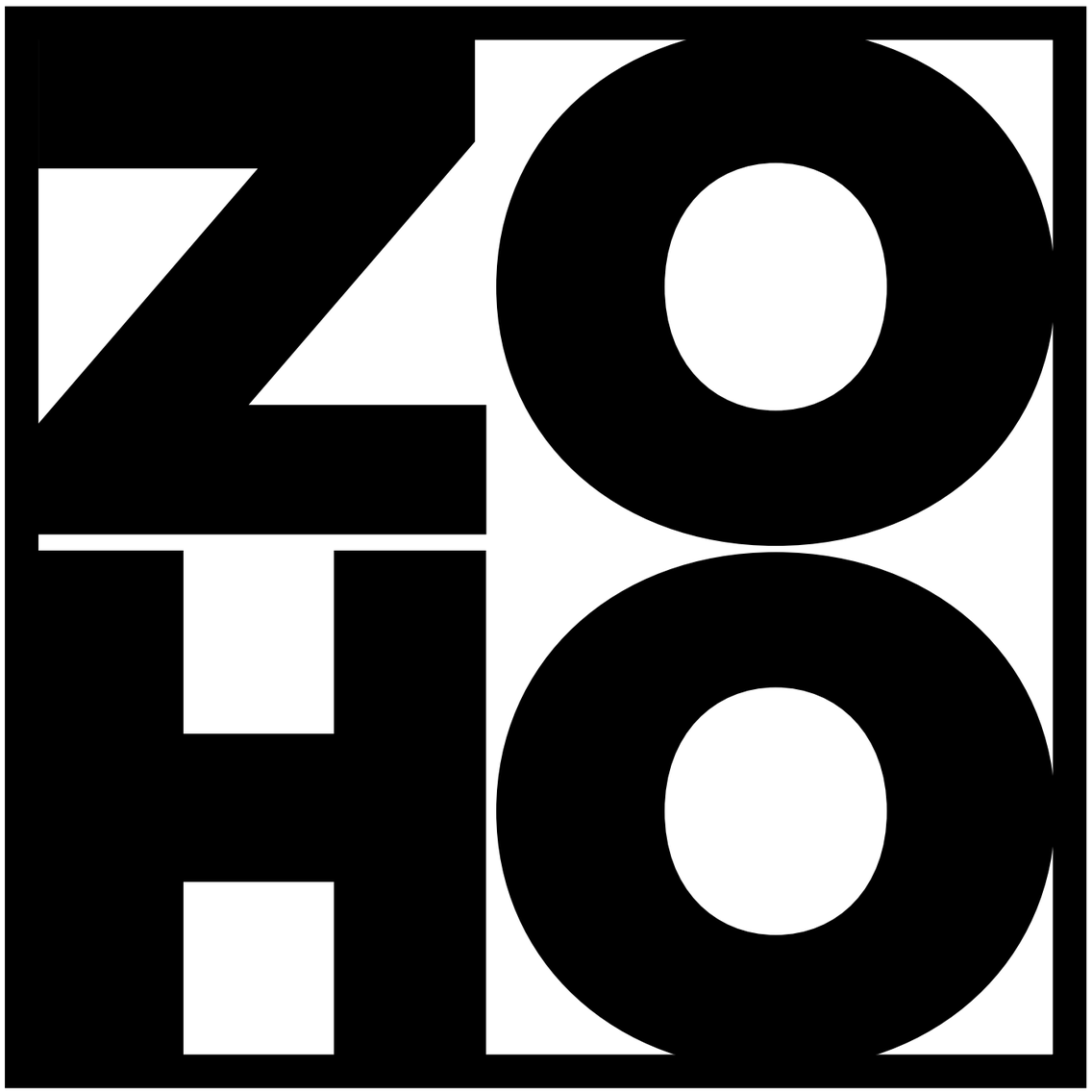


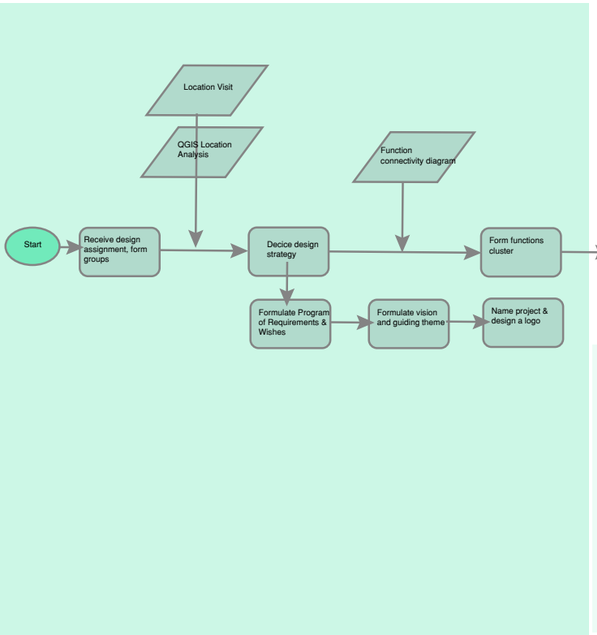
<art_house>

Pim Voermans
Marieke van Esch
Laurens van Rijssel
Valerie van den Broek

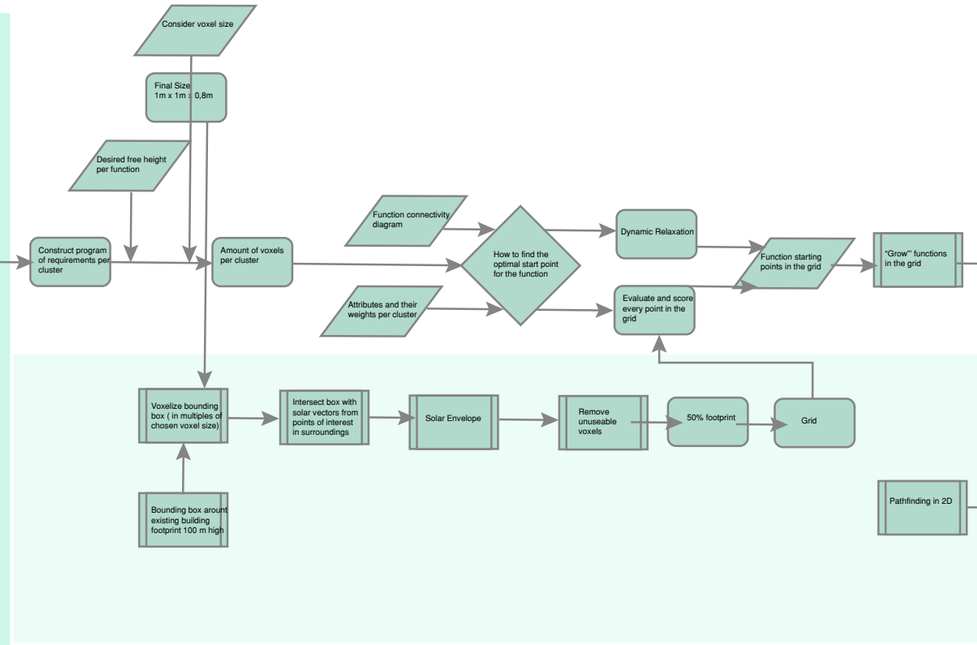
Table of contents

Flowchart

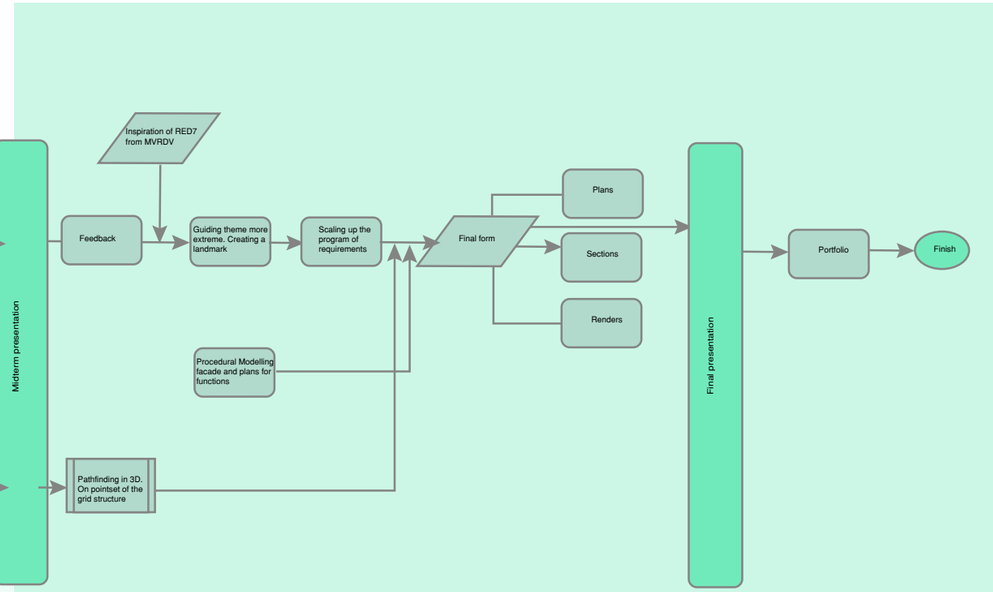
Step 1



Step 2

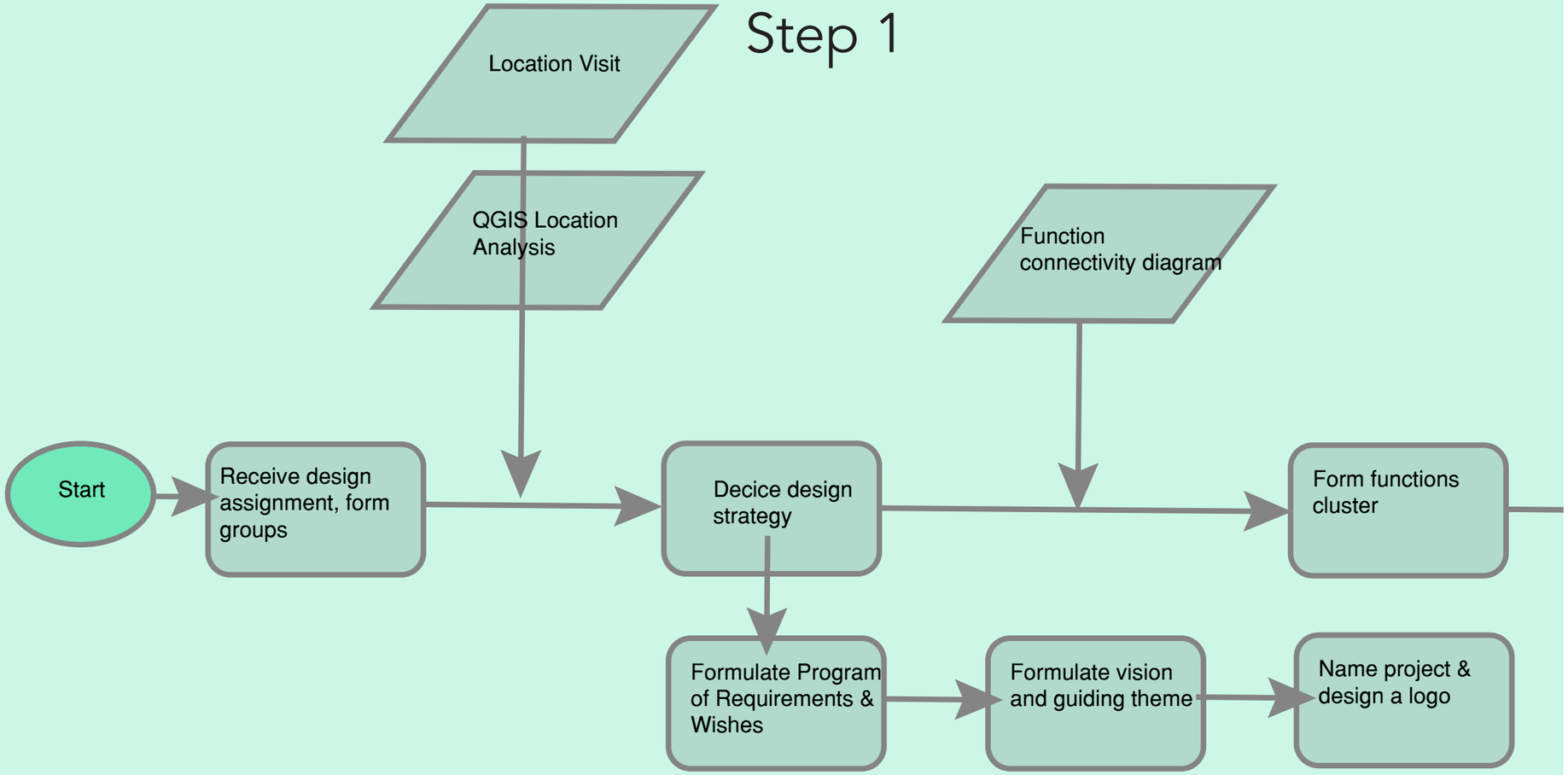


Step 3



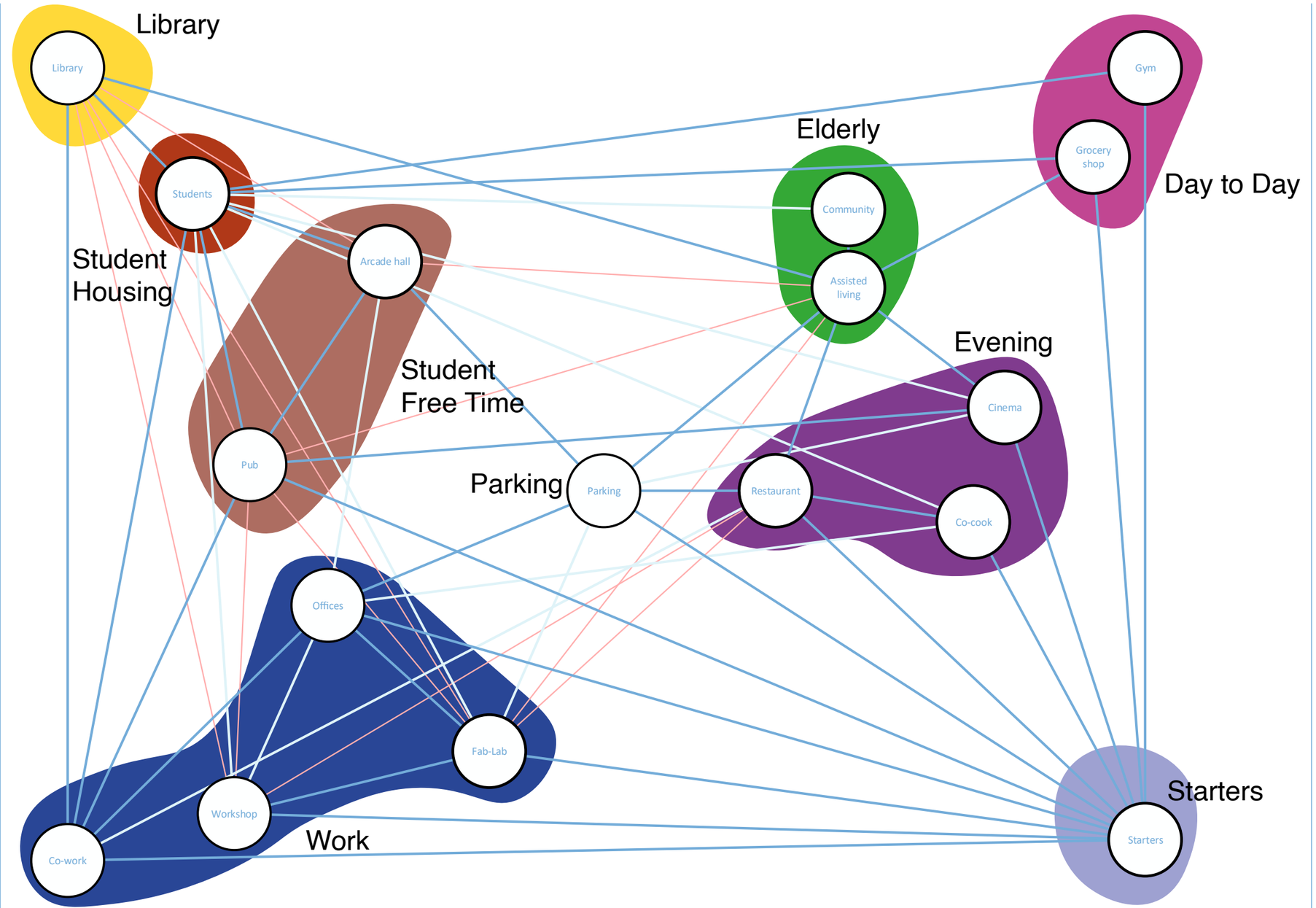
Flowchart

Step 1



Area

Connectivity



Clusters

Library
Starters
Student Housing

Student Freetime:
- Arcade
- Pub

Work:
- Offices
- Co-work
- Fablab
- Workshop

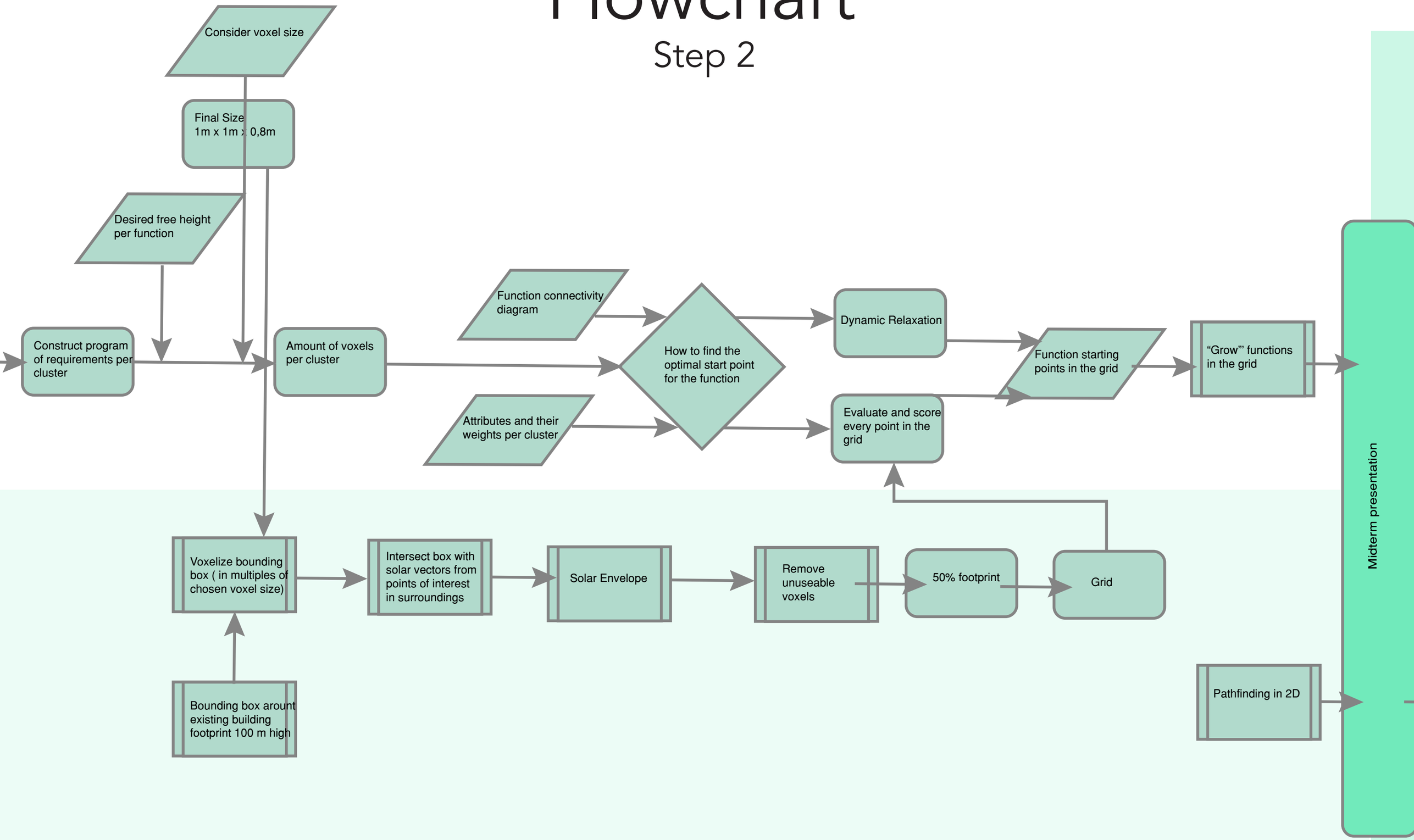
Evening:
- Restaurant
- Co-Cook
- Cinema

