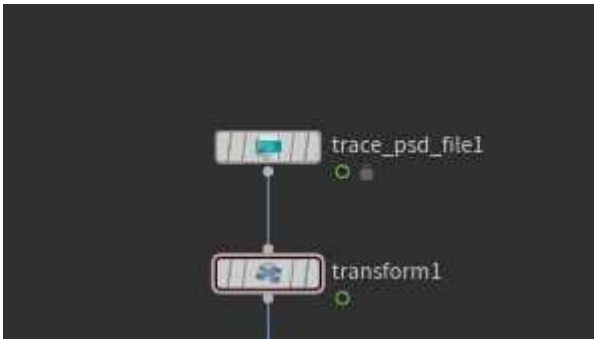
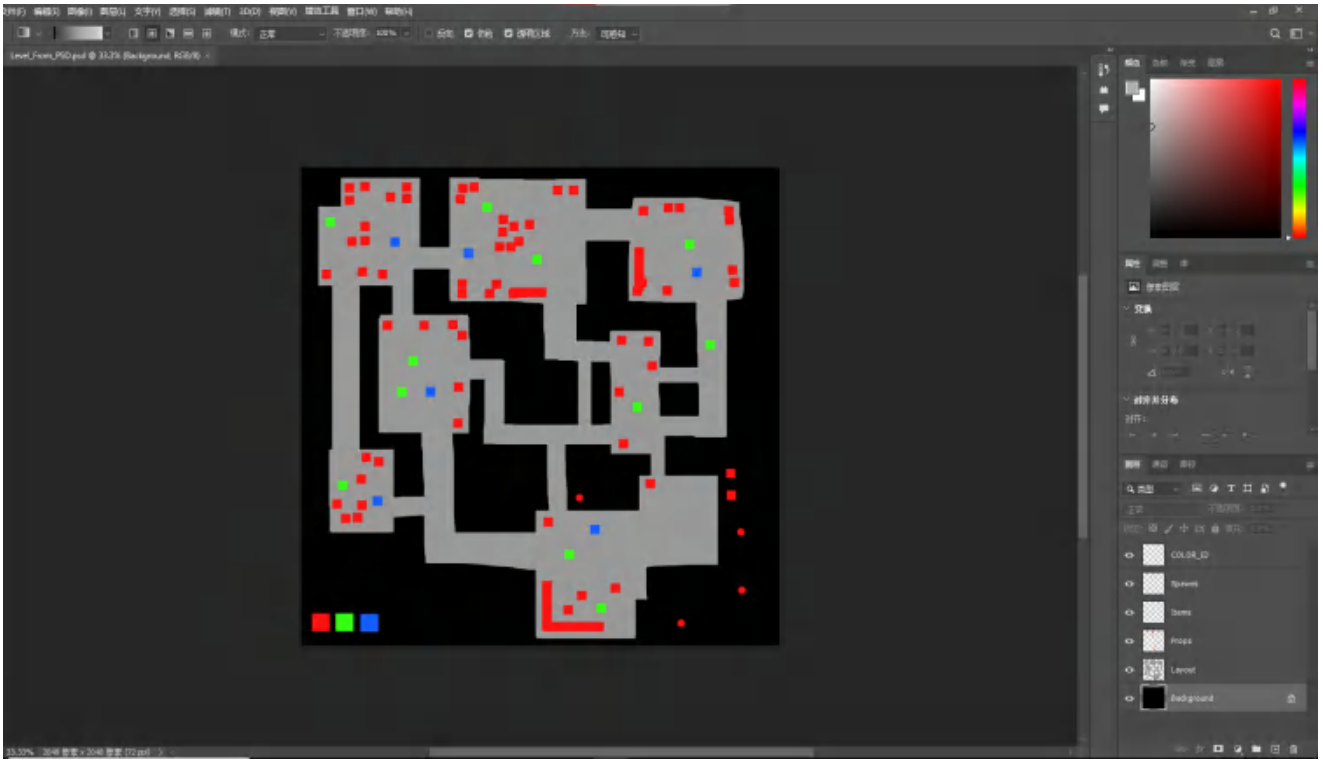
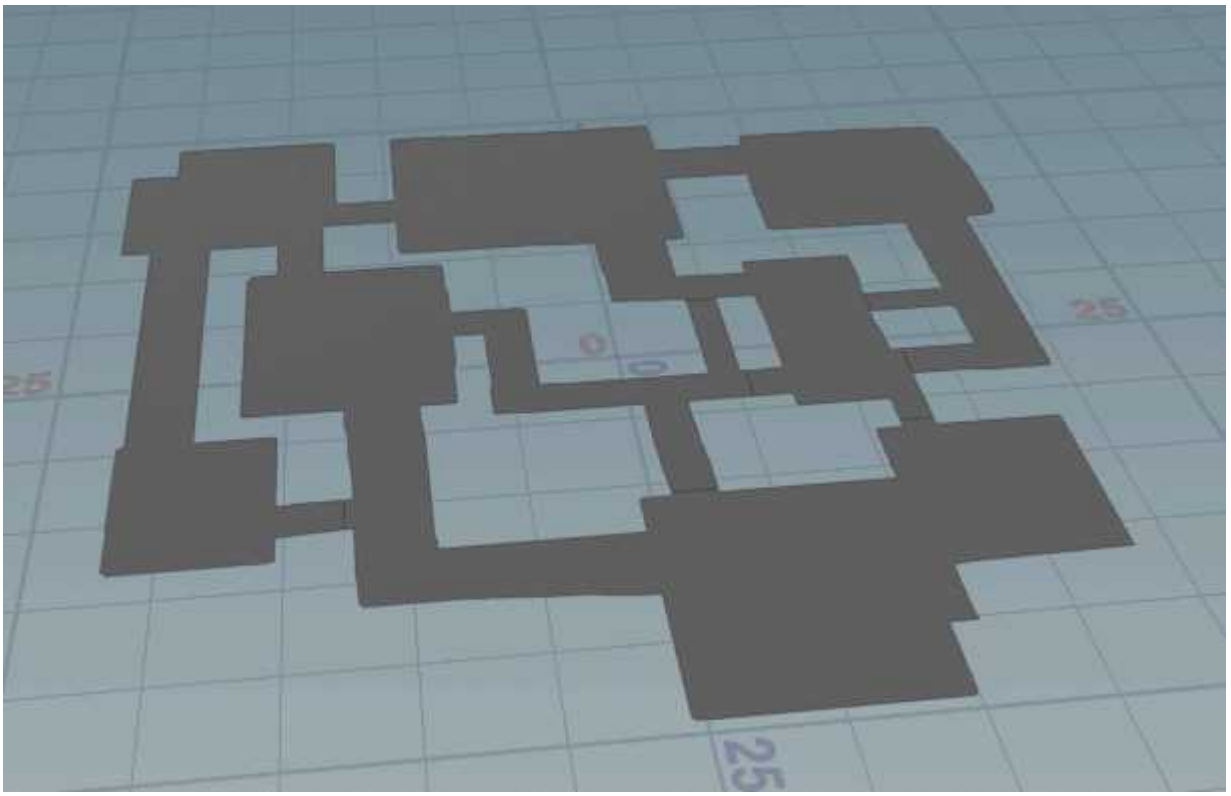
# PSD level



