ZOHO CUB3D

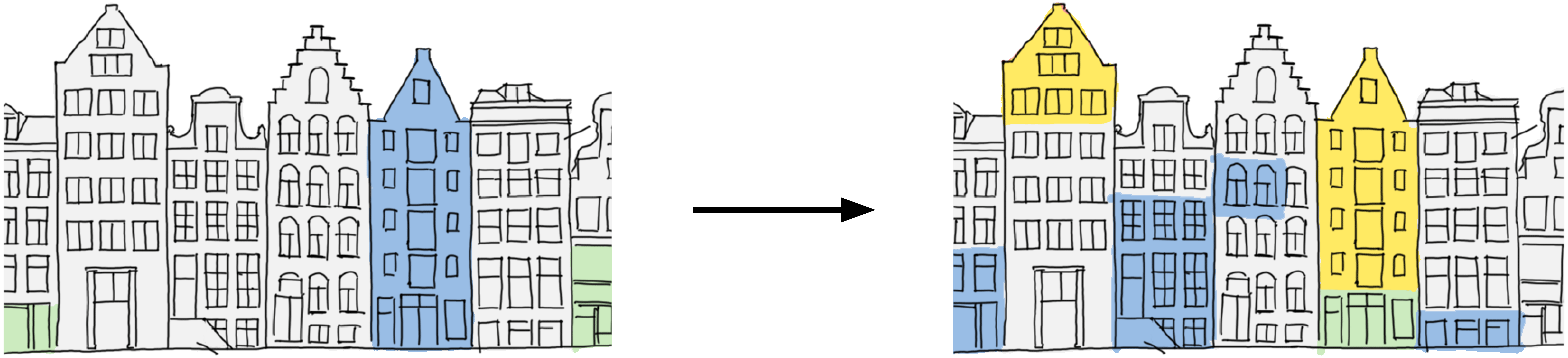
Hugo van Rossum \\ Maren Hengelmolen \\ Liva Sadovska \\ Sander Bentvelsen

Densify the city with sustainable living and working space, which benefit both the user and neigbouring community, with tailored modular and flexible units.

Planning

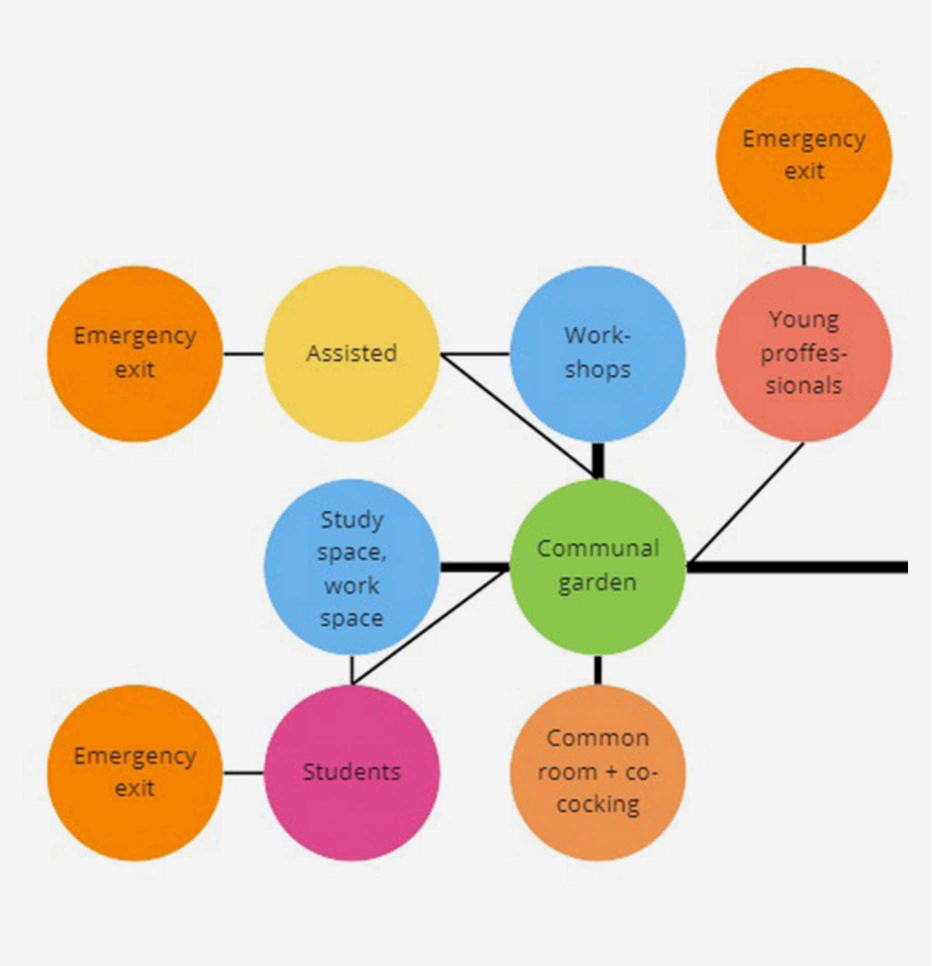
```
tllib immediate_context2.mtl
semtl diffuse_Black
 -98.2011337 -26.1900005 16.191867800000001
 -100.093903 -35.261440299999997 14.177298499999999
 -98.118652299999994 -35.244319900000001 16.1606369
 -100.24218 -23.928264599999999 14.1758118
 -100.24218 -23.928264599999999 14.1758118
 -100.093903 -35.261440299999997 12.591865500000001
 -100.09281900000001 -35.803661300000002 1.4069900500000001
 -100.09281900000001 -35.803661300000002 12.591865500000001
 -100.24475099999999 -23.542282100000001 1.4069900500000001
 -100.24475099999999 -23.542282100000001 13.8321896
 -97.761390700000007 -35.796188399999998 1.4069900500000001
 -97.761390700000007 -35.796188399999998 12.591865500000001
 -97.750381500000003 -37.921386699999999 1.4069900500000001
 -97.750381500000003 -37.921386699999999 12.591865500000001
 -95.181633000000005 -37.933364900000001 1.4069900500000001
 -95.181633000000005 -37.933364900000001 12.591865500000001
 -95.350120500000003 -23.4732895 13.841865500000001
 -95.218528699999993 -35.2453918 12.591865500000001
 -95.350120500000003 -23.4732895 1.4069900500000001
 -95.218528699999993 -35.2453918 13.841865500000001
 -98.118652299999994 -35.244319900000001 12.591865500000001
 -100.450073 -35.251792899999998 12.591865500000001
 -105.39431 -35.279693600000002 12.591865500000001
 -105.37593099999999 -37.9424858 12.591865500000001
 -102.961838 -35.818519600000002 12.591865500000001
 -103.150261 -35.258884399999999 12.591865500000001
 -102.963425 -37.9529724 12.591865500000001
 -105.39431 -35.279693600000002 13.731510200000001
 -105.532951 -23.590339700000001 13.731510200000001
 -105.37593099999999 -37.9424858 1.3314895600000001
 -105.532951 -23.590339700000001 1.3314895600000001
```

Main project goal:
Sustainability & Flexibility



Less **waste**. Less **emission**. Less **material use**.

Additional design goals

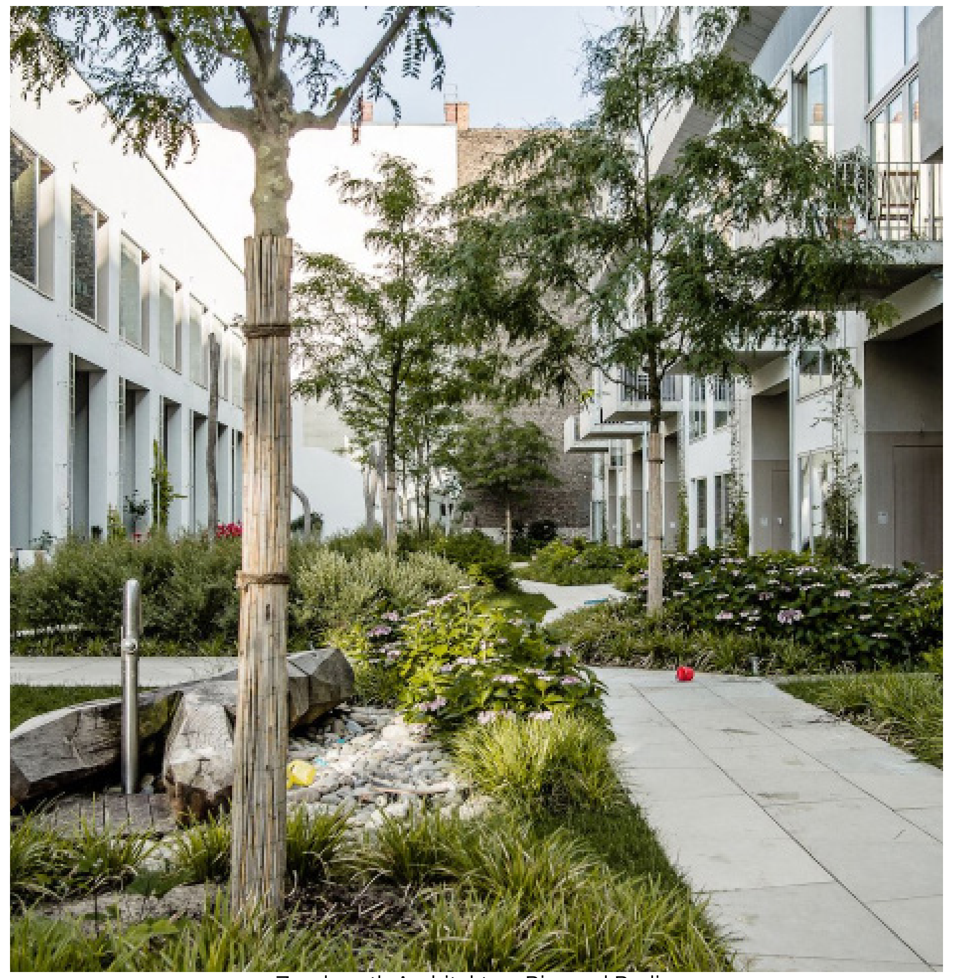


Creating Clusters

The residential functions are **clustered** around their preferred communal node (for example, the study space). This way they are more **accessible** to those that use them the most, while also **separating** the users with different lifestyles.

Separating public/private

A privacy gradient ensures **separation** between the public and private areas inside the building, while in between communal areas serve as **transition**. This way the residents can enjoy a peaceful and quiet living space, without them having to worry about noise or compromised **privacy**.



Zanderroth Architekten: Big yard Berlin

Outdoor Garden

All residential units are **connected** to the central communal garden. This way, they all have access to a pleasant open and green area to relax in. Furthermore, commuting through it **stimulates encounters** between neighbours.

De Urbanisten: Hofbogenpark

Activating the street

The Vijverhofstraat is 'activated' with **opportunities** for people to dine and shop there. this aligns with the city's plan to turn the old metroline into a '**Highline**'. This contributes to the amount of **visitors** and significance of the area.

Program of requirements

Housing

Student housing 80 units
Assisted living 30 units
Starter housing 100 units

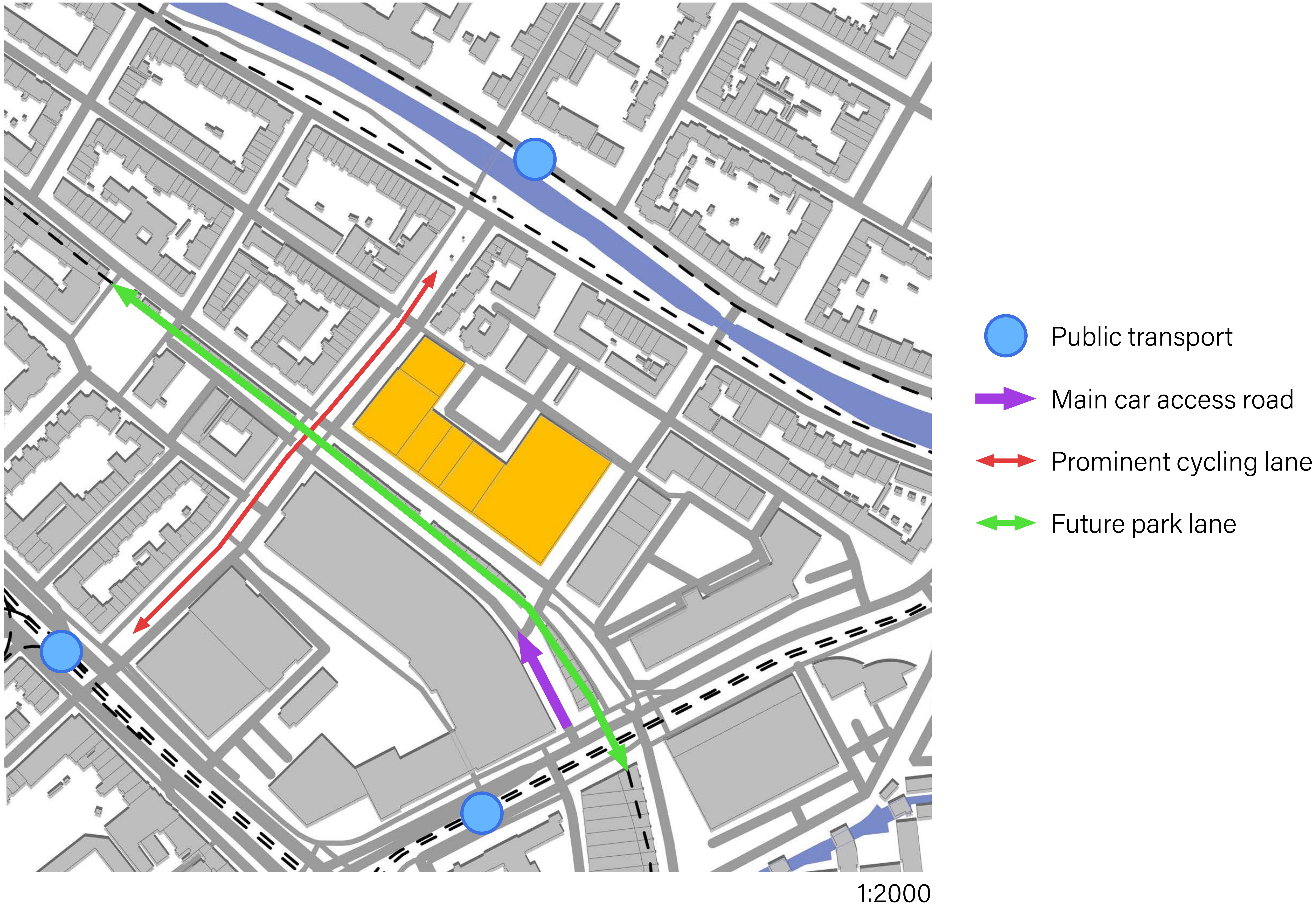
Communal Spaces

Underground parking (0.5 parking lots per apartment or more)
Communal garden
Workshop
Common room (co-cooking)
Study space
Bike parking (1 per resident)

Public Spaces

Shared car parking
Hub
Community center
Library
Music rooms
Offices
Gym
Makerspace

Context connection analysis



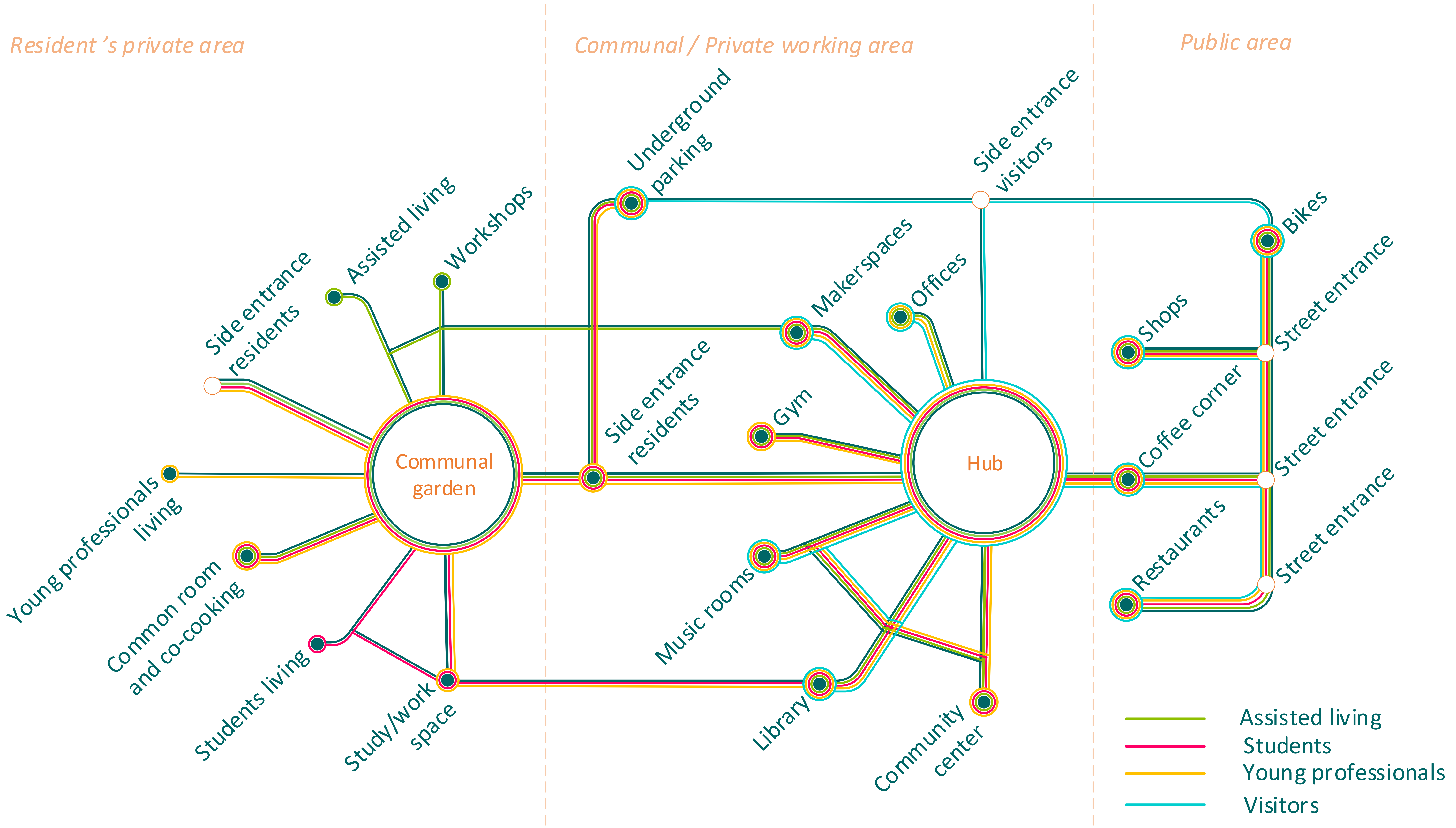
Residential perspectives

			Private area				Communal/Private working area								Public area				Scores		
Possibility to:	Residents	Weight	Workshops spaces	Study-/workspaces	Communal garden	Common room	Hub	Community center	Library	Music rooms	Gym	Makerspaces	Offices	Underground parking	Restaurants	Coffee corner	Shops	Underground bikes	Assisted living	Students	Young professionals
work in a quiet space	Assisted living	1																	2		
	Students	3																		6	
	Young professionals	3																			9

Residential perspectives

			Private area				Communal/Private working area								Public area				Scores		
Possibility to:	Residents	Weight	Workshops spaces	Study-/workspaces	Communal garden	Common room	Hub	Community center	Library	Music rooms	Gym	Makerspaces	Offices	Underground parking	Restaurants	Coffee corner	Shops	Underground bikes	Assisted living	Students	Young professionals
work in a quiet space	Assisted living	1																	2		
	Students	3																		6	
	Young professionals	3																			9
work in groups	Assisted living	2																	4		
	Students	3																		3	
	Young professionals	3																			6
rest in a quiet space	Assisted living	3																	6		
	Students	2																		4	
	Young professionals	2																			4
be entertained	Assisted living	3																	12		
	Students	3																		9	
	Young professionals	3																			9
interact with other residents and meet new people	Assisted living	2																	22		
	Students	3																		27	
	Young professionals	2																			22
shop for groceries	Assisted living	3																	3		
	Students	3																		3	
	Young professionals	3																			3
start or to start working at a company	Assisted living	2																	4		
	Students	2																		0	
	Young professionals	3																			4
store bikes or cars	Assisted living	2																	4		
	Students	2																		4	
	Young professionals	3																			4
go out	Assisted living	2																	4		
	Students	3																		6	
	Young professionals	2																			4
Total score																			61	62	65

Metro diagram



Configuring

REL chart

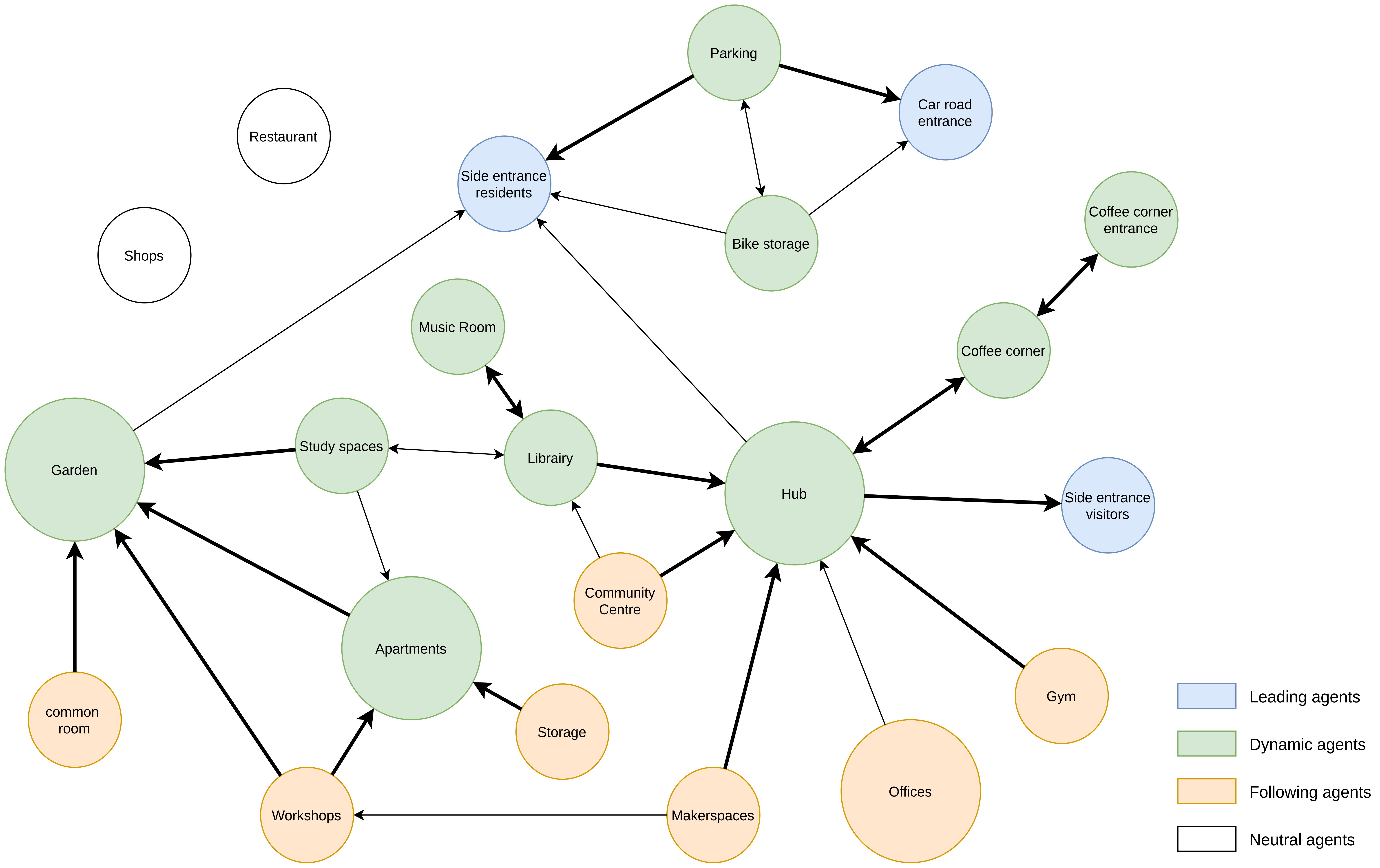
Relative relations

Programs	Underground Parking	Underground bikes	Communal garden	Workshops	Coffee corner	Culture center (music room)	Culture center (library)	Restaurant	Common room / Co-cooking	Community Centre	Gym	Offices	Street entrance to coffee corner	Side entrance residents SE	Side entrance residents NE	Side entrance visitors	Makerspaces	Study space	Hub	Students	Assisted	Young professionals	Storage	Car/ bycicle road entrance	Shops
Underground Parking	1	0.2	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0.3	0	0	0	0	0	0	0	1	0
Underground bikes	0.5	0.3	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0.8	0
Communal garden	0	0	1	0	0	0	0	0	0	0	0	0	0	0.3	0.3	0	0	0	0	0.2	0.2	0.2	0	0	0
Workshops	0	0	0.5	1	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0	0	0	0.2	0	0	0	0
Coffee corner	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0.8	0	0	0	0	0	0
Culture center (music room)	0	0	0	0	0	0.5	0.8	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0
Culture center (library)	0	0	0	0	0	0.2	1	0	0	0	0	0	0	0	0	0	0	0.2	1	0	0	0	0	0	0
Restaurant	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Common room / Co-cooking	0	0	0.8	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Community Centre	0	0	0	0	0	0	0.5	0	0	1	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0
Gym	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0
Offices	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0.3	0	0	0	0	0	0
Street entrance to coffee corner	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Side entrance residents SE	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Side entrance residents NE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Side entrance visitors NW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Makerspaces	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0.5	0	0	0	0	0	0
Study space	0	0	0.5	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0.5	0	0.2	0	0	0	0	0
Hub	0	0	0	0	0	0	0	0	0	0	0	0	0.8	0.5	0	0.8	0	0	1	0	0	0	0	0	0
Students	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0.5	0	0	0	0	0
Assisted	0	0	0.6	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0
Young professionals	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0
Storage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Car road entrance	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Shops	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Relative preferences

Programs	Sun acces	Sky view factor	Ground floor preference	Subterranean preference	Closeness street entrance coffee corner	Closeness side entrance visitors NW	Closeness car road entrance	Closeness to NE facade	Closeness to NW facade	Closeness to SE facade	Closeness to SW facade	Closeness to facade
Underground Parking	0	0	0	1	0	0	0	0	0	0	0	0
Underground bikes	0	0	0	1	0	0	0	0	0	0	0	0
Communal garden	1	1	1	0	0	0	0	0	0	0	0	0
Workshops	0.5	0	0.5	0	0	0	0	0	0	0	0	0
Coffee corner	0.5	0	1	0	0	0	0	0	0	0	0	0.3
Culture center (music room)	0.3	0	0.3	0	0	0	0	0	0	0	0	0
Culture center (library)	0.6	0	0.3	0	0	0	0	0	0	0	0	0
Restaurant	0.3	0	0.5	0	0	0	0	0	0	0	1	0.5
Common room / Co-cooking	0.5	0.2	0.5	0	0	0	0	0	0	0	0	0
Community Centre	0.4	0.2	1	0	0	0	0	0	0	0	0	0
Gym	0.4	0	0.3	0	0	0	0	0	0	0	0	0.3
Offices	0.5	0.5	0	0	0	0	0	0	0	0	0.8	1
Street entrance to coffee corner	0	0	1	0	1	0	0	0	0	0	0	1
Side entrance residents SE	0	0	1	0	0	0	0	0	0	1	0	1
Side entrance residents NE	0	0	1	0	0	0	0	1	0	0	0	1
Side entrance visitors NW	0	0	1	0	0	1	0	0	0.5	0	0	1
Makerspaces	0.3	0	0.8	0	0	0	0	0	0	0	0	0.3
Study space	0.8	0.2	0	0	0	0	0	0	0	0	0	0
Hub	0.5	0.8	1	0	0.3	0	0	0	0	0	0	0
Students	0.7	0.5	0	0	0	0	0	0	0	0	0	0
Assisted	0.7	0.5	0.3	0	0	0	0	0	0	0	0	0
Young professionals	0.7	0.5	0	0	0	0	0	0	0	0.5	0	1
Storage	0	0	0	1	0	0	0	0	0	0	0	0
Car road entrance	0	0	1	0	0	0	1	0	0	1	0	1
Shops	0.3	0	1	0	0	0	0	0	0	0	1	0

