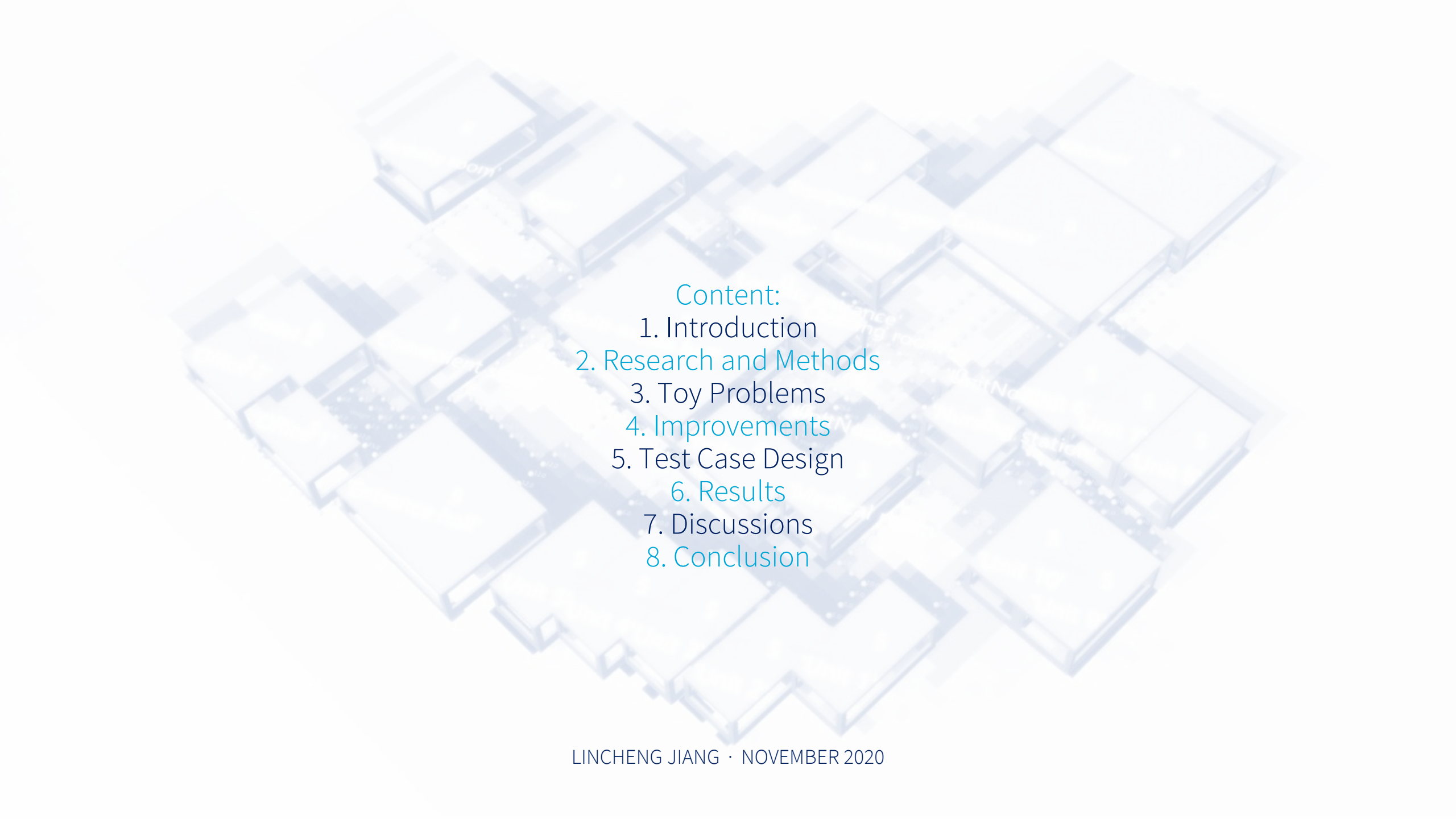
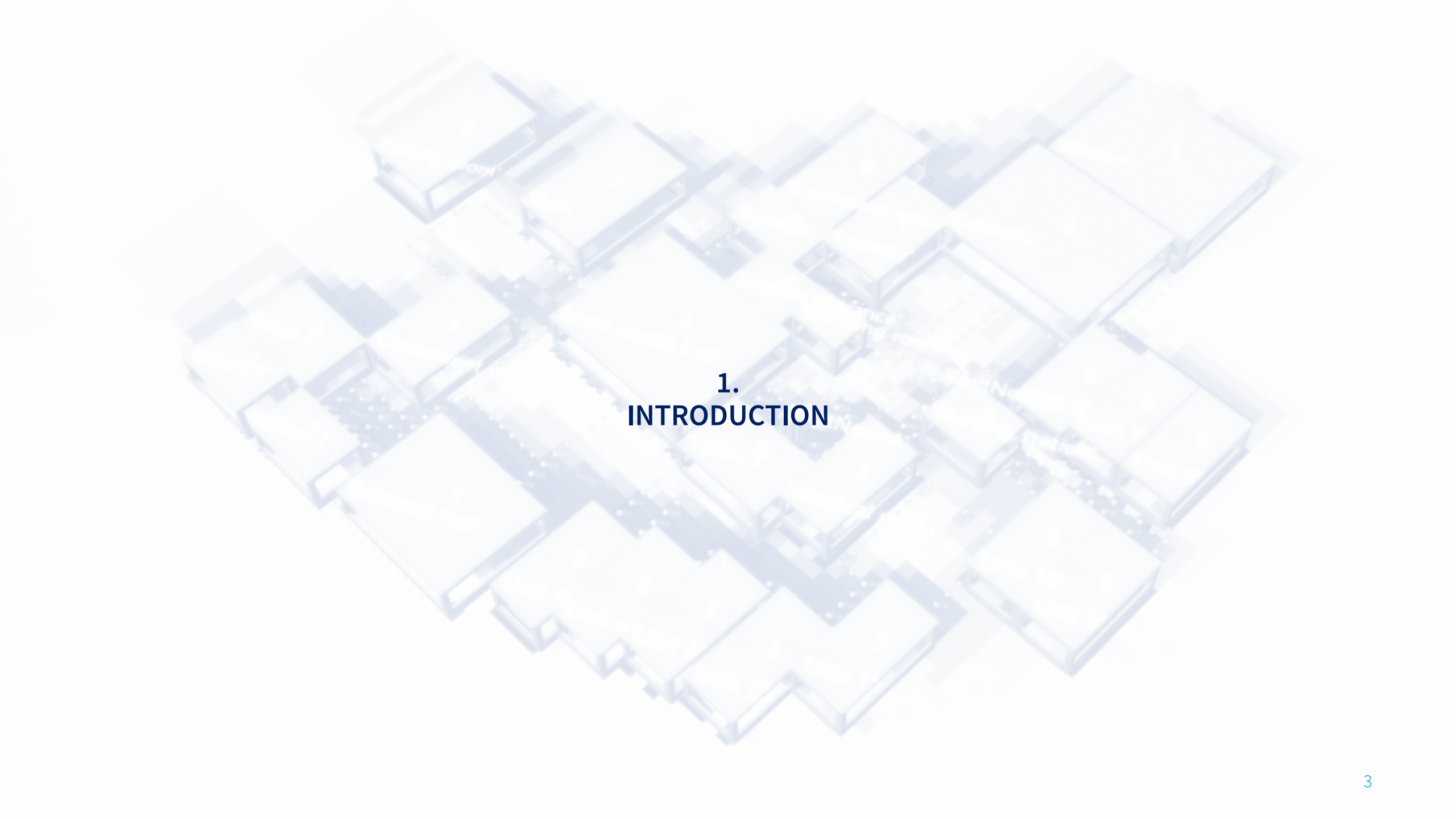
A COMPUTATIONAL METHOD TO GENERATE ONE-STORY FLOOR PLANS FOR NURSING HOMES BASED ON DAYLIGHT HOUR POTENTIAL AND SHORTEST PATH OF CIRCULATIONS

