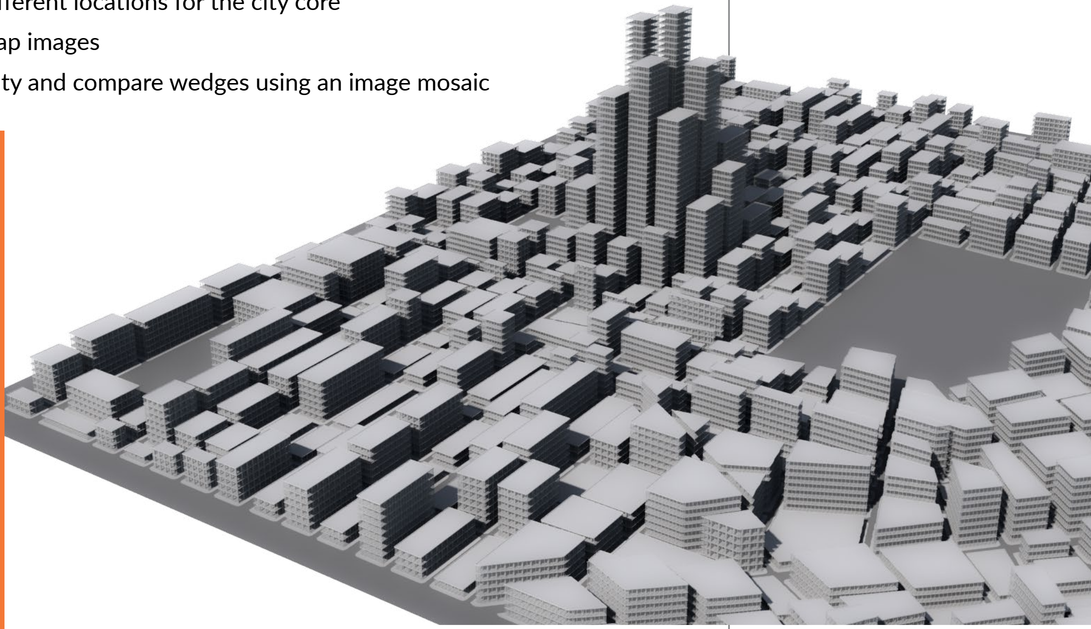
LESSON COMPATIBILITY

Written for the features in Houdini 19.0

The steps in this lesson can be completed using the following Houdini Products:

Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 2.0 | October 2021

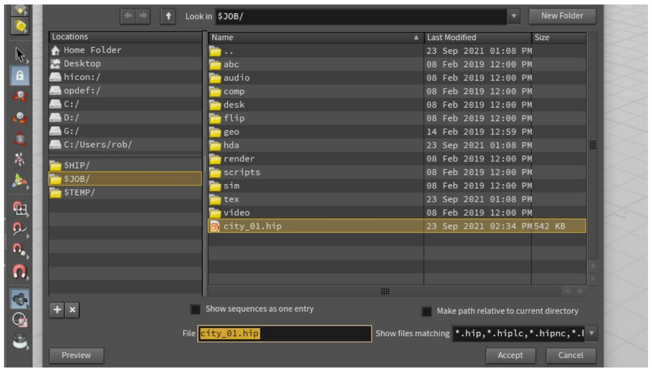


PART ONE: Create a City Grid

To build a procedural city, you will start with a city grid. You are going to create the geometry by tracing an image file that includes a black and white image of a city map. This will be the input geometry for the network you will be building in TOPs.

PROJECT FILES

Go to the **Foundations tutorial** page on **SideFX.com**, where you got this tutorial, and download the ***tops_lesson*** directory. Put it into the **Houdini Projects** directory which you can find in either the **home** directory or the **documents** directory.



Look in $JOB/

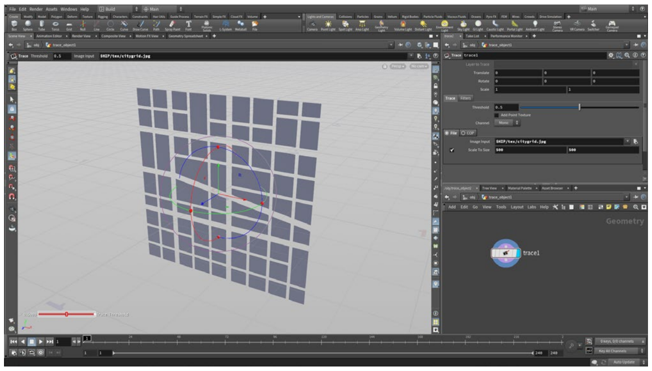
Locations
Home Folder
Desktop
hicon:/
opdef:/
C:/
D:/
G:/
C:/Users/rob/
$HIP/
$JOB/
$TEMP/

Name	Last Modified	Size
..	23 Sep 2021 01:08 PM	
abc	08 Feb 2019 12:00 PM	
audio	08 Feb 2019 12:00 PM	
comp	08 Feb 2019 12:00 PM	
desk	08 Feb 2019 12:00 PM	
flip	08 Feb 2019 12:00 PM	
geo	14 Feb 2019 12:59 PM	
hda	23 Sep 2021 01:08 PM	
render	08 Feb 2019 12:00 PM	
scripts	08 Feb 2019 12:00 PM	
sim	08 Feb 2019 12:00 PM	
tex	23 Sep 2021 01:08 PM	
video	08 Feb 2019 12:00 PM	
city_01.hip	23 Sep 2021 02:34 PM	542 KB

Show sequences as one entry
Make path relative to current directory
File city_01.hip
Show files matching *.hip,*.hiplc,*.hipnc,*.

01 Select **File > Set Project.** Find the *tops_lesson* directory that you downloaded and press **Accept**. This makes this project directory and its sub folders the place for all the files associated with this shot.

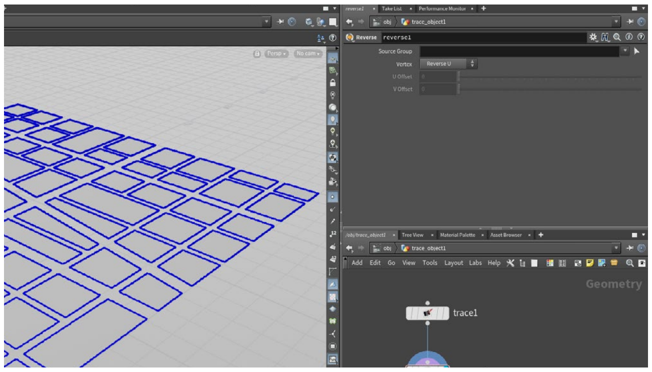
Select **File > Save As...** You should be looking into the new *tops_lesson* directory. If not then click on **$JOB** in the left hand column. Set the file name to *city_01.hip* and click **Accept** to save.



Trace Threshold 0.5 Image Input $HIP/tex/citygrid.jpg

02 In the viewport, **press tab** to bring up a menu and start typing **TRACE**. Choose **Trace** and your cursor now shows the outline of a square waiting to be placed in the scene. **Press Enter** to place it at the origin. Right now it is a traced circle.

Double-click on the node in the Network view to dive down to the geometry level. Select the *trace* node and click on the **File chooser** next to the **Image Input** parameter. Click on **$HIP** then navigate into the *tex* directory. Select the *citygrid.jpg* image and then click **Accept**. Next turn **ON** the **Scale to Size** option and set it to **500, 500**. This will give you more accuracy.



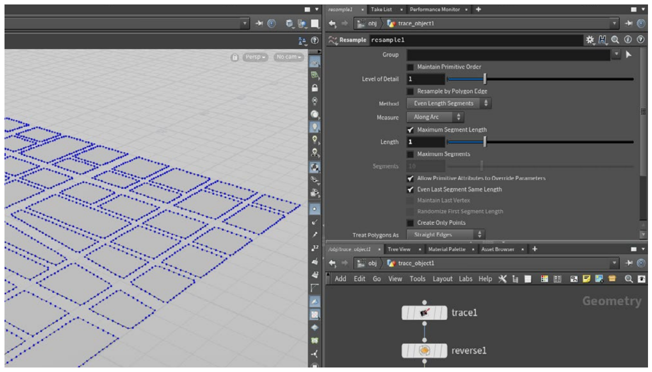
Reverse reverse1
Vertex Reverse U
U Offset 0
V Offset 0

03 In the upper section of the *trace* node, set the following:

- **Rotate X** to **-90**
- **Scale** to **100, 100**

In the Network view, press **tab** and start typing **Reverse.** Choose the **Reverse** node and place it down then wire the *trace* node into it. Set its **Display Flag**. This node will point the normals up. In the viewport, press **Spacebar-H** to view the whole city grid.

Turn on **Display Points** in the **Display option** bar to see that there are lots of trace points around each city block.



Resample resample1
Maintain Primitive Order
Level of Detail 1
Resample by Polygon Edge
Method Even Length Segments
Measure Along Arc
✓ Maximum Segment Length
Length 1
Maximum Segments
Segments 10
✓ Allow Primitive Attributes to Override Parameters
✓ Even Last Segment Same Length
Maintain Last Vertex
Randomize First Segment Length
Create Only Points
Treat Polygons As Straight Edges

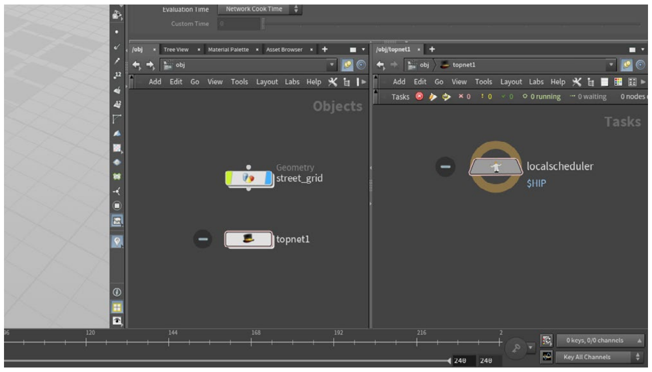
04 In the Network view, press **tab > Resample** and click to place it under the *reverse* node. Don't wire it in yet. Set **Length** to **1** then connect the output of the *reverse* node into the *resample* node's input.

RMB-click on the output of the *resample* node and choose **Null**. Click to place the null node at the end of the chain. Set its **Display flag** and rename it *CITYBLOCKS_OUT*.

Save your work.

PART TWO:
Generate and Display Work Items

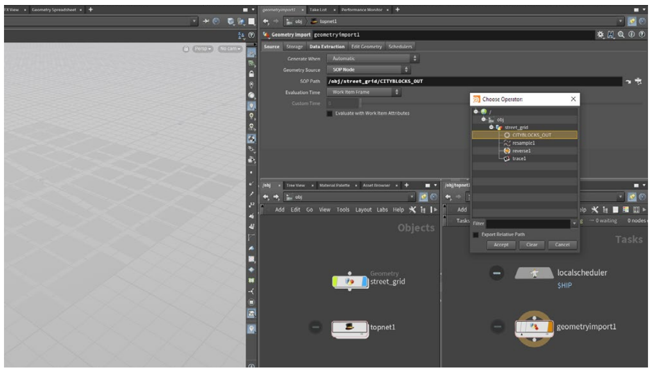
Now that you have a city grid, you can break out the different city blocks into separate work items. That will allow you to use each block to generate buildings. You will set up a TOP network to save out each of the blocks. At the same time, you will create an object for visualizing selected work items to verify that you are getting the results you want.



01

Go back to the object level and rename the object to *street_grid*. Go to the Network view and **press Tab** and start typing **TOP...** and select **TOP Network**. Click to place the node in the network.

Click on the little **arrow** in the top right of the Network View. From the menu choose **Split Pane Left/Right.** Click on the **Pin** icon on the Network view to the left. The other Network view is already pinned. In the Network view on the right, **double-click** on the *topnet* to dive into it. You can now work with both networks as you develop the city.



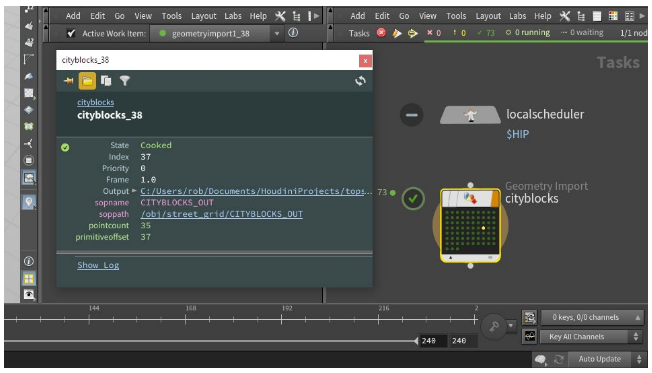
Geometry Import geometryimport1

Generate When: Automatic
Geometry Source: SOP Node
SOP Path: /obj/street_grid/CITYBLOCKS_OUT
Evaluation Time: Work Item Frame

02

In the TOP network press **tab > Geometry Import**. Click to place the node. Set **Geometry Source** to **SOP Node** and then click on the **Choose Operator** icon next to **SOP Path.** Use the floating window to navigate to and select the *CITYBLOCKS_OUT* null node.

Under **Storage**, leave **Store Geometry As** set to **External File.** Under Data Extraction, set **Copy from Class** to **Primitive**. This will store the files to disk so that they can be retrieved by the next TOP node. This is especially important if you set up a TOP network to distribute tasks to a compute farm.