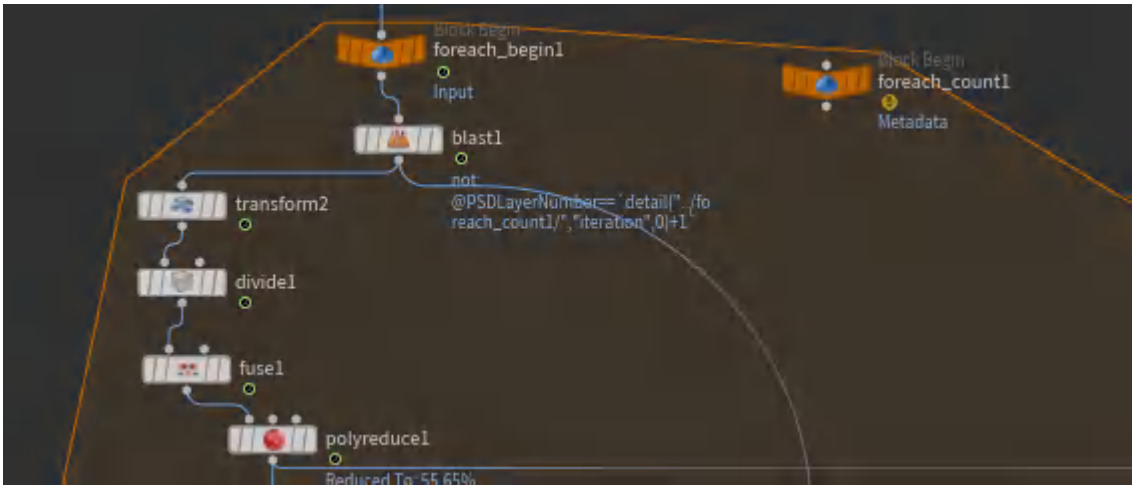

用trace psd file节点读取psd文件



在psd里面把layer分好，这边我是直接用了simon老师提供的psd，但可以自己画一个也可以根据simon老师的这个改，分好layer就行。



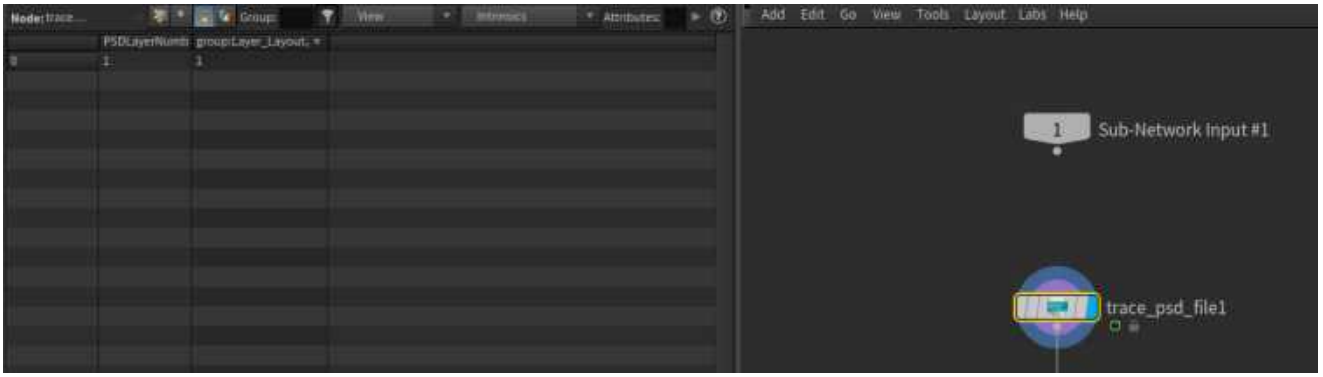
transform节点就是拿来放大这个layout的。



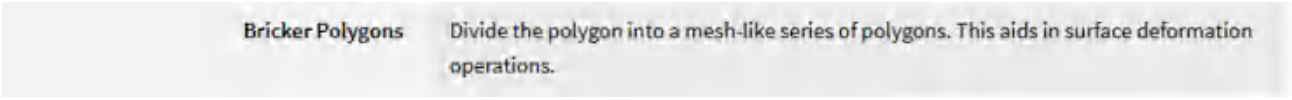

遍历多层psd 导入进来的layer。
利用blast节点来提取layer。

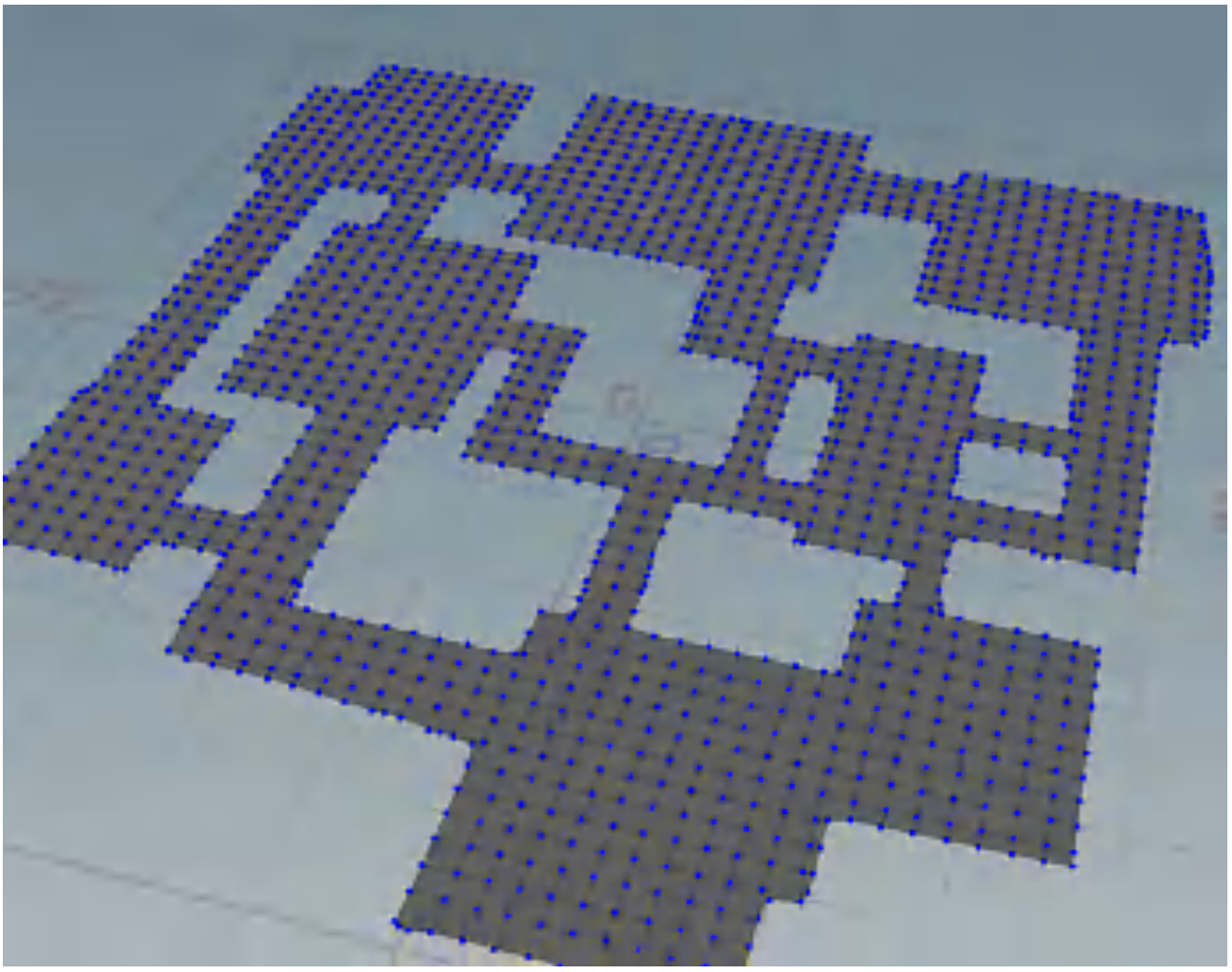
```
@PSDLayerNumber==`detail("../foreach_count1/","iteration",0)+1`
```

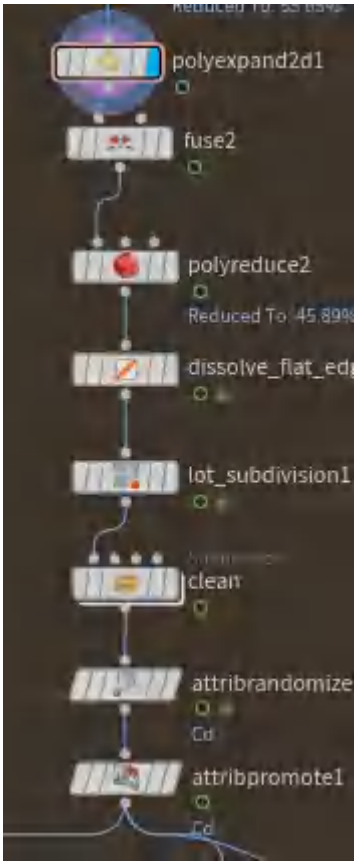
按layer进行逐层处理，每次只保留一层参与后续操作。
这边的PSDLayerNumber就是 trace节点自己内部的一个变量，在geometry hseet可以看到。