LINCHENG JIANG · NOVEMBER 2020



Content:

LINCHENG JIANG · NOVEMBER 2020



1.
INTRODUCTION

1.1
AN AGEING SOCIETY

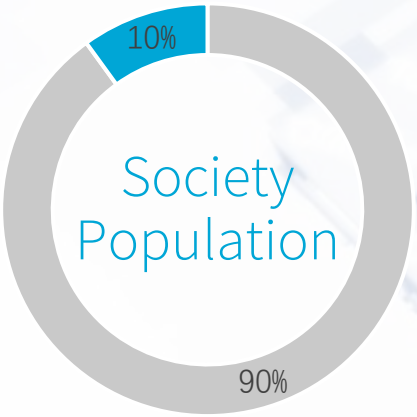
1.1 An Ageing Society

Elderlies exercising in a park in Beijing in winter.
(Source: http://blog.sina.com.cn/s/blog_4a6c8c8b0102wt0h.html)

1.1 An Ageing Society

Elderlies exercising in a park in Beijing in winter.
(Source: http://blog.sina.com.cn/s/blog_4a6c8c8b0102wt0h.html)

1.1 An Ageing Society

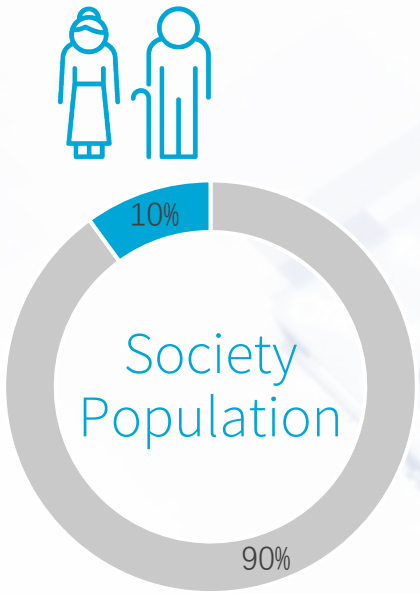


■ Age < 60 ■ Age ⩾ 60

Ageing Society

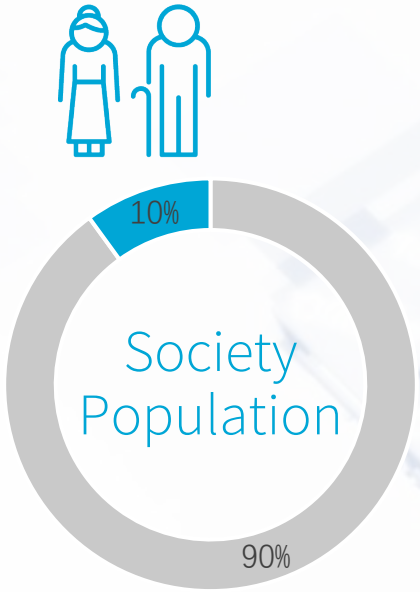
Elderlies exercising in a park in Beijing in winter.
(Source: http://blog.sina.com.cn/s/blog_4a6c8c8b0102wt0h.html)

1.1 An Ageing Society



Elderlies exercising in a park in Beijing in winter.
(Source: http://blog.sina.com.cn/s/blog_4a6c8c8b0102wt0h.html)