Active Work Item: geometryimport1_38

cityblocks_38

cityblocks
cityblocks_38

```
         State  Cooked
         Index  37
      Priority  0
         Frame  1.0
        Output  C:/Users/rob/Documents/HoudiniProjects/tops...
      sopname   CITYBLOCKS_OUT
      soppath   /obj/street_grid/CITYBLOCKS_OUT
    pointcount  35
primitiveoffset 37
```

Show Log

03

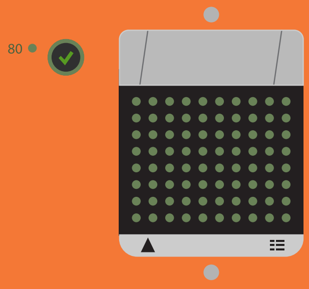
Rename the *geometryimport* node to *cityblocks* and from the **Task bar** at the top, click on the **Cook Selected Node** button. You could also select it and hit **Shift-G** to cook it. You will see dots representing work items appear as the node is cooked.

Ctrl-MMB-click on the dots to view the work item attributes. Some attributes like **Index** are basic tops attributes that every work item will have. Others are specific to the type of work item. You will see that each work item is associated with an **Output** file. You can **RMB-click** on a work item and choose **View Work Item Output** to see the geometry it contains in a separate geometry viewer.

WORK ITEMS

TOP nodes create work items for each task that you ask it to perform. These are represented on the TOP node as dots and you can use them to visualize their status and can select them individually to evaluate how that work item is progressing.

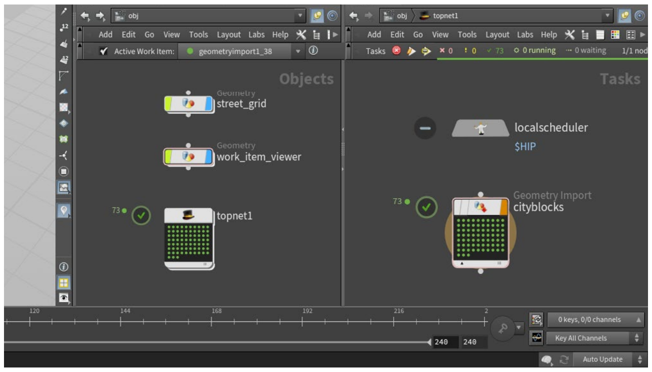
When all the work items are fully cooked, the node gets a checkmark and the completed work items are colored green.



WHAT IS THE SCHEDULER NODE?

When you first create a TOP network, a local scheduler node is created to organize tasks as they are cooked. The local scheduler points to your local computer and will use a percentage of available cores to cook. You can set **Total Slots** to **Equal to CPU Count Less One** option to match available cores.

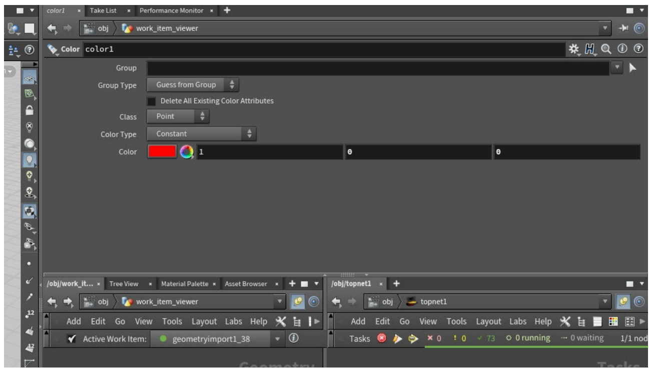
You can also set up scheduler nodes for **HQueue, Deadline, Tractor** and **Python** to send tasks to a larger compute farm. This will allow more work items to be processed in parallel to make your graph more efficient.



04 If you want to speed up processing on your network, select the *localscheduler* node and set **Total Slots** to **Equal to CPU Count Less One.** Now future cooks will be faster because more processors are being used.

To see the work items in the viewport, you will need to set up a work item viewer. In the Network view on the left, **press tab > File**. Click to place the node and rename it *work_item_viewer*.

Normally you would point a File node to a file on disk. You will do this at first to find the files then use a different method to acquire the work item directly from TOPS.

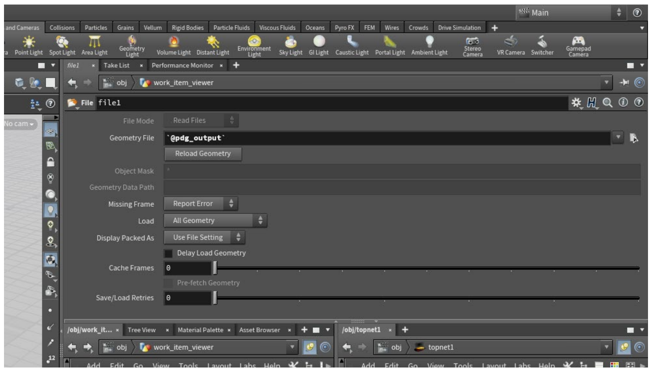


Color color1

Group
Group Type: Guess from Group
Delete All Existing Color Attributes
Class: Point
Color Type: Constant
Color: 1 0 0

05 Double-click to dive inside the *work_item_viewer* node and in the Parameter pane, click on the the **Geometry File** file selector button. In the File selector, click on **$HIP** and then **double click** on the *geo* folder.

Select the *city_01_cityblocks* file sequence. This imports the saved out geometry as a numbered sequence. Add a **color** node after the geometry sequence and set its color to **red (1, 0, 0)**. Scrub in the timeline to see the different pieces of geometry loading in sequence. There are 73 pieces that have been saved out to disk.



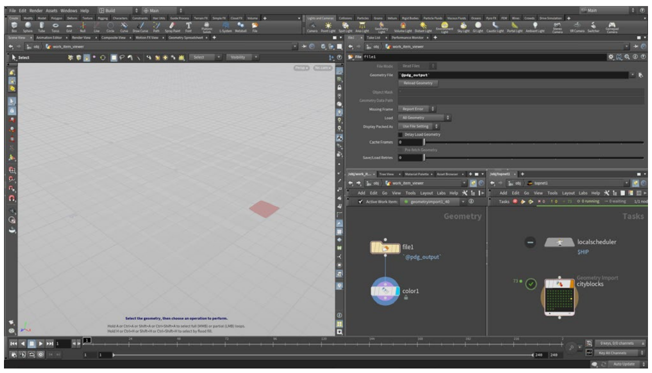
File file1

File Mode: Read Files
Geometry File: `@pdg_output`
Reload Geometry
Object Mask
Geometry Data Path
Missing Frame: Report Error
Load: All Geometry
Display Packed As: Use File Setting
Delay Load Geometry
Cache Frames: 0
Pre-fetch Geometry
Save/Load Retries: 0

06 Instead of linking the display of the city blocks to the frame number, link it directly to the top network. Click on the **File** node and change **Geometry File** to the following expression:

```
`@pdg_output`
```

This means that instead of loading the files from disk in sequence, you will load whichever work item is active in the topnet. At first, you get an error because there arent any work items being output from the PDG network.

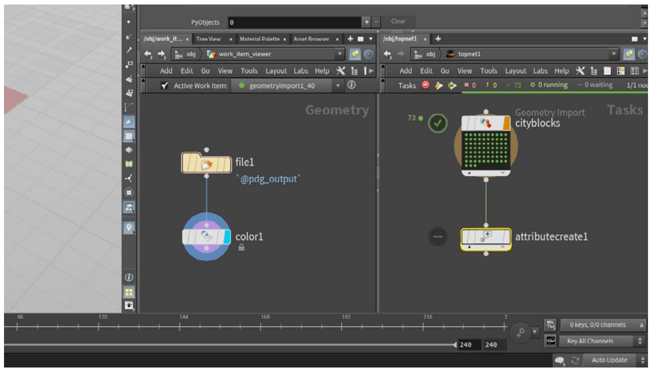


07 From the **Active Work Item** menu in the Geometry Network view, select different work items to select them in the Scene view. The red color moves to the selected geometry. The geometry related to that work item will appear in the viewport. In the topnet, you will see that dot on the *cityblocks* TOP node highlighted in yellow.

Save your work.

PART THREE: Add Attributes

To create buildings, you want to set a fixed base height and a random height variation so that your buildings are not all the same size. You could set up these attributes at the geometry level of the city map but you can also assign them here in TOPs. This will make it easier to make changes at the TOP level if needed down the line.



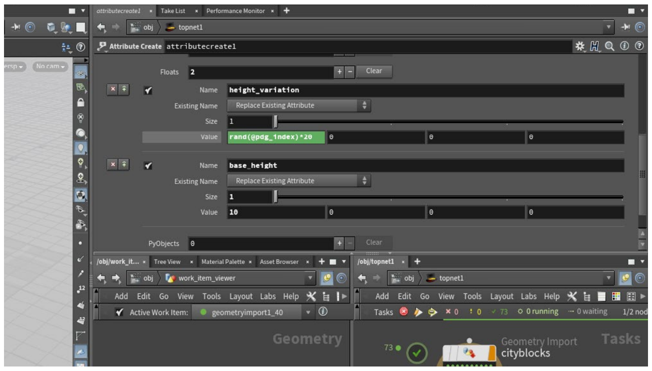
01

In the TOP net, add an **Attribute Create** TOP. Set **Generate When** to **Each Upstream Item is Cooked**.

Click on the **Plus** sign under **Float Attributes** to create a new attribute then enter the following:

- **Name** to *base_height*
- **Value** to **10**

This will ensure that all the buildings will have a minimum value of 10. You can adjust this number later if you want the lowest buildings to be a different height.



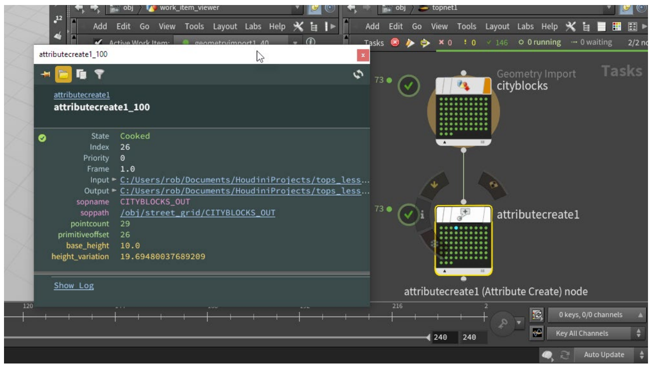
```
Attribute Create  attributecreate1

Floats  2
  Name           height_variation
  Existing Name  Replace Existing Attribute
  Size           1
  Value          rand(@pdg_index)*20   0   0   0

  Name           base_height
  Existing Name  Replace Existing Attribute
  Size           1
  Value          10   0   0   0

PyObjects  0
```

02

Click on the **Plus** sign again and enter the following:

- Name to *height_variation*
- **Value** to `rand(@pdg_index)*20`

This will create a random number from 0 to 20 using the work item attribute **Index** as the seed. When you build the network for creating a building, you will add this value to the base height value to determine how tall each building will be.



```
attributecreate1
attributecreate1_100

          State  Cooked
          Index  26
       Priority  0
          Frame  1.0
        Input ►  C:/Users/rob/Documents/HoudiniProjects/tops_less...
       Output ►  C:/Users/rob/Documents/HoudiniProjects/tops_less...
        sopname  CITYBLOCKS_OUT
        soppath  /obj/street_grid/CITYBLOCKS_OUT
     pointcount  29
primitiveoffset  26
    base_height  10.0
height_variation 19.69480037689209

Show Log
```

03

Select the *attributecreate* node and press **Shift-G** to cook it. In the 3D View, you will still see footprints because you haven't created buildings yet.

If you **MMB-click** on the work items, you can see that each one has a *base_height* of **10** and a *height_variation* that is a number somewhere between **0 and 20**. The attributes have been generated in this TOP node but they aren't being used yet.

ADDING ATTRIBUTES IN TOPS

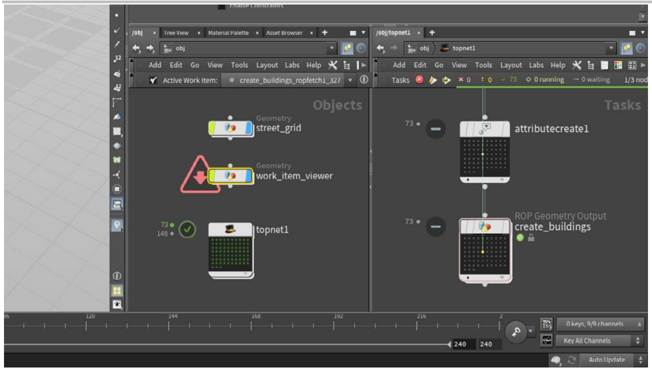
It would have been possible to add attributes in the geometry network of the city grid but adding them in TOPs makes it easier to make changes to them within the context of TOPs.

Attributes are an important way of feeding important information through your pipeline and setting them up properly is important.

```
     pointcount   29
primitiveoffset   26
    base_height   10.0
height_variation  19.69480037689209
```

PART FOUR: Create Buildings for the City Grid

You are now going to use a ROP Geometry node to create a simple building. To see what you are doing in the viewport, you need to generate the work items and select one of them. You can then use that work item to design the building at the geometry level.

