RESQUAKE

Version 1.0
For Houdini Apprentice
Created by CT2

ResQuake is a Geometry Nodes setup for Houdini that creates diverse and random Philippine houses procedural earthquake damage effects.

The base version comes with 46 quality models of our 50 generated models

---

Documentation

Getting started

When you download this product, you get a ZIP file. This is not an addon, so it can't be installed the way an addon would be.

Here's how to set it up once you've downloaded it:

1. Make sure you downloaded the latest version of Houdini. There's an available free license for Houdini which is the Houdini Apprentice
2. Extract the zip file and place the folder somewhere convenient
3. Open Houdini. Go to File>Open>Downloads
4. Find the Resquake.hipnc file and Select it.
5. Done!

The download contains a .hipnc file.

Settings

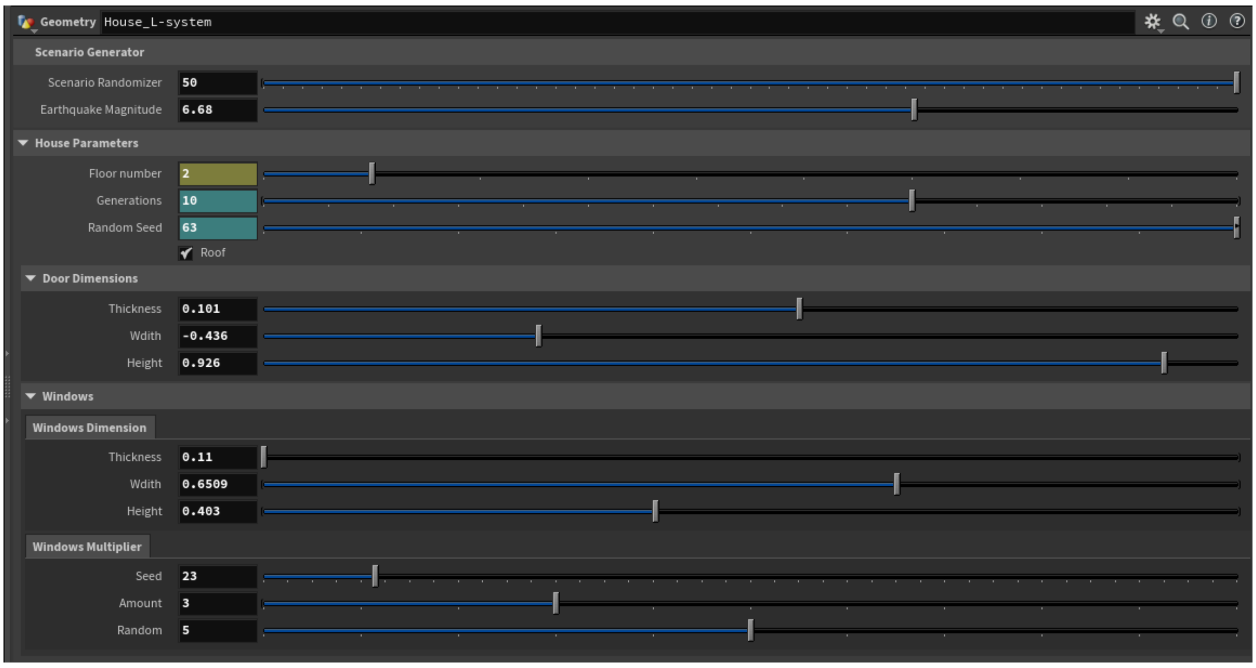
Since the generators and presets are built from the same modular pieces, they share most of the settings that are accessed directly from the modifier.

Scenario Generator Controls:

Scenario Randomizer: This control will randomize the creation of the models. This can control the *Floor Number, Generations* and *Random Seed.*
Apply Post-earthquake effects checkbox: Checking this box will apply the post-earthquake damage effects on the random generated models.
Earthquake Magnitude: This controls the magnitude of earthquake.

House Parameters Controls:

Floor Number: Controls the number of *Floor numbers* of the desired model.
Generations: Generates the desired number of grids for the model.
Random Seed: Randomizes the growth path of the grids.
Roof Checkbox: Unchecking this will remove the *Roof* of the house model.
Door Dimensions: Controls the dimensions of the door
Windows Controller: Controls the dimensions of the window and the number of the windows to be placed in the model.



Geometry House_L-system

Scenario Generator

Scenario Randomizer 50
Earthquake Magnitude 6.68

House Parameters

Floor number 2
Generations 10
Random Seed 63
✓ Roof

Door Dimensions

Thickness 0.101
Width -0.436
Height 0.926

Windows

Windows Dimension

Thickness 0.11
Width 0.6509
Height 0.403

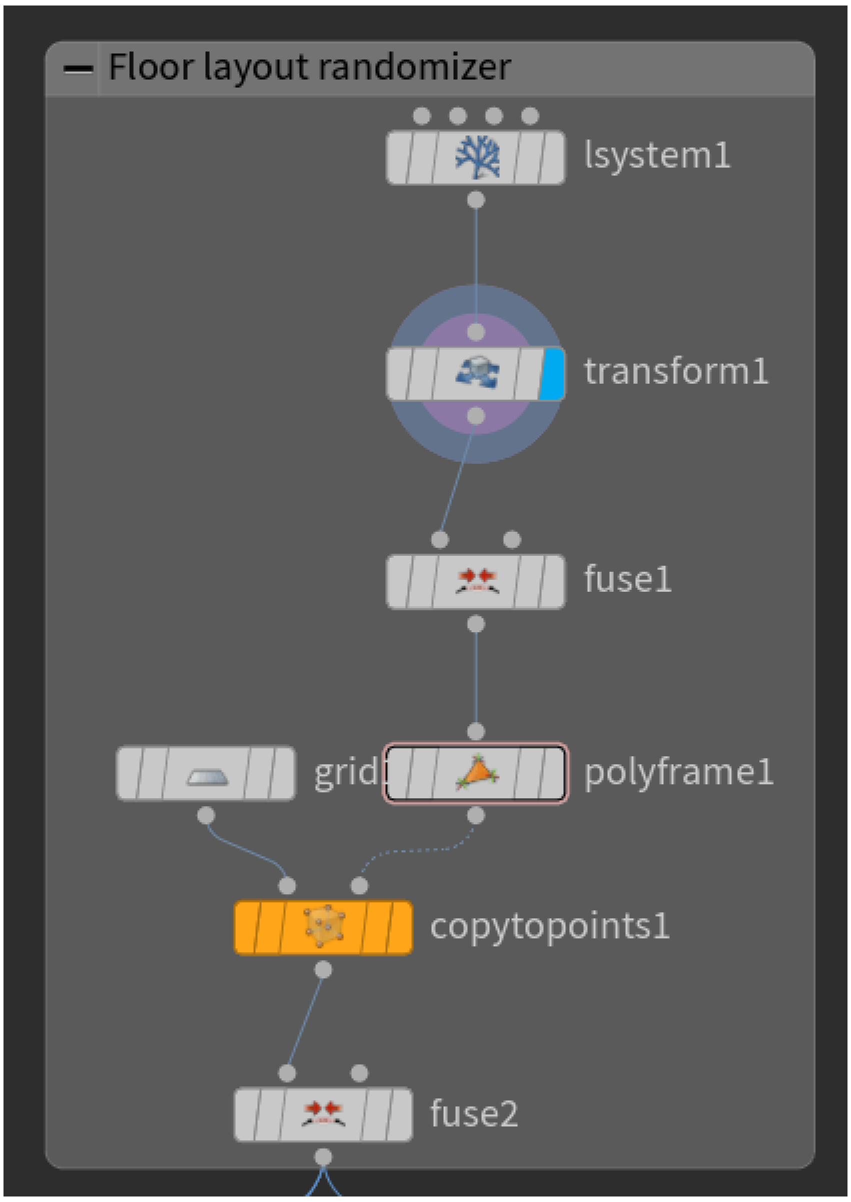
Windows Multiplier

Seed 23
Amount 3
Random 5

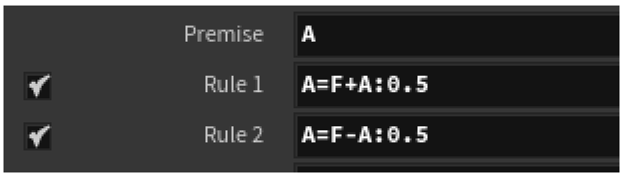
Nodes

I. FLOOR LAYOUT RANDOMIZER

1. **lsystem1:**
   - create lsystem generated line path



a. Rules



Premise A
☑ Rule 1 A=F+A:0.5
☑ Rule 2 A=F-A:0.5

a.a: A

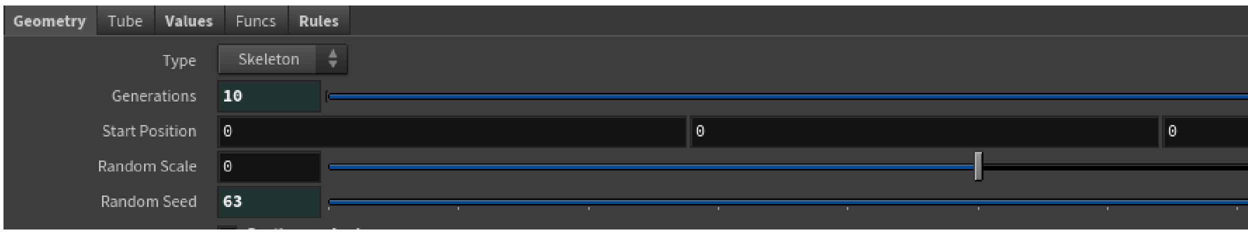
a.b: A=F+A:0.5

- means turn right (+) every instance of A in the string is replaced with F+A 50%

