Houdini System

Robin van Grinsven - 141266

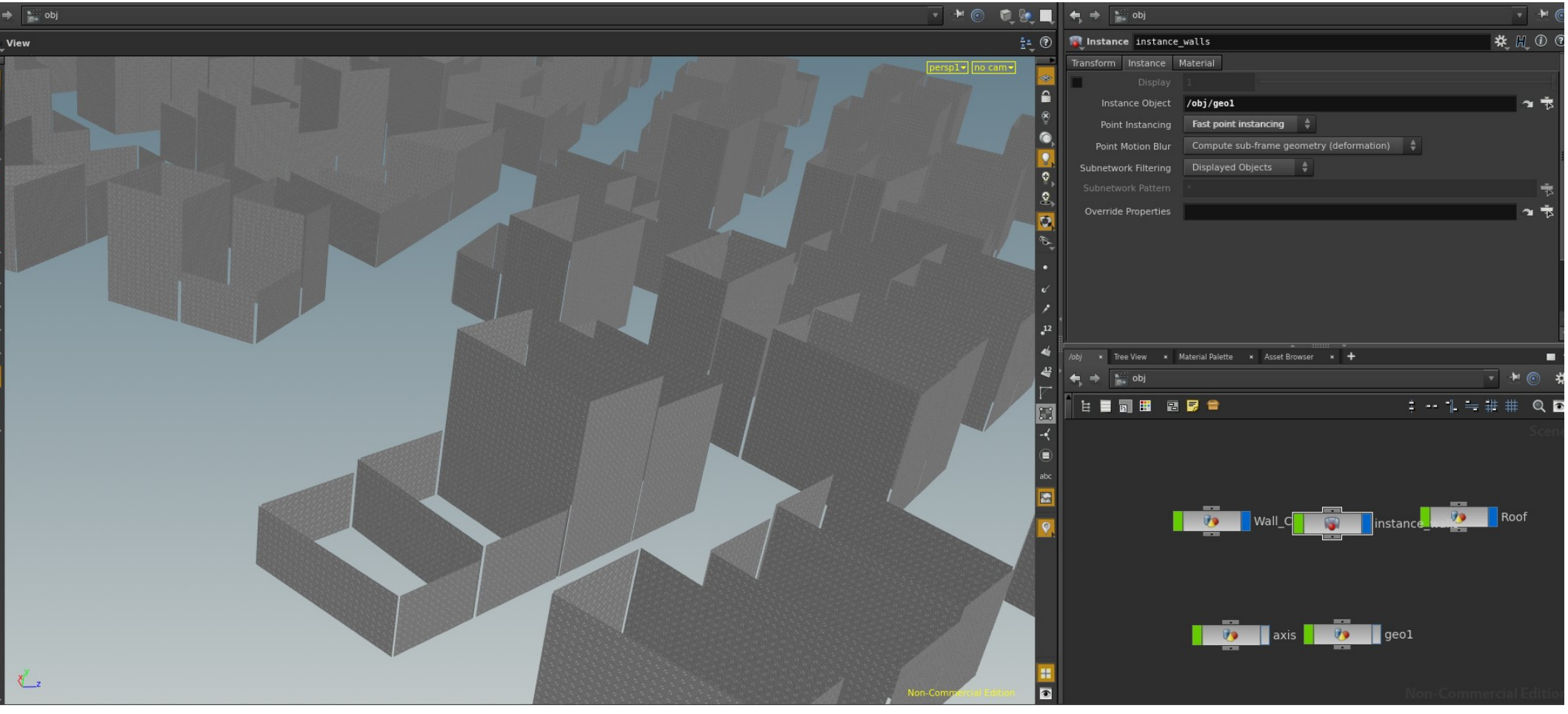
Table of Content

Intro

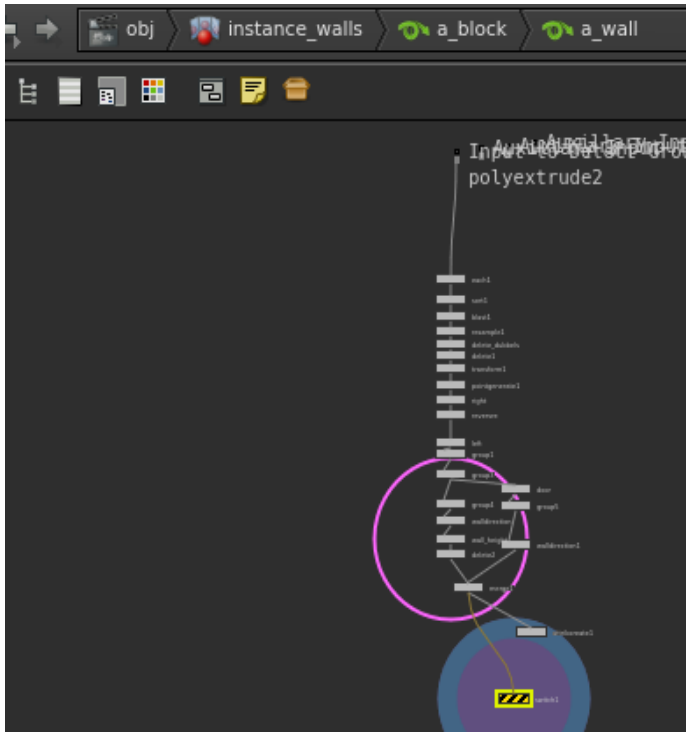
This block I had joined a 3 year team. They had the problem that they want a huge environment with many buildings. There were already buildings created however they where too heavy for the system. Taking those buildings as reference, I made my own city with Houdini with a point cloud system for the wall positions. These positions have been later saved to Unreal. The result was nearly there; we had a solid 28 FPS on my laptop. This presentation shows how it is done in Houdini.

First system result

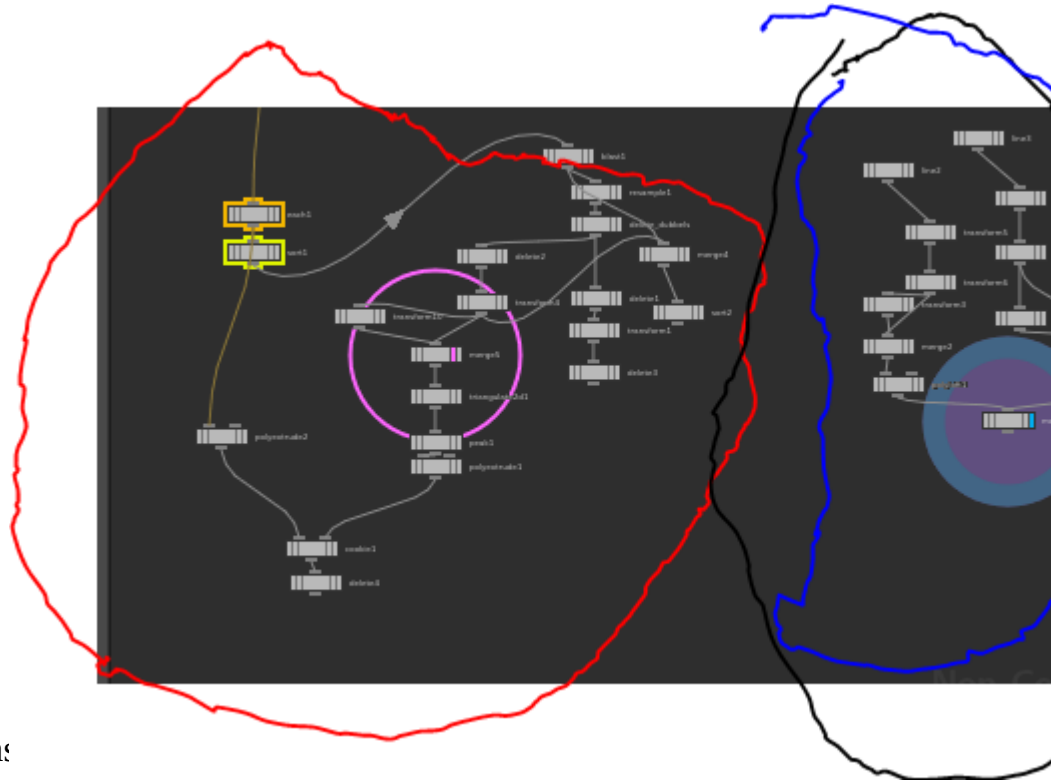
- On the right side of the screenschot you see that I did not make a digital asset from the system. This was my main fault. This way changing parameters becomes very hard because then many other parameters needed to be changed in order to work. See it as not using variables in coding.



Main issues with it



- Has the old for loop what brought less clarity on how the system works.
- The system was far from perfect. There was no deleting in the points of the middle. And the algorithms where working on a very limited occasion. What lead to to me having a hard time keeping up with new requests
- There was no central place to modify the values of the height of the buildings for instance. What lead to a bad pipeline where you go trough some nodes that need to change value. What meant that you might forget to change one what leads to mistakes.



- Red is the same old system that had imported almost the same algorithm. blue makes use of it to create corners. However with this system the first algorithm needs to be calculated 2 times.

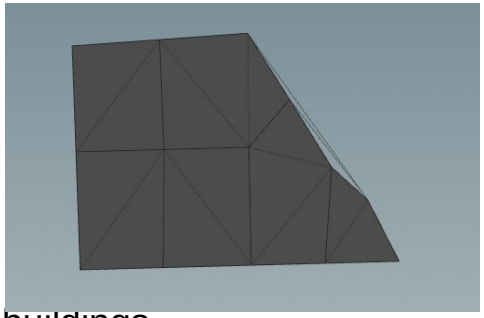
What it did do

- Create point cloud for the walls.
- Work in the Y direction.
- Cooking time of 1 minute(new system:22 minutes)
- Creating bad roofs and corners.
- No collision boxes created.
- Nor was there a point deletion of double unseen points.
- Placed some points at a wrong position

Next system and why

- Redid the system. So everything is more clear and I know what it did.
- For each new loop could be implemented. Which is more clear and is able to access nodes outside the for loop what came in handy later
- Now I have more knowledge of Houdini I am able to make tasks more clear.
- Learned the Houdini way of working. So now I can implement it

Added algorithm



For the intersecting building points we used this

- Used triangulate first to make a secondary shape that delete the points between buildings.
- Need other system since it is restricted to non convex shapes
- New algorithm picks the shape of the other buildings and use those to delete. The points that are intersecting with the current building.
- To get no gaps we made 2 checkpoints that represent the boundary's of the walls

  when zooming out in the far distance you see that there are no building pieces since every object get loaded if you are near enough. However you have a big mech that is flying trough city's so you can look very far.

- We solved this by make walls inside the building. So there are flat planes at far sight. It was a quick solution for delivering a reasonable result

  because we do not want to have collision boxes on every wallpiece I developed a collisionbox generator in Houdini that just is a convex shape of every building. So we would not have a overkill with collisions.

Improve algorithm

- Delete all intersecting points. By making the other buildings collide with the points and delete those points
- Was not enough some points that were the wall was sticking outside the point cloud but point was intersecting the other building
- Did a point duplicate for the bounding boxes of the walles. Check if any is not in collision box if true do not delete them

Layout change

- We had some feedback that the building area's where clearly separated from each other. By changing the layout to have buildings switched districts we made the environment more diverse and the city does not look that perfect then.
- We've done this by switching layout faces between districts. So some houses in the corporate are business. This prevented hard transition between districts

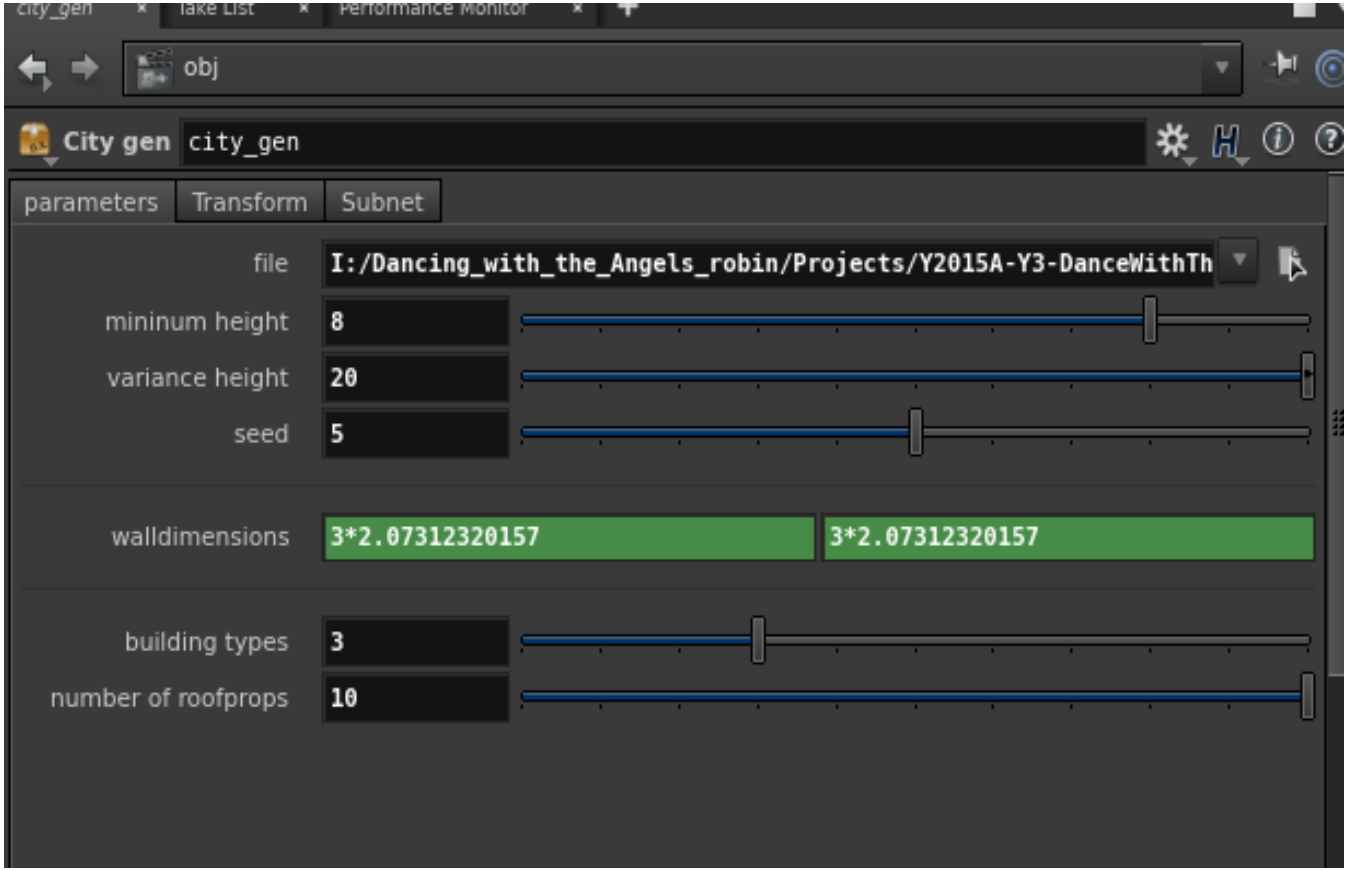
Importing to unreal

- Houdini engine not a solution. Because we where not able to obtain a Houdini engine license. Because we where not sure what the future would hold. And what the profit will be of the game.
- First time point data creation with python
- Write/Read csv files from pointcloud
- To slow so made it binary
- Used the up rotation value. For defining what wallpiece should be placed there

Prop placement

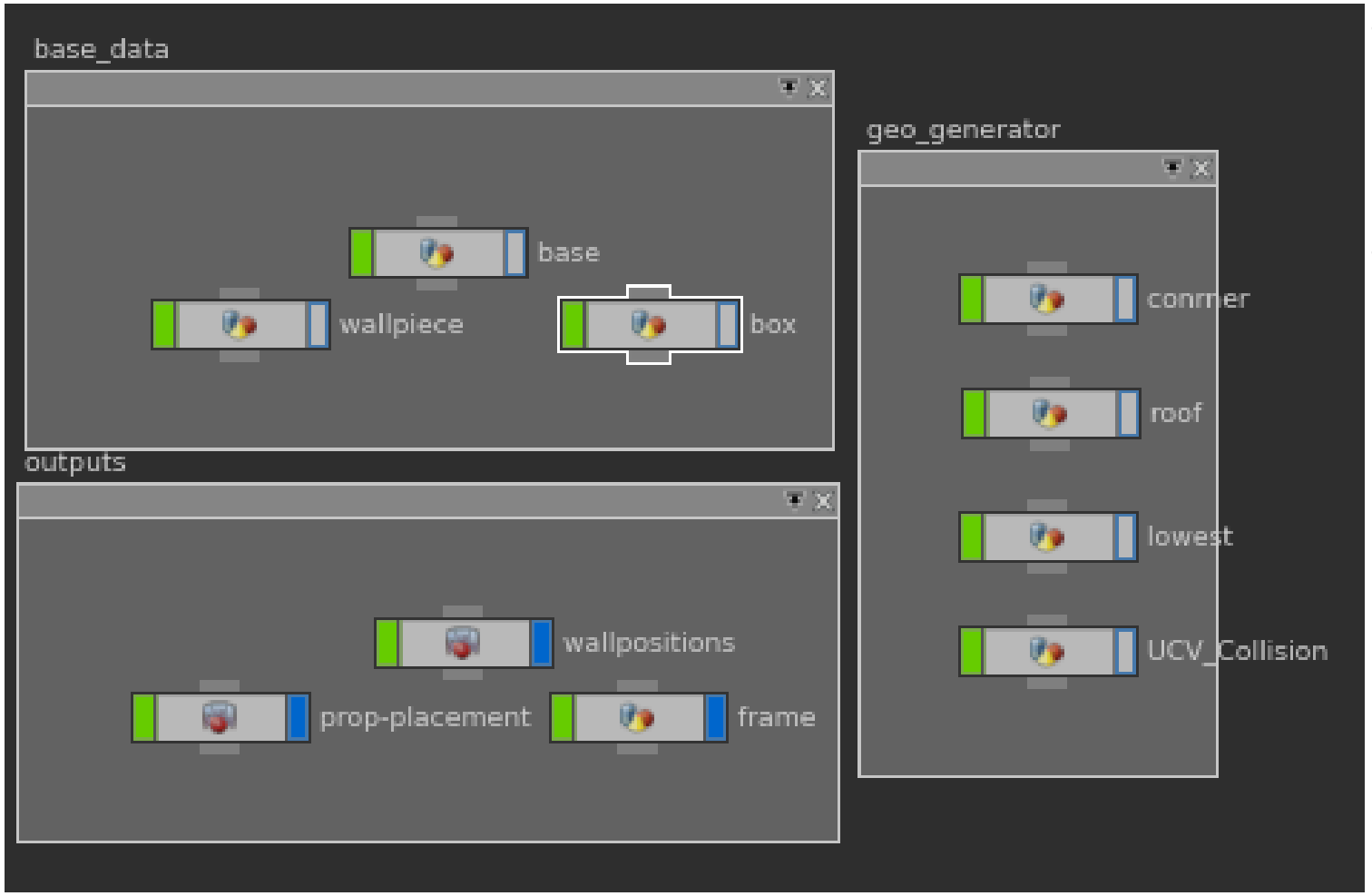
- We had one prop placer in the team. However it is a big city so it was a big task. To enlighten his work I made an algorithm that made points on rooftops that pick random prop and random rotation. So the props where instant placed.
- The props had some rules. For instance the props can not be placed on the sides of the shape because then you get meshes intersecting the edge or even hanging over it. Also it needed to look arbitrary. Not place many props on the side of the building. All of the things I have solved later in this presentation at dia 36.