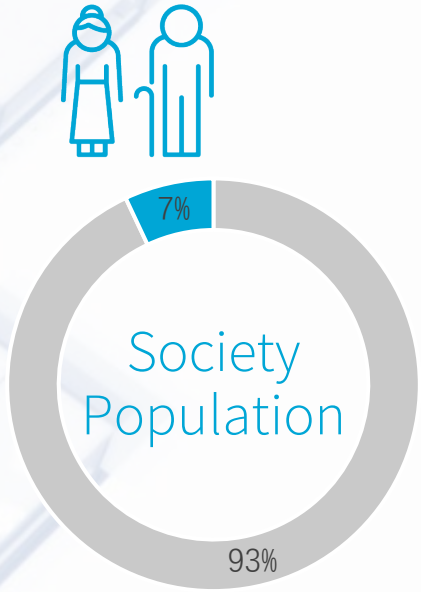
1.1 An Ageing Society



Ageing Society

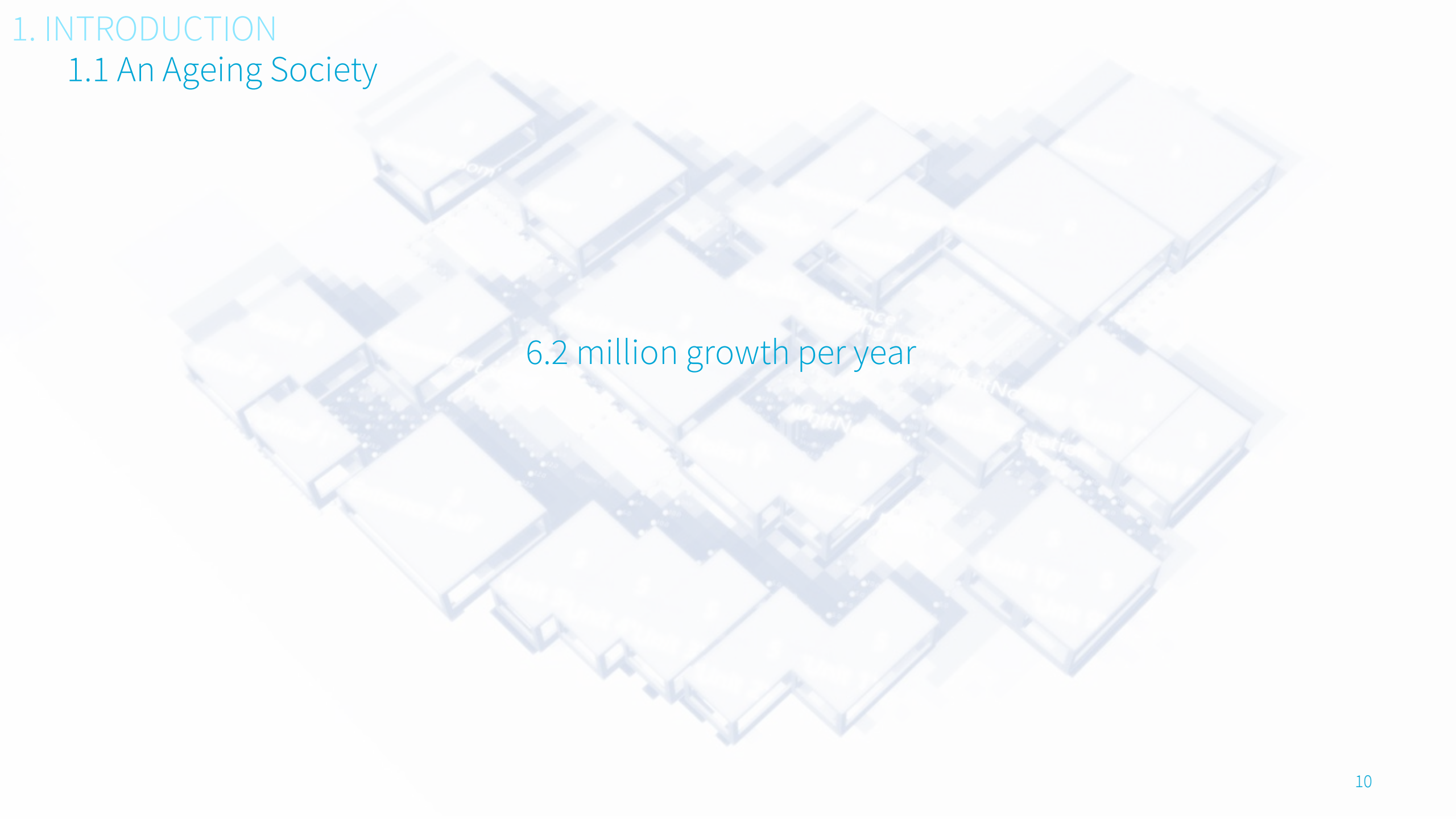
OR

Elderlies exercising in a park in Beijing in winter.
(Source: http://blog.sina.com.cn/s/blog_4a6c8c8b0102wt0h.html)