Day-To-Day:
- Shop
- Gym

Elderly:
- Assisted living
- Community Centre

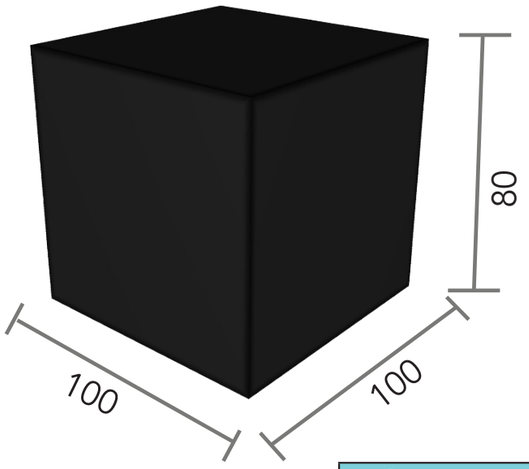
Flowchart

Step 2



Voxels

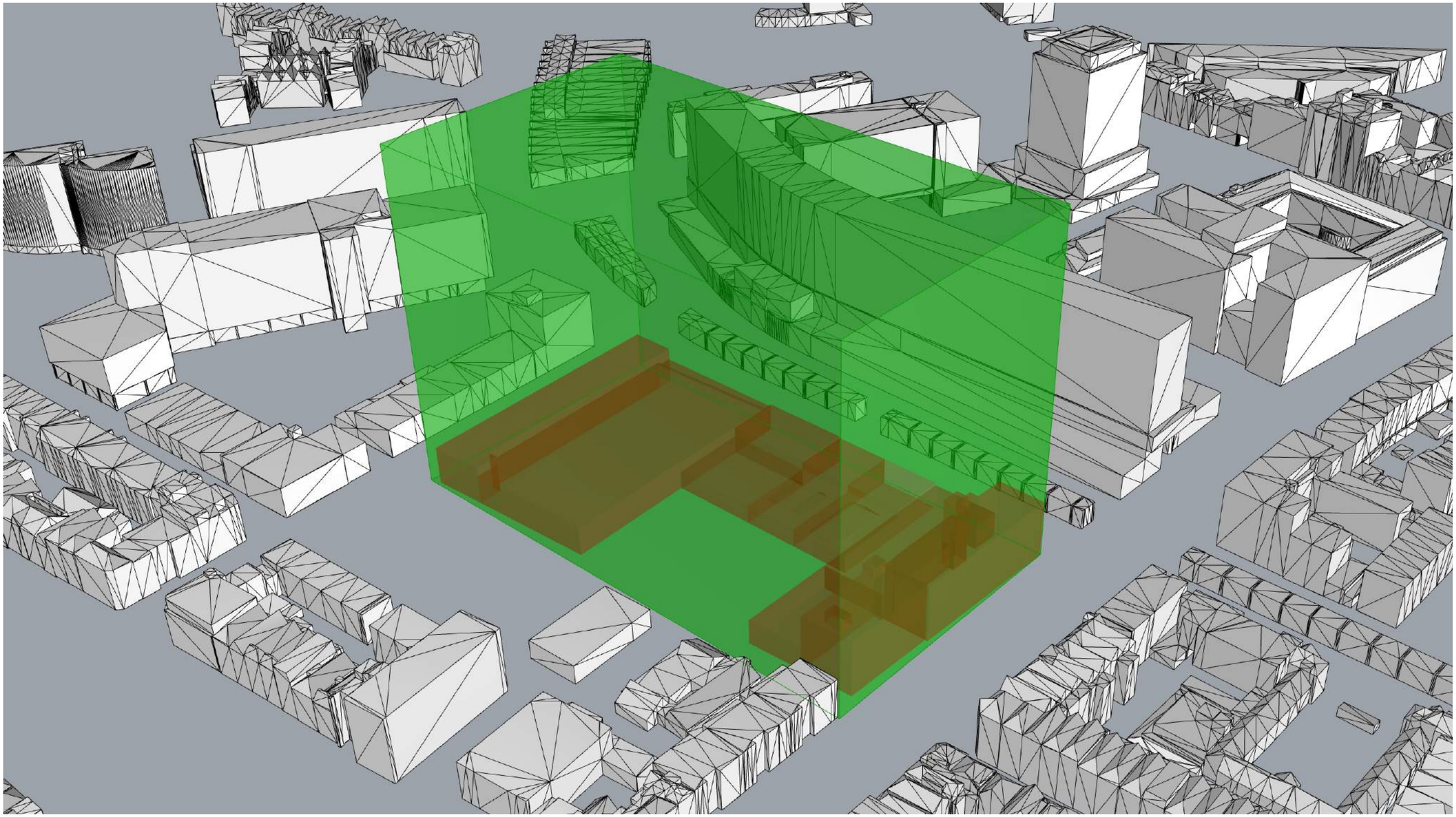
Size



Area and amount of voxels

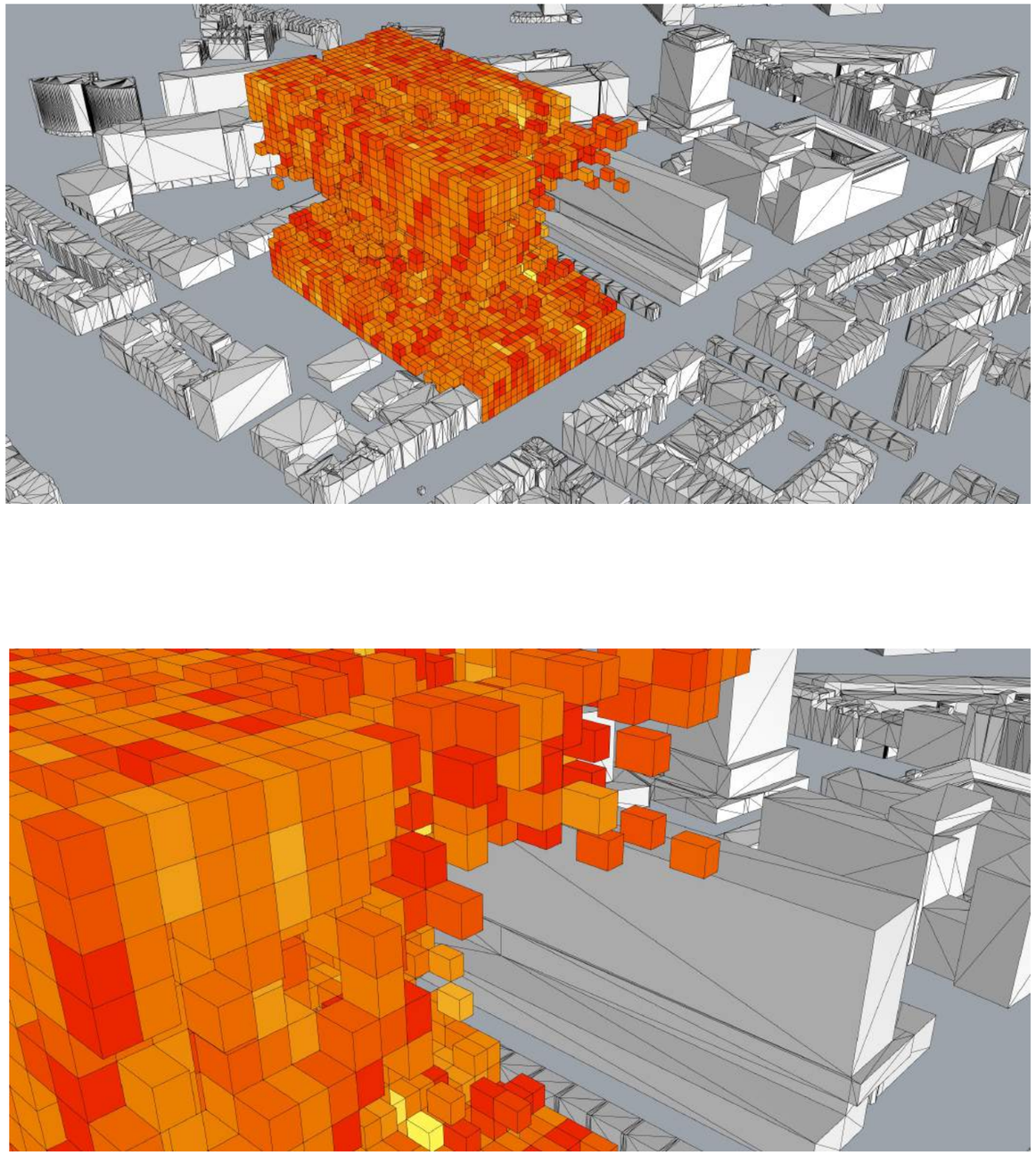
Cluster	Area (m2)	Voxels (Amount)	4x Voxel size (Amount)
Student	1200	4800	75
Student Freetime	150	600	11
Work	2950	11800	187
Evening	1260	5040	80
Day-to-day	800	3200	50
Elderly	2000	8000	125
Starters	3600	14400	225
Library	200	800	13
Parking	2250	9000	141

Bounding box

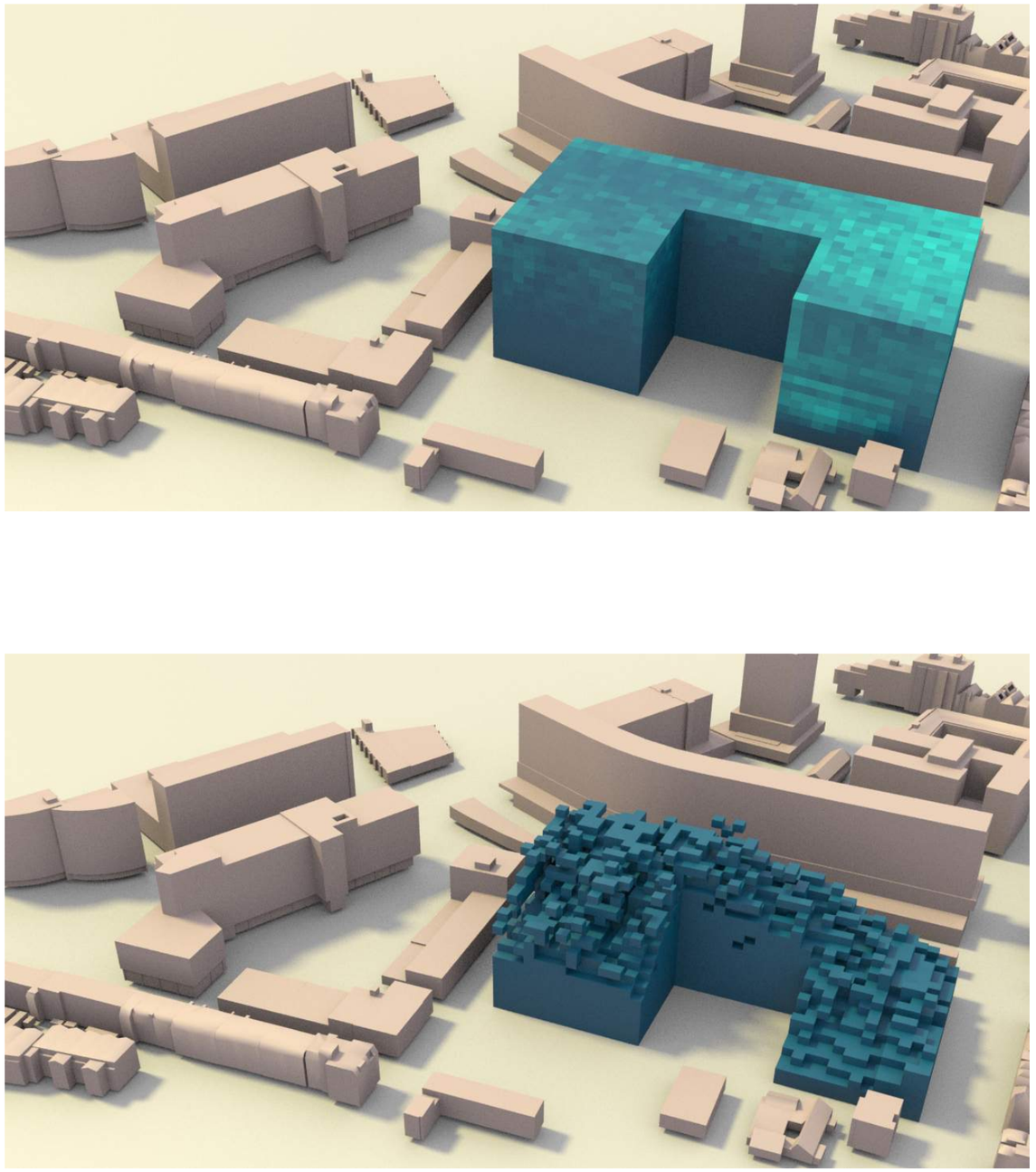


Solar Envelope

Rhino



Houdini



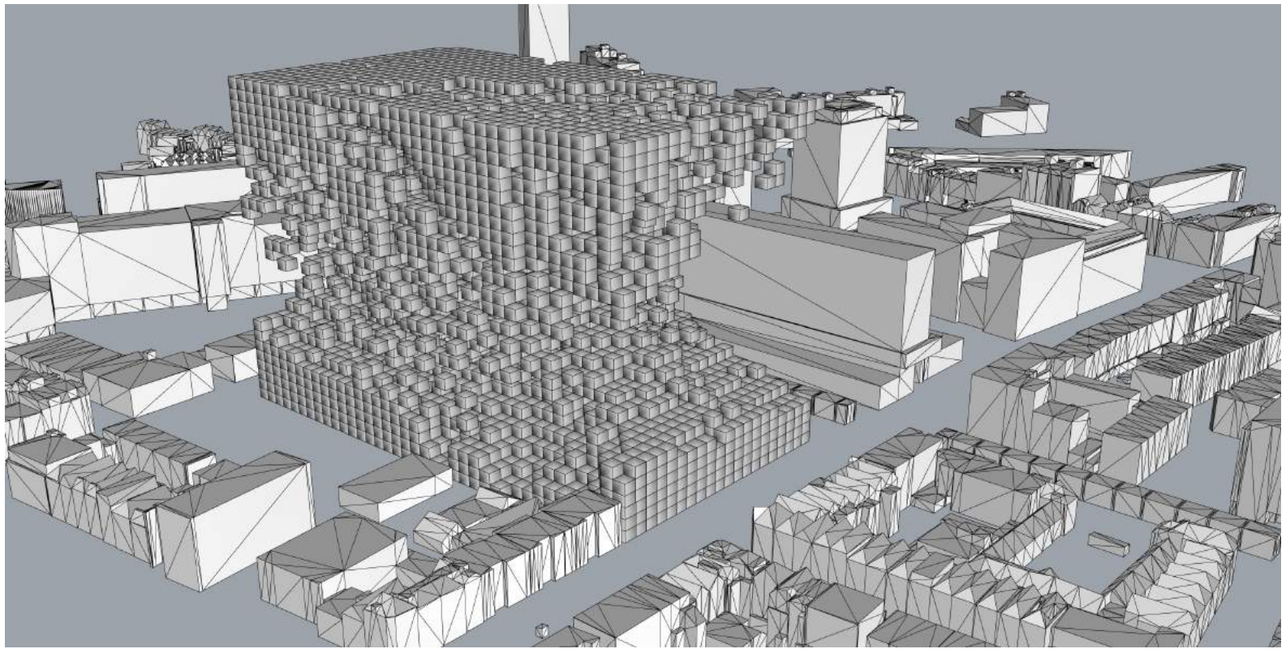
Culling unfunctional points

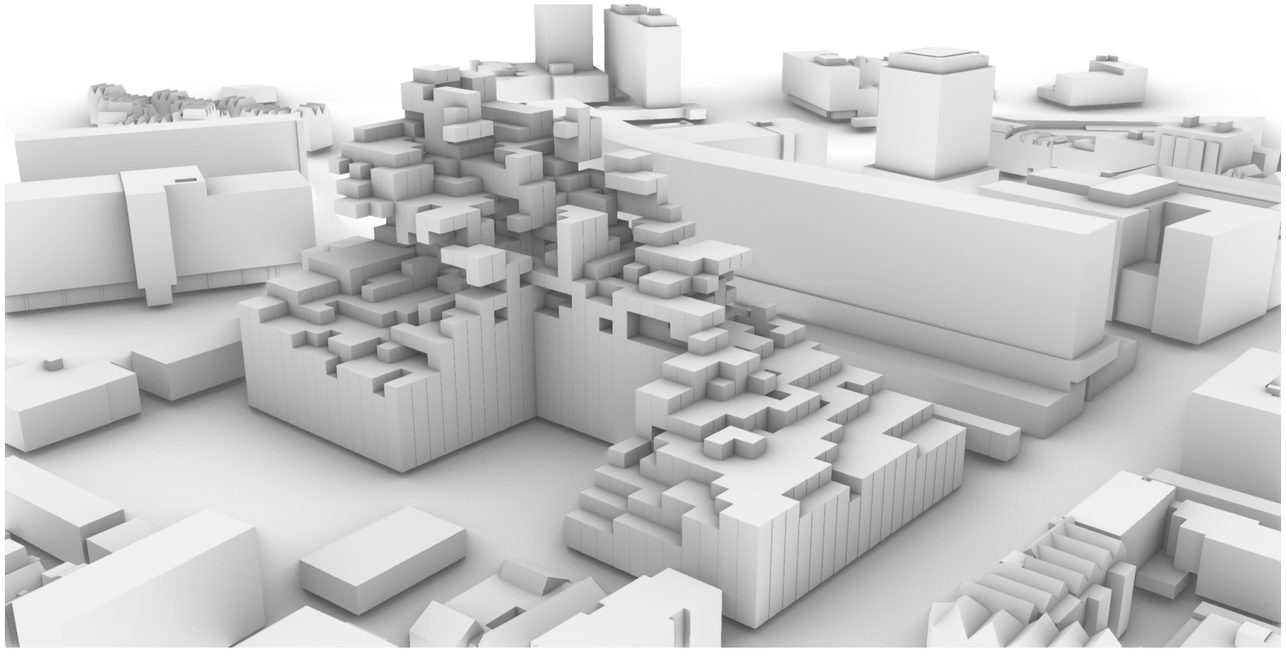
Pseudocode

```
//Deleting Unfunctional Voxels

put points of Solar Envelope in list

for each point in list
      create a list of distances to all the other points
if the distance between to points is 0<distance<4.1:
            neighbours += 1
      if neighbours > 2
            add point to new list
```