用divide，fuse，polyreduce节点清理模型以及网格化模型（利用divide节点里面的Bricker Polygons）

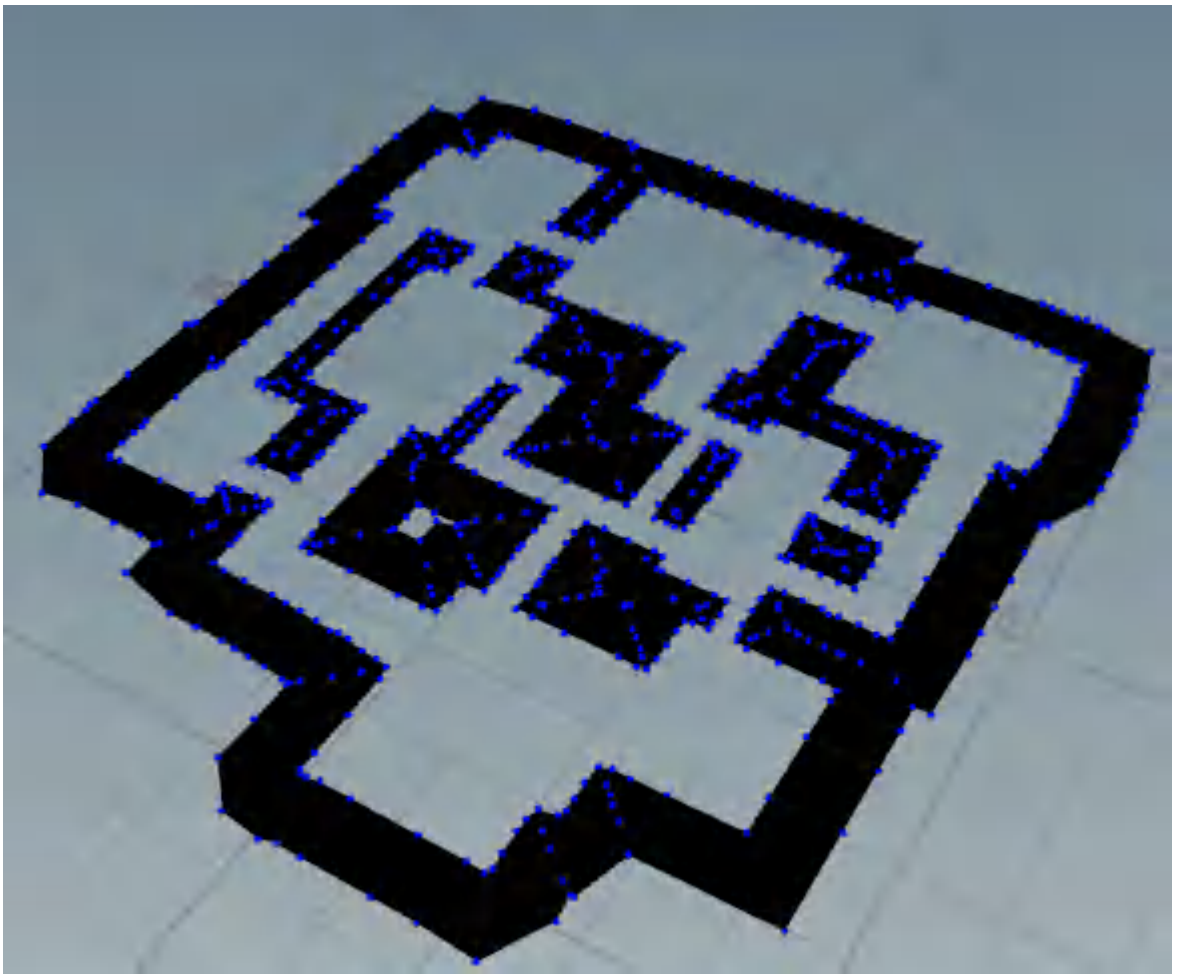


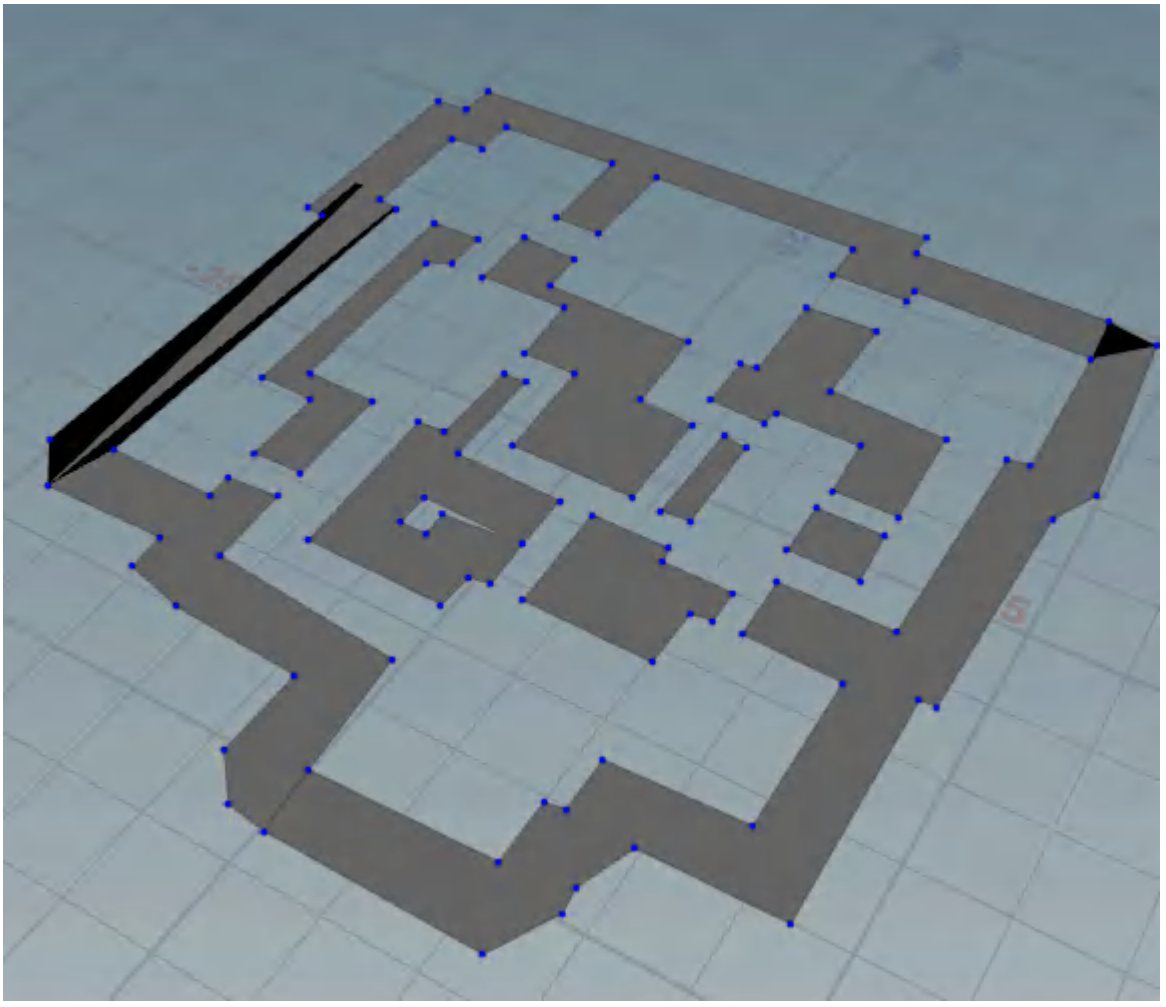


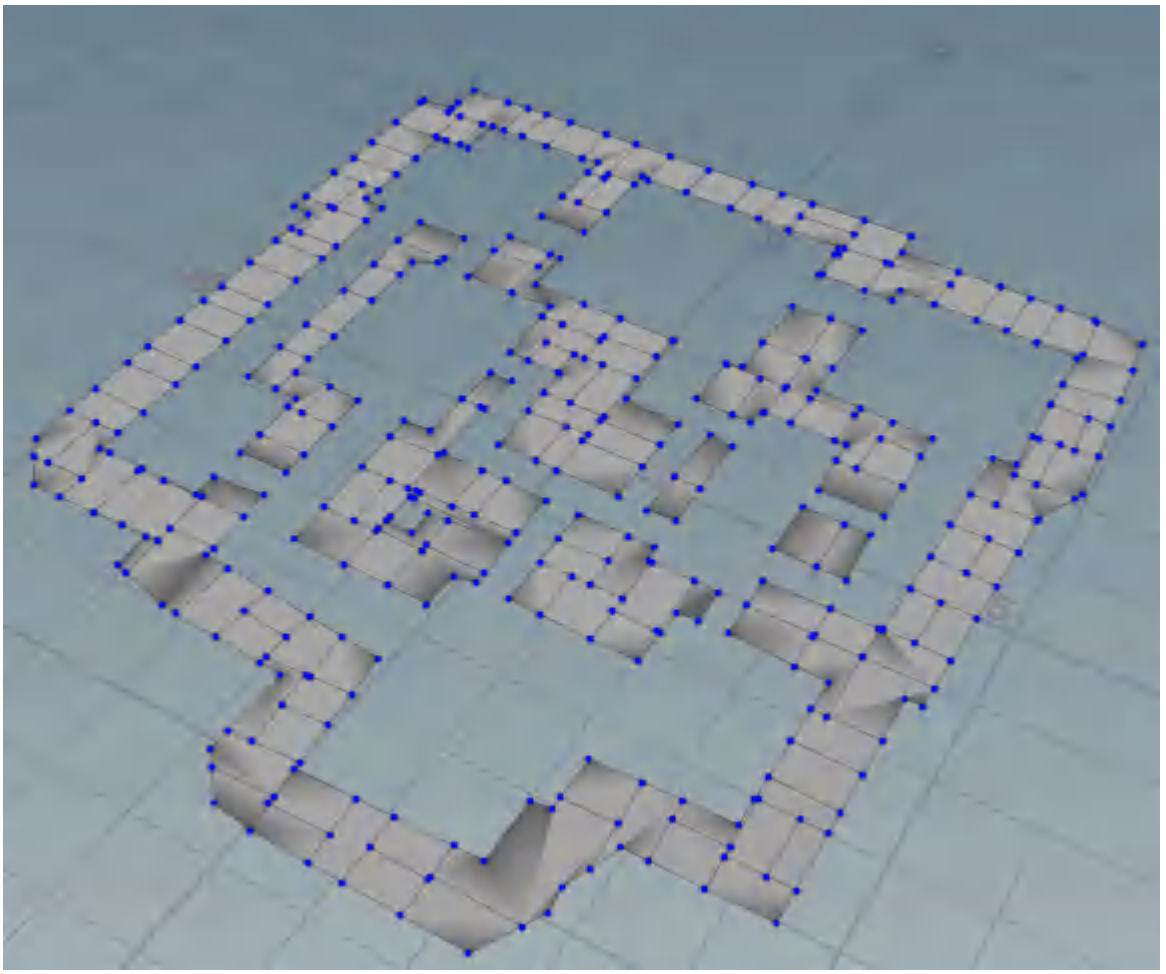


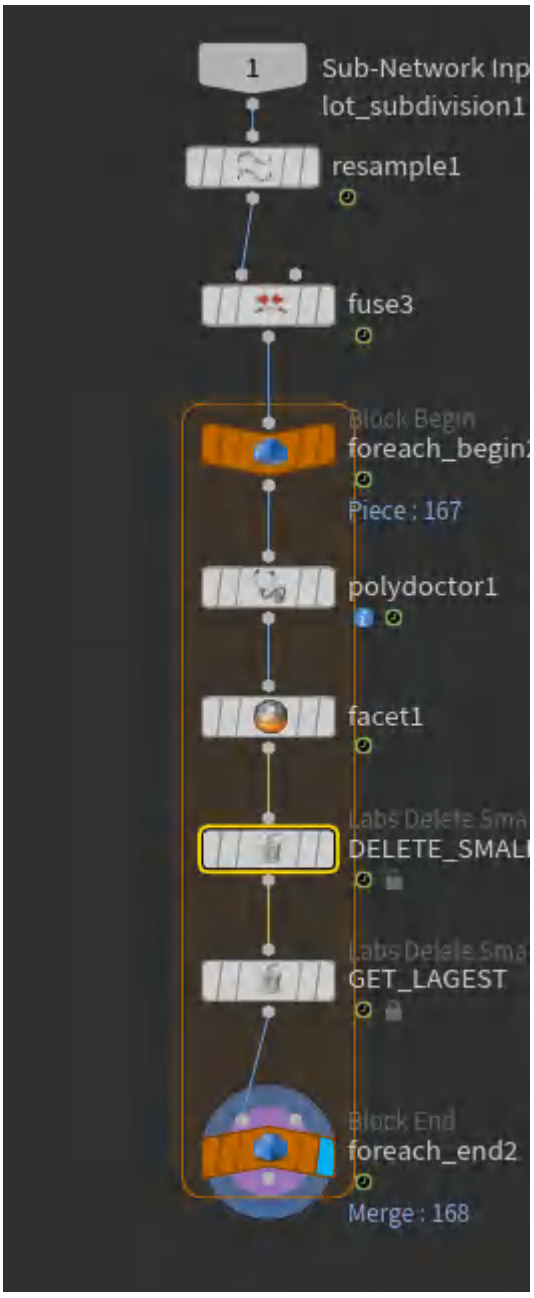

用polyexpand节点扩展出layout的其余部分。但一般来说会有很多问题。所以利用fuse，polyreduce，dissolve flat edge n等节点对模型进行清理，再用lot subdivision节点进行细化区

域。简单来说就是把一个或多个闭合多边形（即地块）切割成多个较小的子多边形（子地块）。





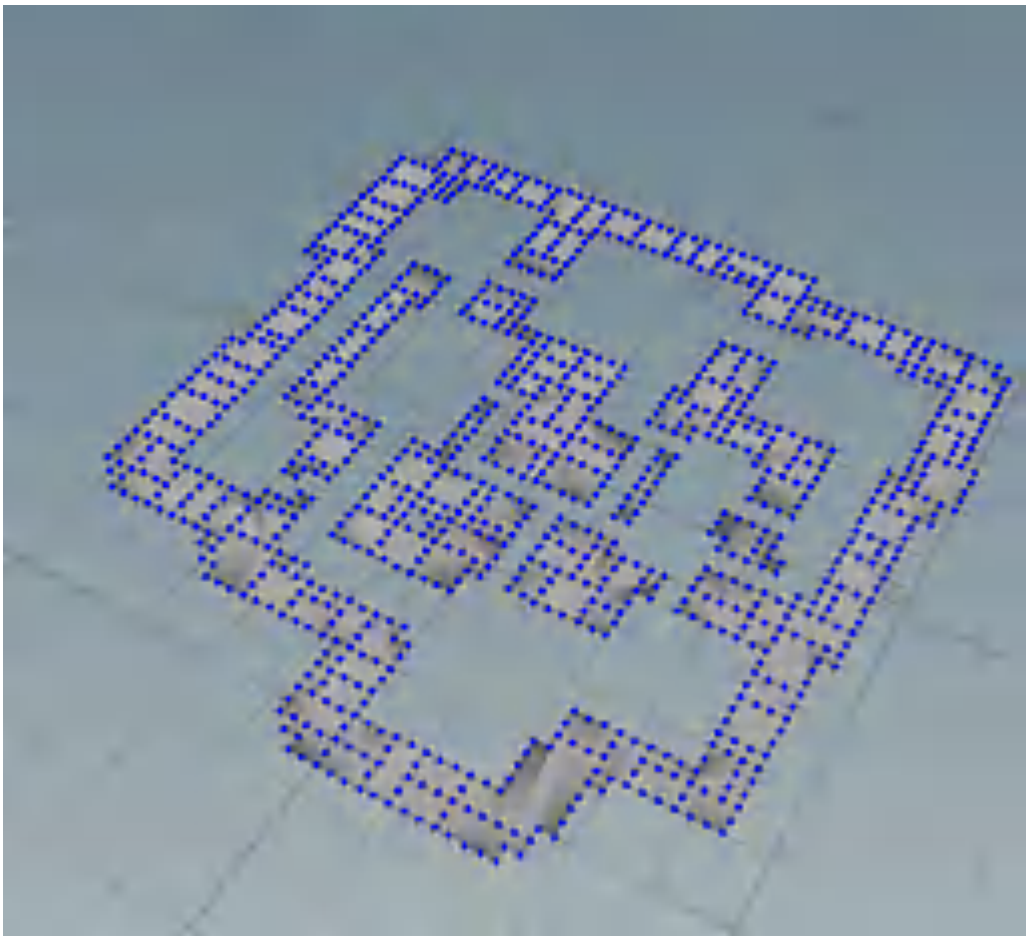




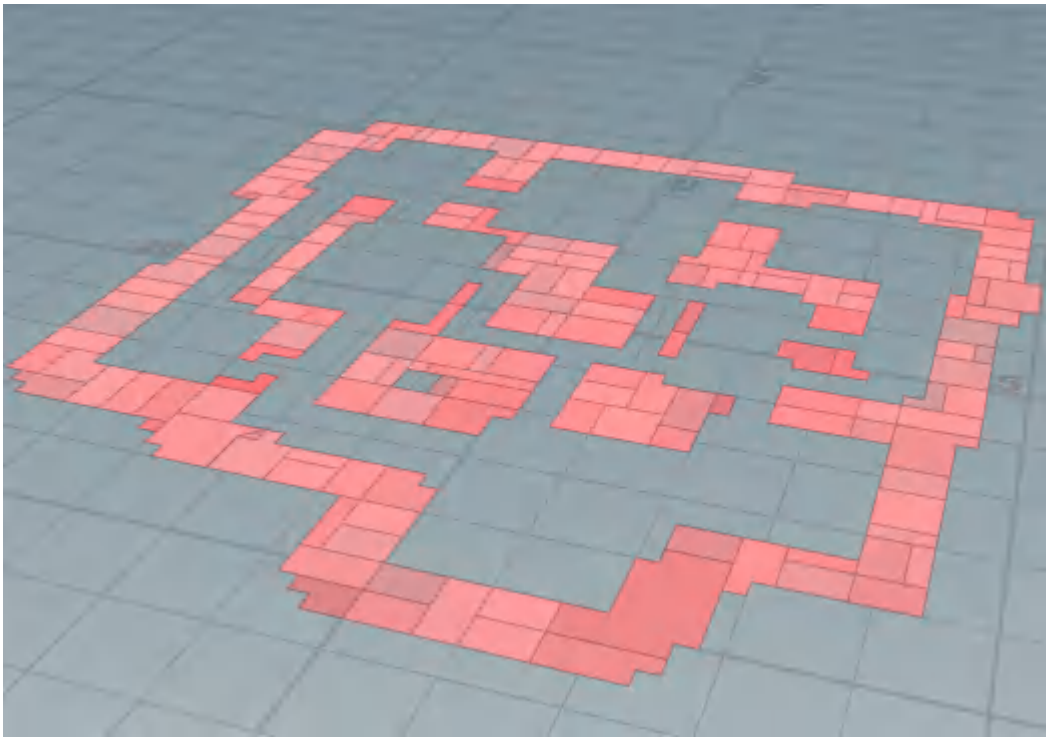

#clean

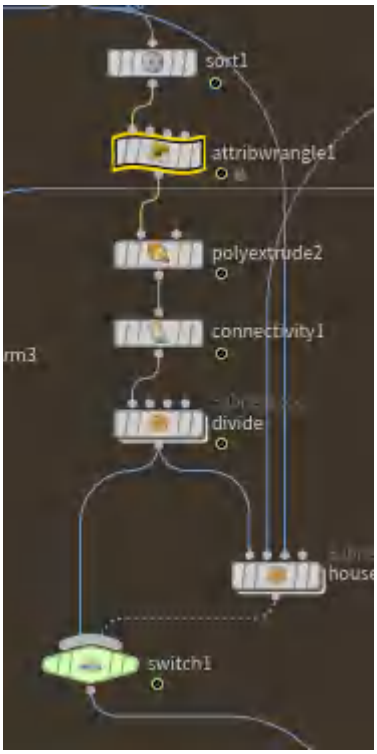
用resample重新采样几何曲线或边界，与fuse节点做搭配合并在一定距离内的重合点，修复裂缝、断边或拼接缝以及减少重复点、统一拓扑。

foreach遍历每个piece，用polydoctor节点检测几何错误，勾选5+edges，不处理复杂的多边形面（有 5 条以上边的面），避免过度修复，用facet节点修复面法线等问题，第一次的delete small part节点目的是删除体积或面积小于指定阈值的部分，第二次则是**只保留最大的连通几何**，其他都删掉。目的是最终只留下最主要的主体几何，确保输出是一个干净的模型。