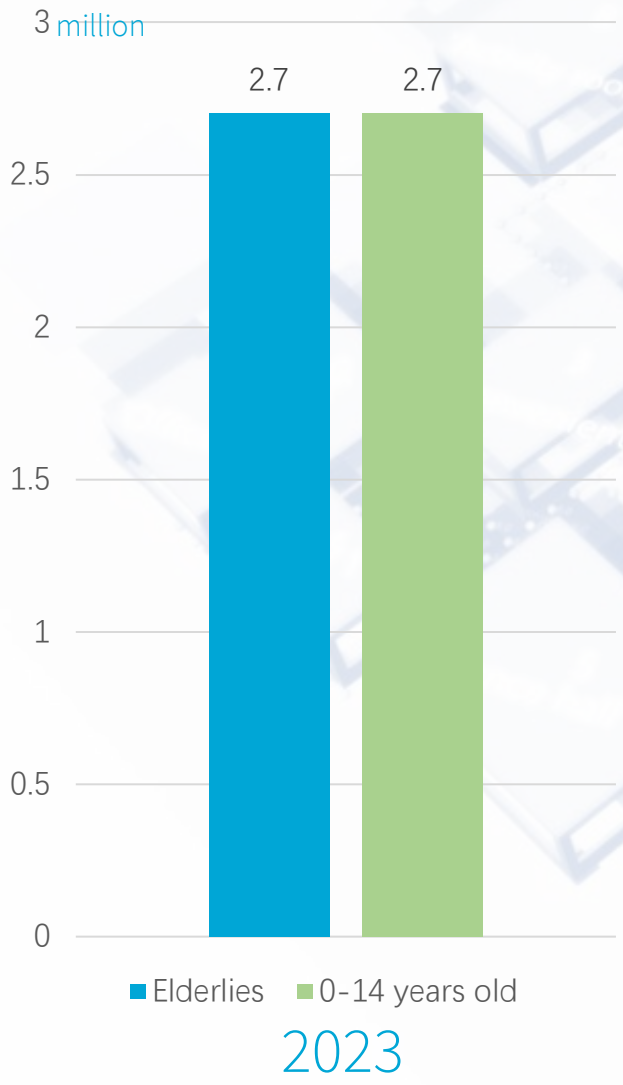


Ageing Society

1.1 An Ageing Society

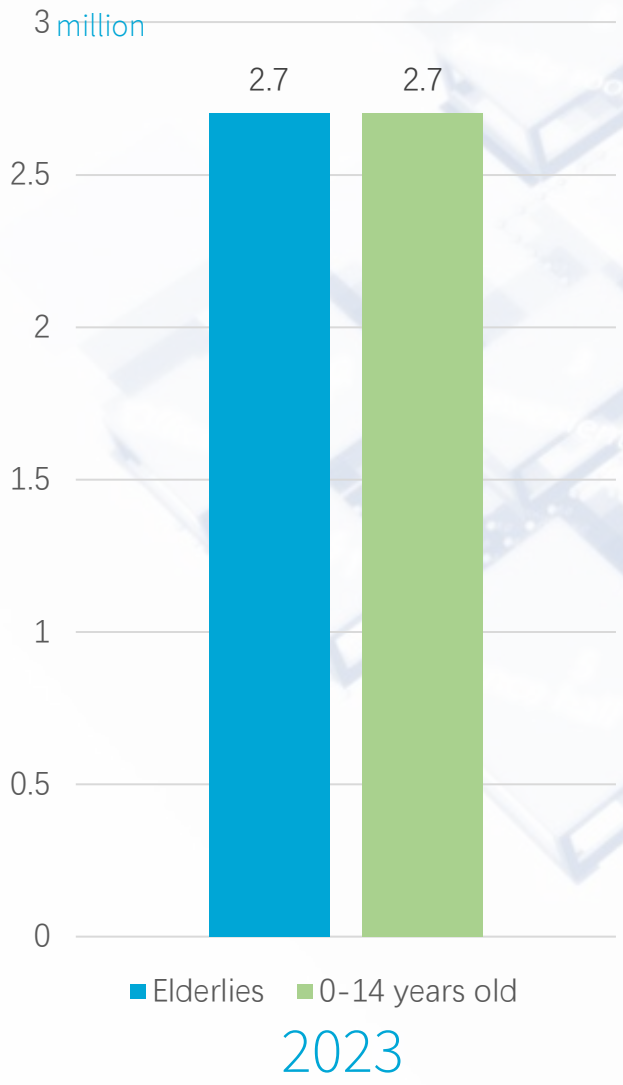


1.1 An Ageing Society

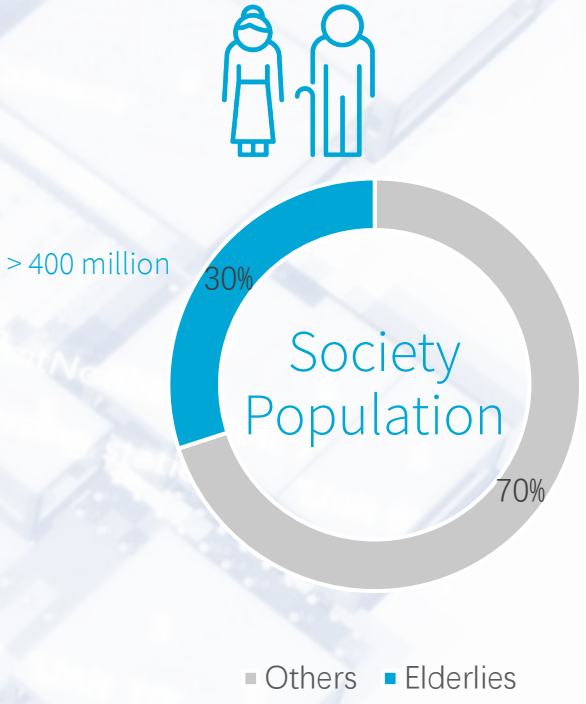


6.2 million growth per year

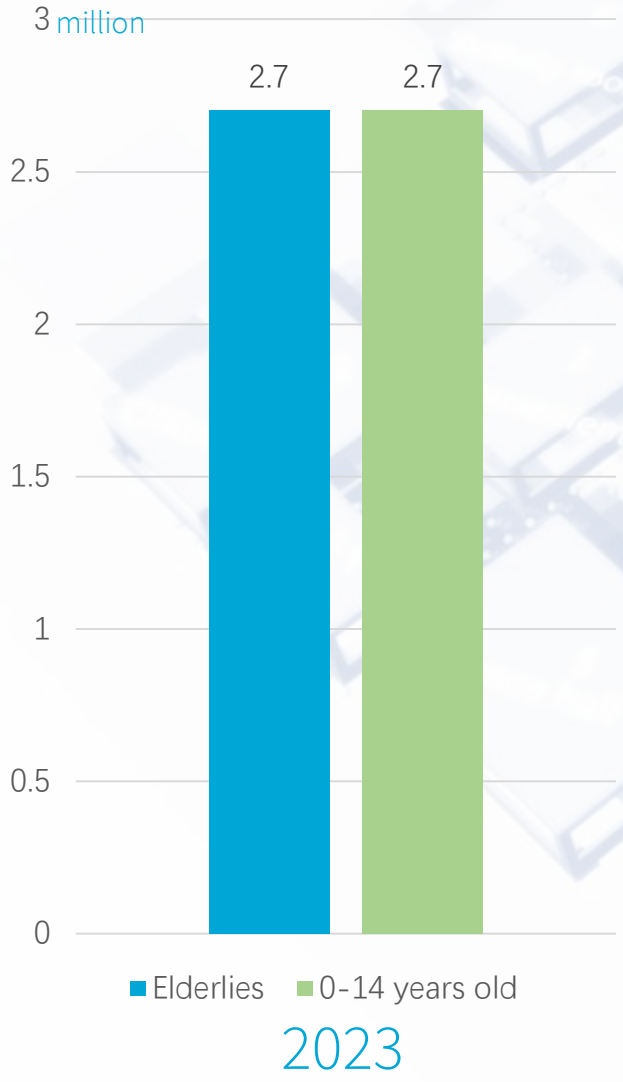
1.1 An Ageing Society



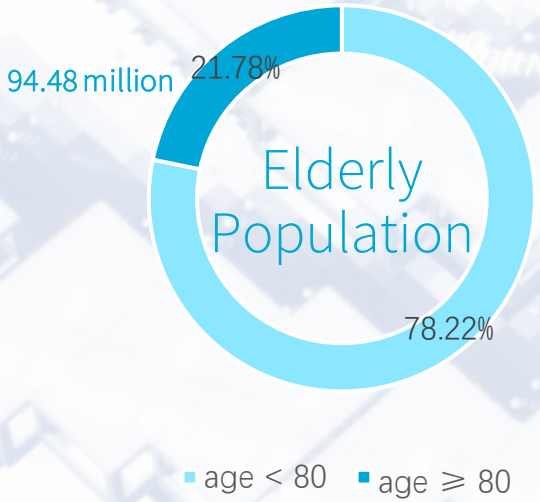
6.2 million growth per year

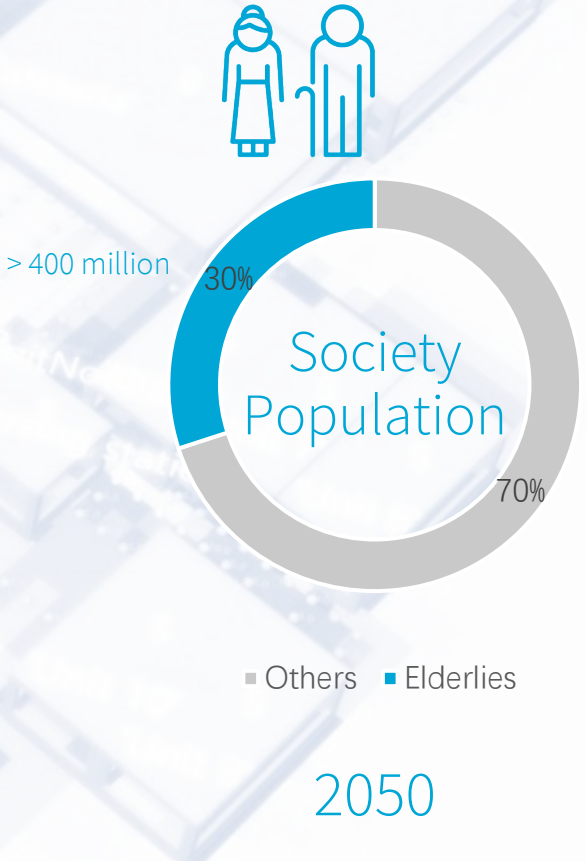