Growth hierarchy



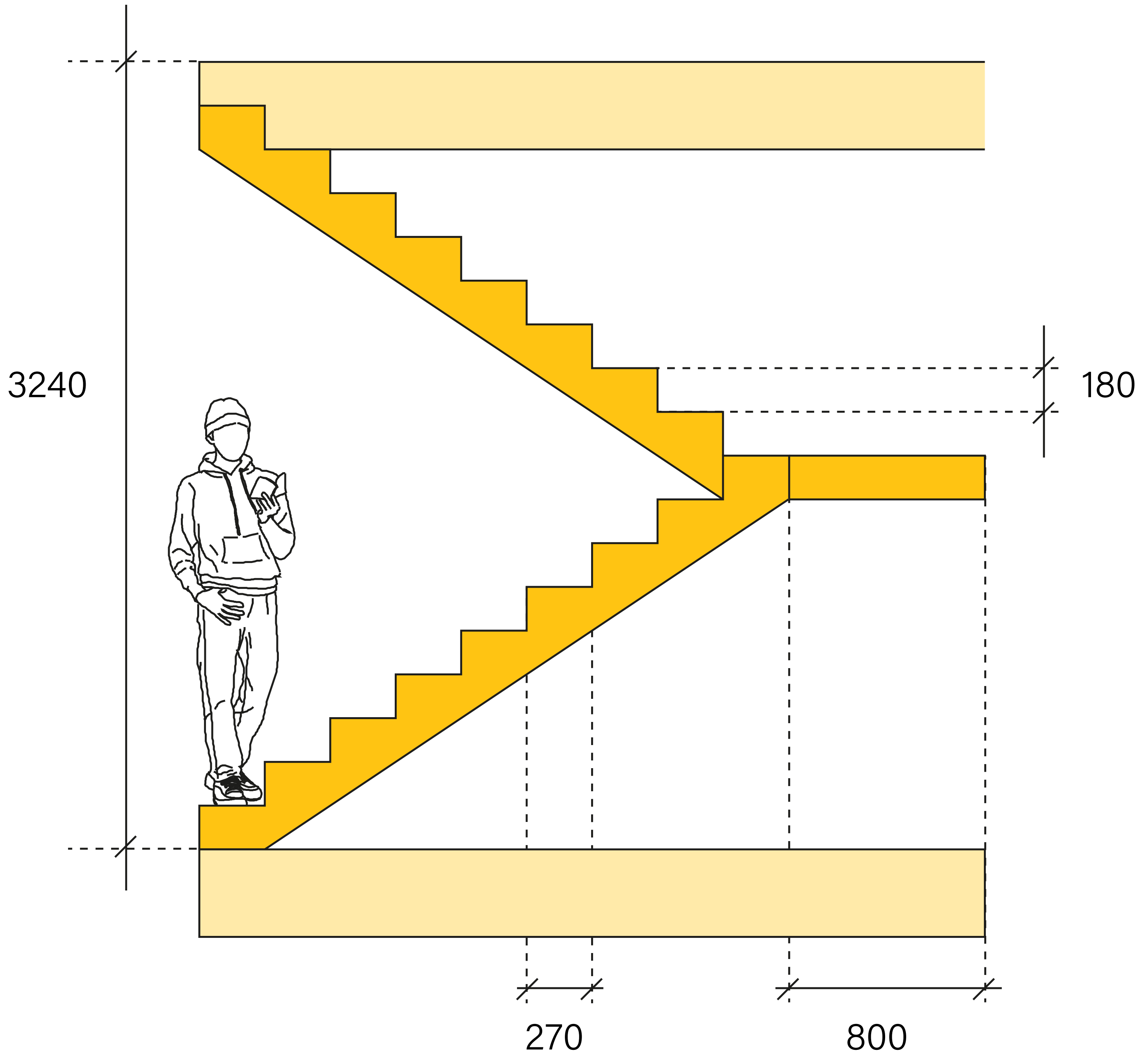
Programs	Underground Parking	Underground bikes	Communal garden	Workshops	Coffee corner	Culture center (music room)	Culture center (library)	Restaurant	Common room / Co-cooking
Underground Parking	1	0.2	0	0	0	0	0	0	0
Underground bikes	0.5	0.3	0	0	0	0	0	0	0
Communal garden	0	0	1	0	0	0	0	0	0
Workshops	0	0	0.5	1	0	0	0	0	0
Coffee corner	0	0	0	0	1	0	0	0	0
Culture center (music room)	0	0	0	0	0	0.5	0.8	0	0
Culture center (library)	0	0	0	0	0	0.2	1	0	0
Restaurant	0	0	0	0	0	0	0	1	0
Common room / Co-cooking	0	0	0.8	0	0	0	0	0	0.5

An extension of the REL chart

According to our design strategy with privacy gradients and the decision to cluster functions around hubs, a hierarchy of spaces arises. When the growth algorithm seeds and grows spaces, the matrix is used to look up which spaces should grow or "follow" which spaces. However, not every space finds it important to follow another. Some spaces are dependant on the location of the hubs but the hubs themselves are not affected by the spaces following them. This relationship indicated in the matrix by lack of symmetry across the diagonal.

The following bubble diagram illustrates the meaning of this asymmetry along the diagonal in the REL chart. For example, the co-cooking area and community garden are connected in the metro diagram, this is also reflected in the REL chart. However, because the co-cooking area indicates that it would need to grow towards the garden, and the garden does not indicate any preference for growing towards the co-cooking, a hierarchy arises: co-cooking follows the garden, not the other way around.

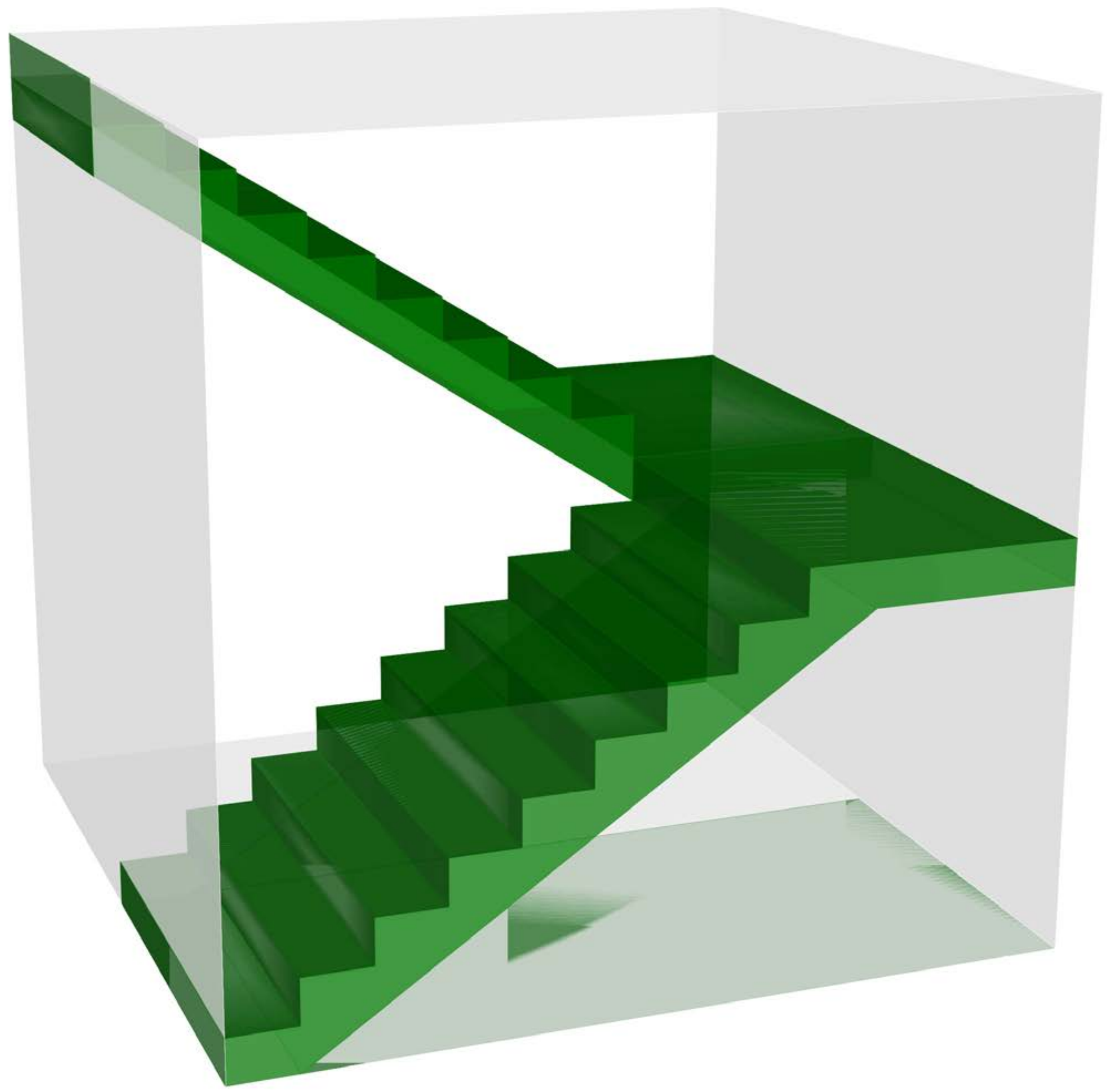
Voxel size



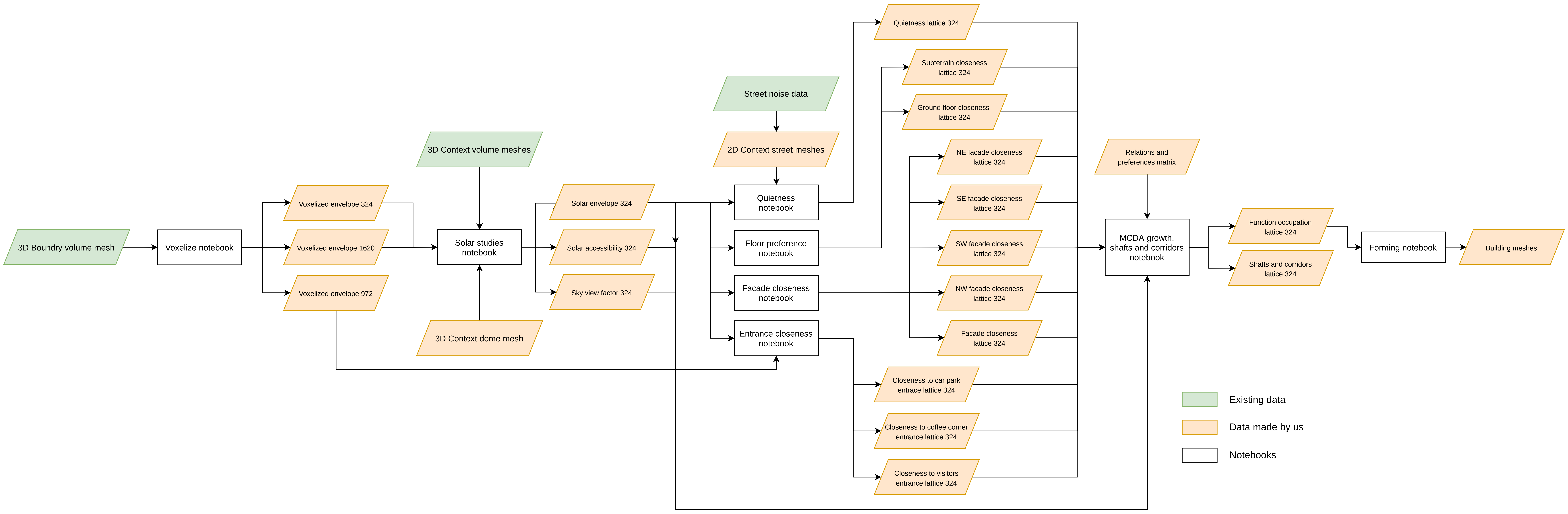
Final voxel size: **3240 x 3240**

Why this size?
- Height & width = ceiling height
- Staircase from floor to floor should fit
- Staircase is fit for multiple functions (residential, commercial)
- Can be a size for tiles of different functions

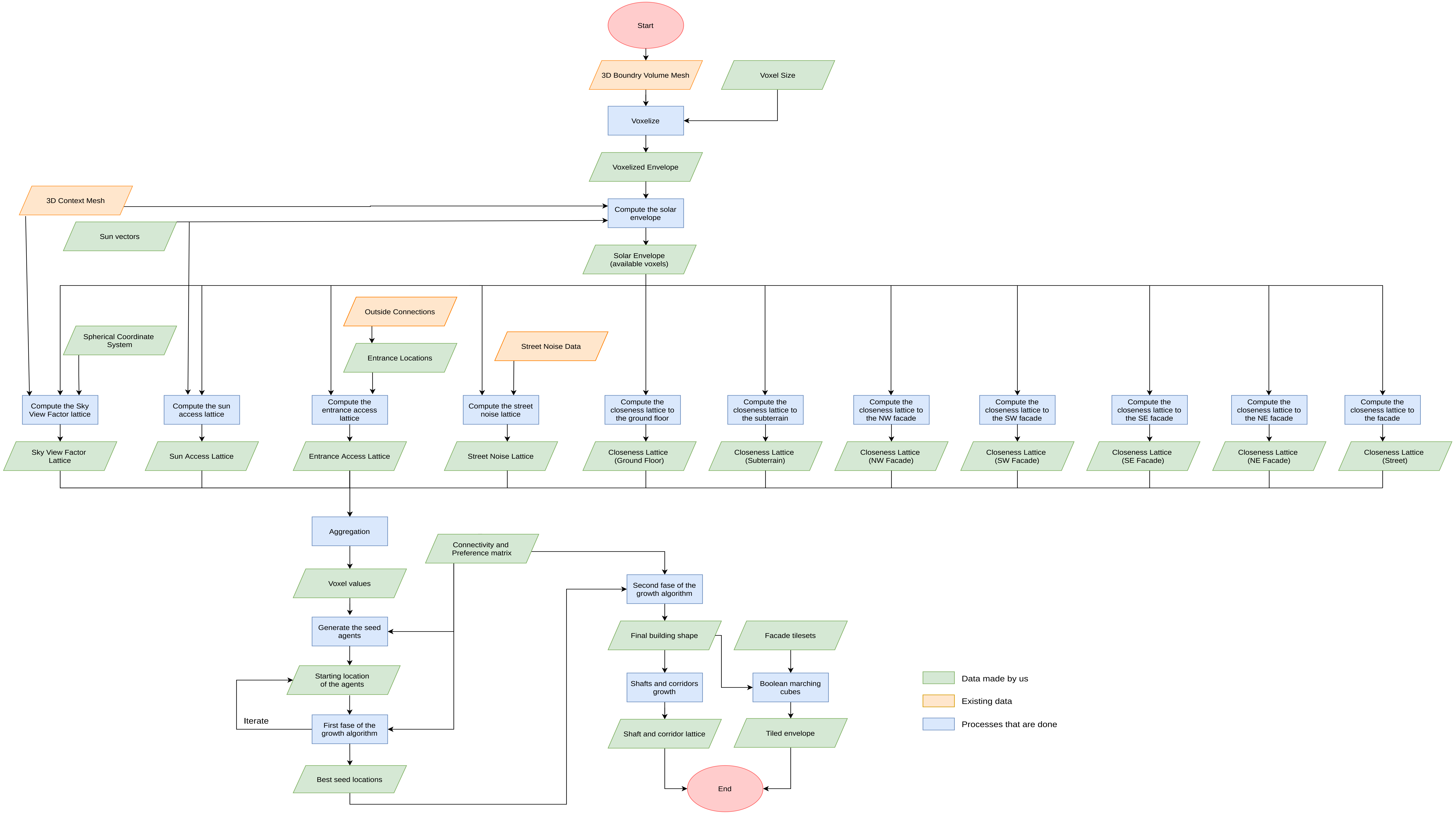
Bouwbesluit
- Width stairs: minimum is 800 mm
- Riser: minimum is 180 mm
- Tread width: minimum is 220 mm
- Head room: minimum is 2300 mm



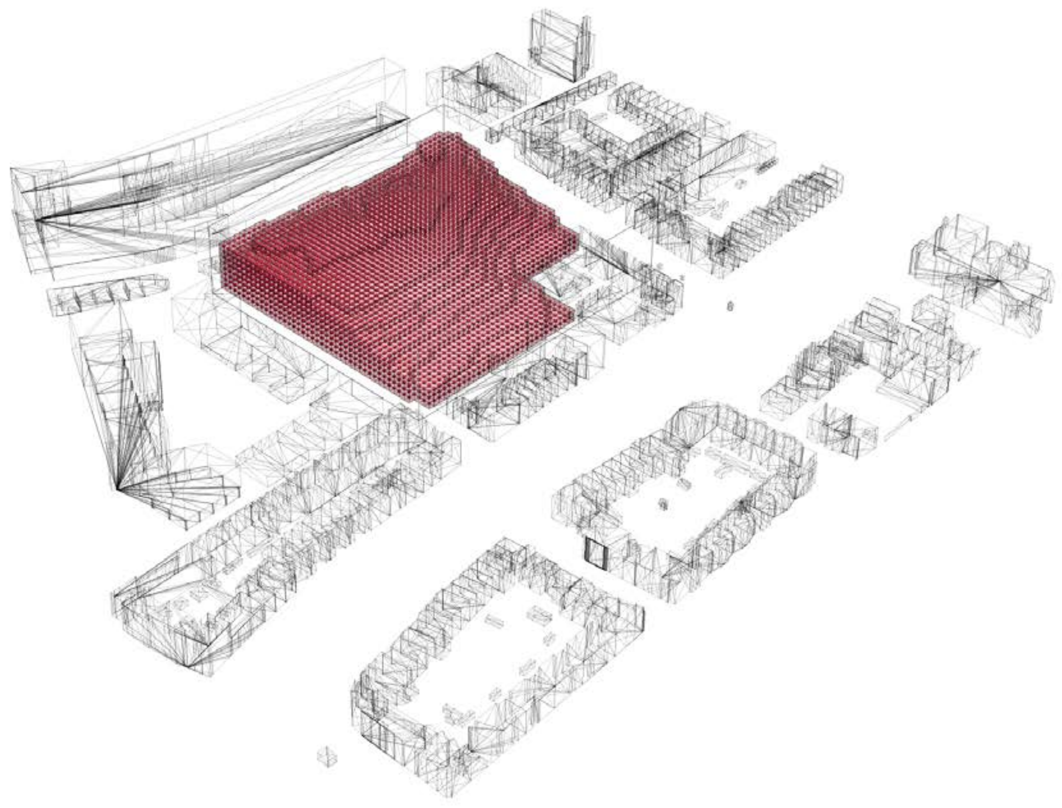
Notebook Flowchart



Computation Overview Flowchart



Solar envelope



Create an envelope based on solar blockage

The created envelope will be used as the base availability lattice on which all other calculations for static data and the growing of the agends are built upon.

Input: Voxelized envelope, context mesh

Output: Solar envelope

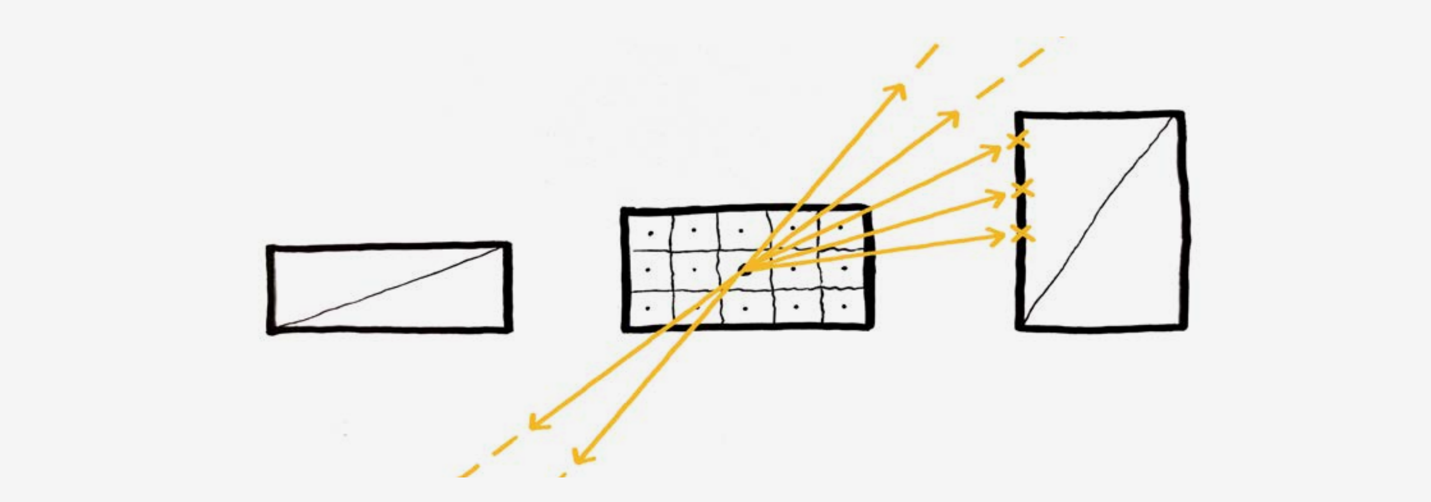
Create a list of all vectors pointing towards the sun locations over the year

For all voxels inside of the envelope:
    Cast a ray from the list of sun vectors from the voxel centroid
    If the ray intersects with a mesh:
        Ignore the ray and continue the loop
    Else:
        Check if the reversed ray intersects with a mesh
        If this new ray intersects with a mesh:
            Register the intersection for the voxel to a new list
        Else:
            Register a non intersection to the list

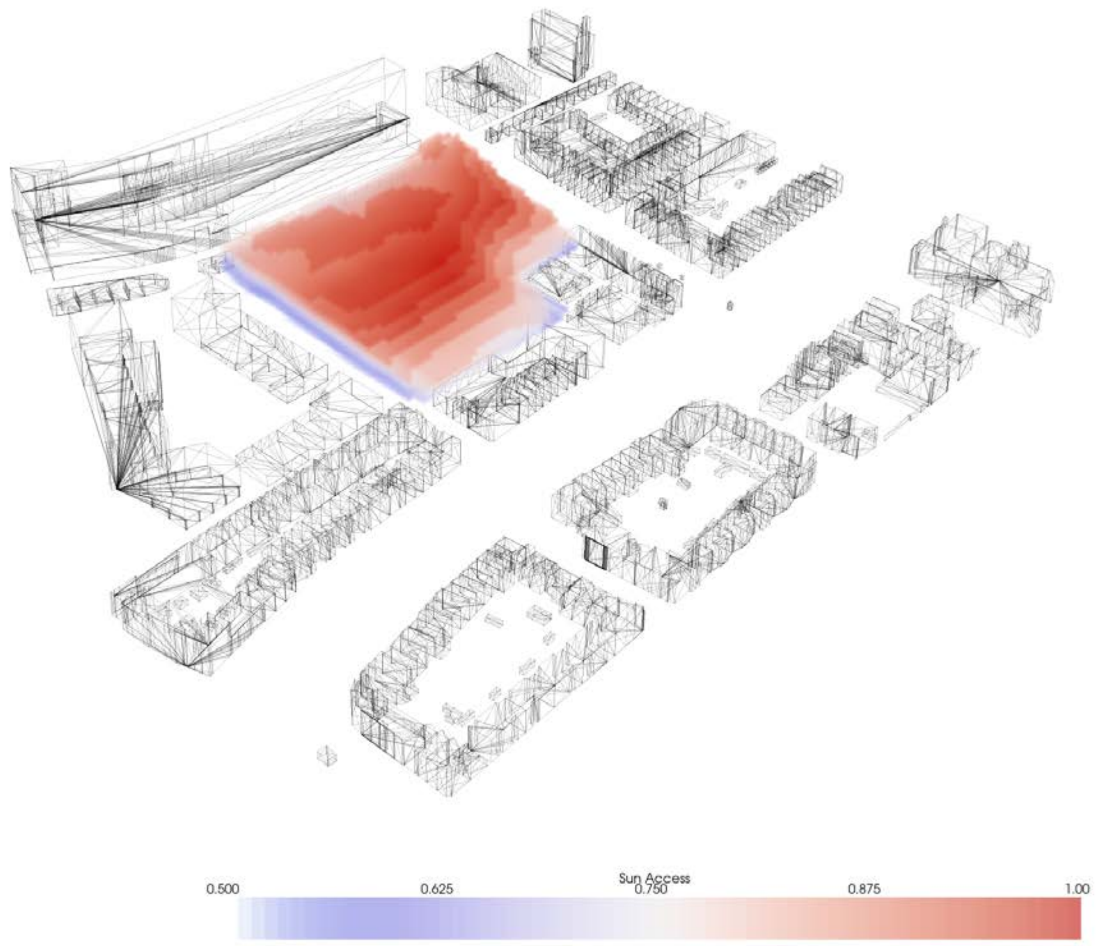
For each voxel inside the envelope:
    Map the amount of intersections in a range between 0 and 1, where 0 means blocking a lot of light for neighbouring buildings and 1 not blocking any light.

Set a limit to how much light the voxels are allowed to block and create a new lattice with either True or False values, depending on the amount of light blocked

Export this lattice as the new availability lattice



Solar accessibility



Ensure spaces get enough sunlight

This data is used for the growing algorithm by certain agents that prefer a high solar accessibility, for instance: the residential quarters and study spaces.