利用attributerandomize节点给每个区块上色，这边的cd深度就代表了建筑的强度。后面利用polyextrude挤出。

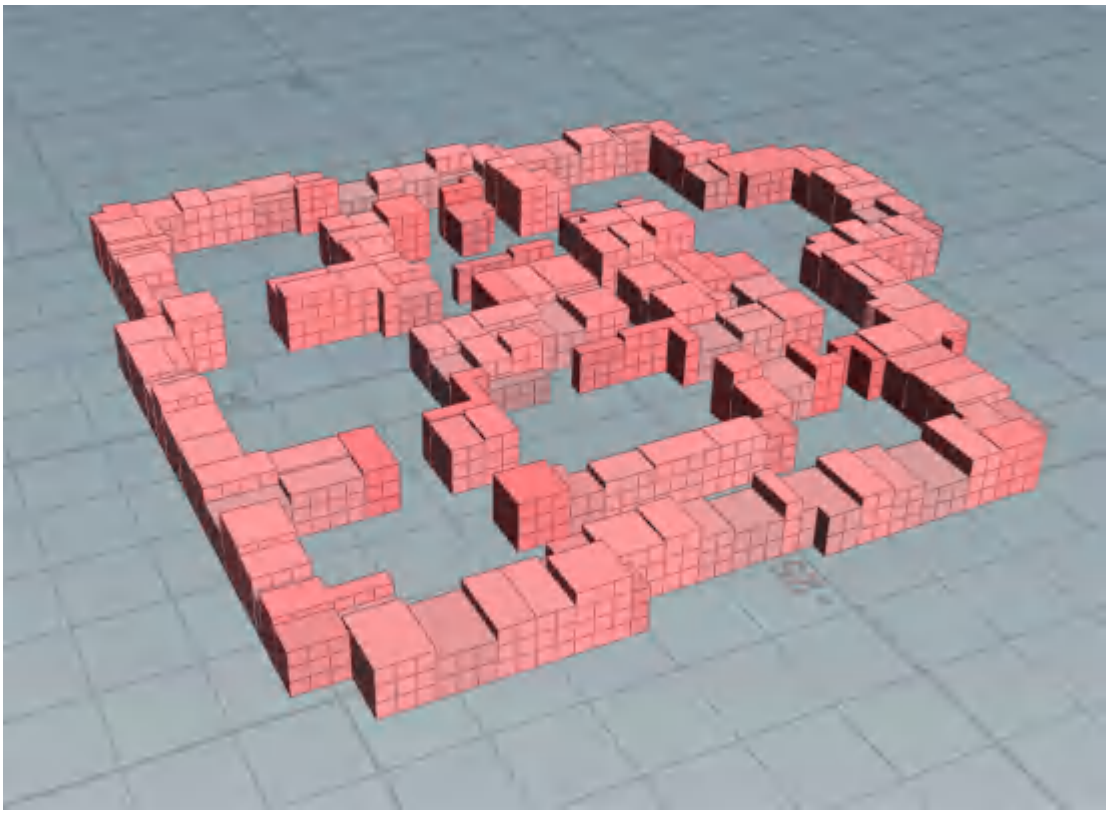


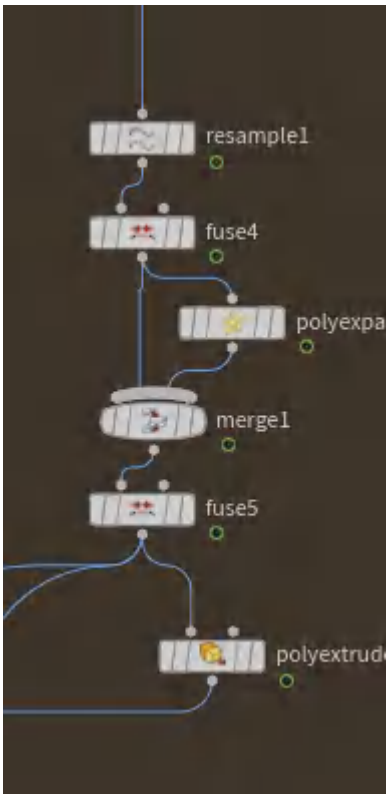


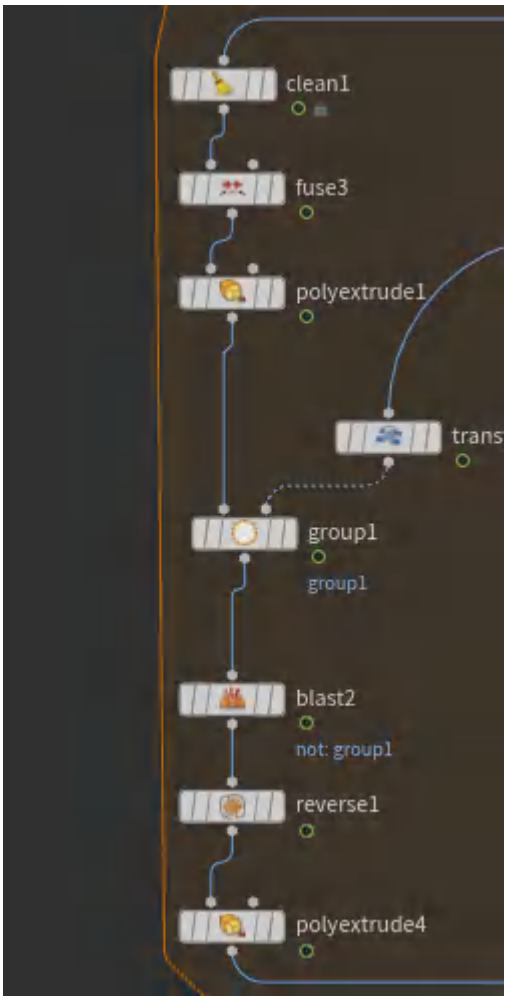

sort打乱primitive的排序，然后利用cd属性进行polyextrude，divde也是清理模型面片的作用，主要是给模型划分窗户的位置，利用divide节点的bricker polygon功能，将面划分成类似砖块（tile）的网格。然后里面下面这个表达式就能根据模型自身的高度自动确定砖块分布，避免出现高度方向上砖块被切残的问题。其目的是为了后面copy windows 做准备，但其实blockout阶段我觉得这样就够了，做这个house主要是美观向的，可做可不做。不过因为是学习，所以我跟着做了一遍，理解一下building的原理。

Y轴

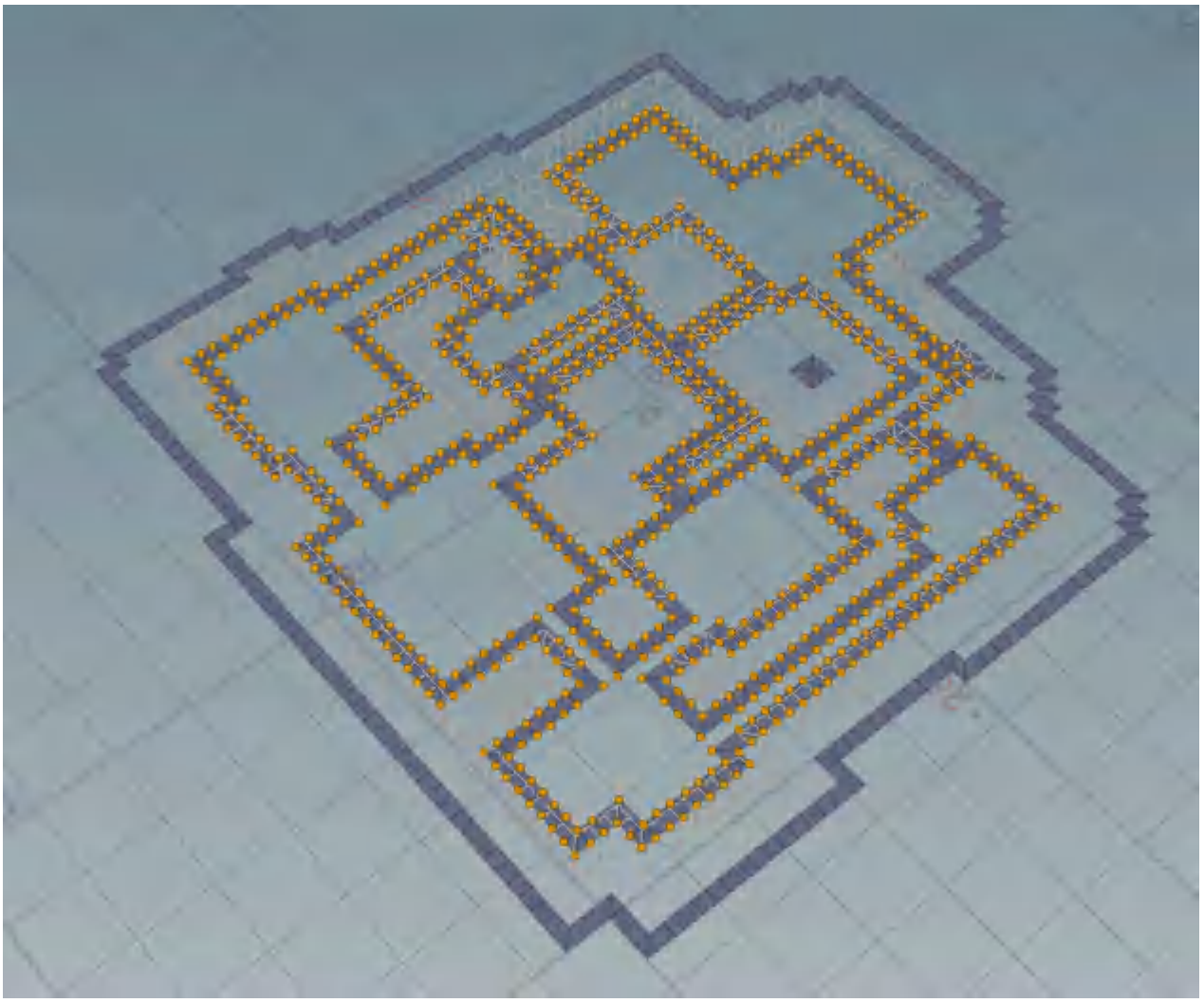
```
bbox(0,D_YSIZE)/rint(bbox(0,D_YSIZE))
```