1.1 An Ageing Society



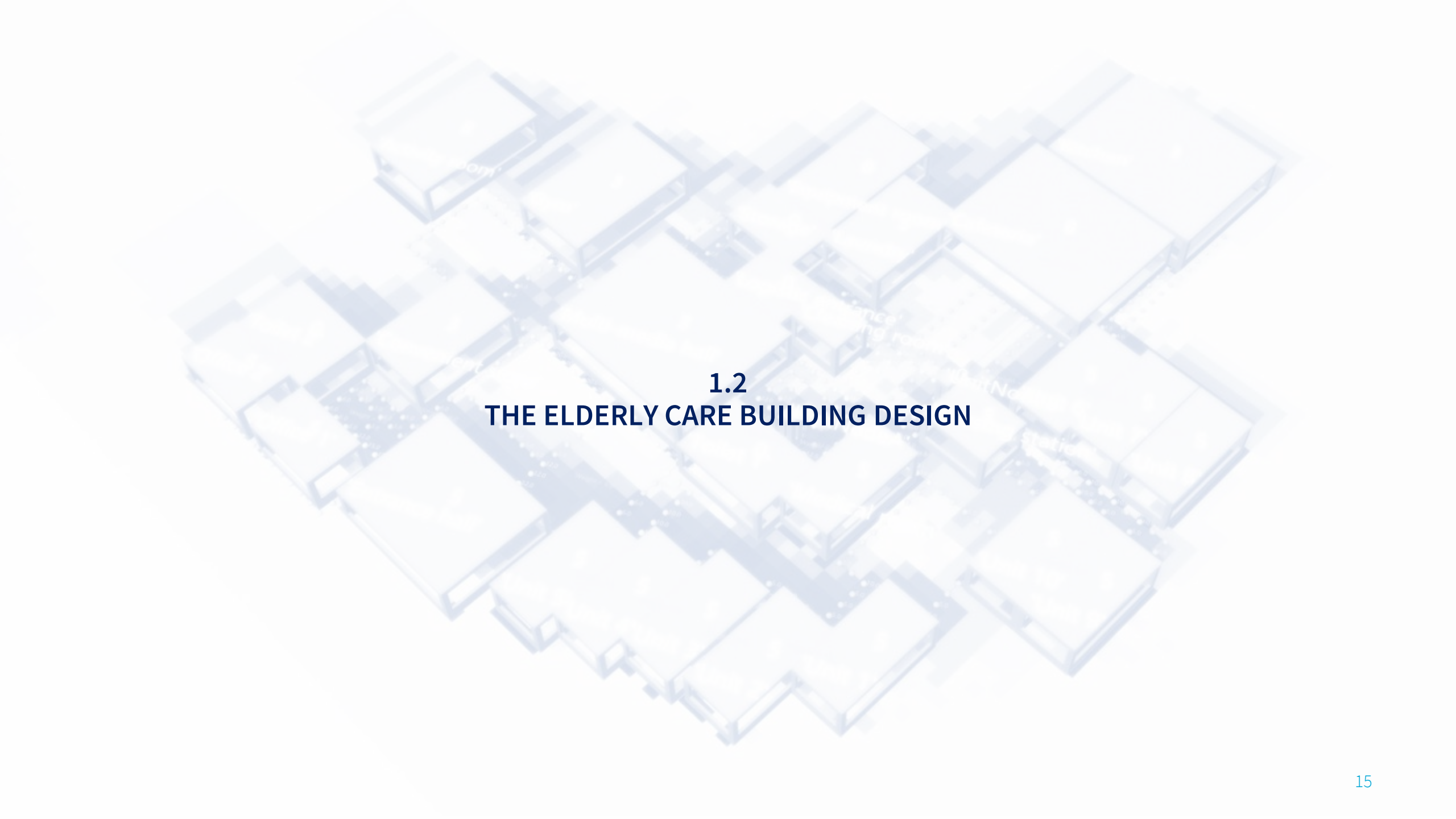
6.2 million growth per year





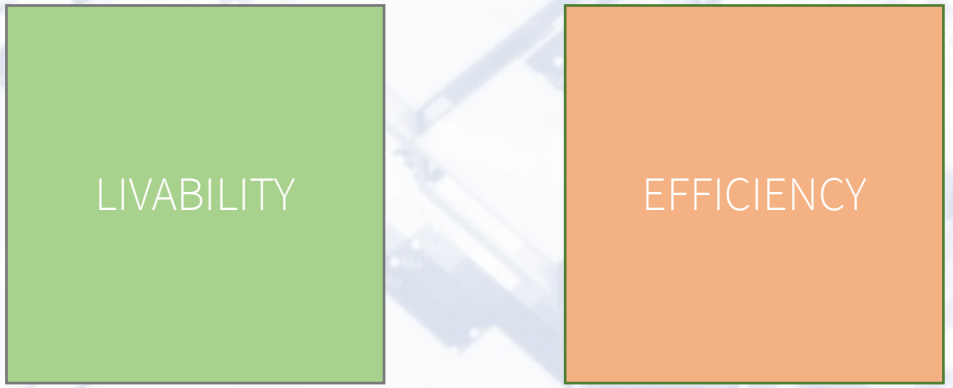
1.1 An Ageing Society

Home for Dependent Elderly People and Nursing Home / Dominique Coulon & associés(source: https://www.archdaily.com/794834/home-for-dependent-elderly-people-and-nursing-home-dominique-coulon-and-associes)