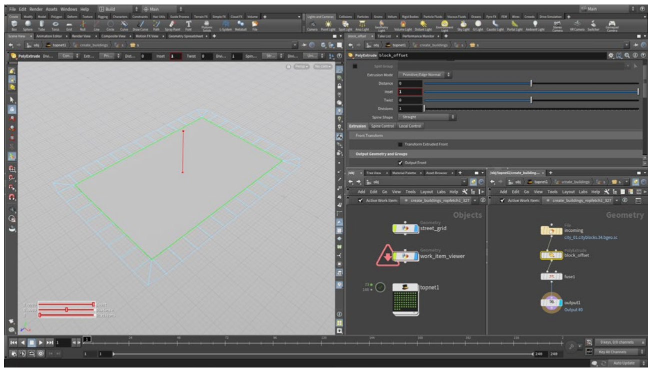


01

Add a **ROP Geometry Output** TOP. Rename it *create_buildings*. Turn **OFF** the **Use External SOP** option.

RMB-click on it and choose **Generate node** to create the work items. These are grey work items that have not been processed yet. In effect they are place holders for the tasks that will take place once you set up this node.

IMPORTANT: Click on one of the work items. This will produce an error in the *work_item_viewer* network because this node isn't producing any geometry yet.

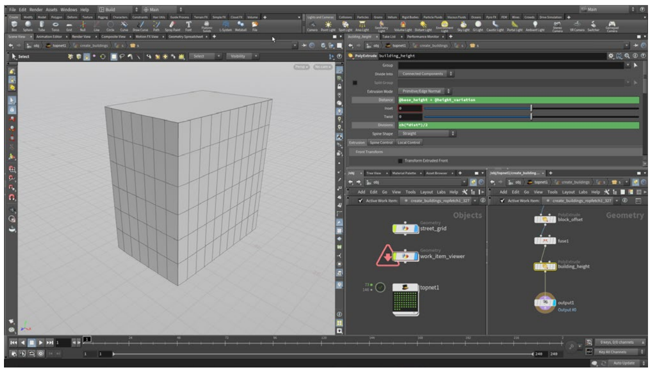


02

Double-click on this node to dive to the geometry level.

Create a **PolyExtrude** node and place it between the *incoming* and *output* nodes. Rename it *block_offset*. Set the **Inset** to 1 and under **Extrusion**, turn **OFF** the **Output Side** option. This will inset the buildings to allow for sidewalks.

Next, add a **Fuse** node after the *polyextrude* node and set **Snap Distance** to **0.2** to remove any lines that are small. Note that you are seeing something in the viewport even though the *work_item_viewer* nodes are showing errors. You will fix this later.



03

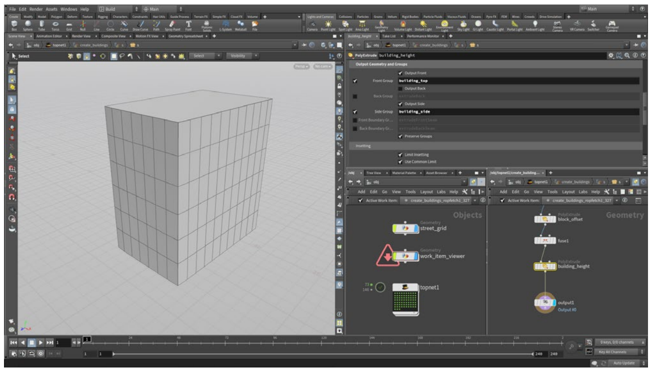
Create a second **PolyExtrude** node after the *fuse*. Rename it to *building_height* and set the **Distance** to:

```
@base_height + @height_variation
```

Make sure the **Distance** parameter is highlighted. **RMB-click** on the parameter and choose **Copy Parameter**. **RMB-click** on the **Divisions** parameter and choose **Paste Relative References**. Edit the resulting channel reference by dividing it by 2:

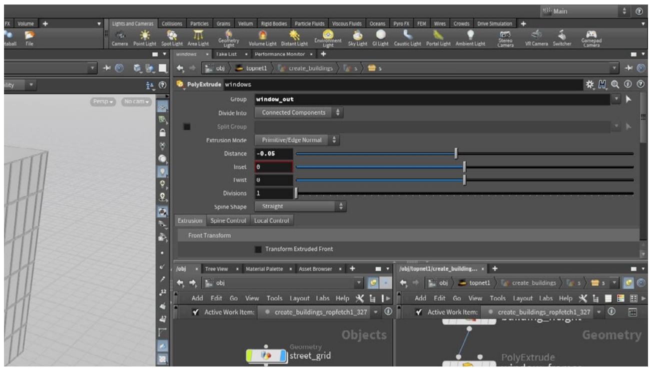
```
ch("dist")/2
```

Now you can see a grid of floors and windows.



04

In the **Extrusion** tab, turn **ON** the **Front Group** and name it *building_top* and then turn **ON** the **Side Group** and name it *building_side*. You will use these to add detail to the building.

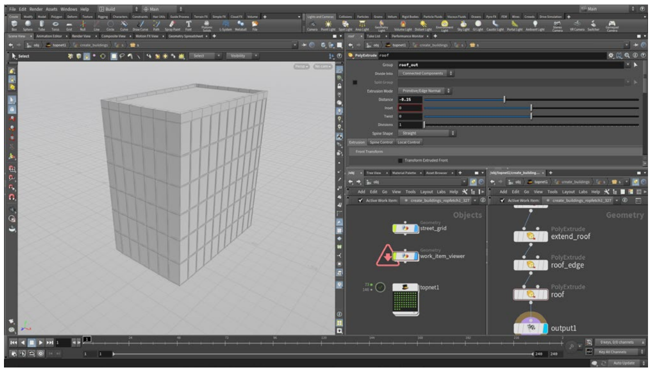


05

Add a new **Poly Extrude** node into the chain and rename it *window_frames*. Set the following:

- **Group** to *building_side*
- **Inset** to 0.05
- **Divide Into** to **Individual Elements**
- In the **Extrusion** tab, turn **ON** the **Front Group** and name it *window_out*.

Add another new **Poly Extrude** node and rename it *windows*. Set **Group** to *window_out* and **Distance** to **-0.05**.



06

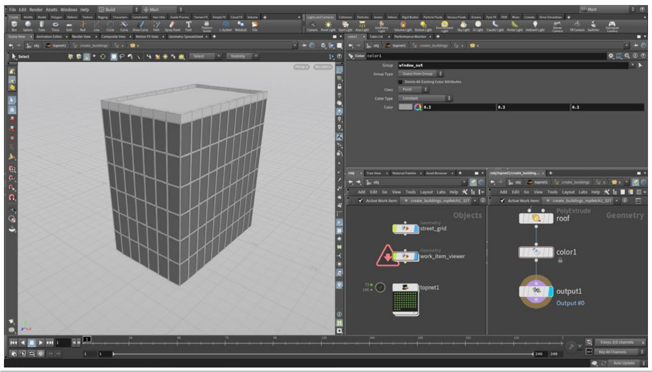
Add a new **Poly Extrude** and rename it *extend_roof*. Set

- **Group** to *building_top*
- Distance to **0.5**

Add another **Poly Extrude** node and rename it *roof_edge*. Set

- **Group** to *building_top*
- Inset to **0.3**
- In the **Extrusion** tab, turn **ON** the **Front Group** and name it *roof_out*.

Add another new **Poly Extrude** node into the chain and rename it *roof*. Set **Group** to *roof_out* and **Distance** to **-0.25**.

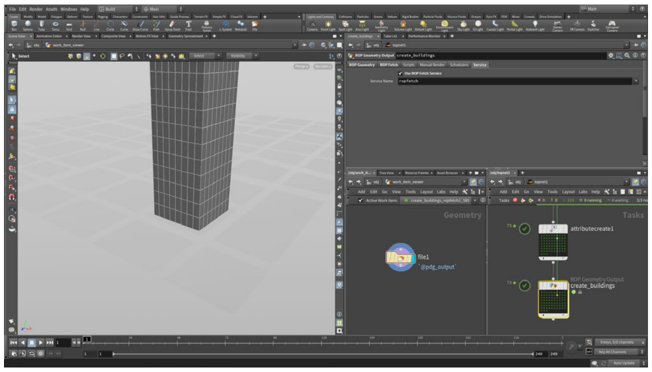


07

Add a color node after this and set the following:

- **Group** to *windows_out*
- **Color** to a **dark grey (0.2, 0.2, 0.2)**.

Make sure that these nodes are wired into the *output* node and set the **Display flag** on that node. Now the frames will be a light color and the windows will be darker. This will help you read them in the viewport.



08

Go back up one level and on the *create_buildings* TOP change `$F` to `` `@pdg_index` `` to set the **Output File** to:

```
$HIP/geo/$HIPNAME.$OS.`@pdg_index`.bgeo.sc
```

This will use the index file of each building instead of the frame. Click on the **ROP Fetch** tab and set the **Cache Mode** to **Write Files.**

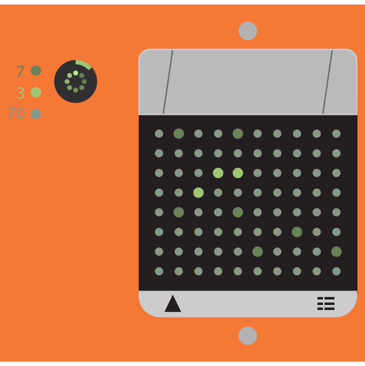
Save your scene file then select the *create_buildings* TOP and press **Shift-G** to cook it.

Go back inside the *work_item_viewer* and remove the color node so you no longer see red. Now on the *create_buildings* TOP node, click on the work items to see the buildings with different heights.

WORK ITEM PROGRESS

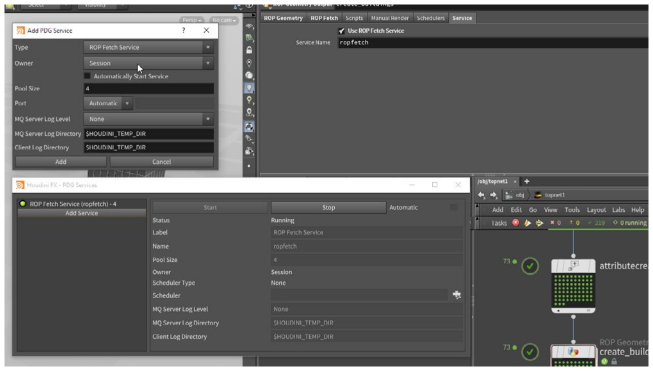
Up until now the work items have processed very quickly but this node takes a little longer. You can see the progress wheel which shows the completed tasks in dark green, the in-progress tasks in yellow and the queued work items in grey. This lets you observe the node as it completes the assigned tasks.

At the Object level, the TOP Network also displays work item progress for all the tasks in the network.



PART FIVE: Combine the Buildings

Once the buildings have been completed, you will want to bring them back together to create the city. This involves a partition node called Wait for All and a Geometry Import. You will also add in a city core where the buildings will be taller than elsewhere in the city.



Add PDG Service

Type	ROP Fetch Service
Owner	Session
	Automatically Start Service
Pool Size	4
Port	Automatic
MQ Server Log Level	None
MQ Server Log Directory	$HOUDINI_TEMP_DIR
Client Log Directory	$HOUDINI_TEMP_DIR

Service: ✓ Use ROP Fetch Service — Service Name: ropfetch

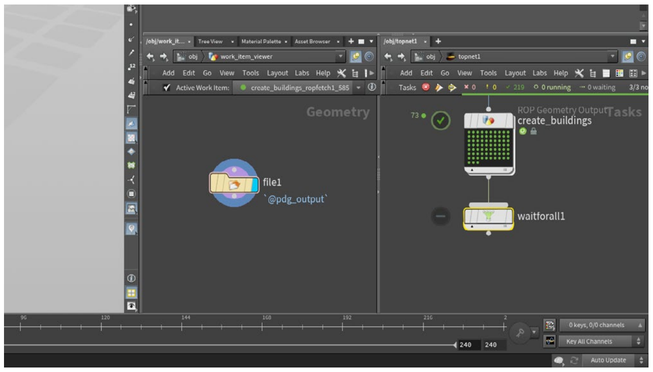
Houdini FX - PDG Services: ROP Fetch Service (ropfetch) - 4

Status	Running
Label	ROP Fetch Service
Name	ropfetch
Pool Size	4
Owner	Session
Scheduler Type	None
Scheduler	
MQ Server Log Level	None
MQ Server Log Directory	$HOUDINI_TEMP_DIR
Client Log Directory	$HOUDINI_TEMP_DIR

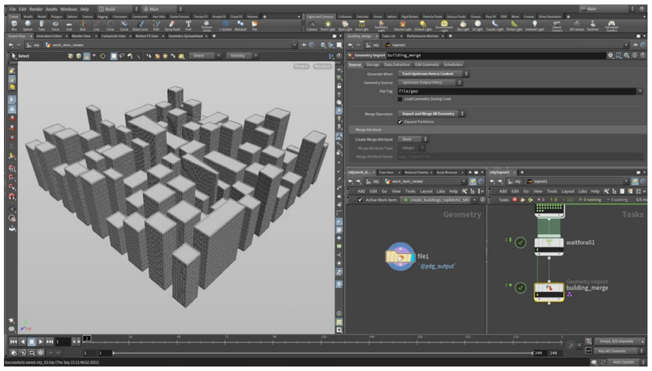
01 To speed up the processing of the tasks, you can use PDG Services. In the TOP Network view select **Tasks > PDG Services**. Click **Add Service. Set Type** to **ROP Fetch Service** and **Pool Size** to **4**. Click **Add** then click **Start.**

Close the Services window then go to the **Service** tab of the *create_buildings* TOP and set **Enable ROP Fetch Service** to **On**. This will speed up the processing of this TOP node.