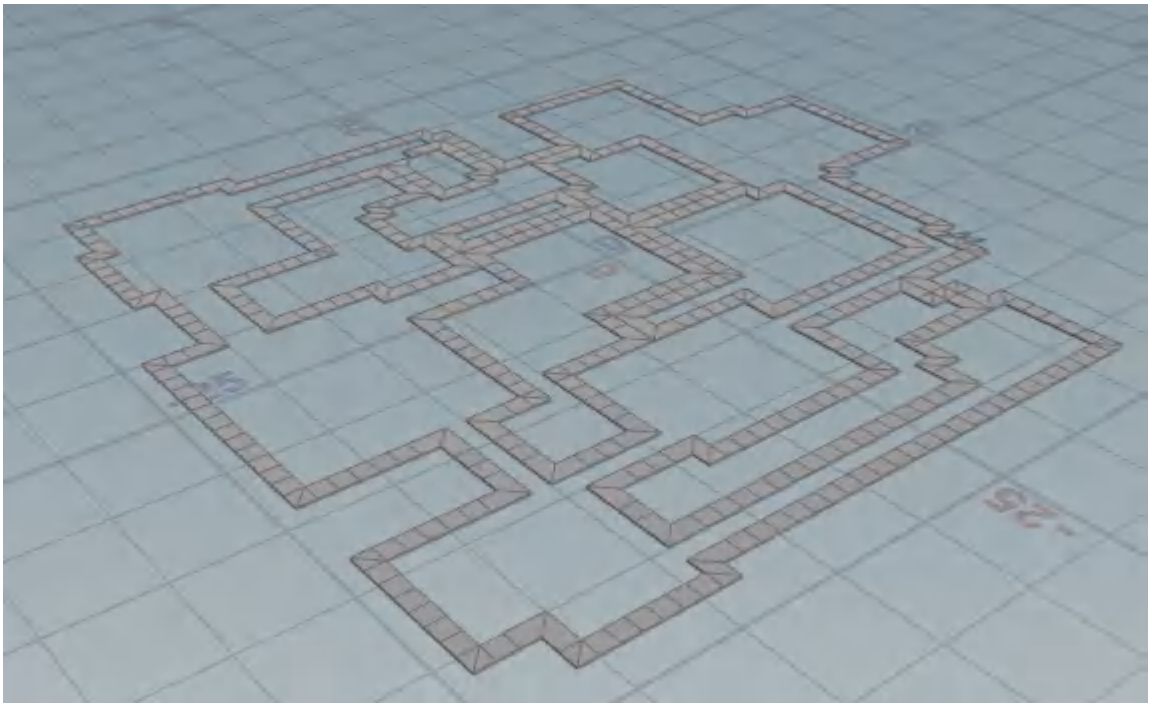


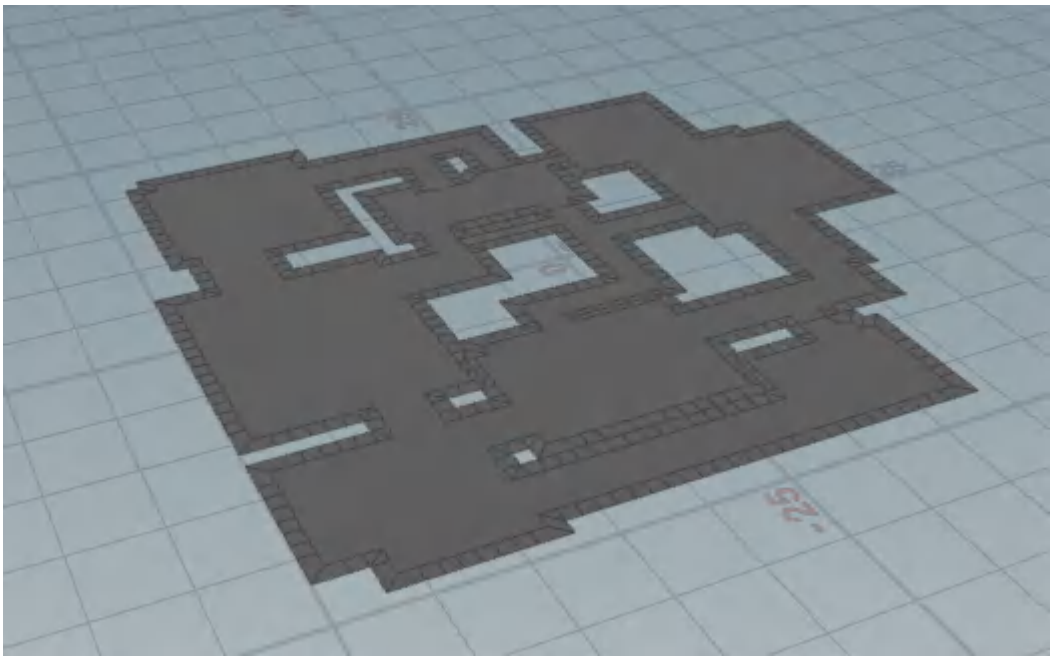

resample1
fuse4
polyexpa
merge1
fuse5
polyextrud



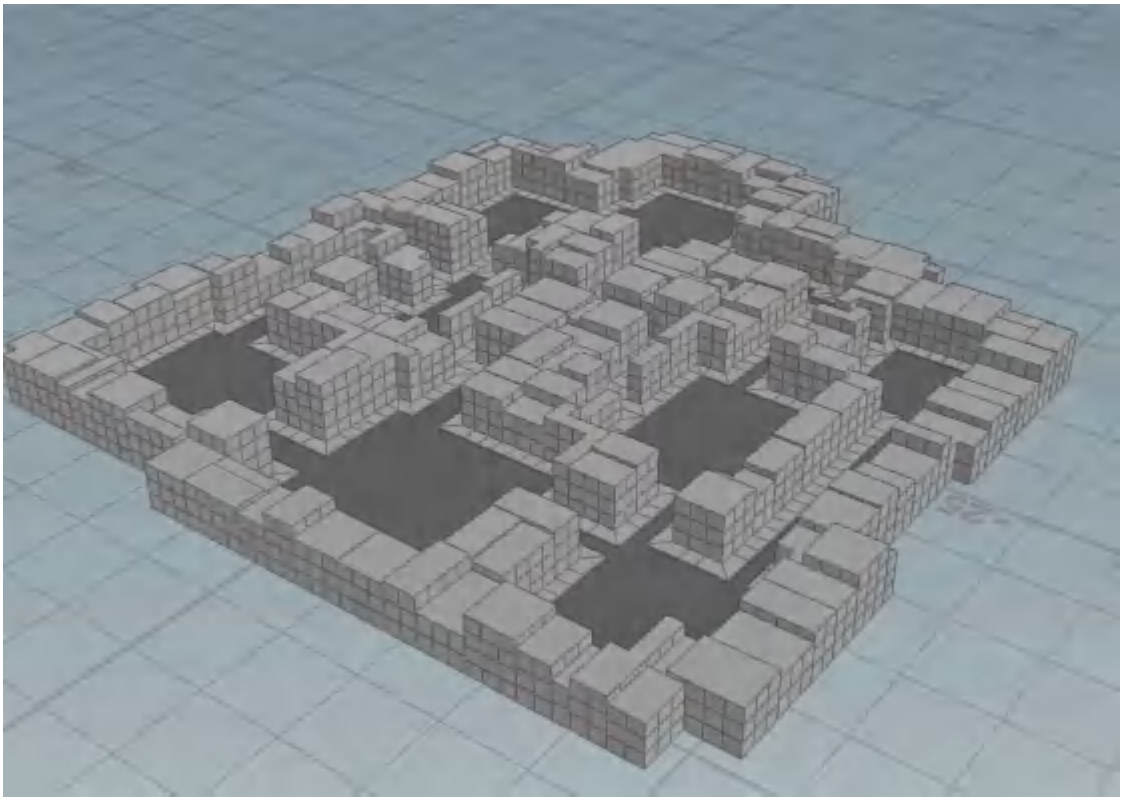

这边从上面blast节点resample一个分支出来，主要是做一个group节点的bounding box以及底部的mesh。用polyexpand扩展出一点然后merge起来，在将其extrude出来就能当做包围盒了。blast出来并extrude就得到了台阶了。



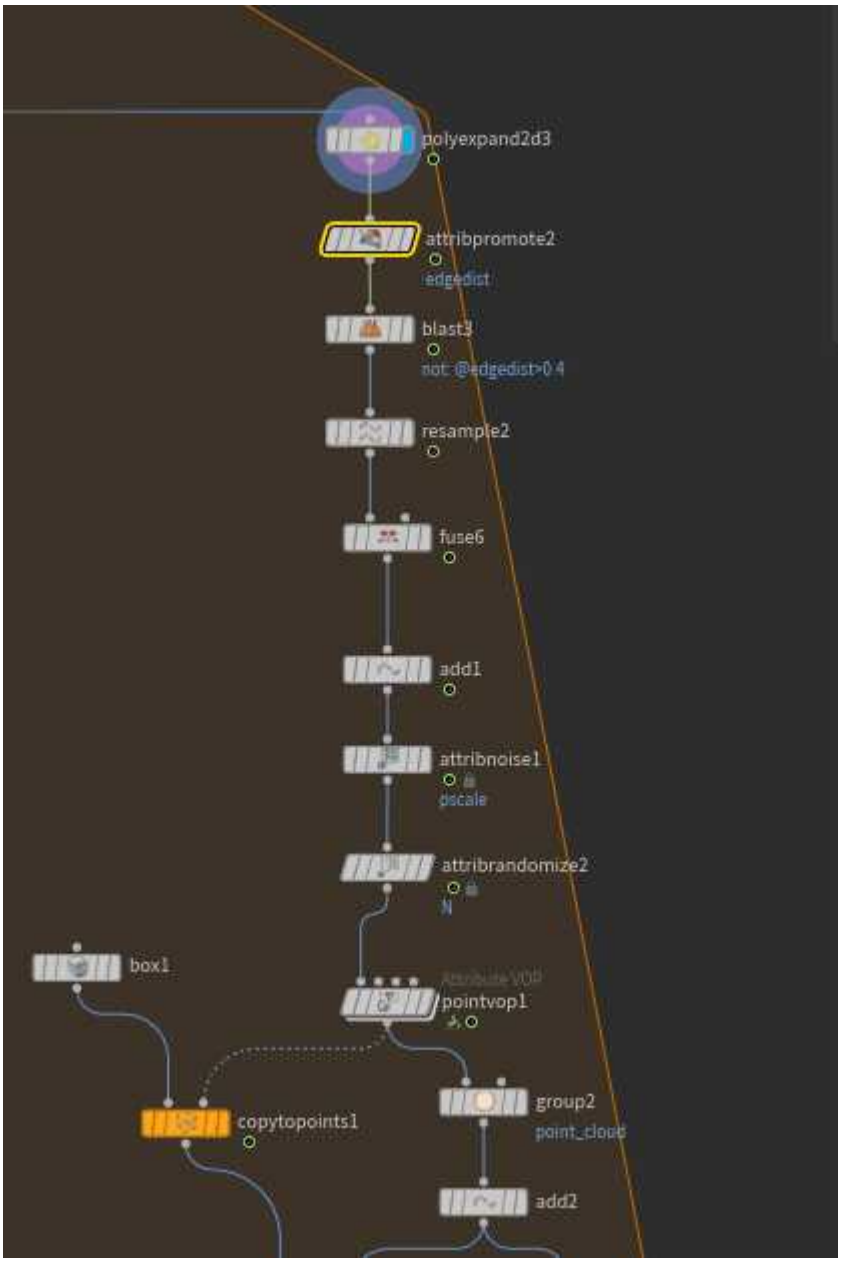


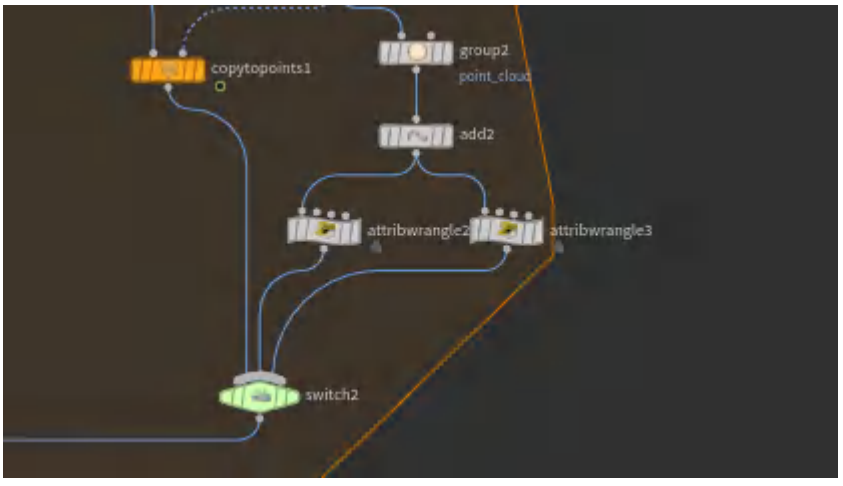


将上面得出的结果merge起来就是一个最基础的白盒



# Other item

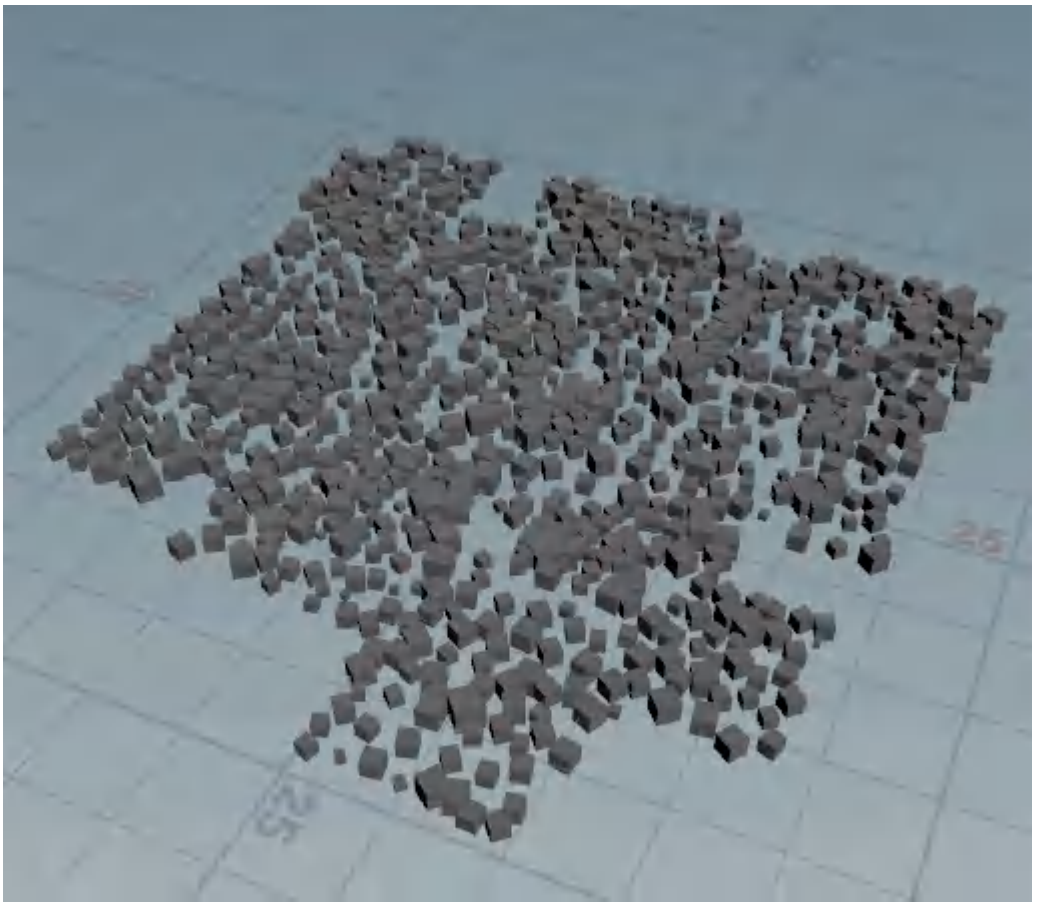



利用polyexpand的特性，expand出点，利用expand自带的attribute（edgedist）进行筛选，然后在resample，在利用noise扰动一下pscale值，便能够得到物品白盒的点，然后copy to point即可，这边也创建了两个attribute，分别是在unity以及unreal当中可以input prop的模型进来。

```
s@unity_instance=chs("Prop");
```