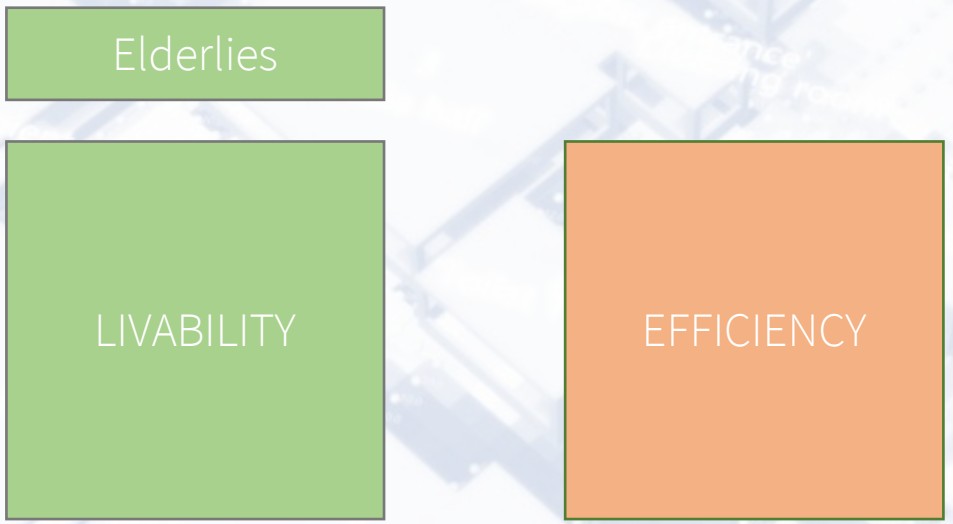


1.2
THE ELDERLY CARE BUILDING DESIGN

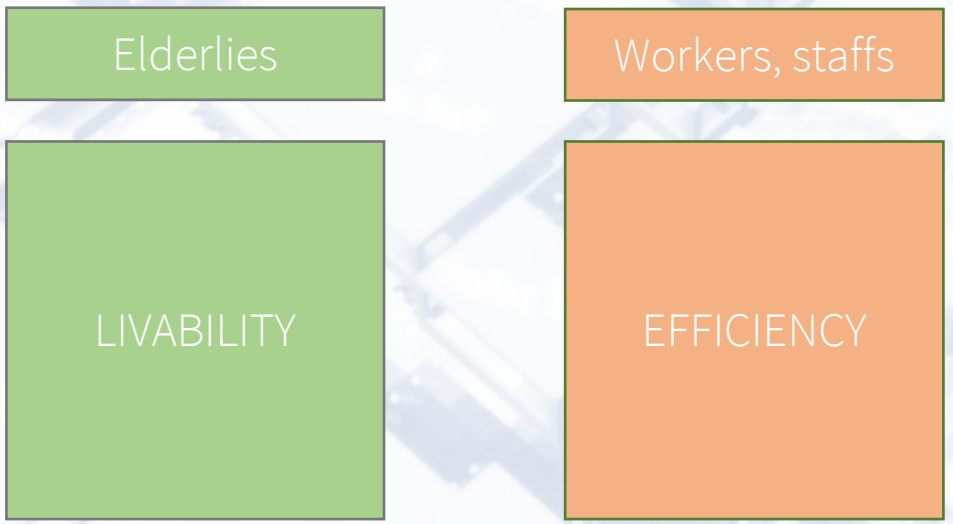
1.2 The Elderly Care Building Design



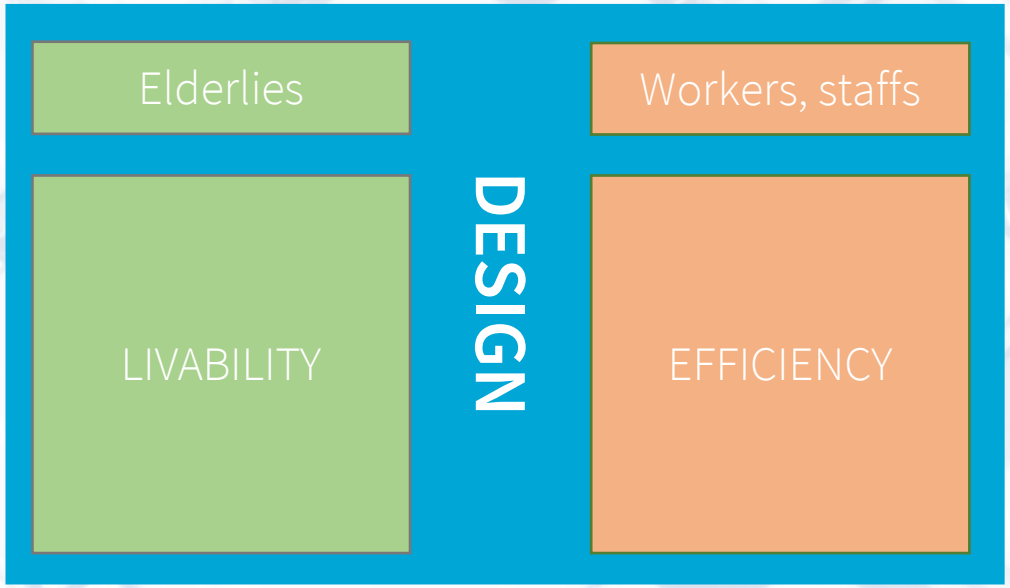
1.2 The Elderly Care Building Design



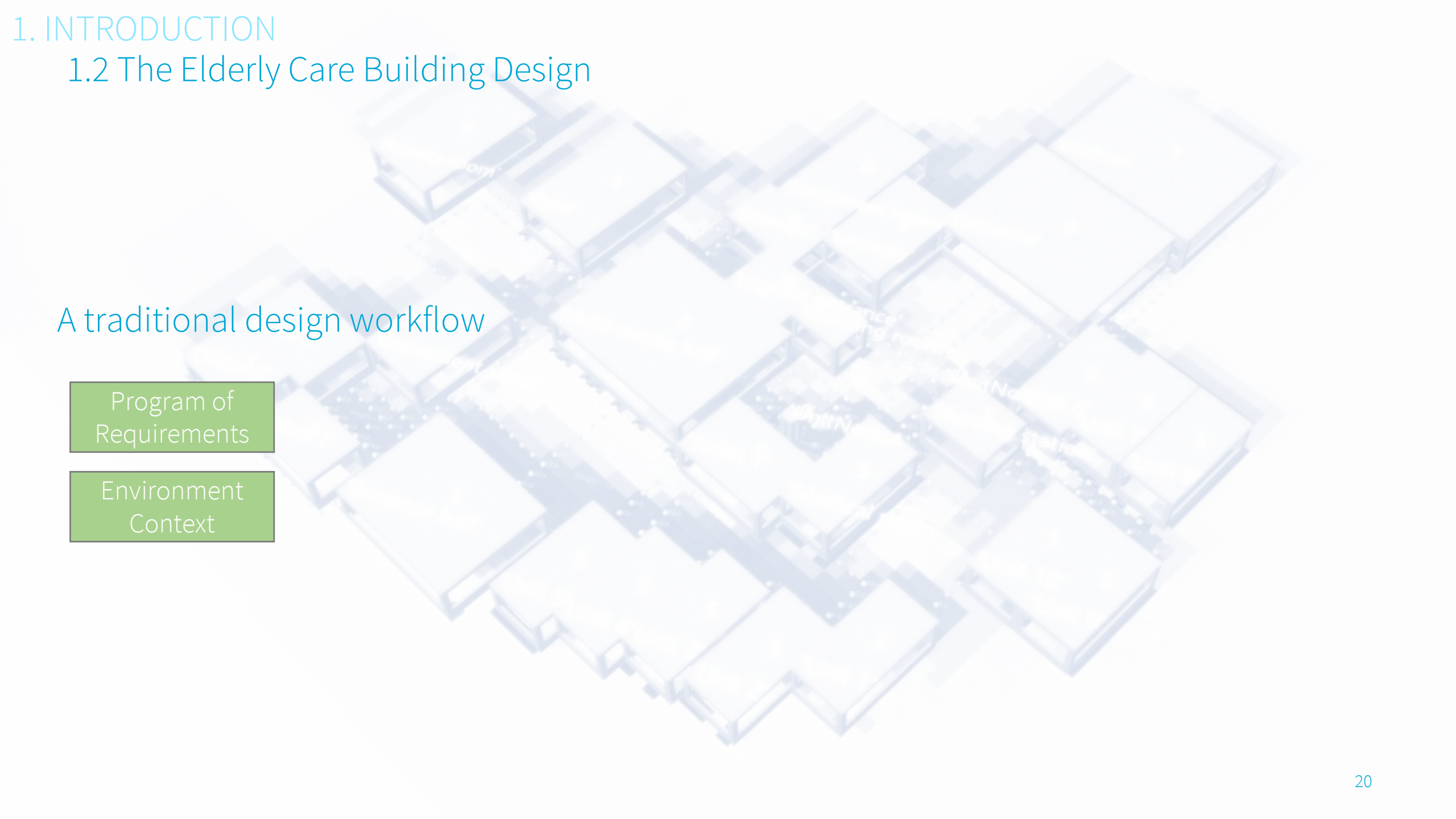
1.2 The Elderly Care Building Design



1.2 The Elderly Care Building Design



1.2 The Elderly Care Building Design