a.c: A=F-A:0.5

- means turn left (-) every instance of A in the string is replaced with F+A 50%

b. Geometry



Type: Skeleton
Generations: 10
Start Position: 0 0 0
Random Scale: 0
Random Seed: 63

b.1 Generations:

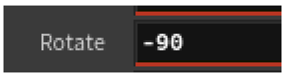
- The number of path to be generated

b.2 Random Seed:

- Randomizer

2. Transform:

- rotates the lsystem path to x-axis so that the growth is on the x-axis



Rotate: -90

3. Fuse: Connects the generated path
4. Copypoints:

- Copies the Grid to the fused Lines which is the Polyframe

4.1 Grid

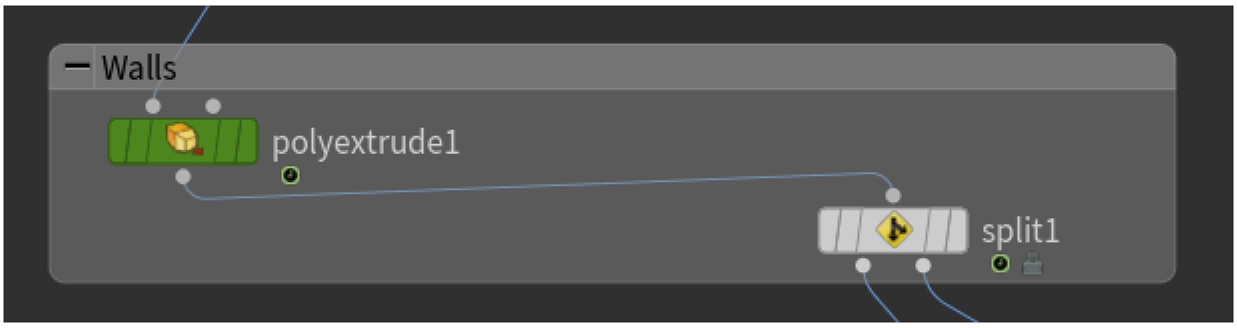
4.2 Polyframe

- calculates the coordinated frames

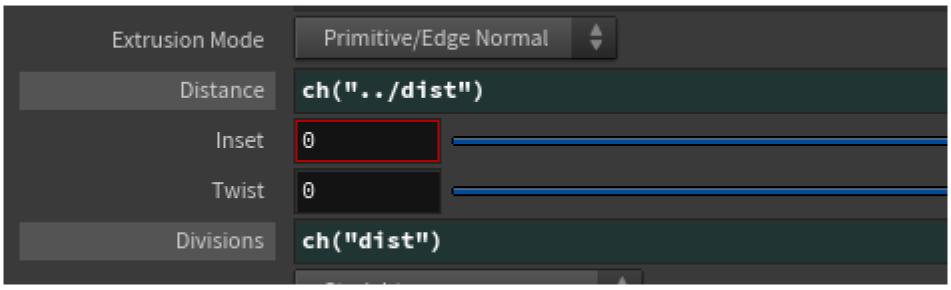
5. Fuse:

- Connects the generated grids

II. Walls



1. Polyextrude:



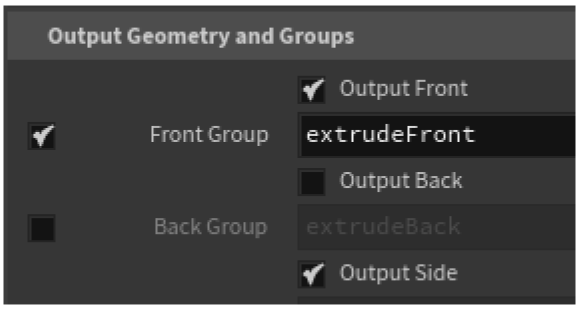
Extrusion Mode: Primitive/Edge Normal
Distance: ch("../dist")
Inset: 0
Twist: 0
Divisions: ch("dist")

1.1 Extrusion Mode: Primitive

- Mode to primitive so it selects the edges

1.2 Distance:

- This is the *Floor Level*



Output Geometry and Groups

☑ Output Front
☑ Front Group: extrudeFront
□ Output Back
□ Back Group: extrudeBack
☑ Output Side

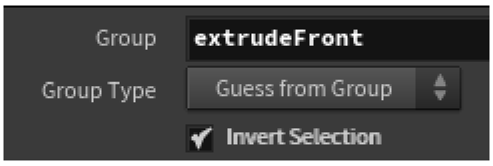
1.3 OutputFront:

- Top Grid (Roof)
- Names the *Roof* to extrudeFront

1.4 OutputSide:

- Side

2. Split:
    - to only be left with the walls and roof

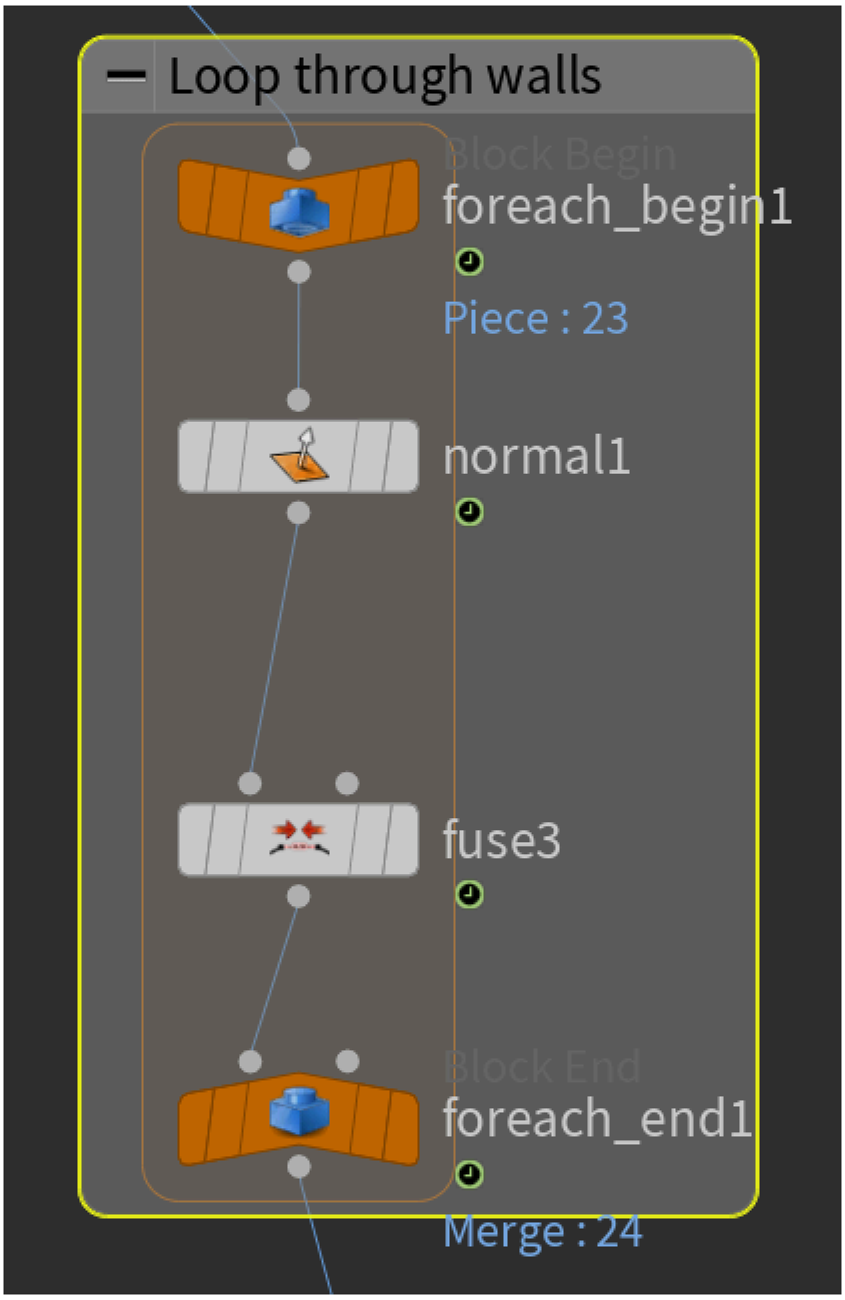


Group extrudeFront

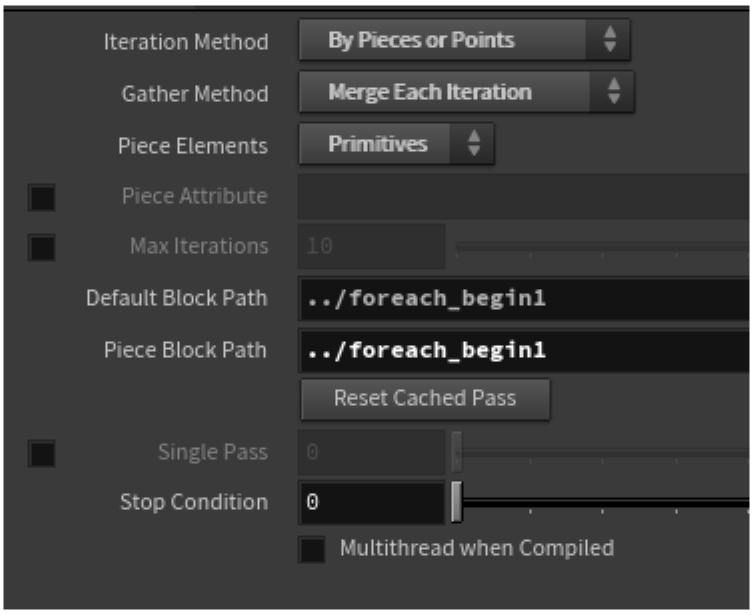
Group Type Guess from Group

☑ Invert Selection

III. LOOP THROUGH WALLS



1. **Foreach loop:**
    - Iteration by Piece or points
    - And the piece elements is primitives so it will get the walls



Iteration Method: By Pieces or Points

Gather Method: Merge Each Iteration

Piece Elements: Primitives

☐ Piece Attribute

☐ Max Iterations: 10

Default Block Path: ../foreach_begin1

Piece Block Path: ../foreach_begin1

Reset Cached Pass

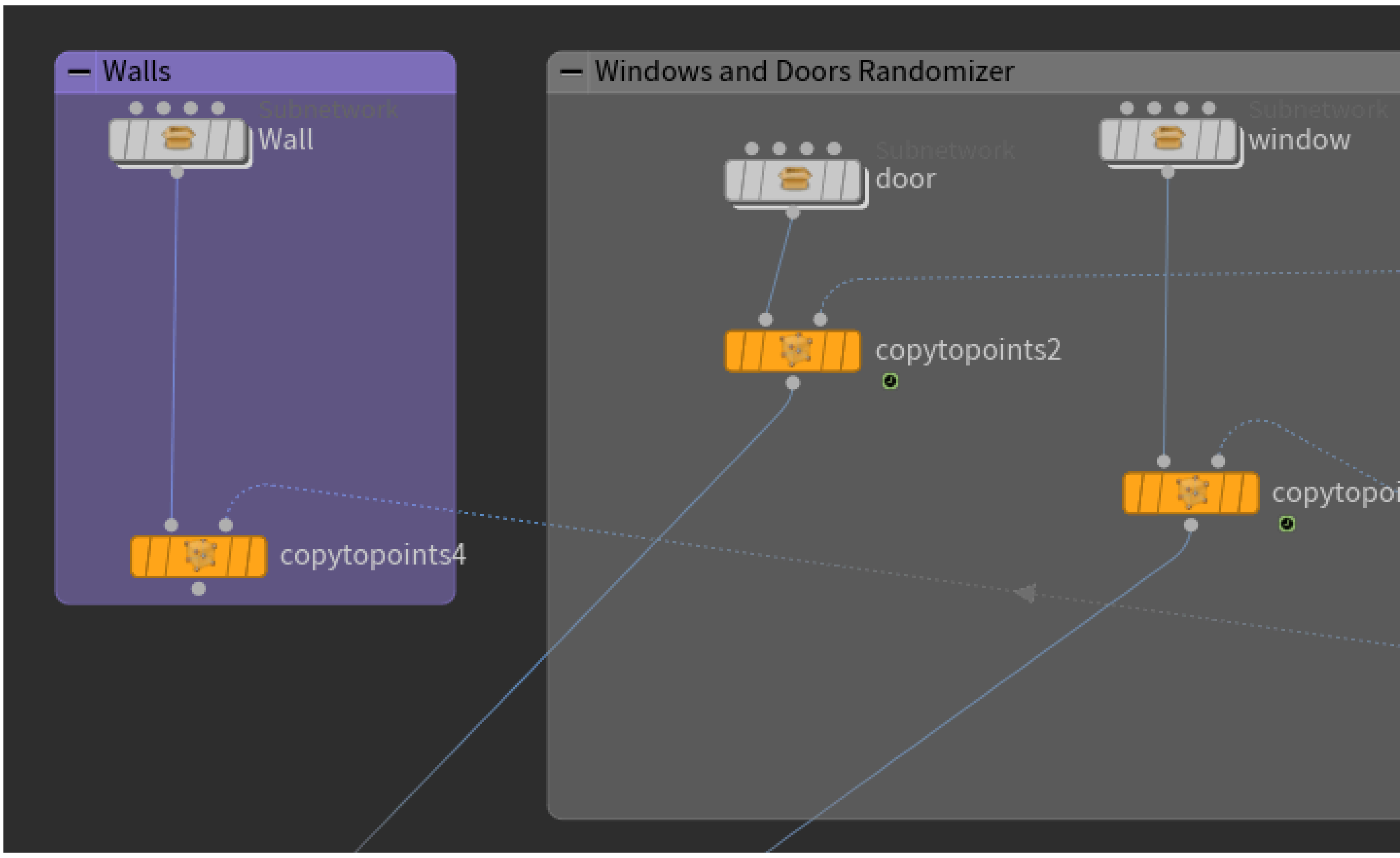
☐ Single Pass: 0

Stop Condition: 0

☐ Multithread when Compiled

2. Normal
    - Computes points and primitives
3. Fuse
    -

IV. Walls, Door, and Windows



1. Walls
    1.1 Wall subnetwork:
    - A subnet that holds the *Wall* object

    1.2 Copy to points:
    - places the walls according to the given points
2. Door
    2.1 Door subnetwork:
    - A subnet that holds the *Door* object

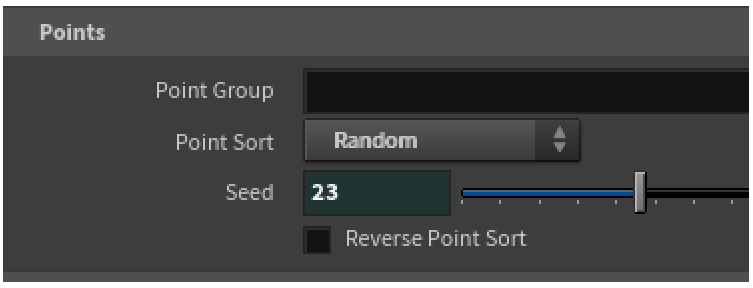
    2.2 Copy to points:
    - places the walls according to the given points
3. Window
    2.1 Window subnetwork:

- A subnet that holds the *Window* object

2.2 Copy to points:

- places the walls according to the given points

4. Split
    - Segregates the walls
5. Sort



Points

Point Group

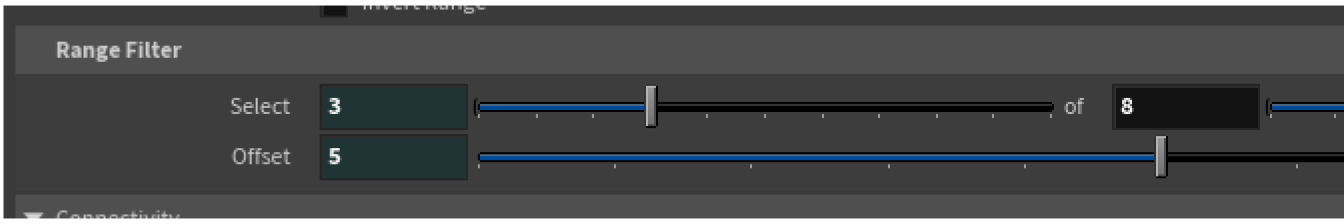
Point Sort: Random

Seed: 23

□ Reverse Point Sort

- Randomize selected points

6. Grouprange



Range Filter

Select: 3 of 8

Offset: 5

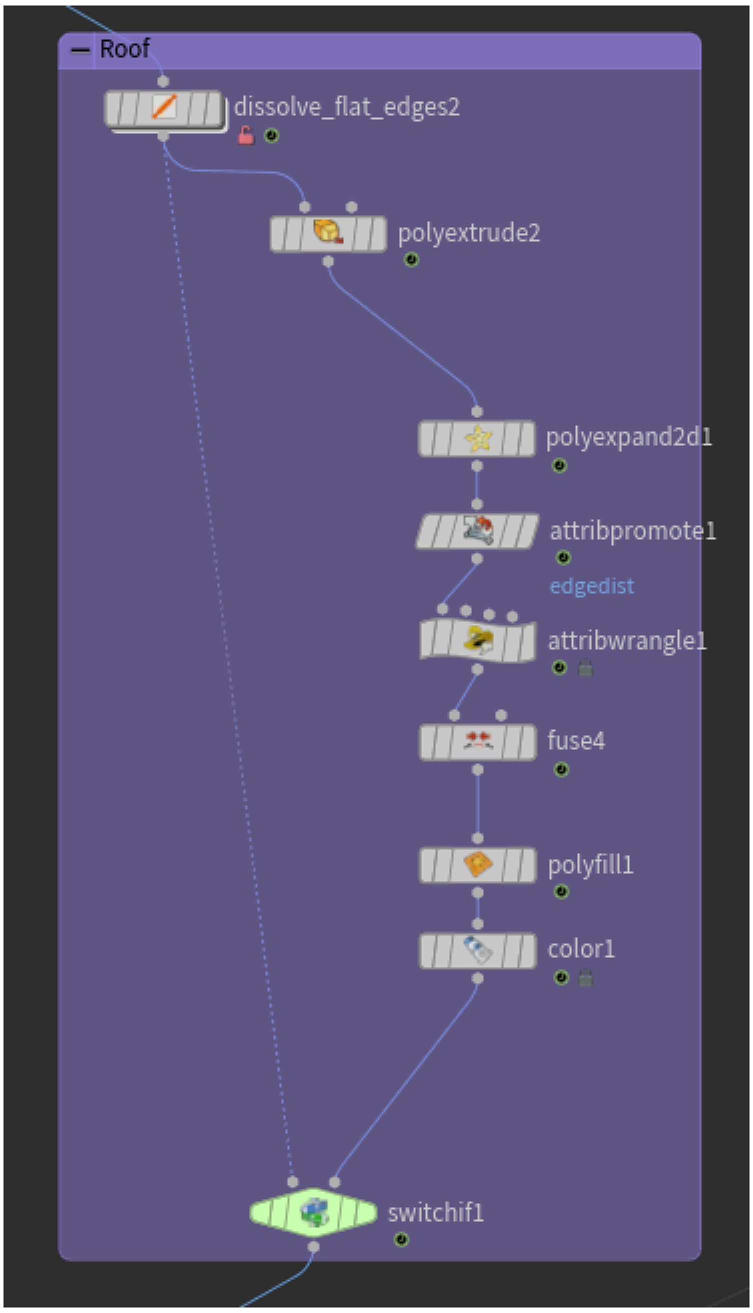
6.1 Select

- number *points*

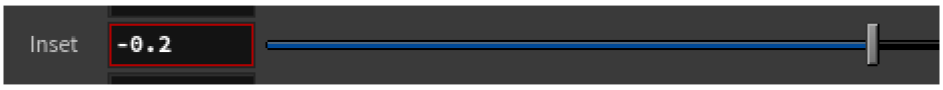
6.2 Offset

- Offset of the points

V. ROOF

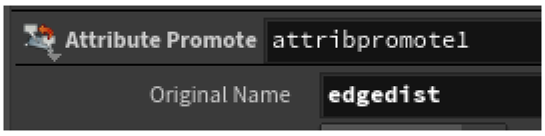


1. Dissolve_flat_edges
    - Removes edges below certain number creating smooth surface
2. Polyextrude



Inset -0.2

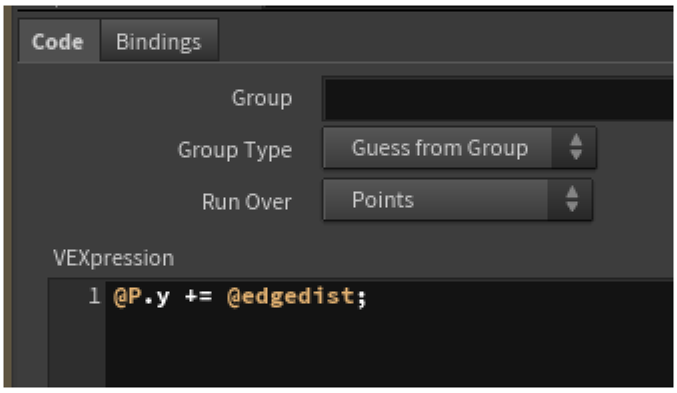
    - Yung ilalagpas nung roof kaya negative
    - This nodes can control the thicknesss but we wont do it in this node yet
3. Polyexpand
    - Expand to increase the size
4. Attribpromote



Attribute Promote attribpromote1

Original Name edgedist

    - Promotes points or primitive to polygon
5. Attribwrangle



Group:

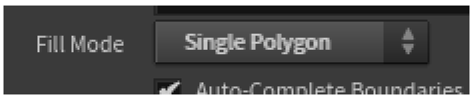
Group Type: Guess from Group

Run Over: Points

VEXpression

```
@P.y += @edgedist;
```

6. Fuse
    - Merge points
7. Polyfill

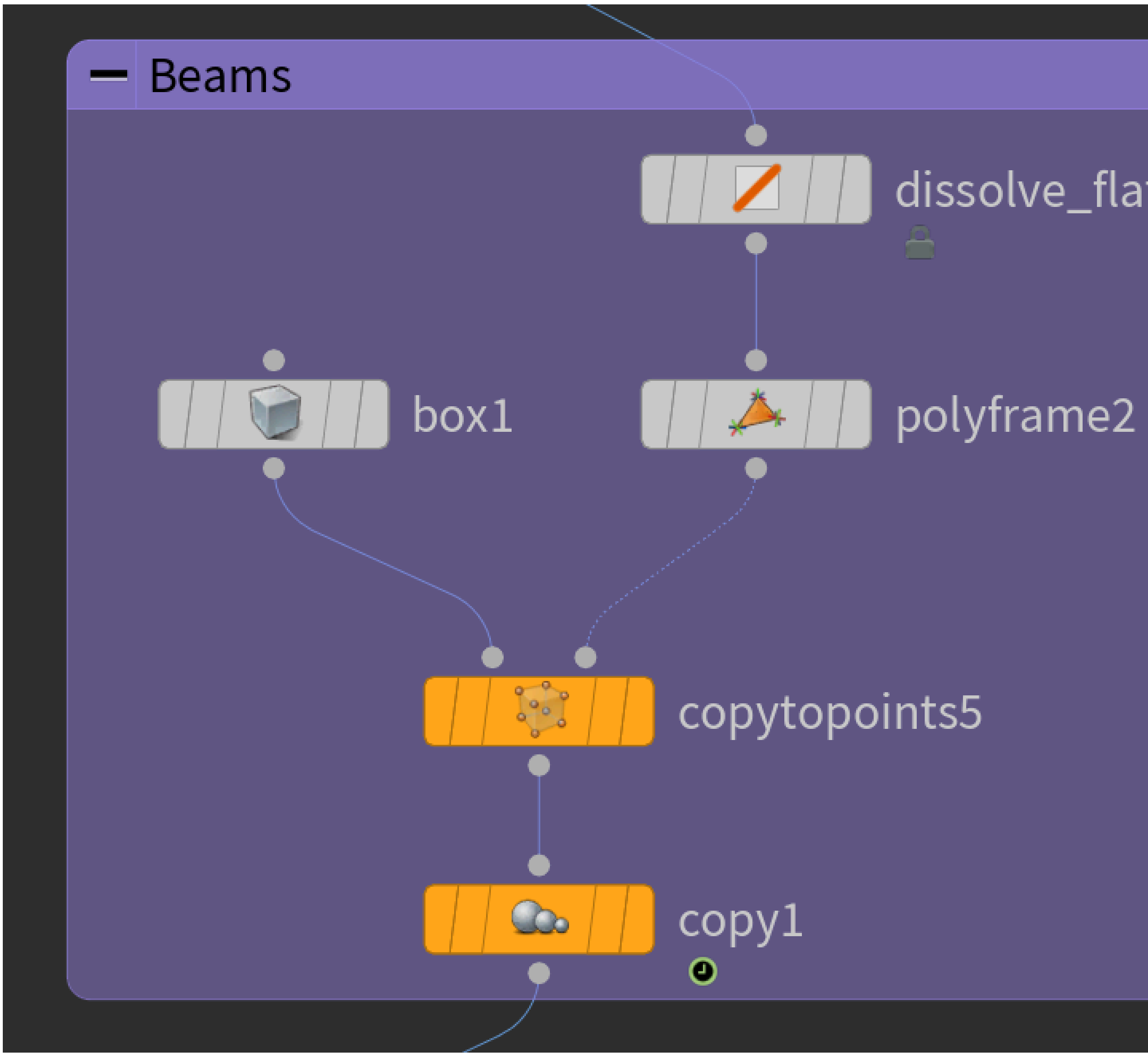


Fill Mode: Single Polygon

☑ Auto-Complete Boundaries

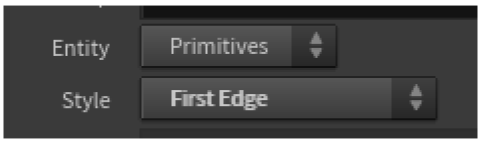
    - Fills holes in mesh to create new polygon
8. Color

VI. BEAMS



1. Dissolve_flat_edges
    - Removes edges below certain number creating smooth surface
2. Copytopoint

2.1 box
2.2 polyframe



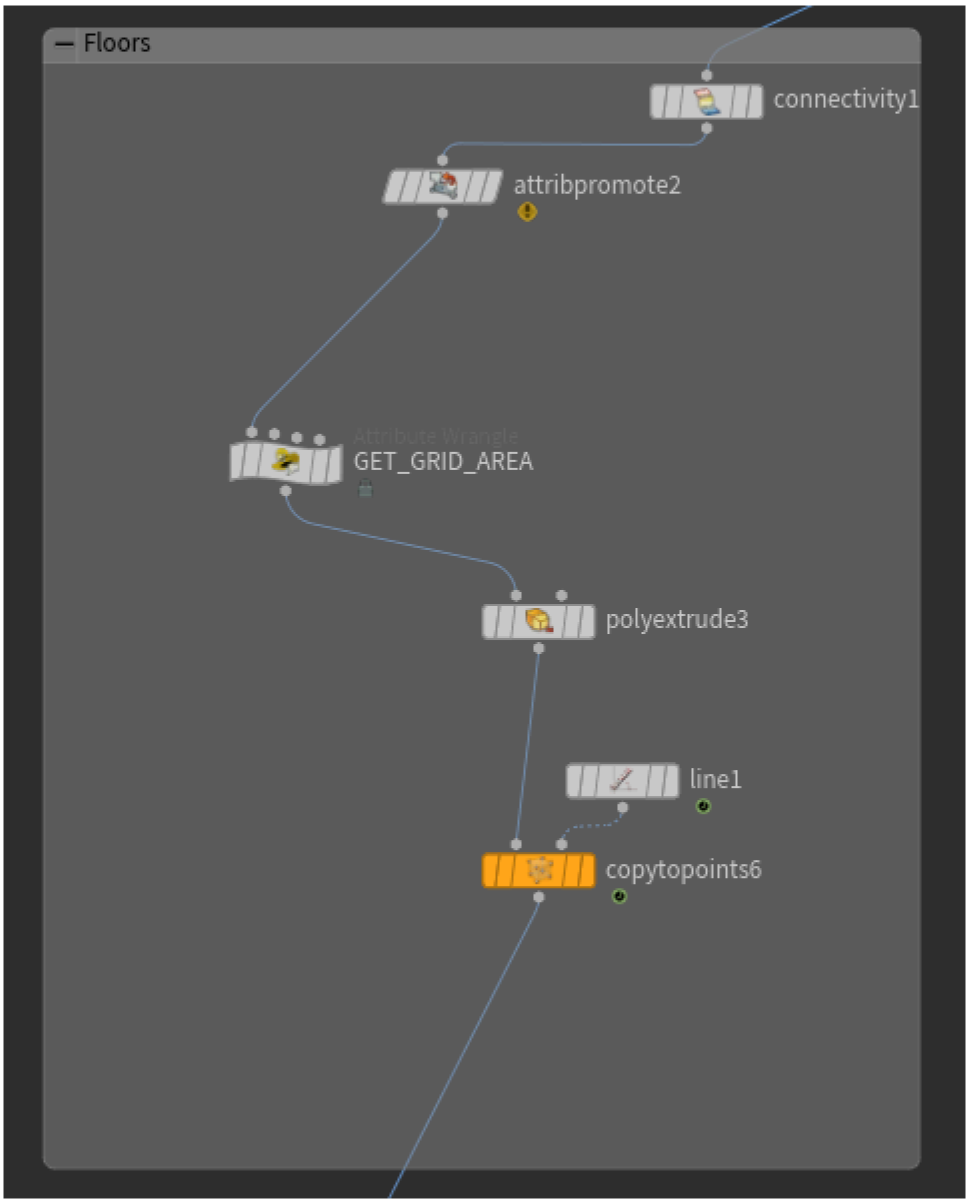
Entity Primitives

Style First Edge

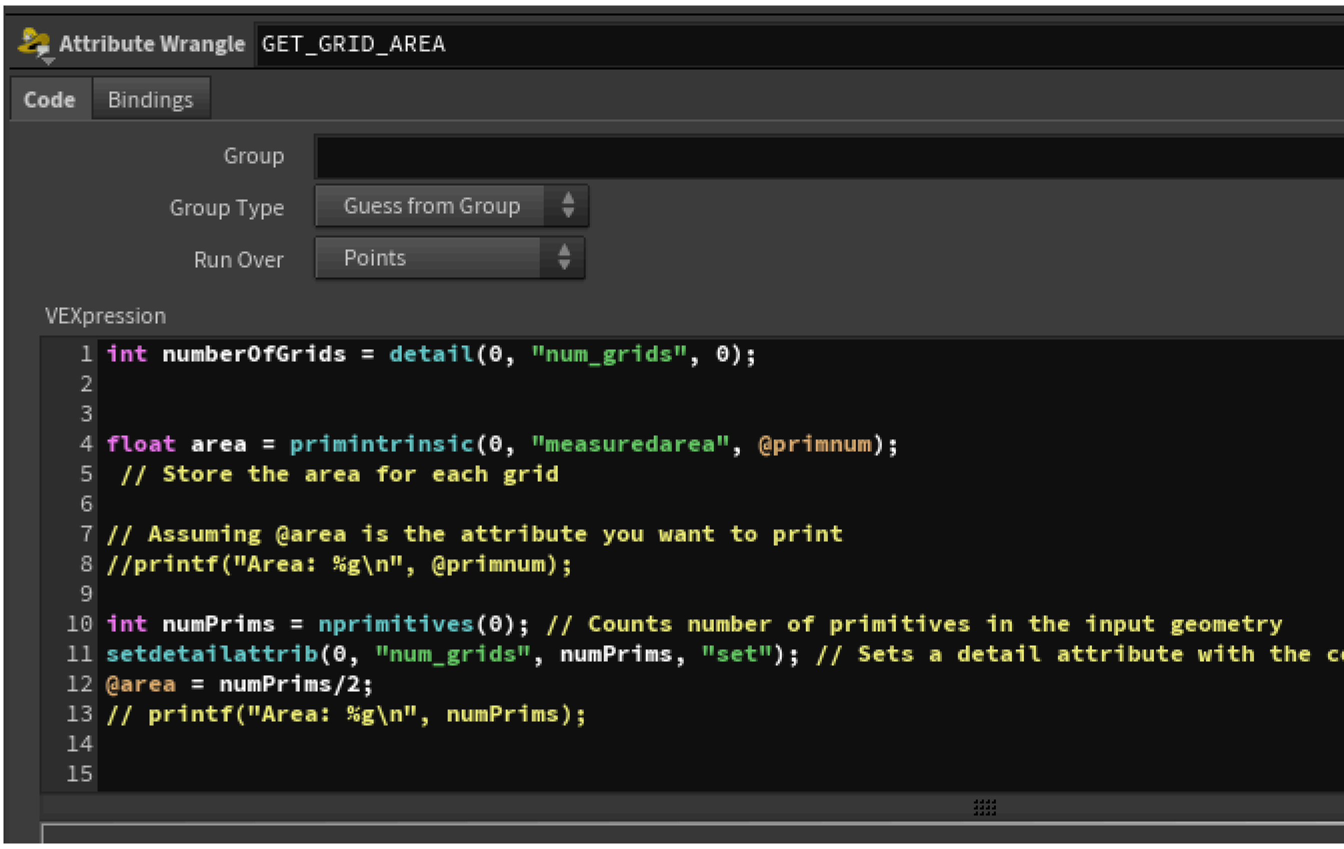
- Connected lines

3. Copy

VII. FLOORS



1. Connectivity
   - Analyze connections points or faces
2. Attributepromote
   - Promotes points to primitive or polygon
3. Get_grid_area