Input: Solar envelope, context mesh

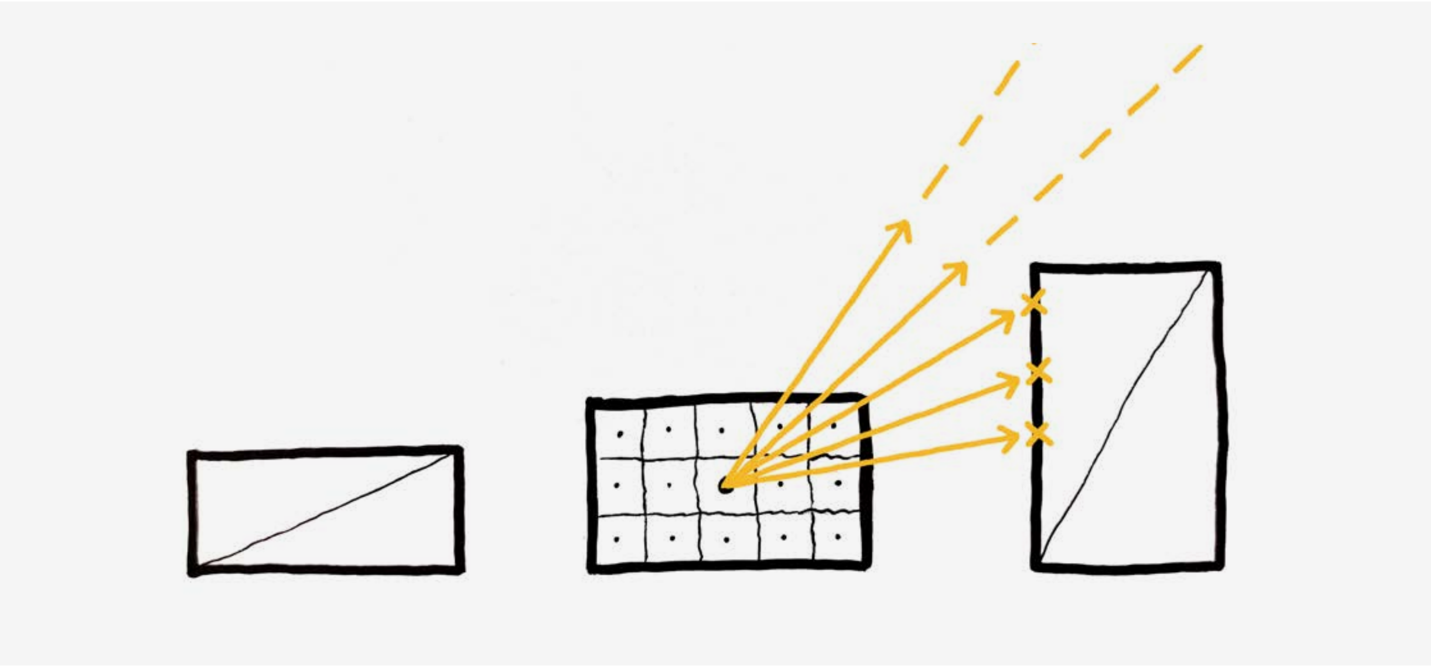
Output: Solar accesibility lattice

Create a list of all vectors pointing towards the sun locations over the year

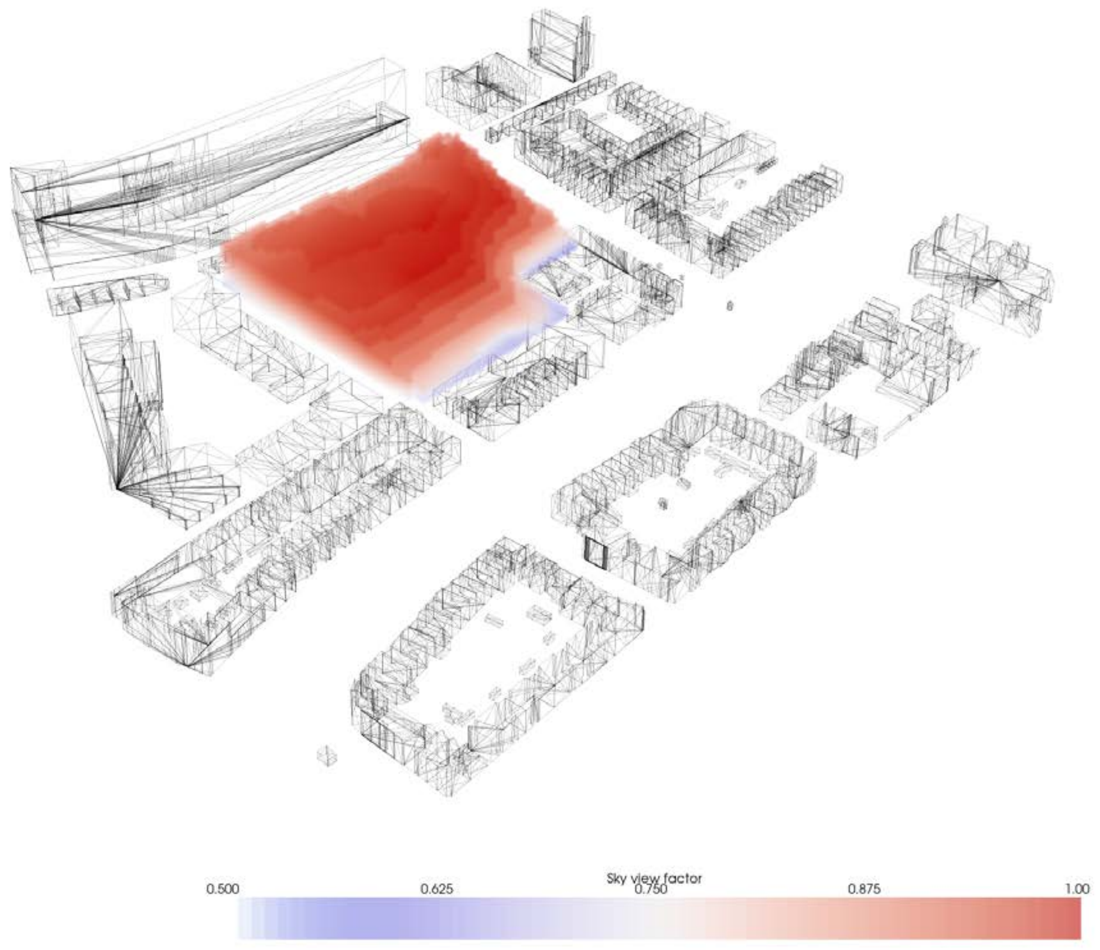
For all voxels inside of the envelope:
    Cast a ray from the list of sun vectors from the voxel centroid
    If the ray intersects with a mesh:
        Append an intersection to a new list
    Else:
        Append a non intersection to the list

For each voxel inside the envelope:
    Map the amount of intersections in a range between 0 and 1, where 1 means receiving the most of light and 0 receiving the least amount of light

Export the newly created lattice that lists the values of solar accessibility in a range from 0 to 1



Sky view factor



Ensure functions are able to see enough of the sky

This data is used for the growing algorithm by certain agents that prefer a high sky view factor, for instance: the office spaces and garden.

Input: Solar envelope, context mesh, dome mesh

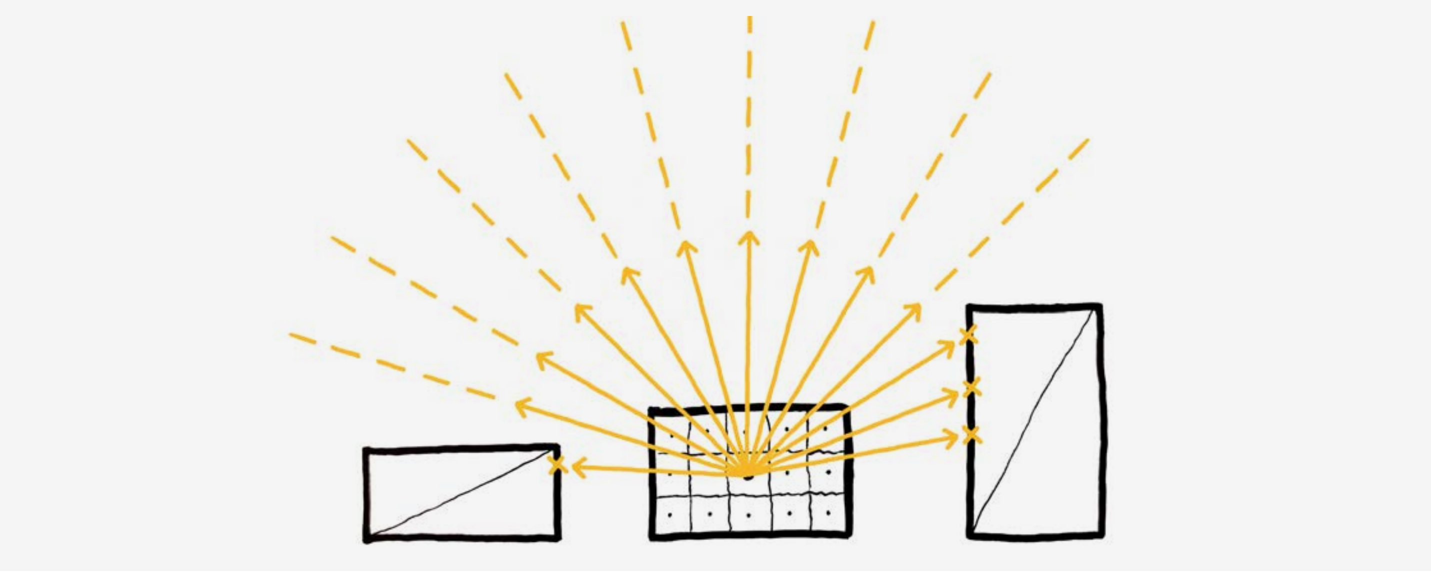
Output: Sky view factor lattice

Instead of creating a list of vectors pointing towards the sun locations over the year, append the normals of a dome mesh to a list, created to map the sky in equal proportions

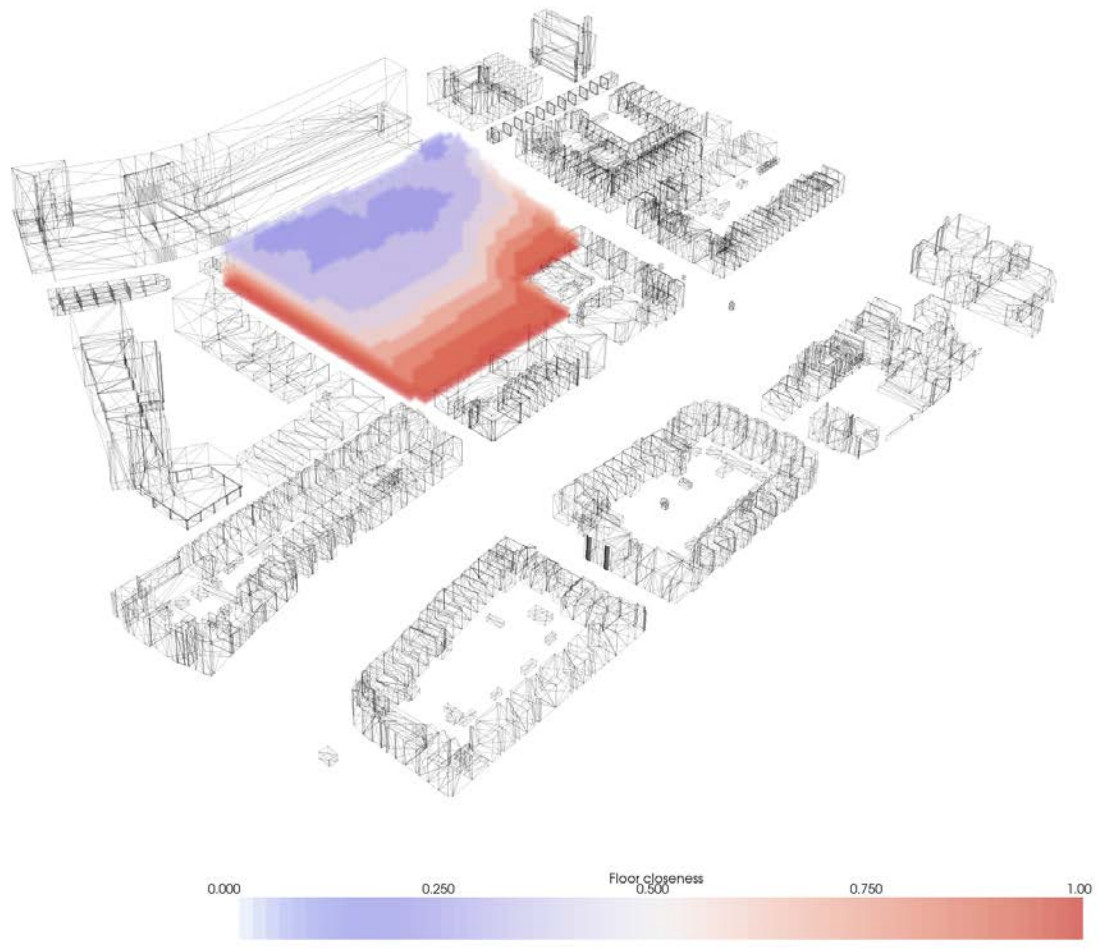
For all voxels inside of the envelope:
  Cast a ray from the list of normals from the voxel centroid
  If the ray intersects with a mesh:
    Append an intersection to a new list
  Else:
    Append a non intersection to the list

For each voxel inside the envelope:
  Map the amount of intersections in a range between 0 and 1, where 1 has the least intersections, which means having a high sky view factor and 0 the opposite

Export the newly created lattice that lists the values of the sky view factor in a range from 0 to 1



Floor level preference



Setting floor levels for agents

This data is used for the growing algorithm by certain agents that prefer a proximity to certain floors, for instance: the hub and garden prefer to be on the ground floor.

Input: Solar envelope

Output: Floor level preference

Create a list of entries based on the height of the imported lattice

Create a matrix that maps the neighbouring entries as if connected from bottom to top

Select an entry as you would select a floor level (in the visualization it's 0)

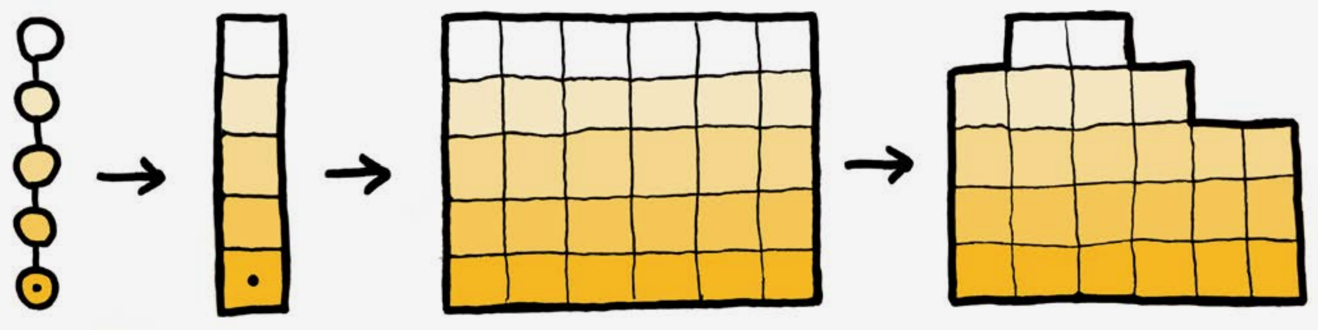
Calculate the distance from that entry to every other one

Map the values from 0 to 1, where 1 is the entry itself and 0 for the entry that is the furthest from the selected one. Then append them from bottom to top to in a one dimensional array

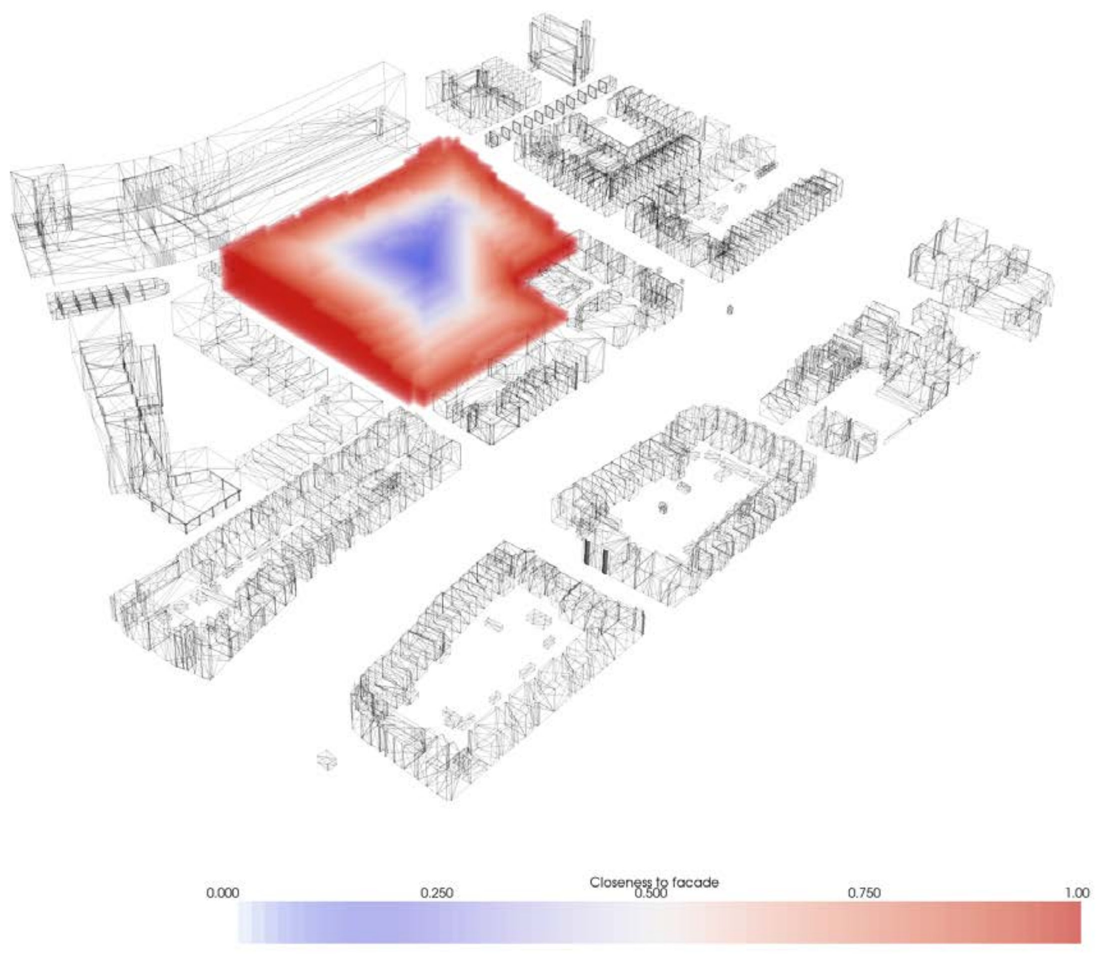
Map this array along the z-axis of the entire imported lattice

Multiply this newly created lattice with the solar envelope to set all unoccupied voxels to 0 and export it

Note:
The reason the Floyd–Warshall algorithm isn't used here for the full envelope is because as it is, it's too heavy to run for the selected voxel size. For now, we are using a custom algorithm to get a higher resolution.



Closeness to the facade (high resolution)



Ensure access to the facade

This is another parameter to optimize the placement of spaces that need direct daylight or adjacency to the street.

Input: Avalability lattice, Custom Stencil

Output: Facade closeness lattice

Define stencil as Von Neumann neighborhood with top and bottom neighbors removed

Apply the stencil to the voxel envelope

Find the number of neighbors for each voxel in the lattice

Create a condition for boundary voxels, where the number of neighbors is < 4, then select only the ground level voxels

Check envelope with the condition, create a new envelope with only boundary voxels

For each available voxel inside a 2D slice of the envelope:

    Append the ID's of its neighbours to an adjacency list

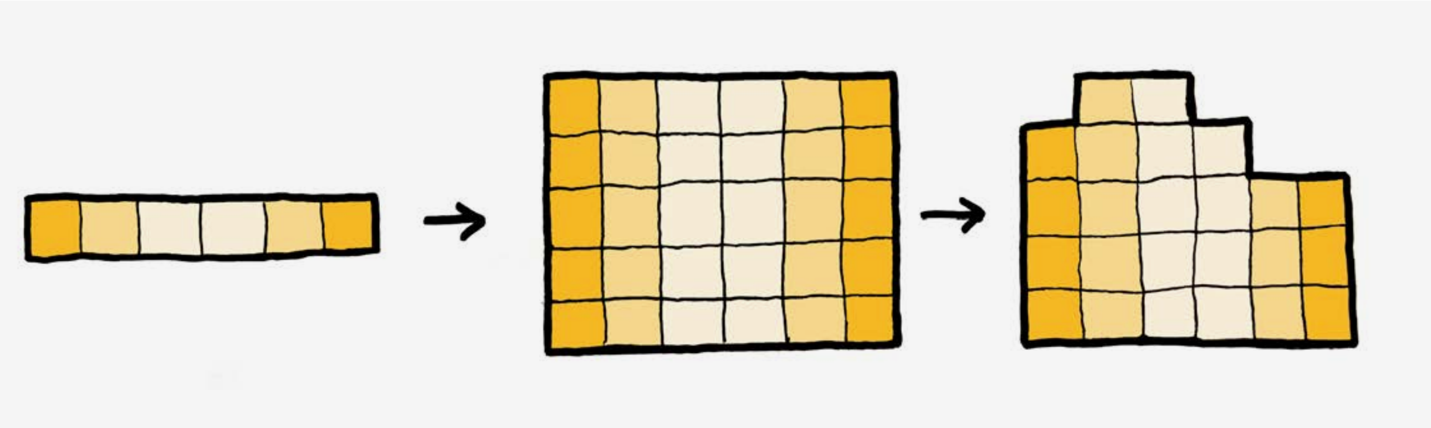
Create a sparce matrix that contains the connectivity data

Compute distances from all boundry voxels to all other voxels in a 2D slice

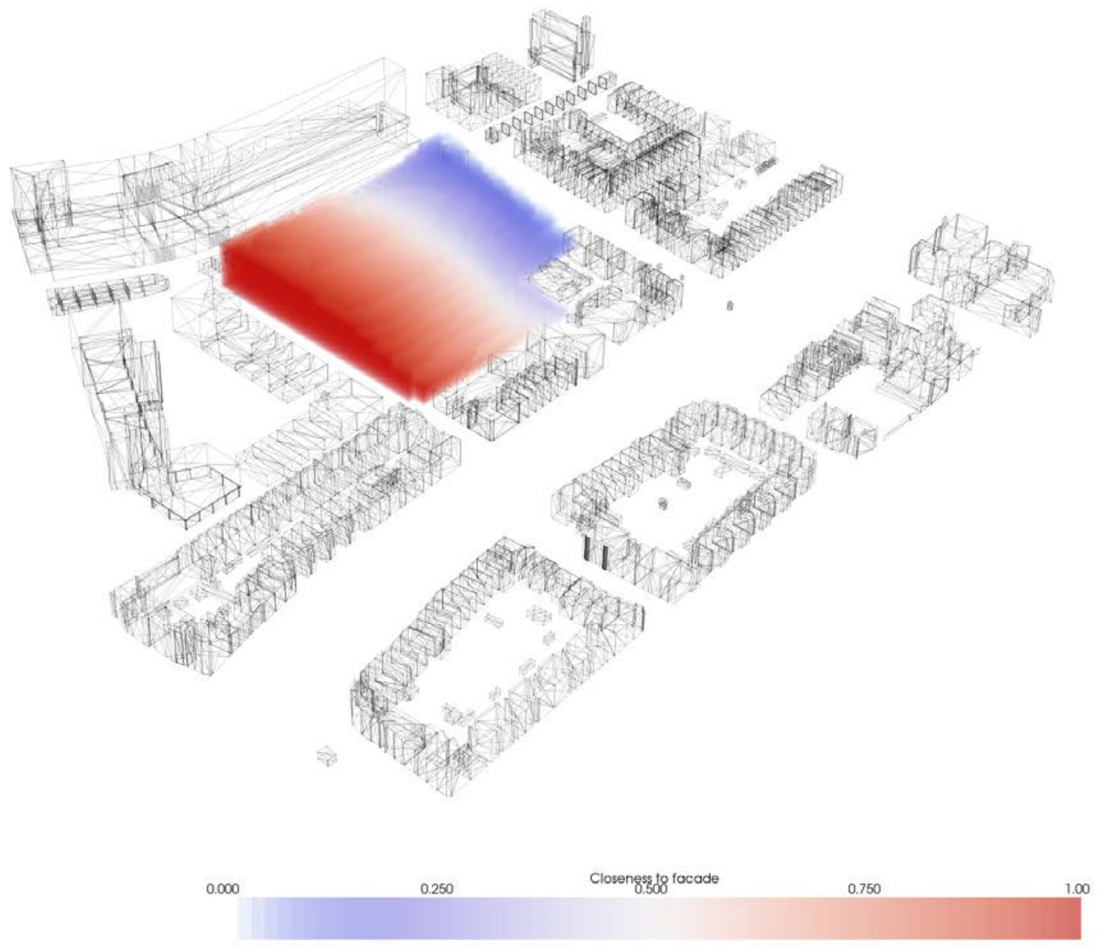
Find the minimum distance for all boundry voxels the other voxels

Add the minimum distance to the corresponding voxel value field

Map the field distance values from 0 - 1, where 0 is the furthest distance and 1 is the closest



Closeness to a specific facade (high resolution)



Orient for site accessability on a specific side

In accordance to pre-existing program, routes and greenery on the site, some spaces and entrances require access from a specific facade. By setting their preference to this facade, an axis is created along which the algorithm can seed the space.

Input: Avalability lattice, Custom Stencil

Output: Specific facade closeness lattice

Define stencil as Von Neumann neighborhood with all but one neighbour removed

Apply the stencil to the voxel envelope

Find the number of neighbors for each voxel in the lattice

Create a condition for boundary voxels, where the number of neighbors is < 1, then select only the ground level voxels

Check envelope with the condition, create a new envelope with only boundary voxels

For each available voxel inside a 2D slice of the envelope:

  Append the ID's of its neighbours to an adjacency list

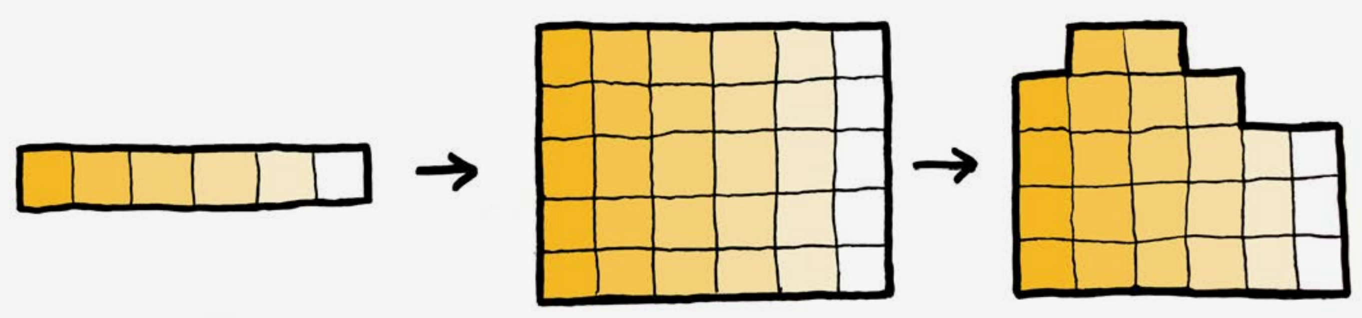
Create a sparce matrix that contains the connectivity data

Compute distances from all boundry voxels to all other voxels in a 2D slice

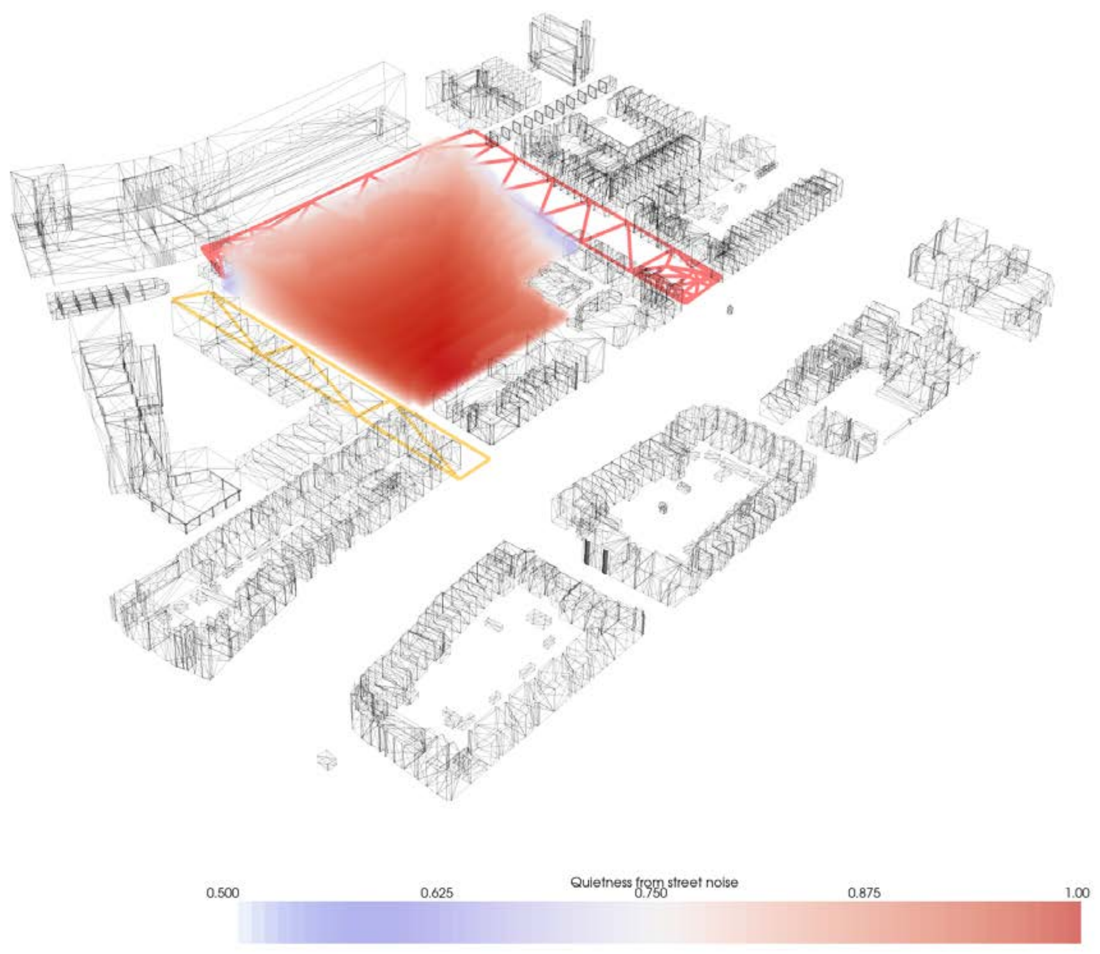
Find the minimum distance for all boundry voxels the other voxels

Add the minimum distance to the corresponding voxel value field

Map the field distance values from 0 - 1, where 0 is the furthest distance and 1 is the closest



Quietness from street noise



Orient according to traffic noise fall-off

The two main streets around the plot produce significant traffic noise. According to European Environment Agency, these streets produce 50 and 70db of noise. By mapping the noise fall-off from the street, the growth algorithm can take into account the spaces where quietness is especially preferable, such as the library.