Digital asset



- Through the process I noticed that some parameters needed to be changed during the time. Also when a new request came in I needed to change multiple parameters to get the right effect. So I made a digital asset out of the system so I only need a few parameters to fix that

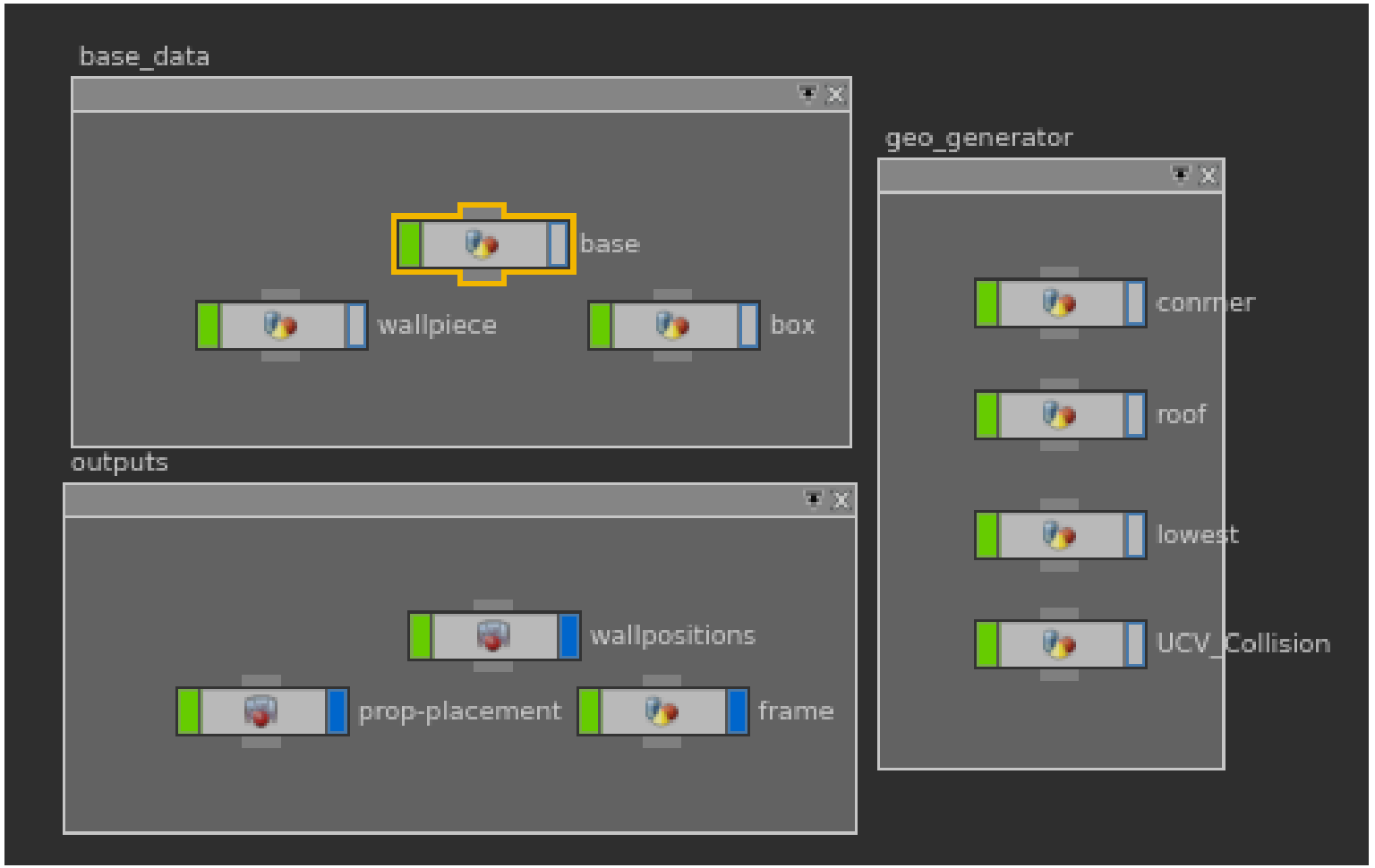
Walkthrough



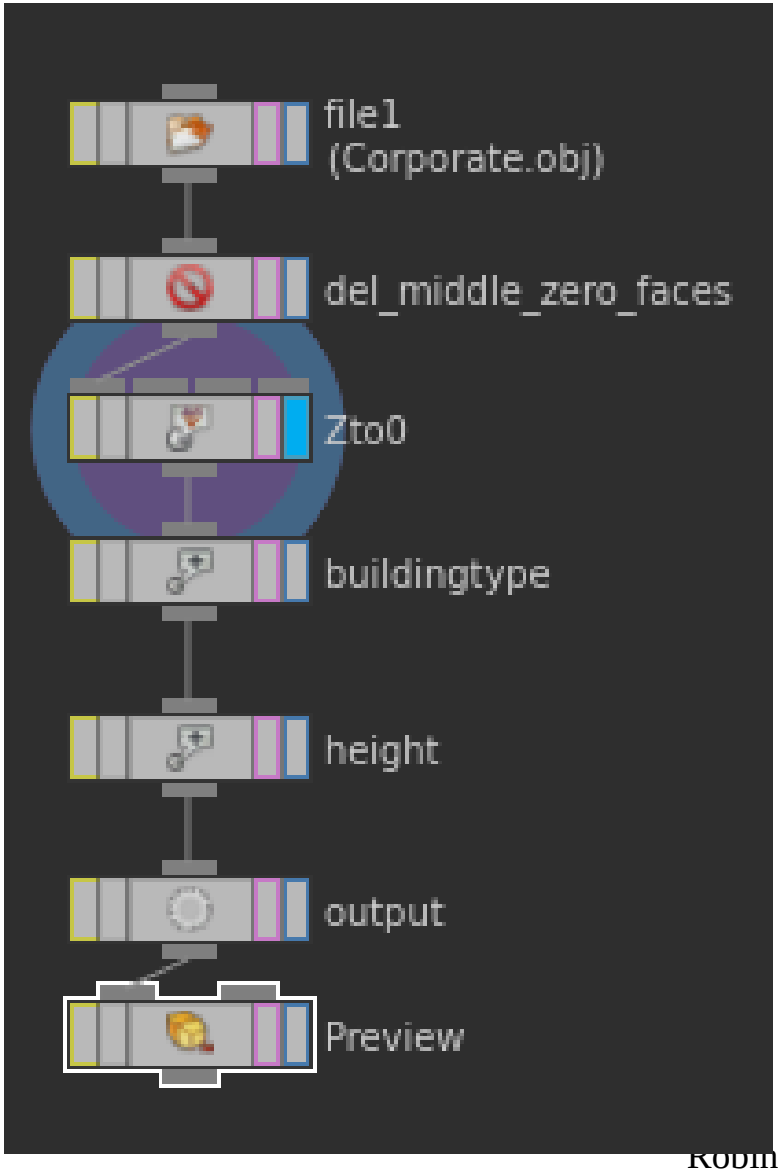
- Wall-piece and box are both predefined shapes for seeing what the results are in the point clouds(instances)
- We will go trough base, collision, lowest,roofs,Corners,frame,prop-placement and wall positions

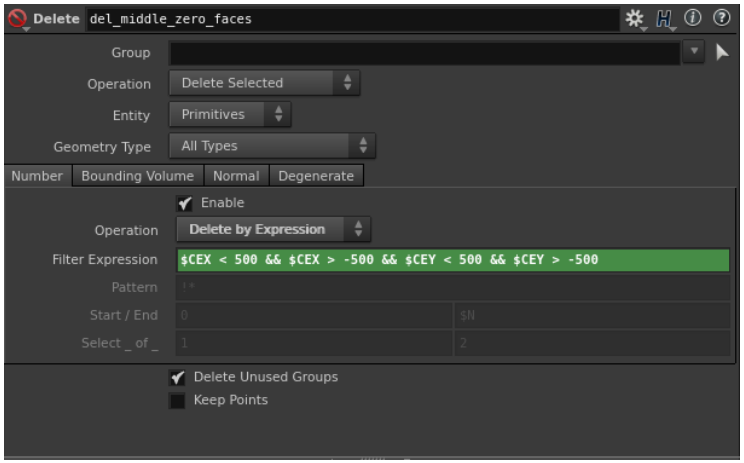
Base node

Where the layout get fixed to work with the system, and pre-generates the height and building type for the game



Delete node

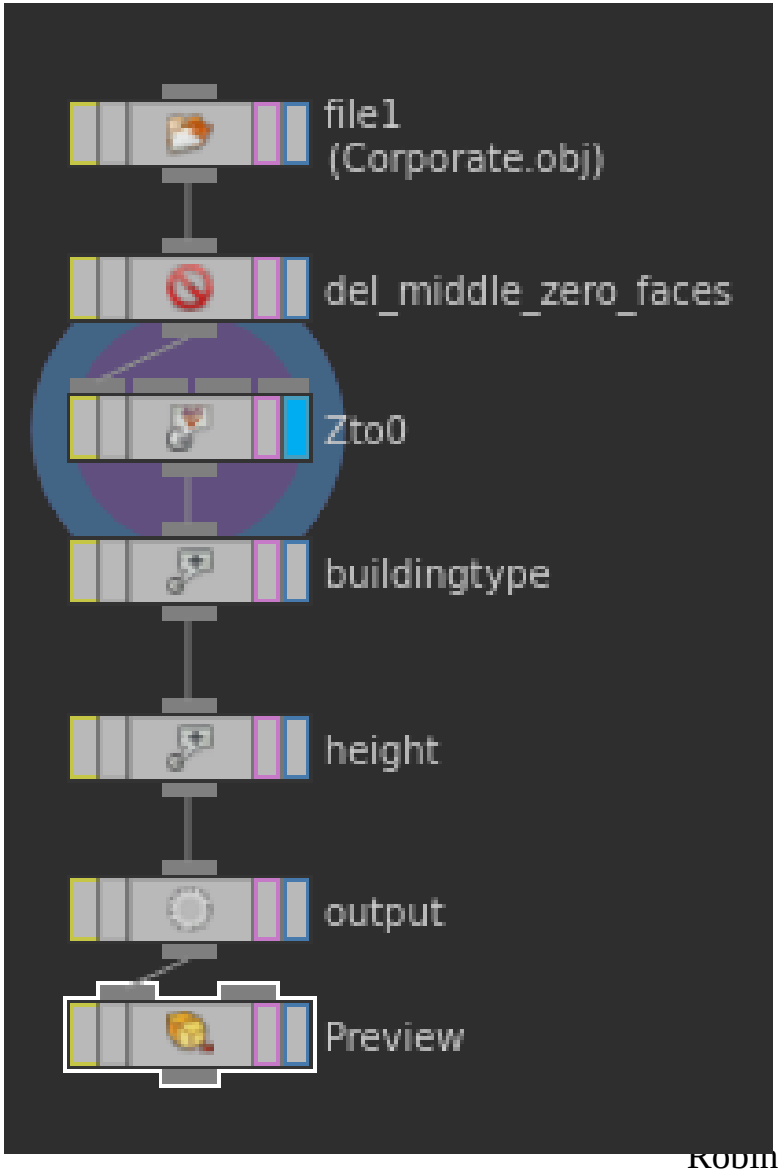


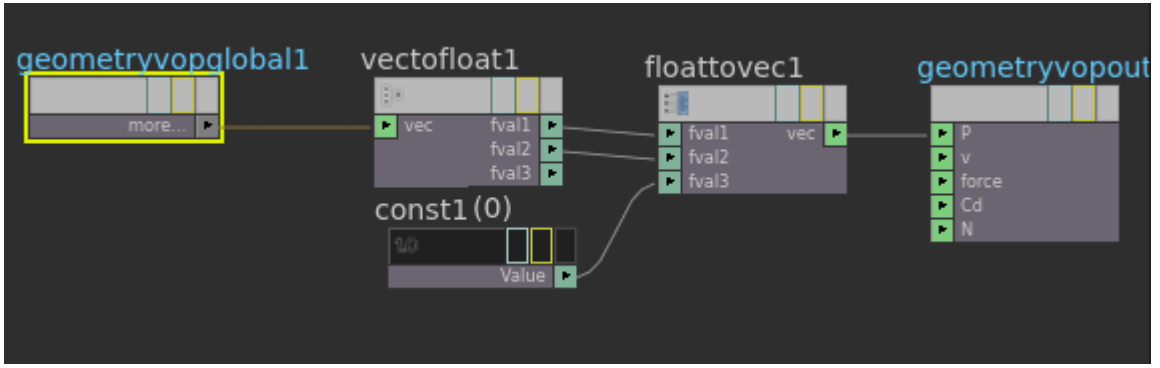


- First object import the layout geometry
- Second one is cleaning the layout with that there are no zero faces and no faces in the middle since . There is a sea there there can be no building there.

  The algorithm for this is: $CEX < 500 && $CEX > -500 && $CEY < 500 && $CEY > -500. this makes a box
- For cleaning purpose the delete node also deletes zero faces. Zero faces are redunant faces and messes up the system

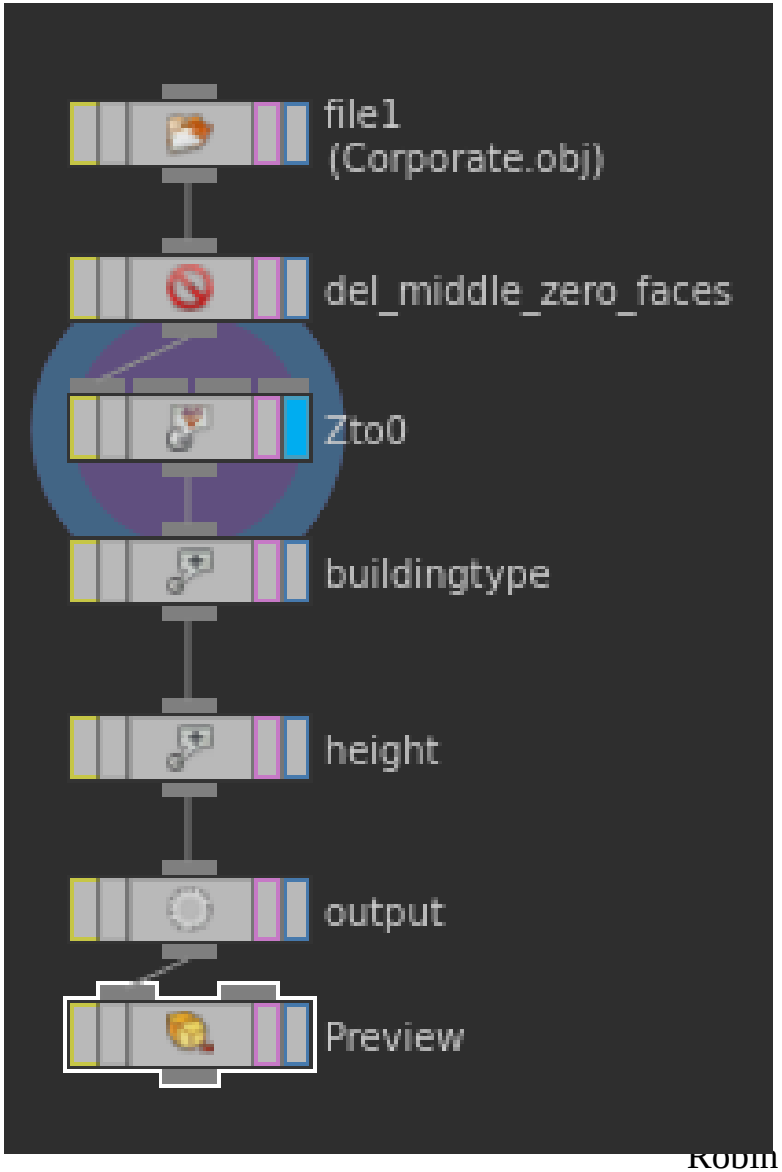
VOP node (perform math on point level)

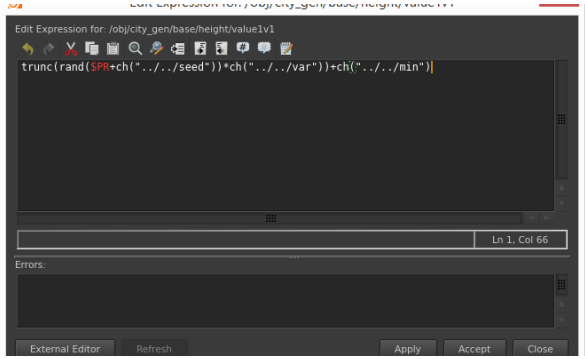




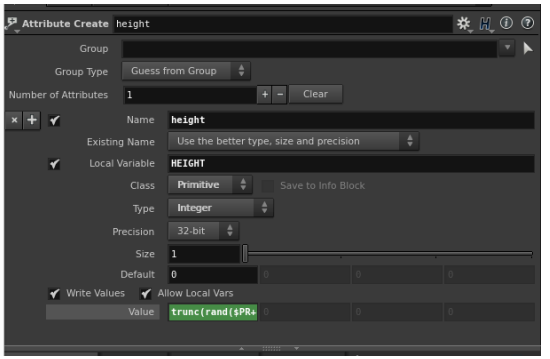
- Third node is there since the points z value are not exactly 0 what lead to floatpointerror.
- With this knowledge we tried to set all points to 0 by making the scale 0. this did not work
- So picked the vop node. Where we can overwrite the value to 0

Attribute create node





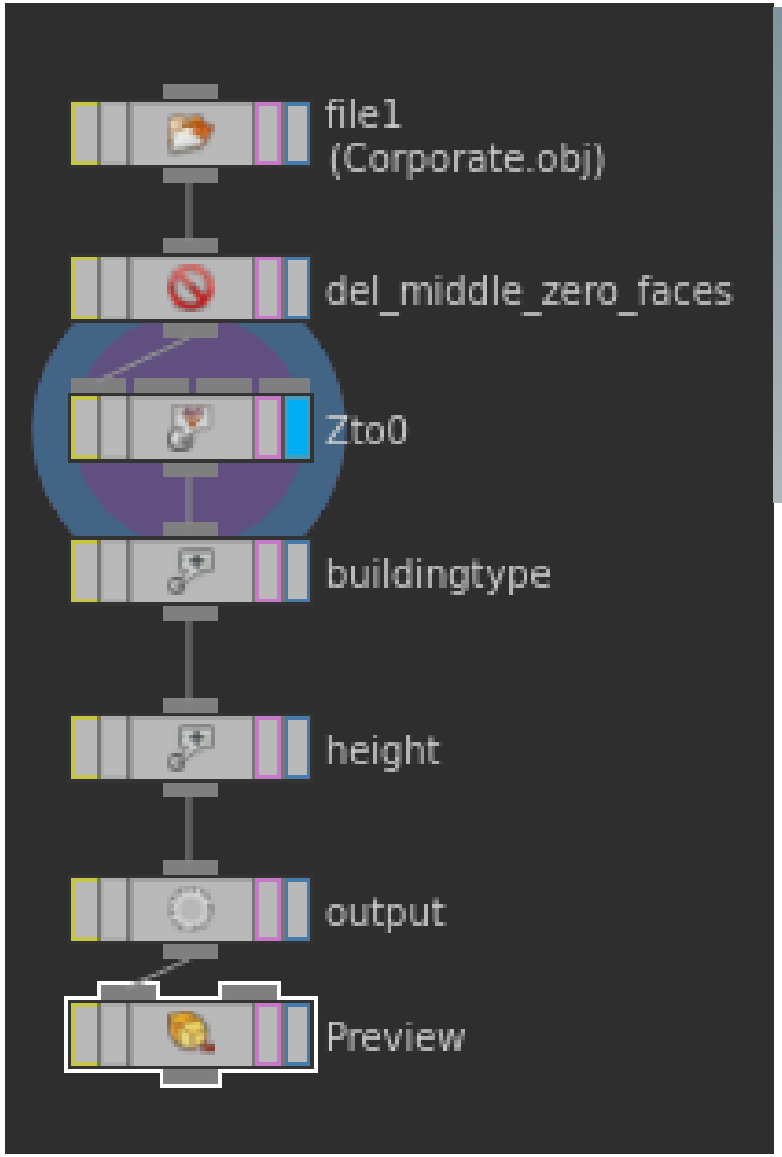
The algorithm

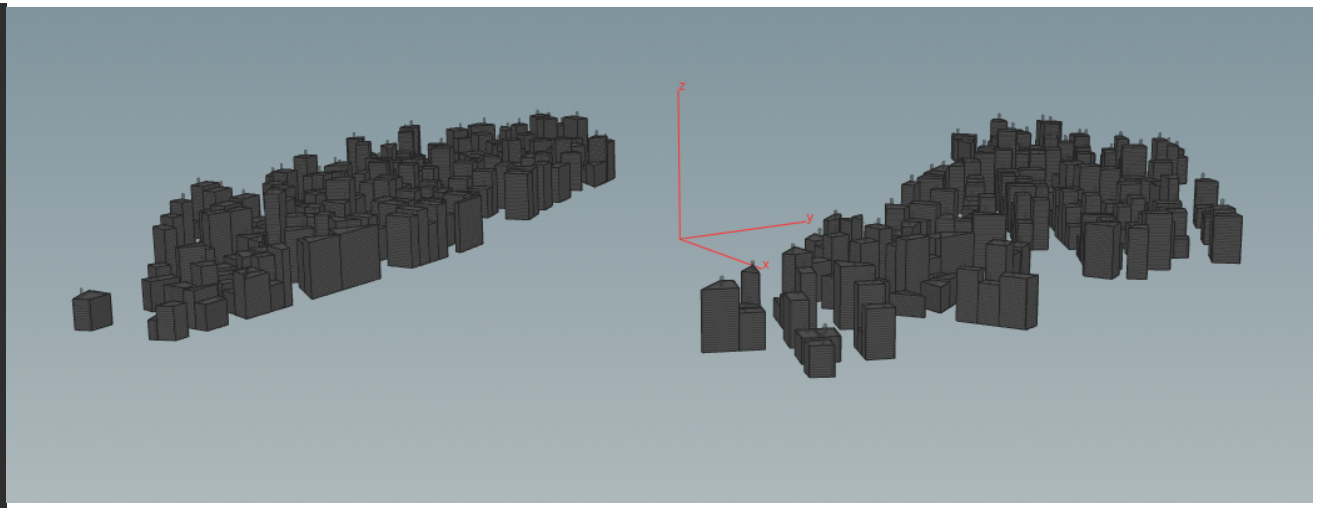


Settings

- Before I used the primitive number as seed number for random.
- But than the calculation needed to perform multiple times and you are not sure when some things change. Here I assign the final values to the faces that stay on the faces
- The values are influenced by the settings in the digital asset so I do not have go to every node that needs this value when it needs different settings requested by designers