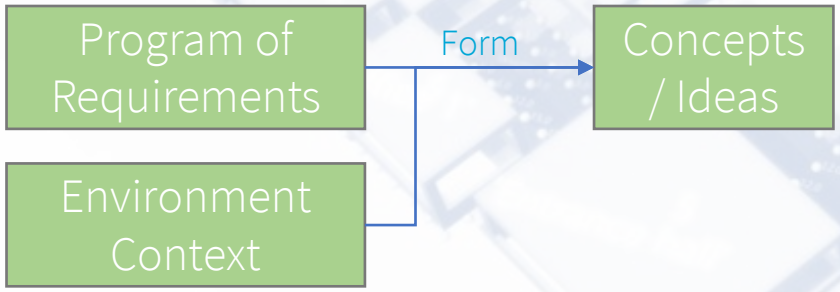
A traditional design workflow

Program of Requirements

Environment Context

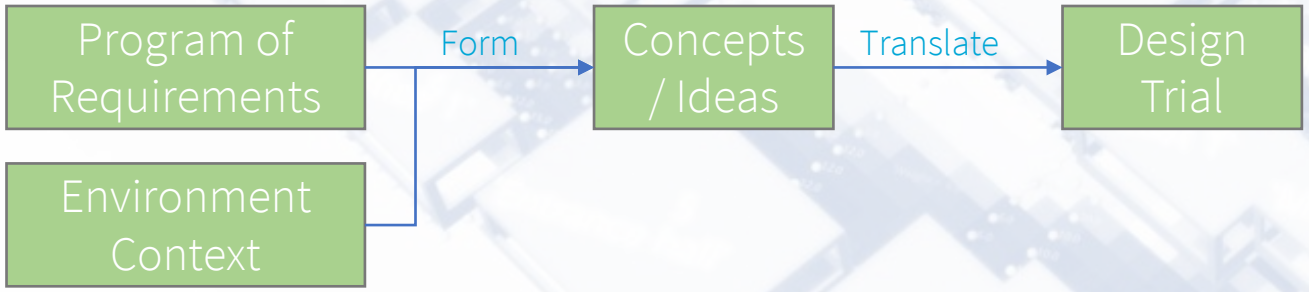
1.2 The Elderly Care Building Design

A traditional design workflow



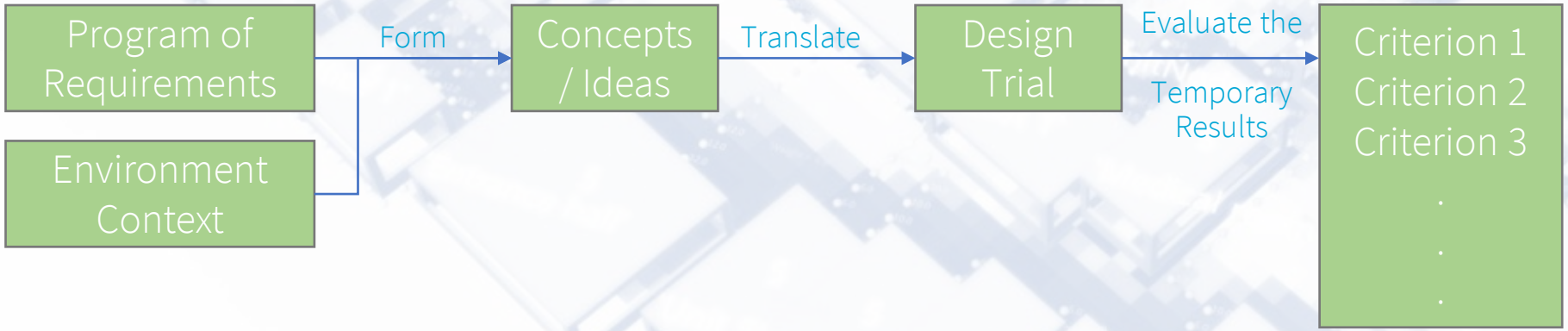
1.2 The Elderly Care Building Design

A traditional design workflow



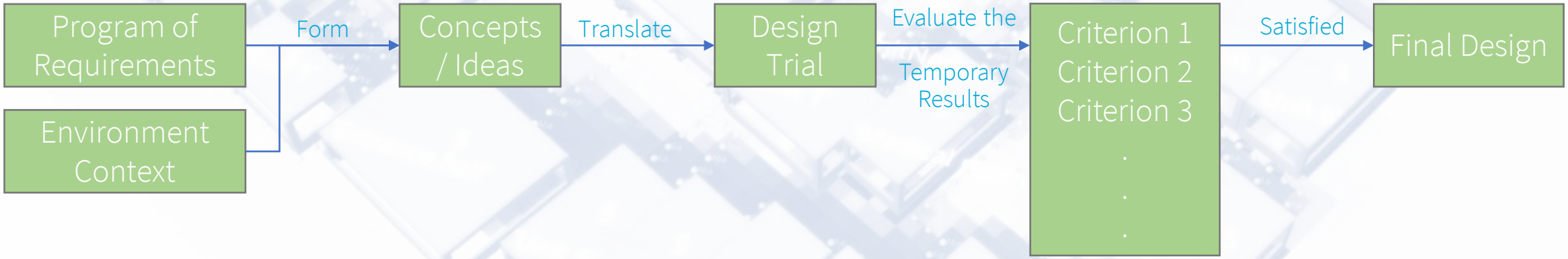