Input: Avalability lattice, meshes representing the streets with different noise levels
Output: Quietness from street noise lattice

Load several meshes representing streets with different noise levels
Get all voxel centers as points

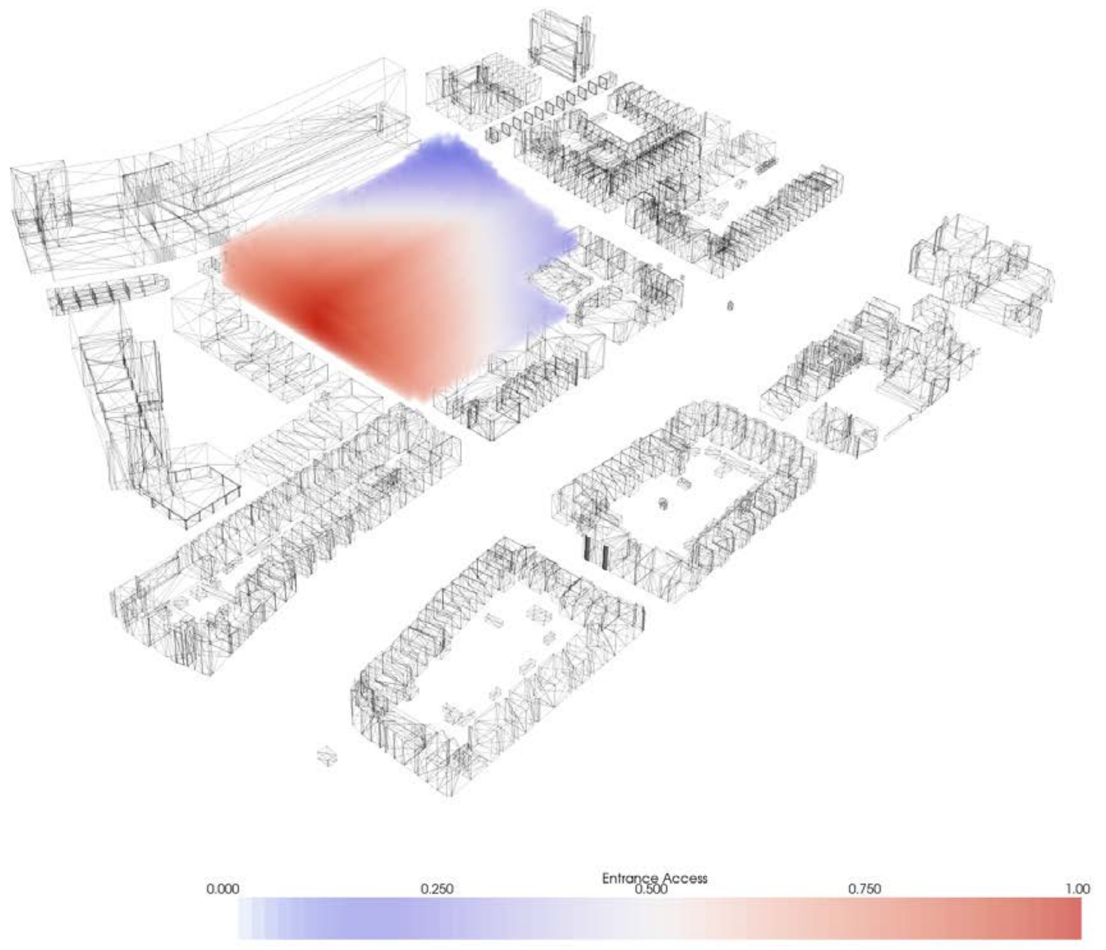
For each voxel :

- Calculate the smallest euclidian distance from voxel center to the first street mesh, using trimesh.proximity
- Using the inverse square law, calculate noise values from the acquired distance and data of level of noise on the street
- Add the noise value to the corresponding voxel in the value field

Map the inverse field of noise values to a field of quietness values from 0 - 1, where 0 is the least quiet value and 1 is the quietest value

Repeat quietness value field construction for the second street
Combine the quietness value fields by choosing the lowest quietness values for each point in the field

Entrance closeness



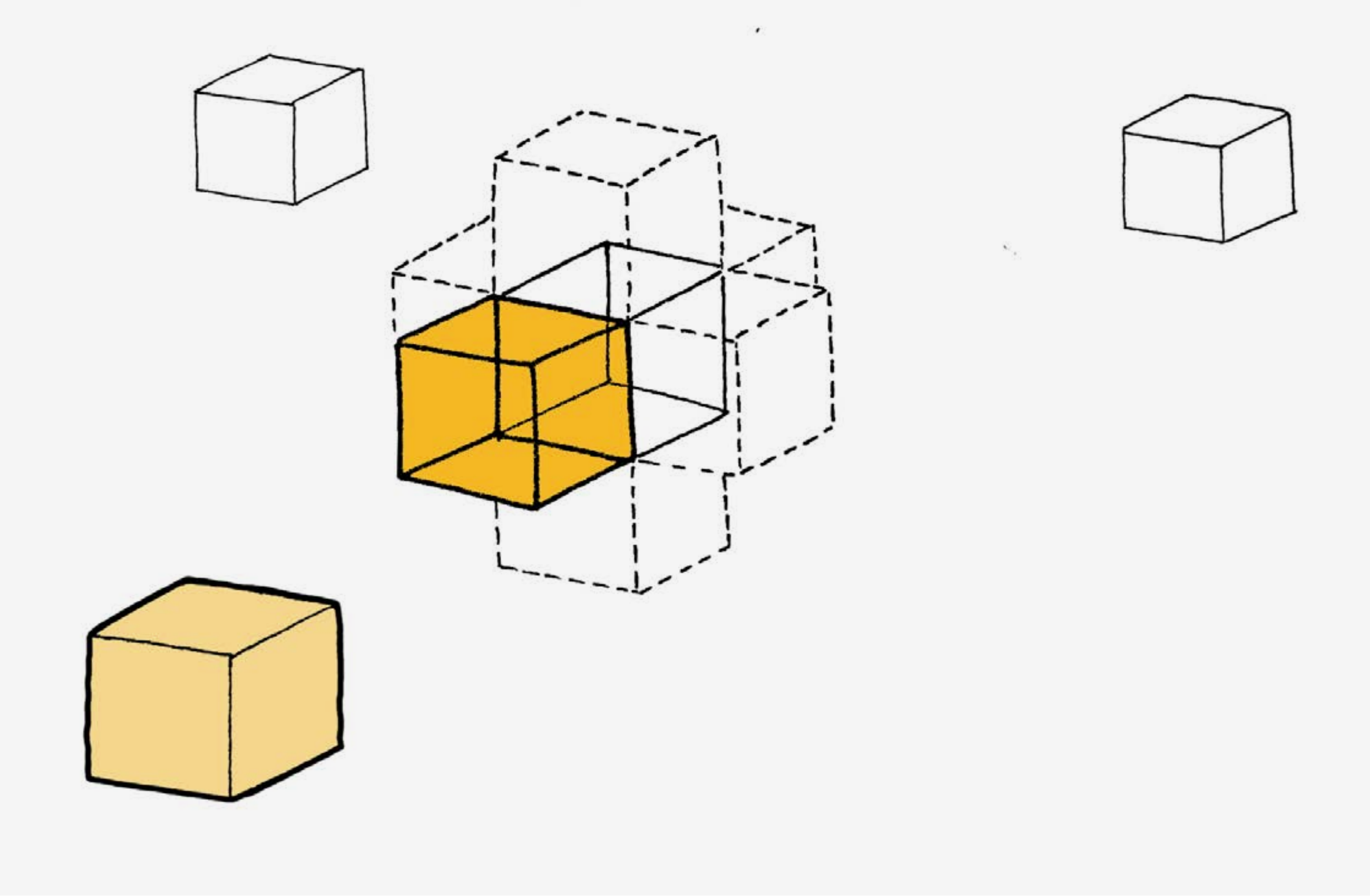
Ensure access to an entrance

To make sure the agents who need to be close to an entrace can grow in that direction, an entrance accessibility lattice must be created.

Input: Voxelized envelope, entrance locations based on street accessibility

Output: Entrance closeness Lattice

Compute the Floyd-Warshall distance of **all voxels to all voxels**
Set the entrance voxels based on the entrance locations
For each non-entrance voxel:
    Find the **closest entrance voxel**
    Link the distance to that entrance to that voxel
    Convert the distance values into **values between 0 and 1**
Construct the entrance lattice
Interpolate the entrance lattice

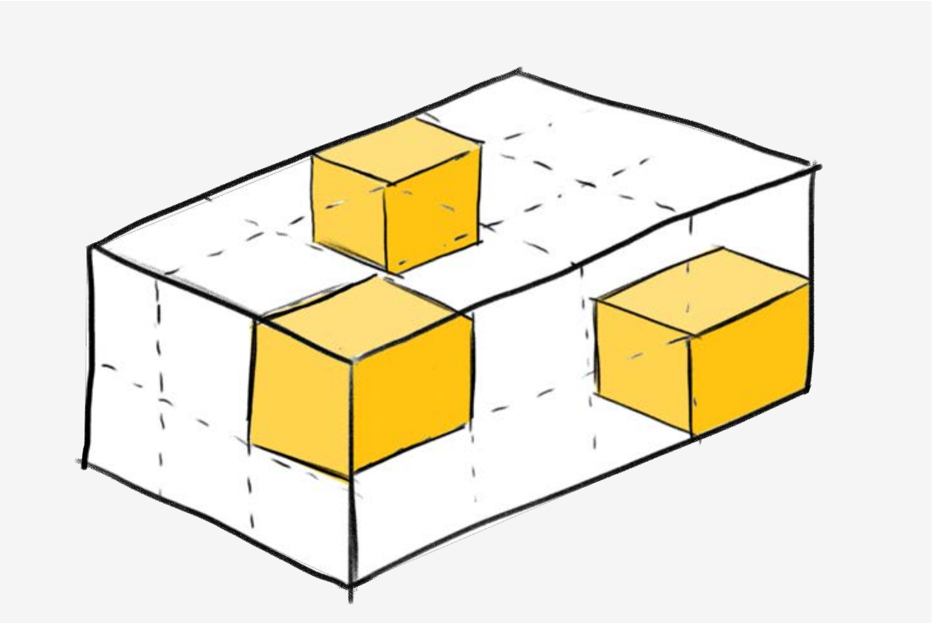


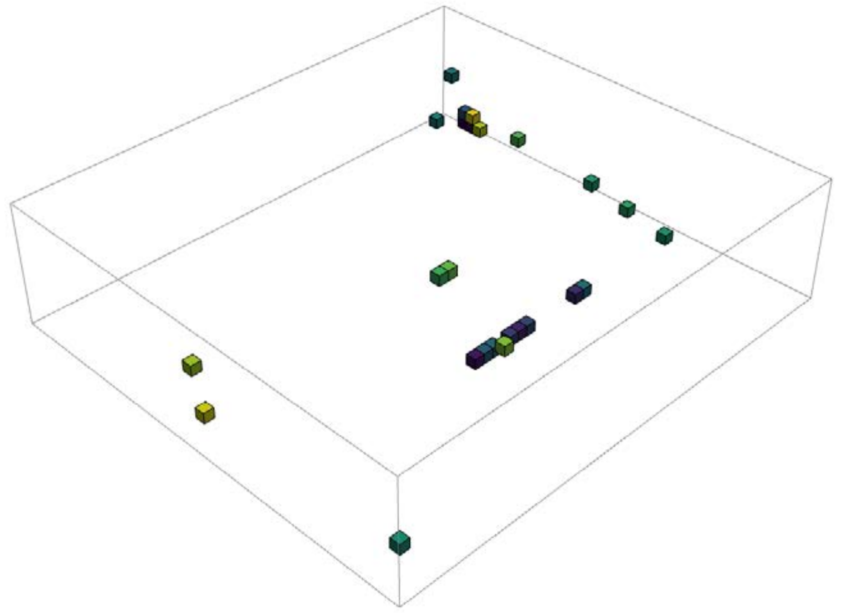
Massing

w2_solar_envelope > data > 324 > growth_output.csv

```
minbound,shape,unit
-42.12,49,3.24
-116.64,44,3.24
-3.24,12,3.24

IX,IY,IZ,value
0,0,0,-1
0,0,1,-1
0,0,2,-1
0,0,3,-1
0,0,4,-1
0,0,5,-1
0,0,6,-1
0,0,7,-1
0,0,8,-1
0,0,9,-1
0,0,10,-1
0,0,11,-1
0,1,0,-1
0,1,1,-1
0,1,2,-1
0,1,3,-1
0,1,4,-1
0,1,5,-1
0,1,6,-1
0,1,7,-1
0,1,8,-1
0,1,9,-1
0,1,10,-1
0,1,11,-1
0,2,0,-1
0,2,1,-1
0,2,2,-1
0,2,3,-1
```

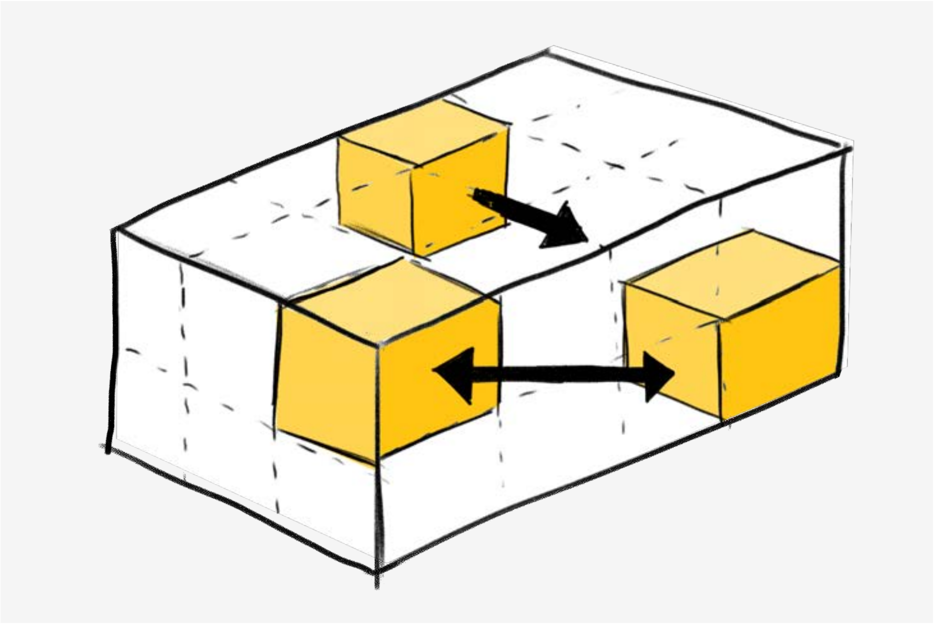
MCDA Seed Allocation

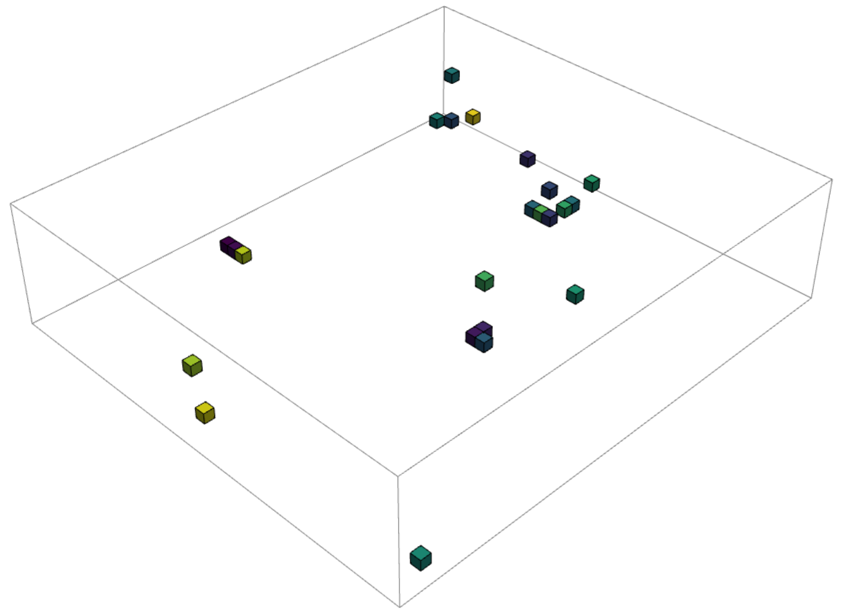




1 - Initial location

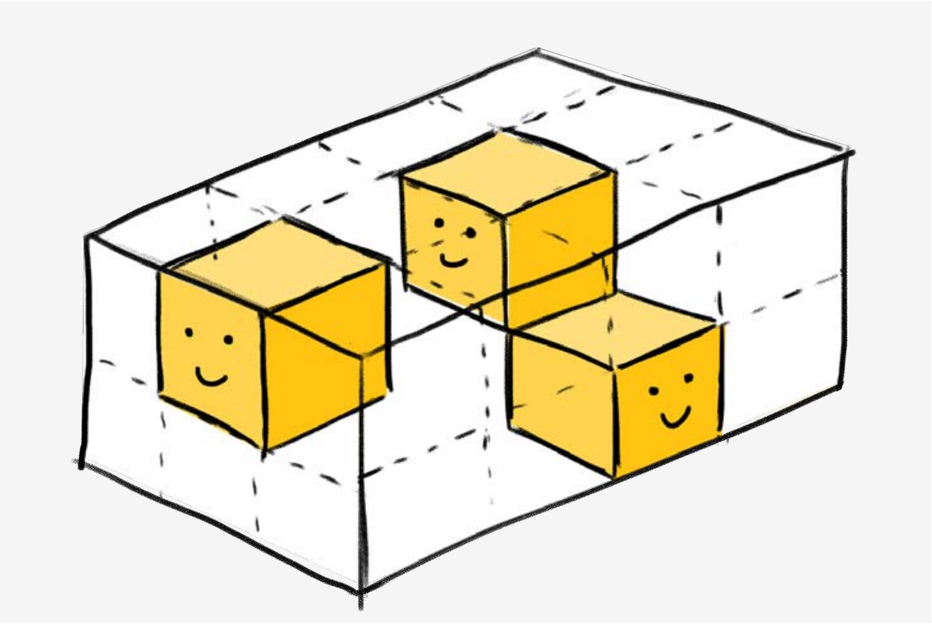
The location of the seed agents is calculated by looking at the **static environmental data**: Entrance access, street noise, sky view factor etc.

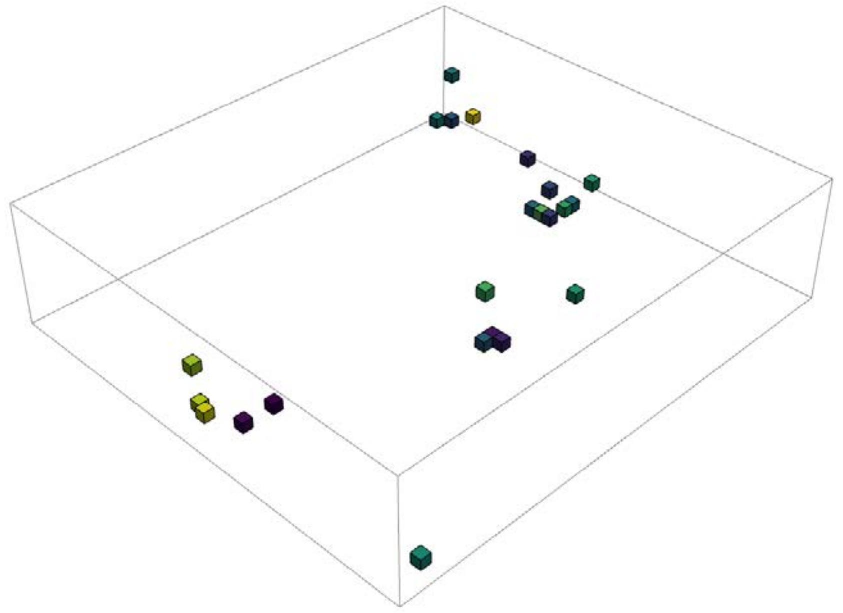




2 - Attraction

The different seed agents are attracted to each other, based on the **connectivity matrix**. They 'walk' around, until they have reached an ideal location based on internal attraction and external data.





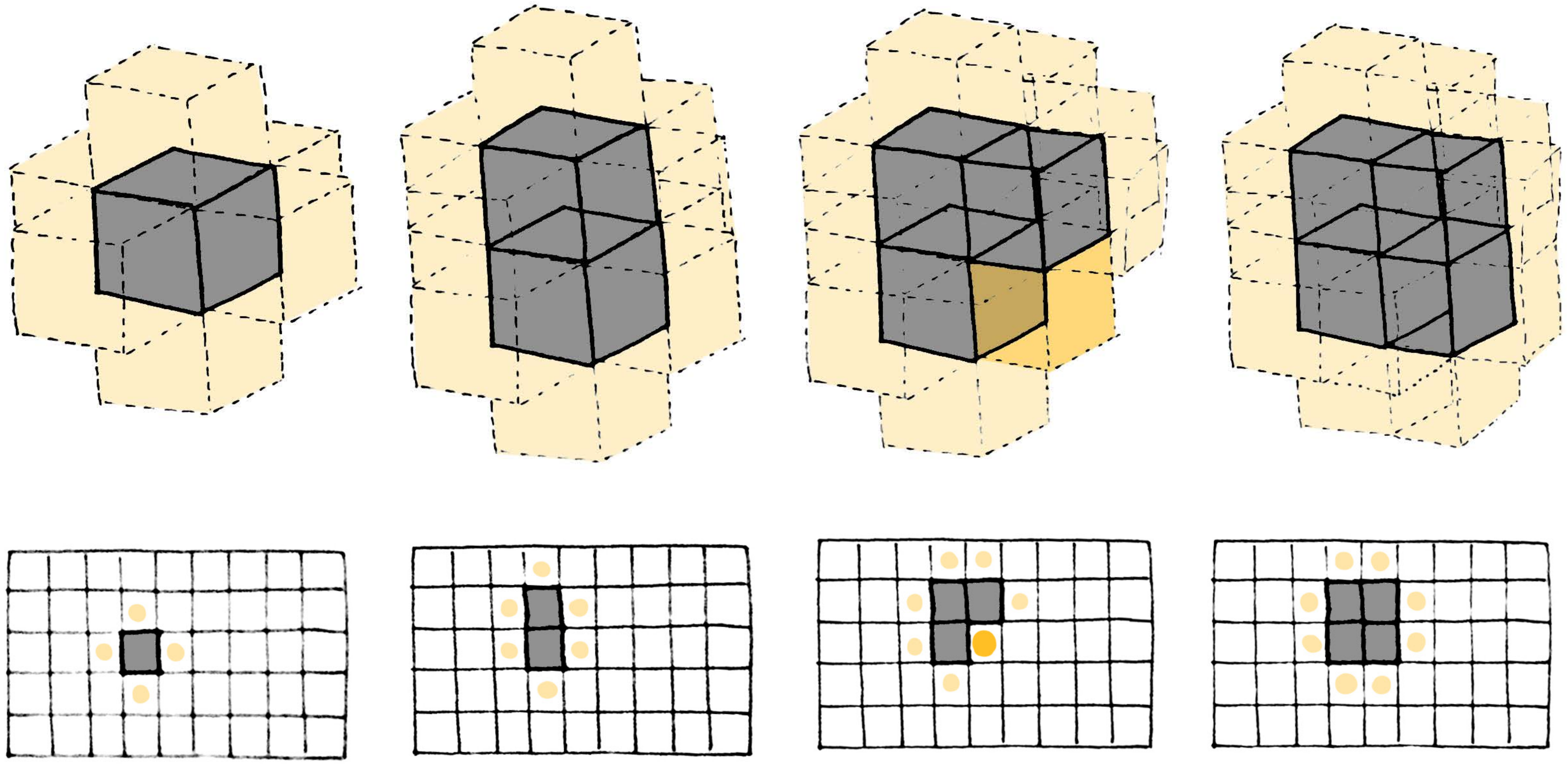
3 - Final Location

The seed agents have reached an equilibrium.

Input:	static env-data, pref and connectivity matrix
Output:	seed agent positions

```
def select-neighbours:
    circumvent the encountered bug

def distance-lattice:
    calculate the euclidian distance from the seed agent to
    every voxel

for each agent:
    for each voxel:
        check if voxel is available:
            calculate 'grade' (based on
                         env-data and agent
                         preference)
            append best voxel to agent list

while t < threshold:
    for each agent:
        calculate a closeness lattice to the seed voxel
        select-neighbours:
            check which neighs are available:
                grade those neighs on dist and env-data
                append best voxel to agent list
                remove previous voxel of this agent
    t += 1
```

Spatial behaviours: Squareness



Input:	Voxelized envelope, squareness preferences
Output:	Impacts the growing algorithm

For each agent (during the growth process):
Find the **free neighbours** based on the chosen stencil,
Check if the agent has free neighbouring voxels
Check if those neighbours are also **neighbours of the previous agent**

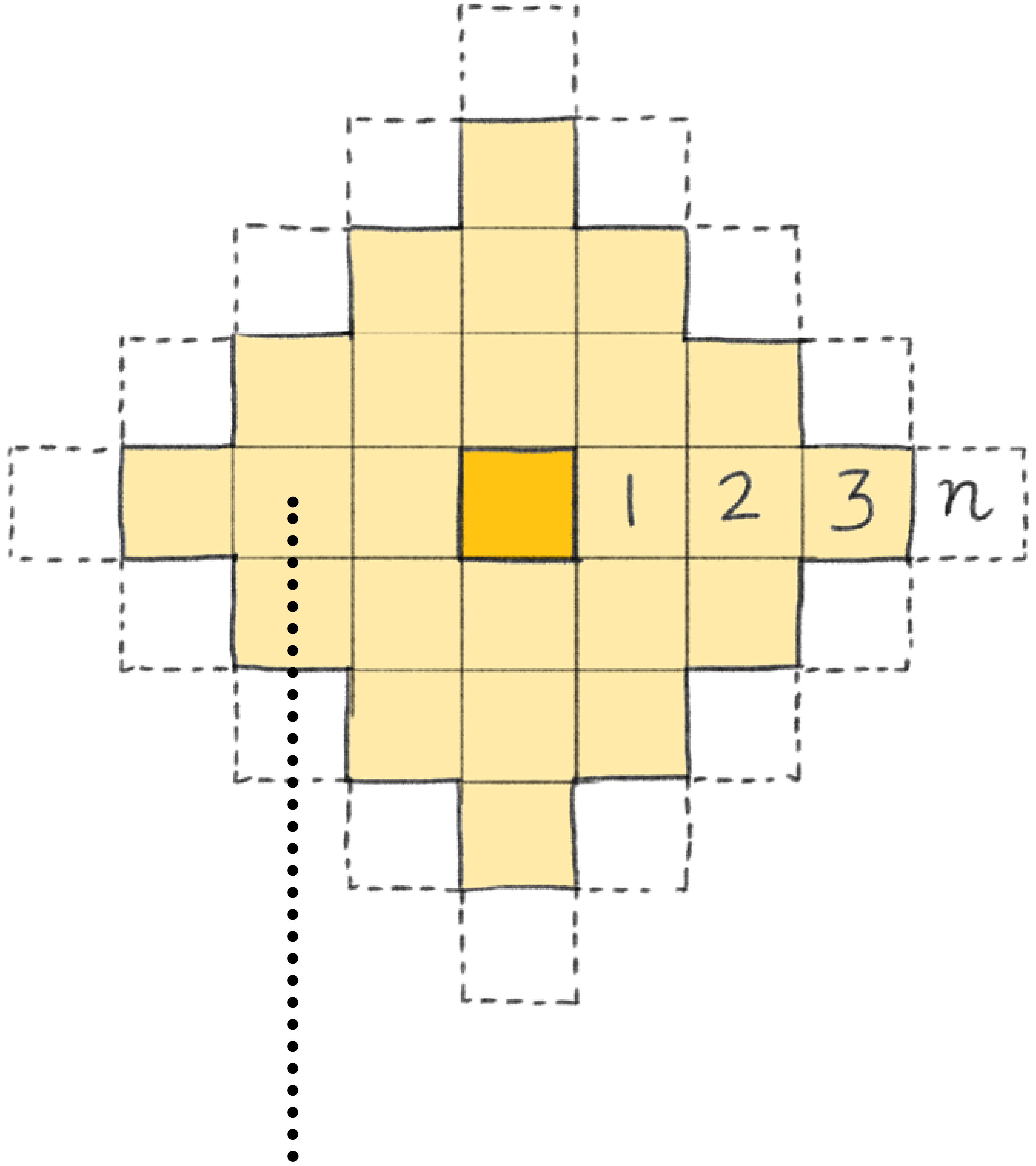
For those voxels that were neighbours to the previous agent, **increase the voxel value** (the more often a voxel has been a neighbour of an agent, the more the voxel value increases)

Select the neighbour with the **highest voxel value**
Set the selected neighbour as **unavailable**
The selected neighbour is now the **new agent**.