Preview node

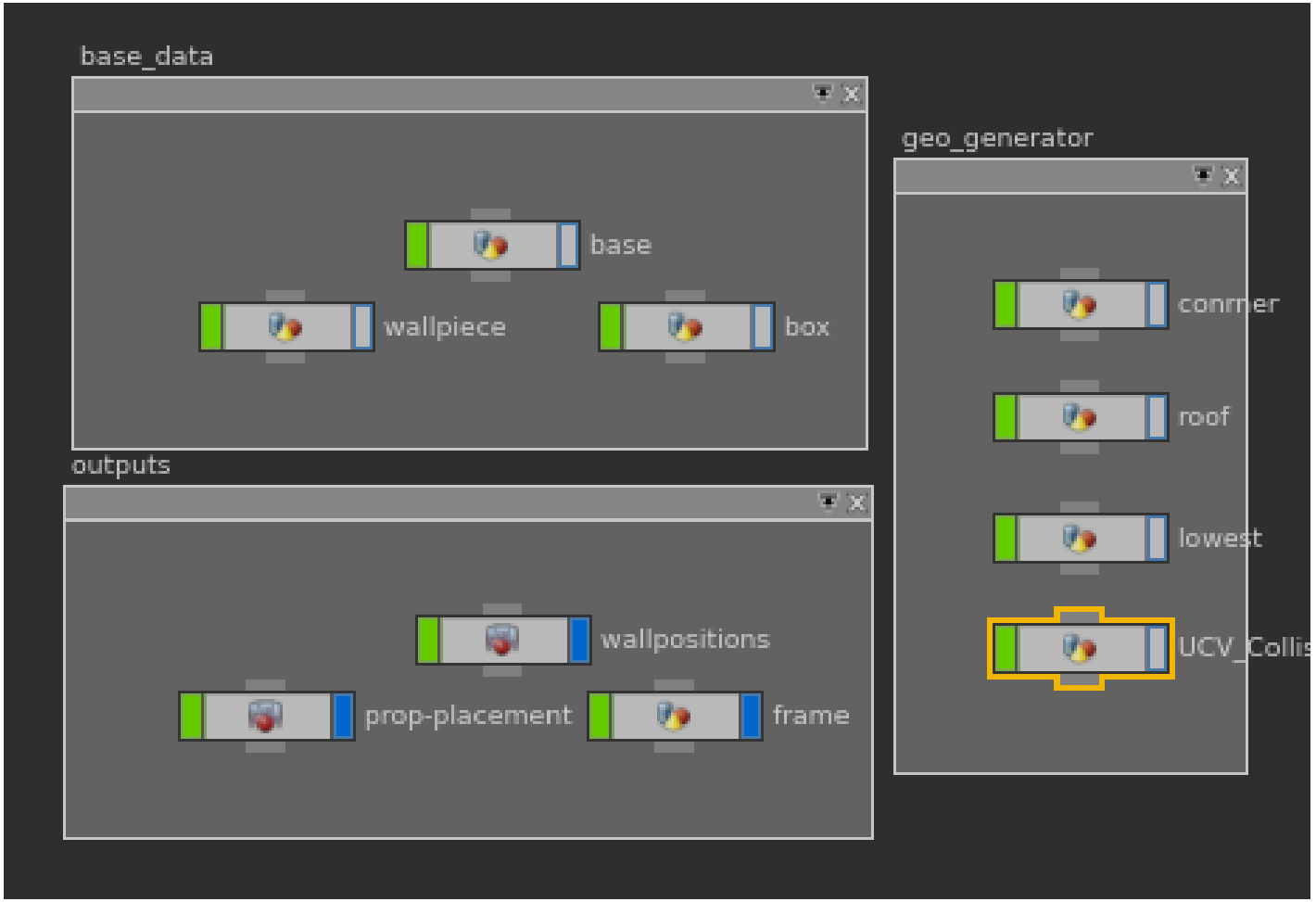




- Created a preview of the shape of the buildings

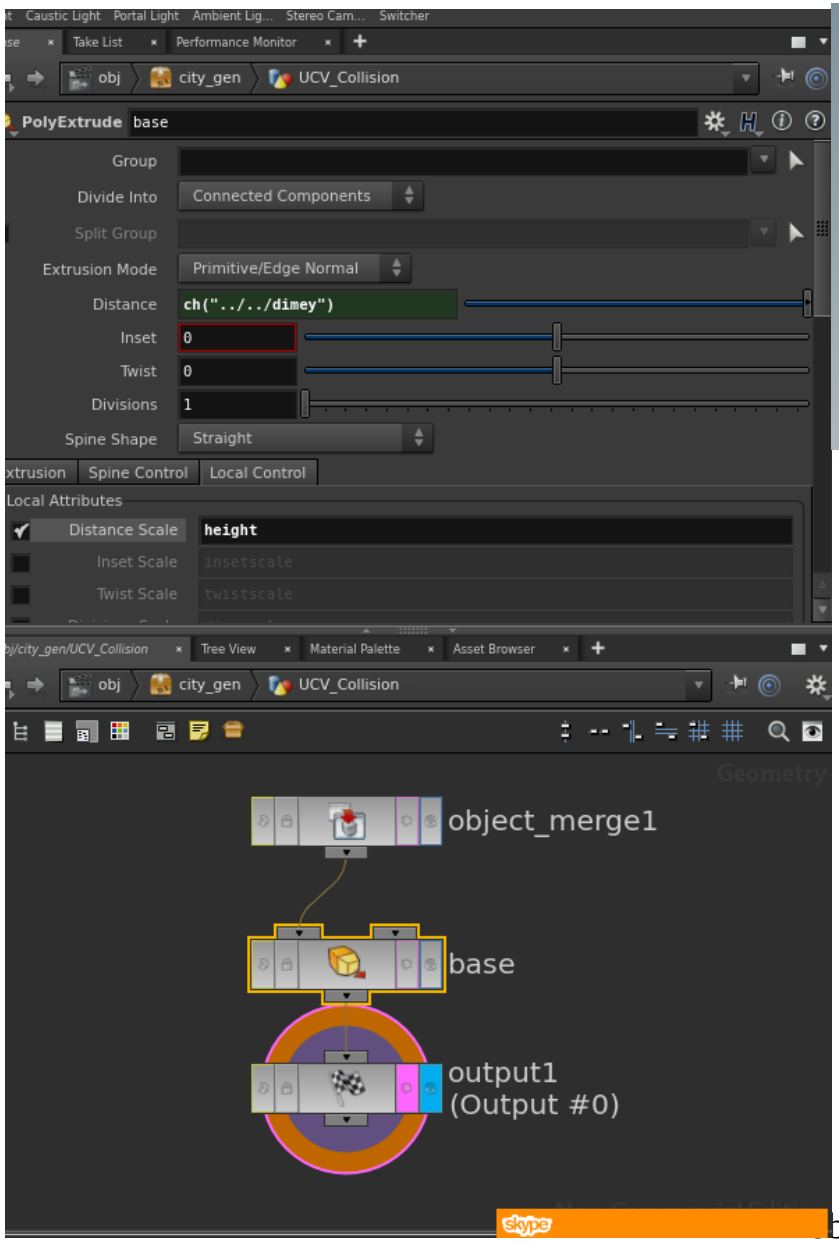
Collision node

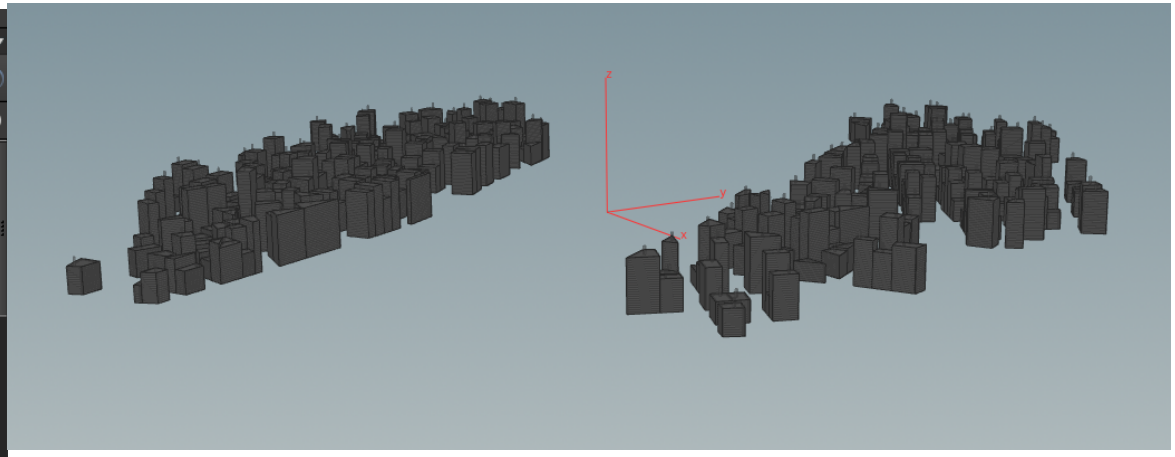
makes simple boxes that represent the shape of the buildings



Basic collision needed so we do not need to have collisions per wall piece what results in 1000 times less collision boxes in the level

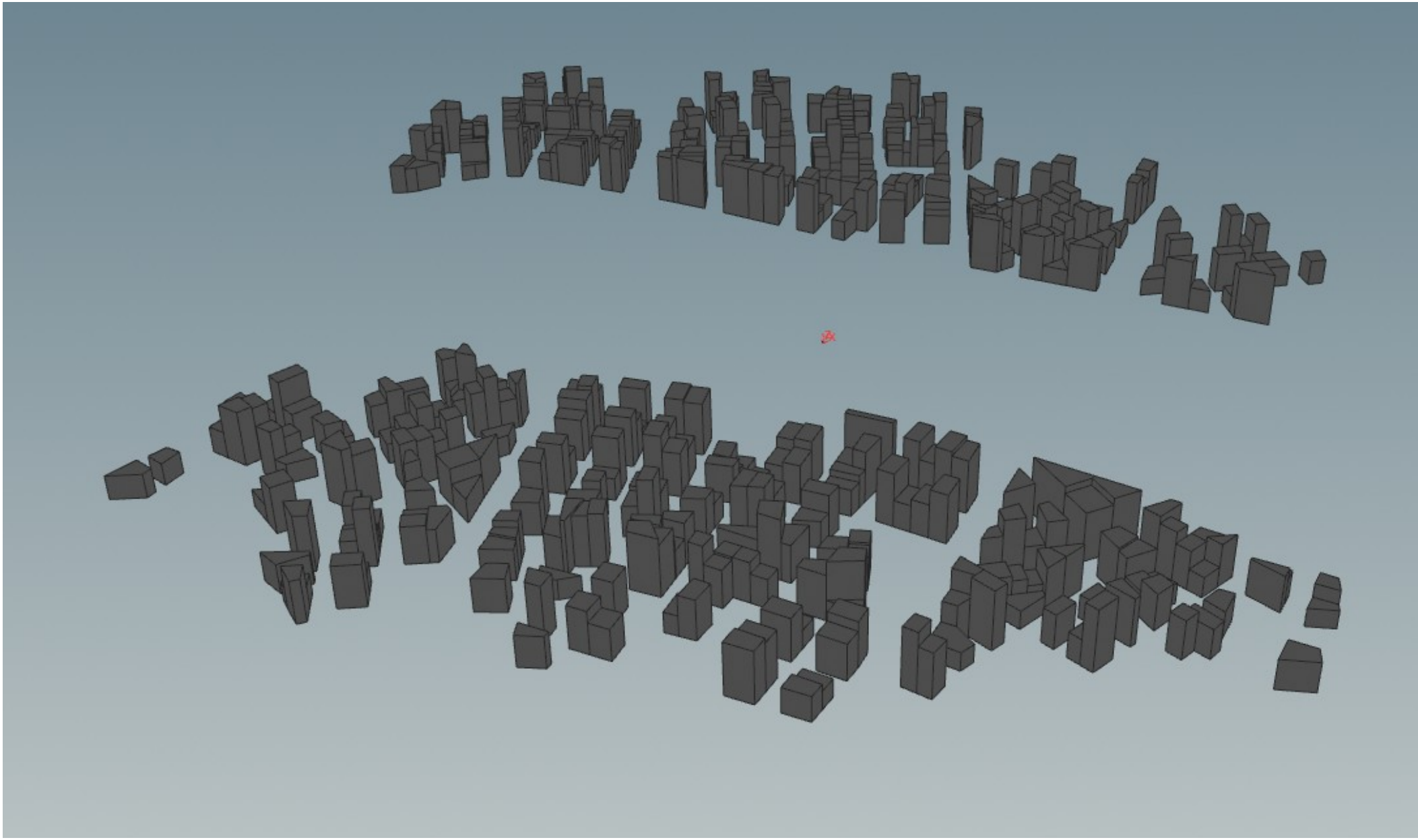
Base node





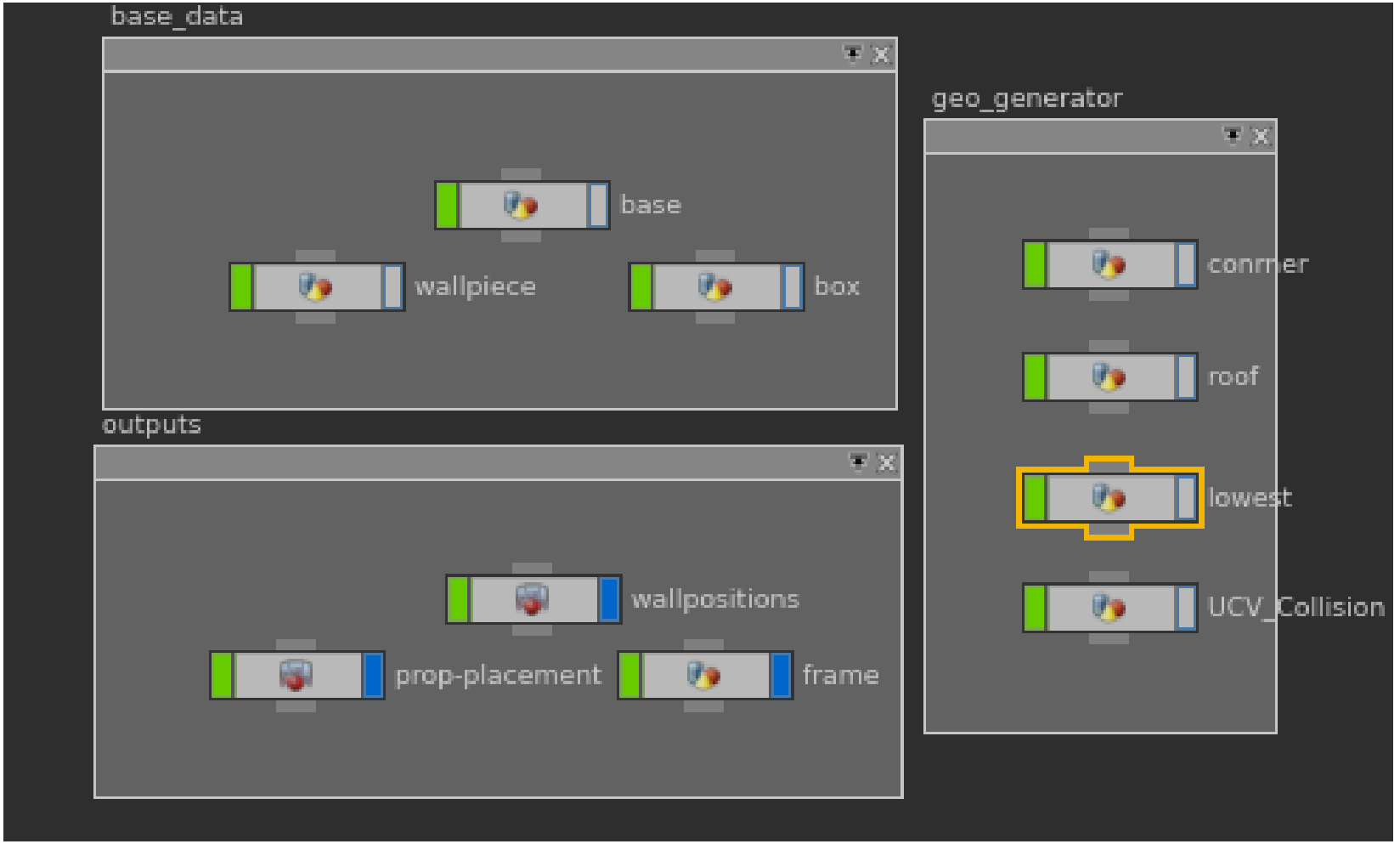
- Since the distance is pre calculated in the base node. All that needed to be done was to add the height. What is set in the attribute
- And the distance is set in the digital asset. When asset change all nodes know it so it always works

Result



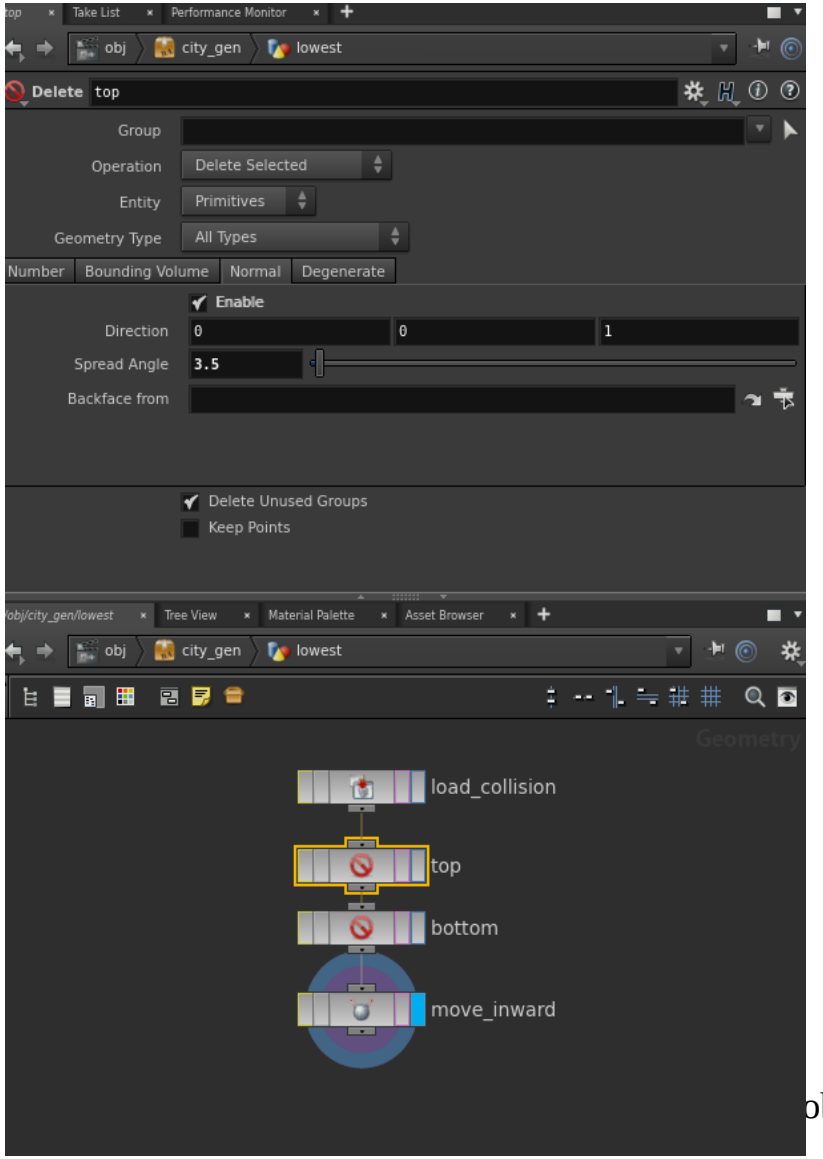
Lowest node

create the lowest LOD of the buildings



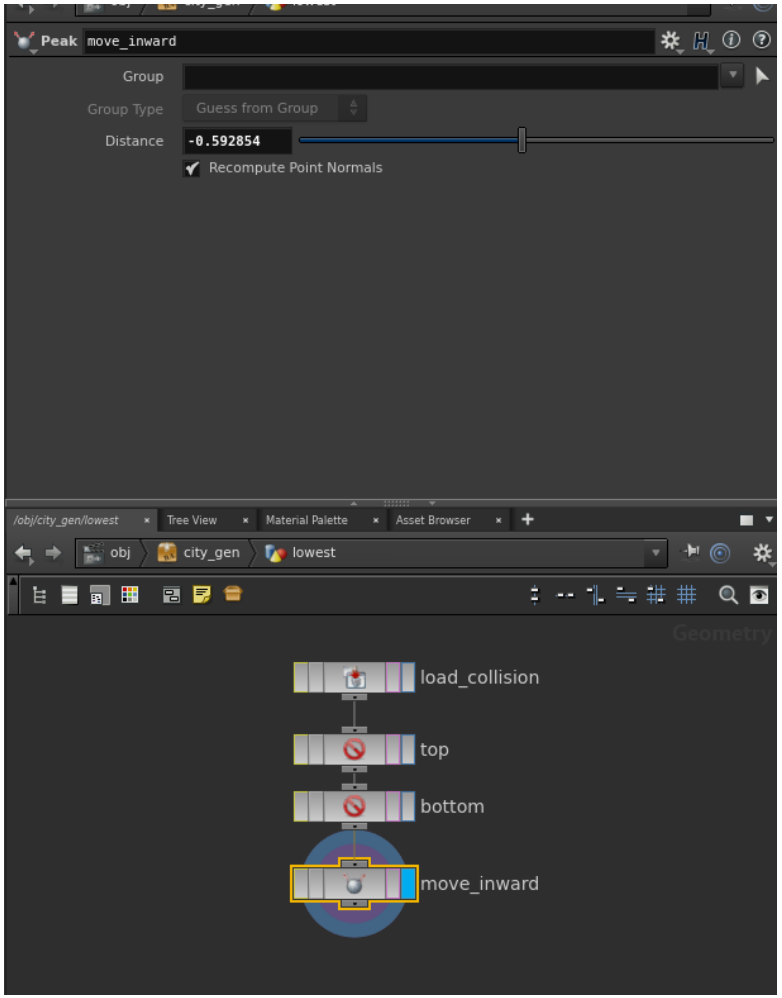
This node was needed because the camera clipping deleted the wall pieces too early when you are in the air. What resulted in buildings disappearing