Save your scene file then select the *create_buildings* TOP and press **Shift-D** to dirty it then **Shift-G** to cook it. You will see that it is much faster than it was the last time.

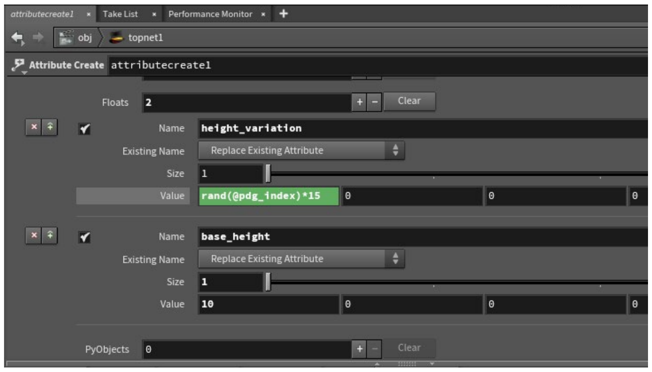


02 Add a **Wait for All** TOP node. This will just take all the work items that the ROP Geometry node generates and combine them into a single work item. It will also prevent any descendent tops from cooking until all the work items above have completed cooking.



03 Add a **Geometry Import** TOP. Name it *building_merge*. Set **Generate When** to **Each Upstream Item is Cooked** and set **Merge Operation** to **Import and Merge All Geometry**.

Save your scene file then select the *building_merge* node and press **Shift-G** to cook it. When the network finished cooking, click on the work item on the final *building_merge* node. You will see the whole city in the viewport.



Attribute Create attributecreate1

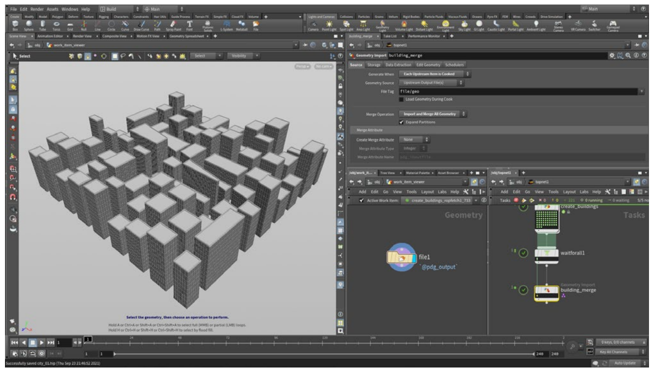
Floats	2			
Name	height_variation			
Existing Name	Replace Existing Attribute			
Size	1			
Value	rand(@pdg_index)*15	0	0	0
Name	base_height			
Existing Name	Replace Existing Attribute			
Size	1			
Value	10	0	0	0
PyObjects	0			

04 Go back to the **Attribute Create TOP** and edit the *height_variation* attribute to the following:

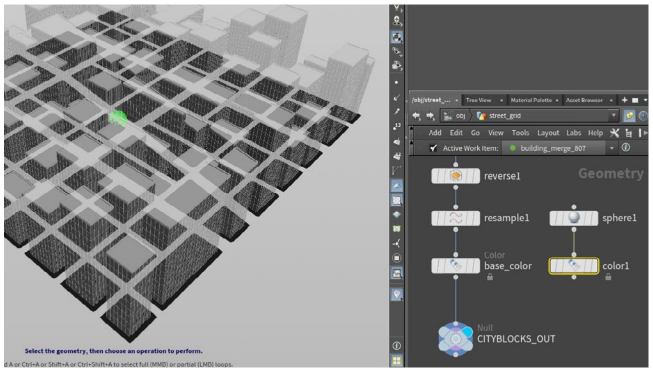
- **Value** to `rand(@pdg_index) * 15`

This will make the buildings a bit shorter overall.

RMB-click on the *attributecreate* TOP and choose **Dirty This Node**. This will clear out all the work items going down the chain.



05 **Save** your scene file. Now select the *building_merge* TOP node and press **Shift-G** to cook it. When the network finishes cooking, click on the work item on the final *building_merge* node to see the change.

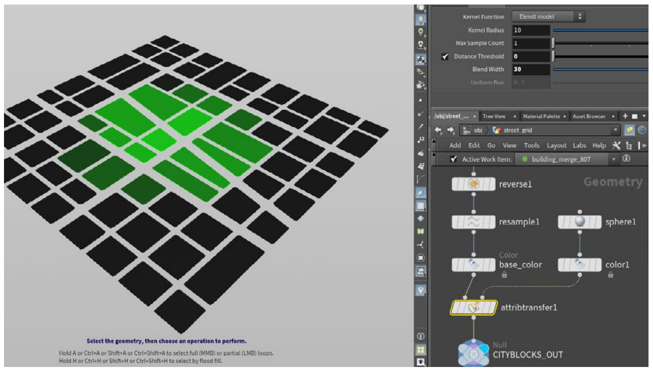


06 Navigate to the *street_grid* object and dive into it. Create a **color** node and wire it into the chain after the *resample* node. Set the **Class** to **Primitive** and the **Color** to **black (0, 0, 0)**. Rename this node *base_color*.

Create a sphere in the *street_grid* network and place it off to the side and set:

- **Primitive Type** to **Polygon**
- **Uniform Scale** to **5**

Add a Color node to the output of the *sphere* and set **Class** to **Primitive** and **Color** to **green (0, 1, 0)**. Name this node *transfer_color*.

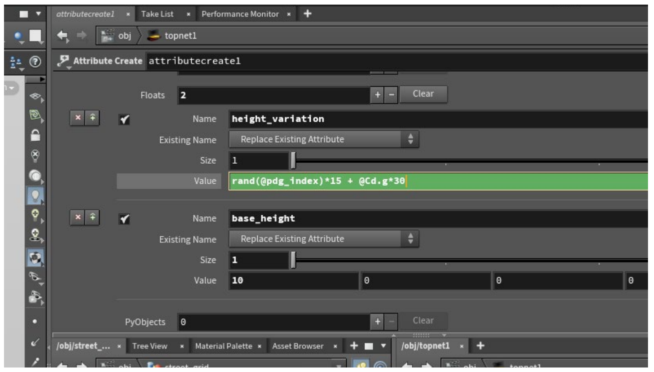

Kernel Function: Elendt model
Kernel Radius: 10
Max Sample Count: 1
✓ Distance Threshold: 0
Blend Width: 30
Uniform Bias: 0.5

07 Add an **Attribute Transfer** node after the *base_color* node. Then connect the *transfer_color* node into its second input.

Uncheck the **Points** checkbox, and set the **Primitives** field to **Cd**. Then in the **Conditions** tab, set:

- **Distance Threshold** down to **0**
- **Blend Width** to about **30**.

With your cursor over the Scene view, press **Spacebar-Y** to google to **Hide Other Objects**. You can now see the green color gradually permeate the tiles as they approach the center of the sphere.


Attribute Create attributecreate1

Floats: 2

Name: height_variation
Existing Name: Replace Existing Attribute
Size: 1
Value: rand(@pdg_index)*15 + @Cd.g*30

Name: base_height
Existing Name: Replace Existing Attribute
Size: 1
Value: 10 0 0 0

PyObjects: 0

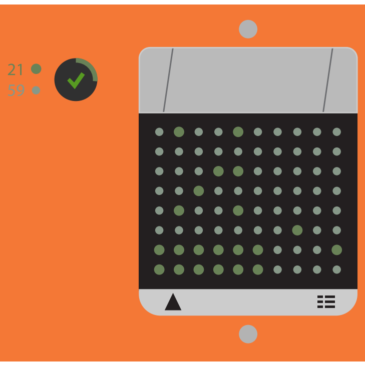
08 Go back to the TOP network and select the *attributecreate* node and edit the *height_variation* attribute to the following:

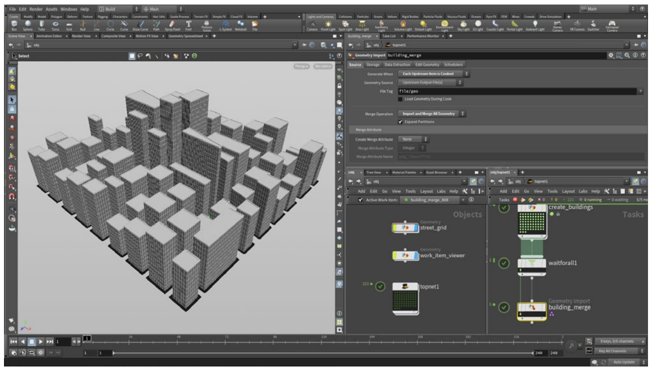
- **Value** to `rand(@pdg_index) * 15 + @Cd.g*30`

This will add height to those buildings in the city core. The value of 30 will be the maximum height added. You can change this value if you want taller or shorter buildings.

DIRTY & CLEAN WORK ITEMS

One of the main purposes of the PDG technology is the dependencies that arise in a complex system. As you make changes to the downtown core, you will see that some nodes become dirty and have to be recooked while others are fine the way they are. The way TOPs handles these dependencies is one of its strengths.



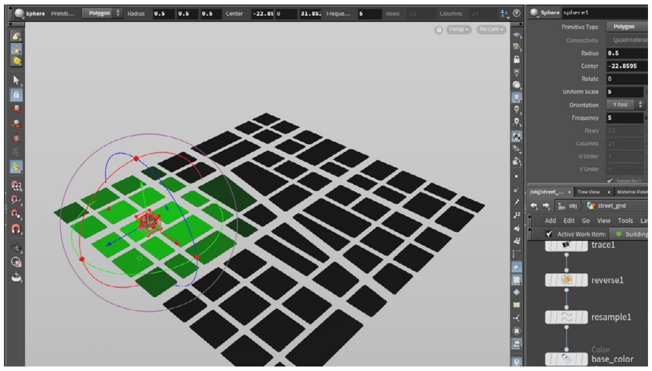


09

RMB-click on the *building_merge* node and from the menu select **Dirty This Node**. You could also select it and hit **Shift-D** to dirty it.

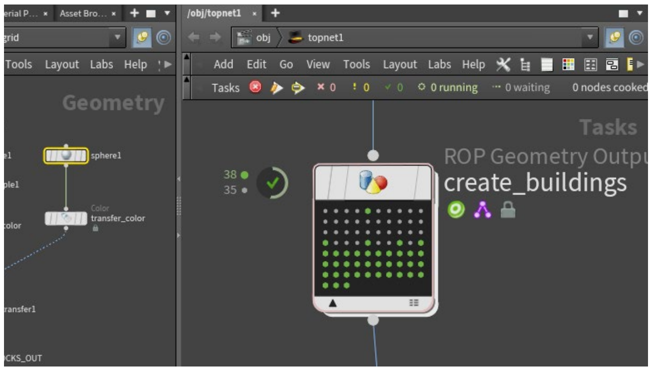
Select the *building_merge* node and press **Shift-G** to cook it. You will be prompted to Save. When it is ready, click on the work item on the final *building_merge* node.

In the other Network view, navigate back up to the object level. You will see the city updated with the city core in the viewport.



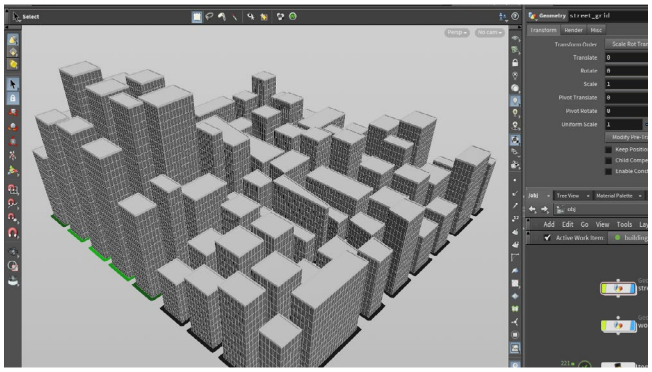
10

Navigate to the *street_grid* object and with the *CITYBLOCK_OUT* node displayed, you can select the *sphere* node and use the **handle** tool to move it around to see it affect a different part of the city grid.



11

Go back to the TOP net and **RMB-click** on the *create_buildings* node and select **Generate Node.** You will see that 35 of the work items have been "dirtied" automatically because of the change at the geometry level. The other 38 are still considered clean and will not need to be recooked.



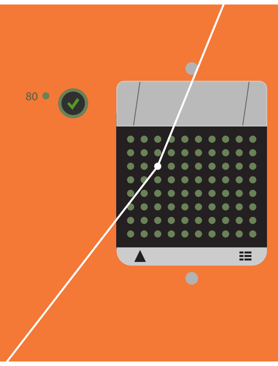
12

Now when you **RMB-click** on the *building_merge* node and select **Cook Selected Node**, only the dirty work items will be updated. Click on the work item to see the new city core. This is one of the ways that the dependency graph in TOPs works efficiently as you develop your workflow.