this function is repeatable to get a stricter culled solar envelope

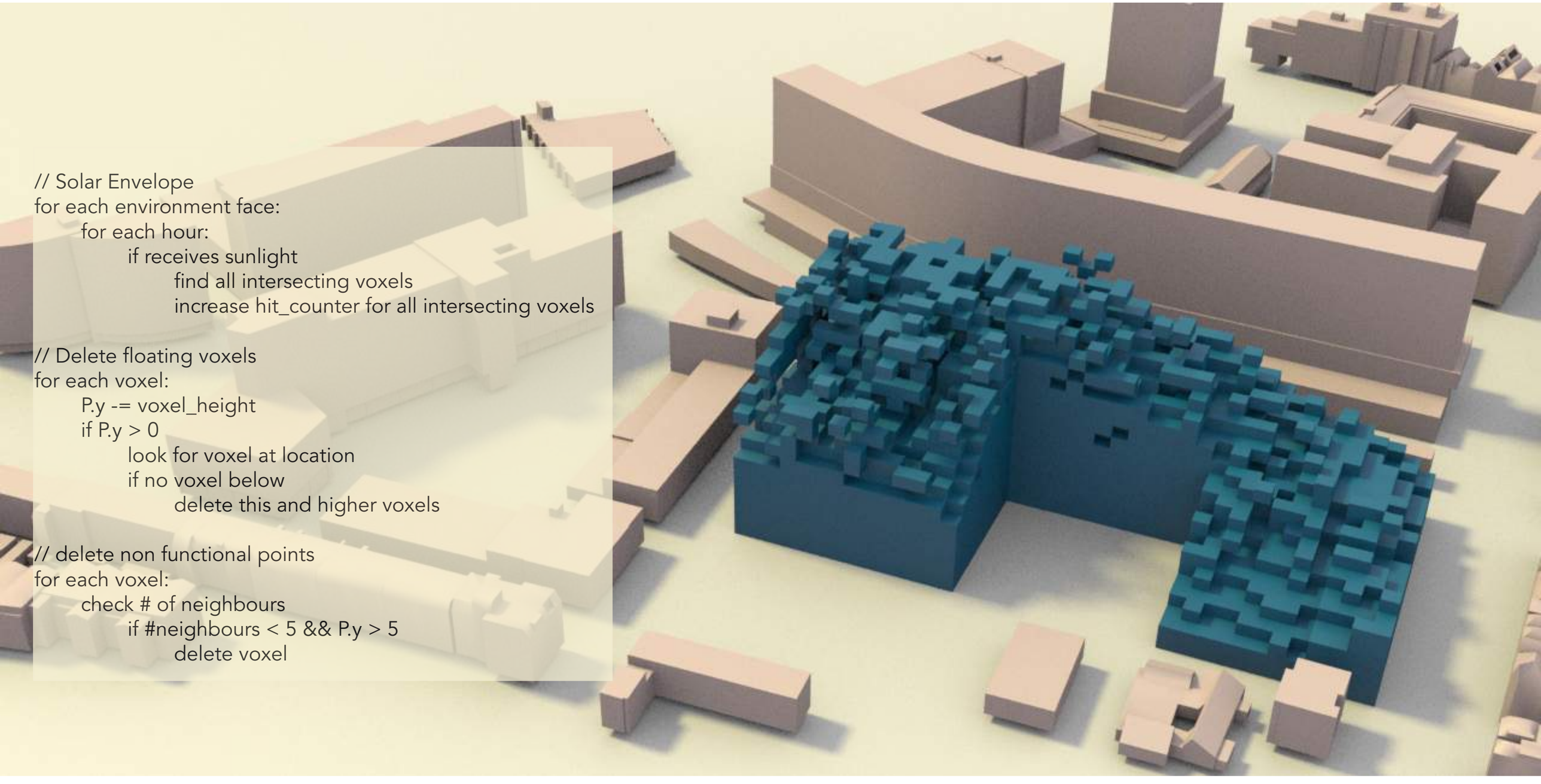
Rhino





Culling unfunctional points

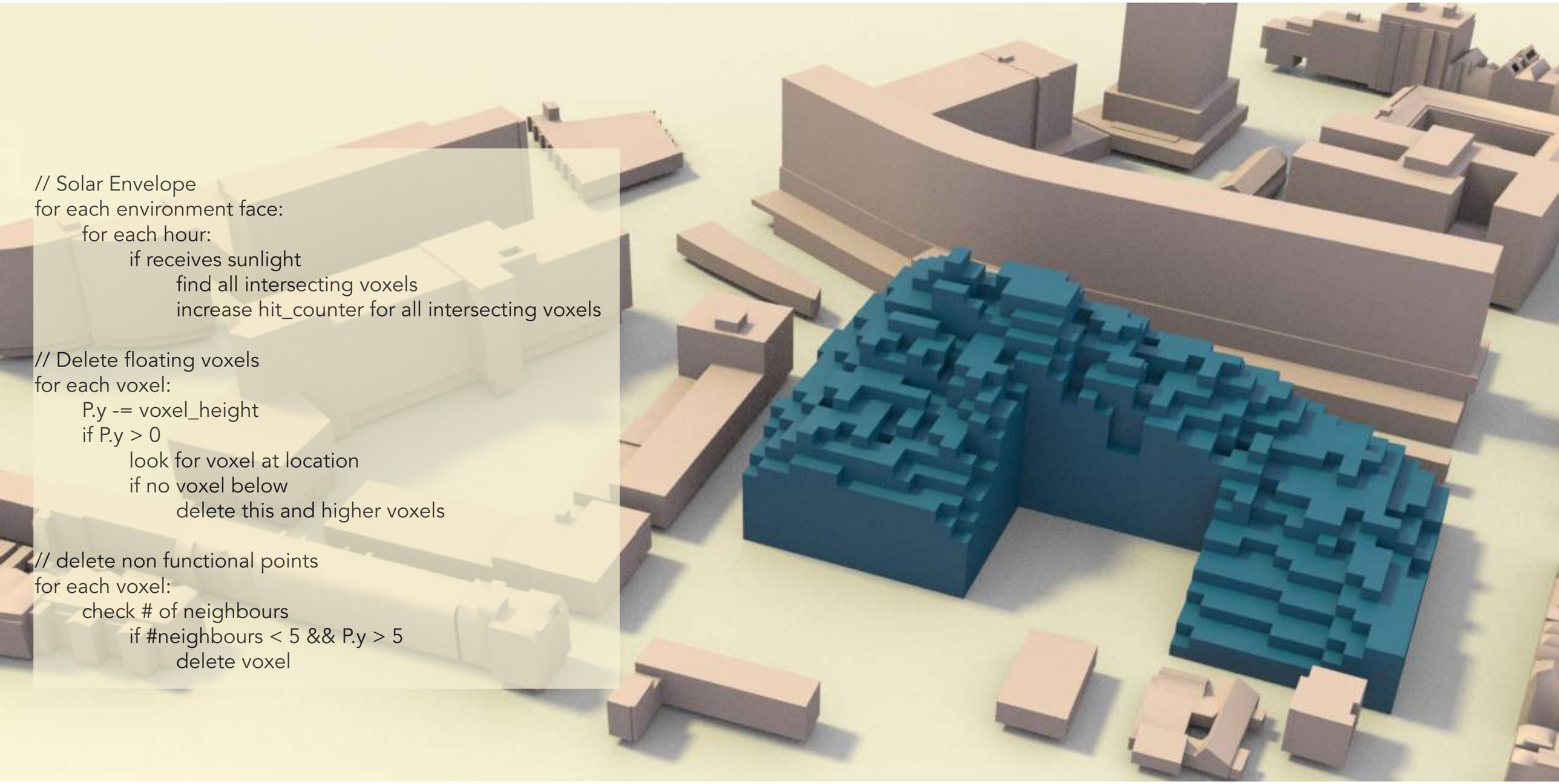
Houdini



```
// Solar Envelope
for each environment face:
        for each hour:
                if receives sunlight
                        find all intersecting voxels
                        increase hit_counter for all intersecting voxels

// Delete floating voxels
for each voxel:
        P.y -= voxel_height
        if P.y > 0
                look for voxel at location
                if no voxel below
                        delete this and higher voxels

// delete non functional points
for each voxel:
        check # of neighbours
                if #neighbours < 5 && P.y > 5
                        delete voxel
```

Culling unfunctional points

Houdini

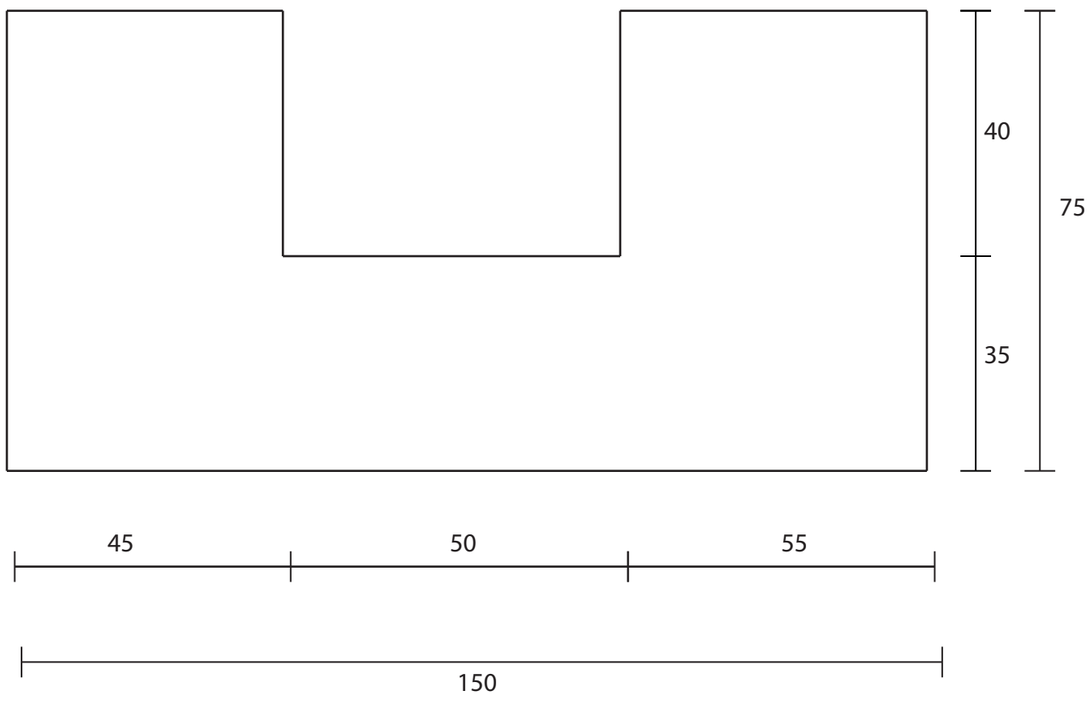


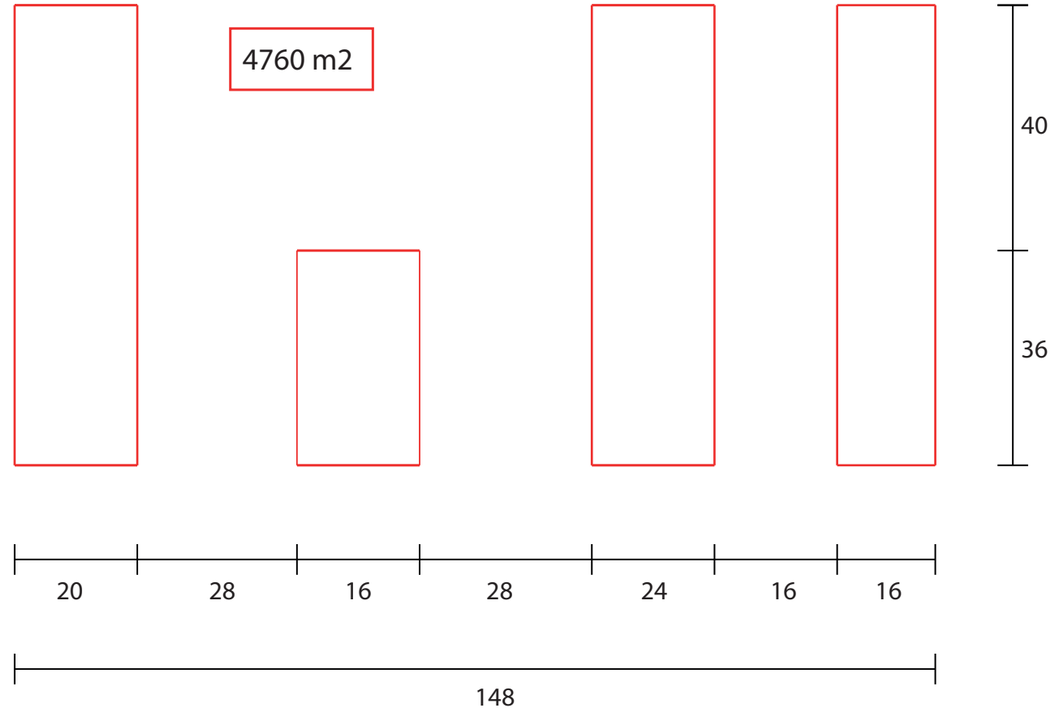
```
// Solar Envelope
for each environment face:
        for each hour:
                if receives sunlight
                        find all intersecting voxels
                        increase hit_counter for all intersecting voxels

// Delete floating voxels
for each voxel:
        P.y -= voxel_height
        if P.y > 0
                look for voxel at location
                if no voxel below
                        delete this and higher voxels

// delete non functional points
for each voxel:
        check # of neighbours
                if #neighbours < 5 && P.y > 5
                        delete voxel
```

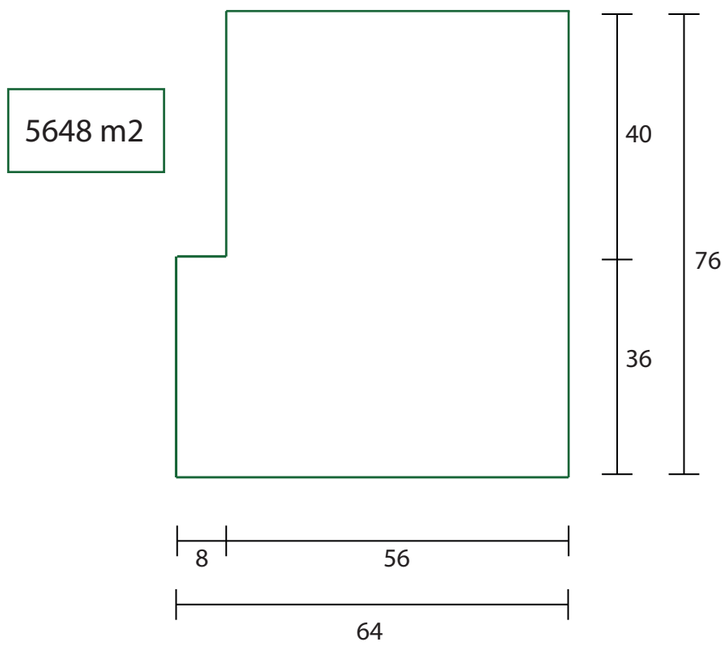
DESIGN INTERMEZZO

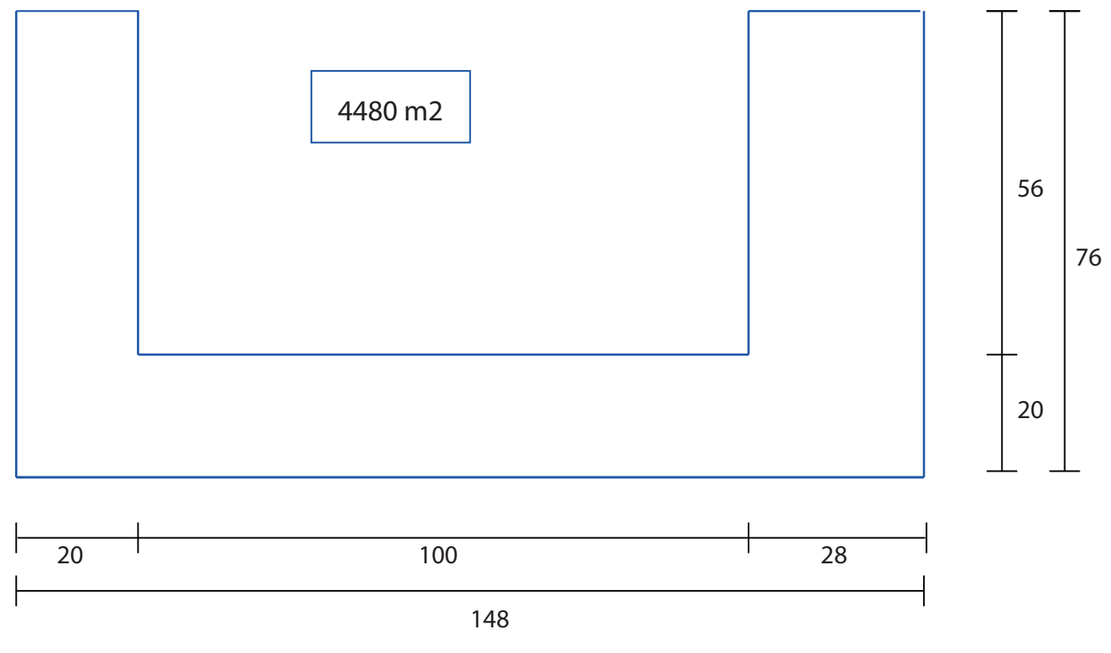
DESIGN INTERMEZZO

Footprints


40
75
35
45
50
55
150


4760 m2
40
36
20
28
16
28
24
16
16
148


5648 m2
40
76
36
8
56
64


4480 m2
56
76
20
20
100
28
148

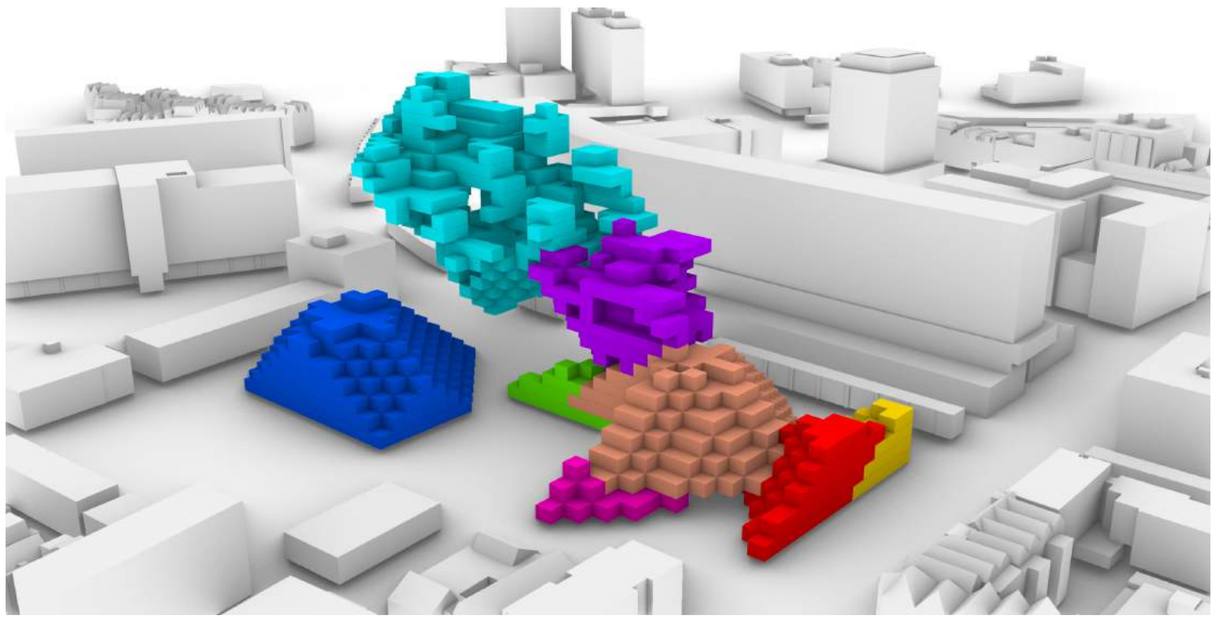
Urban Plan

Growing

Pseudocode

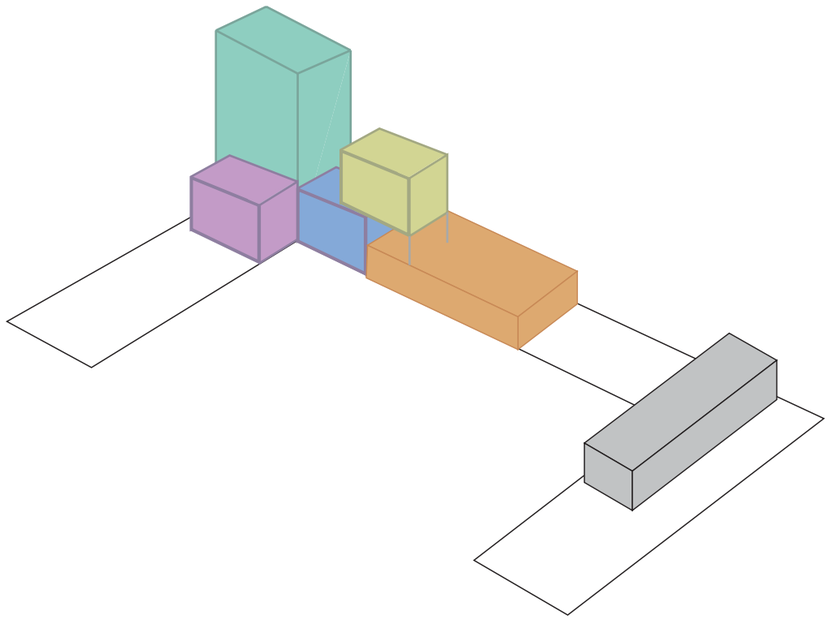
```
//defining classes
Define Gridpoint class
        x = x coordinate
        y = y coordinate
        z = z coordinate
        height = normalized height
        noisedist = normalized distance from noisepoint
        courtdist = normalized distance from the courtyard
        vijvdist = normalized distance from Vijverhofstraat

Define Cluster class
        name = name of the cluster
        voxels = desired size of the cluster in voxels
        hfac = the height factor (-1 to 1)
        nfac = the noise distance factor (-1 to 1)
        courtfac = the courtyard distance factor (-1 to 1)
        vijvfac = the vijverhof distance factor (-1 to 1)

//constructing the clusters
for each cluster:
        set the size of the cluster
        set the factors per cluster
        excluster = cluster('EXAMPLECLUSTER', size, hfac, nfac,
courtfac, vijvfac)
```