Top/Bottom Nodes



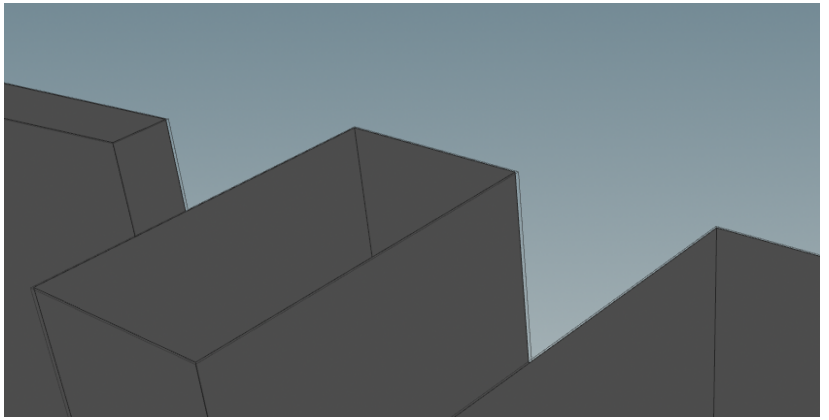
- Top/bottom nodes are nodes that look at the face normal and delete them according to the normal.
- We do this so the geometry is as minimalistic as possible and so the peak works fine.
- The small spread angle is there for catching error's in the system.

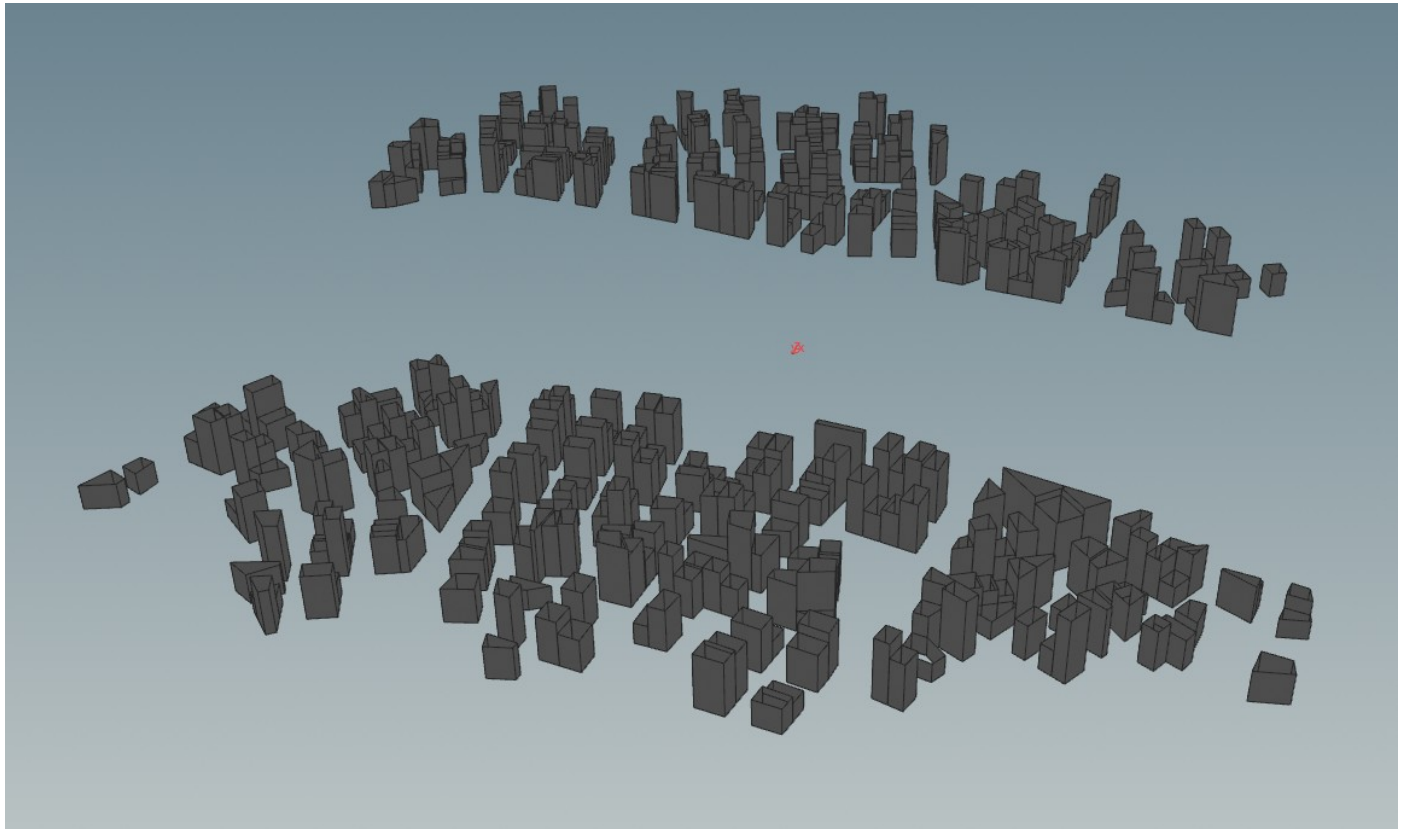
Move_inward node



- Picked the peak node this one moves points according to the normals
- Since the top face and bottom faces are deleted normals do not point in the z axis. Therefore moves only in x and y
- We do the small movement for 2 reasons
  - If we not do this z fighting would occur. We do this high value since you would be miles away before you see this object therefore the points are more likely to stick together.
  - The object exists at all times. Since we had no time to make a system around this LOD. But since the wall pieces have depth the wall can not clip through it

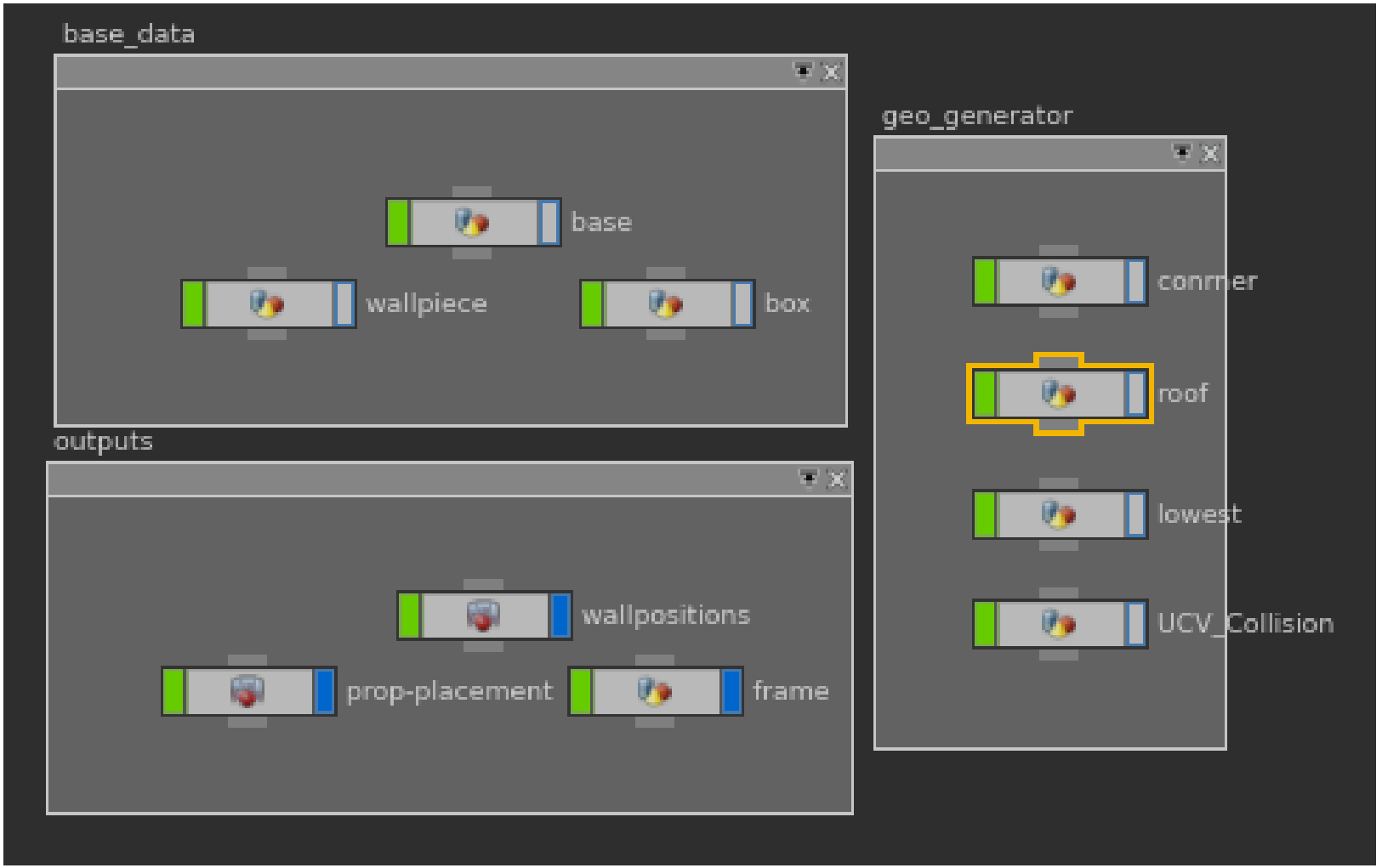
Result





Roofs

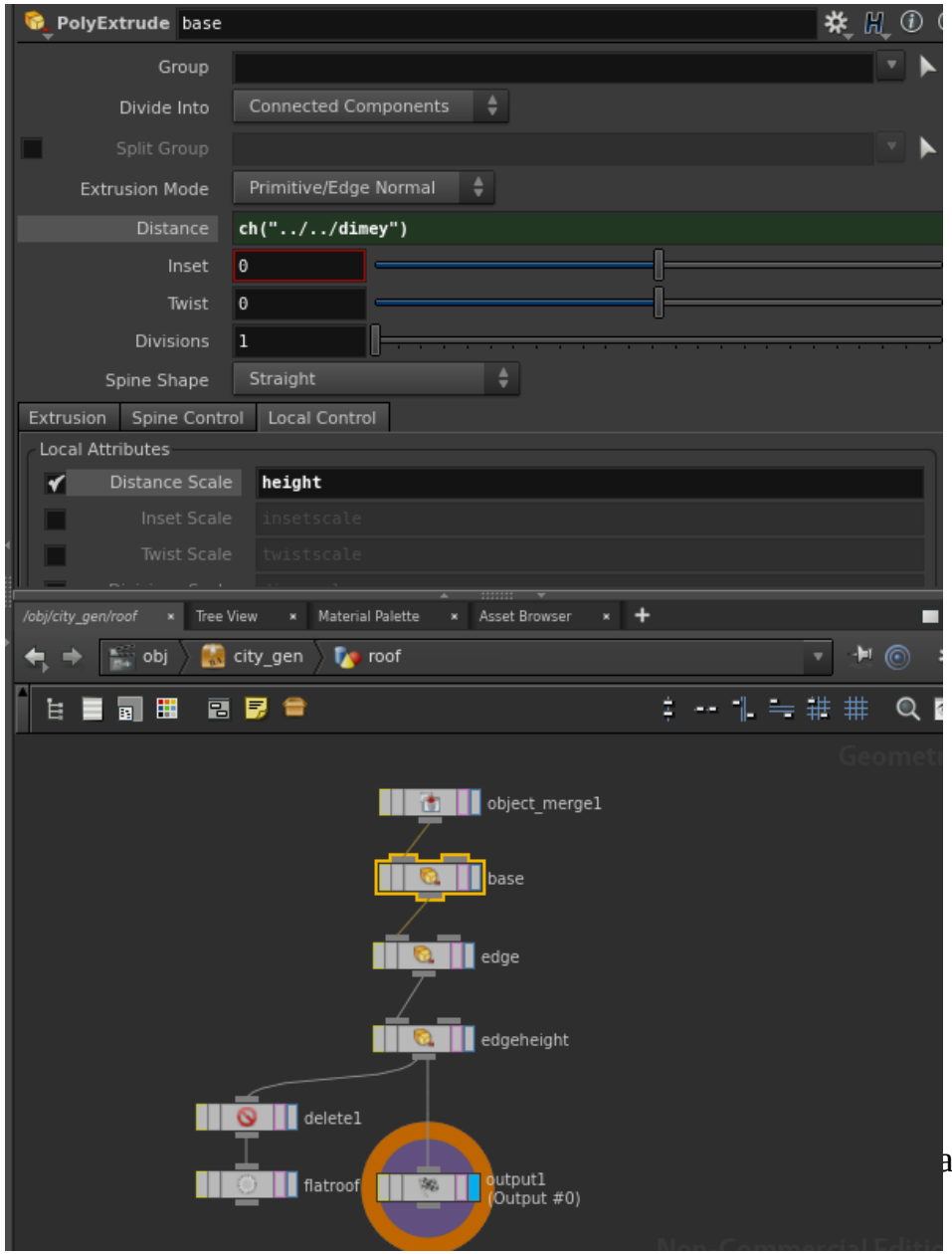
The cap of every building this is because every shape is different and different height the roof only is 1 type, the roofs needed to be flat since there needed to be props placed on it.



Roofs where had hard restrictions. One of them was that the most roofs should be flat. And there needed to be some props on them.

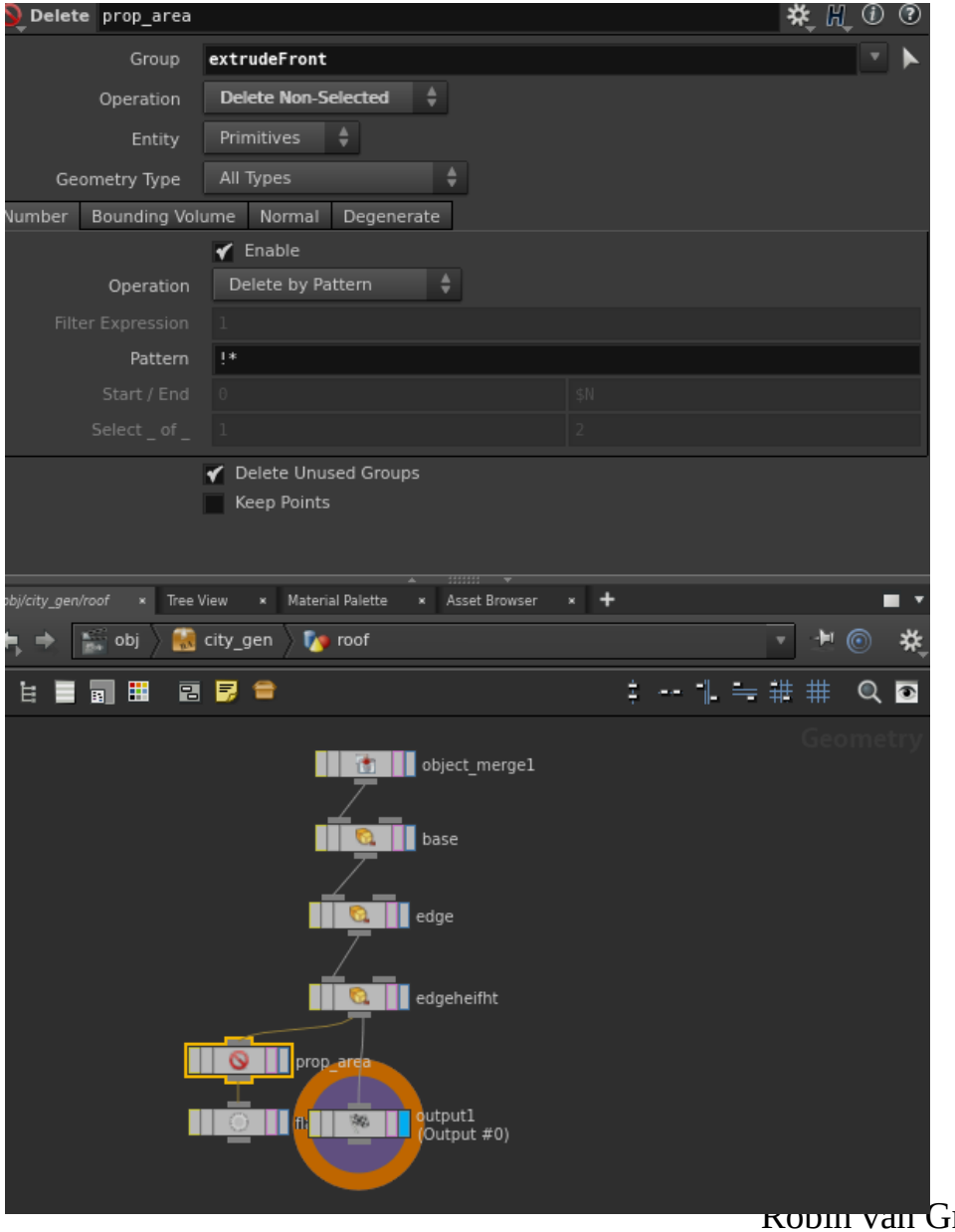
They needed to work on n-gons shapes and the orientation of the faces where arbitrary