UPDATED DEPENDENCIES

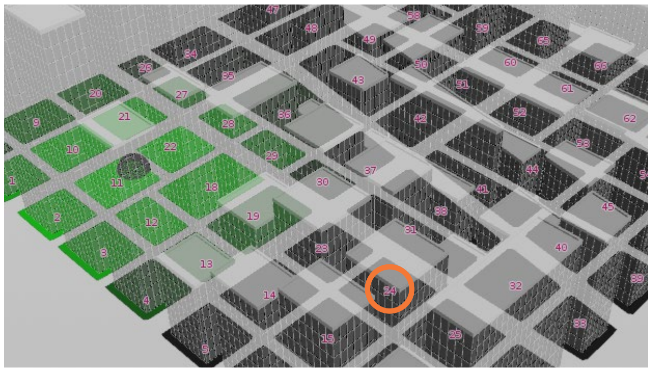
When you select on a work item, you get lines that show how that work item is connected to other work items in the system.

This is a mapping of the dependencies and this helps your graph work efficiently because only dirty work items will be processed unless you explicitly dirty all the work items on a node with **Shift-D**.



PART SIX: Focus on One Building

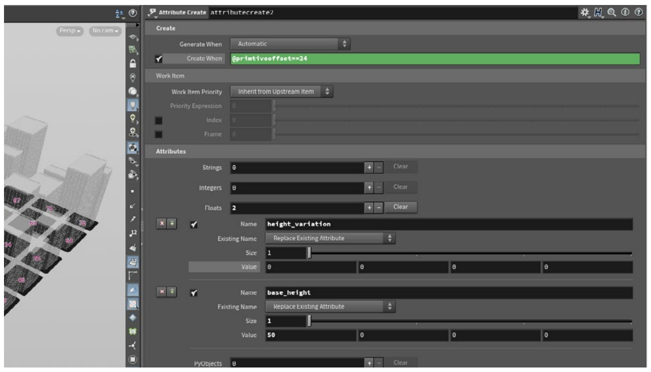
Up until now the building heights are determined by the randomness of the height_variation. If there is one building that you want to fix to a certain height, a second attribute create node can be used to choose a primitive and set specific values. This makes it easier to art direct a system when you want to override the randomness.



01

Press **Spacebar-Y** to google to **Ghost Other Objects**. Go back to the *street_grid* object and turn on **Display Primitive Numbers** in the **Display Options**. You can see the block numbers. You can identify the buildings and decide if you want to set one of them explicitly. Here **block 24** is the chosen block.

You are now going to set specific parameters for this city block to get the exact height you want instead of relying on the randomness of our current setup. This will give you some artistic control within the TOP network instead of relying on the randomness to determine everything.



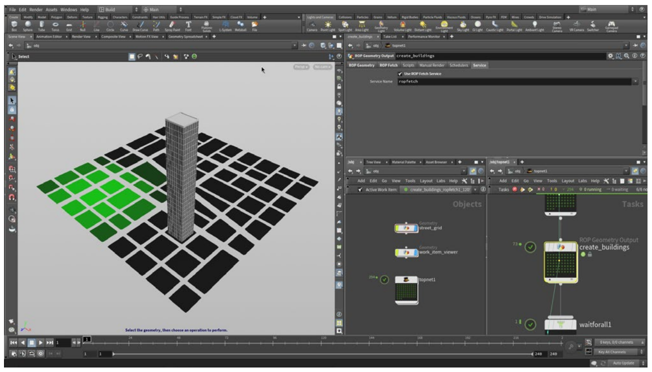
```
Attribute Create  attributecreate2
Create
    Generate When        Automatic
  ✓ Create When          @primtiveoffset==24
Work Item
    Work Item Priority   Inherit from Upstream Item
    Priority Expression  0
    Index                0
    Frame                0
Attributes
    Strings    0
    Integers   0
    Floats     2
  ✓ Name           height_variation
    Existing Name  Replace Existing Attribute
    Size           1
    Value          0    0    0    0
  ✓ Name           base_height
    Existing Name  Replace Existing Attribute
    Size           1
    Value          50   0    0    0
    PyObjects  0
```

02

Now **Alt-click**-drag on the *attributecreate* TOP node to make a copy of it. Don't wire it in yet. Turn on the checkbox next to **Create When** and add the following expression:

@primitiveoffset==24

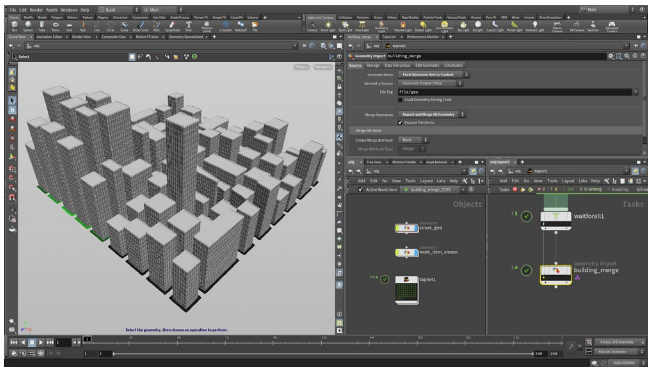
RMB-click on *height_variation* **value** parameter and choose **Delete Channels** then set its value to **0** and then set *base_height* value to **50**. This will explicity make the building on **block 24** exactly 50 units high.



03

Insert this node in between the first *attributecreate* and the *create_buildings* node. **Save** your work then recook the *building_merge* node.

Click on the work item at index 24 on the *create_buildings* node to see the building on block 24 at 50 units.



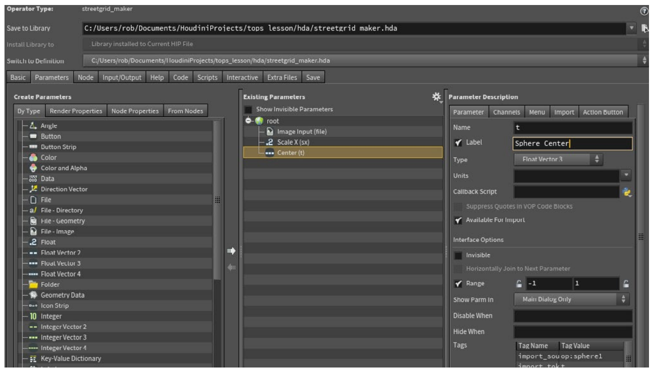
04

Click on the work item on the *building_merge* node to see the new cityscape with the new building rising up. Now anytime you recook the network, the building on block 24 will not update but will remain set specifically.

If you want to change that building, you can use the second attribute create node to explicitly set its height.

PART SEVEN: Wedge the City Core Location

In order to wedge the position of the sphere within the street_grid object, the network will need to be turned into a Houdini Digital Asset [HDA] and run through an HDA Processor TOP node. The ability to load these assets and add them to the system lets you create a complex system out of discreet tools that can be updated and adapted as needed.



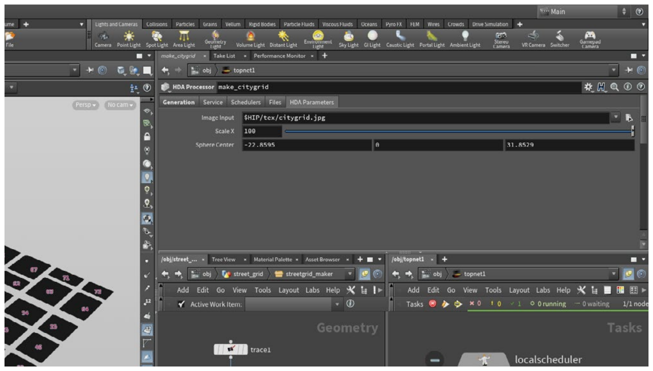
Operator Type: streetgrid_maker
Save to Library: C:/Users/rob/Documents/HoudiniProjects/tops lesson/hda/streetgrid maker.hda
Install Library to: Library installed to Current HIP File
Switch to Definition: C:/Users/rob/Documents/HoudiniProjects/tops_lesson/hda/streetgrid_maker.hda

Existing Parameters: root, Image Input (file), Scale X (sx), Center (t)

Name: t
Label: Sphere Center
Type: Float Vector 3
Available For Import: ✓
Range: ✓ -1 1
Show Parm In: Main Dialog Only
Tags: import_sou op:sphere1

01 In the *street_grid* network, select all the nodes except *CITYBLOCKS_OUT*. From the **Assets** menu, choose **New Digital Asset from Selection.** Name the asset *streetgrid_maker*, label it *Street Grid Maker*, and save it to the `$HIP/hda/` directory.

In the **Operator Type Properties** panel, click on the **Parameters** tab. Dive into the asset. Drag **Image Input** from the *trace* node to the parameter list. Drag **Scale X** to **Scale Y** and choose **Relative Channel Reference**. Now drag **Scale X** to the parameter list and name it *Scale*. Drag **Center** from the *sphere* node and name it *Sphere Center*. Drag **Blend Width** from the *attributetransfer* node then click **Accept**.

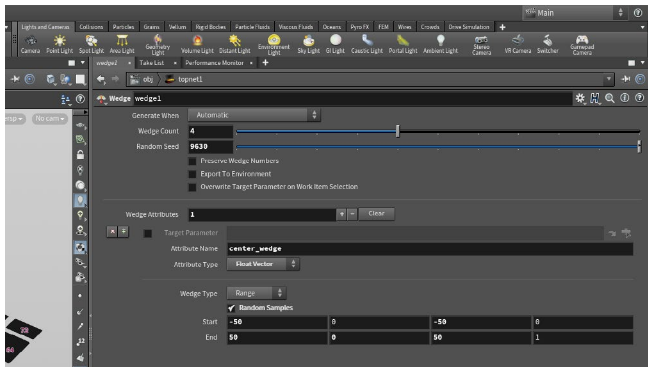


HDA Processor: make_citygrid
Image Input: $HIP/tex/citygrid.jpg
Scale X: 100
Sphere Center: -22.8595 0 31.8529

02 Go back to the TOP network. Create an **HDA Processor TOP** and wire it into the **Geometry Import** node. On the *geometryimport* node, change **Generate When** to **Each Upstream Item is Cooked** and **Geometry Source** to **Upstream Output File**.

Select the **HDA Processor** node and set **HDA File** to *$HIP/hda/streetgrid_maker.hda*. Click on the **HDA Parameters** tab to see the **Image Input**, **Center, Scale** and **Blend Width** parameters listed.

Rename this TOP *make_citygrid* then press **shift-V** to *Dirty and Cook* the node to make sure that it is still generating the city grid the way it was before.



Wedge: wedge1
Generate When: Automatic
Wedge Count: 4
Random Seed: 9630
Preserve Wedge Numbers
Export To Environment
Overwrite Target Parameter on Work Item Selection
Wedge Attributes: 1
Target Parameter
Attribute Name: center_wedge
Attribute Type: Float Vector
Wedge Type: Range
✓ Random Samples
Start: -50 0 -50 0
End: 50 0 50 1

03 Create a **Wedge** TOP node and wire it into the *make_citygrid*. Set the following:

- **Wedge Count** to **4**

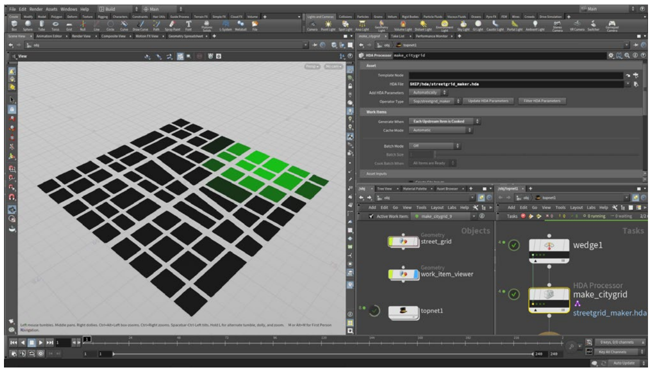
Click the **plus sign** next to **Wedge Attributes** and set the **Attribute Name** of the first one to *center_wedge* then set:

- **Type** to **Float Vector**,
- **Start range** to **-50, 0, -50, 0**
- **End range** to **50, 0, 50, 0**
- Turn **ON** the **Random Samples**

WEDGING

Wedging is an idea that comes from photoography where a shot is taken using a variety of settings so that the best one can be chosen later in the lab. Here the idea is the same except you will use random values to influence the system and get four unique results that can be compared.

Later you will render out a mosaic wiith all the options labelled with the random values displayed.

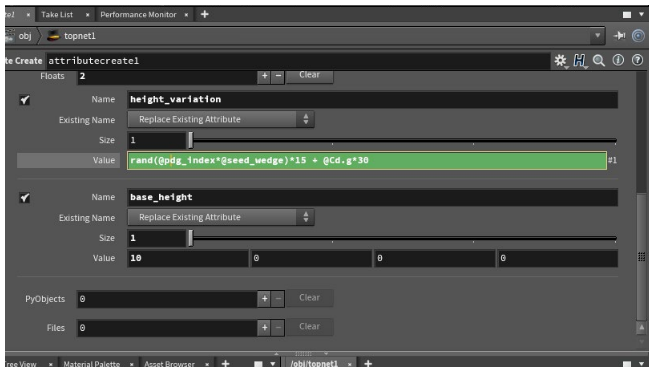


04 Select the **HDA Processor** node and on the **HDA Parameters** tab, set the **Center** parameter to:

```
@center_wedge.0, @center_wedge.1, @center_wedge.2
```

Select the *make_citygrid* HDA processor node and press **shift-V** to *Dirty and Cook* the node - now there are four citygrids being created.