Attribute Wrangle GET_GRID_AREA

Group

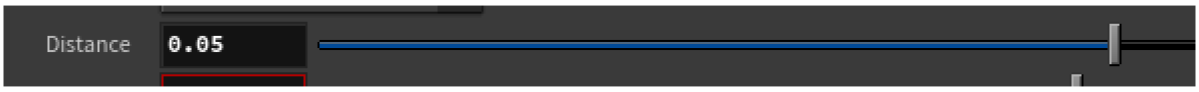
Group Type: Guess from Group

Run Over: Points

VEXpression

```
int numberOfGrids = detail(0, "num_grids", 0);


float area = primintrinsic(0, "measuredarea", @primnum);
 // Store the area for each grid

// Assuming @area is the attribute you want to print
//printf("Area: %g\n", @primnum);

int numPrims = nprimitives(0); // Counts number of primitives in the input geometry
setdetailattrib(0, "num_grids", numPrims, "set"); // Sets a detail attribute with the c
@area = numPrims/2;
// printf("Area: %g\n", numPrims);


```

4. Polyextrude

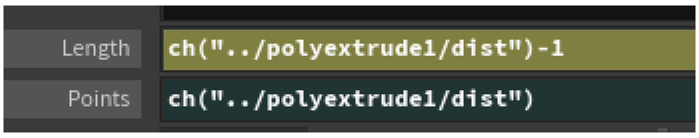


Distance: 0.05

   - Thickness of the *Floor*

5. copytopoints

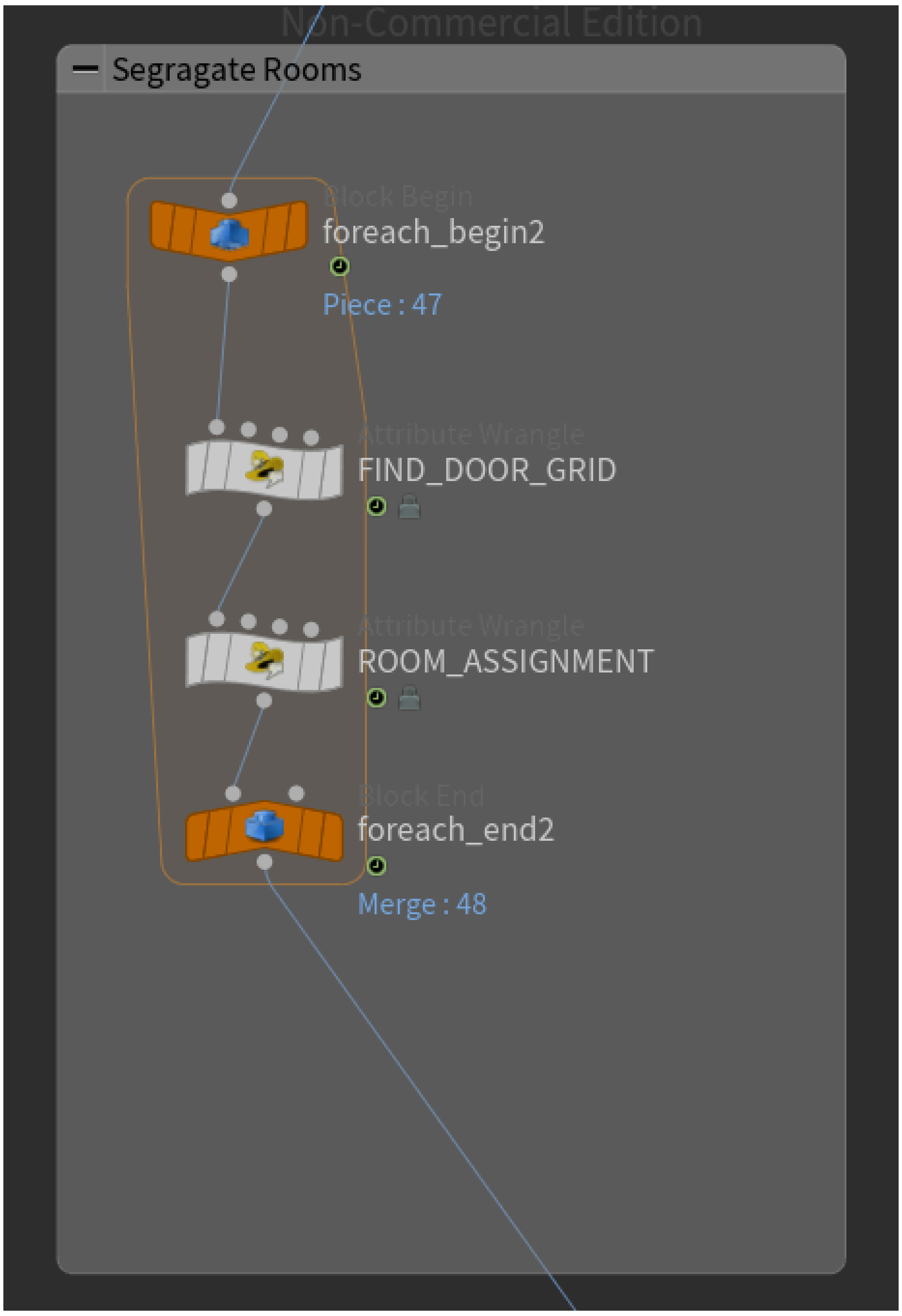
   4.1 line



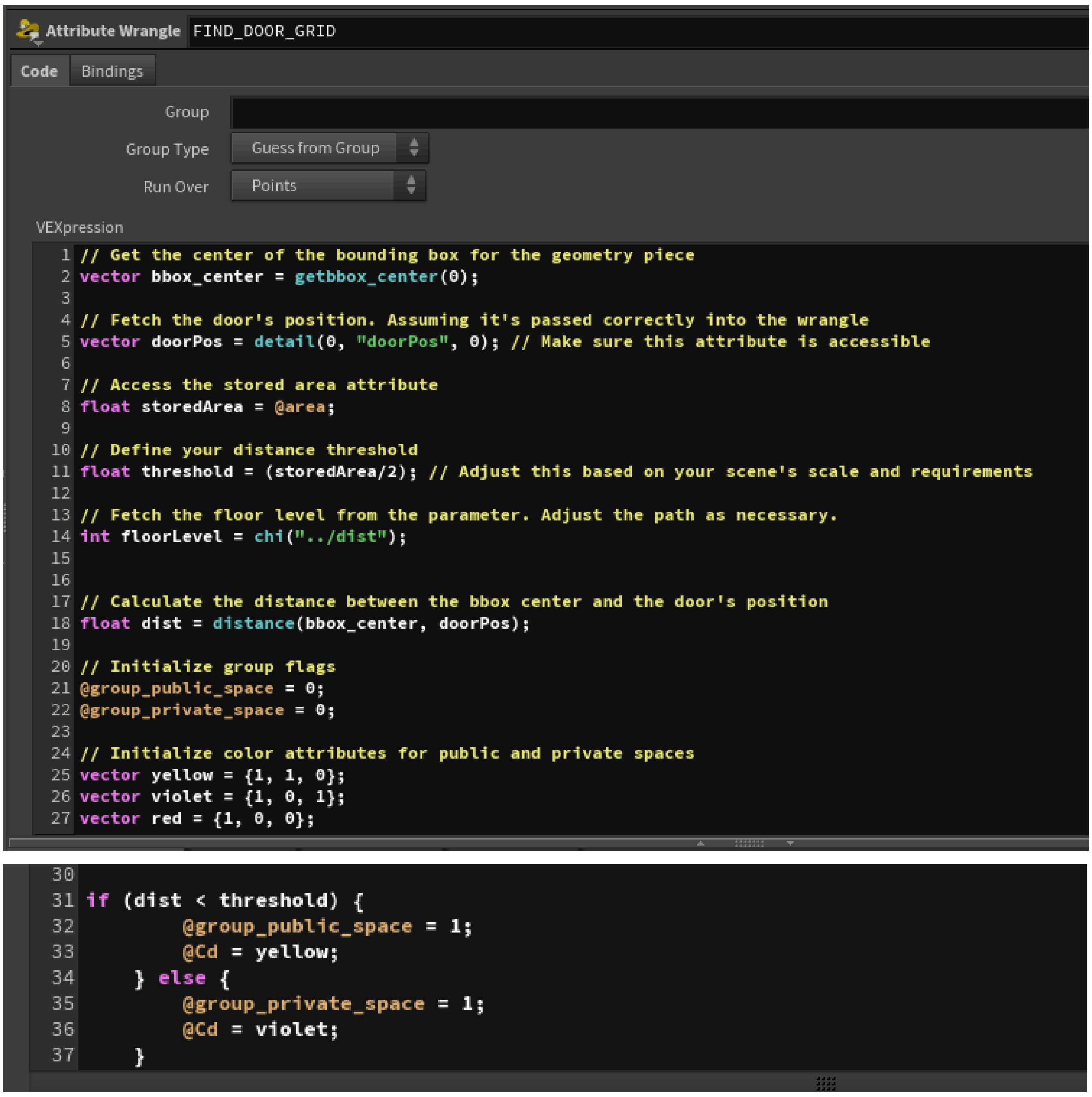
Length: ch("../polyextrude1/dist")-1

Points: ch("../polyextrude1/dist")