Extrude nodes



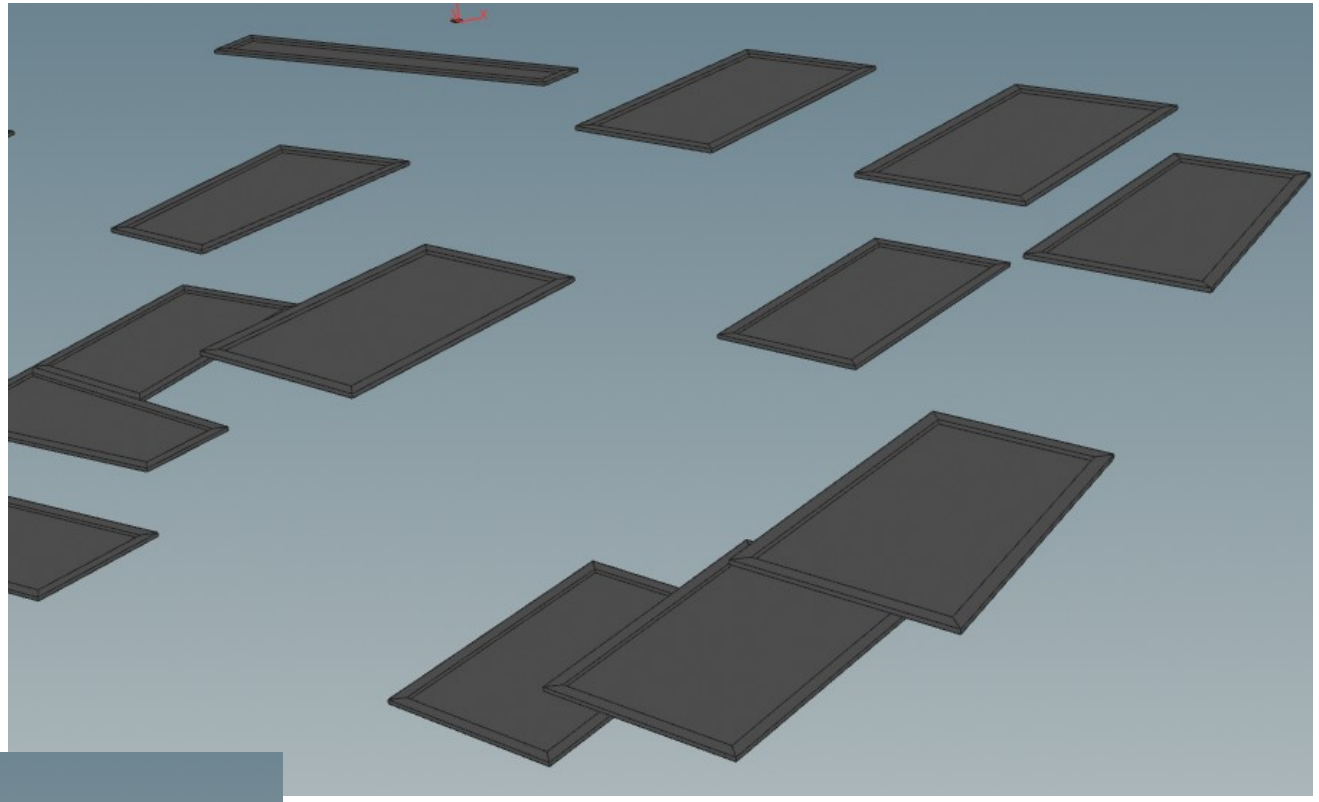
- Made one roof type. To be sure it works and that it wont take to much time. We had only 1 roof type.
- First extrude moves the faces just up to the building height.
- Second is for the movement inward of the shape.
- Third is for move the edge up

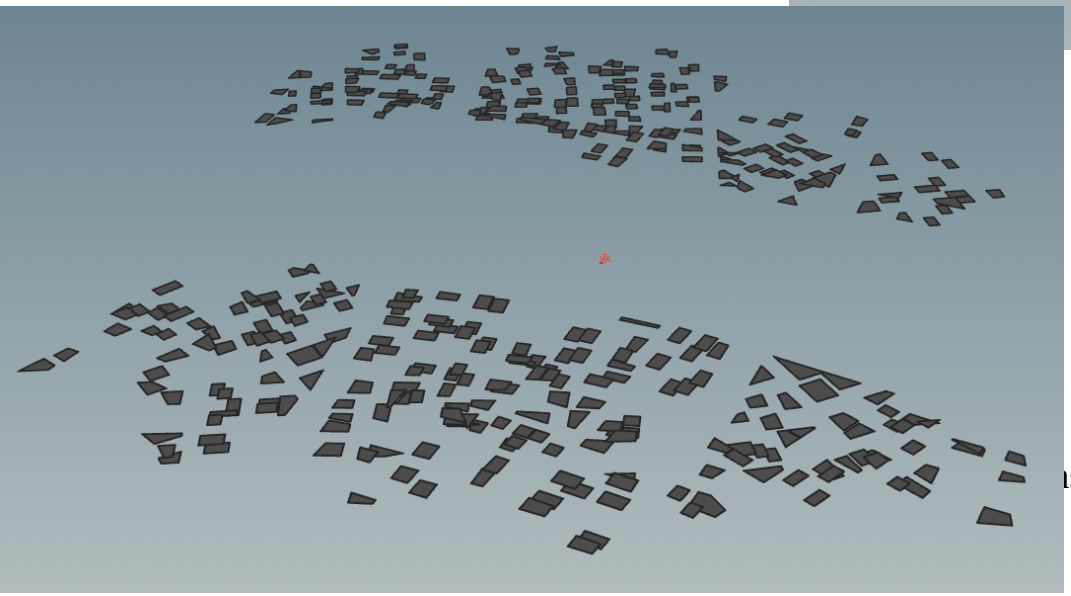
Delete1 node



- Here I delete all faces that can not be populated with props.
- For this I created a group in the edge node what outputed the middle face

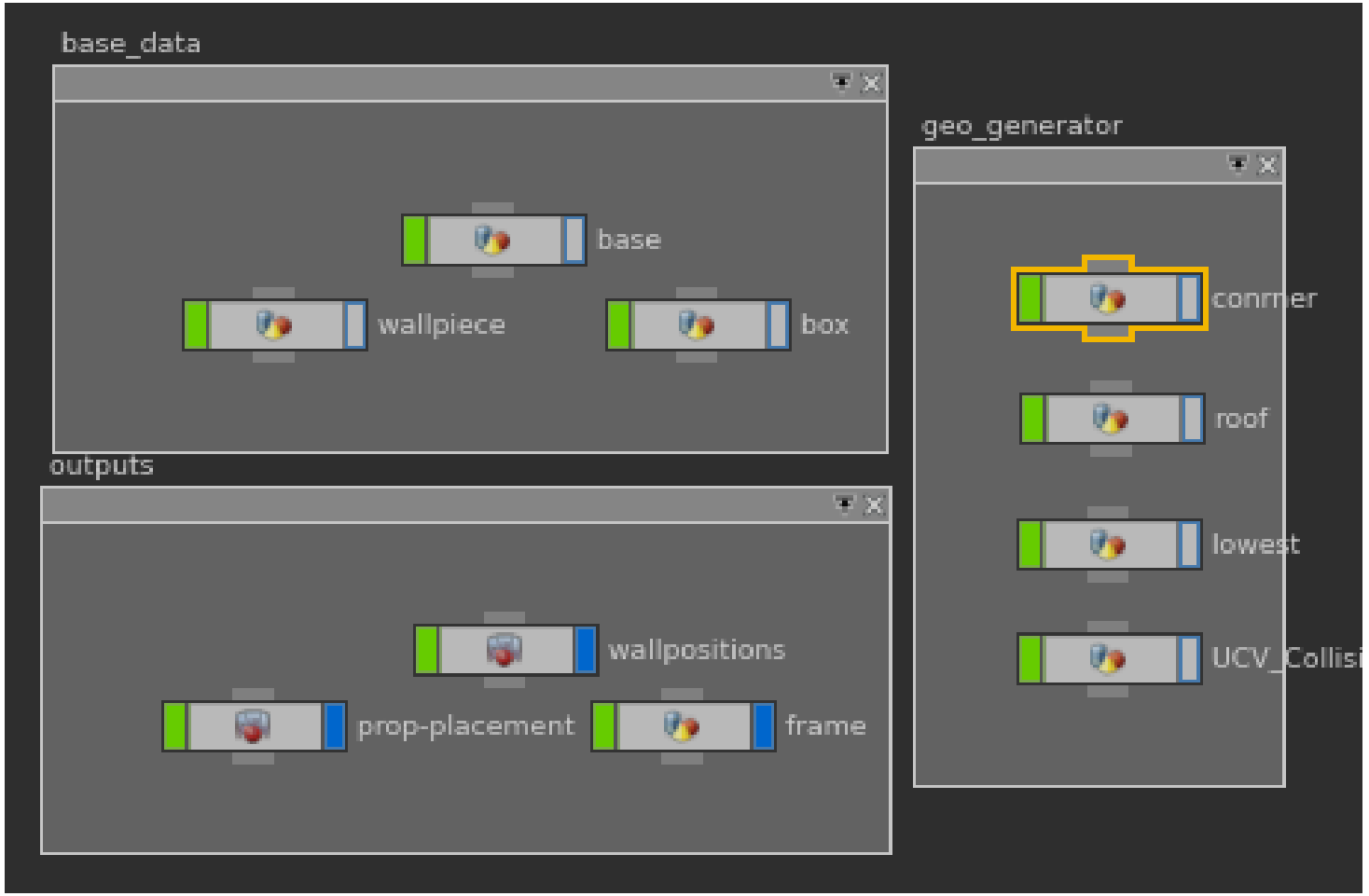
Result



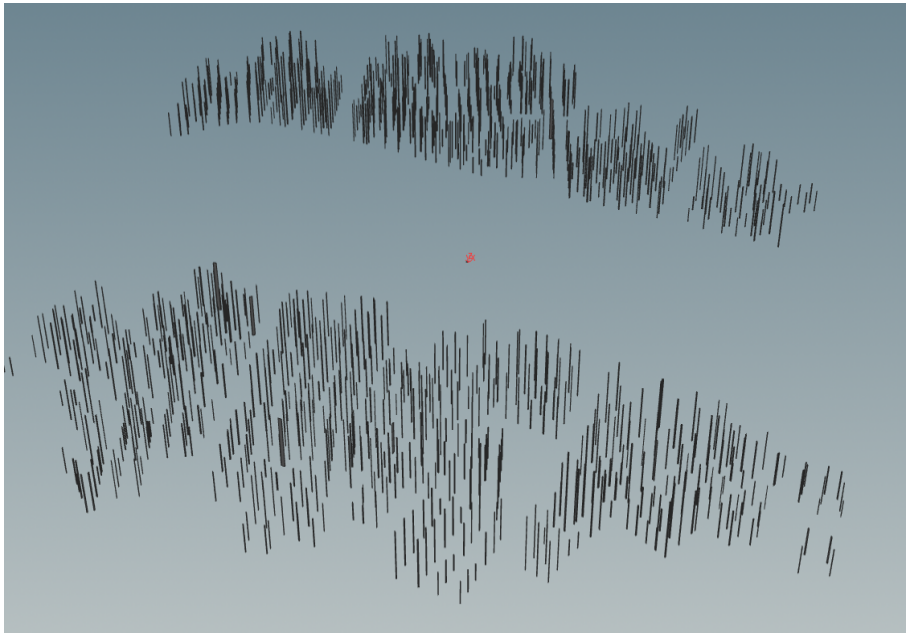


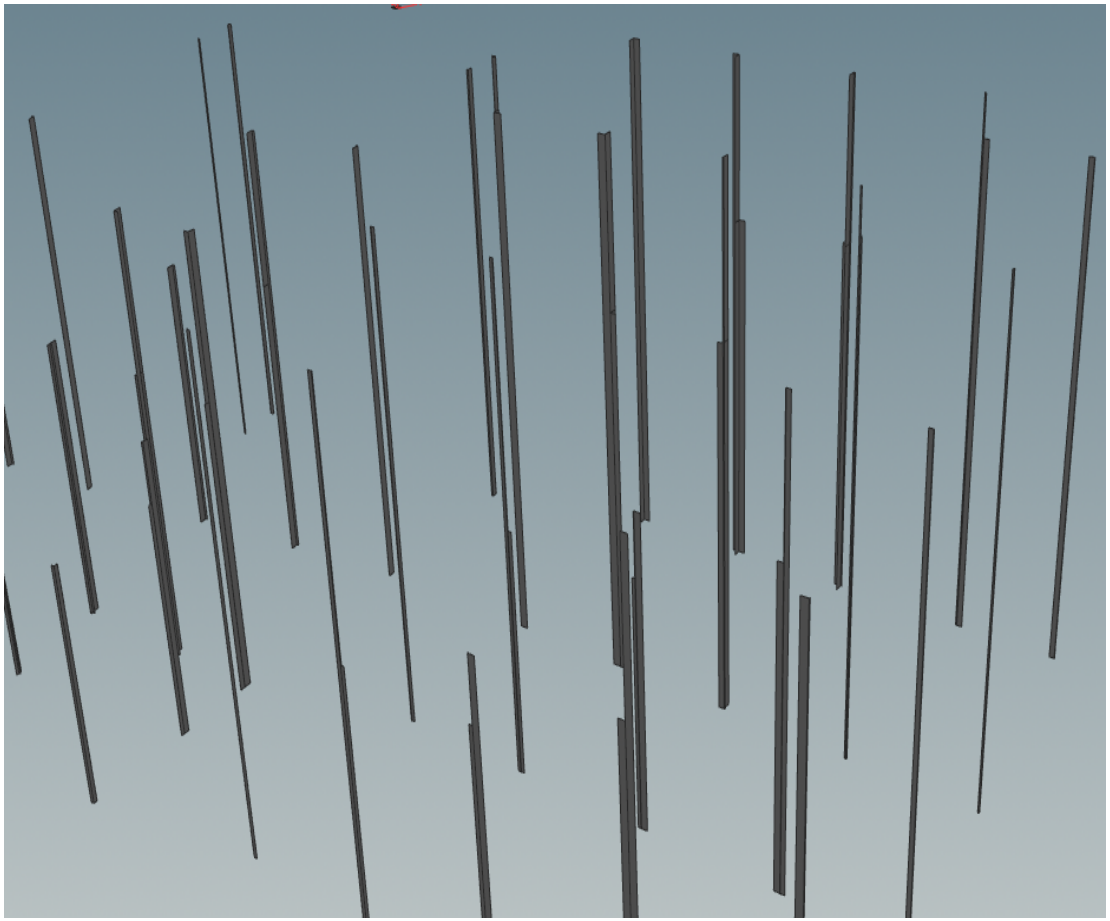
Corners

To close the gaps between the wall pieces. The node takes the inputs of wall positions, so we take the end of the wall corners



Result

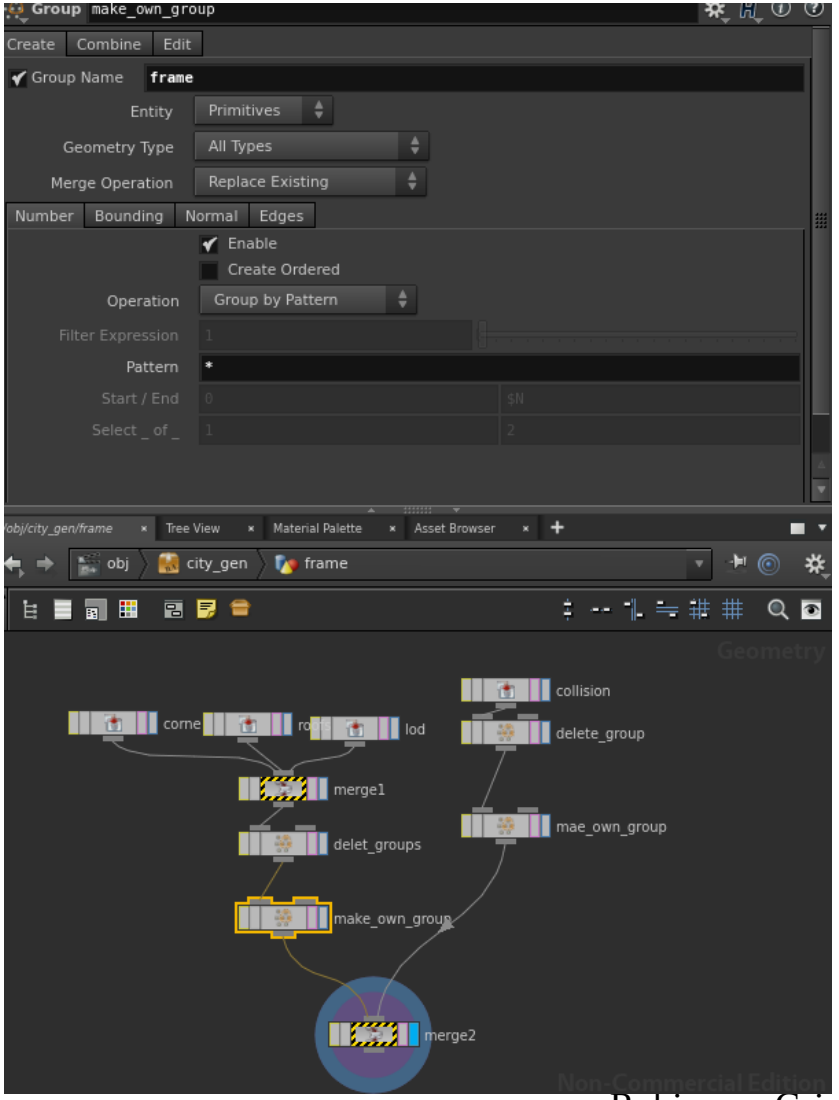




Frames

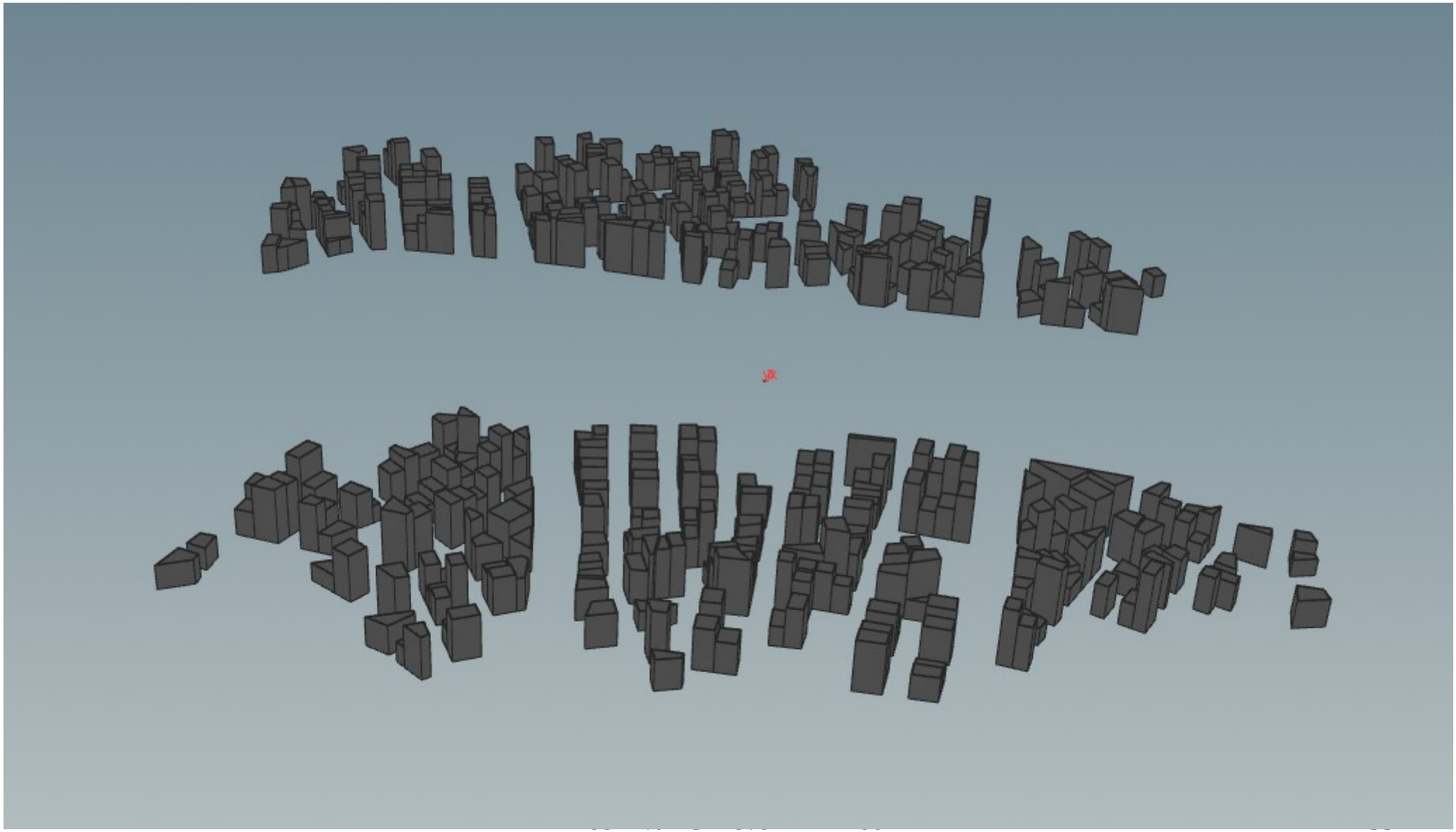
Combines all geometry. Makes them in 2 separate object so Blender has a easier time processing it