Hide the **street_grid** object. Click on the work items to visualize the different grids. Each of them show a different position for the green color which is where the city core is located.



attributecreate1

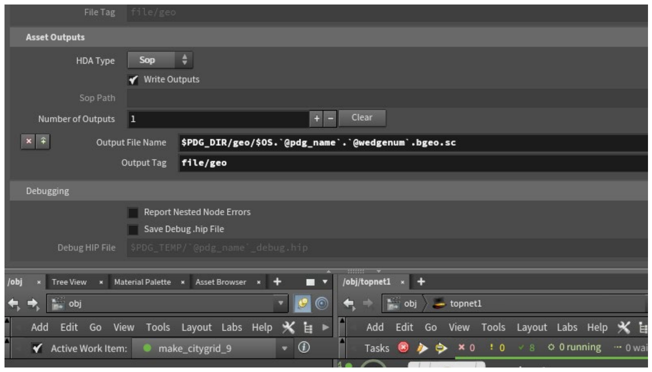
Floats	2
Name	height_variation
Existing Name	Replace Existing Attribute
Size	1
Value	rand(@pdg_index*@seed_wedge)*15 + @Cd.g*30
Name	base_height
Existing Name	Replace Existing Attribute
Size	1
Value	10 0 0 0
PyObjects	0
Files	0

05 Now add some wedging to the height variation so that you get four different looks. Select the *wedge* TOP and click the **plus sign** next to **Wedge Attributes** and set the **Attribute Name** of the second one to *seed_wedge*, leave **Type** set to **Float**, and set its **Start/End** to **0, 1000**.

Turn **ON** the **Random Samples** parameter.

On the *first attributecreate* TOP, change the *height_variation* attribute's **Value** to:

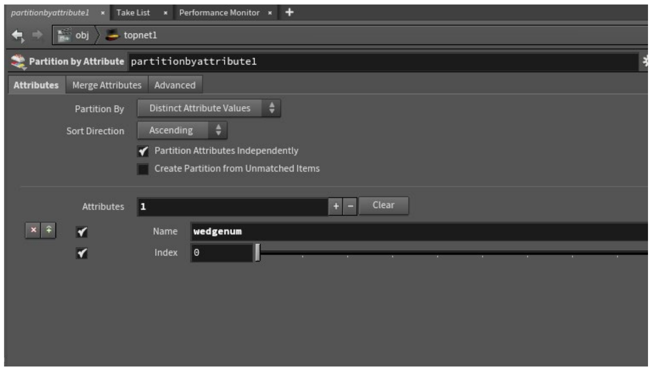
```
rand(@pdg_index*@seed_wedge) * 15 + @Cd.g*30
```



File Tag	file/geo
Asset Outputs	
HDA Type	Sop
Write Outputs	✓
Sop Path	
Number of Outputs	1
Output File Name	$PDG_DIR/geo/$OS.`@pdg_name`.`@wedgenum`.bgeo.sc
Output Tag	file/geo
Debugging	
Report Nested Node Errors	
Save Debug .hip File	
Debug HIP File	$PDG_TEMP/`@pdg_name`_debug.hip
Active Work Item:	make_citygrid_9

06 You will have to update the output file names for all the TOPS that create an output file to include the wedgenumber, so that the files can be differentiated for the different wedges. Include a `.`@wedgenum`` in the output file parameters for the following TOP nodes just before `.bgeo`:

- *make_citygrid* HDA Processor
- *geometryimport* Geometry import
- *create_buildings* ROP Geometry
- *building_merge* Geometry Import

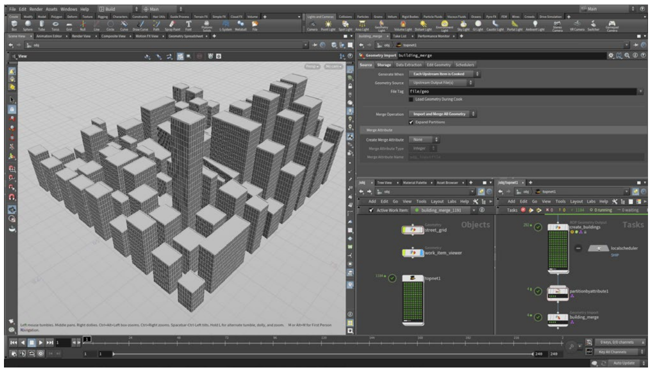


Partition by Attribute partitionbyattribute1

Partition By	Distinct Attribute Values
Sort Direction	Ascending
Partition Attributes Independently	✓
Create Partition from Unmatched Items	
Attributes	1
Name	wedgenum
Index	0

07 Replace the **Wait for All** node with a **Partition by Attribute** TOP node. Make sure that **Partition By** is set to **Distinct Attribute Values**.

Click the **plus sign** next to **Attributes** and set the **Name** to *wedgenum*. This will create a partition for each wedge.

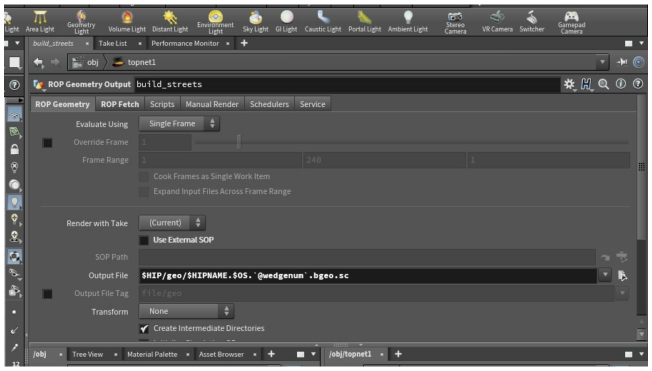


08 **Save** your work then cook the *building_merge* TOP node. This will create the four city maps which you can visualize by clicking on the work items on the *building_merge* node.

Since this is using the local scheduler this will take a little while longer to process. This is where using a scheduler that distributes your tasks to a compute farm can speed things up considerably.

PART EIGHT:
Create Geometry for the Streets

To create context for the buildings, you will now create city blocks and streets to use in the final rendering. While the four wedges currently all use the same map, you will set up this geometry in TOPs to allow for the use of different maps down the line. It is always a good idea to build in flexibility to create the most robust system.



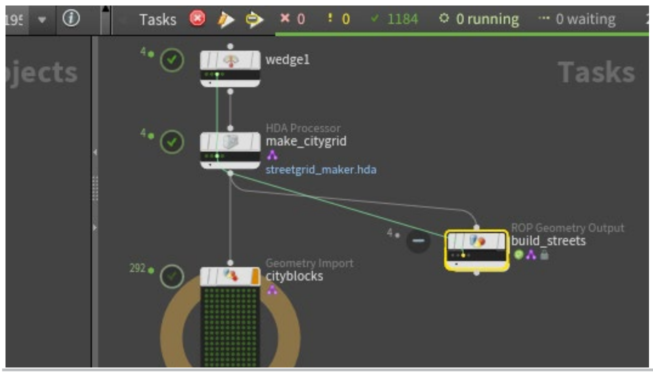
ROP Geometry Output build_streets

Evaluate Using: Single Frame
Override Frame: 1
Frame Range: 1 240 1
Cook Frames as Single Work Item
Expand Input Files Across Frame Range
Render with Take: (Current)
Use External SOP
SOP Path
Output File: `$HIP/geo/$HIPNAME.$OS.`@wedgenum`.bgeo.sc`
Output File Tag: file/geo
Transform: None
✓ Create Intermediate Directories

01 In the TOP network, add a **ROP Geometry Output** node and branch it off from the *make_citygrid* HDA processor. Rename this new node *build_streets*.

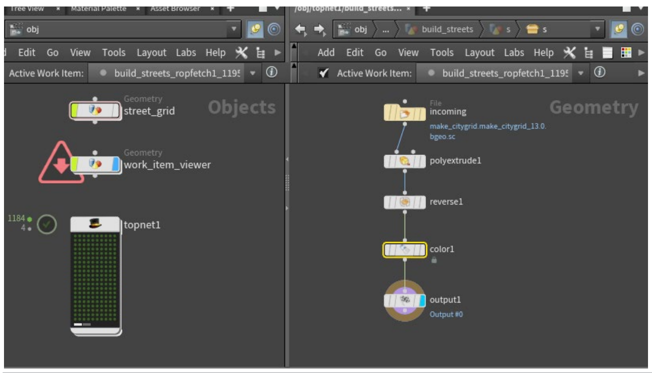
Turn **OFF** the **Use External SOP** option and in the **Output File** add `.`@wedgenum`` to the expression in place of $F just before `.bgeo`.

On the **ROP Fetch** tab, set **Cache Mode** to **Write Files**.



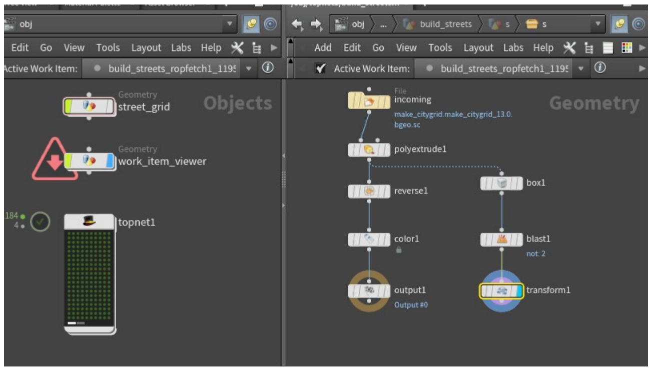
02 **RMB-click** on this node and choose **Generate Node** to get the four work items. Click on one of them to highlight the work item and then double click on the node to dive into it.

This will bring you to the geometry level where you will create the streets.



03 Add a **PolyExtrude** node between *incoming* and *output* nodes. Set **Distance** to **-0.05**. Turn **OFF** the **Output Front** checkbox and turn **ON** the **Output Back** option.

Add a **Reverse** node after to fix the normals and then add a **Color** node to turn all the city blocks white.

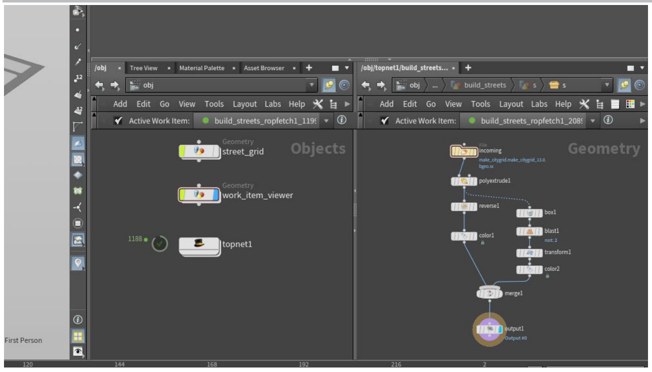


04 Add a **Box** node into the network off to the side. Wire *polyextrude* into the *box* node to match the bounding box to the city grid geometry.

Add a **Blast** node after the *box* and set **Group** to **2** and turn **ON** the **Delete Non Selected** option.

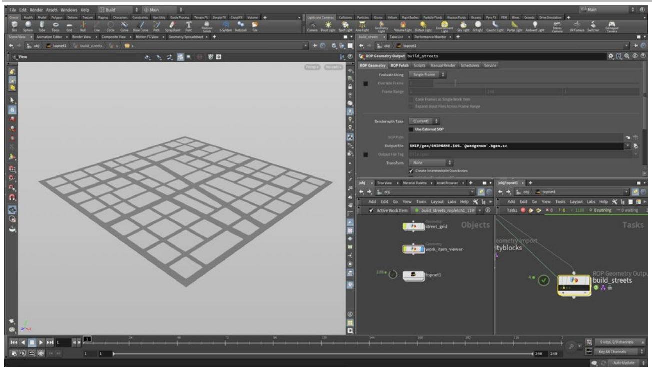
Add a **Transform** node after the *blast* and set:

- **Translate Y** to **-0.05**
- **Scale X** to **1.05**
- **Scale Z** to **1.05**



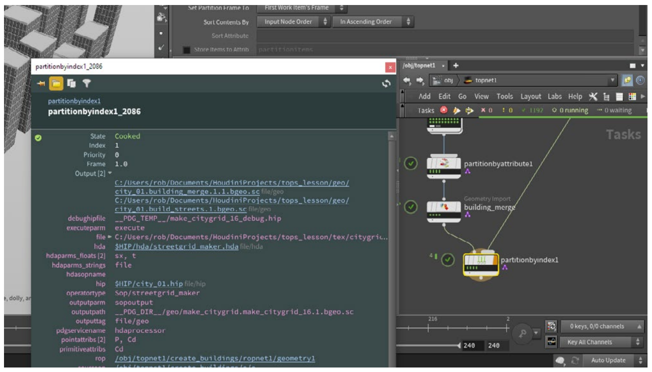
05 Add a **Color** node with a medium grey (0.33, 0.33, 0.33).

Now create a Merge node and connect it to the two *color* nodes. Make sure the **display flag** is set to output to see the street geometry in place.



06 **Save** the Scene File. Go back to the TOP Network and Cook the node, you can click on the work items to see the grid.