Ensure agents grow into their desired shape

If there is the need for a space to be **more rectangular**, instead of free-form, the squareness algorithm can be used

Spatial behaviours: Distance between functions



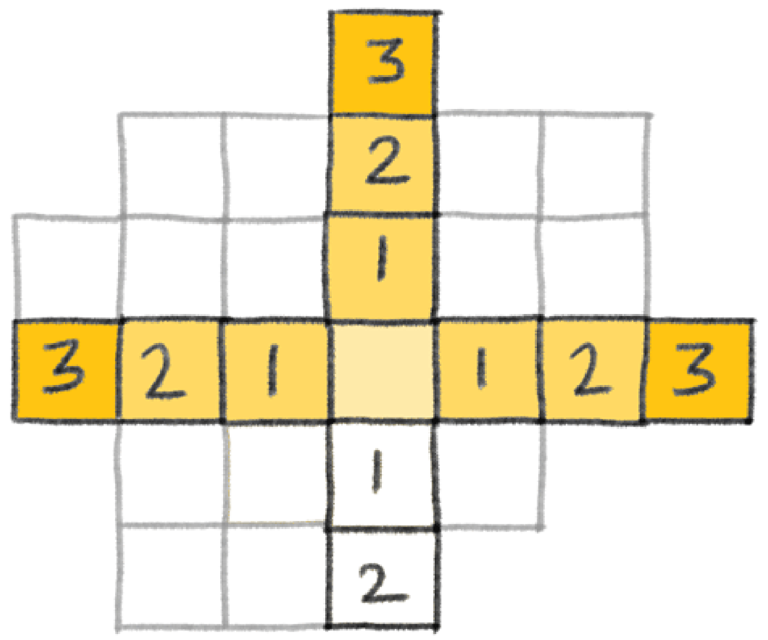
Only accessible for this specific function

Input: location of new agent

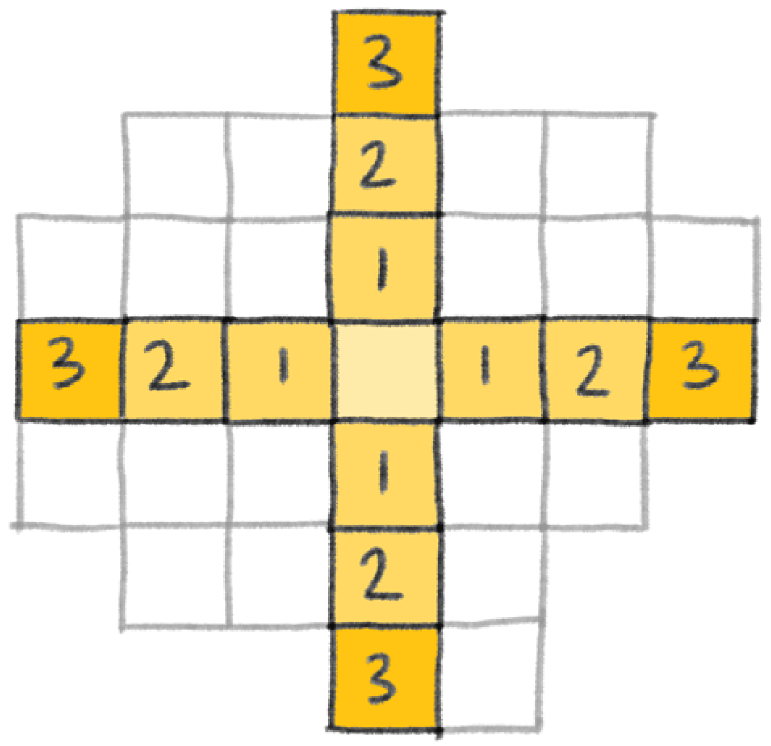
Output: keep-distance-lattice

```
field = [list of neighbours in a given radius]
for i in field:
    keep-distance-lattice[ loc + i ] = agent-ID

###later on, when determining neighbours

if keep-distance[neighbour-location] == agent-ID or -1:
    neighbours . append(neighbours-location)
```

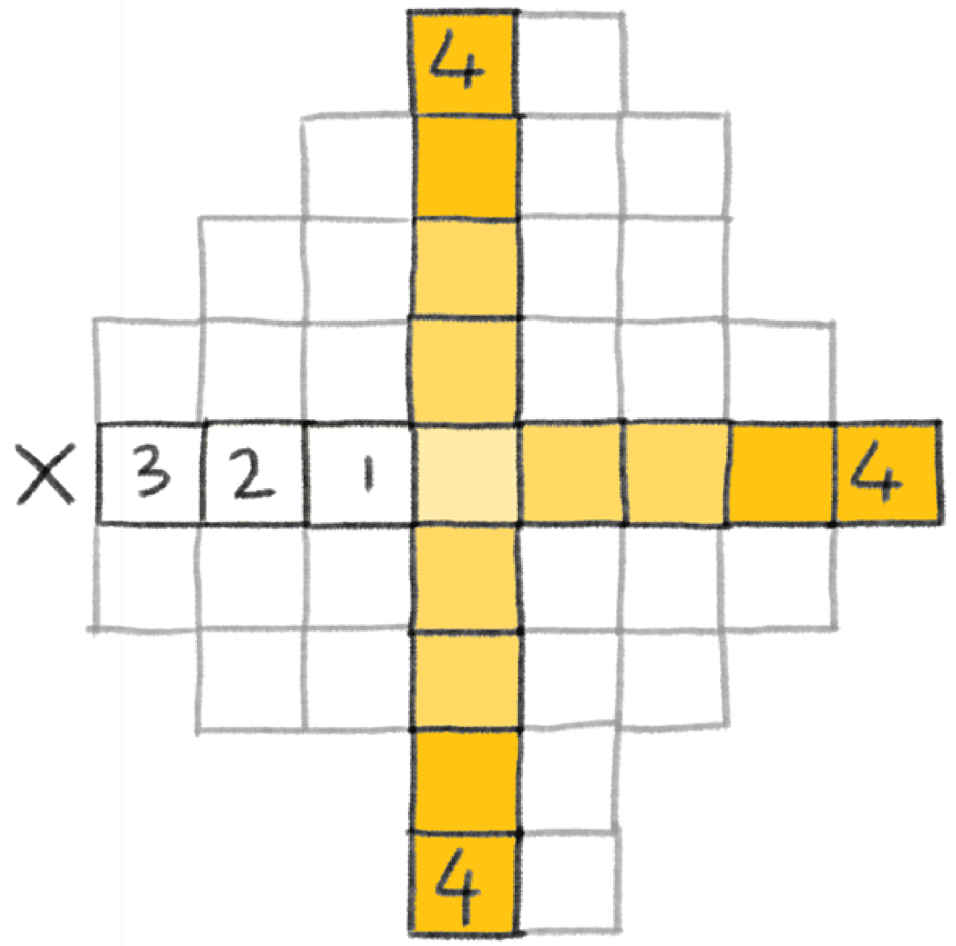
Spatial behaviours:
Maximum building depth



1 - **Three directions** are filled



2 - All **four directions** are filled



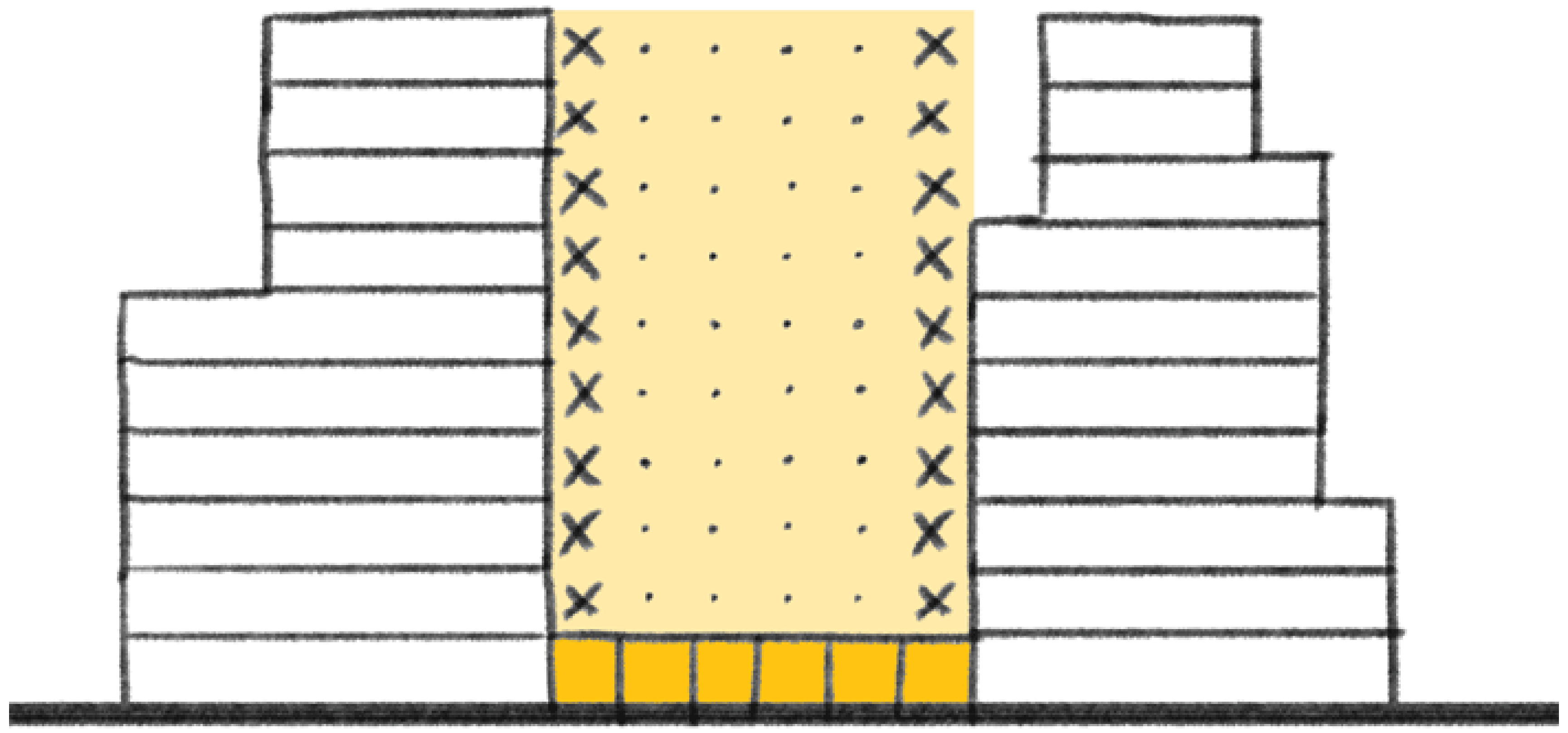
3 - If there is one direction with only three voxels remaining, the fourth voxel is made **unavailable**

Input: agent locations

Output: updates to avail-lattice

for **each voxel-location** of agent:
    check if all voxels in given distance are **occupied in x and y axis**:
        check how many axes **don't** have a **n+1th voxel**:
            if amount is 1:
                make remaining n+1th voxel **unavailable**

Spatial behaviours: Roof light



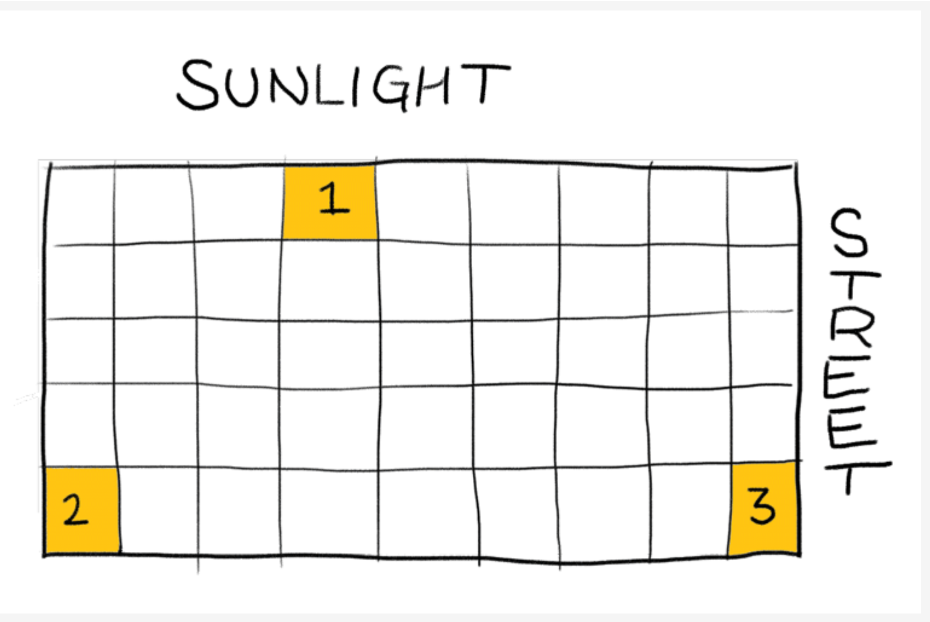
Input: New agent locations

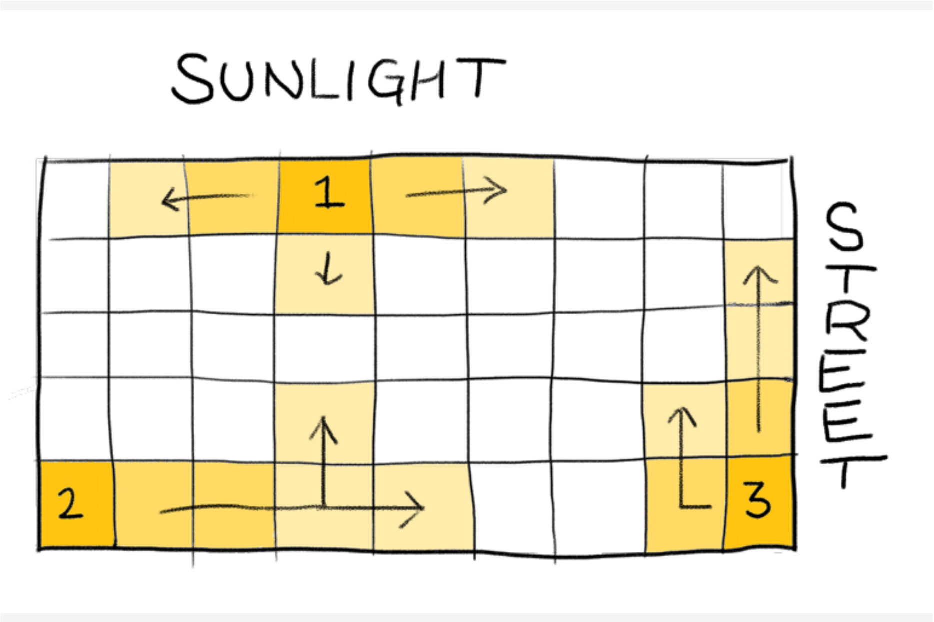
Output: updates to avail-lattice

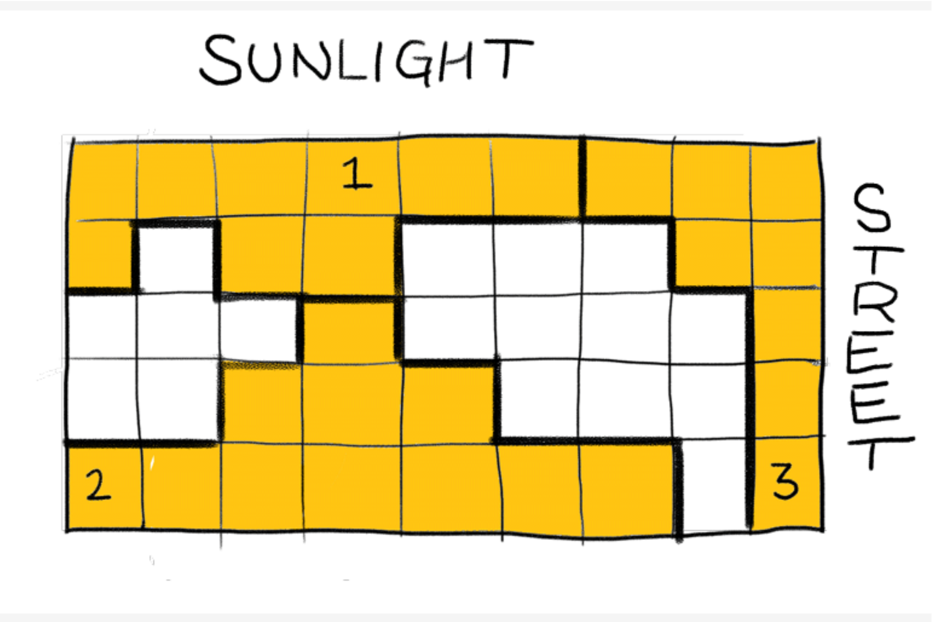
roof-light = [list of functions that do not want voxels above them]

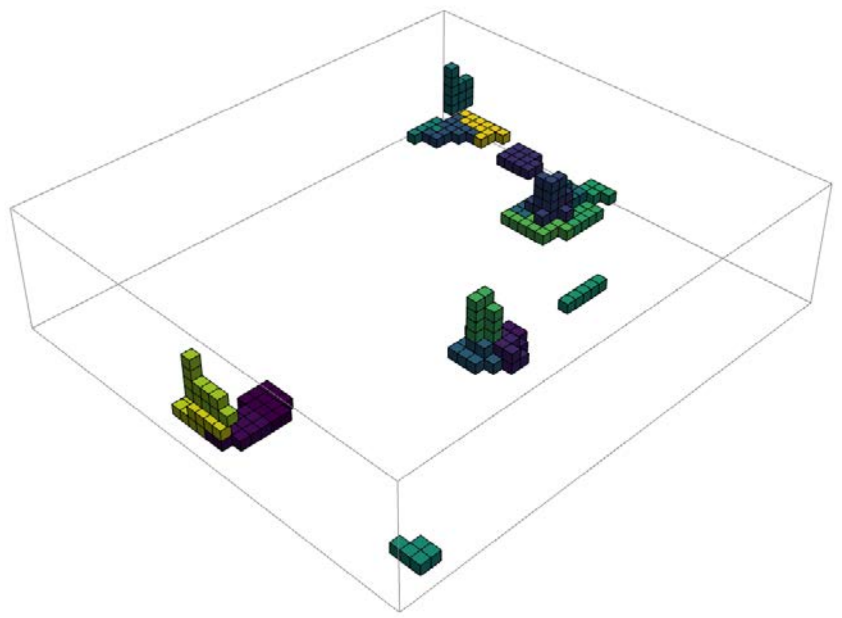
```
if agent-id in roof-light:
    avail-lattice[ neigh-3d-loc[0],  neigh-3d-loc[1]] , 2: ]  =  -1
```

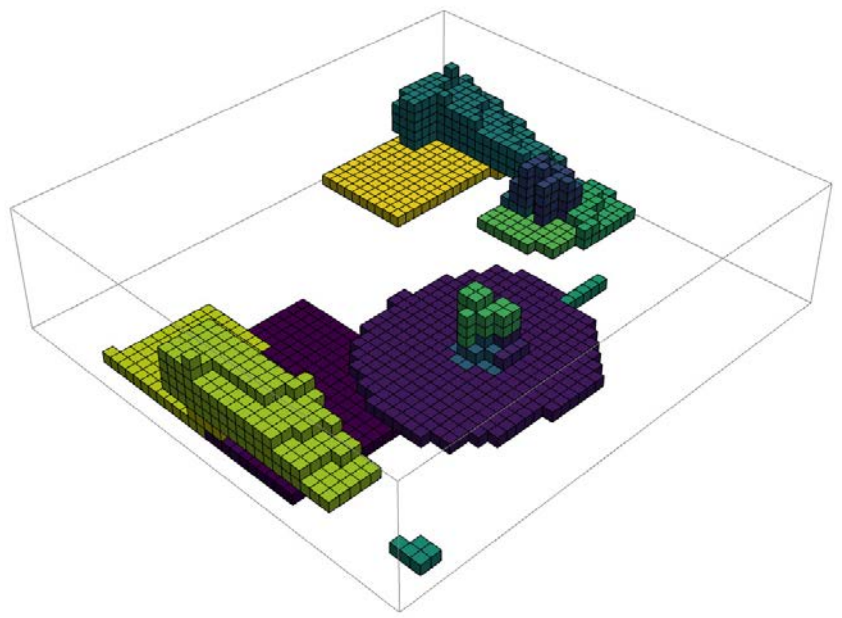
MCDA Growth algorithm

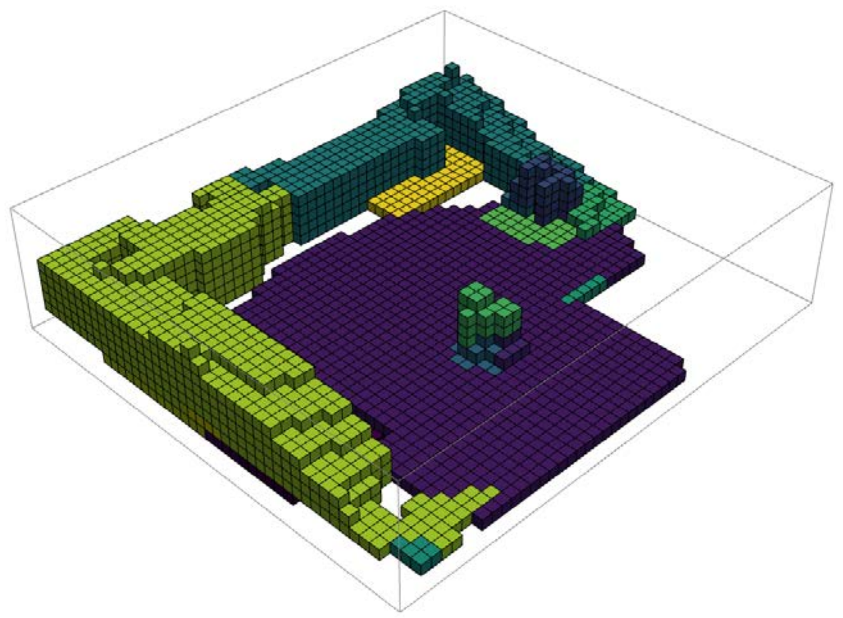












1 - Starting the growth

All the agent seeds are evaluated and their best neighbour is chosen based on the static **env-data** and **closeness** to other agents. This is done with the connectivity and preference matrix.

2 - Growing

For each agent, the algorithm evaluates **every voxel** and calculates all the possible neighbors. The best one is chosen.

3 - Finished growth

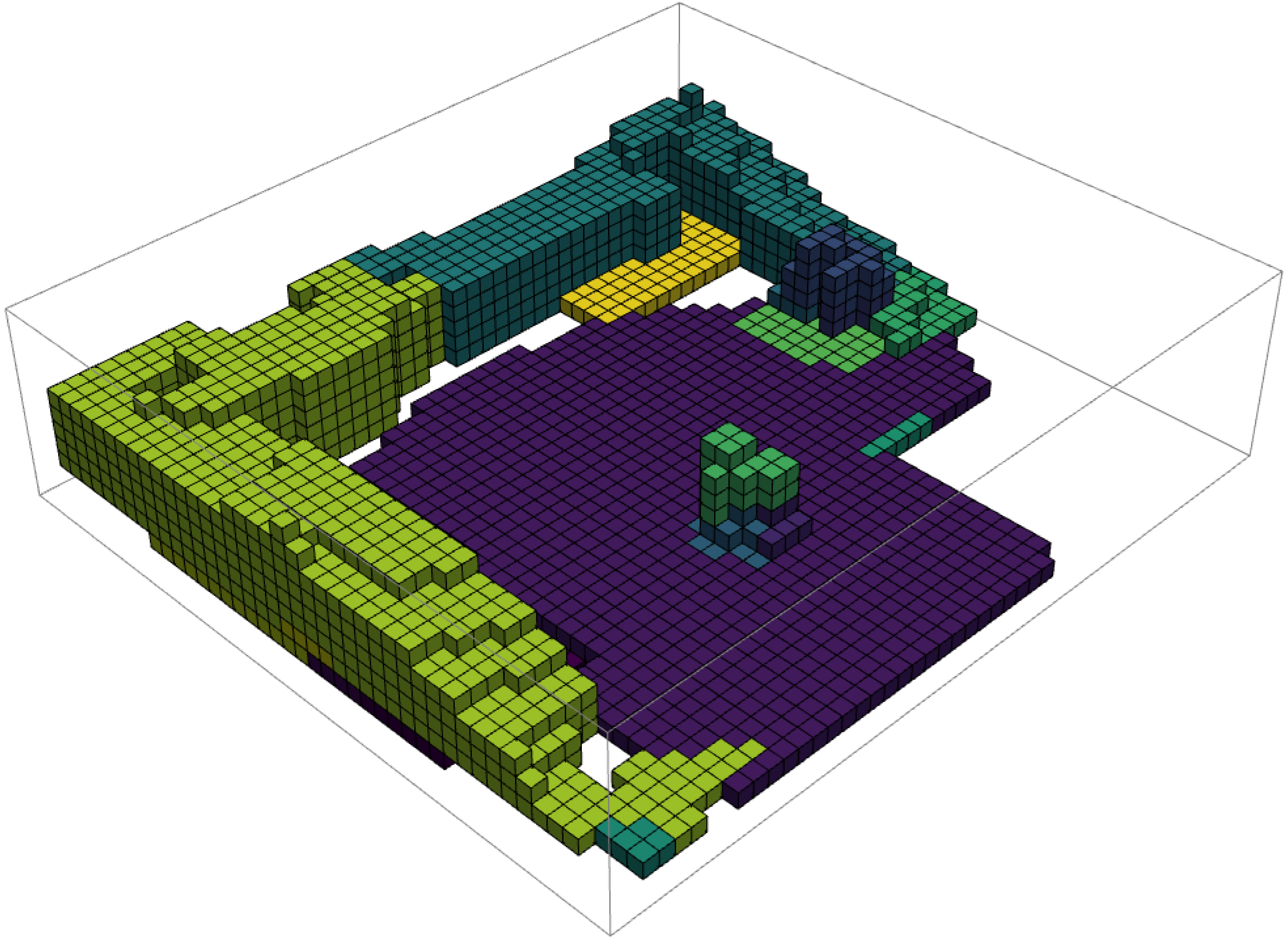
The max. number of voxels per agent has been reached, the division of the spaces has ended.