Combine



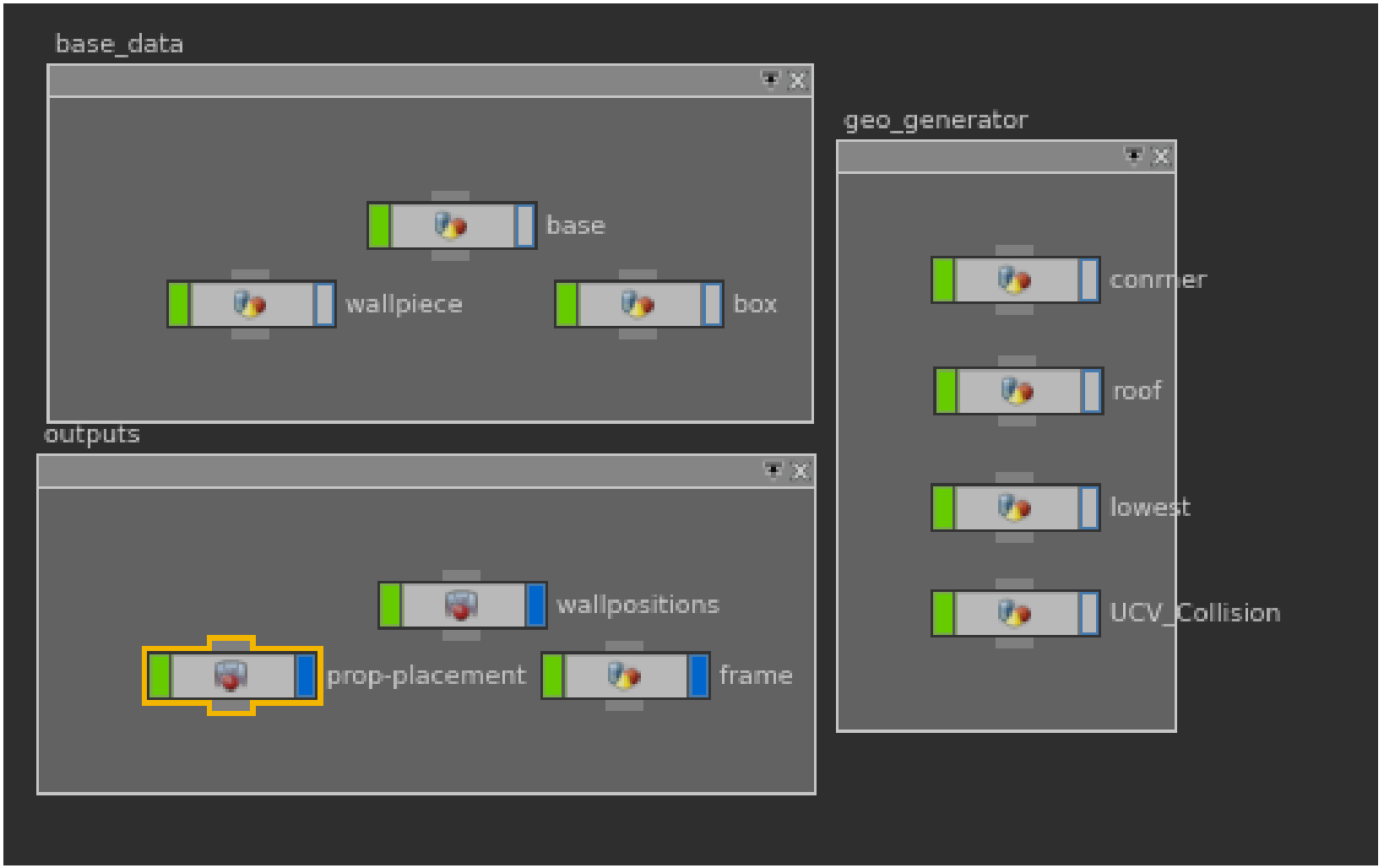
- Here I combine all items so I can easy export them.
- Found out that obj get saved per group selection. This way I could figure out how I can get the right shapes to blender

Result

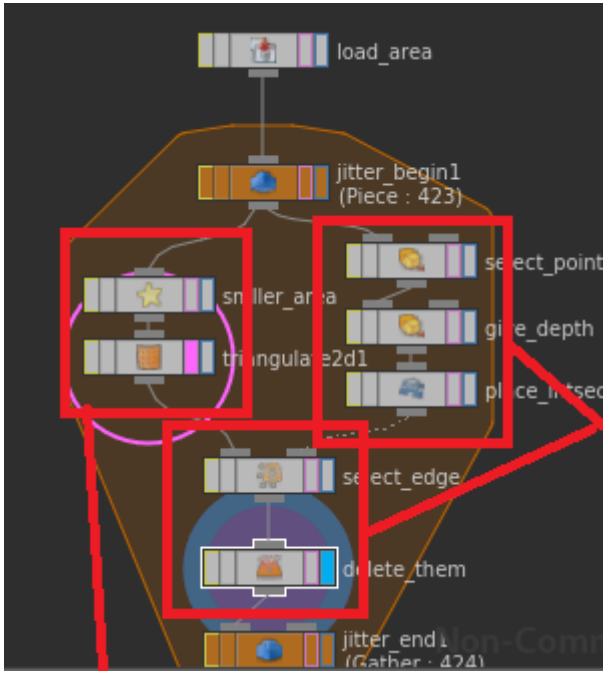


Props placement

a designer was placing props but is was tedious job so I suggested to let me place the roof props procedural.



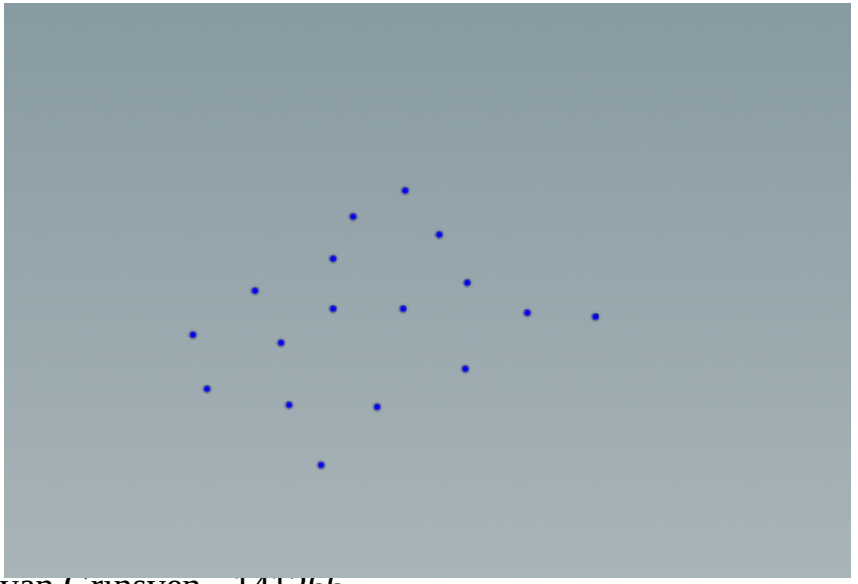
Jitter points



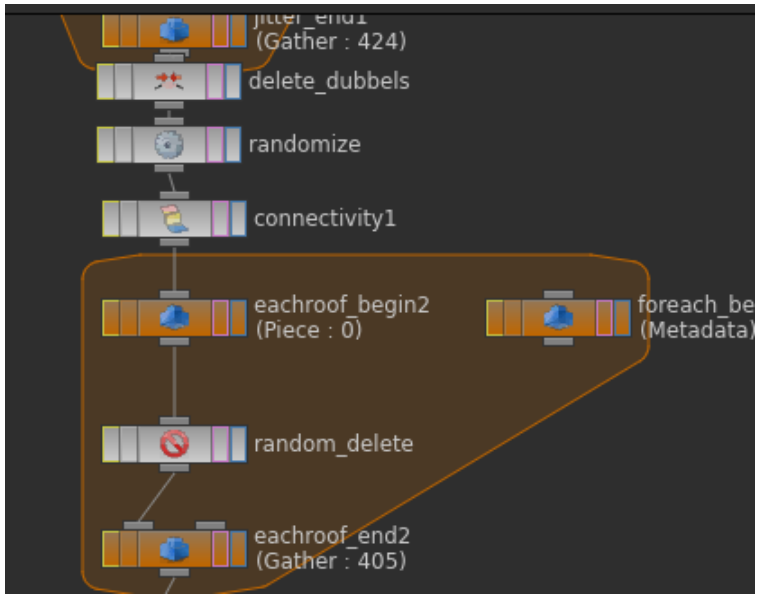
Makes a cage for deleting the edge points to make sure there is no high chance that the objects get placed in the middle

First node is there to make sure that the points are not at the edge of the roof
Second node one generates the pattern

Output

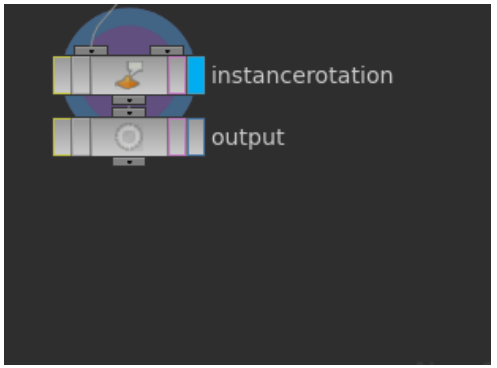


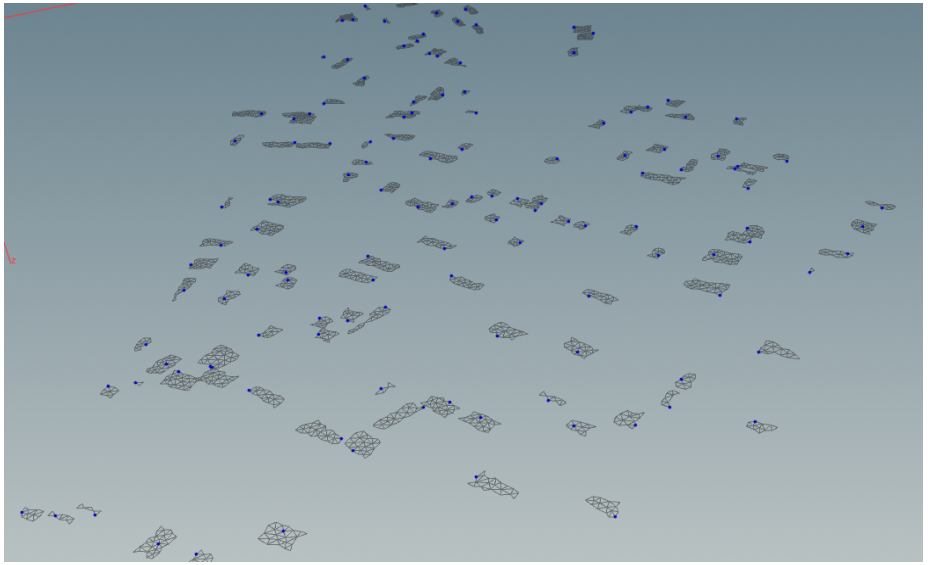
Delete points/add rotation



First node is to make less points. For when points seems to be at the same location.
Second node is for randomize the point numbering
 so I only need to delete the points above a curtain value.
The for loop and connection node are there to loop over the roof tops. And leave a random value between 0 and 2 this way the rooftops do not get overloaded with items

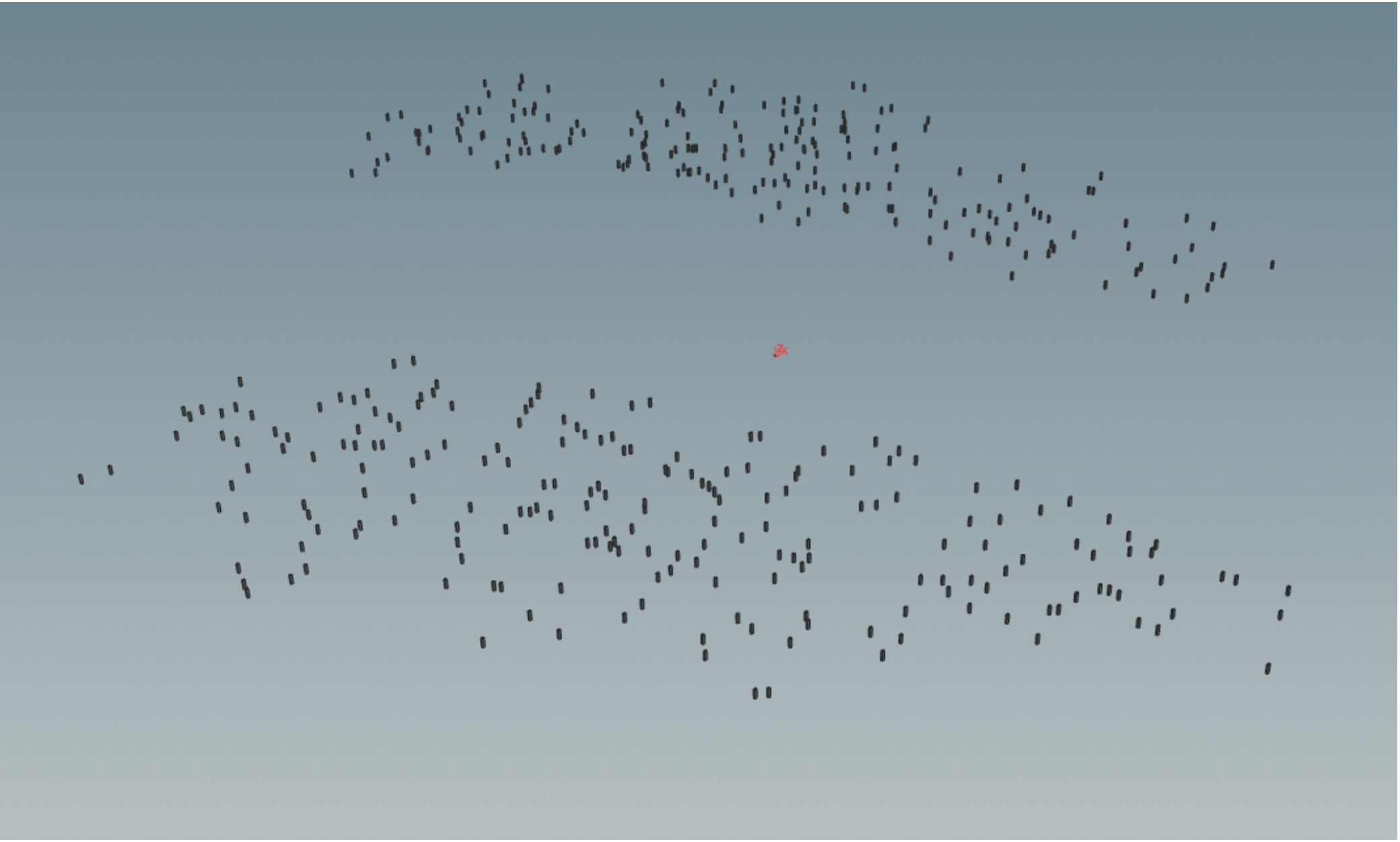
This give extra data to the points by setting the normal value
We use the normal y to find out the identity of the point to find out what instance needs to be placed there
The normal z is used to have a rotation between 0 and 360
We predefine all of this so when other player load the level they do not have a different game then the other





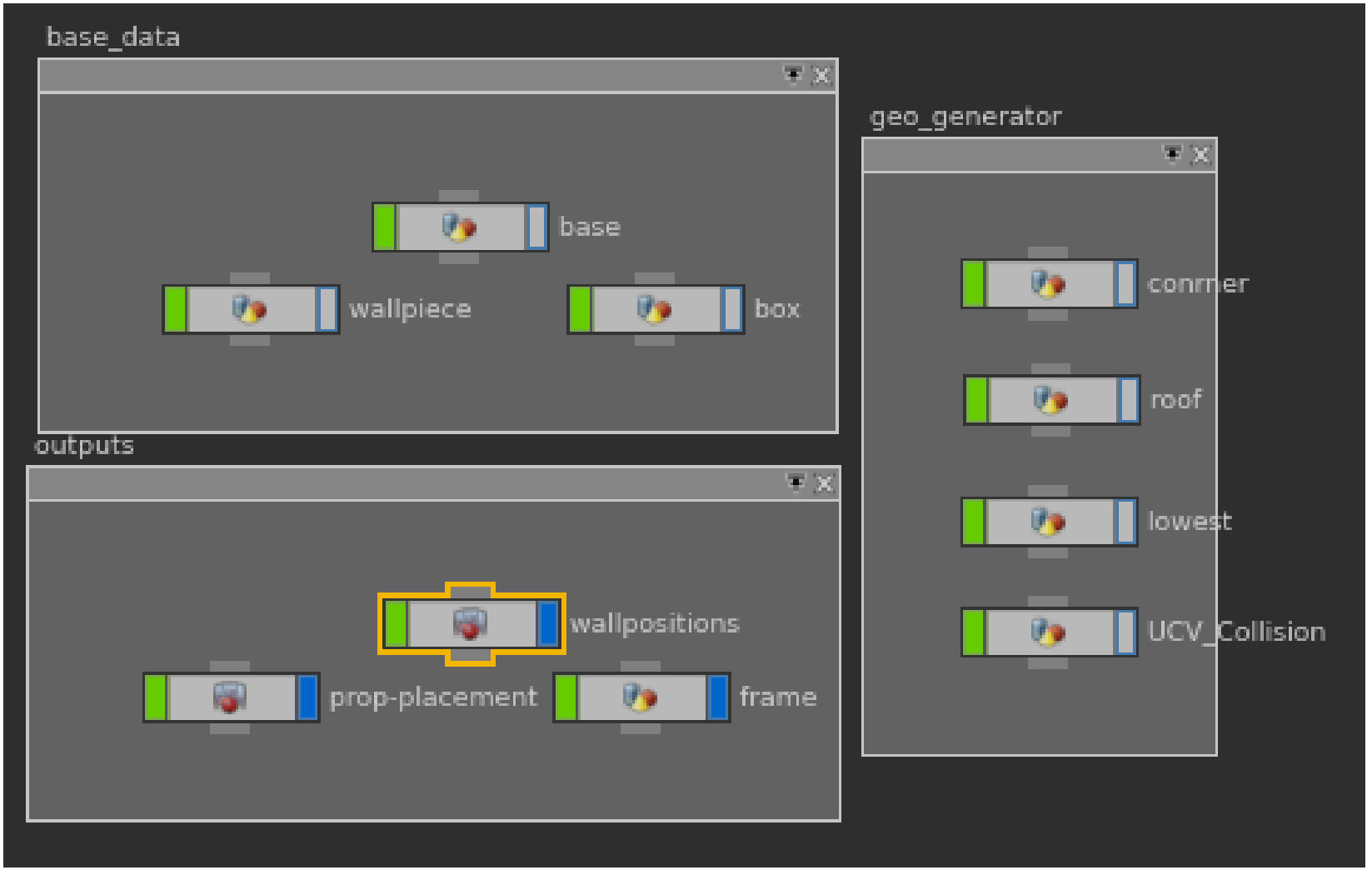
Blue points are existing grey geomerty are layouts on wich they are deleted from

Result

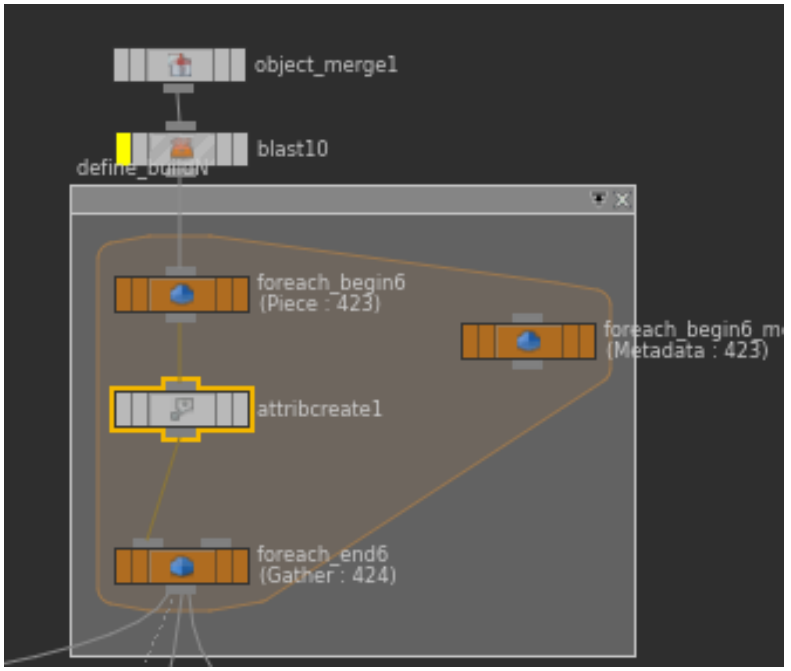


Wall positions

Makes points for where the 3 by 3 wall pieces fit. They are rotated correctly and also the z gives a wall type with it