Rhino

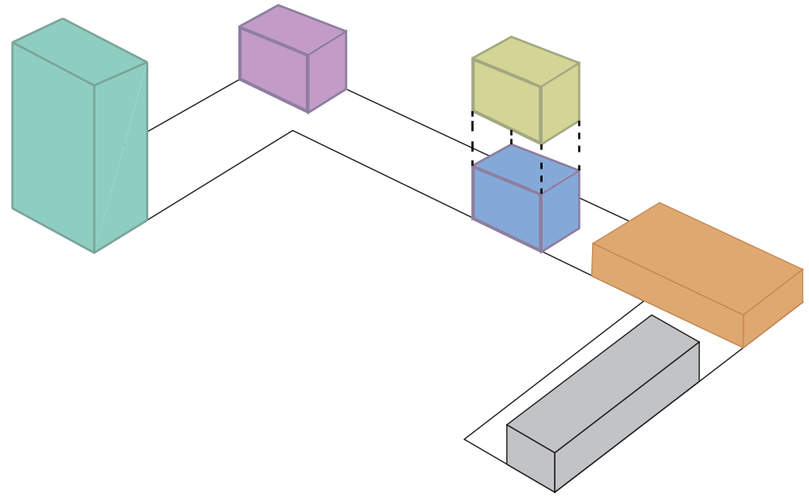


Growing dilemma

With

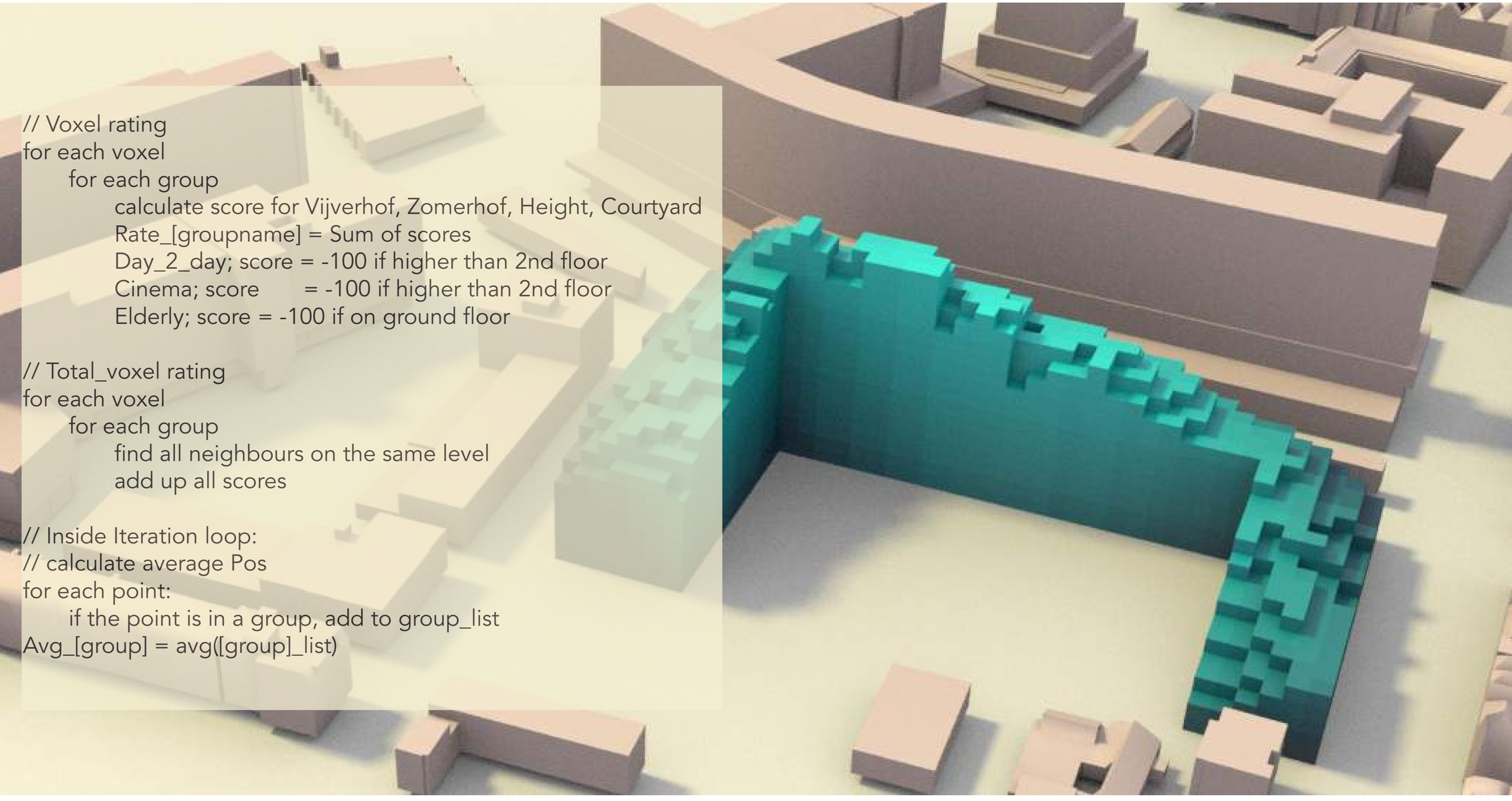


Without



Growing

Houdini

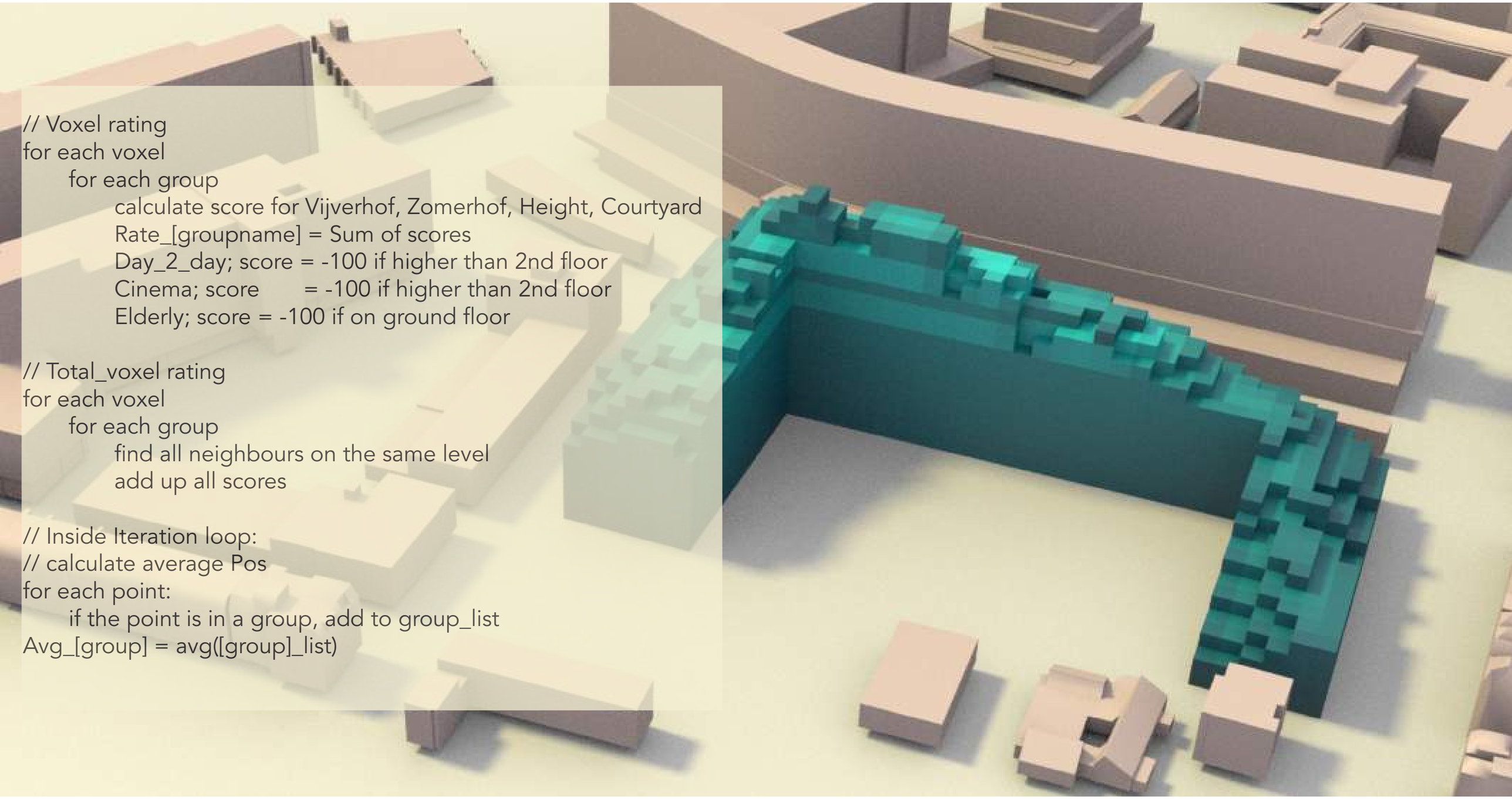


```
// Voxel rating
for each voxel
      for each group
            calculate score for Vijverhof, Zomerhof, Height, Courtyard
            Rate_[groupname] = Sum of scores
            Day_2_day; score = -100 if higher than 2nd floor
            Cinema; score       = -100 if higher than 2nd floor
            Elderly; score = -100 if on ground floor

// Total_voxel rating
for each voxel
      for each group
            find all neighbours on the same level
            add up all scores

// Inside Iteration loop:
// calculate average Pos
for each point:
      if the point is in a group, add to group_list
Avg_[group] = avg([group]_list)
```

Growing

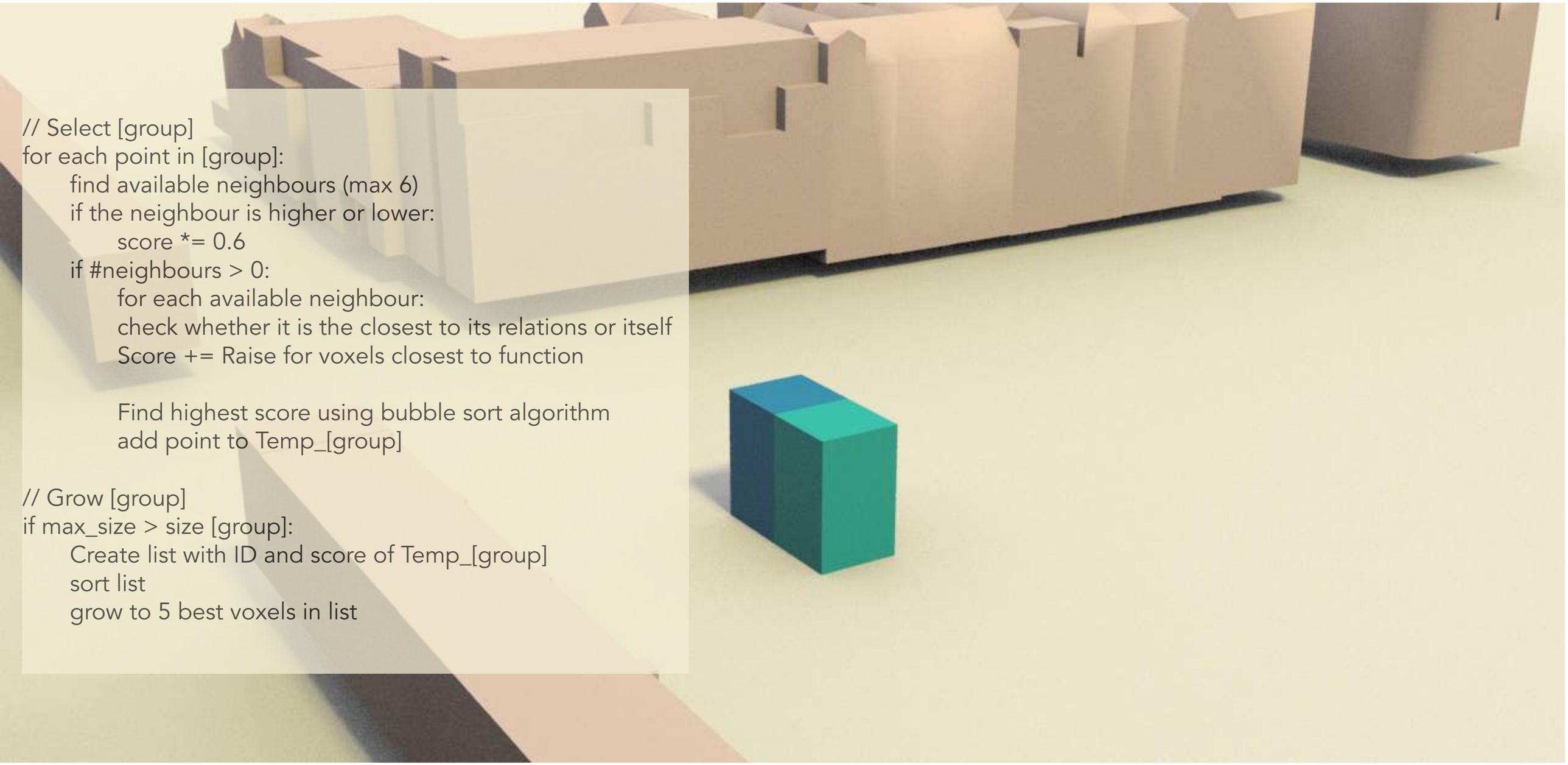
Houdini

```
// Voxel rating
for each voxel
      for each group
            calculate score for Vijverhof, Zomerhof, Height, Courtyard
            Rate_[groupname] = Sum of scores
            Day_2_day; score = -100 if higher than 2nd floor
            Cinema; score       = -100 if higher than 2nd floor
            Elderly; score = -100 if on ground floor

// Total_voxel rating
for each voxel
      for each group
            find all neighbours on the same level
            add up all scores

// Inside Iteration loop:
// calculate average Pos
for each point:
      if the point is in a group, add to group_list
Avg_[group] = avg([group]_list)
```



Growing

Houdini

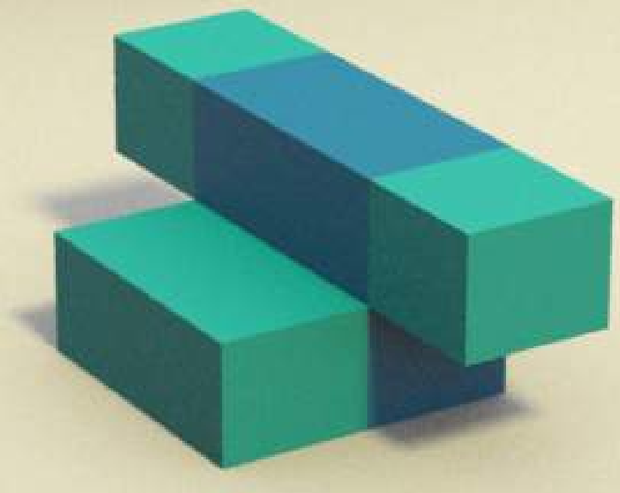


```
// Select [group]
for each point in [group]:
    find available neighbours (max 6)
    if the neighbour is higher or lower:
        score *= 0.6
    if #neighbours > 0:
        for each available neighbour:
        check whether it is the closest to its relations or itself
        Score += Raise for voxels closest to function

        Find highest score using bubble sort algorithm
        add point to Temp_[group]

// Grow [group]
if max_size > size [group]:
    Create list with ID and score of Temp_[group]
    sort list
    grow to 5 best voxels in list
```

Growing

Houdini

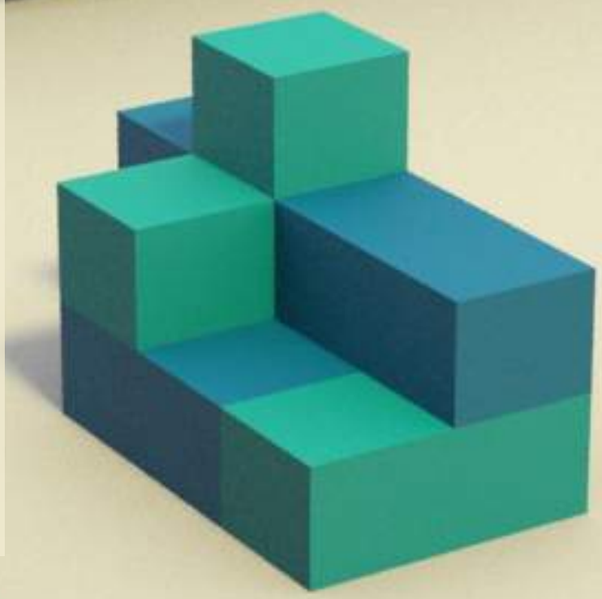
```
// Select [group]
for each point in [group]:
    find available neighbours (max 6)
    if the neighbour is higher or lower:
        score *= 0.6
    if #neighbours > 0:
        for each available neighbour:
        check whether it is the closest to its relations or itself
        Score += Raise for voxels closest to function

        Find highest score using bubble sort algorithm
        add point to Temp_[group]

// Grow [group]
if max_size > size [group]:
    Create list with ID and score of Temp_[group]
    sort list
    grow to 5 best voxels in list
```



Growing

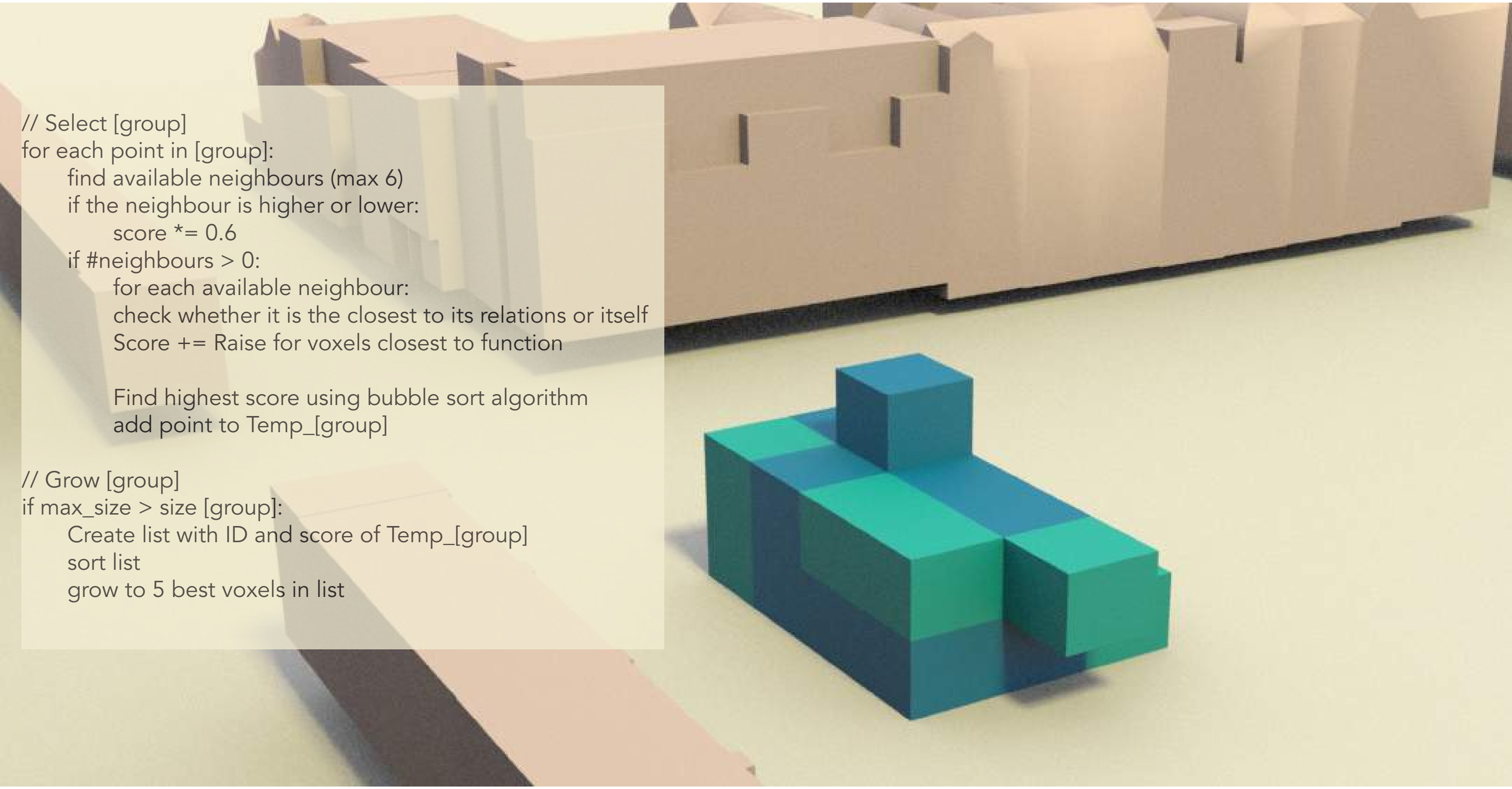
Houdini

```
// Select [group]
for each point in [group]:
    find available neighbours (max 6)
    if the neighbour is higher or lower:
        score *= 0.6
    if #neighbours > 0:
        for each available neighbour:
        check whether it is the closest to its relations or itself
        Score += Raise for voxels closest to function

        Find highest score using bubble sort algorithm
        add point to Temp_[group]

// Grow [group]
if max_size > size [group]:
    Create list with ID and score of Temp_[group]
    sort list
    grow to 5 best voxels in list
```