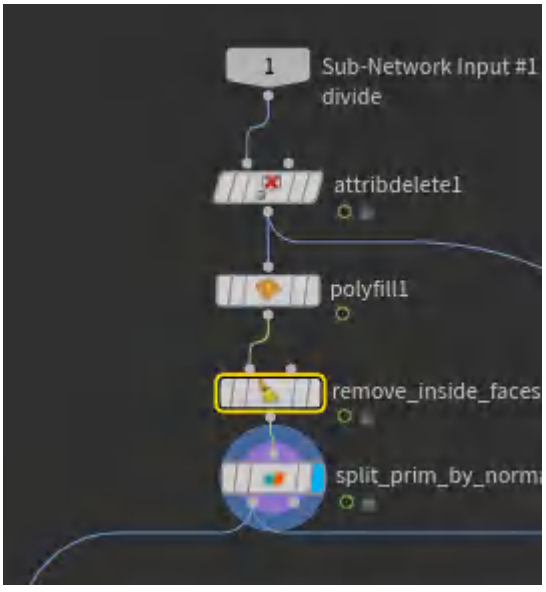
```
s@unreal_instance=chs("Prop");
```



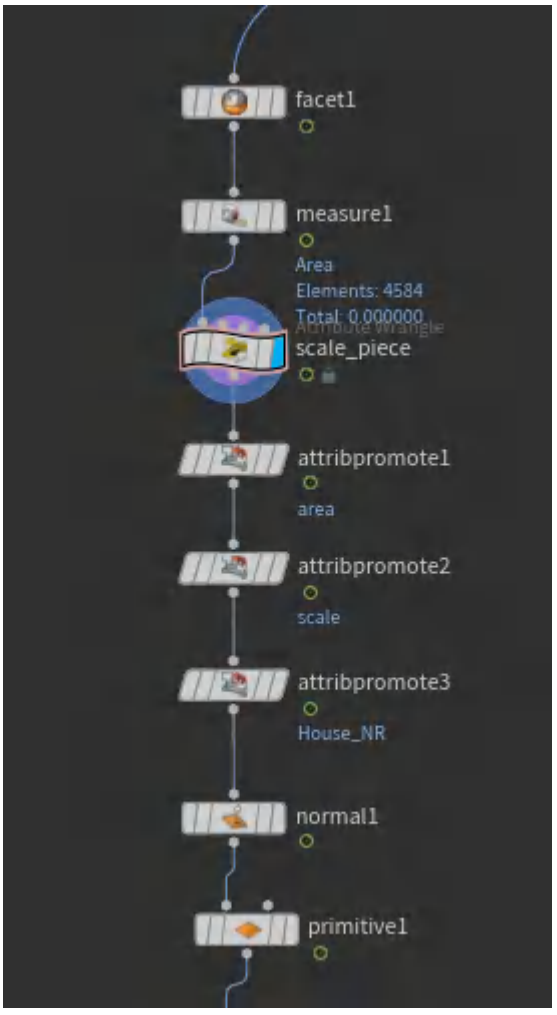
上面是用layout展示的，如果切换成props层就会不一样了。因为props层的mask会不一样。

# house

## wall




用delete节点将cd 属性删掉，polyfill和remove节点清理一下模型，split掉最上面那一层的mesh




```
int pnts[] = primpoints(0,@primnum);

vector pos_0, pos_1, pos_2, pos_3;

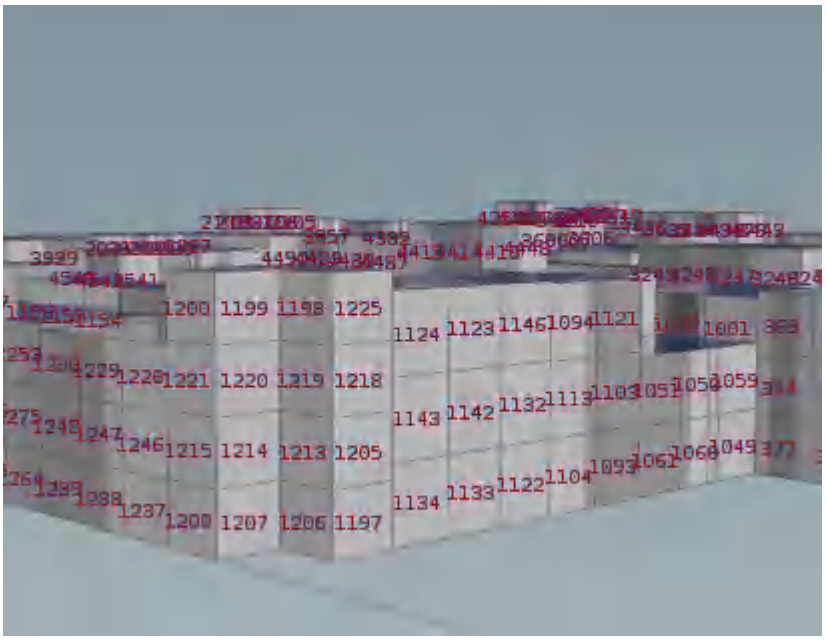
pos_0 = point(0,"P" ,pnts[0]);
pos_1 = point(0,"P" ,pnts[1]);
pos_2 = point(0,"P" ,pnts[2]);
pos_3 = point(0,"P" ,pnts[3]);

float height = abs(pos_0.y - pos_2.y);
float width = @area/height;

@scale = set(width ,height,width);
```

获取当前 primitive（多边形）的所有点编号，并存入数组 `pnts[]`。
取出该多边形 4 个顶点的位置
实际上就是每一个windows的4个点，因为这是嵌套在foreach里面的根据primitive层级。
用点 0 和点 2 的 **y 轴差值** 计算高度。
height是对角点之间的垂直高度差
由面积公式反推宽度：
矩形面积 = 宽 × 高 → 宽 = 面积 / 高。

area(面积)就是上面measure节点计算出来的。
创建scale变量赋值（x=width ,y=height,z=width）
将算好的数据全部转到point层级，后面进行copy to point。

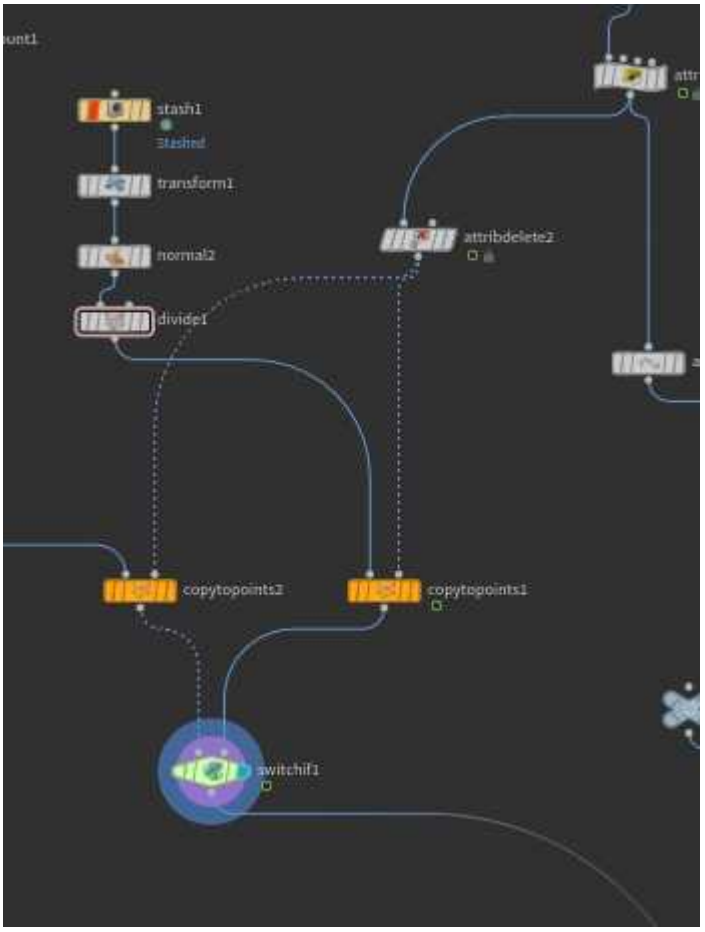



```
i@variant = @House_NR%ch("amount");
```

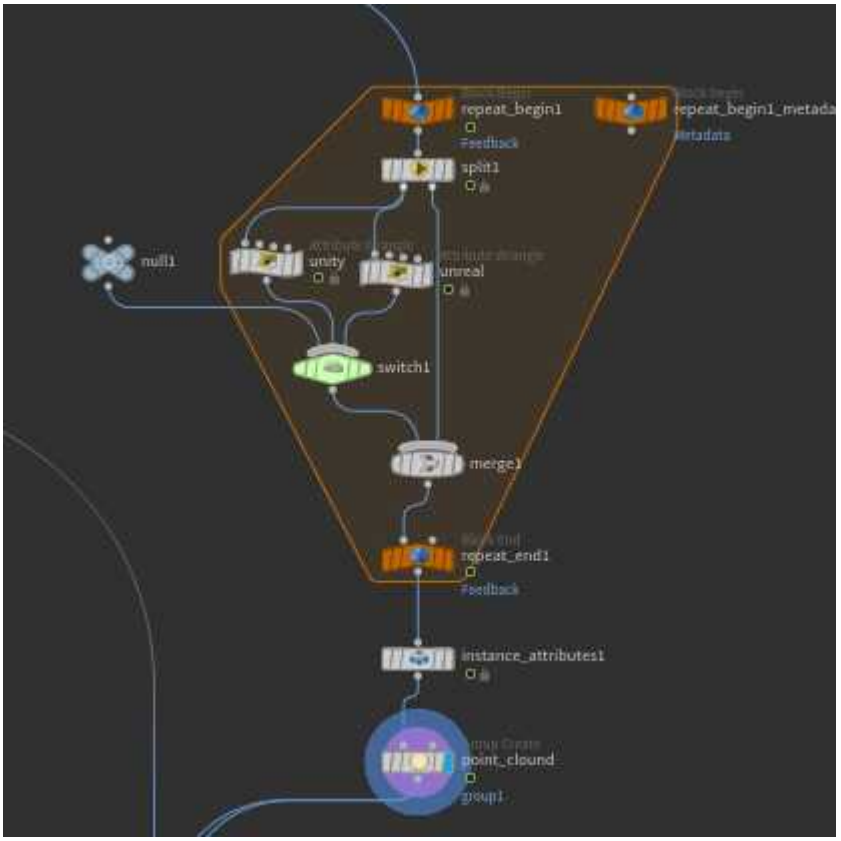
这边创建了一个variant记录一下不同窗口的值。
从刚才的编号中取模（求余数。
余数的范围是 `0` 到 `ch("amount") - 1`。
所以如果你在参数里把 `amount` 设置为 5，那么结果就是：

```
@variant = 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, ...
每 5 个 primitive 就会循环一遍
```

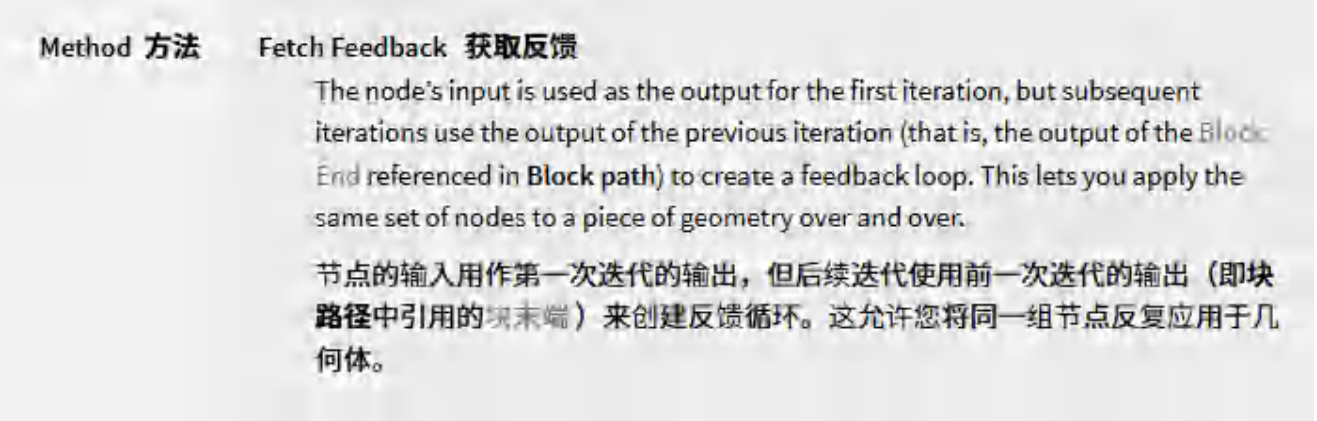
这也就代表你有几个不同的窗口或者wall。




stash节点里面就是windows的模型，其实大致的原理就像拼乐高一样，做好几个不同的wall然后在拼上去，上面算好的scale就是每一个块的大小保证拼接无缝。