Input: static env-data, pref and connectivity matrix

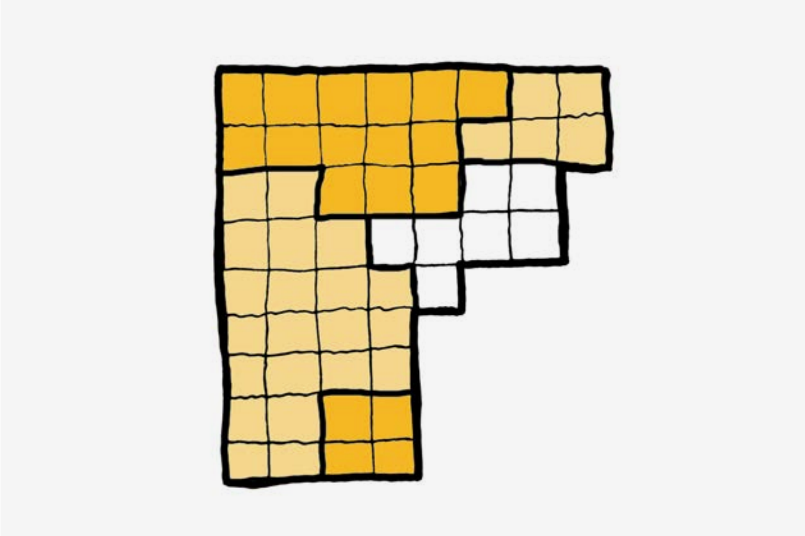
Output: Occupation lattice

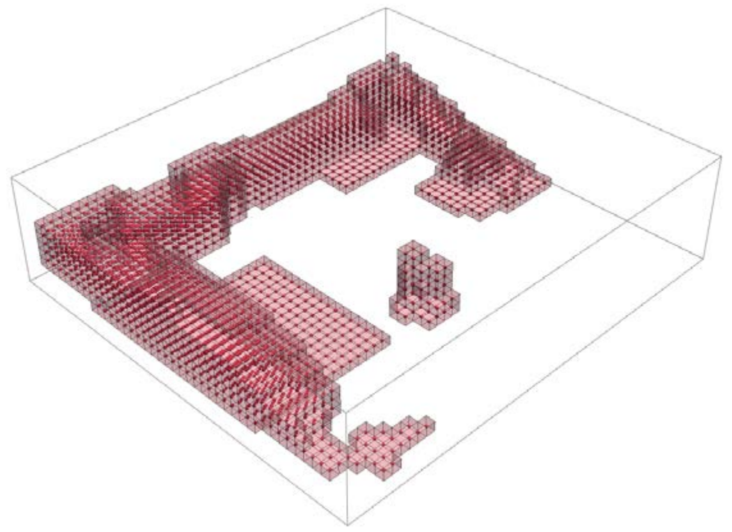
```
while t < threshold:
    for each agent:
        check if max. amount of voxels has been reached
            for each agent location:
                find neighs:
                check which neighs are available:
                    grade those neighs on dist and env-data
                    append best voxel to agent list
    t += 1
```

Final Growth



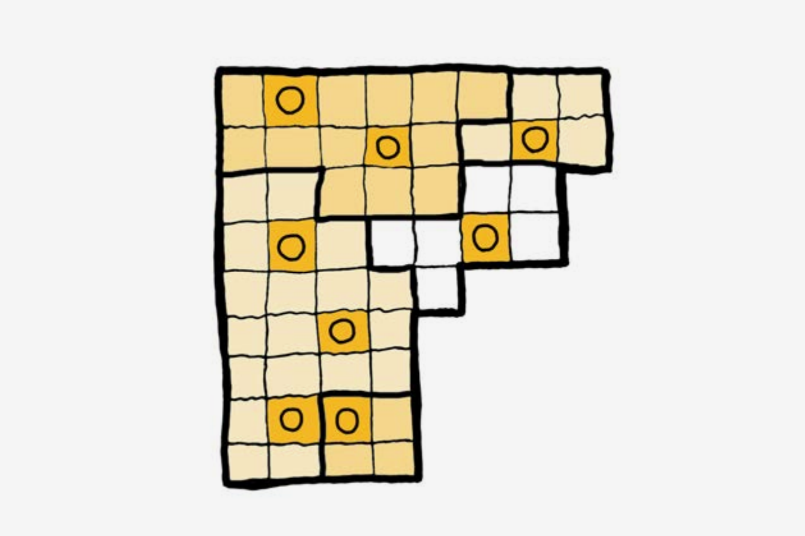
Shafts and corridors growth

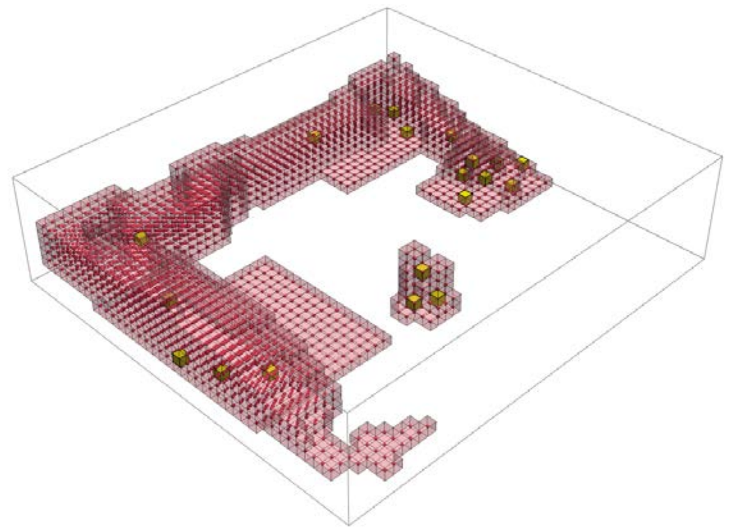




1 - Selecting voxels to evaluate

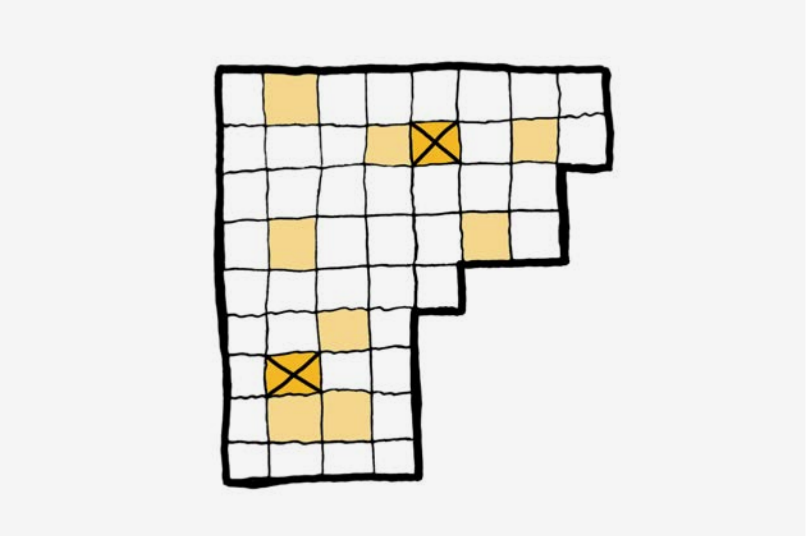
Not all the voxels need a shaft to be placed. The garden for instance would be strange to take into account.

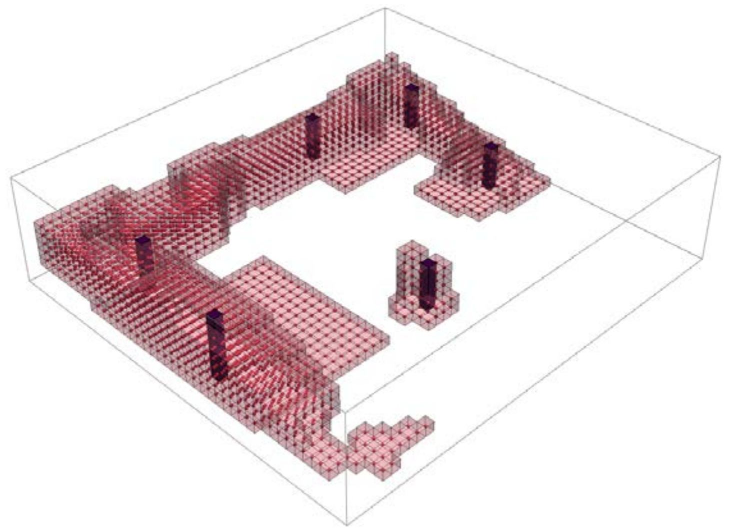




2 - Finding mean voxels

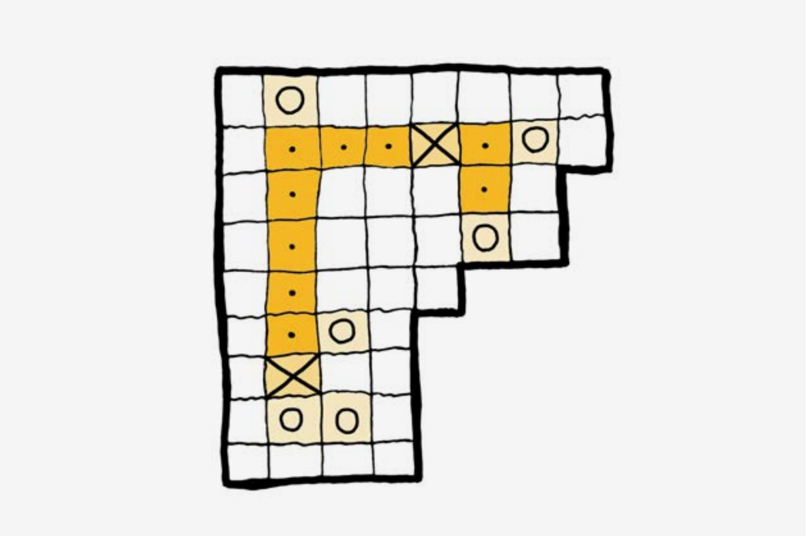
For every function in de occupation lattice, a certain amount of voxels are set, based on the size of each function. Each function has at least 1 mean voxel so that later on corridors can grow and acces all functions.

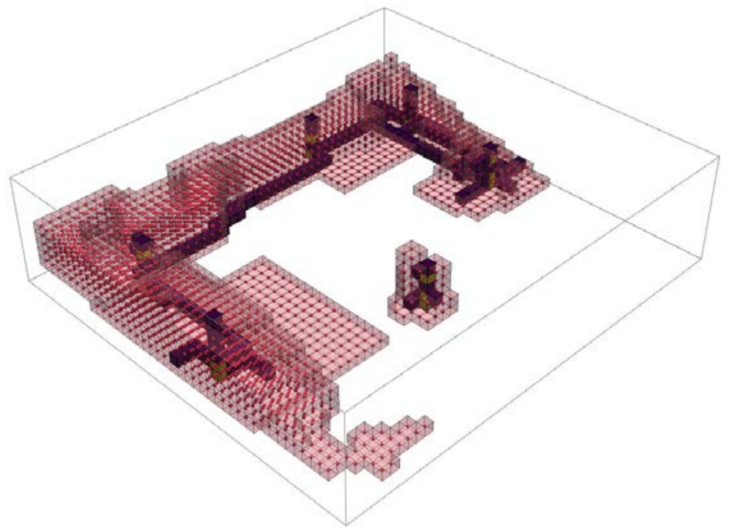




3 - Mean voxels again

From the previous mean voxels, new mean voxels are calculated, that will become the shafts inside the new lattice.





4 - Corridor growth

Each shaft is connected on the ground floor to the other shafts. Also second corridors grow from each mean voxel to their closest shaft.

Input: Occupation lattice

Output: Shafts and corridors lattice

Make a boolean lattice for all important voxels from the occupation lattice

For each agent:
    calculate a number of mean voxels based on the agents occupation

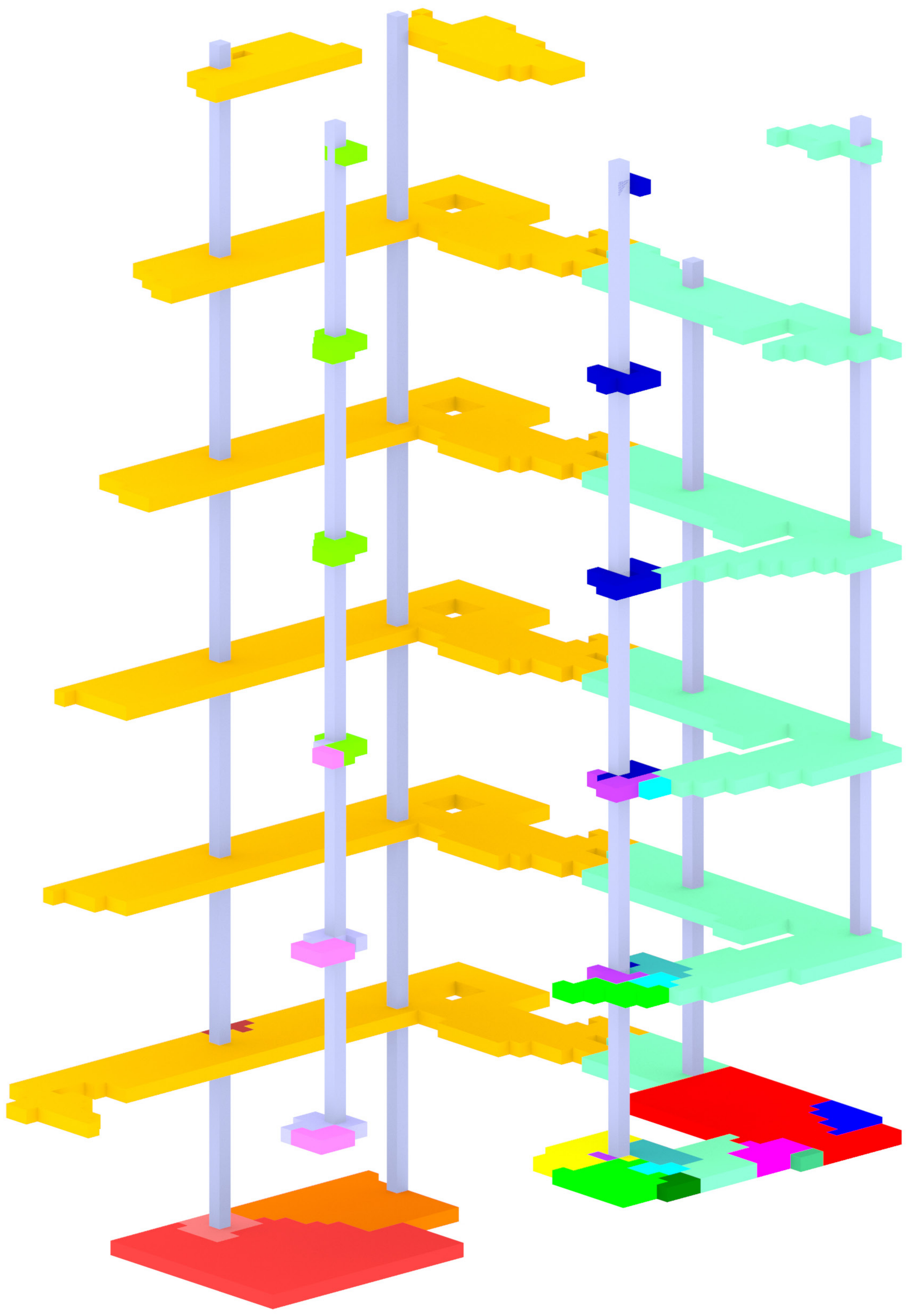
For each mean voxel:
    calculate 6 new mean voxels for shaft placement

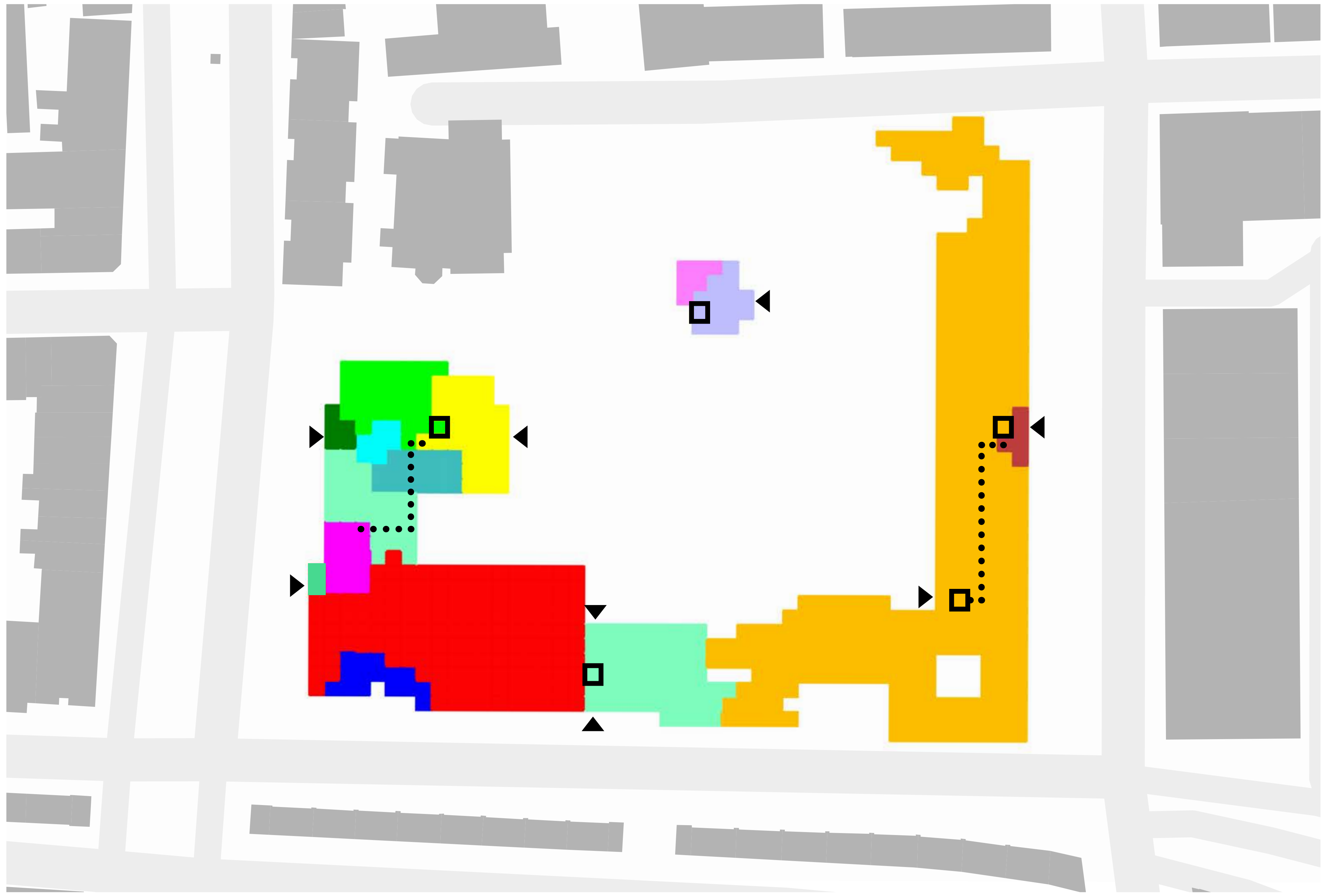
For each mean voxel:
    calculate the closest distance to a shaft
    set this path as a corridor

For each shaft:
    calculate the 2 closest distances to another shaft on the ground floor
    set these paths as corridors

export the shafts and corridors lattice

Massing

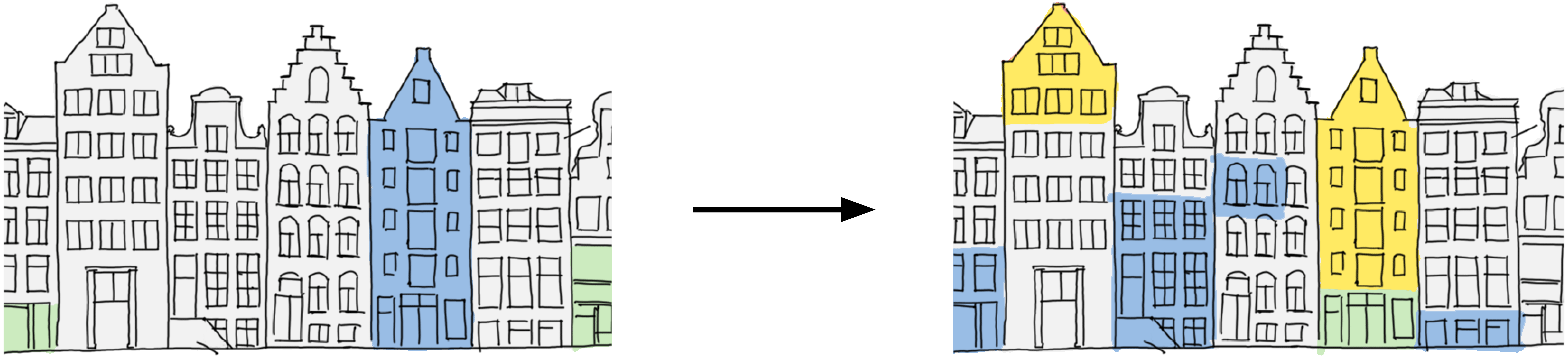




- Shops
- Residents entrance
- Storage
- Apartments
- Hub
- Study space
- Makerspace
- Visitors entrance
- Coffee corner entrance
- Offices
- Gym
- Community center
- Co-cooking
- Restaurant
- Library
- Music room
- Coffee corner
- Workshops
- Bikes parking
- Car parking

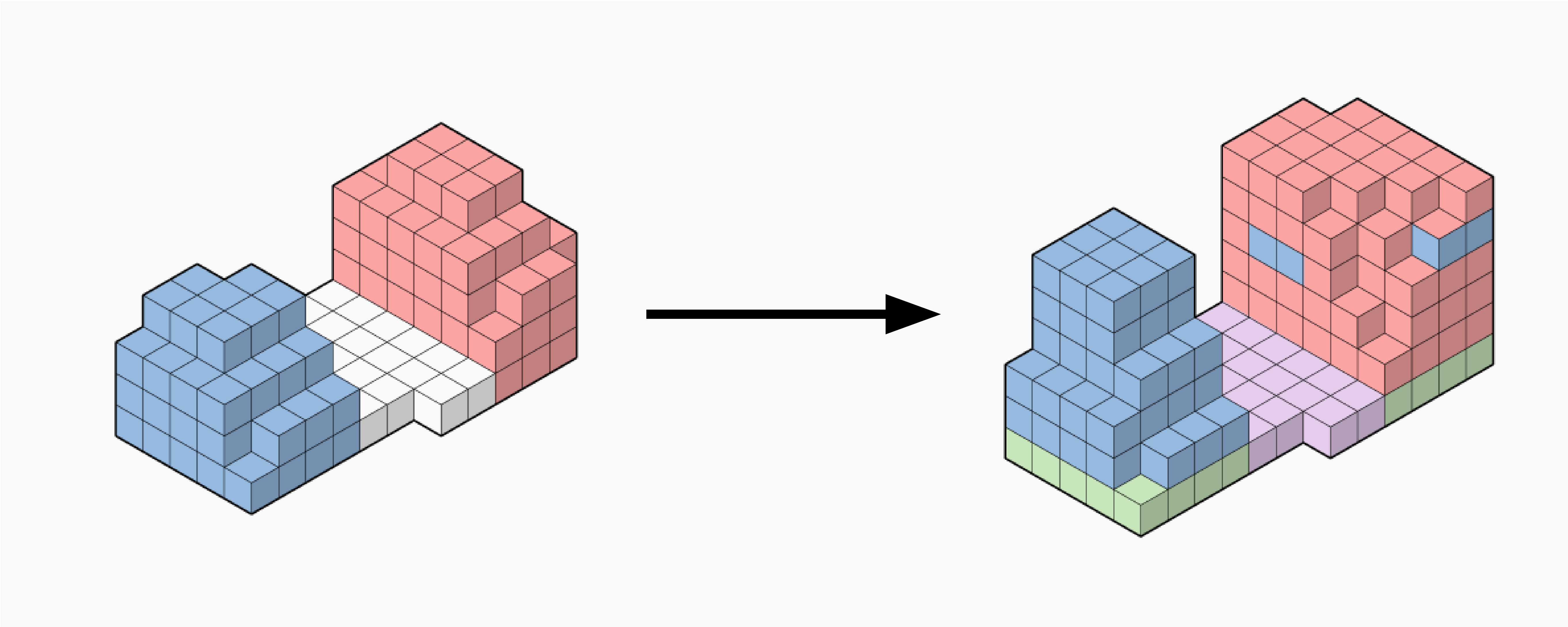
Forming

Flexibility



Less **waste**. Less **emission**. Less **material use**.

Modularity



Add building mass.

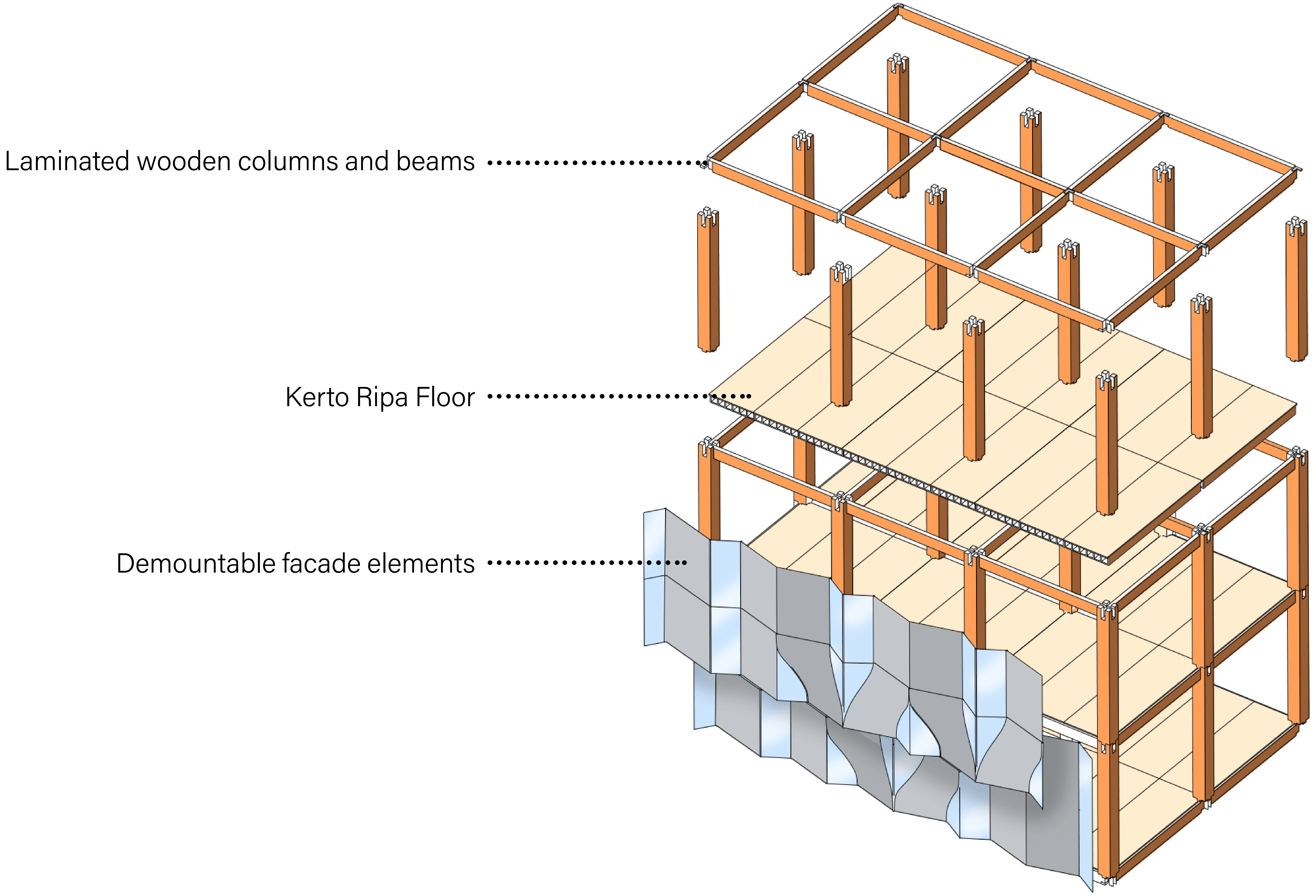
Change function.