Growing

Houdini



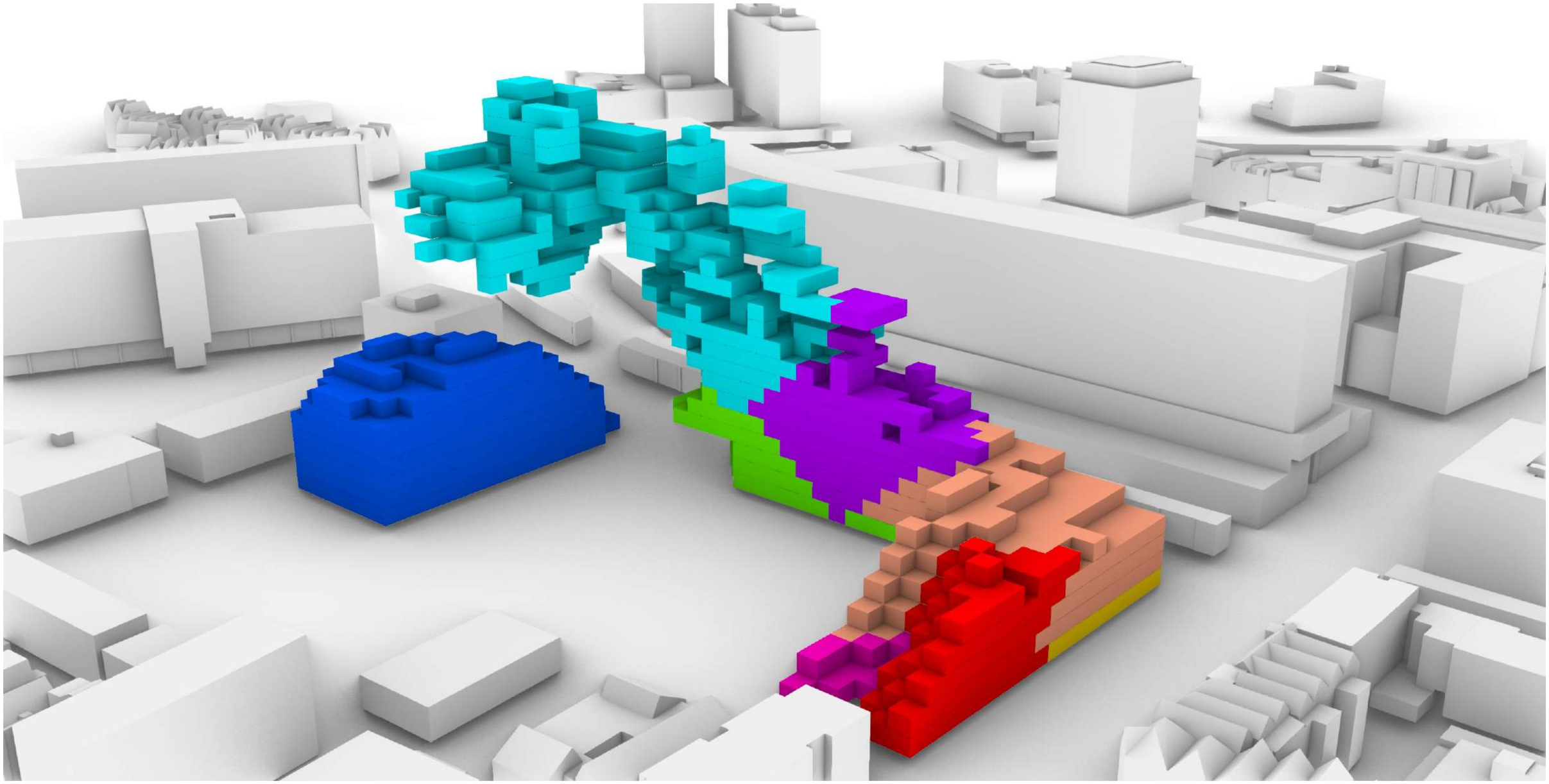
```
// Select [group]
for each point in [group]:
    find available neighbours (max 6)
    if the neighbour is higher or lower:
        score *= 0.6
    if #neighbours > 0:
        for each available neighbour:
        check whether it is the closest to its relations or itself
        Score += Raise for voxels closest to function

        Find highest score using bubble sort algorithm
        add point to Temp_[group]

// Grow [group]
if max_size > size [group]:
    Create list with ID and score of Temp_[group]
    sort list
    grow to 5 best voxels in list
```

Growing

Houdini

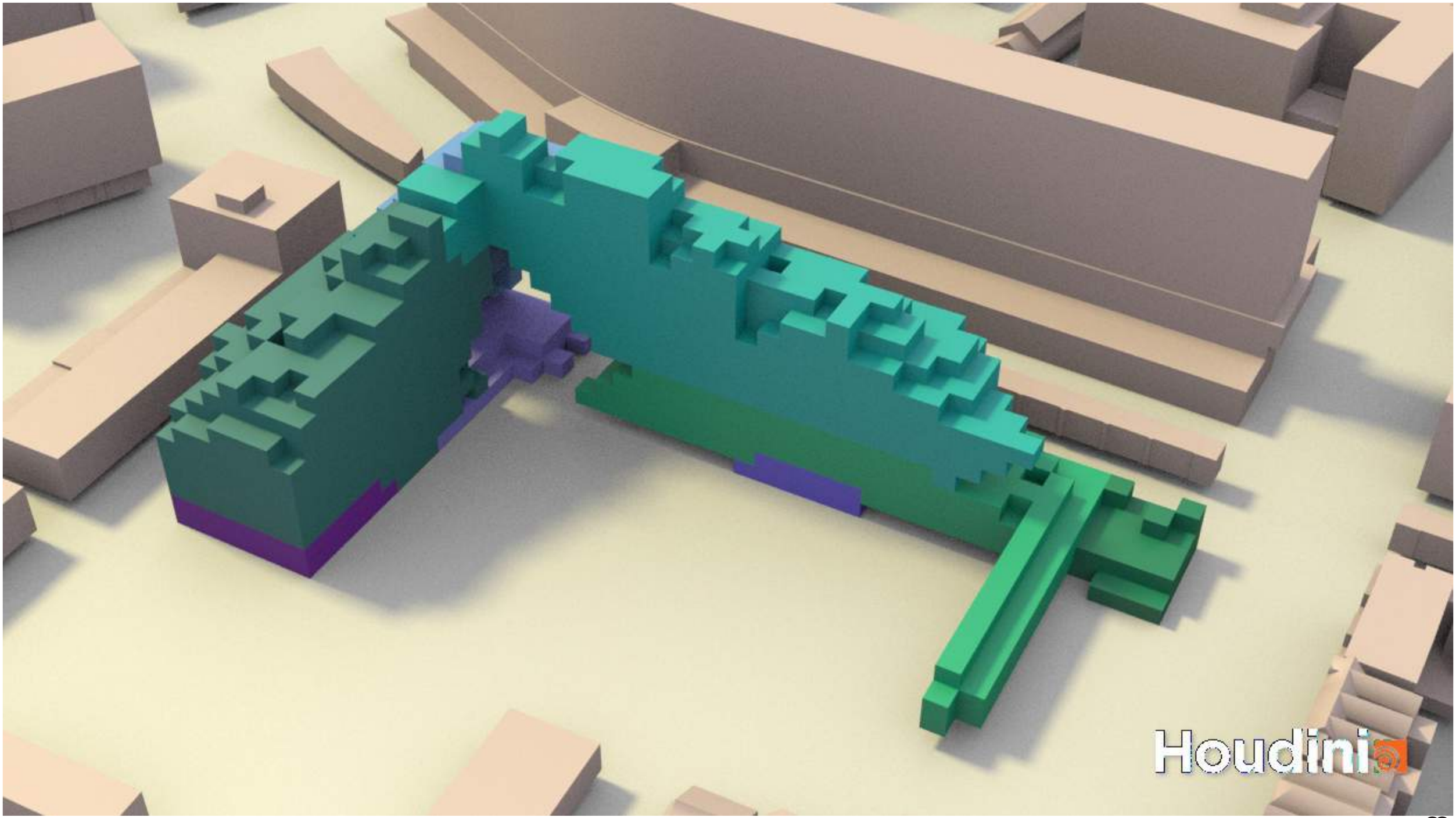
Final Growth

Rhino



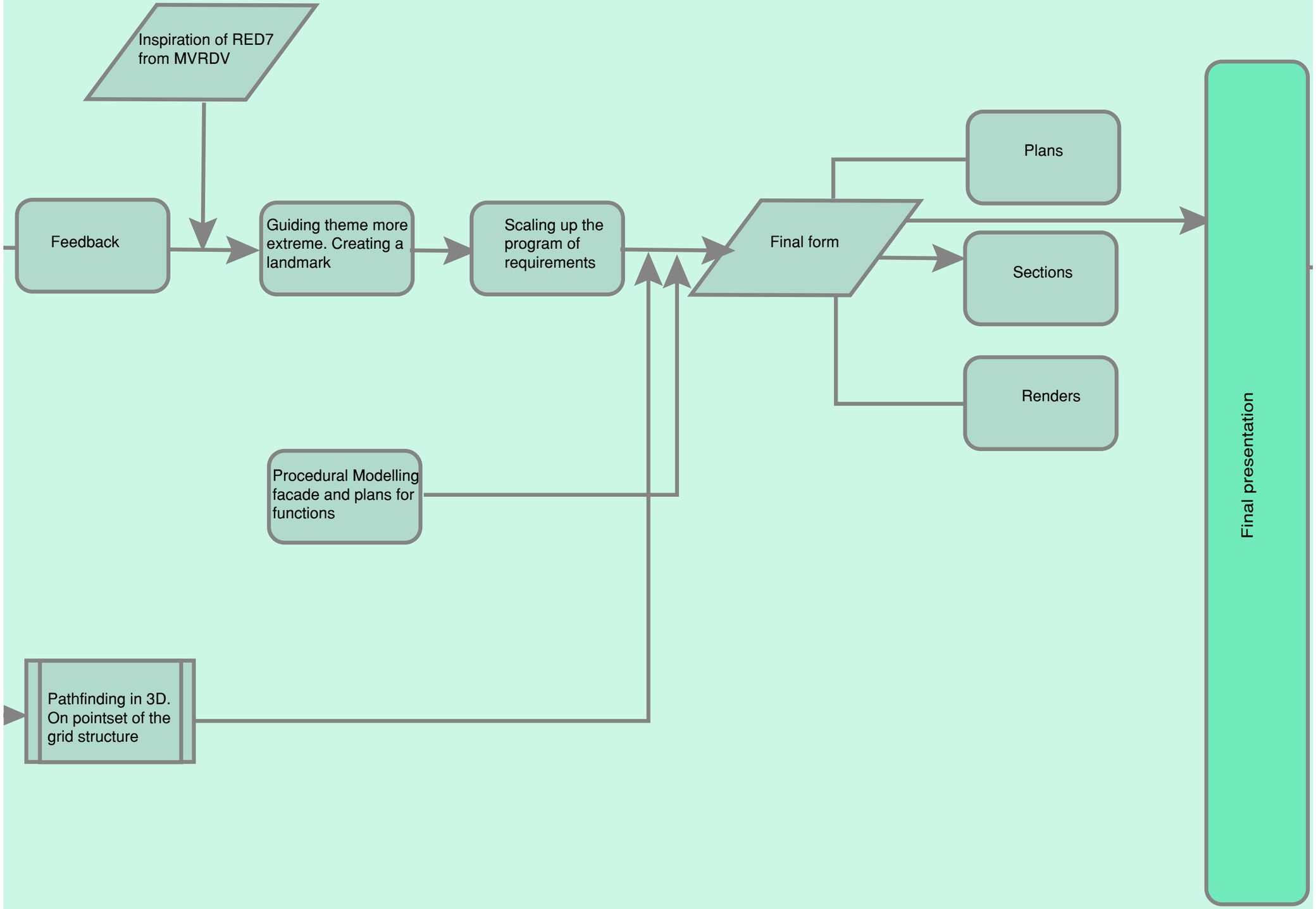
Final Growth

Houdini



Flowchart

Step 3



Pathfinding

Rhino

//Input Voxel Midpoints from growing envelope

//Creating approximate edges according to surrounding voxels and create paths between destination points

Defining for each point neighbours of points (depen ding to shortest distant points) Minradius – 3.1, Maxradius – 4.1

If dist >minradius and dist<maxradius add edge

Destination points list of clusters

Connections list of clusters (A to B)

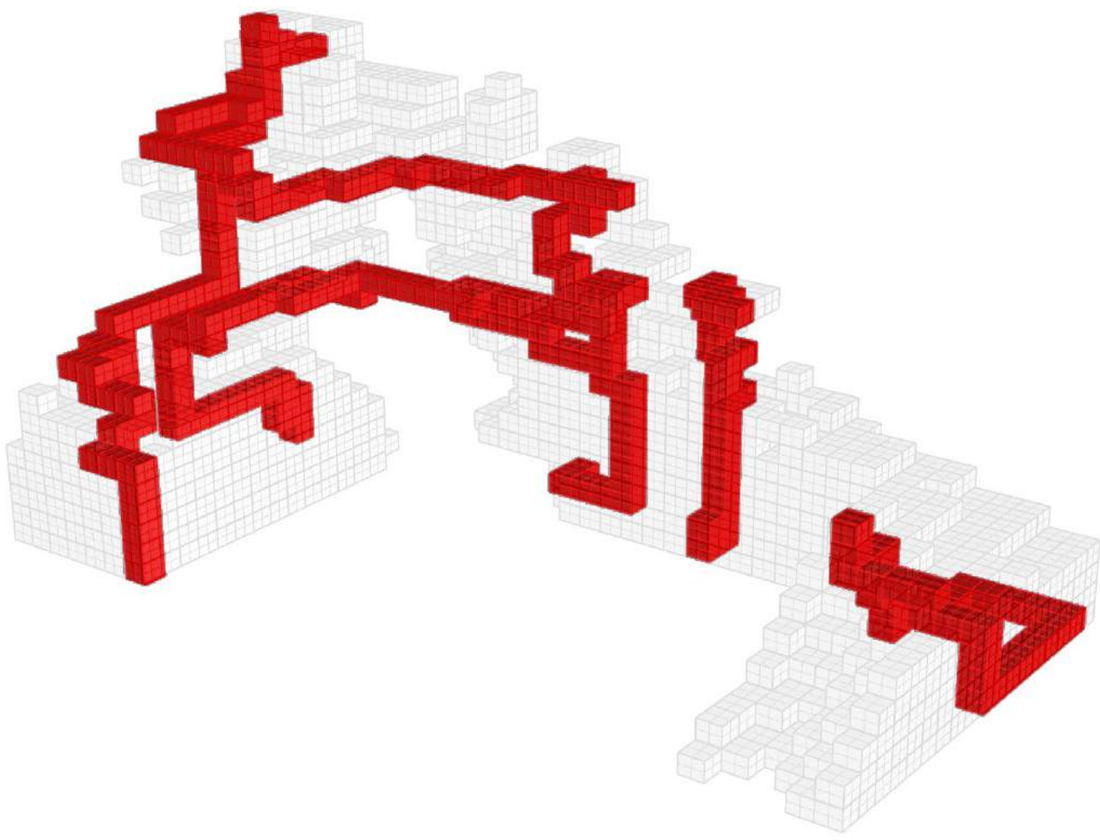
Creating path append to destination points and con- nection list (midpoint voxel nodes)

//Mainroad structure pathfinding:

Points of clusters connected to each other, according to connection diagram (pathfinding d)

//Subroad structure pathfinding:

Points of interest on each level to staircase (path- finding 2d). (Keep in mind: connection to at least 2 stairways)



Pathfinding

Rhino

//Input Voxel Midpoints from growing envelope

//Creating approximate edges according to surrounding voxels and create paths between destination points

Defining for each point neighbours of points (depen ding to shortest distant points) Minradius – 3.1, Maxradius – 4.1

If dist >minradius and dist<maxradius add edge

Destination points list of clusters

Connections list of clusters (A to B)

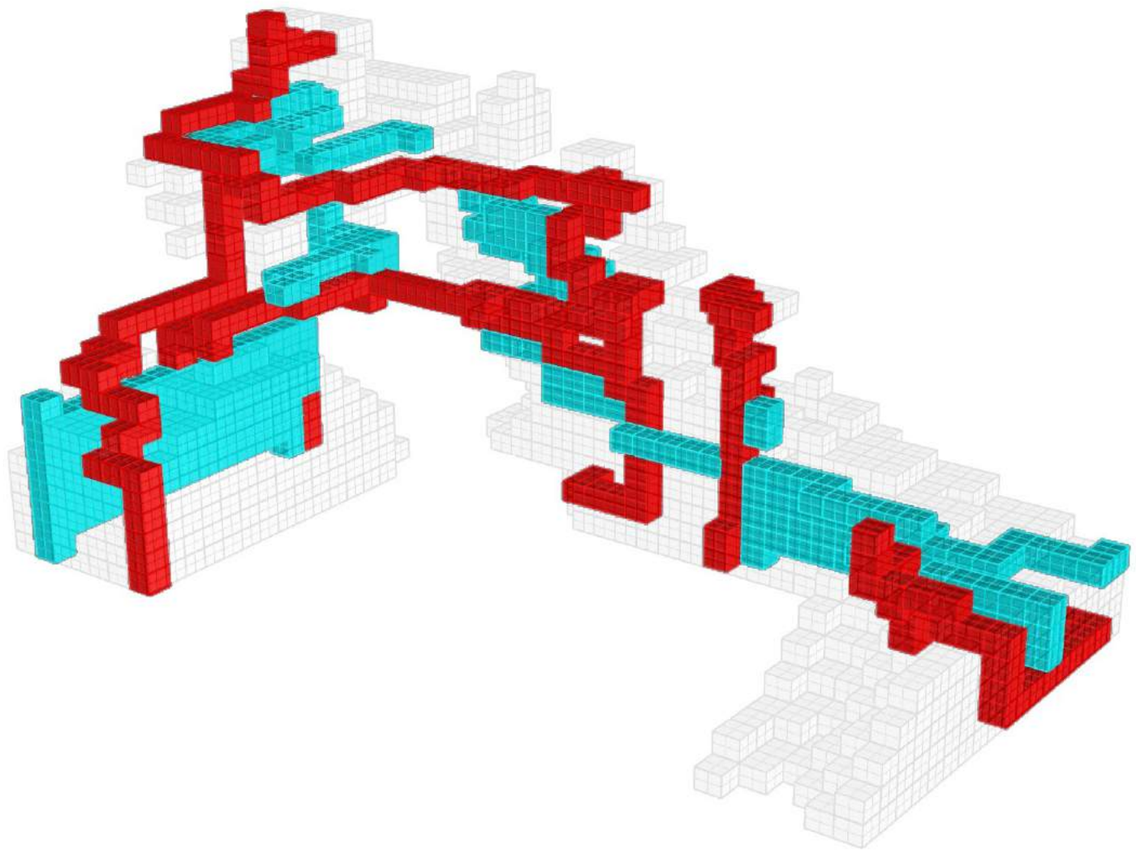
Creating path append to destination points and connection list (midpoint voxel nodes)

//Mainroad structure pathfinding:

Points of clusters connected to each other, according to connection diagram (pathfinding d)