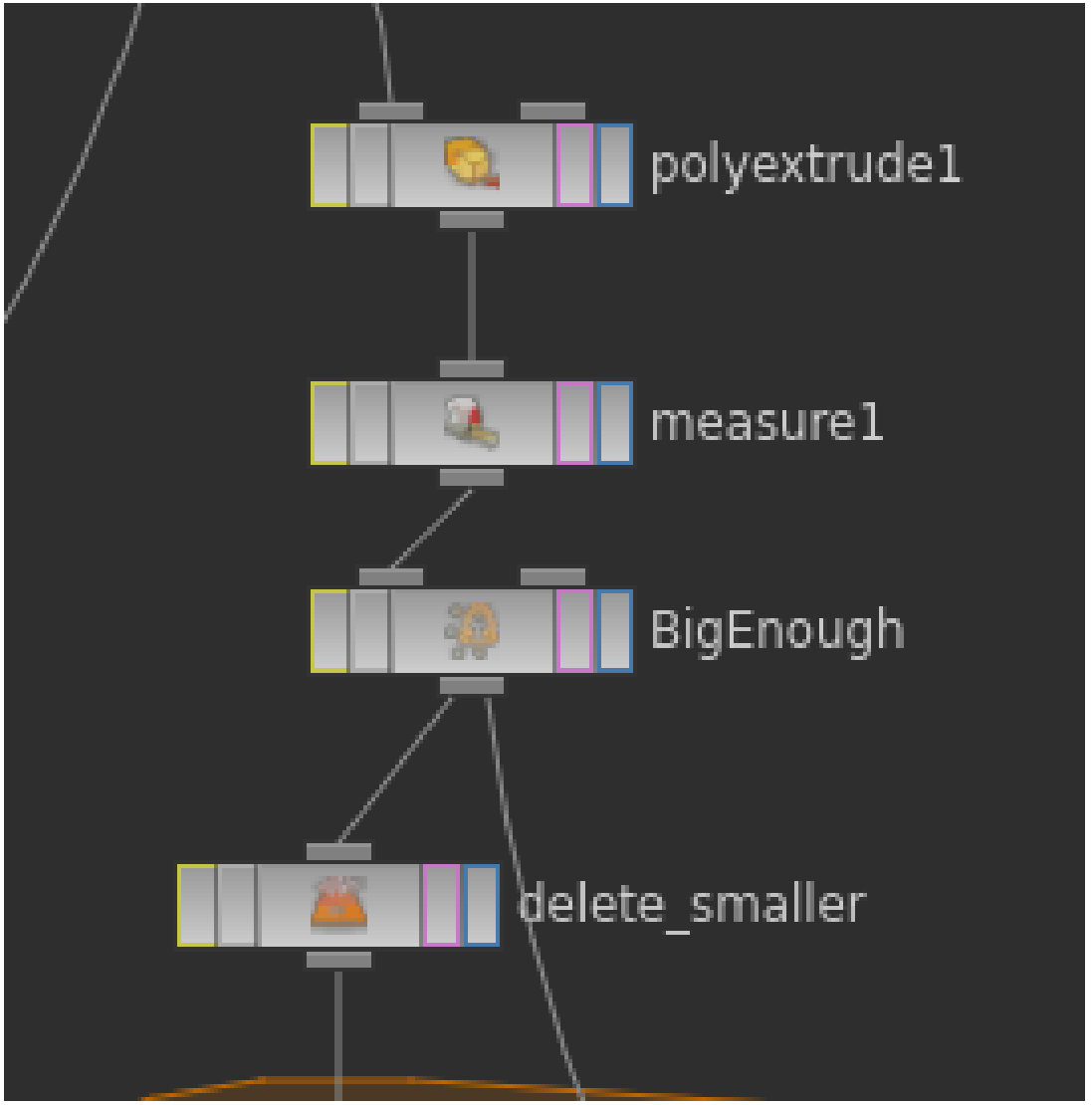


Give building number



- Later in the proces I need to have the building number information in all the points. Therefor I need to assign the primitive number of the faces to the points in the system. This needs to be done in a loop since you can not know what the connection is between the point and face in a simple way because points can be attached to different faces. Unlike vertices.

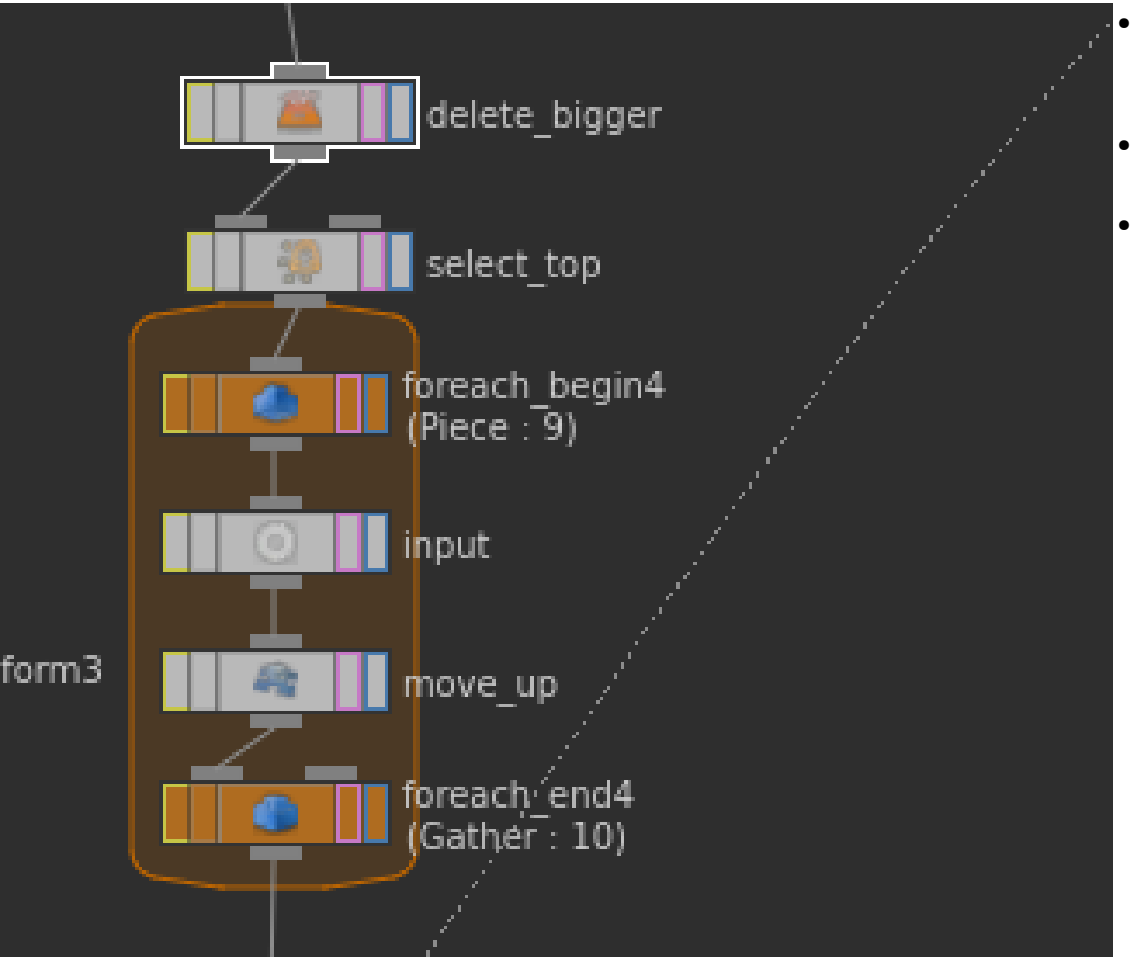
Measuring



We have patterns that would not work under the size 3 in a row.
So in order to find them and fill them we need to find the size of the edges

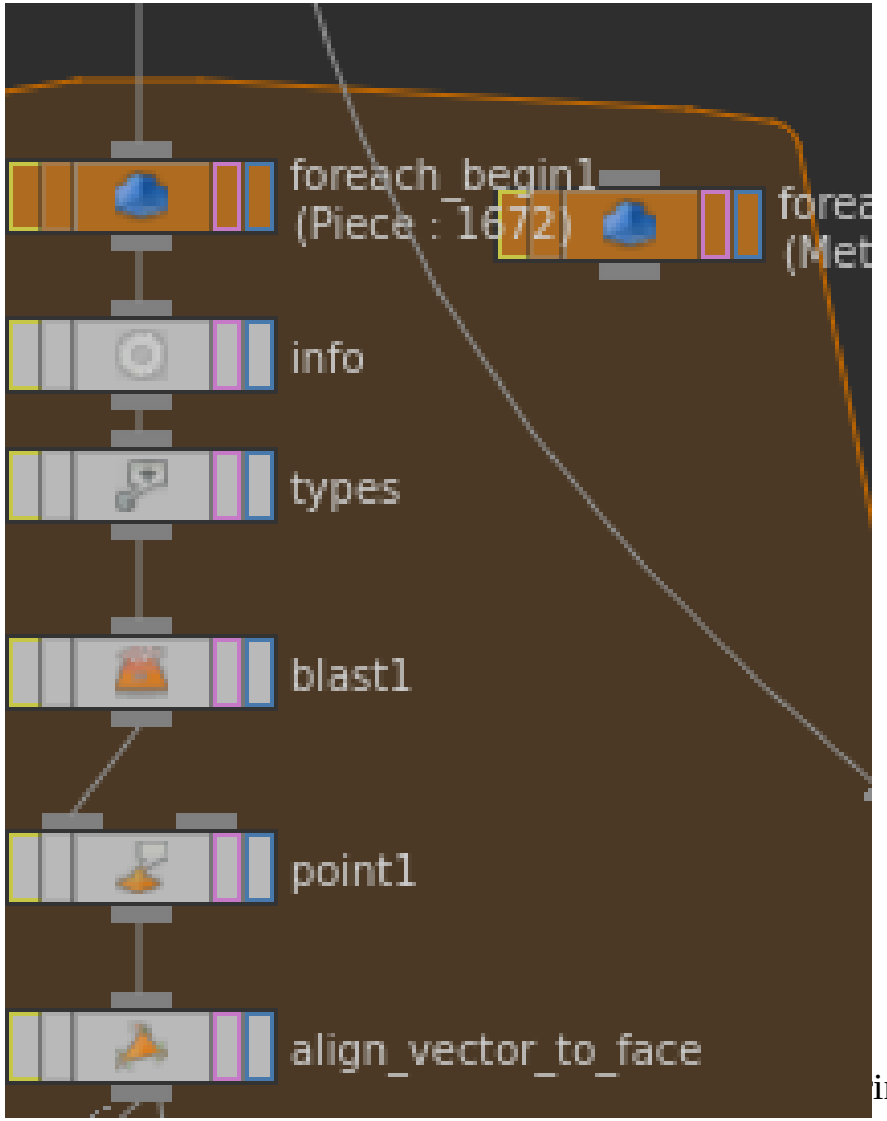
- The first node is there to extrude the faces up by 1
- Then the measure can measure the area of the faces.
  1X width = width
- Then we select the faces that have a area bigger then 3 times the width of a wall piece
- This way we can define if it have to go trough the point wall algorithm or can be replaced by a simple wall
- The last node is there for deleting the faces that are to small.

The empty wall



- This creates walls for places where the line is to short
- First one deletes the walls to long
- The rest make sure that the top points of the face get moved to the set height by the variable set at the beginning

Orientation

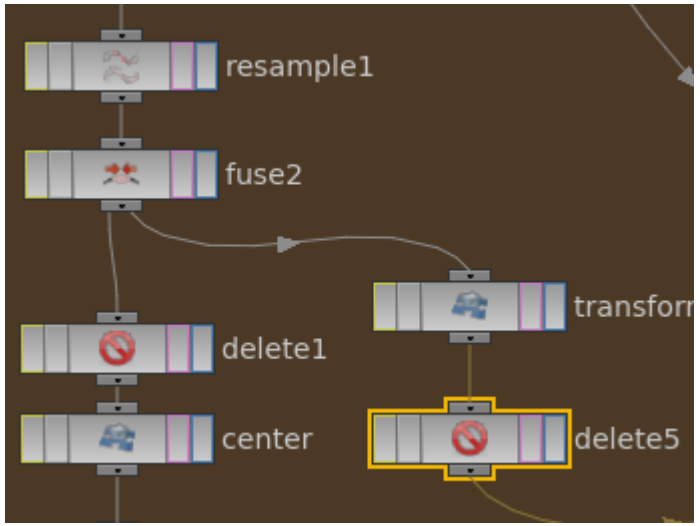


Before I create the points, I need to have them orientated first.

Earlier I picked the normal of the face normal but for some reason it was not stable.
I learned that the normal what the orientation defined by vectors.
If I want to have a solid rotation I need 3 vectors. A up vector a forward vector and a side vector.

I did it as followed: define the forward vector in the point node then made a up vector just by making that vector 0 0 1 since z is up. then cross product them to get the end result

Population

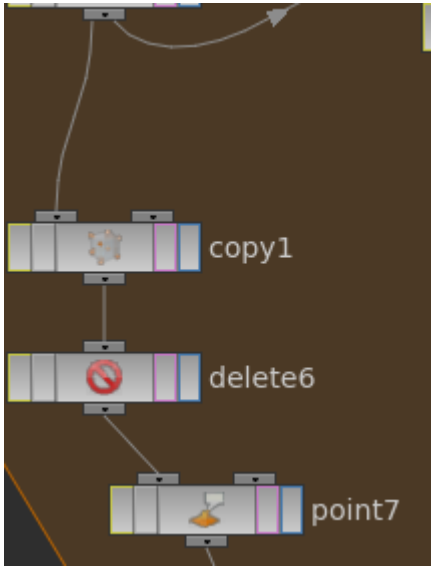


Here I populate the line however since the line is actually a face it loops back and forward what leads to double points. To clean it up I fuse the points on the same spot together.
The length of the total line is now in increments of 3. however take this example:
If the wall length is 27 and so 10 points(because there need to be a end point) get created. if you place the walls next on them you get a length of 10 times 3 what makes 30. so what we need to do is to delete one point and center the line to the old face with: -$CEX + centroid("../blast1",D_X).

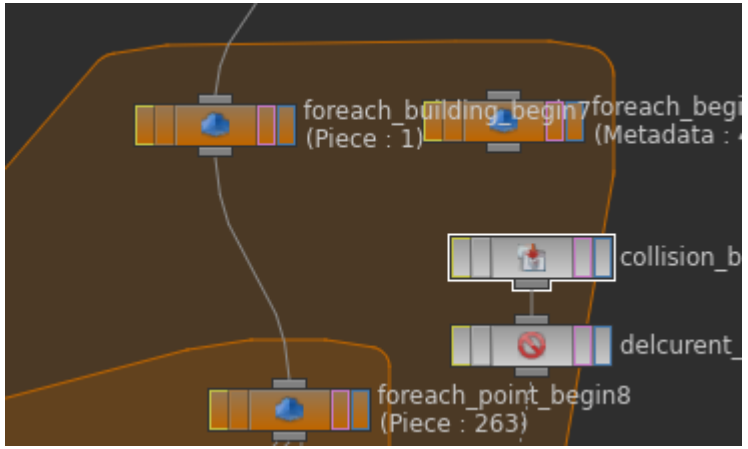
The points always represent 3 meter less than the length of the end result. Since wall pieces are 3 meter and the length and the pieces get placed on the spot there will always be points sticking out

The branch on the right is for the corners. the corners of the line to be centered with one point more since the wall pieces are always 3 meter longer then the point length

The copy node copy's the points with the height difference of the wall piece and the amount of times of how many floors the building has
The point node gives the point data to the point normal z since we do not need it. it is always 0. because it is a vector that does not point upwards
So we use the z value to have a identity of the wall piece in unreal. Because Houdini does export normals in obj but no point attributes.

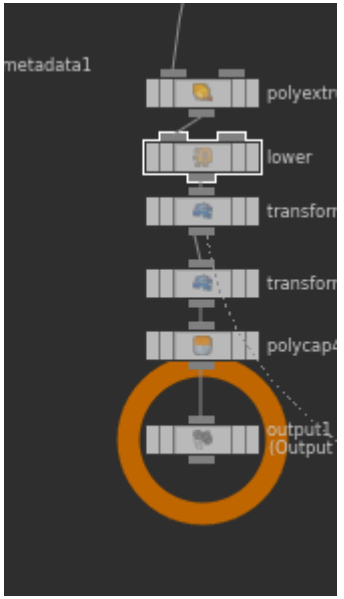


Collision

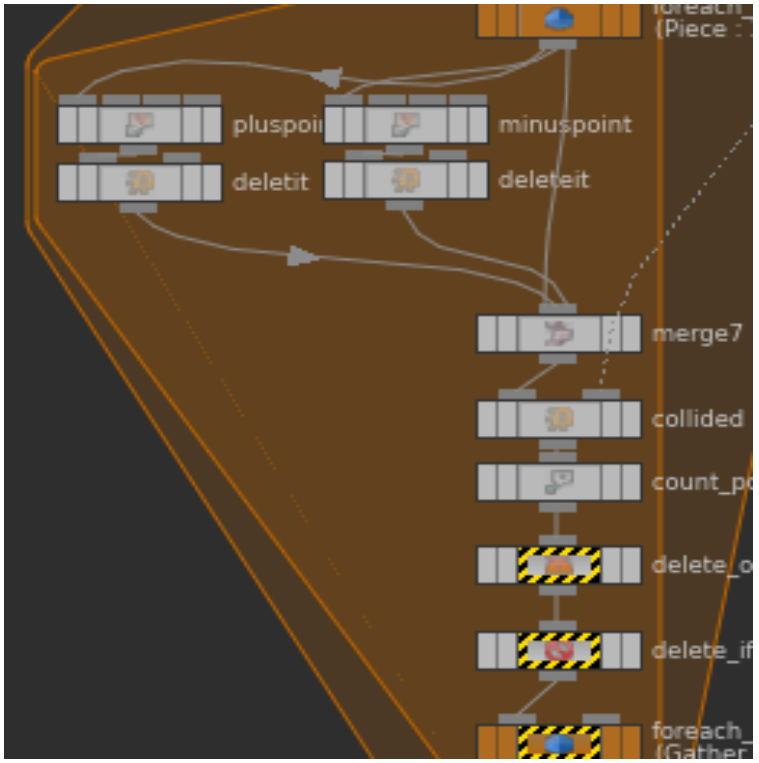


Collision boxes get in the for loop and the for loop delete the building box that are from the same building as the points.
So this way the collision is only between the points and the other buildings.