Modular system

structure + shafts + infill + facade

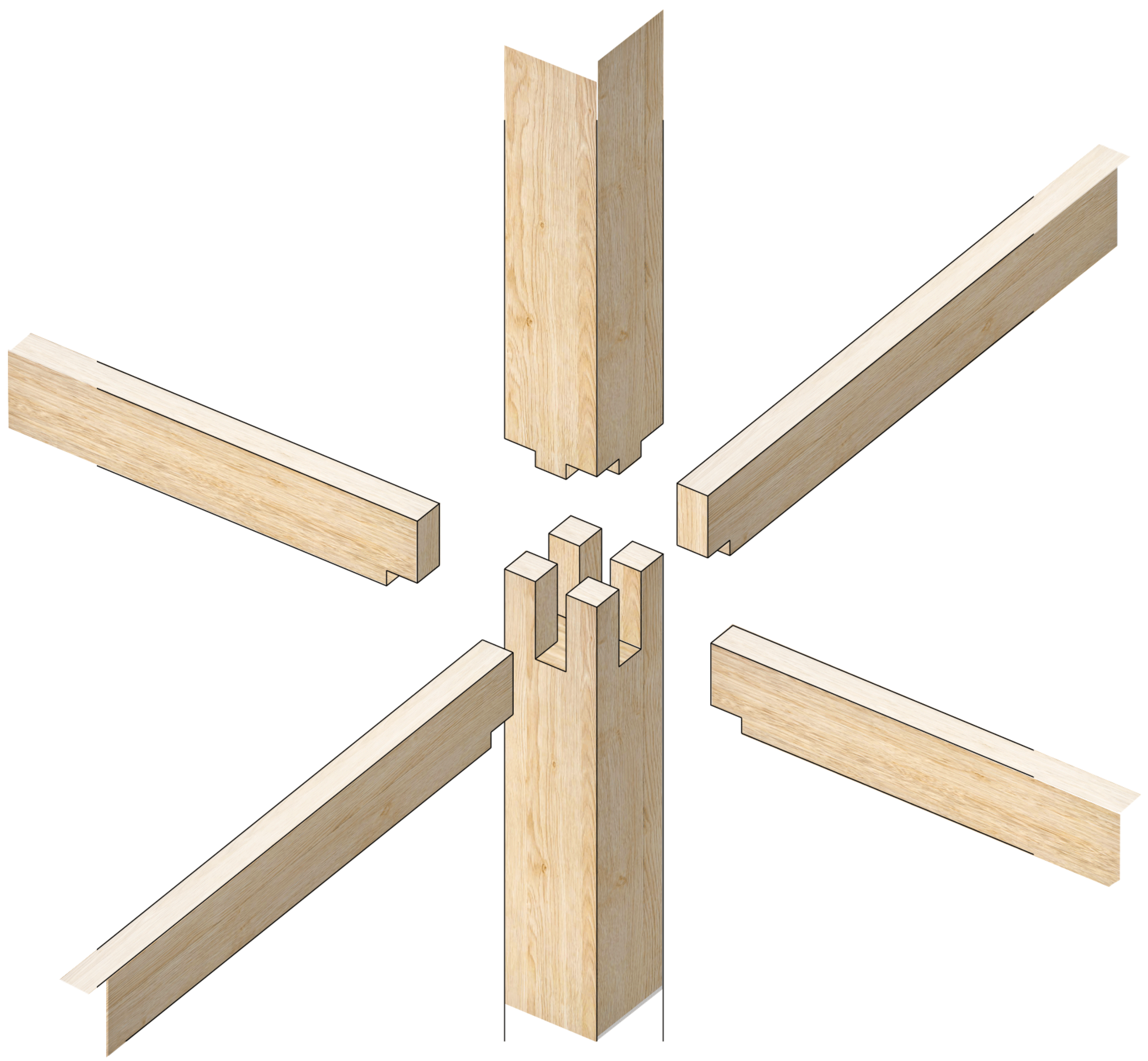
construction | customization

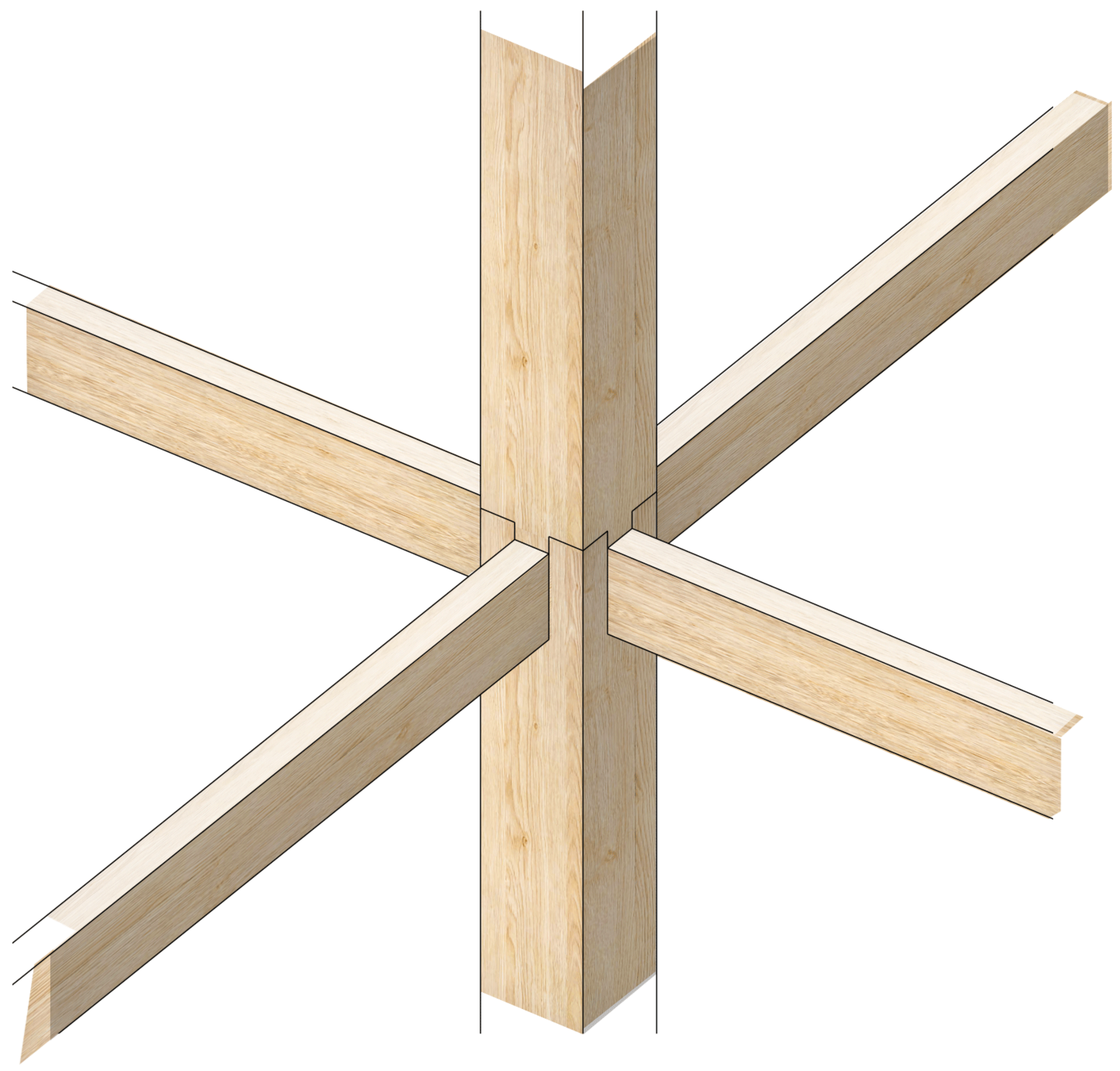
Construction



*Shear forces are countered by constructing the shafts from strong CLT walls

Wooden joint

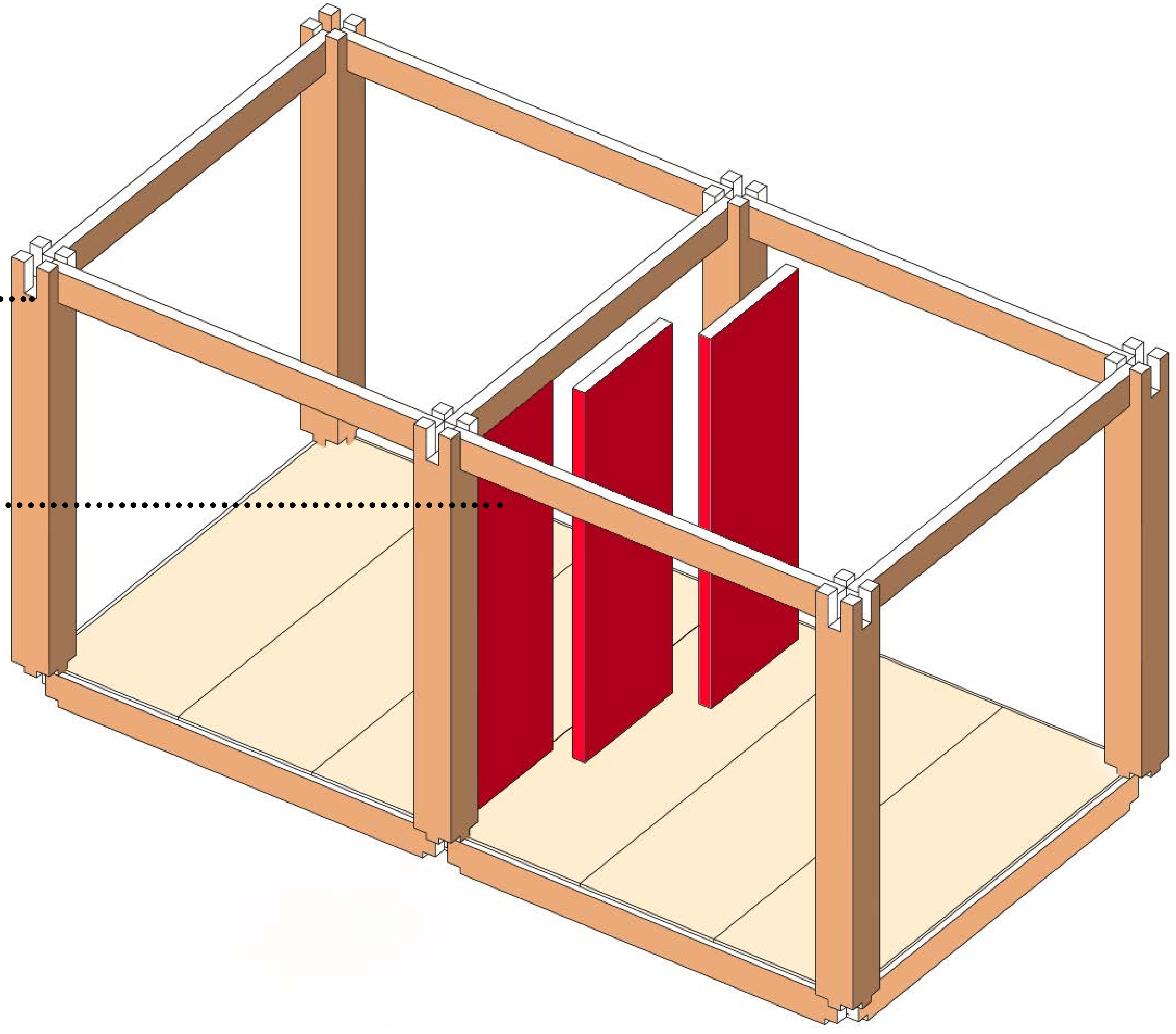




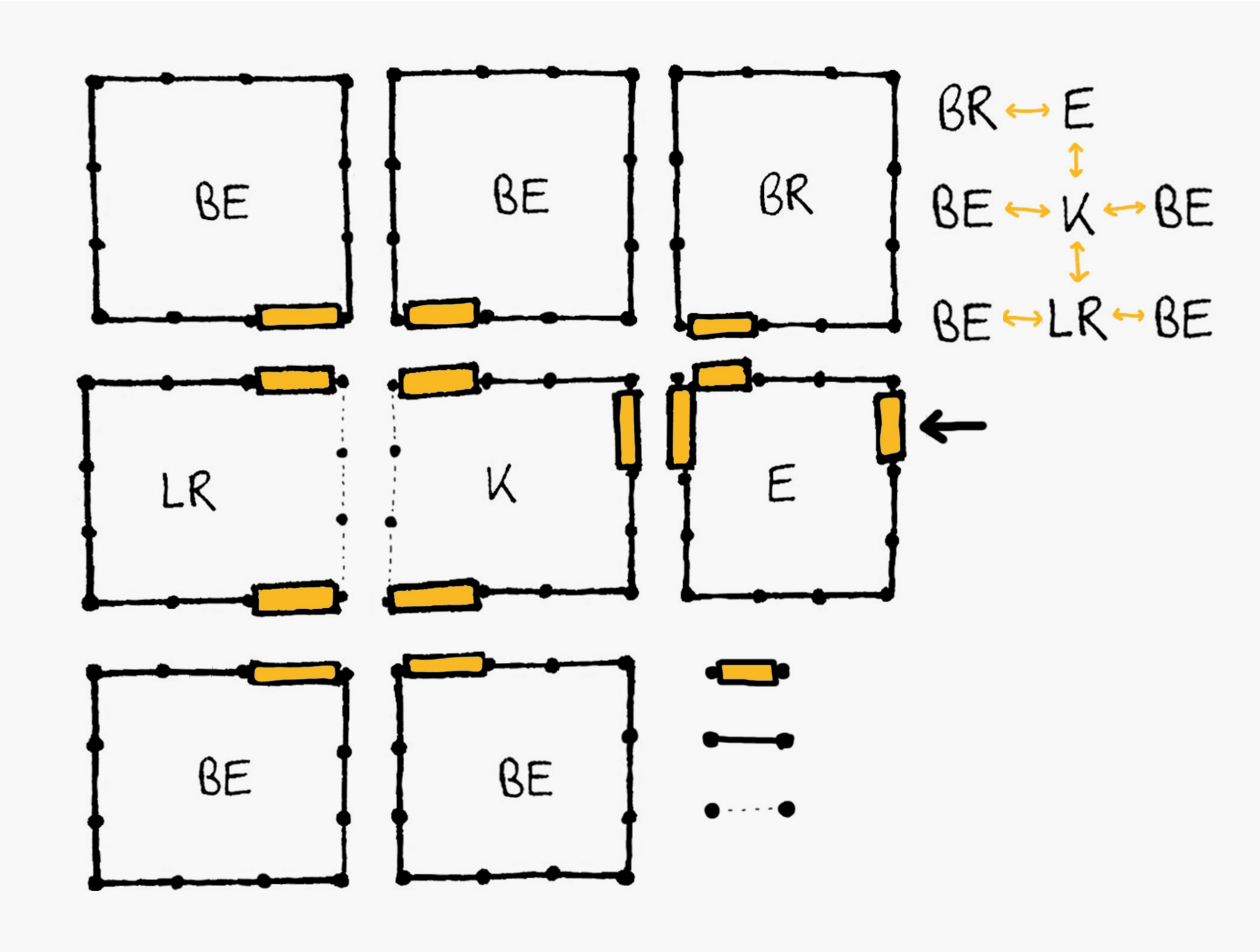
Structure and Infill

The load-bearing structure is constructed first

After that, the partitioning walls are placed



Modular interior tiles



The tiling system

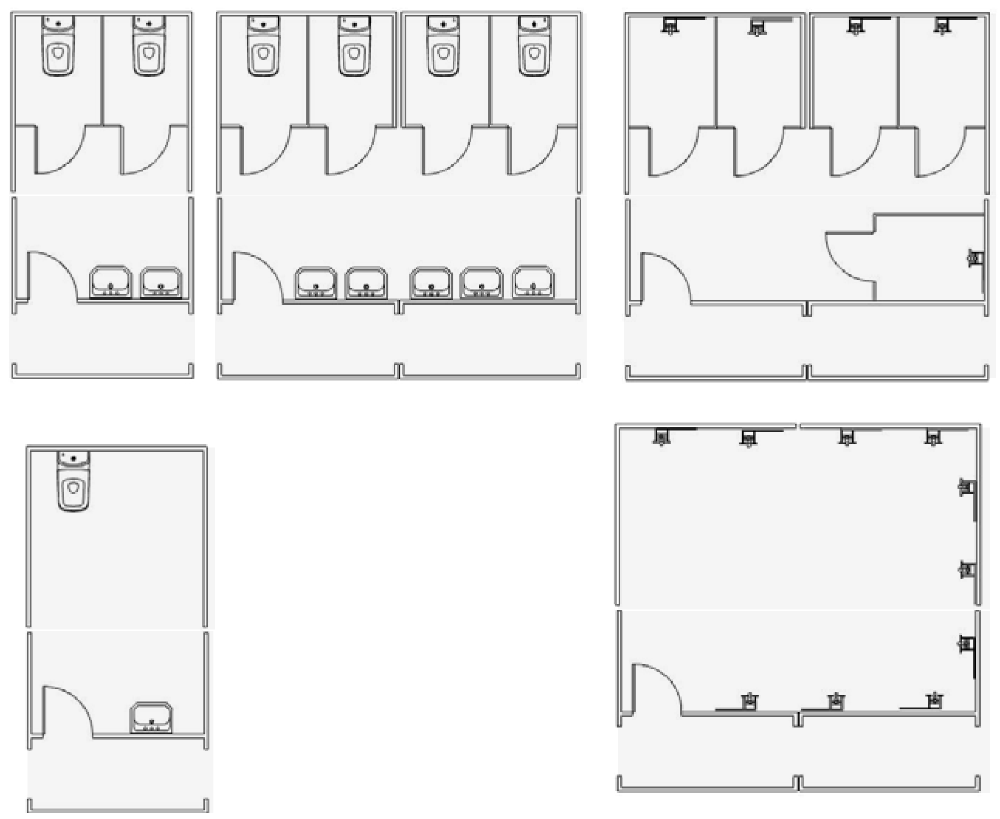
The tiles are created with an underlying system similar to that often seen in tile based board games. The square voxel is subdivided in three parts along each edge. One of these subdivisions is equal to the width of a small corridor or door.

These three parts are then labeled as either a door, wall or open space. By combining different tiles that match the corresponding edge types, different spaces can be created from simple tiles.

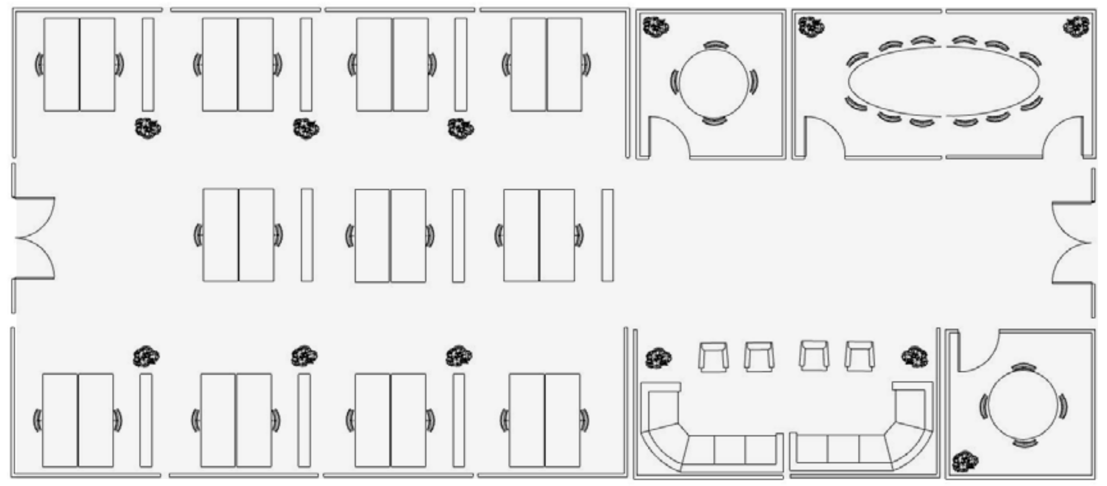
By then also listing the function type of each tile, such as the entrance or kitchen (E & K), limitations and recommendations could be added to the code which tiles can connect to which tiles. Due to time limitations this is something that we have not developed yet, but could be an interesting concept for peers following this course over the following years.

Modularity options

Tiles can be swapped and matched for desired program and area size.

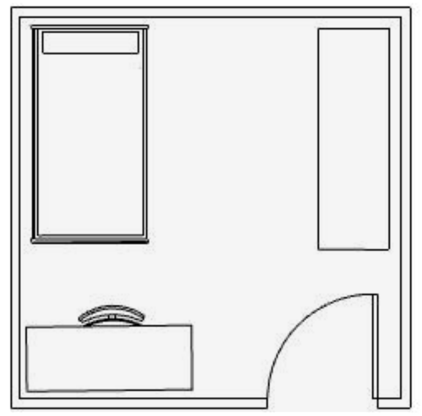


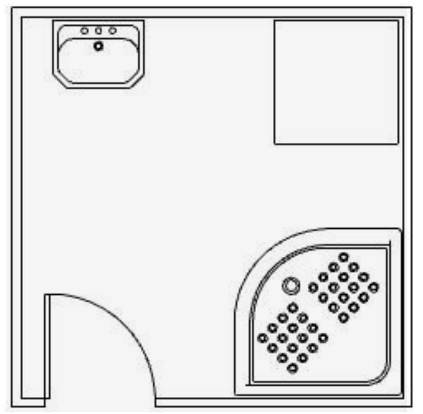
From office bathrooms...