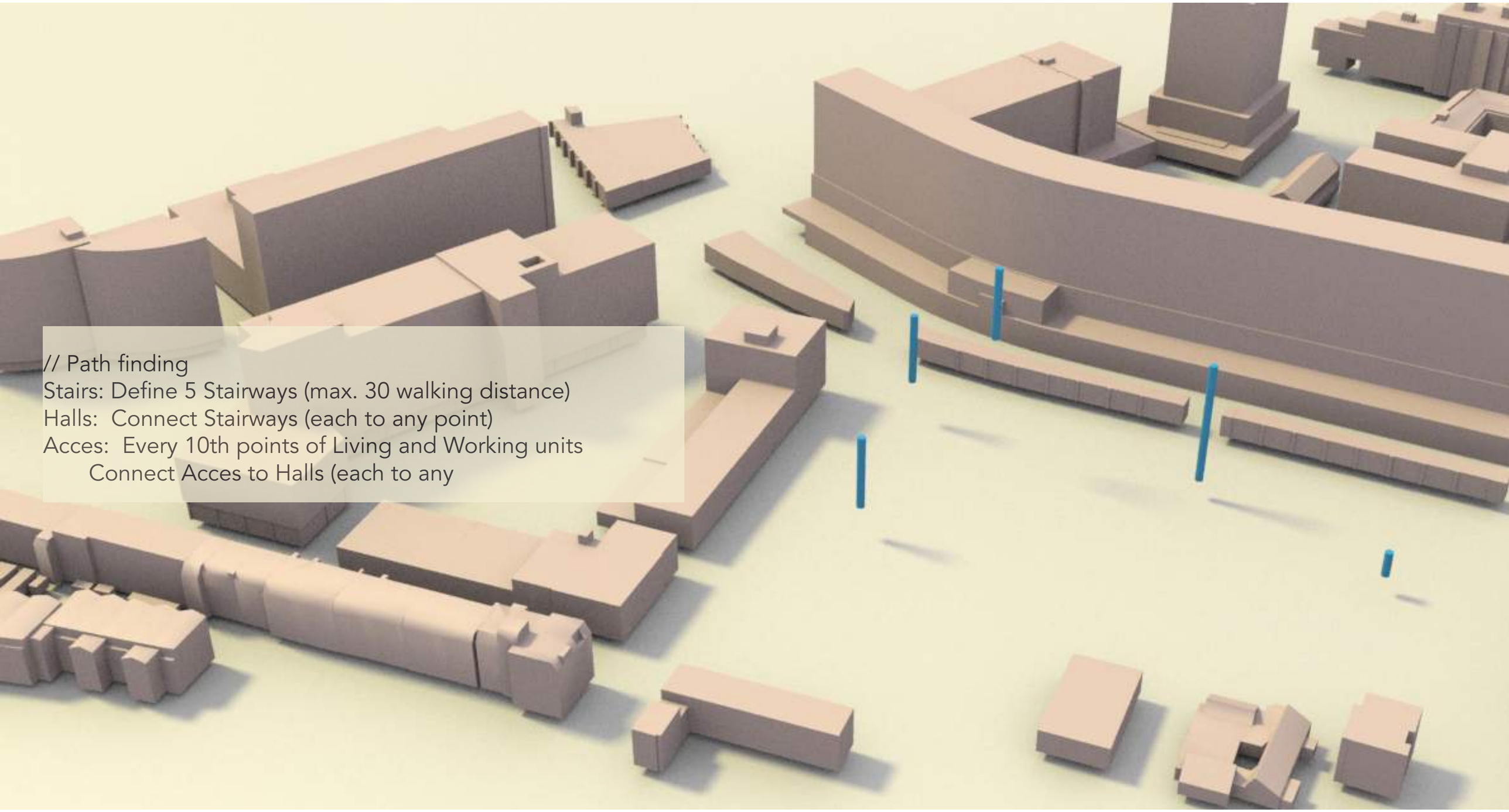
//Subroad structure pathfinding:

Points of interest on each level to staircase (pathfinding 2d). (Keep in mind: connection to at least 2 stairways)



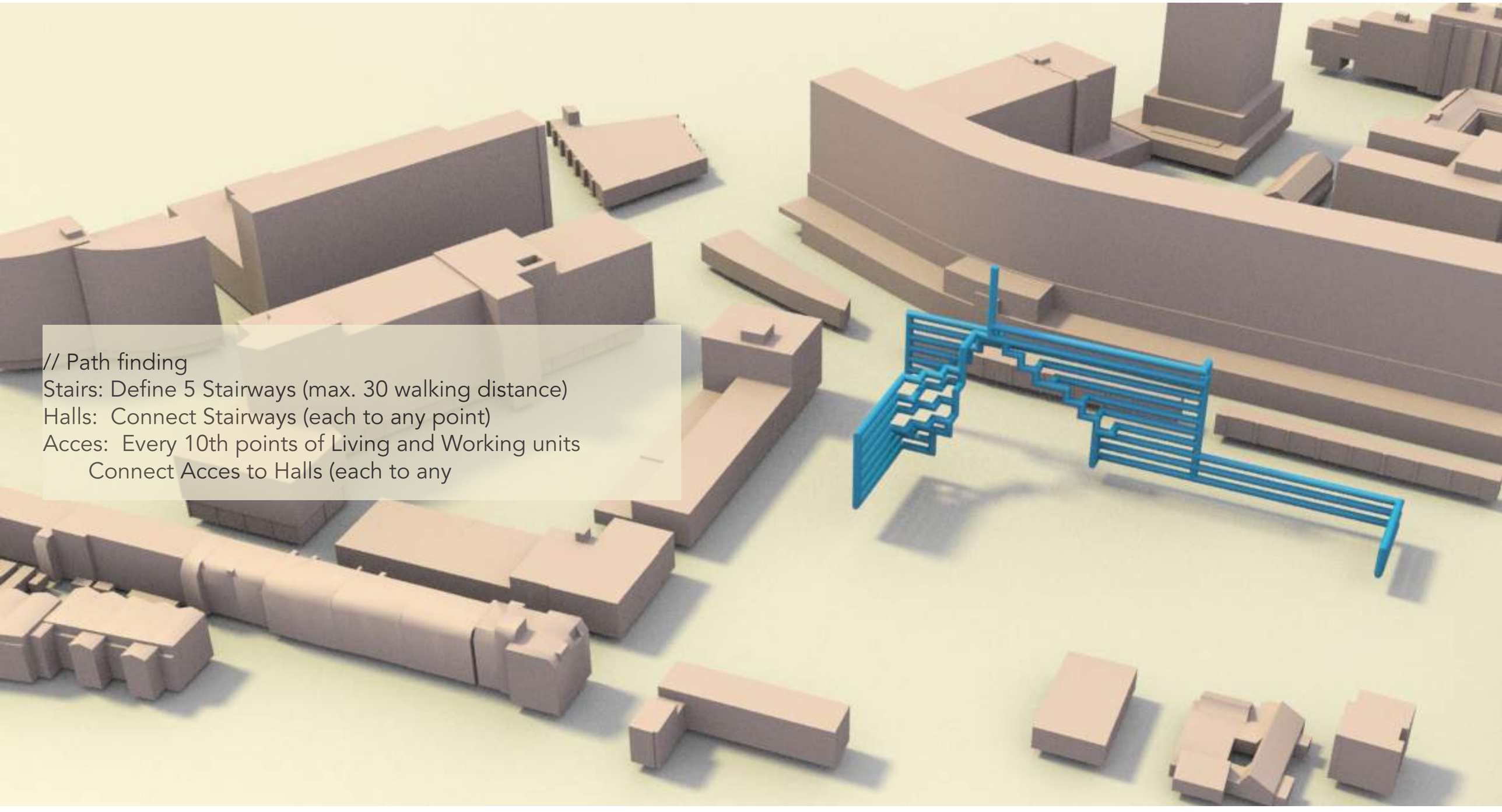
Pathfinding

Houdini



Pathfinding

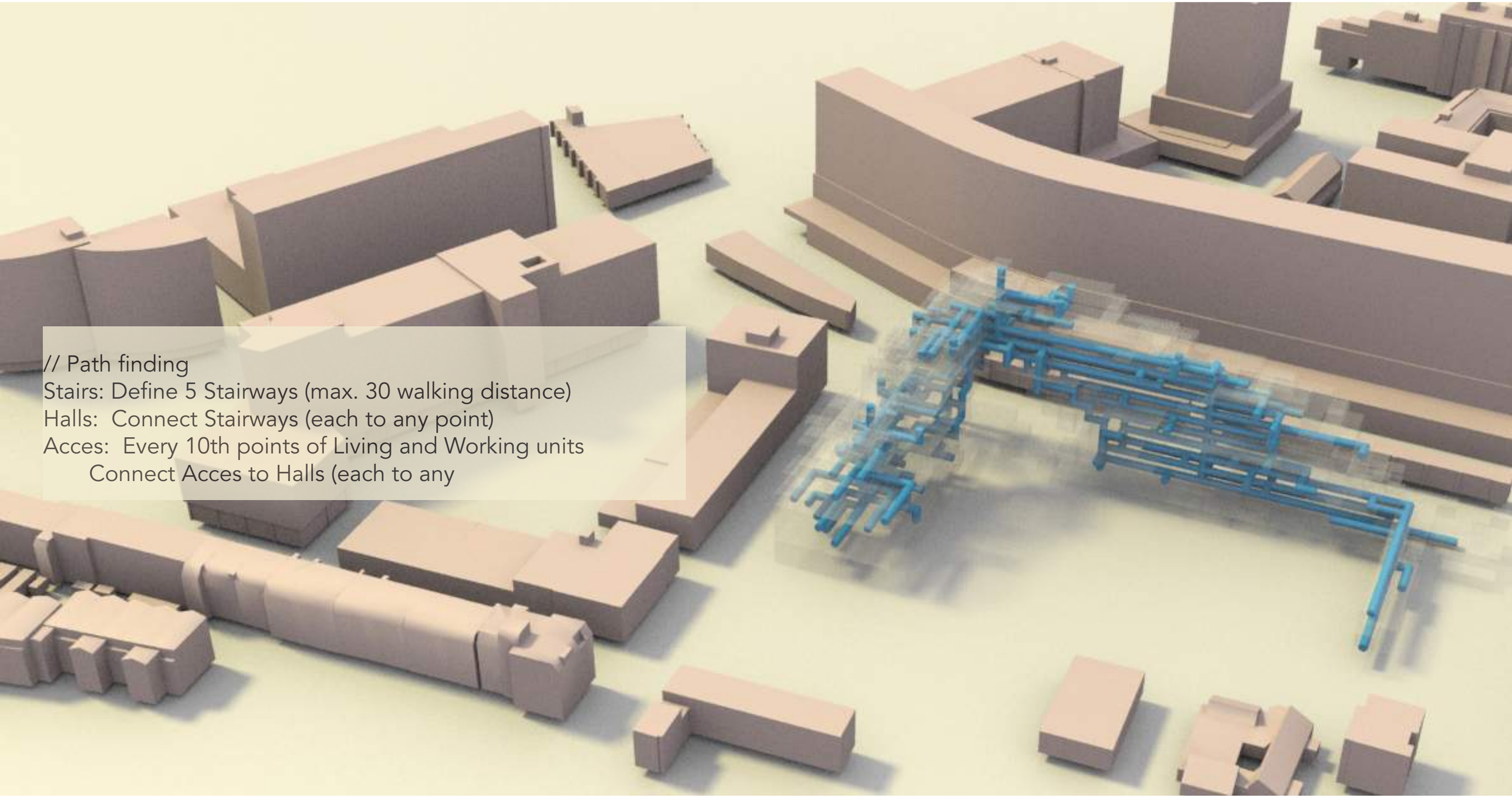
Houdini



// Path finding
Stairs: Define 5 Stairways (max. 30 walking distance)
Halls: Connect Stairways (each to any point)
Acces: Every 10th points of Living and Working units
Connect Acces to Halls (each to any

Pathfinding

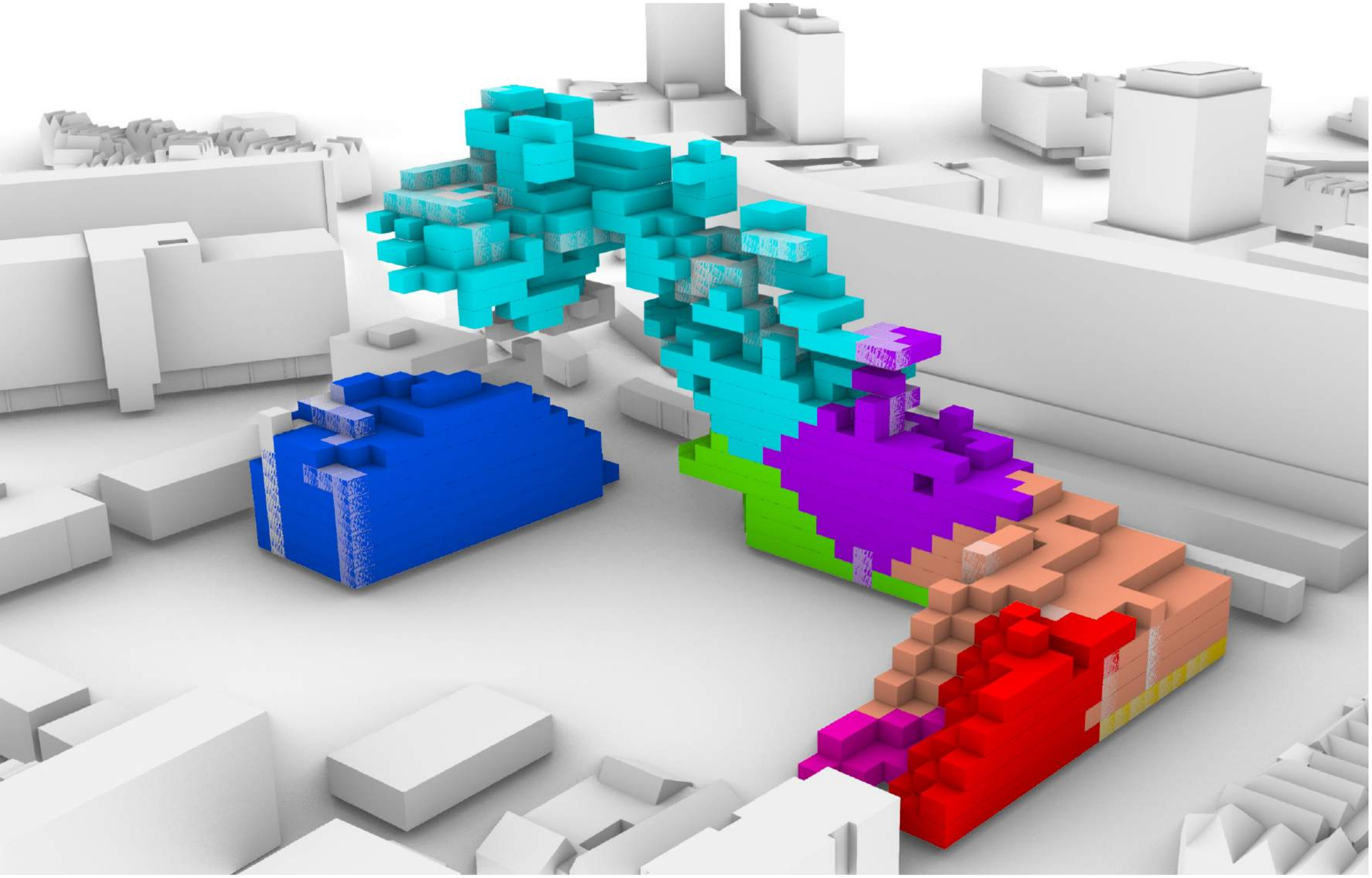
Houdini



// Path finding
Stairs: Define 5 Stairways (max. 30 walking distance)
Halls: Connect Stairways (each to any point)
Acces: Every 10th points of Living and Working units
Connect Acces to Halls (each to any

Final Form

Rhino

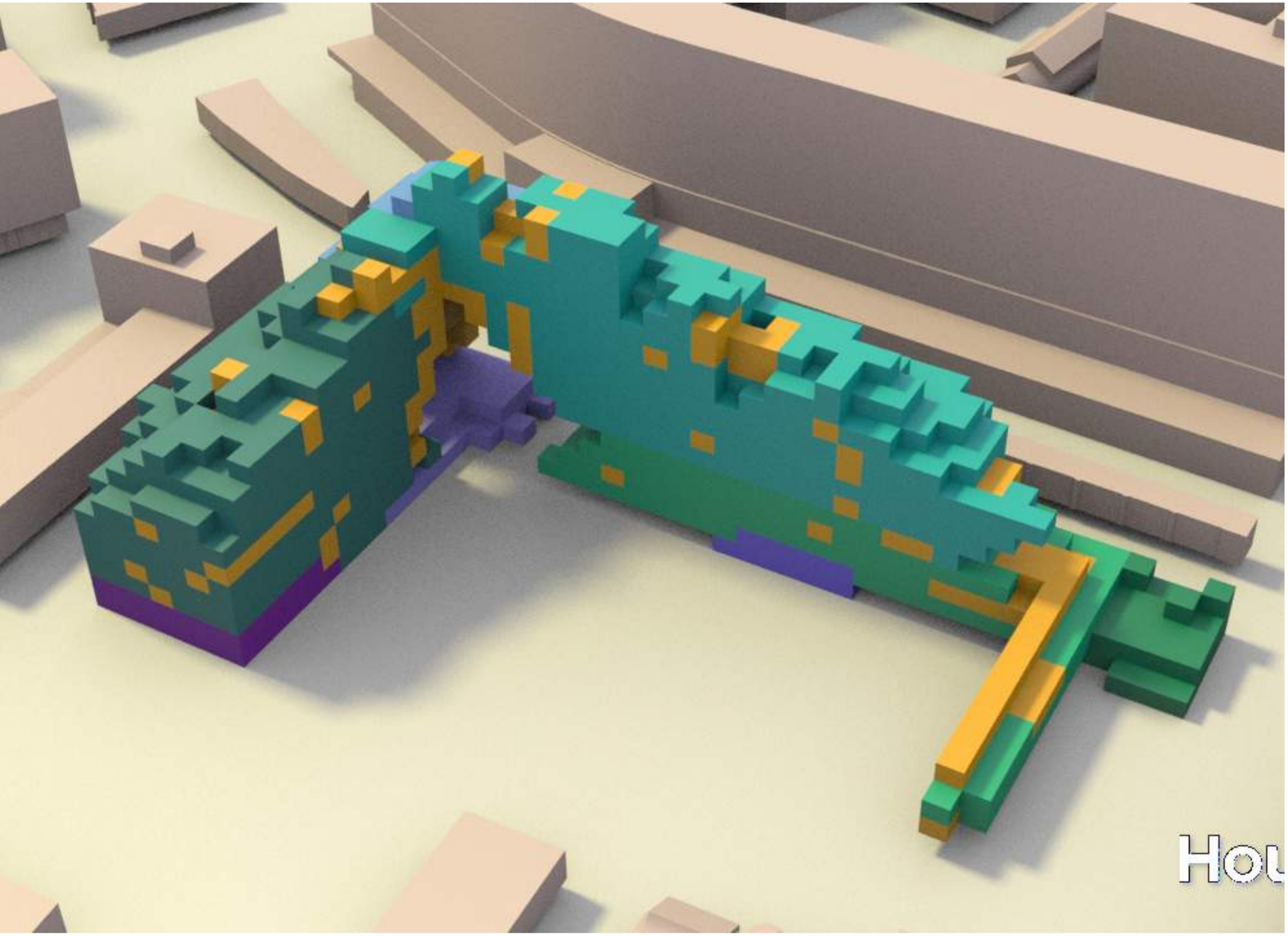


Legend

- *Starters living*
- *Library*
- *Cinema*
- *Restaurant*
- *Day to day shops*
- *Gym*
- *Work*
- *Student freetime*
- *Student living*
- *Elderly living*
- *Paths*