This create the boxes for deleting points for this the extrusion needs to happen again I move the boxes down
So they do not delete points that are to high also moved the ground down so it is more likely to hit the point at the bottom

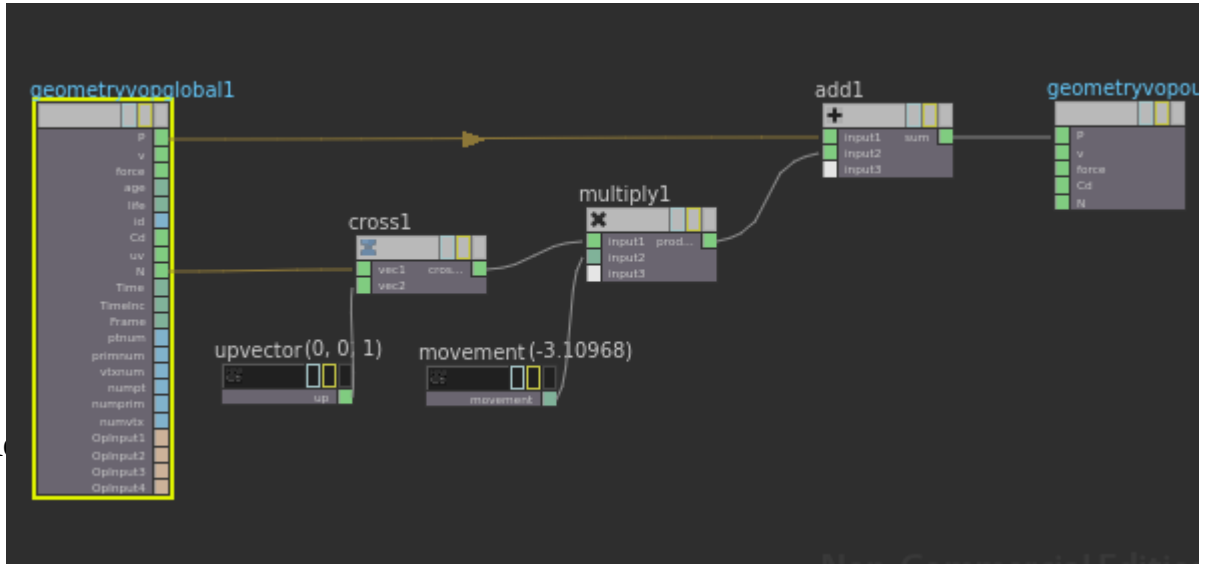


The point deletion

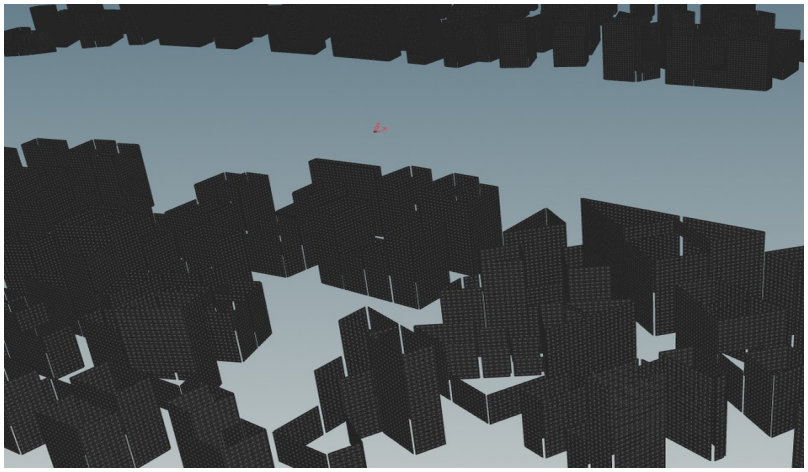


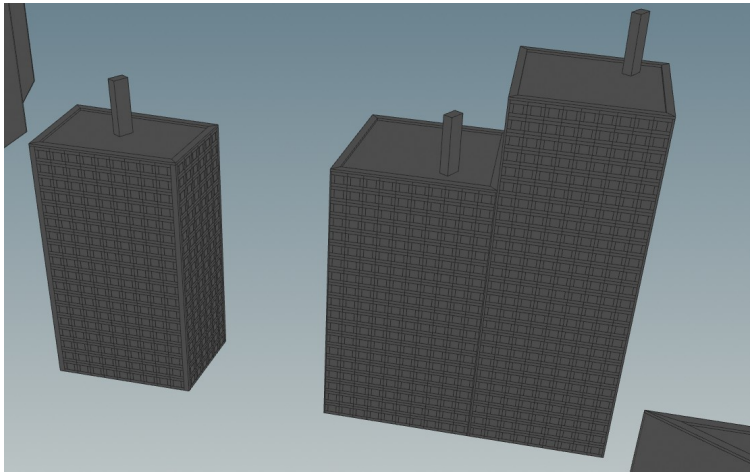
In order to find out if we need to delete the point in the loop. We need to make 2 more points that represent the bounding box for the wall pieces. Because when we did not do that we found gapes in non convex places since the point was not out the box but the side of the wall was supposed to be visible.
The top two nodes create the points on the outside. Then they get there own group what makes them easy to delete, I merge it I check via bounding box. Then I count how many of the 3 points are selected. And when the 3 points are selected then delete them

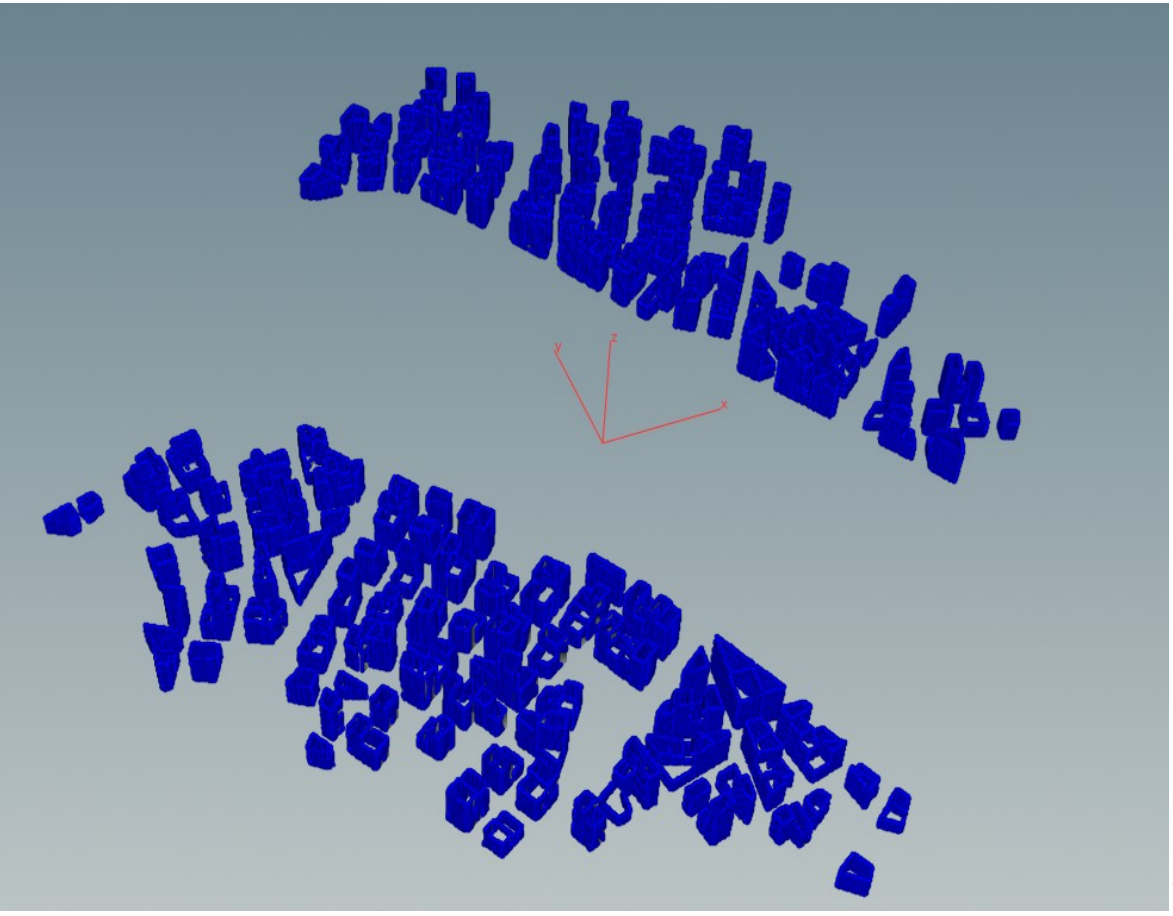
Here I move the points on there own x axis.
So I can find out where the point would be

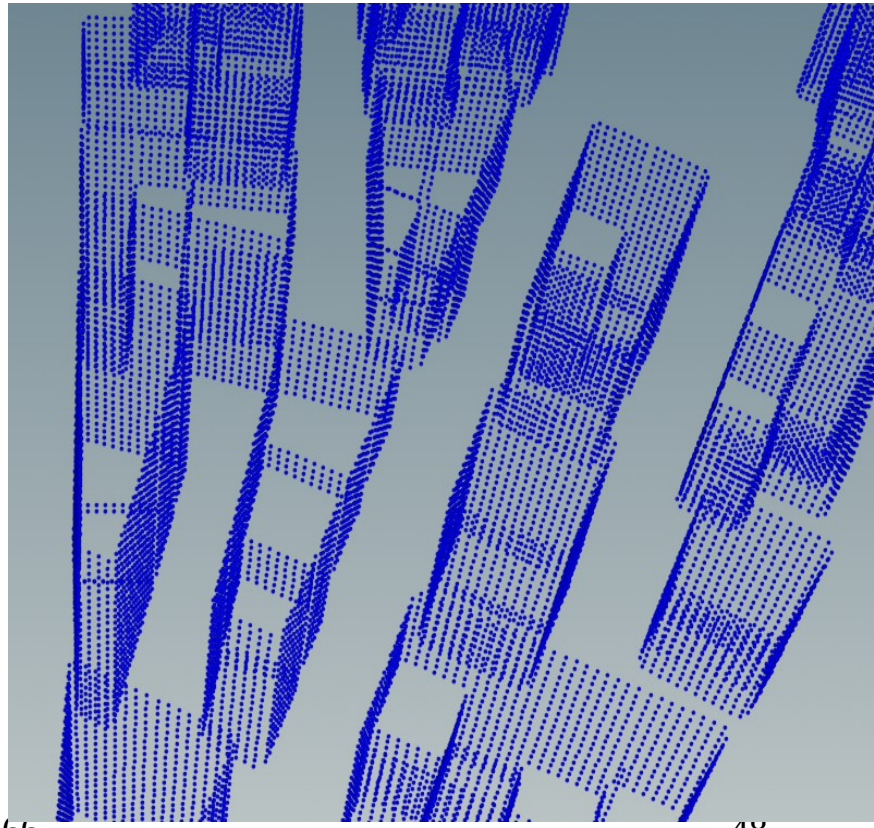


Result









Python

Python in Houdini is not used in the project but I did follow a lecture where a script was given and explained it had explained 3 concepts to use it:

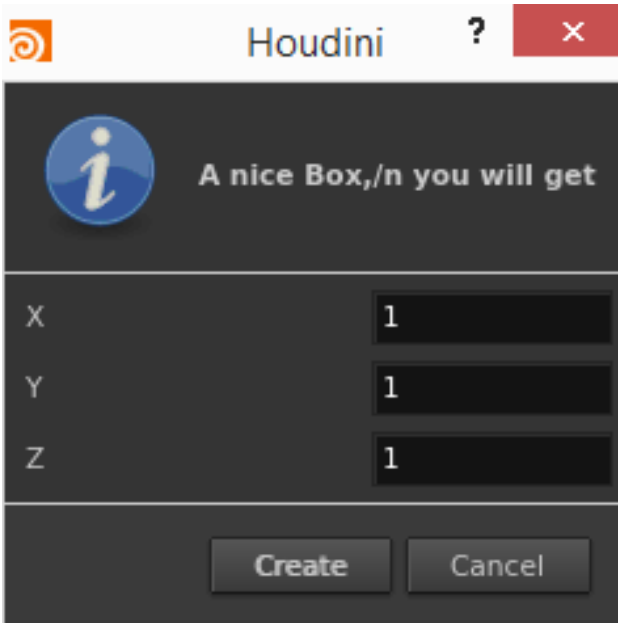
- One was automate actions: make a node in Houdini.(see script 1)
- Second one was fairly similar But it was running outside of Houdini. What was handy for faster proccesing(see script 2)
- Third option was using python expressions in Houdini. This offers more flexibility however to place a action in it took more space.

The lecture was given at the end of the block. So I did not risk the chance of making the system again or it would be unstable.

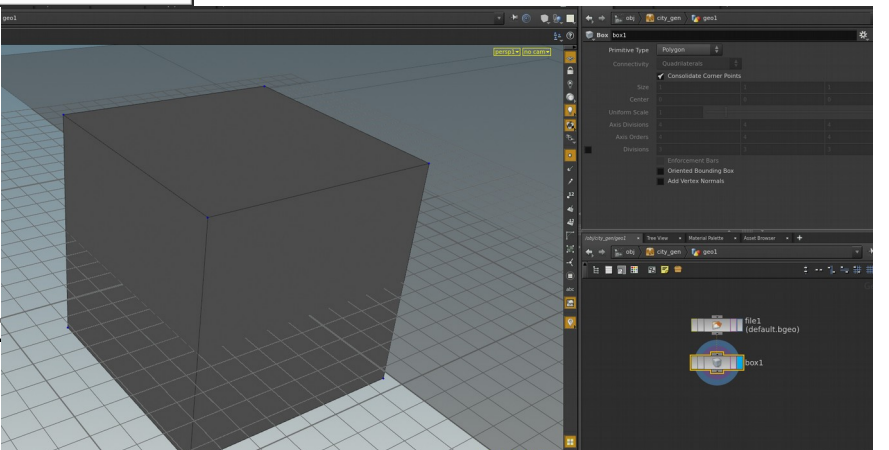
It did show me when python is good for usages. If I would apply python in the system it would be in the sense of the second solution. I would run Houdini with the python script. To run everything in a cmd command.

Script 1

Pop-up



result



Script

Edit tool: firstScript

Save To C:/Users/Robin/Documents/houdini15.0/toolbar\default.shelf

Original tool file: $HOUDINI_USER_PREF_DIR/toolbar\default.shelf.

Options | Script | Help | Context | Hotkeys

```
import hou
nodePanel = hou.ui.paneTabOfType(hou.paneTabType.NetworkEditor)
selected = hou.selectedNodes()
nodeType= "box"

def spawnNode(selected):
    dimentions = hou.ui.readMultiInput("A nice Box,/n you will get",["X","Y","Z"], buttons = ("Create","Cancel"),initial_contents = ('1','1','1'))
    if not dimentions[0]:
        newNode = nodePanel.pwd().createNode(nodeType)
        if not len(selected):
            position = (0,0)
        else:
            position = (selected[-1].position())
            position[1] -= 1
            newNode.setFirstInput(selected[-1])
        newNode.setPosition(position)
        newNode.parm("sizex").set(dimentions[1][0])
        newNode.parm("sizey").set(dimentions[1][1])
        newNode.parm("sizez").set(dimentions[1][2])
        newNode.setCurrent(True)
        newNode.setSelected(True)
        newNode.setName("One_nice_box",True)
        selected = newNode
    else:
        hou.ui.displayMessage("Stop, you must")

if nodePanel.pwd()!= hou.node("obj") and nodePanel.pwd().parent() == hou.node('obj'):
    spawnNode(selected)
else:
    hou.ui.displayMessage("in the geo node, you must be")
```

Ln 1, Col 1

Script Language Python

Apply | Accept | Cancel

Script 2

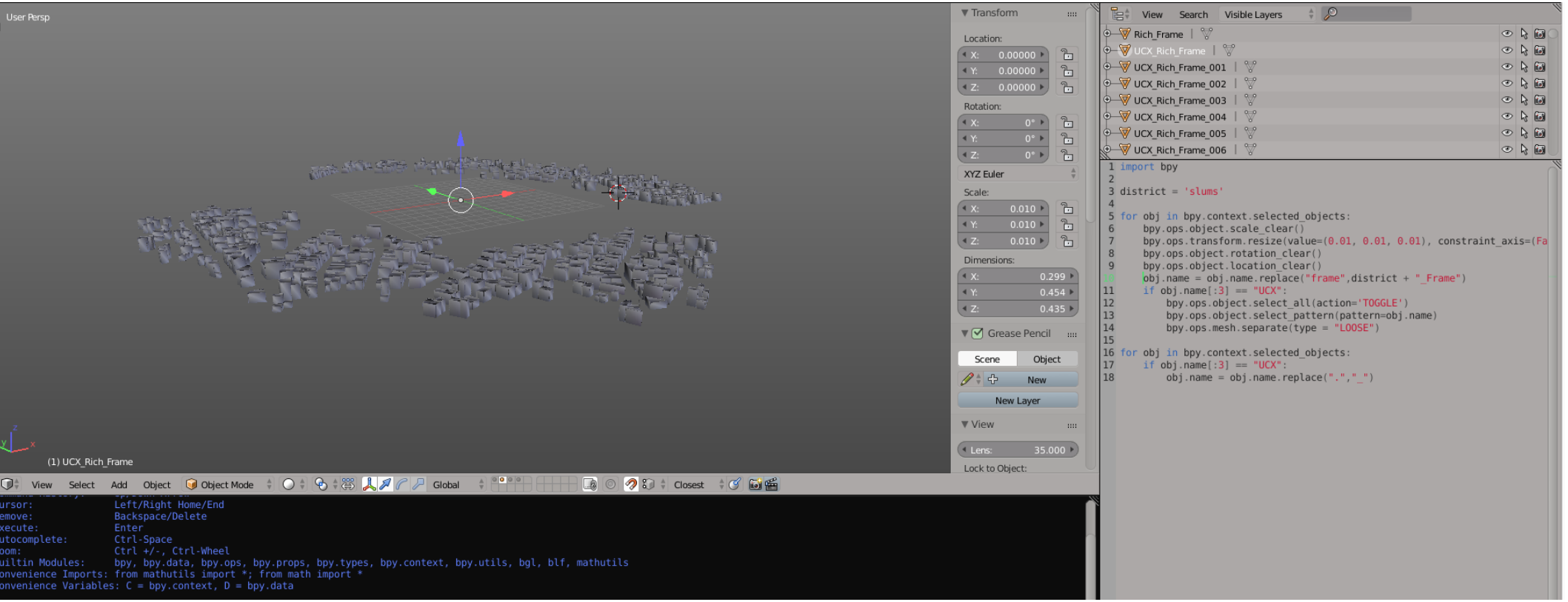
```
import os, sys

os.environ['HFS'] ='C:/PROGRA~1/SIDEEF~1/HOUDIN~2.459'
os.environ['PATH'] ='C:/PROGRA~1/SIDEEF~1/HOUDIN~2.459/bin;'
sys.path.append(os.environ['HFS'] + "/houdini/python%d.%dlibs" %sys.version_info[:2])

import hou
import subprocess


def spawnNode():
    geo=hou.node("/obj").createNode("geo")
    newNode = geo.createNode("box")
    obj=hou.node("obj/geo1/file1")
    position = (obj.position())
    position[1] -= 1
    newNode.setFirstInput(obj)
    newNode.setPosition(position)
    newNode.parm("sizex").set(2)
    newNode.parm("sizey").set(2)
    newNode.parm("sizez").set(2)
    newNode.setName("a_box_you_made",True)

#def printTree(node):
    #for

hou.hipFile.save(file_name="C:/Python34/worldHello.hipnc")
hou.hipFile.load(file_name="C:/Python34/worldHello.hipnc")
spawnNode()
hou.hipFile.save(file_name="C:/Python34/worldHello.hipnc")

subprocess.Popen(['houdinifx','worldhello.hipnc'])
```

This should create a file in the python folder with a box in the scene

The post processing of the data

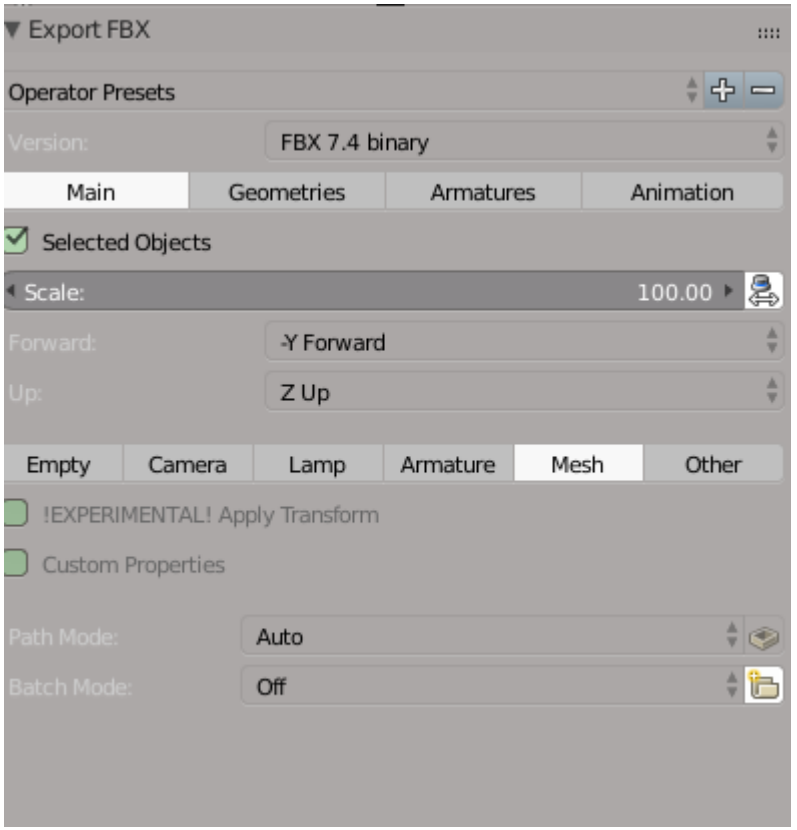
- Since we do not have the Houdini engine we where looking for other ways. One of them was exporting everything to obj. it supported point position and normal. we used the normal to create data for the game. What hold rotation and object identity. We used the identity for the rotations that are always 1 constant value since we can just hard put in those values in unreal.
- For the instances we needed a pointcloud. And that needed to be a list in unreal. So unreal can read it out and define the position rotation and what object should be placed.
  - For this we exported the obj and converted it to binary I am not going to cover this since I did not make it
- A issue that occurred when importing the frame. Was that the collision boxes where meshes and every face had it own material slot. What means when you import it and create material is checked on it needs to compile thousand of shaders.
  - For this issue I made a blender script which I am going to cover. This script automate the task you need to do in blender to export it again. This helps for people who are not experts in blender and still can export it.

Blender script



- Takes as input a geometry and its collision. And make all the collisions separated meshes

The export settings



- Blender scale is smaller. Therefore to work with it it need 100 times smaller in the blender scene. Then the export need to be 100 times bigger again.
- The export orientation. Should be z up and -y forward because that is the settings of unreal. By using this you can work in blender orientation in blender and export it with the right rotation afterward
- Blender export is better than the one from Houdini in this instance since you have more control. And the material is only one. Because it has a different algorithm.

Game result


ENEMIES VS ALLIES
SHIELD 100
HEALTH 100


ENEMIES VS ALLIES
Dev build - [0.36.4]


ENEMIES VS ALLIES
Dev build -
SHIELD 100
HEALTH 100


ENEMIES VS ALLIES
Dev build - [0.36.4]
SHIELD 100
HEALTH 100