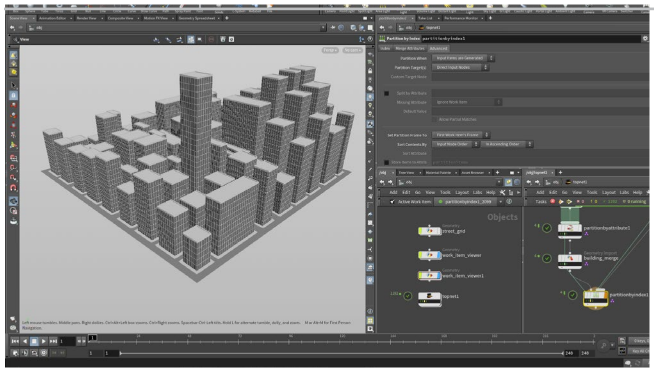
At the moment all four of them are the same. That will be different later when you introduce more city grids into the equation.



```
partitionbyindex1_2086

partitionbyindex1
partitionbyindex1_2086

           State  Cooked
           Index  1
        Priority  0
           Frame  1.0
      Output [2]
                  C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/
                  city_01.building_merge.1.1.bgeo.sc file/geo
                  C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/
                  city_01.build_streets.1.bgeo.sc file/geo
    debughipfile  __PDG_TEMP__/make_citygrid_16_debug.hip
     executeparm  execute
            file  C:/Users/rob/Documents/HoudiniProjects/tops_lesson/tex/citygri...
             hda  $HIP/hda/streetgrid_maker.hda file/hda
hdaparms_floats [2]  sx, t
 hdaparms_strings  file
     hdasopname
             hip  $HIP/city_01.hip file/hip
    operatortype  Sop/streetgrid_maker
      outputparm  sopoutput
      outputpath  __PDG_DIR__/geo/make_citygrid.make_citygrid_16.1.bgeo.sc
       outputtag  file/geo
  pdgservicename  hdaprocessor
 pointattribs [2]  P, Cd
 primitiveattribs  Cd
             rop  /obj/topnet1/create_buildings/ropnet1/geometry1
```

07 Add a **Partition by Index** to the end of the chain. Feed the *building_merge* node into the first input and the **build_streets** into the second. Turn **On** the **Use Dynamic Parenting** option.

Cook the new node. When you are finished, click on the work items and you will see that only the buildings are being displayed. If you **middle click** on a work item you will see that there are two output files being created by the Partition. You need to display the new one to see it in the viewport.



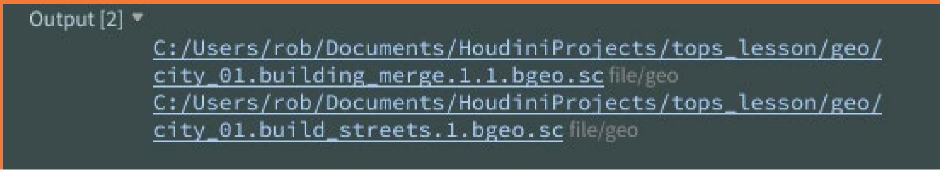
08 Go to the Object level. **Alt drag** on the *work_item_viewer* to create a second one. Dive into the geometry level and set Geometry File to:

`` `@pdg_output.1` ``

Go back up one level to the object level then in the TOP network select one of the work items on the *partitionbyindex* node. Now you will see both of the outputs being displayed in the viewport. This is important for the next step when you will render out images of the city.

ANOTHER PDG OUTPUT

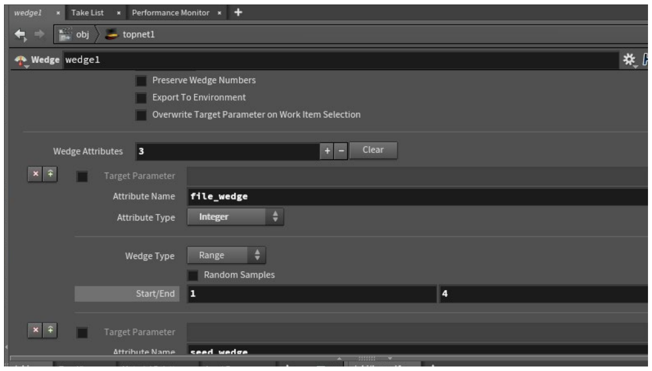
Earlier you used **pdg_output** to create the **work_item_viewer** node to help you visualize what is coming out of each node. Now that you have this partition in place, there are actually two outputs which you can see by **Ctrl-MMB** clicking on a work item. This second output therefore needs its own viewer to be properly displayed.



```
Output [2]
            C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/
            city_01.building_merge.1.1.bgeo.sc file/geo
            C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/
            city_01.build_streets.1.bgeo.sc file/geo
```

PART NINE: Wedge four City Maps

Let's add one extra variable into the wedging. You are going to access four different maps and create one rendering for each one of them. Each of the different black and while images will be traced and used to feed the system. This shows another way to add more content into the pipeline.



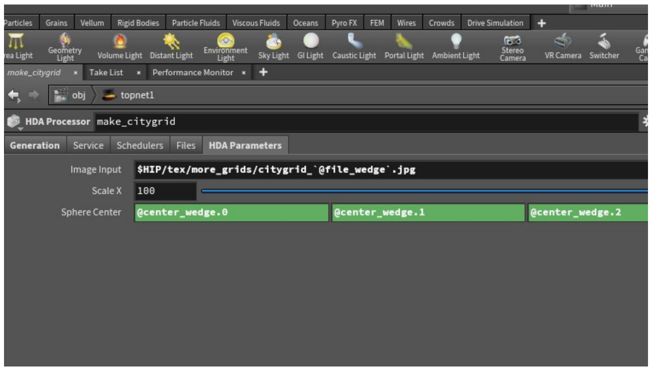
```
Wedge  wedge1
  Preserve Wedge Numbers
  Export To Environment
  Overwrite Target Parameter on Work Item Selection
Wedge Attributes   3
  Target Parameter
  Attribute Name   file_wedge
  Attribute Type   Integer
  Wedge Type       Range
  Random Samples
  Start/End        1        4
  Target Parameter
  Attribute Name   seed_wedge
```

01 Go back to the *wedge* TOP node. Click the **plus sign** next to **Wedge Attributes** to add a new wedge parameter.

Set the following:

- **Attribute Name** to *file_wedge*
- **Attribute Type** to **Integer**
- **Start/End** to **1,4**

This will create integer values of 1, 2, 3 and 4 which will be used to find the different city maps.

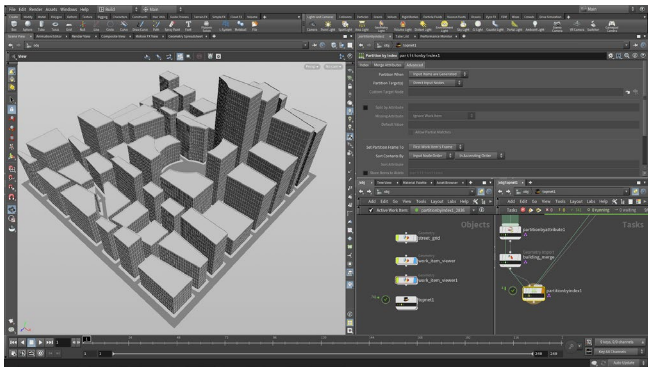


```
HDA Processor  make_citygrid
Image Input    $HIP/tex/more_grids/citygrid_`@file_wedge`.jpg
Scale X        100
Sphere Center  @center_wedge.0   @center_wedge.1   @center_wedge.2
```

02 On the *streetgrid_maker* HDA processor node, click on the **File** button next to **Image Input** and navigate to the *more_grids* folder. Select the file from there then in the Parameter pane, change `$F` in the expression to `` `@file_wedge` ``

There are four files in this folder and the wedge attribute will run through them all in order when you cook the network.

03 Dirty and Cook (**Shift-V**) the *streetgrid_maker* HDA processor node. You will see the four different maps being used and the city core being moved around like before.

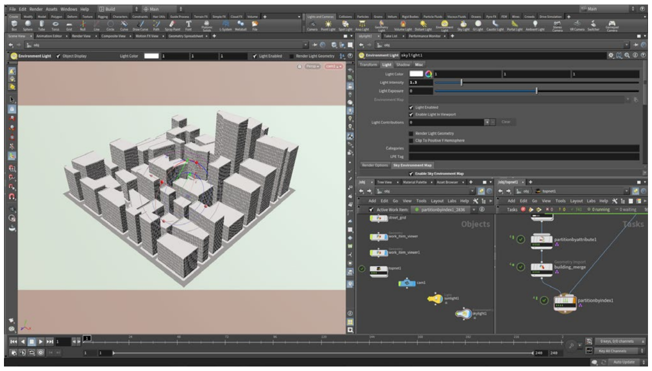


04 **Bypass** the second *attributecreate* TOP node. This was designed for a static map and won't work for four different maps.

Cook the Final partition node (**Shift-G**) and view the four city plans turned into cities.

PART TEN:
Render a Mosaic

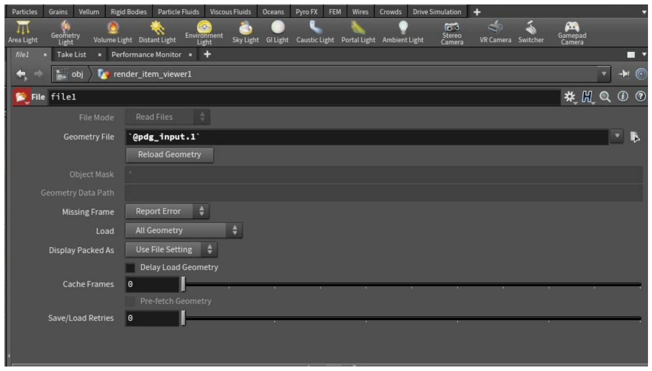
Now you are going to add a camera and a skylight then render out the cities. The resulting images will then be combined into a single mosaic with the wedge attributes on display so that creative decisions can be made based on the results. The mosaic node will use the Image Magick application which needs to be installed on your computer for these steps to work.



01

Tumble your view until you are looking down at the city from an angle. From the **camera** menu, select **New Camera**. Check the four city grids to make sure they all fit in the camera. Adjust the camera view if necessary. You will use this to render out the mosaic.

Add a **Skylight**. Set the Environment Light **Intensity** to **1.3**.



File file1

File Mode	Read Files
Geometry File	\`@pdg_input.1\`
	Reload Geometry
Object Mask	
Geometry Data Path	
Missing Frame	Report Error
Load	All Geometry
Display Packed As	Use File Setting
	Delay Load Geometry
Cache Frames	0
	Pre-fetch Geometry
Save/Load Retries	0

02

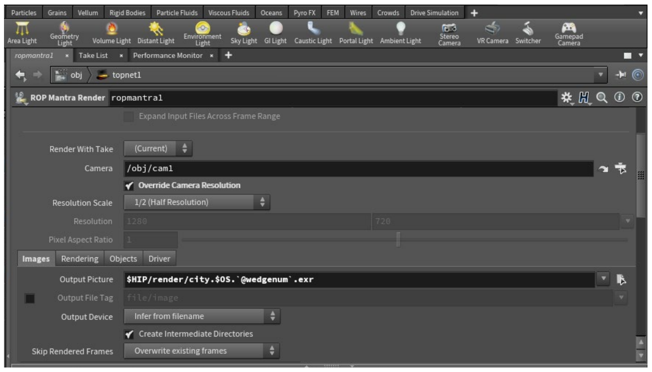
At the object level , **Alt-drag** on *work_item_viewer* to make a copy. **Rename** the new node *render_item_viewer*. Dive into this node and change the **Geometry File** parameter to:

```
`@pdg_input`
```

Repeat these steps to create *render_item_viewer1* with its **Geometry File** parameter set to:

```
`@pdg_input.1`
```

Set the display flag on these new *render_item_viewer* nodes and **turn off** the display on the two *work_item_viewer* nodes.



ROP Mantra Render ropmantra1

	Expand Input Files Across Frame Range
Render With Take	(Current)
Camera	/obj/cam1
	✓ Override Camera Resolution
Resolution Scale	1/2 (Half Resolution)
Resolution	1280 720
Pixel Aspect Ratio	1
Output Picture	$HIP/render/city.$OS.\`@wedgenum\`.exr
Output File Tag	file/image
Output Device	Infer from filename
	✓ Create Intermediate Directories
Skip Rendered Frames	Overwrite existing frames

03

Append a **ROP Mantra Render** top after the *partitionbyindex*. On the **ROP Fetch** tab, set the **Cache Mode** to **Write Files**.

Make sure the camera parameter is set to the camera you made. Turn **On** the **Override Camera Resolution** option and set it to ½. Change the **Output Picture** parm to:

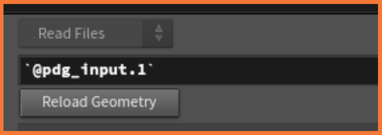
```
$HIP/render/city.$OS.`@wedgenum`.exr
```

Note: If you are running this tutorial using Houdini Apprentice then you should use `.pic` files instead of `.exr` to avoid the Apprentice watermarks.

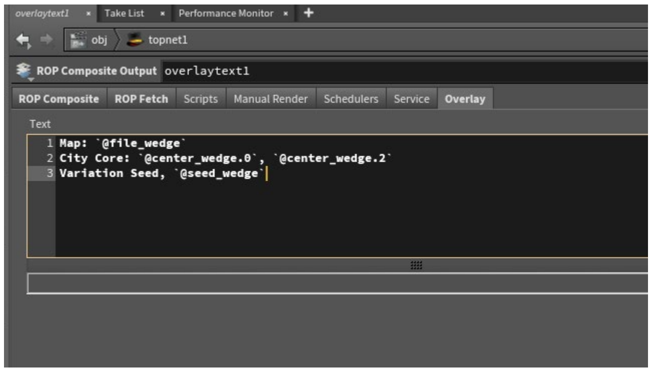
PDG INPUT

While PDG OUTPUT lets you take the result of a node and display it, PDG INPUT will take the result of the node before and display that instead.