Final Form

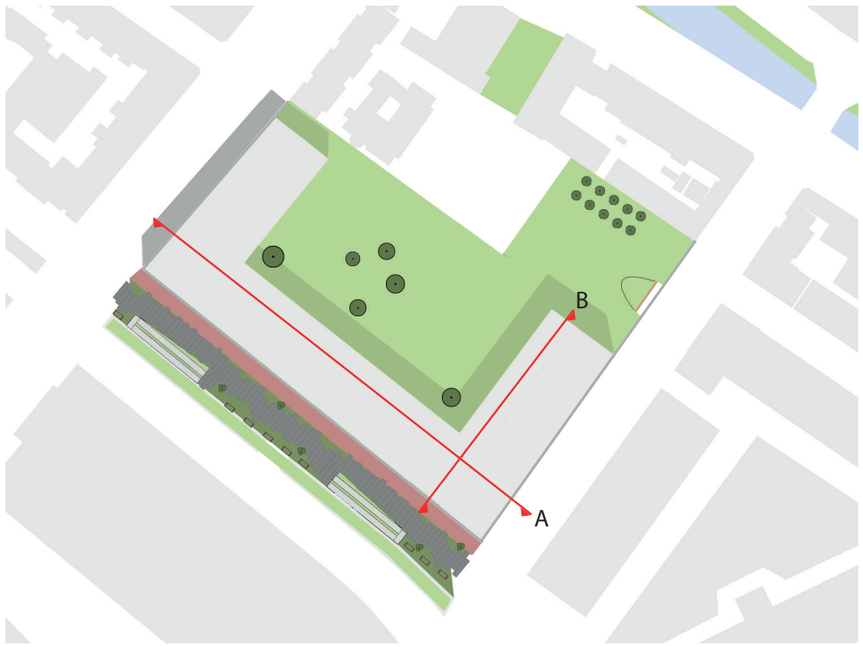
Houdini

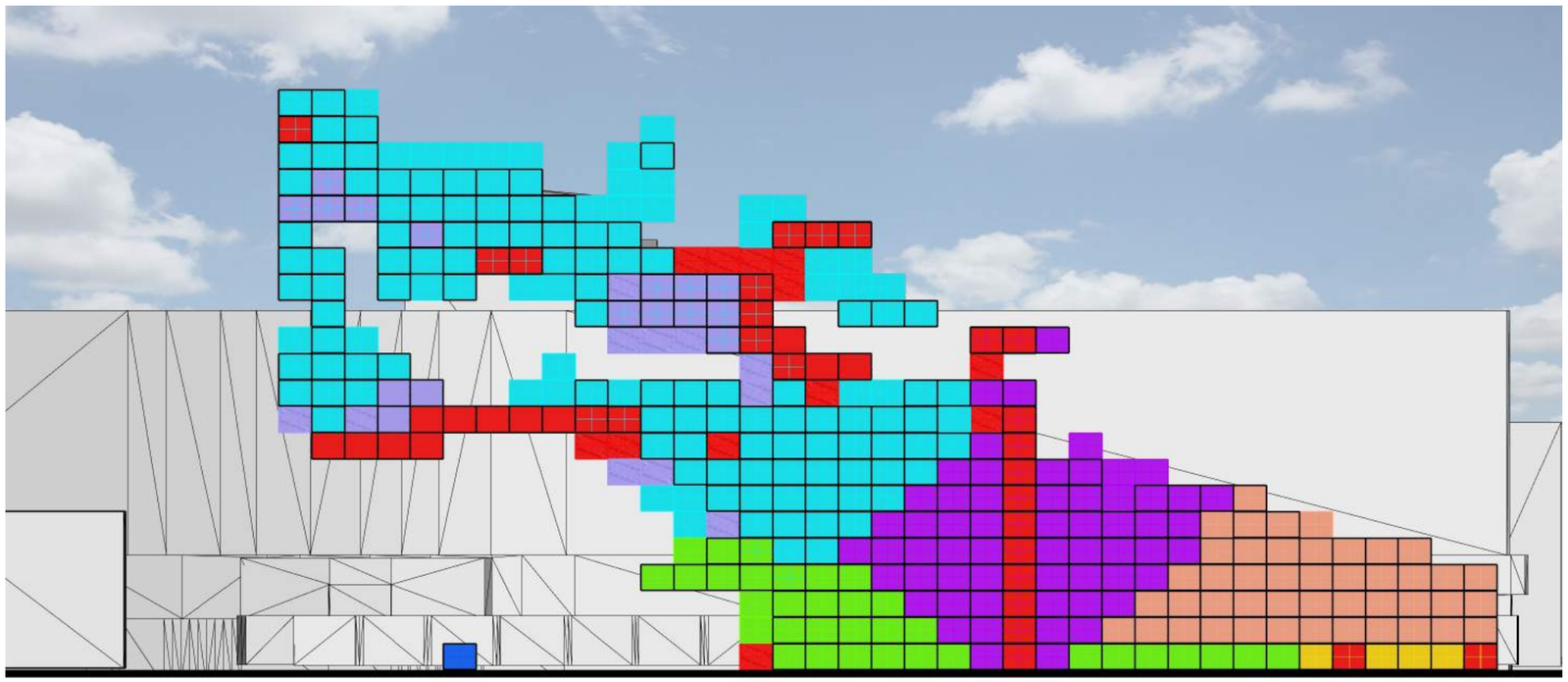


Legend

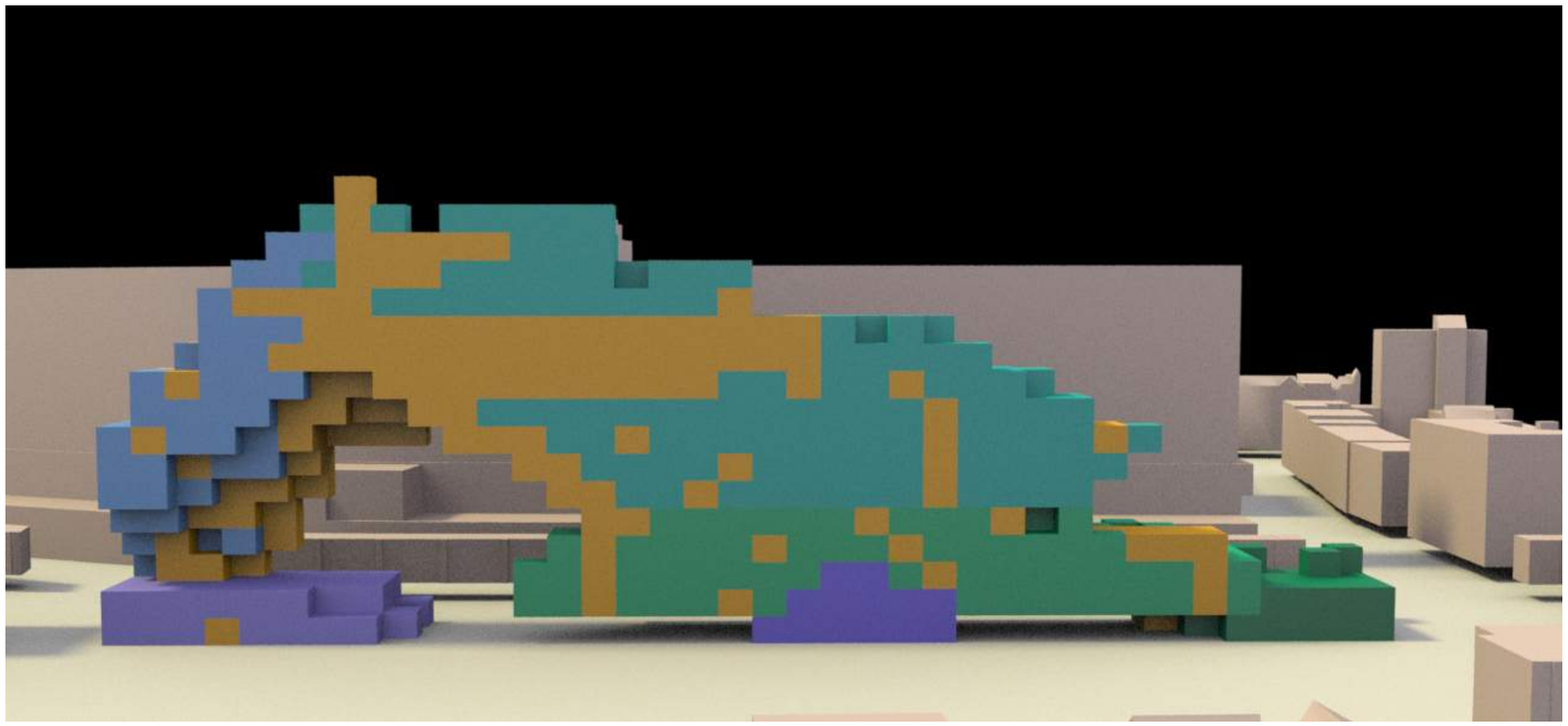
- *Starters living*
- *Library*
- *Cinema*
- *Restaurant*
- *Day to day shops*
- *Gym*
- *Work*
- *Student freetime*
- *Student living*
- *Elderly living*
- *Paths*