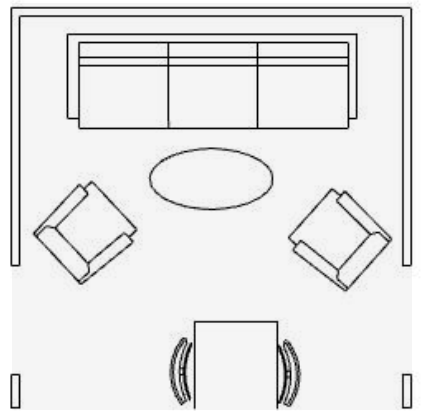


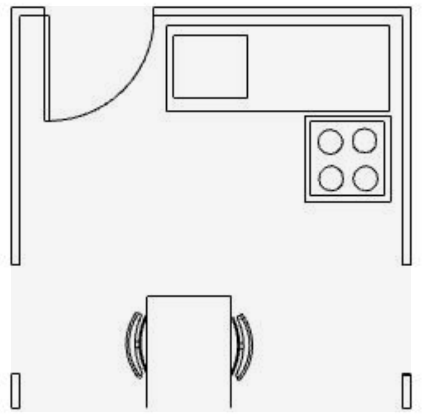
...to large-scale workplaces

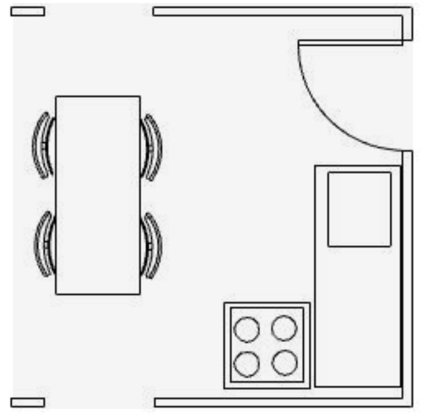
Residential voxel tiles

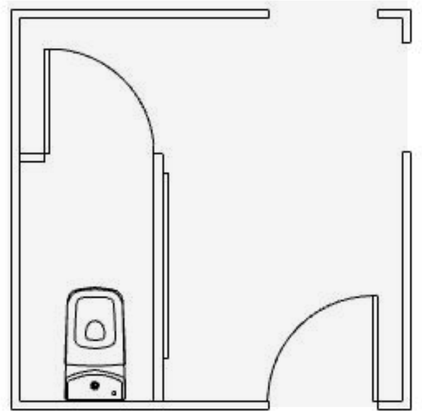

Bedroom

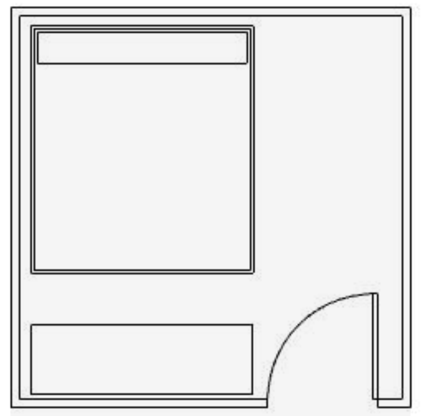

Bathroom

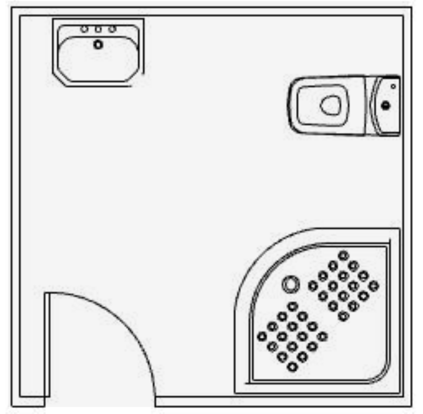

Living room

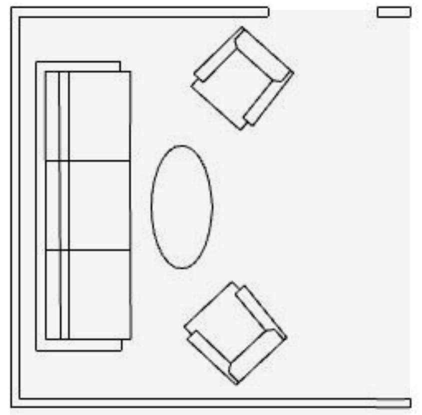

Living room
+ kitchen

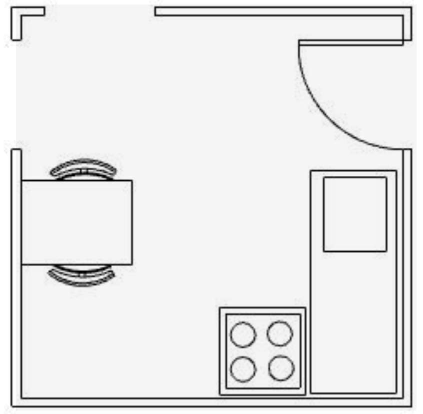

Living room
+ kitchen

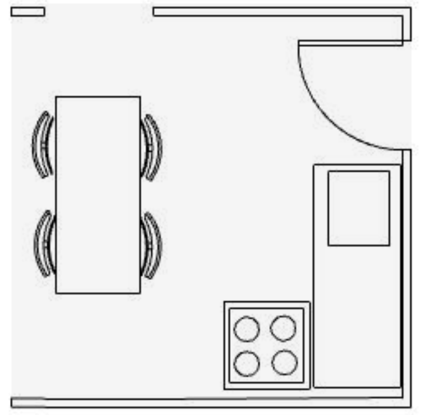

Entrance


Bedroom

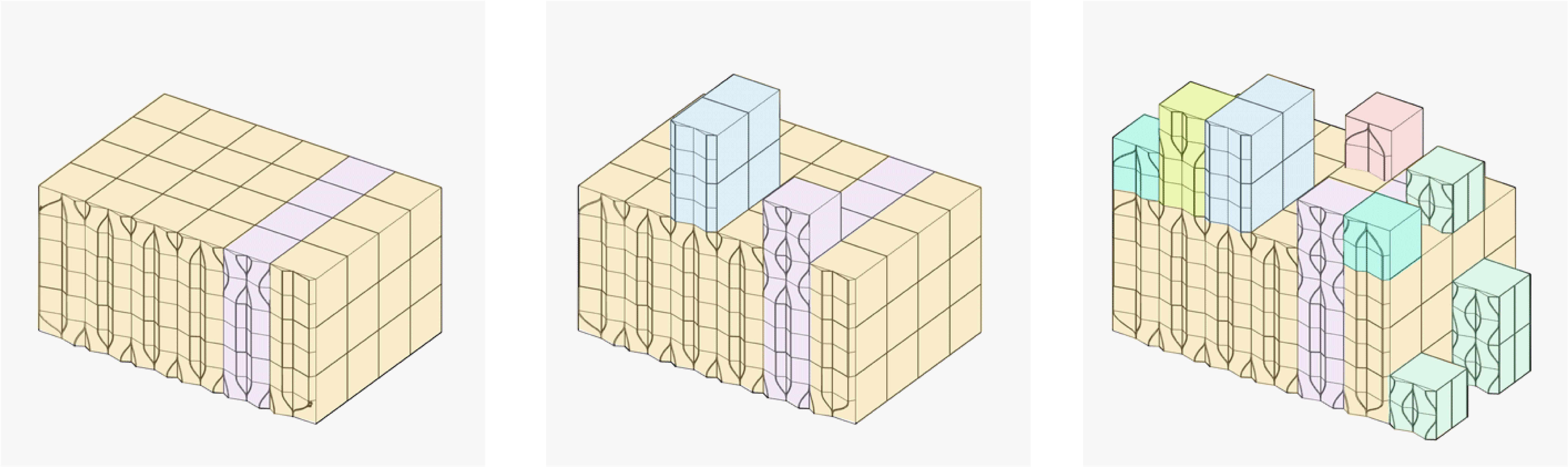

Bathroom


Living room


Kitchen

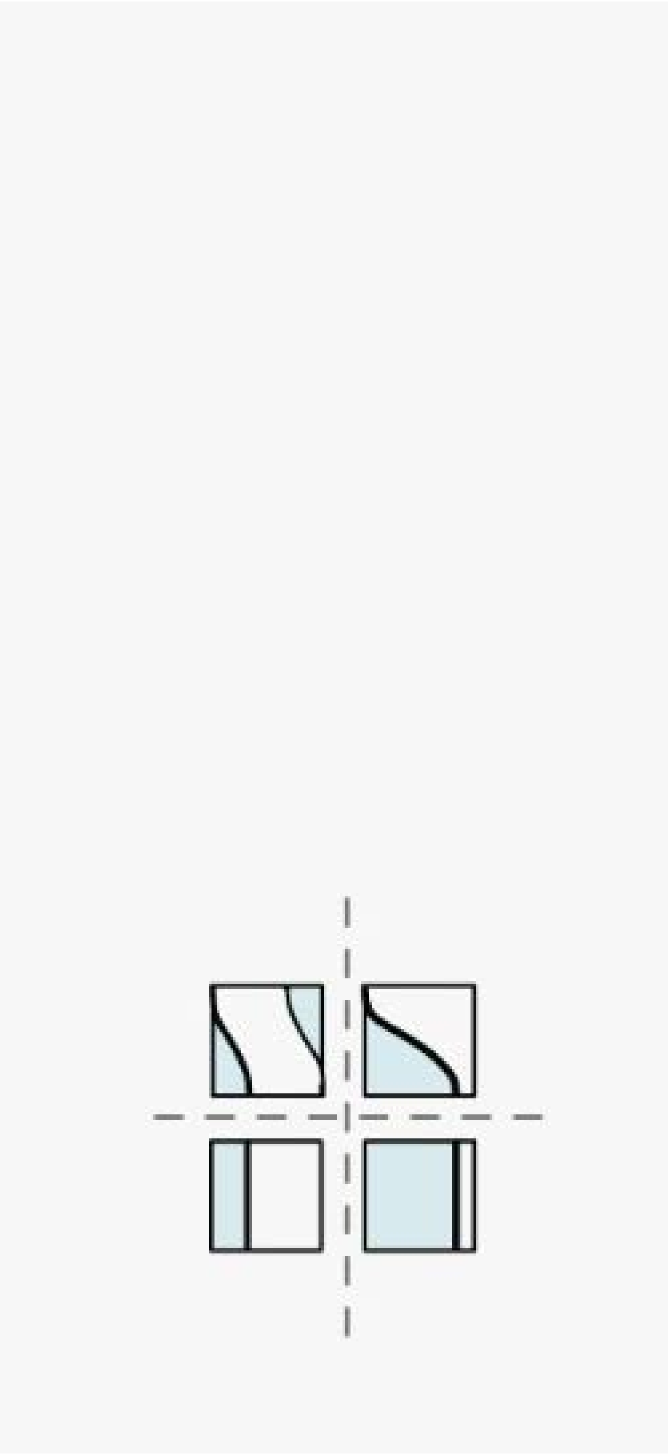

Living room
+ kitchen

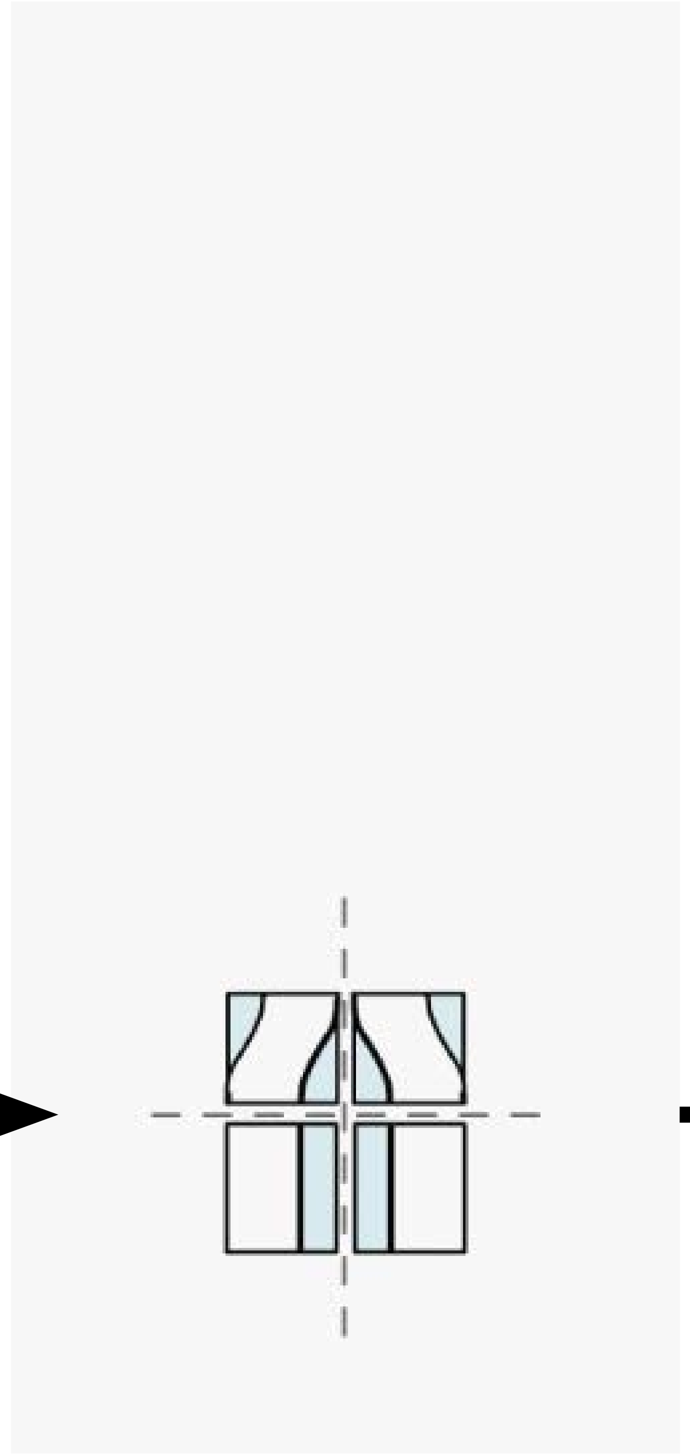
Facade tiles

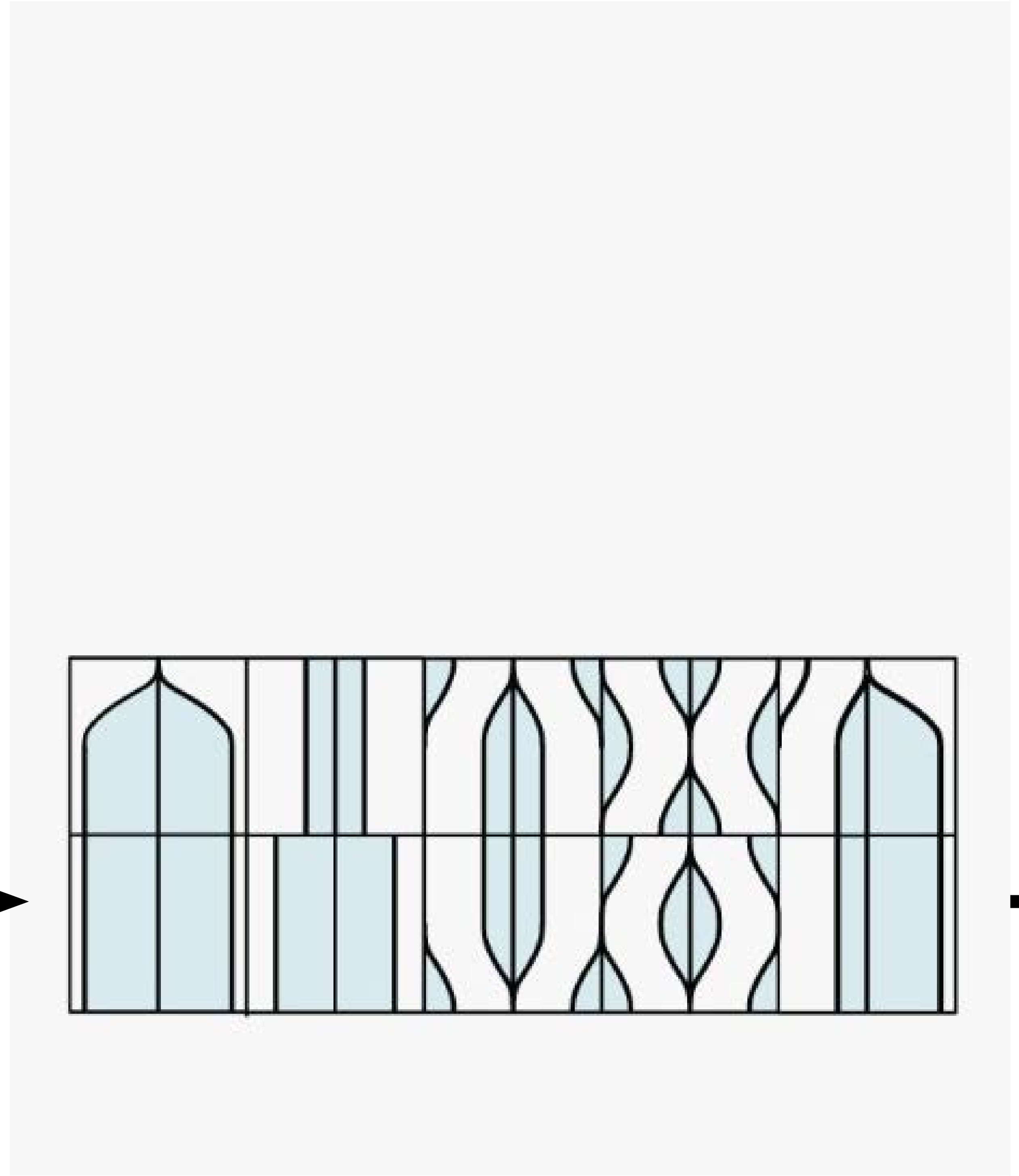


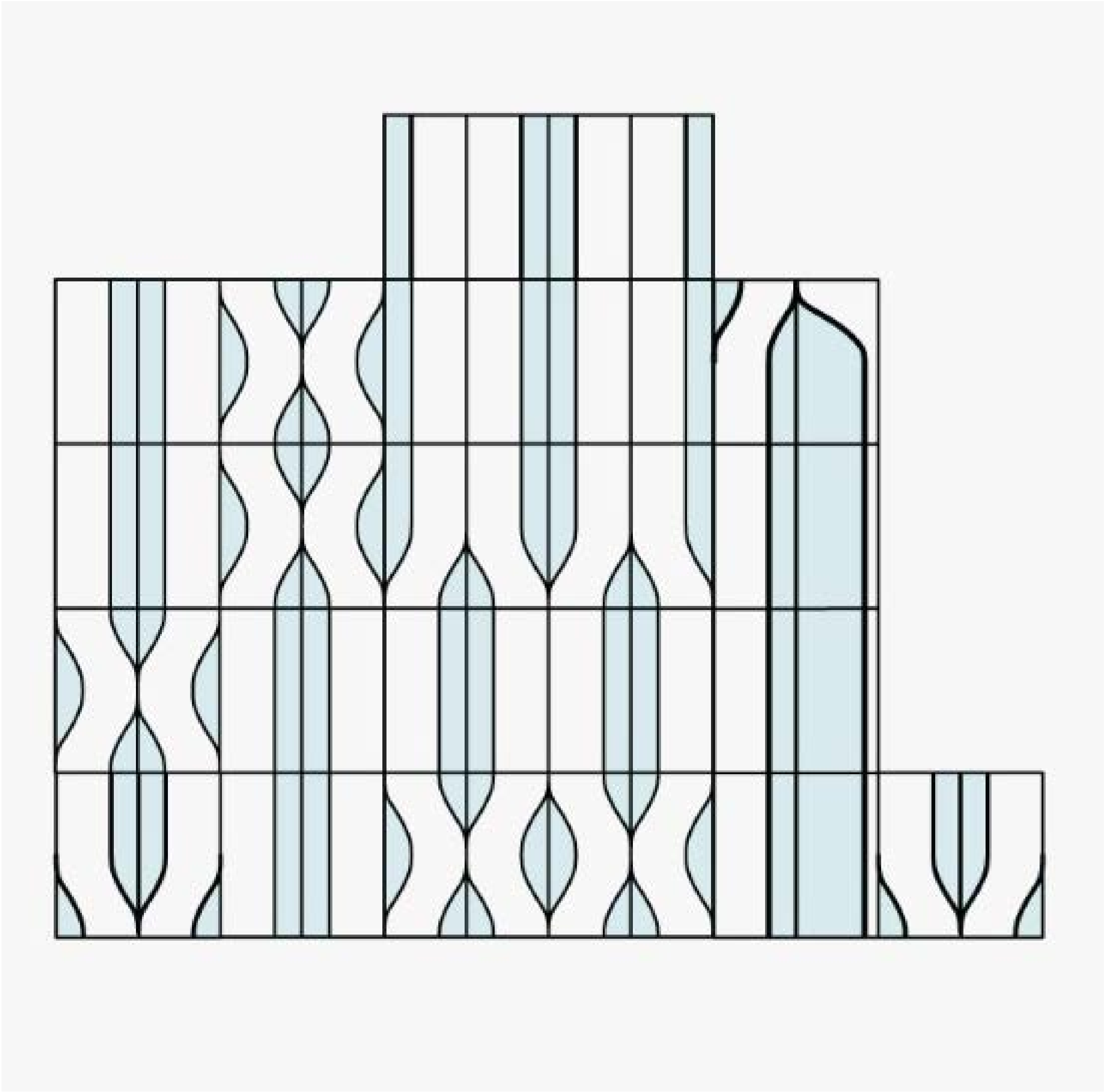
As the buidling grows, more tiles are added and the facade is enriched.

Tile creation

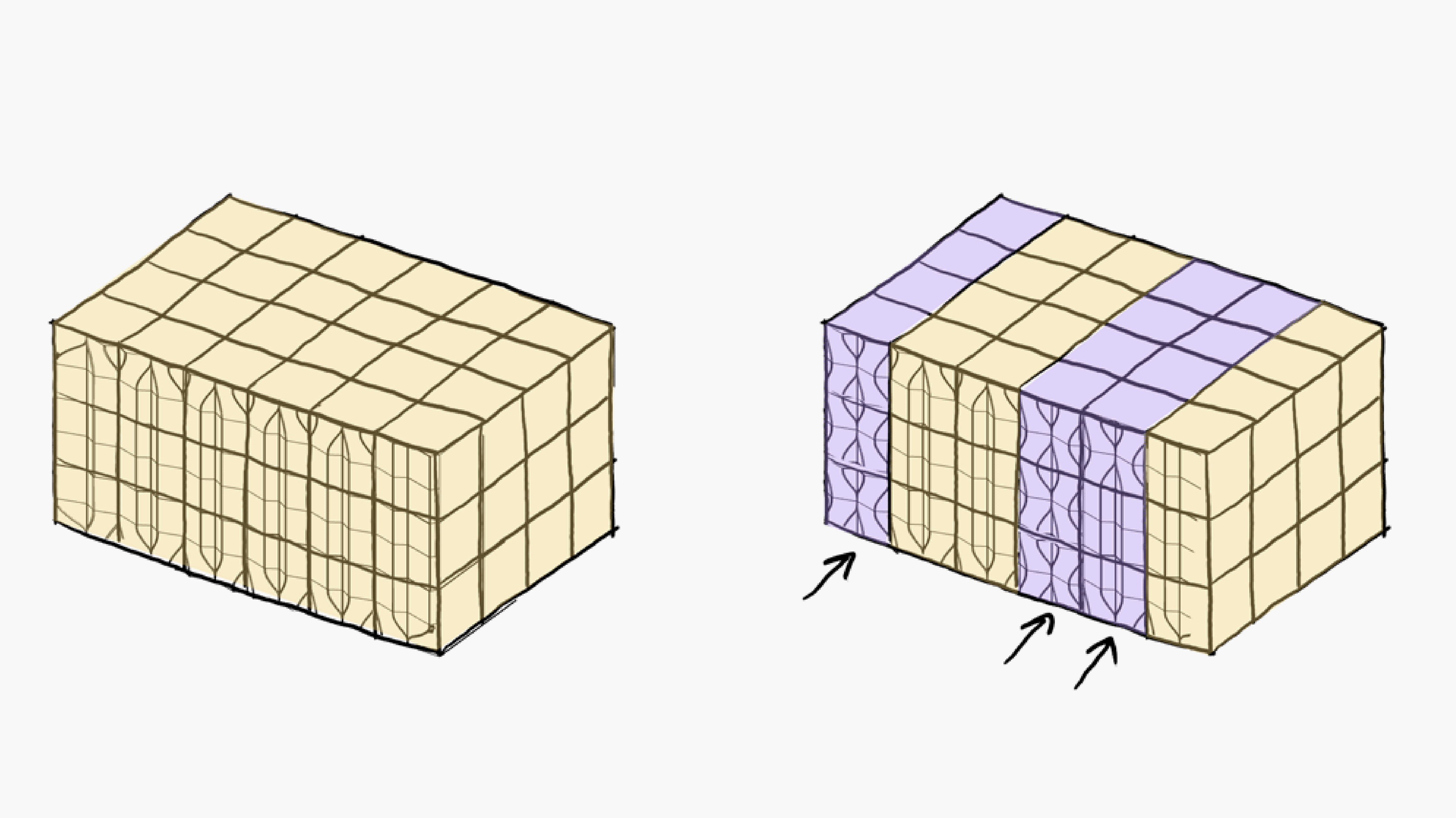

A few subtile corners


Tiles


Tile variations


Forming on voxels

Poligonization



Input: envelope lattice, several custom tile sets

Output: an .obj of a tiled facade

Load envelope lattice

Remove interior voxels by creating a Von Neumann stencil to detect neighbours

Apply stencil to envelope lattice

Remove voxels whose neighbour count is <=5

Extract cube lattice from envelope lattice

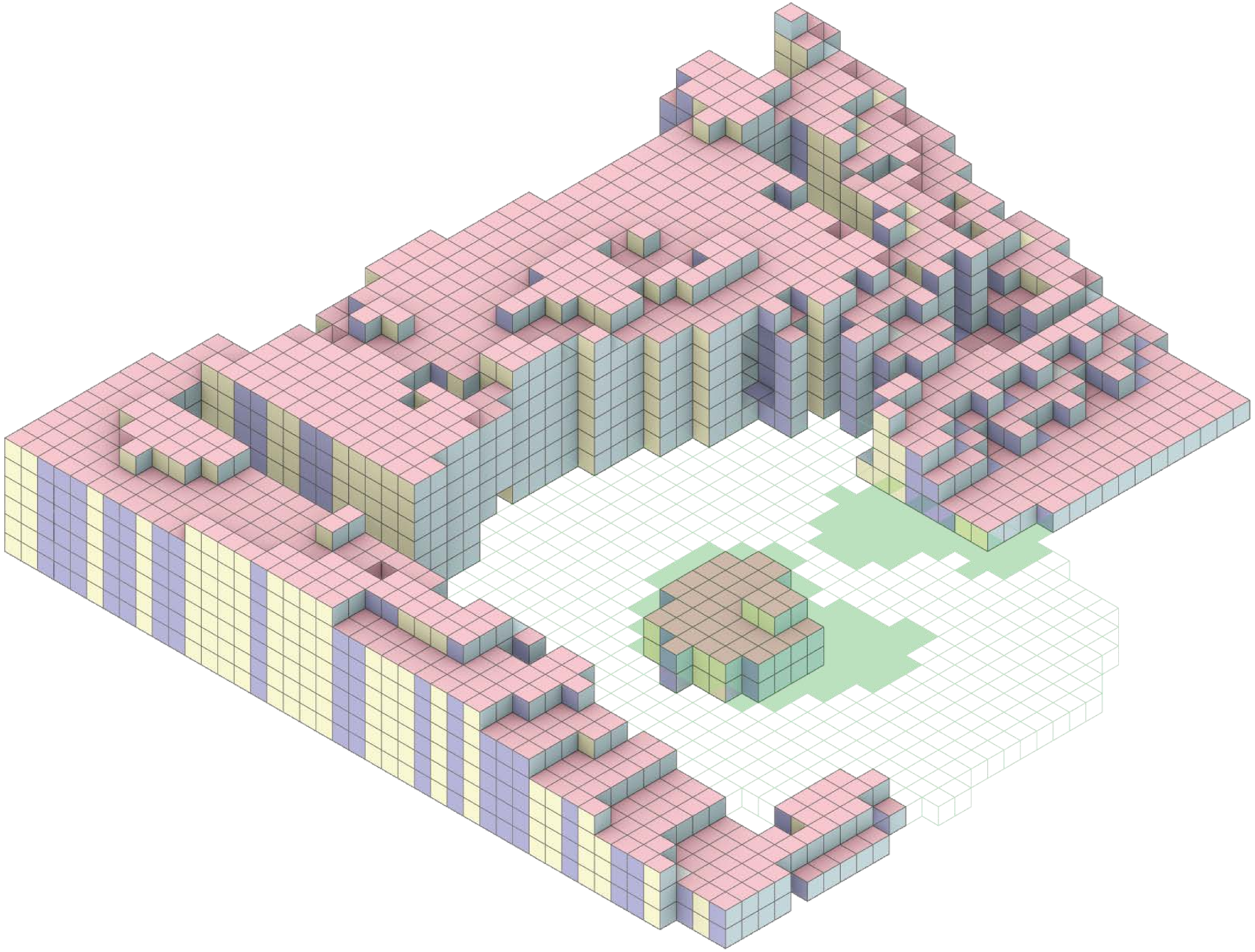
Tile the envelope lattice with tileset1

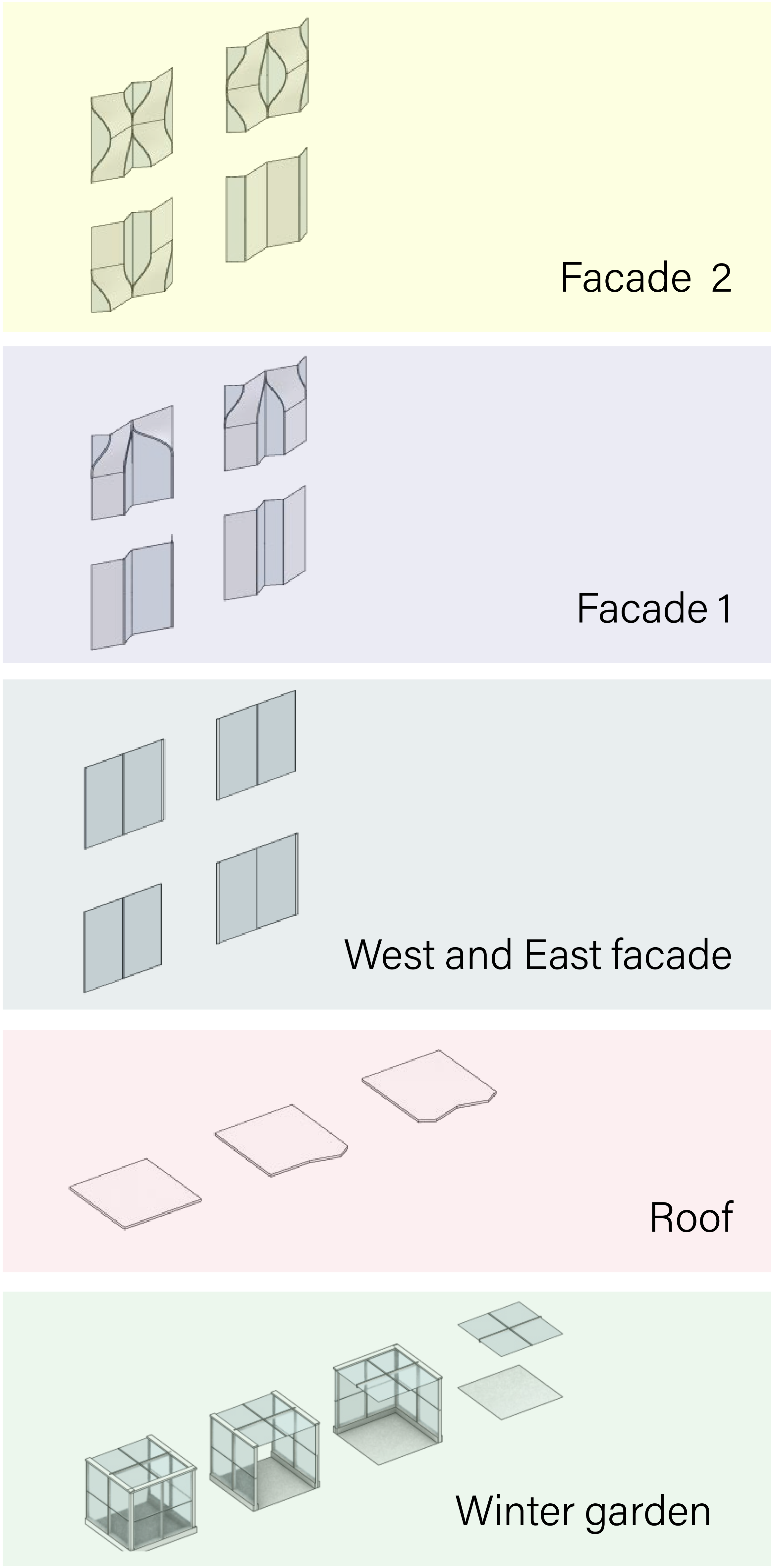
Select vertical slices in the lattice whose tiles to replace

Tile selected slices with tileset 2

Export tiled facades

Tiled voxelized envelope





ZOHO CUB3D

Hugo van Rossum \\ Maren Hengelmolen \\ Liva Sadovska \\ Sander Bentvelsen