This way the Mantra node can use the results from the preceeding node as the input for the geometry and then render that geometry. These two options are similar but understanding the difference is important.



Read Files
\`@pdg_input.1\`
Reload Geometry



ROP Composite Output overlaytext1

Text

```
Map: `@file_wedge`
City Core: `@center_wedge.0`, `@center_wedge.2`
Variation Seed, `@seed_wedge`
```

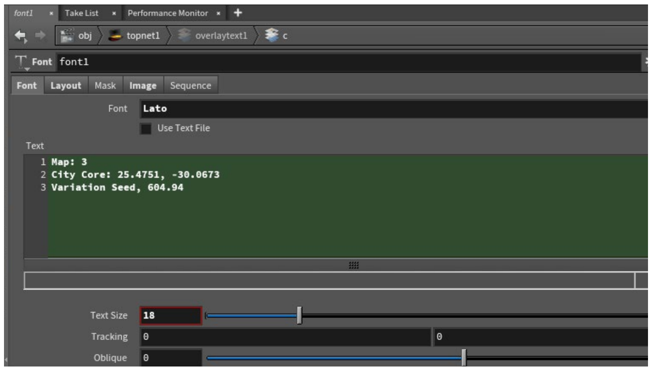
04 Append an **Overlay Text** top and set **Output Picture** to:

```
$HIP/render/$HIPNAME.$OS.`@wedgenum`.exr
```

On the **ROP Fetch** tab, set **Cache Mode** to **Write Files.**

In the **Overlay** tab type:

```
Map:  `@file_wedge`
City Core:  `@center_wedge.0`,  `@center_wedge.2`
Variation Seed:  `@seed_wedge`
```



Font font1

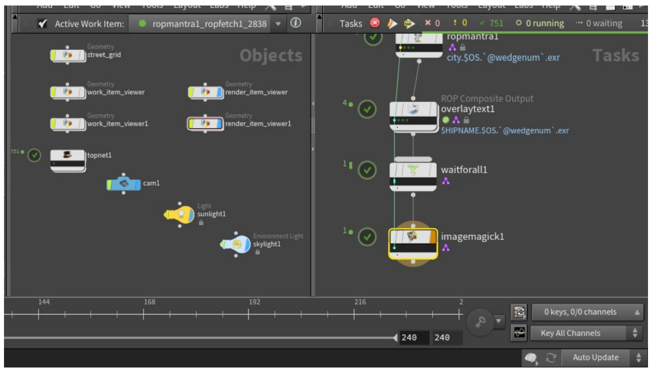
Font: Lato

Text

```
Map: 3
City Core: 25.4751, -30.0673
Variation Seed, 604.94
```

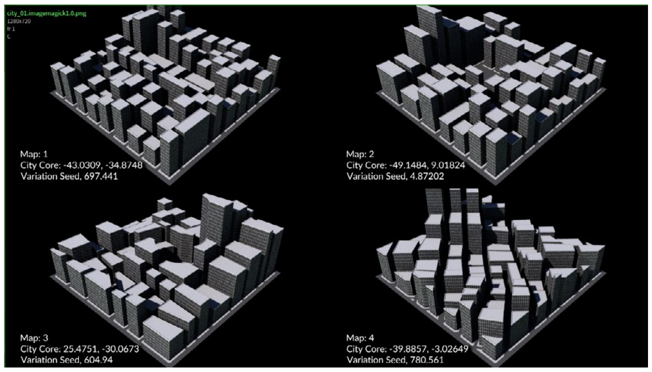
Text Size: 18
Tracking: 0 0
Oblique: 0

05 **Double-click** on the *Overlaytext* node to dive into it. Select the *Font* node and change the **Text Size** to **18.** This will create smaller text and leave more room to see the city.



06 Go back up and add a **Wait for All** partition node. This will bring together all of the elements. Now add an **ImageMagick** node.

Cook this node to process the TOP network. This will create four renderings and then overlay the text and combine them into a mosaic.



07 When you are finished, **RMB click** on the work item and select **View Task Output.** This will show you the final image in the **Mplay** image viewer.

This contact sheet lets you review the different wedges and decide which one you want to move forward with. The parameters displayed on the images are the ones you would need to lock down to get the city you want.

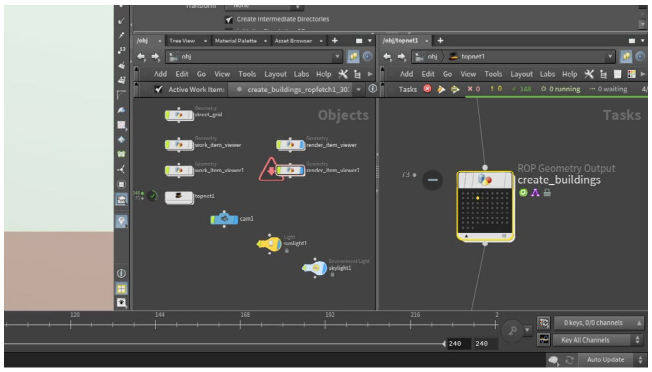
PILOT PDG

There is a standalone application called PilotPDG that is dedicated to creating TOP networks. You would not have been able to create this lesson in PilotPDG because that application can only work with TOP nodes and here you have used Houdini to create geometry networks and set up lights and cameras.

Once the network is finished, you could use PilotPDG to process the graph since it uses the same .hip format that is used with Houdini.

PART ELEVEN: Scale Up to Create More Content

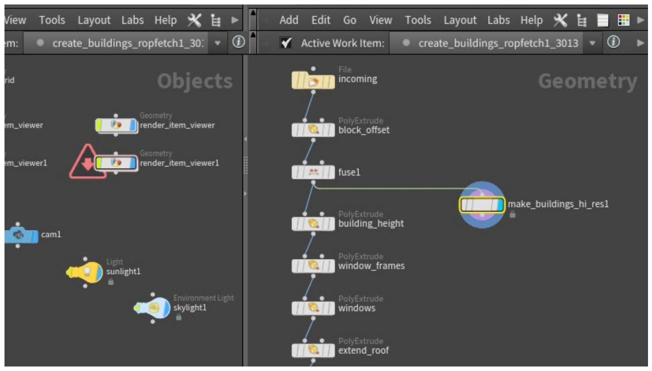
In the beginning you pointed out that this city building example could probably be created in SOPs without too much difficulty. The issue is that as the system gets more complex there is a bottleneck since all the processing happens within a single network. With TOPs, you can distribute work items to a compute farm and as you scale up things don't have to slow down.



01

On the wedge TOP, set the Wedge Count to 1. This will give you only one iteration for now while you scale up the city using a new map.

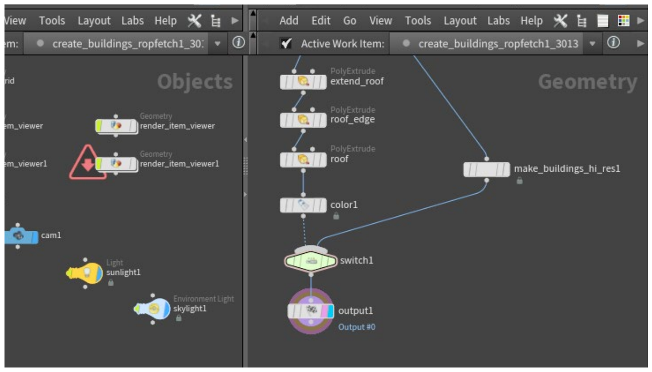
Select *create_buildings* ROP Geometry TOP node and **RMB-click** and choose **Generate Items**. Now you are going to dive in and use a digital asset to create a different type of building with floors and windows and columns.



02

Double-click to dive down. From the **Assets** menu, select **Install Asset Library**. Navigate to the *hda* directory and select the *make_buildings2.hda.* Click **Accept** then click **Install** (not Install and Create) Press **tab > Make Buildings Hi Res** to place the node into the graph.

Wire the *fuse* node into the new asset.

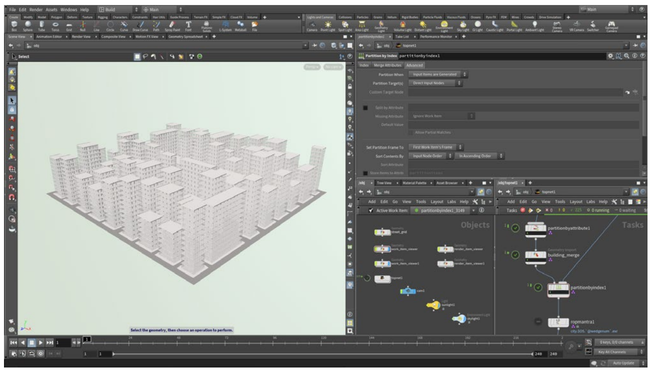


03

On the *make_buildings_hi_res* digital asset, set the following:

- **Floor Height** to **2**.
- **Base Building Height** to `@base_height`
- **Height Variation** to `@height_variation`

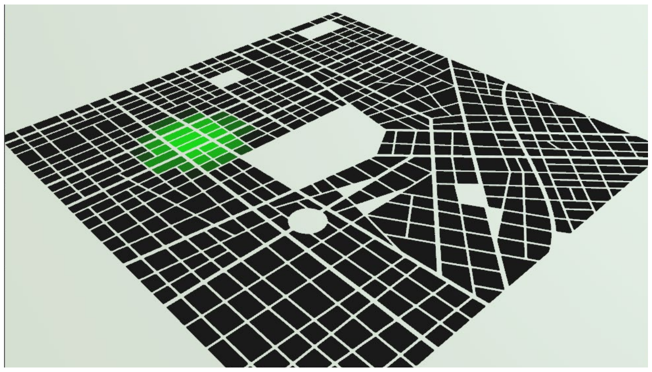
Add a *switch* node between the *color* node and the *output* node then wire the *make_buildings_hi_res* node into the second input and set **Select Input** to 1.



04

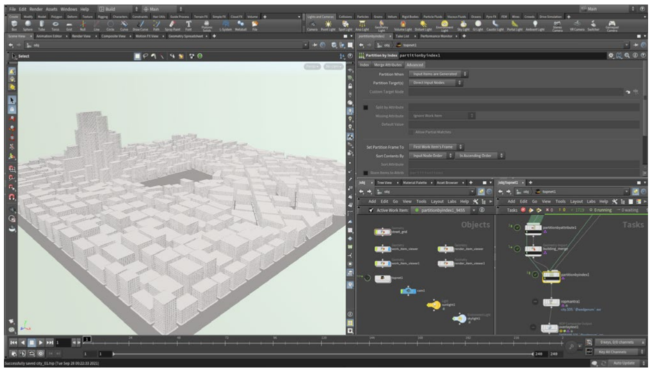
Select the **Partition by Index** node and **Cook** the network to see the new buildings. Now they have floor slabs, columns and windows.

A lot more geometry has been produced and you are starting to see that increasing the number of available work items using a compute farm would be a good idea. You could do this using one of the other scheduler nodes.



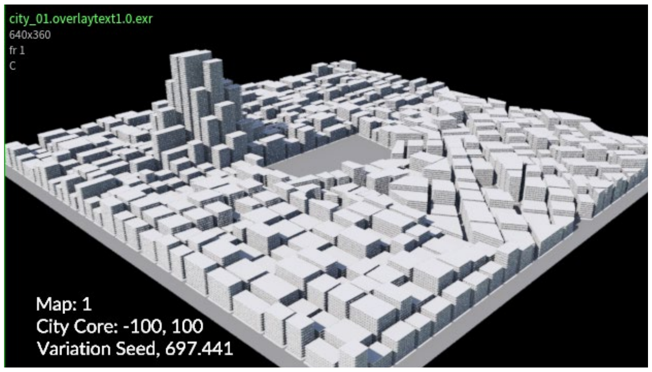
05 You could also make the city bigger using a different city grid image. You would need to increase the **Uniform Scale** on the *streetgrid_maker* HDA processor to **500**, and change the **Image input** to the *citygrid_large* map. You may also want to increase the **Blend Width** to create more falloff in the city core.

On the Wedge node, you could set the *center_wedge* attribute's to match your new map**. Dirty and Cook** the *streetgrid_maker* HDA processor node. You will see the new city map being generated.



06 When this is finished, you will have a lot more city blocks, compared to the original map and a lot more buildings are being generated.

Going to a compute farm would definitely make sense at this point because a single computer is not taking advantage of the parallel processing capabilities of TOPS.



07 **Recook** the final node to render out an image of the city. You can see how the simple system you built has the potential to grow to whatever size you need.

Save your scene.

CONCLUSION

In this lesson, you have used TOP nodes to take a city map, create buildings for each city block and then grow this system to handle more complex buildings and larger city maps. This project has introduced you to typical nodes used in a TOP-based workflow and how they can be used to create and process work item tasks.

There are many things you can create using TOPs and this is only the beginning. Not only can you use this network type to automate and process typical Houdini workflows, you can use it to process workflows that don't involve Houdini at all.

This will make it a great tool for managing dependencies throughout your pipeline, whether you are a small studio looking for efficiencies or a big studio managing lots of data.