Section A



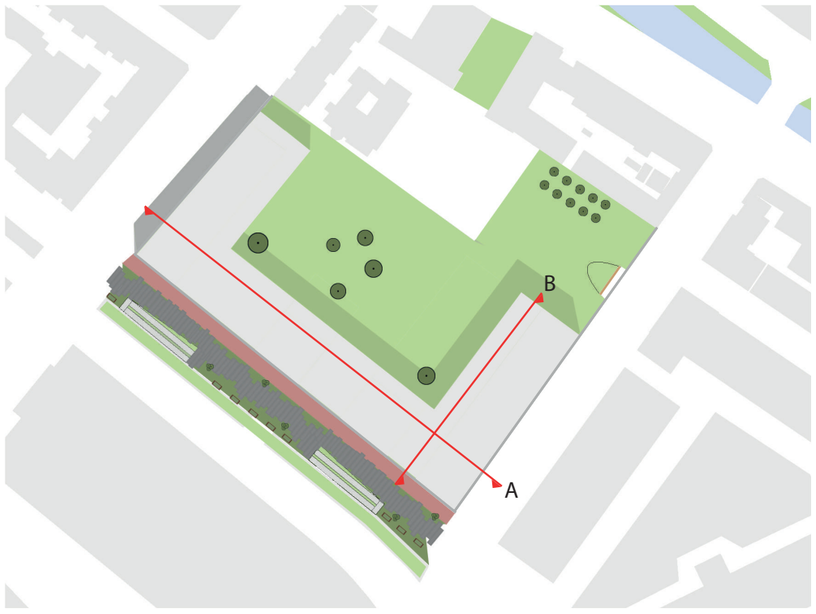


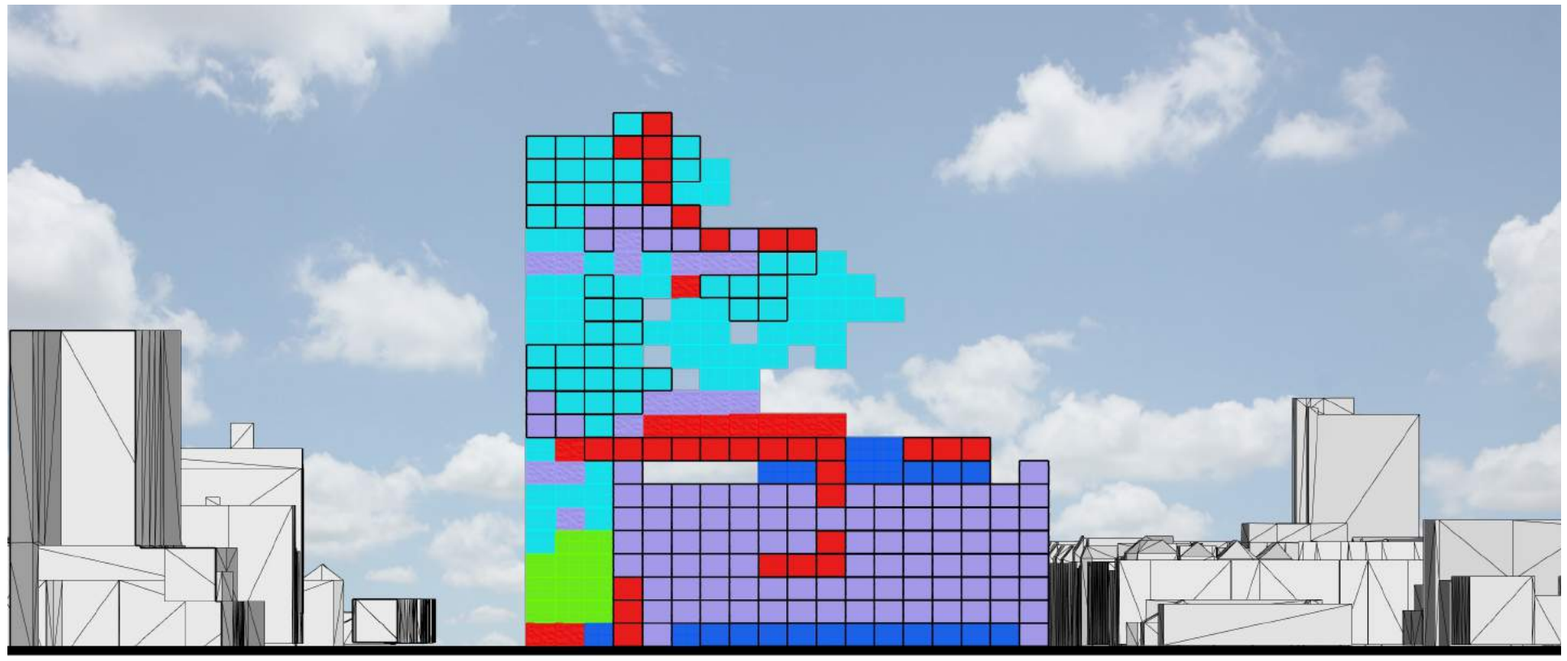
Rhino



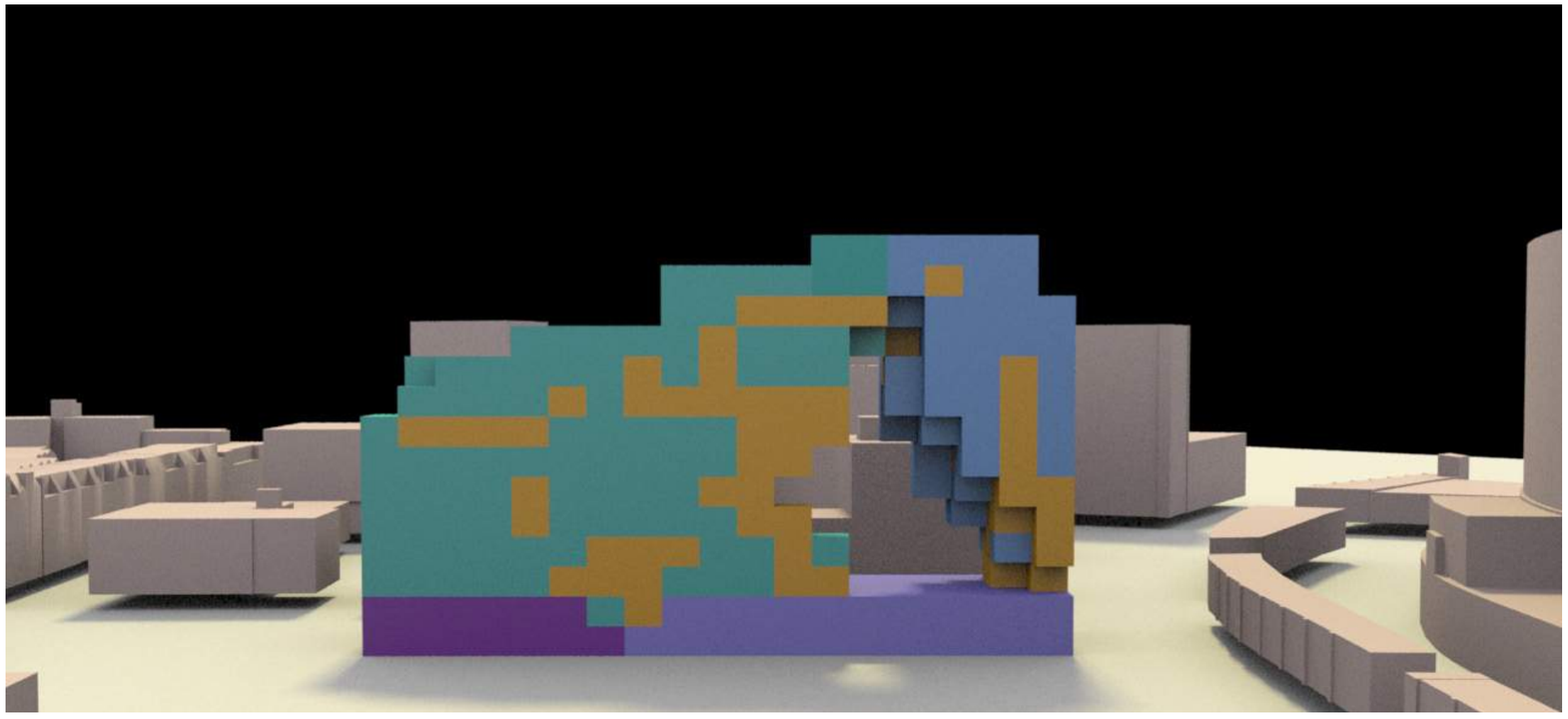
Houdini

Section B





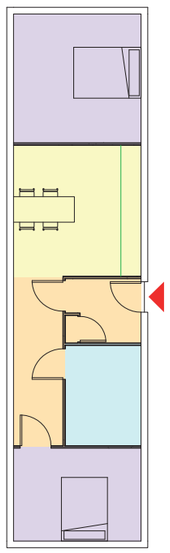
Rhino

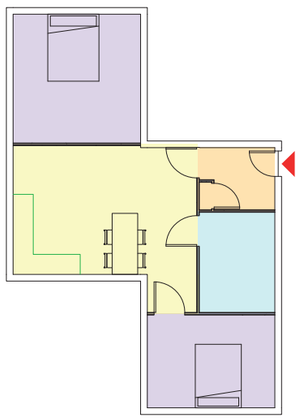


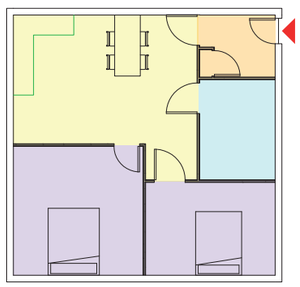
Houdini

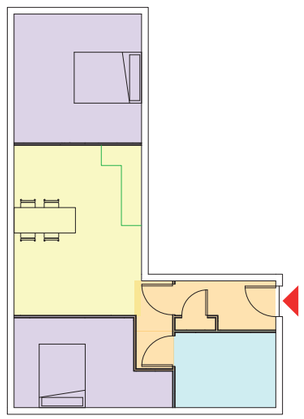
Floorplan

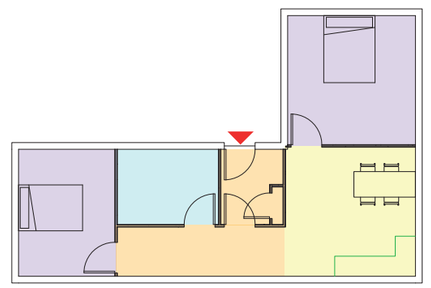
Starters / elderly 60 m2



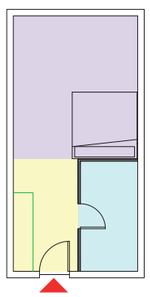


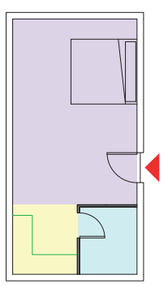






Students 30 m2

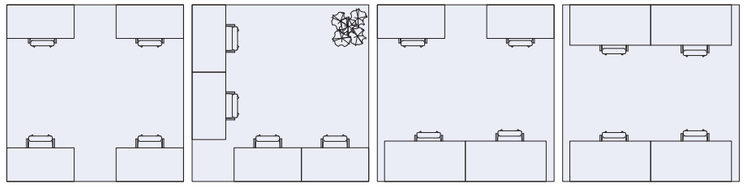




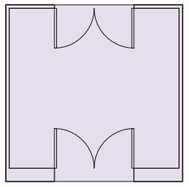
Floorplan

CO-Work

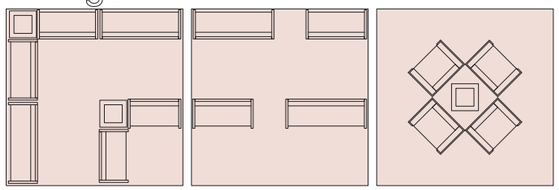
Desks



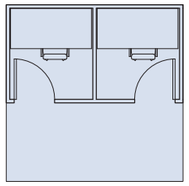
Entrance



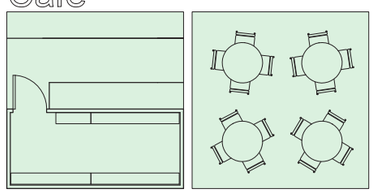
Lounge



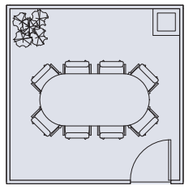
Private Work



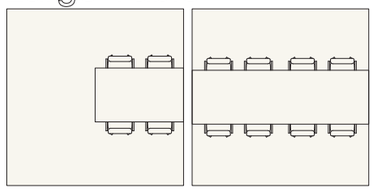
Cafe



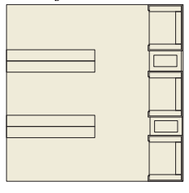
Meeting Room



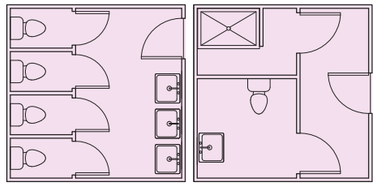
Large Table



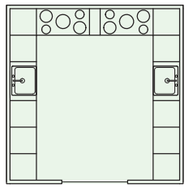
Inspiration



Toilets

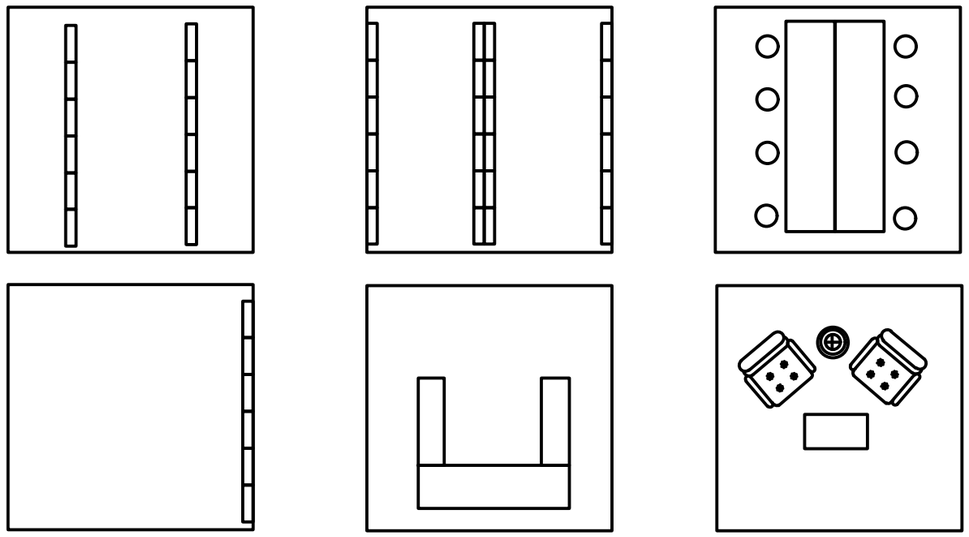


Kitchen Area



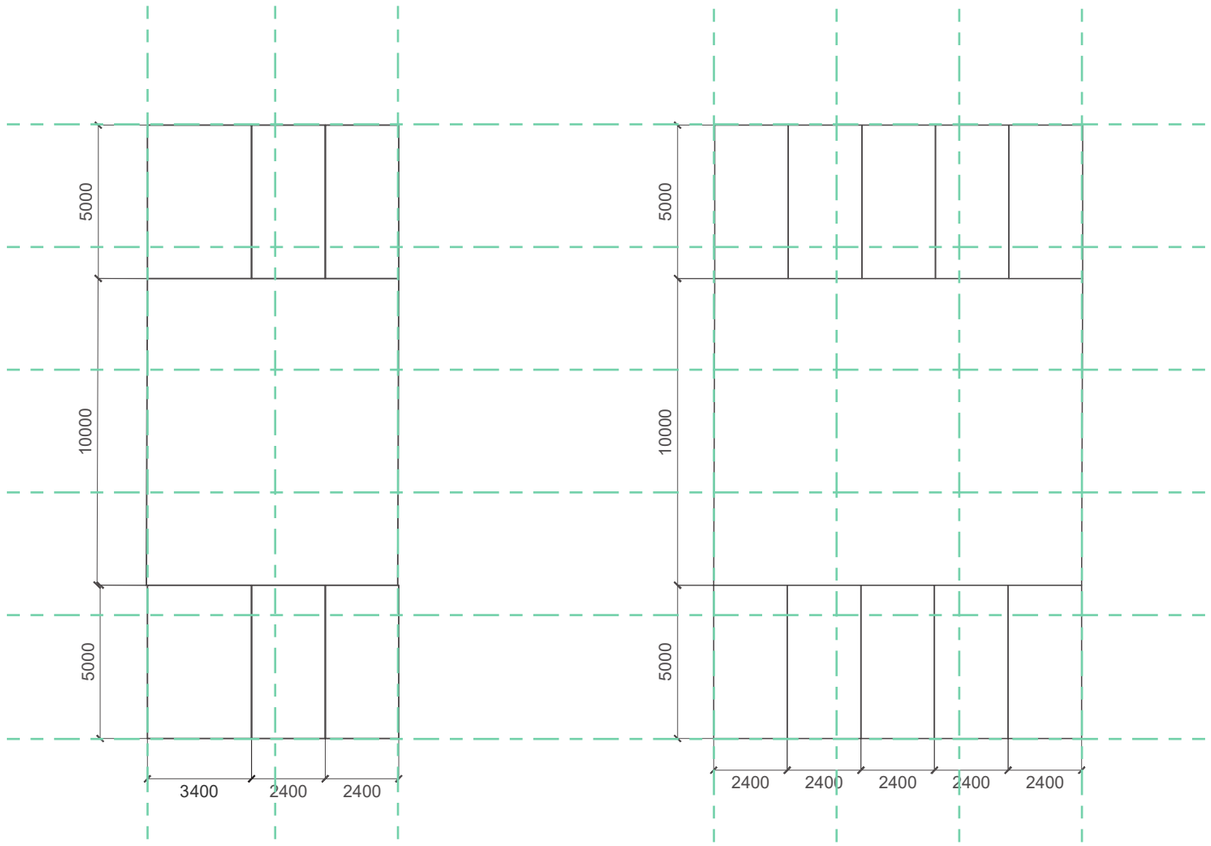
Floorplan

Library



Floorplan

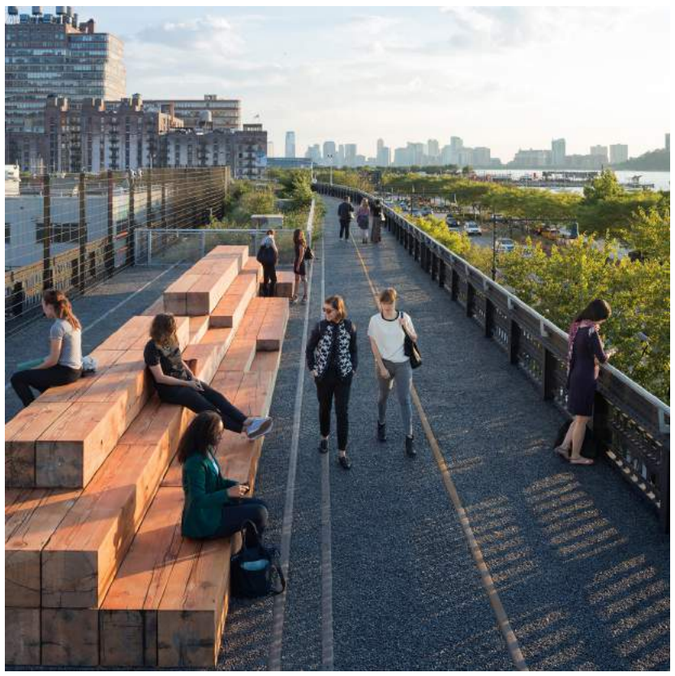
Parking



Floorplan

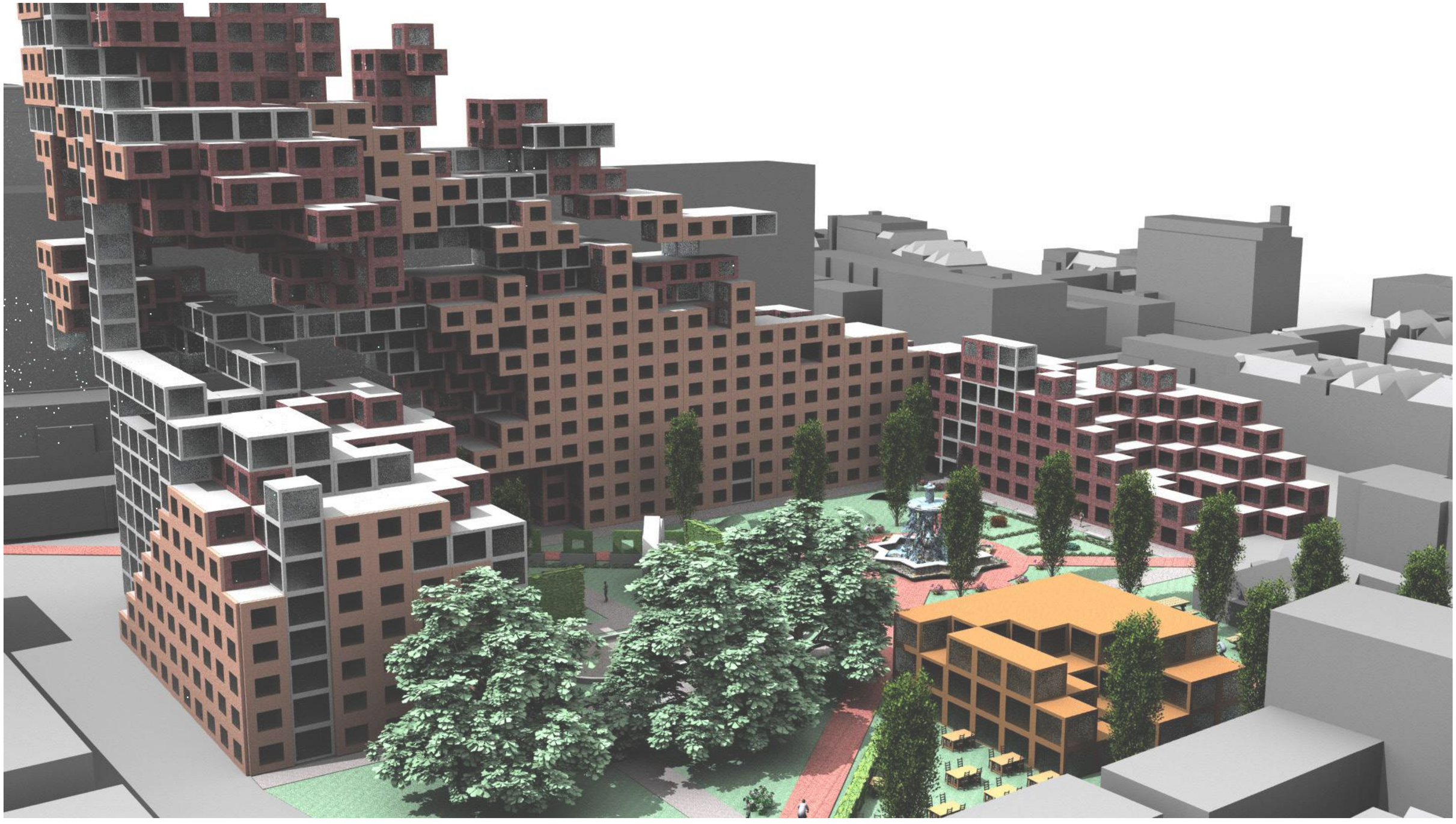

Legend
Library
Cinema
Restaurant
Day to day shops
Gym
Work
Student freetime
Student living
Entrances
Parking Entrance
Main paths
Subpaths
0 5 25 m
N
P

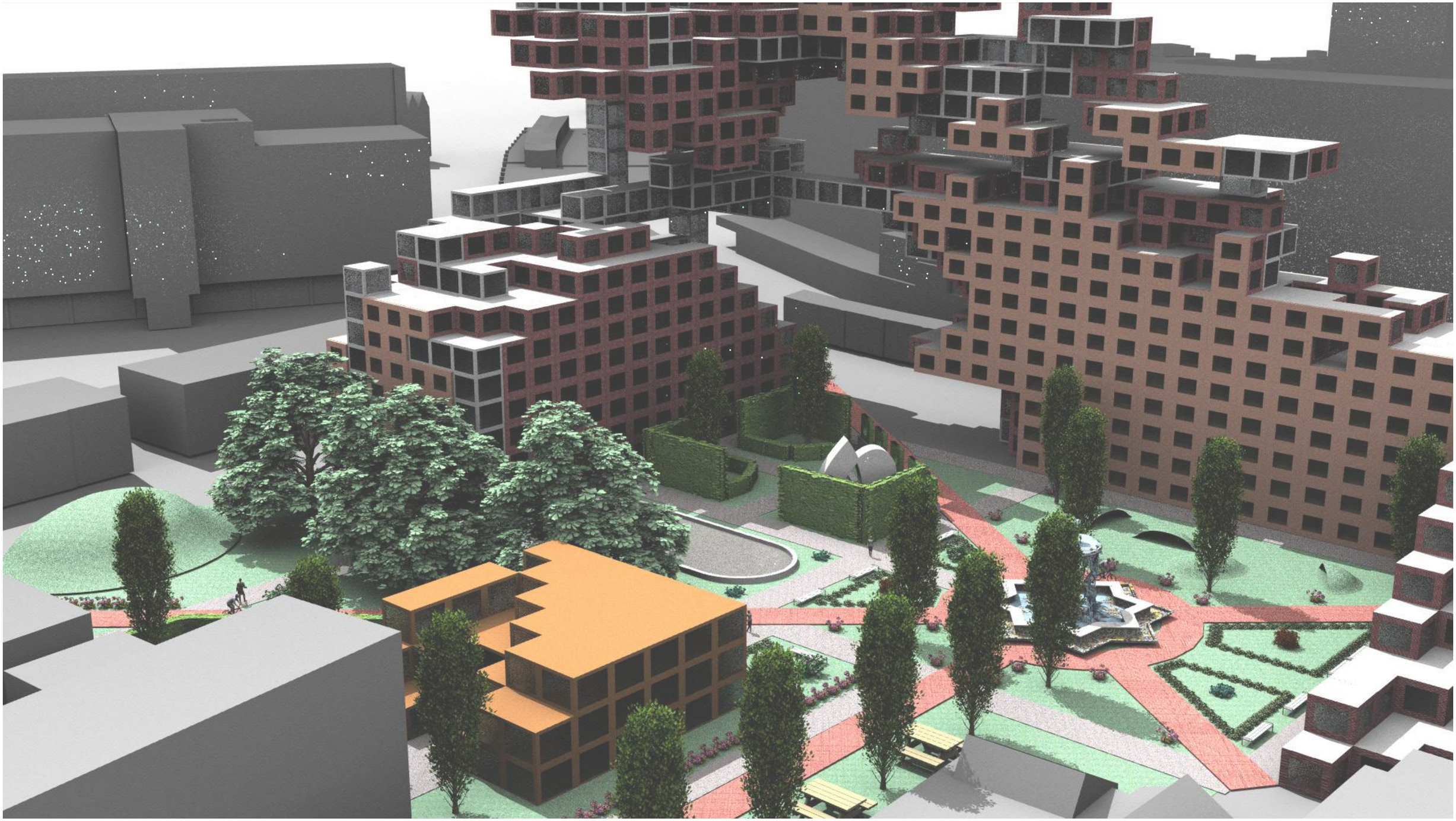
Inspiration

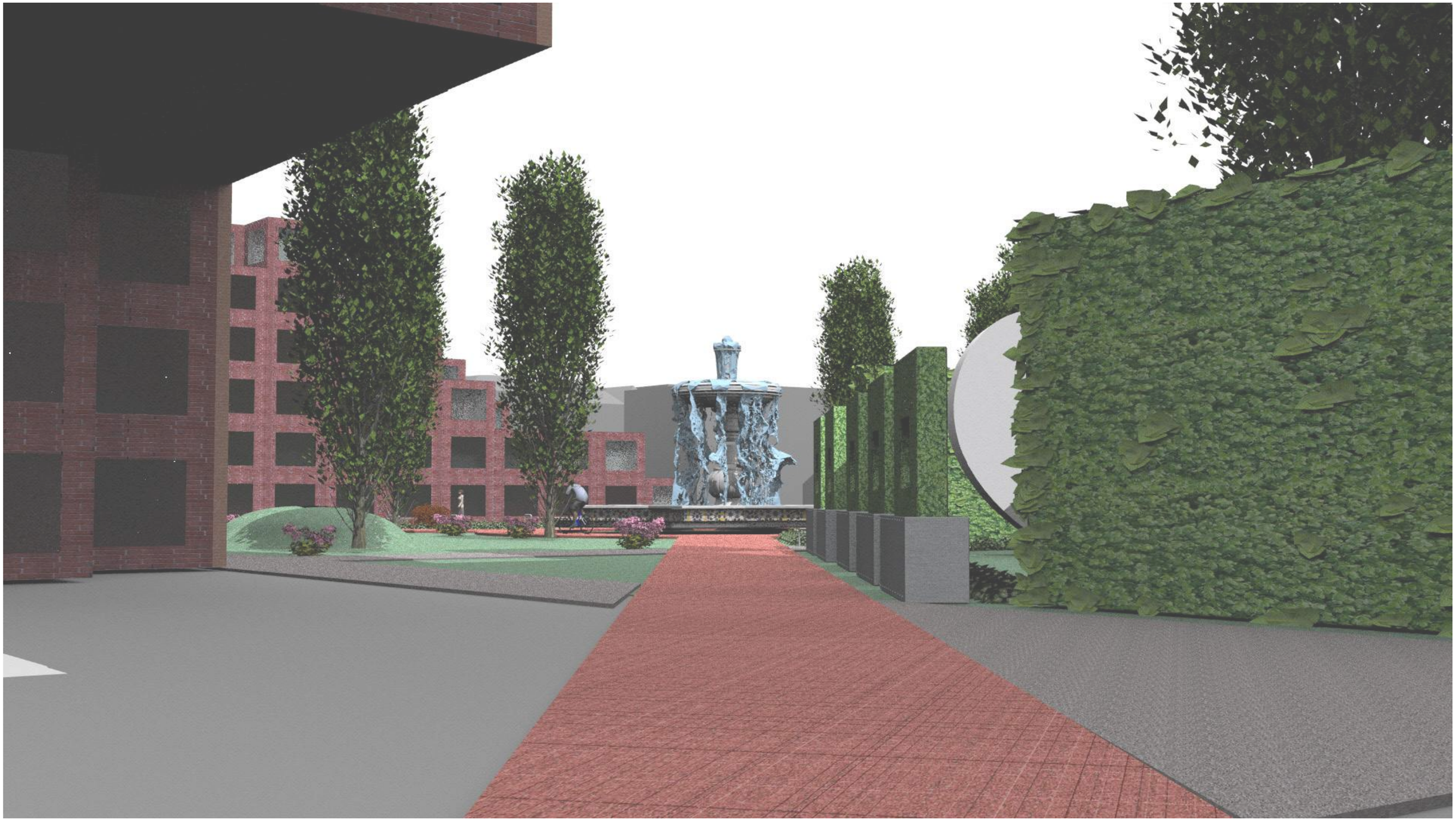


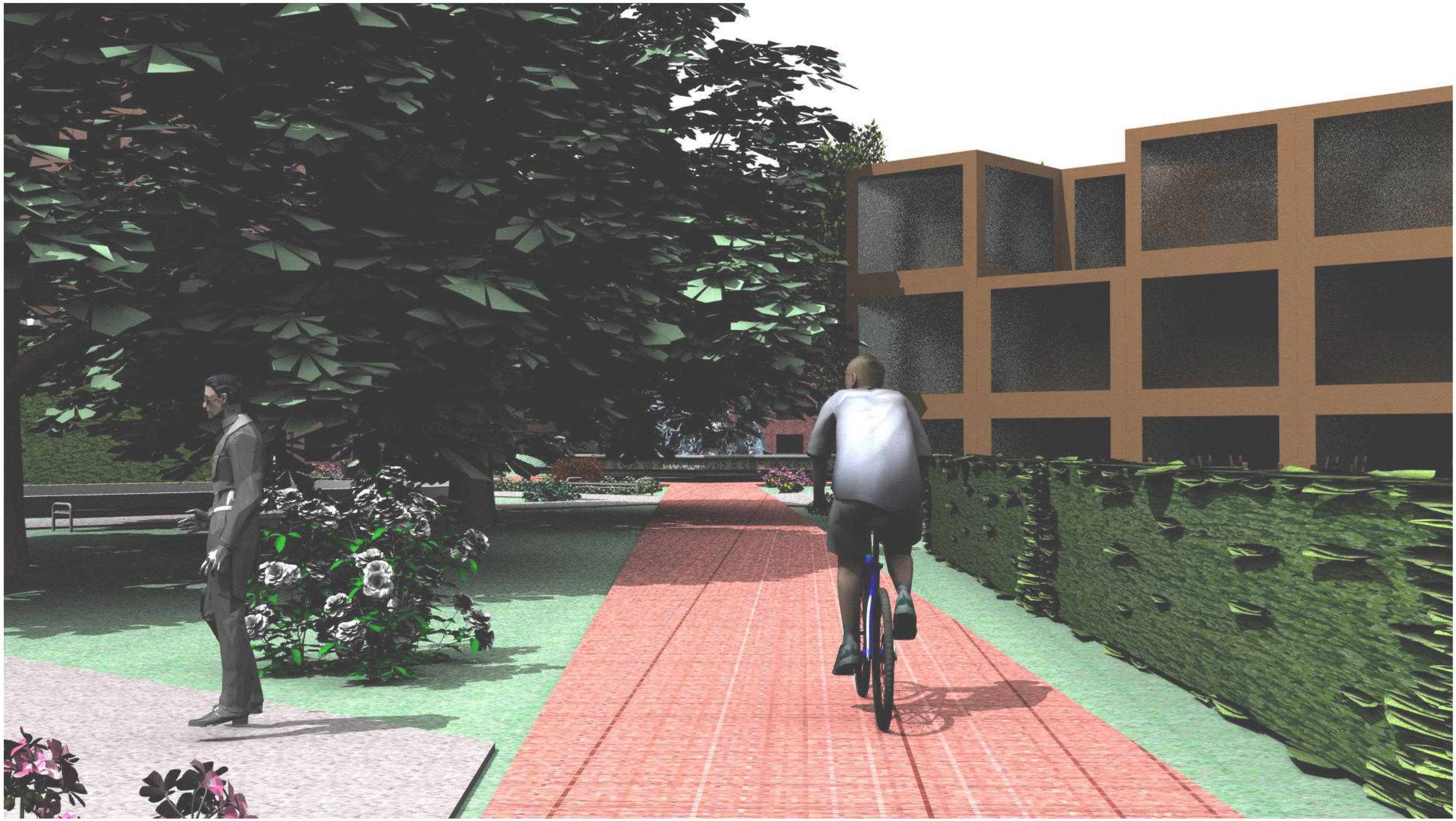
Materials

Renders









Questions?