这边比较特殊的是unreal和unity的instance，需要嵌套多一层for each 并且是使用feedback。



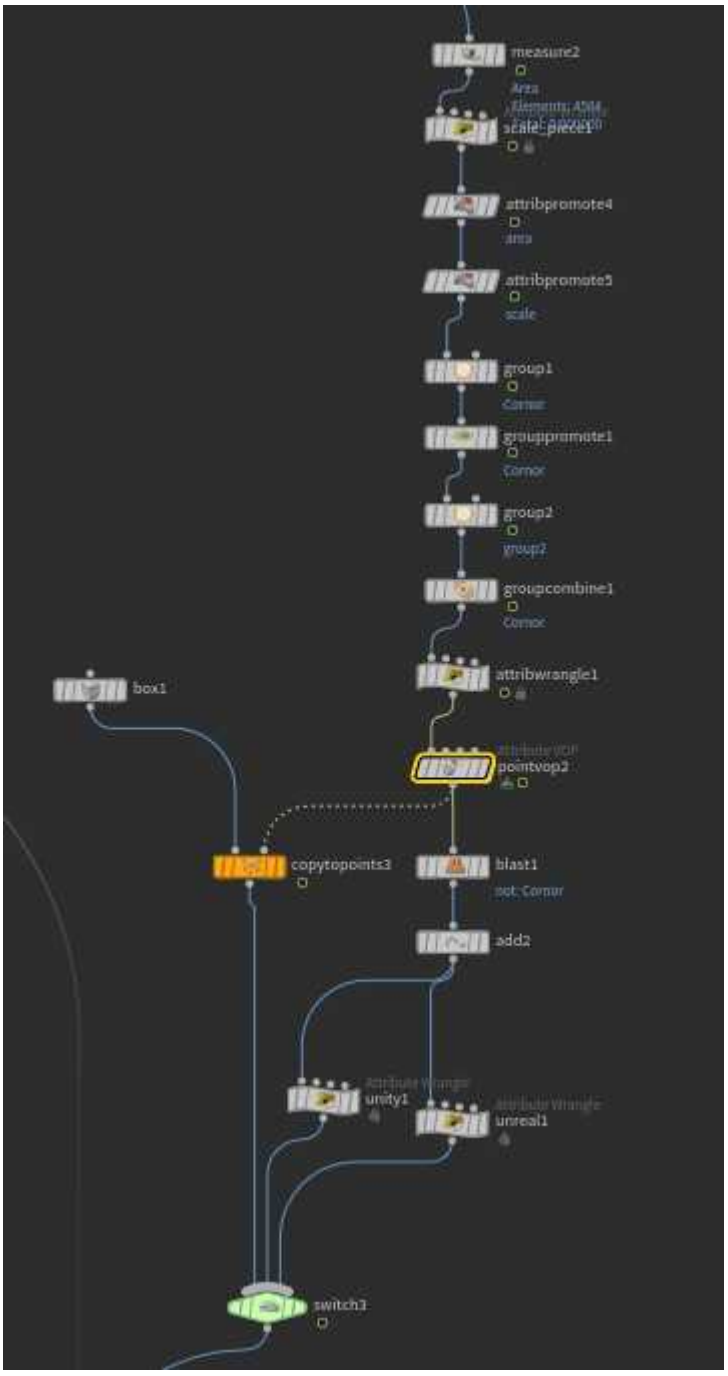

利用split节点将variant的不同wall的值弄出来
在每次循环时迭代值 = 0 时：
表达式变成 `@variant == 0`
当迭代值 = 1 时：变成 `@variant == 1`
当迭代值 = 2 时：变成 `@variant == 2`

```
@variant==`detail("../repeat_begin1_metadata1/","iteration",0)`
```

```
s@unreal_instance = chs("wall");
s@unity_instance = chs("wall");
```

利用上面表达式以及instance_attributes节点进行instance。

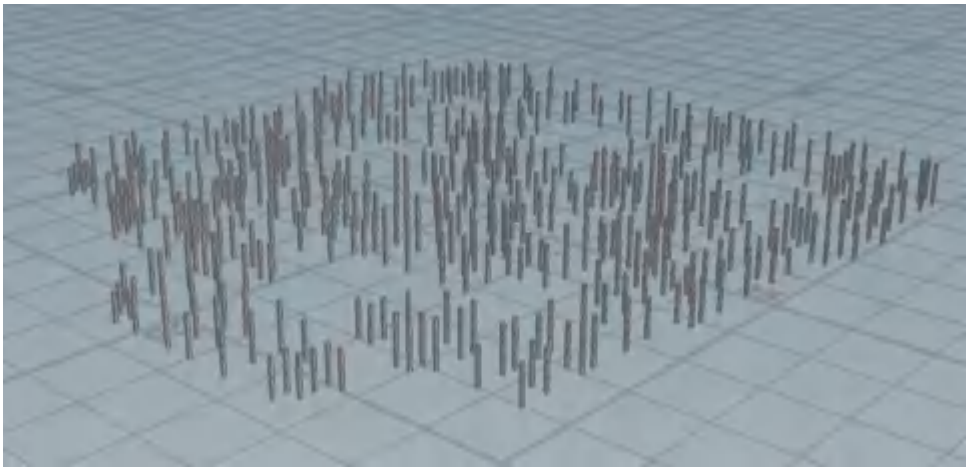
# Pillar



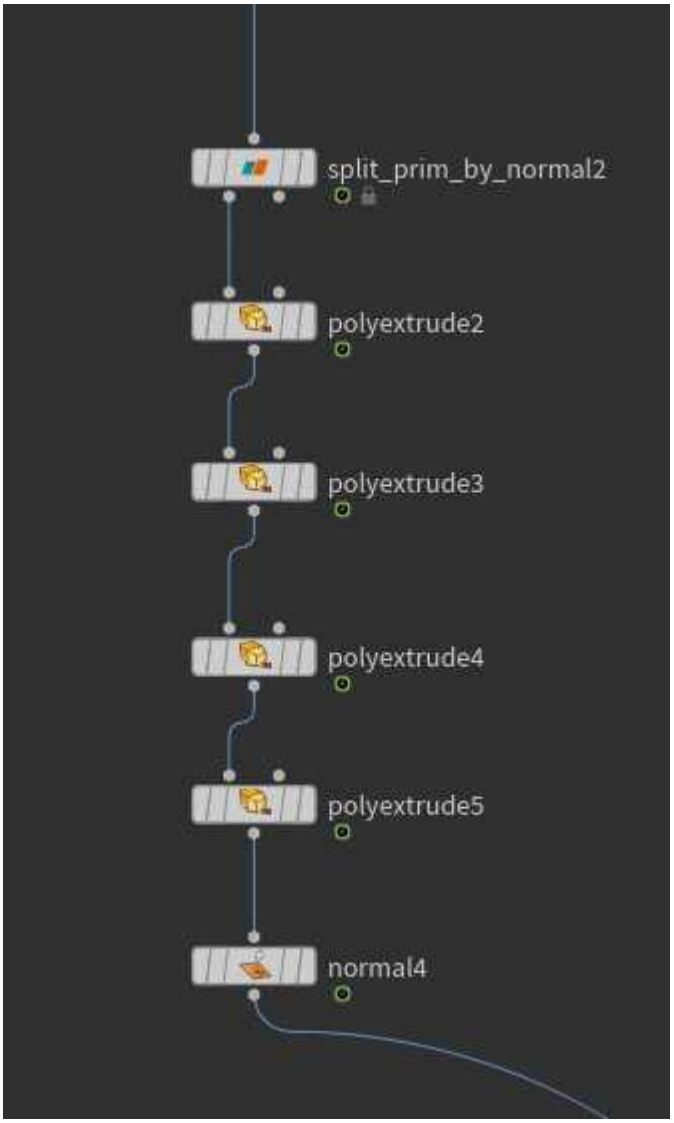



跟上面一样，算好scale，利用group节点edge功能筛选出pillar的边，用group节点@P.y<0.1 选出下面的点，利用group combine节点 减掉，在进行copytopoint。同样进行unreal和unity的