VIII. SEGREGATE ROOMS



1. Foereach
   1.1 find_door_grid



FIND_DOOR_GRID

Group

Group Type: Guess from Group

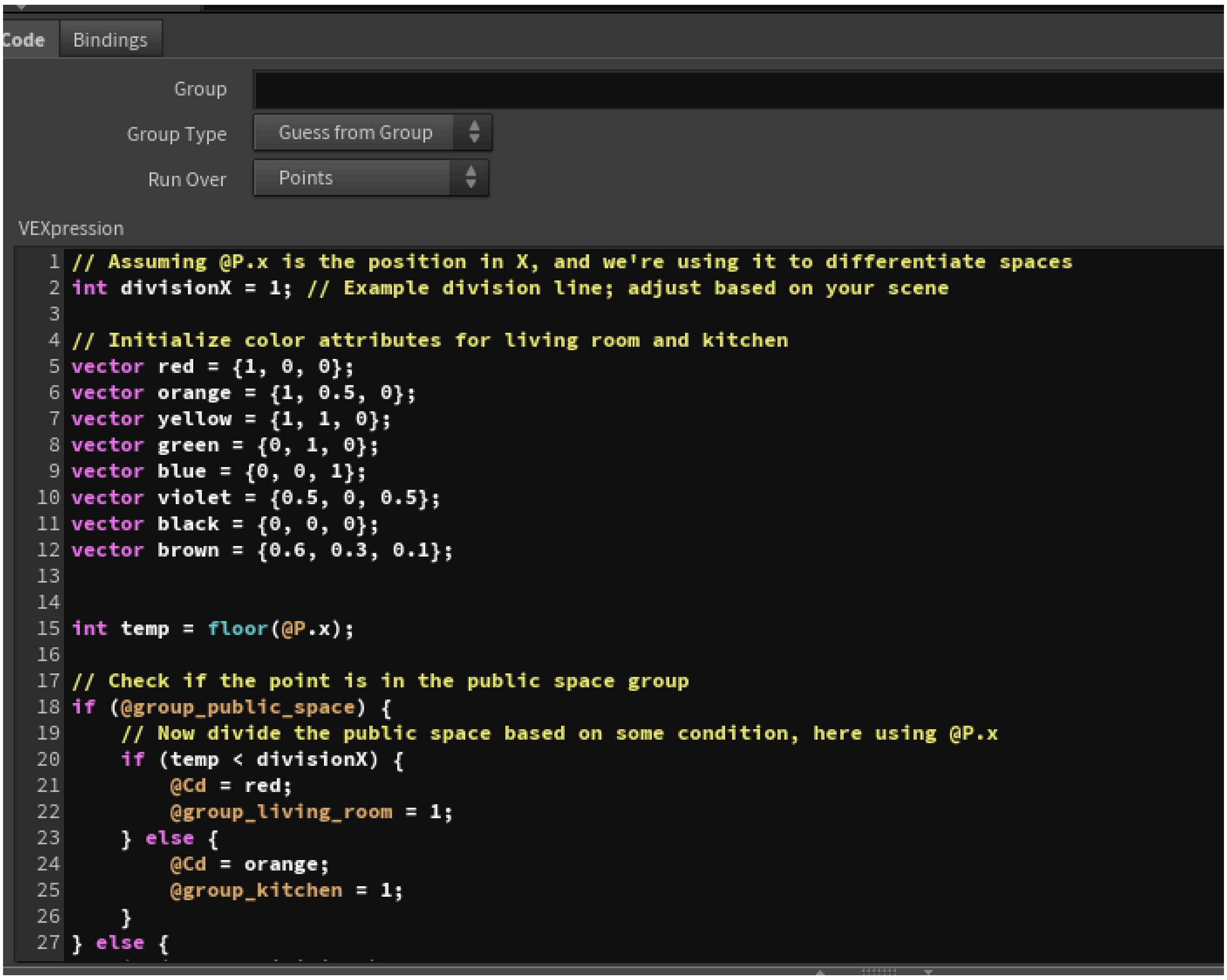
Run Over: Points

VEXpression

```
// Get the center of the bounding box for the geometry piece
vector bbox_center = getbbox_center(0);

// Fetch the door's position. Assuming it's passed correctly into the wrangle
vector doorPos = detail(0, "doorPos", 0); // Make sure this attribute is accessible

// Access the stored area attribute
float storedArea = @area;

// Define your distance threshold
float threshold = (storedArea/2); // Adjust this based on your scene's scale and requirements

// Fetch the floor level from the parameter. Adjust the path as necessary.
int floorLevel = chi("../dist");


// Calculate the distance between the bbox center and the door's position
float dist = distance(bbox_center, doorPos);

// Initialize group flags
@group_public_space = 0;
@group_private_space = 0;

// Initialize color attributes for public and private spaces
vector yellow = {1, 1, 0};
vector violet = {1, 0, 1};
vector red = {1, 0, 0};
```

```

if (dist < threshold) {
        @group_public_space = 1;
        @Cd = yellow;
    } else {
        @group_private_space = 1;
        @Cd = violet;
    }
```