操作，这样可以从unity和unreal中自己input进来model，可以自己custom pillar的模型。

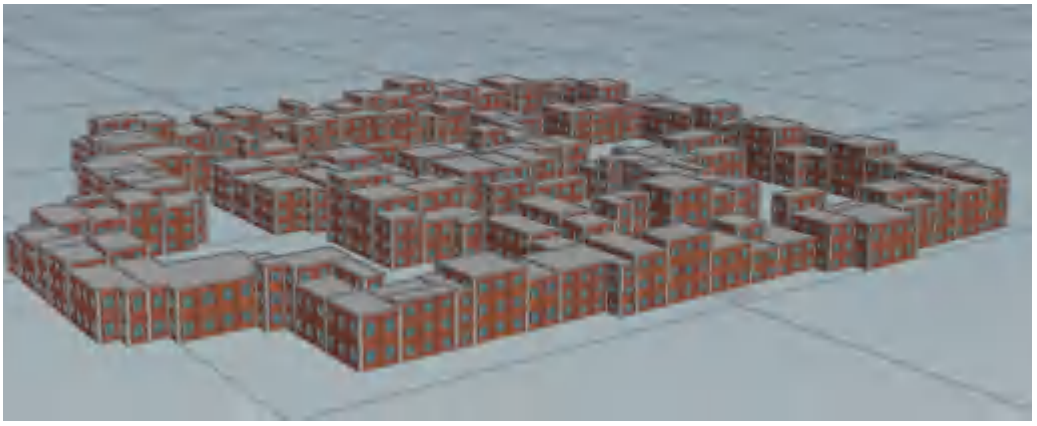


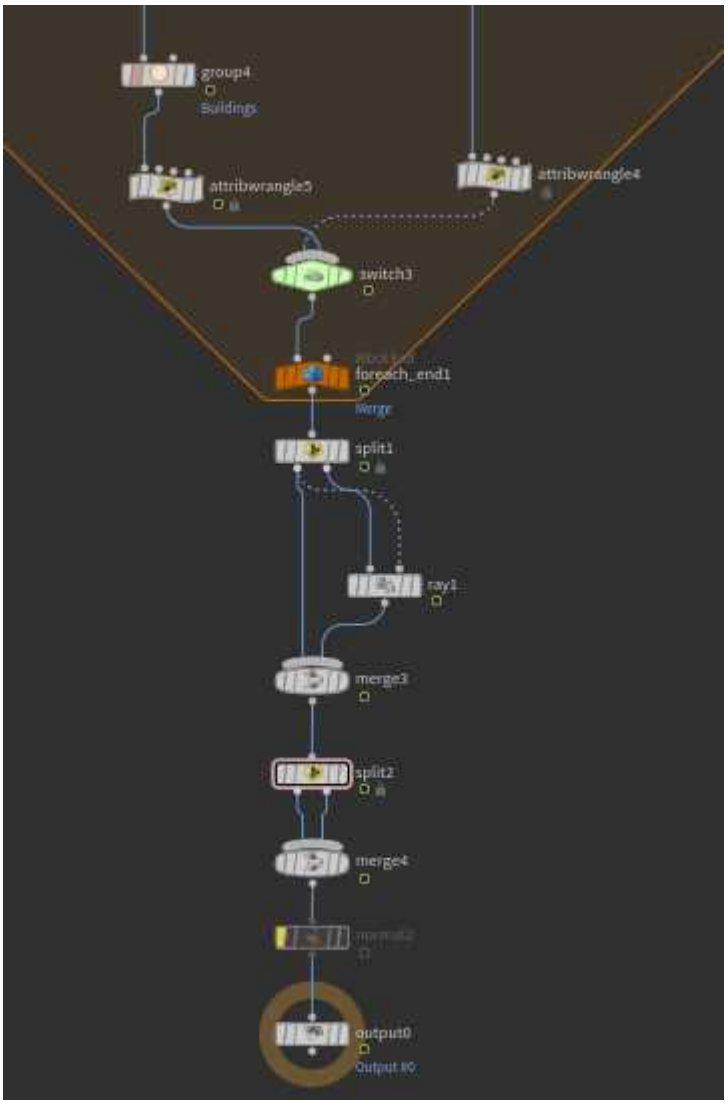
# roof




屋顶就split出上层，polyextrude几下即可

最后将所有结果merge起来，housebuilding就做完了。

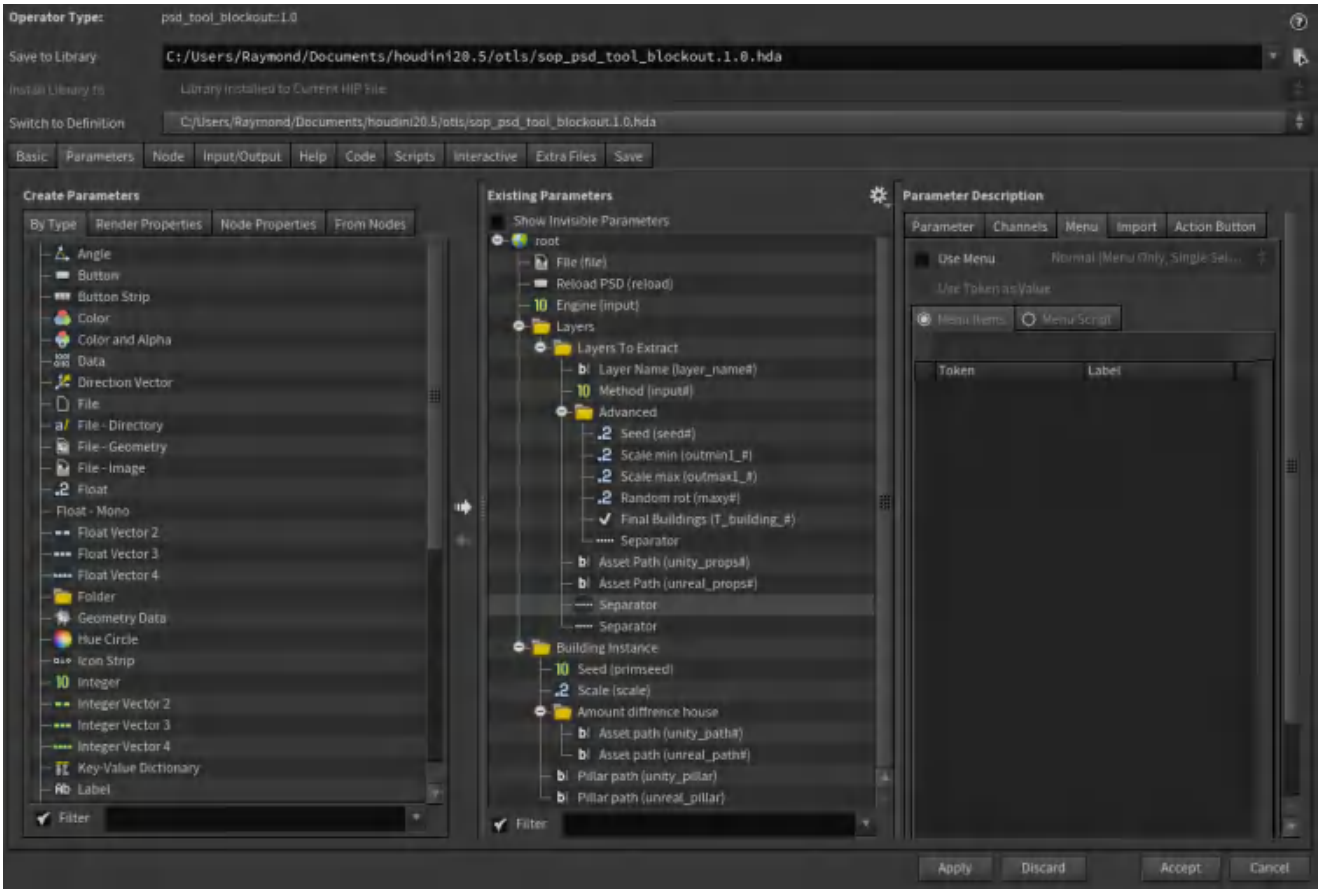





最后在做一些模型的处理以及模式的切换就完成了。

# HDA

本次的HDA相较来说会比较复杂一点，先看list。



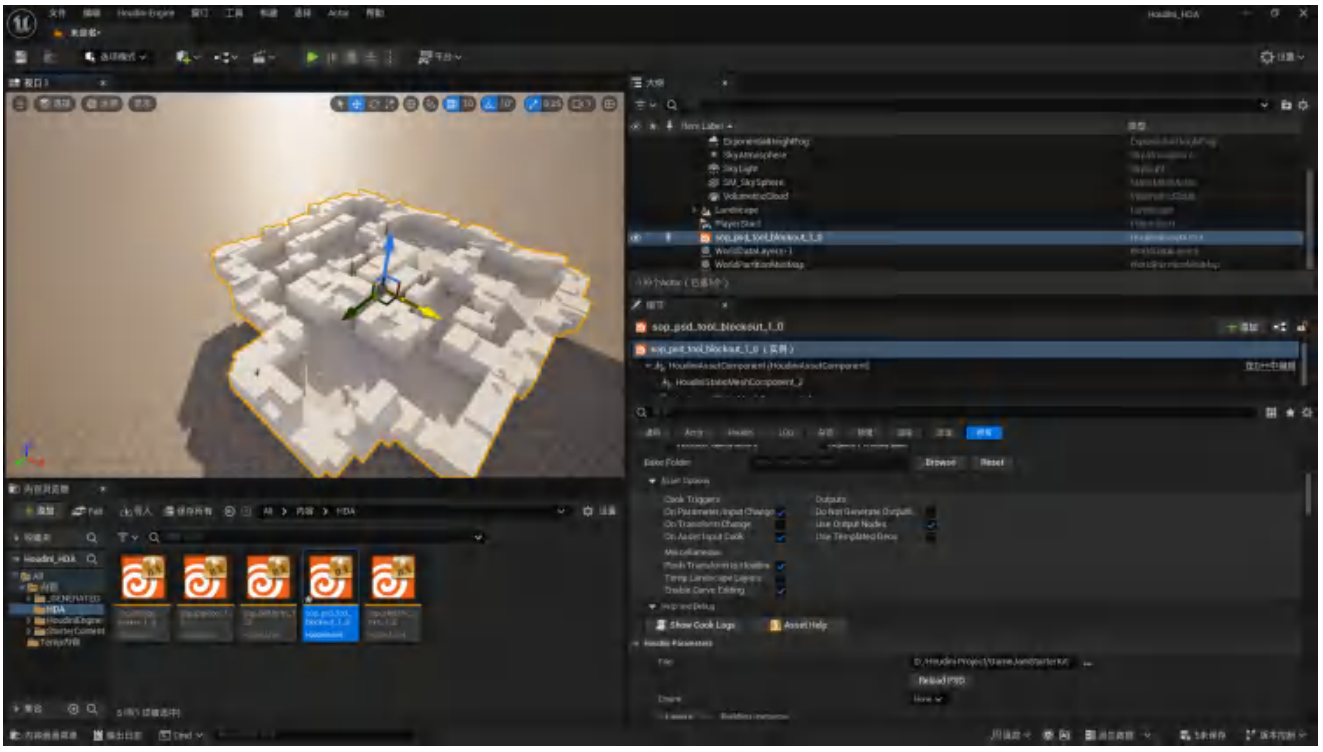

比较大的区别是，因为本次用了许多的循环，因此很多的值都用了#符号配合 strcat表达式给每个primitive进行赋值，利用表达式动态解析HDA里面的参数路径，从而给循环里面的每一个组件进行赋值，特别是这种randomize，又想在HDA里面进行操控的就得使用这种方式。

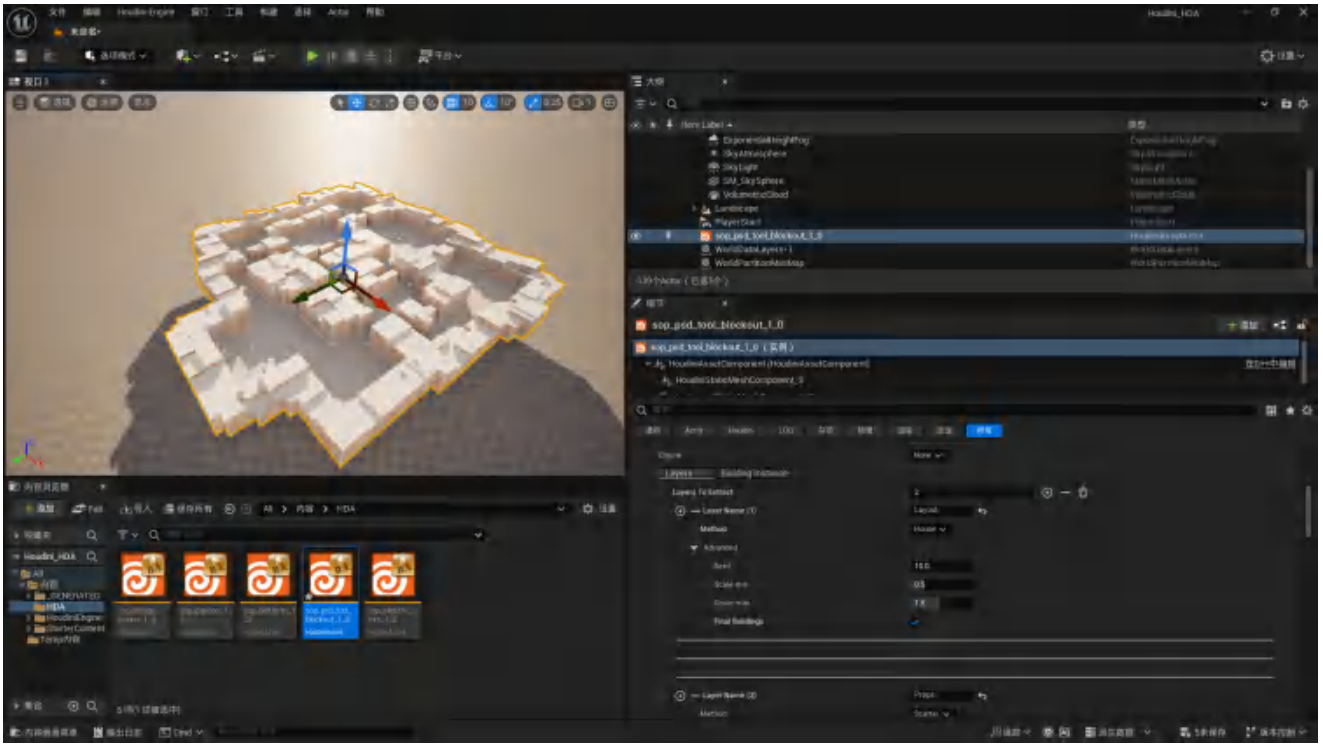
```
ch(strcat("../outmin1_",detail("../foreach_count1","iteration", 0) +1))
ch(strcat("../outmax1_",detail("../foreach_count1","iteration", 0) +1))*3.5
```

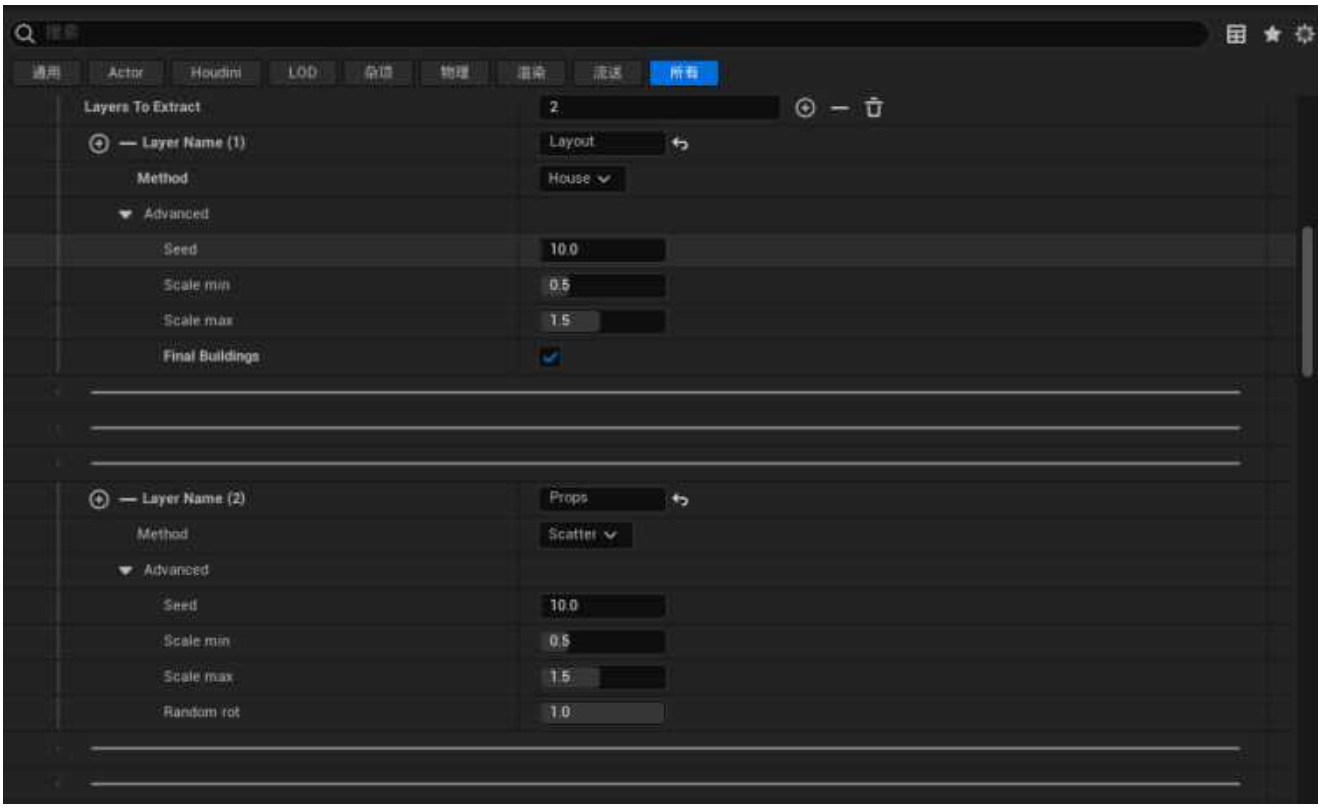
像上面的两个表达式就是对房子的大小进行操控。

只要是有关HDA来更改循环内部数据的，就都需要使用这种表达式进行更改。

# Import to Unreal






Actor
Houdini
LOD
所有
Layers To Extract
2
— Layer Name (1)
Layout
Method
House
Advanced
Seed
10.0
Scale min
0.5
Scale max
1.5
Final Buildings
— Layer Name (2)
Props
Method
Scatter
Advanced
Seed
10.0
Scale min
0.5
Scale max
1.5
Random rot
1.0