1.2 room_assigment



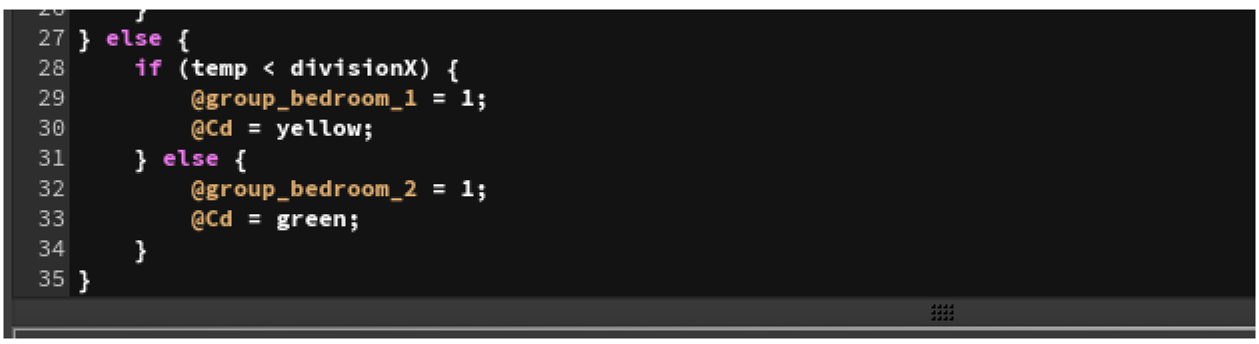
Group

Group Type: Guess from Group

Run Over: Points

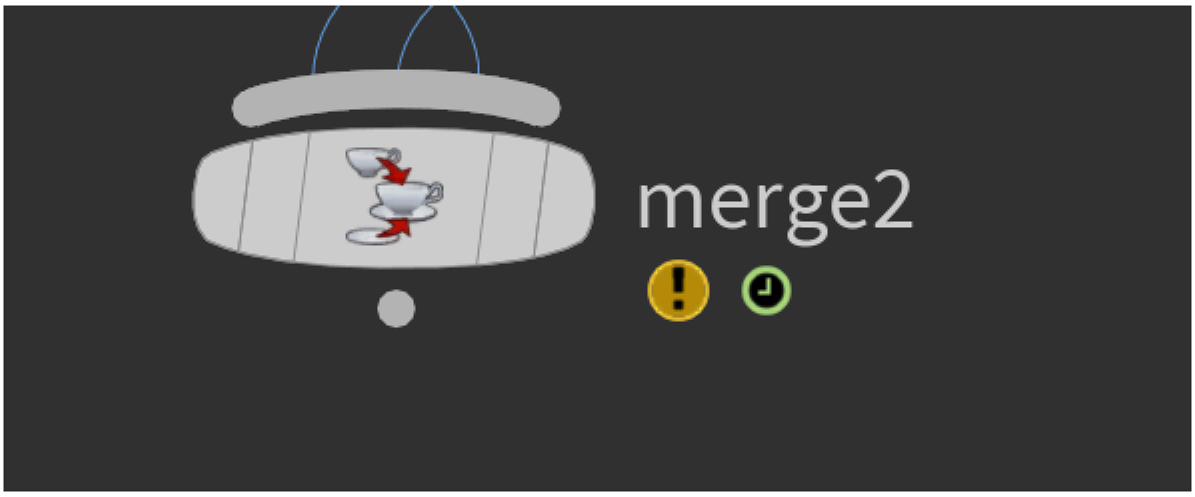
VEXpression

```
// Assuming @P.x is the position in X, and we're using it to differentiate spaces
int divisionX = 1; // Example division line; adjust based on your scene

// Initialize color attributes for living room and kitchen
vector red = {1, 0, 0};
vector orange = {1, 0.5, 0};
vector yellow = {1, 1, 0};
vector green = {0, 1, 0};
vector blue = {0, 0, 1};
vector violet = {0.5, 0, 0.5};
vector black = {0, 0, 0};
vector brown = {0.6, 0.3, 0.1};


int temp = floor(@P.x);

// Check if the point is in the public space group
if (@group_public_space) {
    // Now divide the public space based on some condition, here using @P.x
    if (temp < divisionX) {
        @Cd = red;
        @group_living_room = 1;
    } else {
        @Cd = orange;
        @group_kitchen = 1;
    }
} else {
```

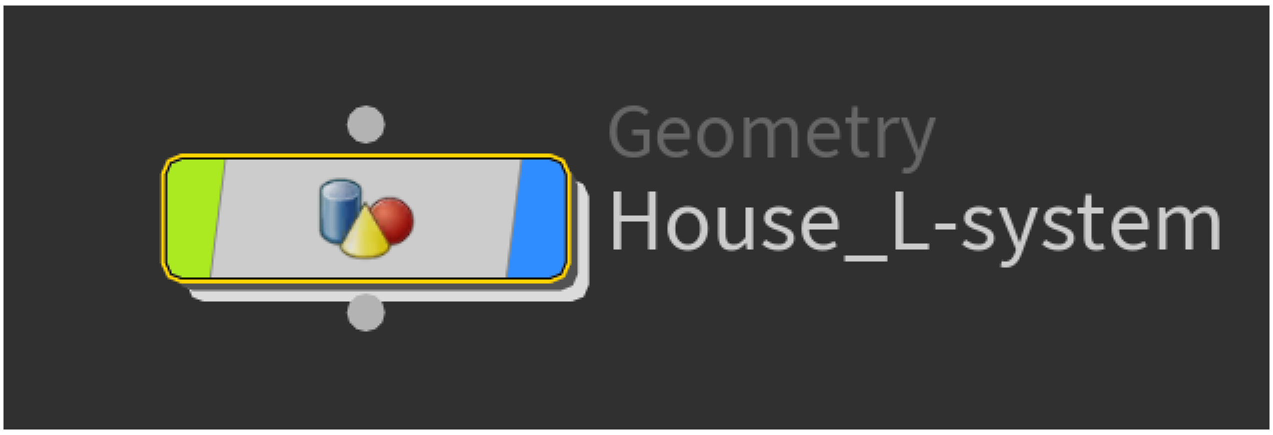


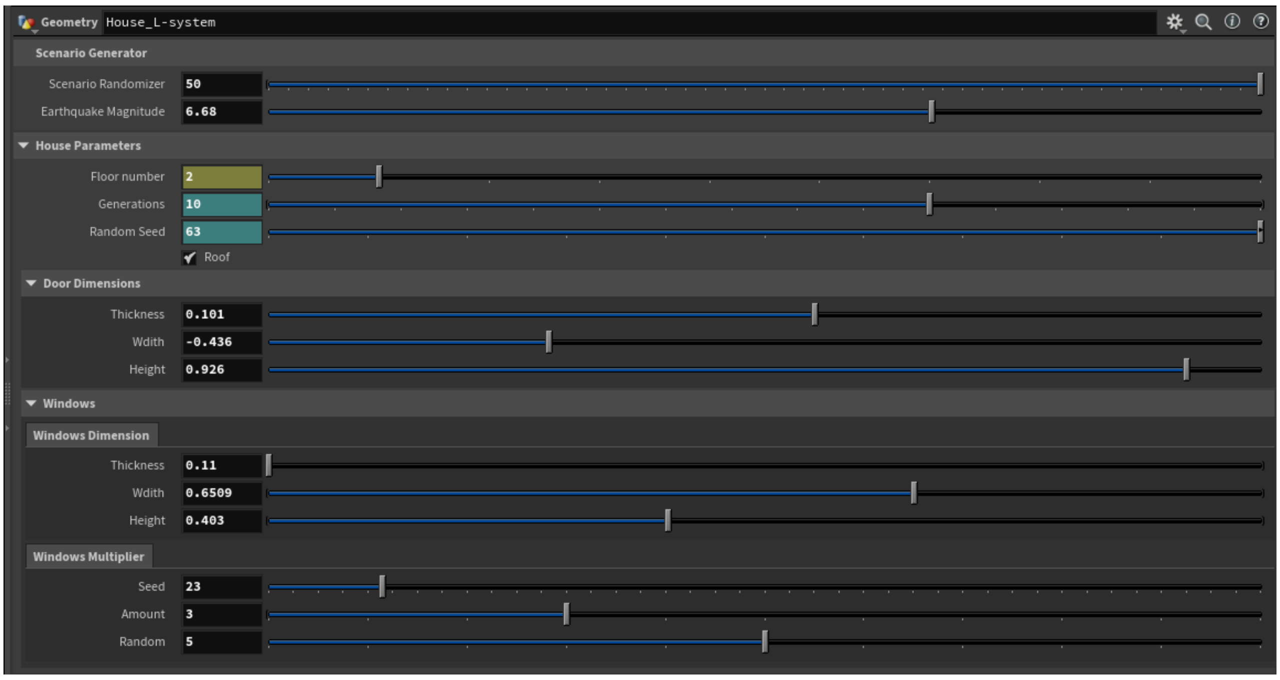
```
} else {
    if (temp < divisionX) {
        @group_bedroom_1 = 1;
        @Cd = yellow;
    } else {
        @group_bedroom_2 = 1;
        @Cd = green;
    }
}
```

IX. MERGE ALL



X. FINAL GEO NODE





Geometry House_L-system

Scenario Generator

Scenario Randomizer 50

Earthquake Magnitude 6.68

House Parameters

Floor number 2

Generations 10

Random Seed 63

☑ Roof

Door Dimensions

Thickness 0.101

Wdith -0.436

Height 0.926

Windows

Windows Dimension

Thickness 0.11

Wdith 0.6509

Height 0.403

Windows Multiplier

Seed 23

Amount 3

Random 5

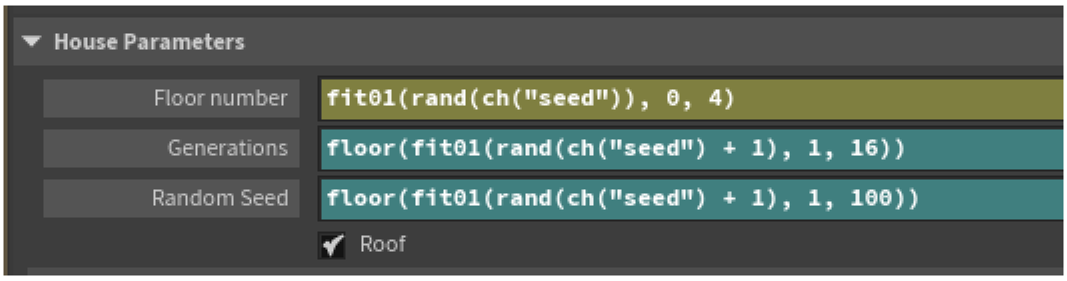
1. Scenario Randomizer

    1.1 Scenario Randomizer

    - Randomizes *Floor Number, Generations,* and *Random Seed*

    1.2 Earthquake Magnitude

2. House Parameters



House Parameters

Floor number `fit01(rand(ch("seed")), 0, 4)`

Generations `floor(fit01(rand(ch("seed") + 1), 1, 16))`

Random Seed `floor(fit01(rand(ch("seed") + 1), 1, 100))`

☑ Roof

    2.1 Floor number

    2.2 Generations

    2.3 Random seed

3. Door dimensions
4. Windows