1.2 The Elderly Care Building Design

A traditional design workflow



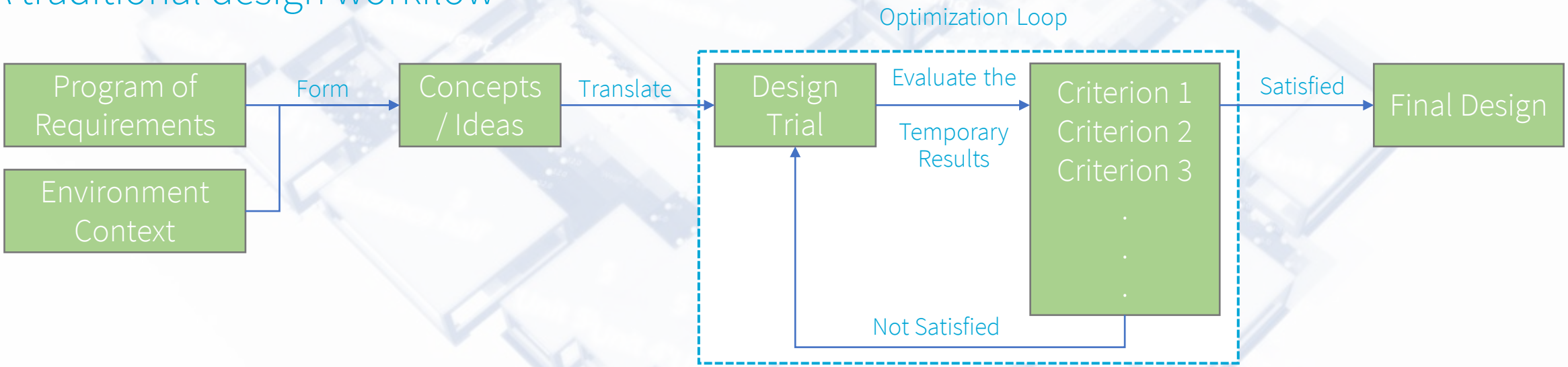
1.2 The Elderly Care Building Design

A traditional design workflow



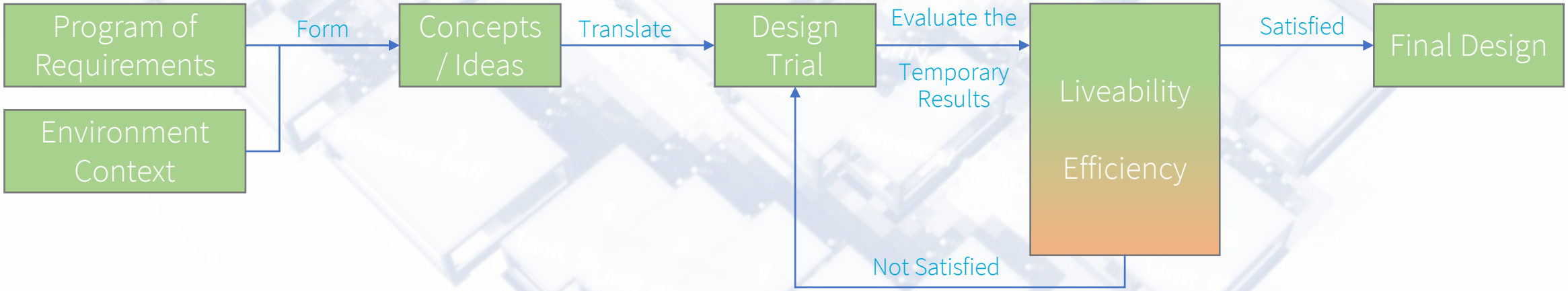
1.2 The Elderly Care Building Design

A traditional design workflow



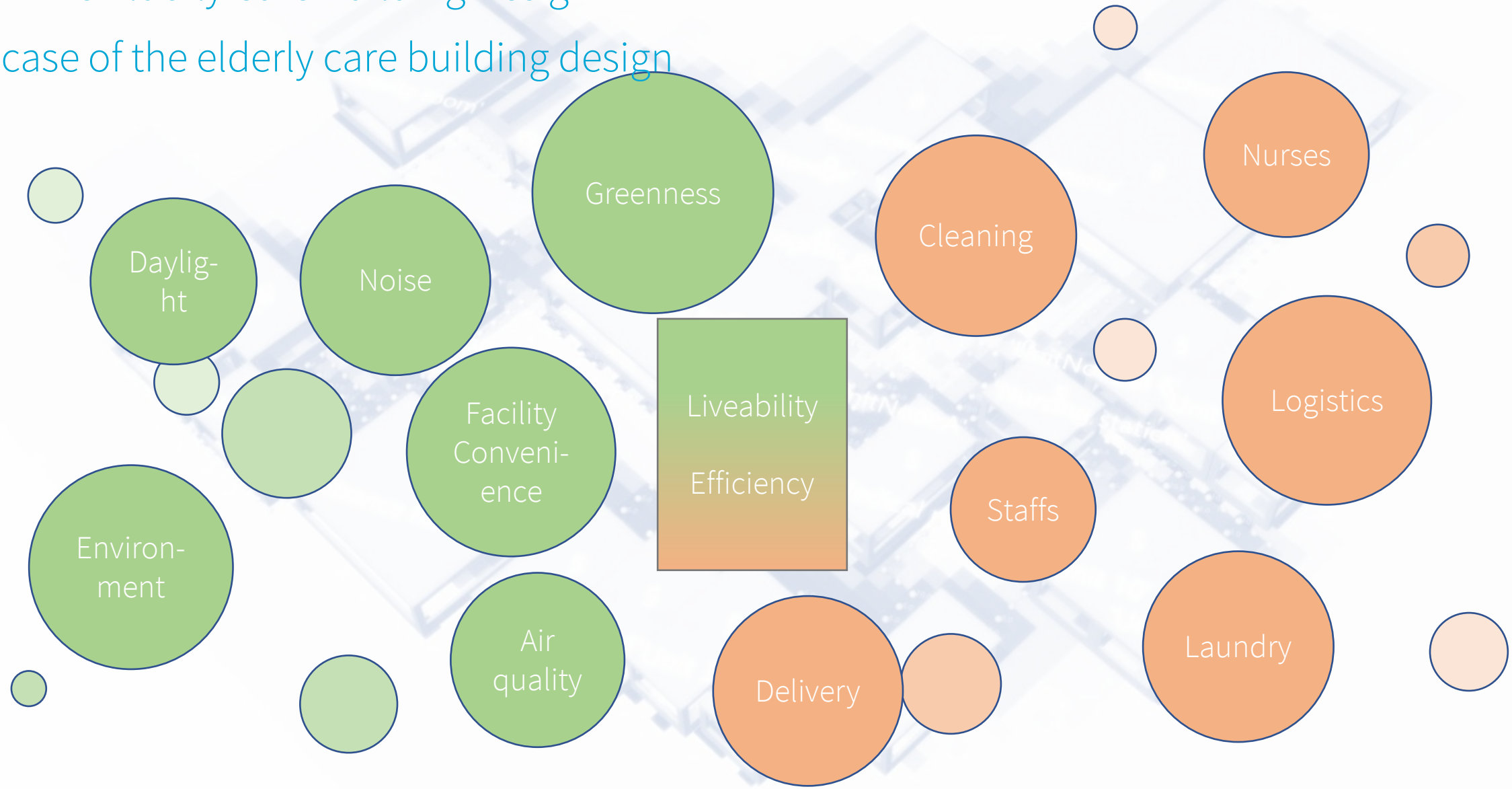
1.2 The Elderly Care Building Design

In case of the elderly care building design



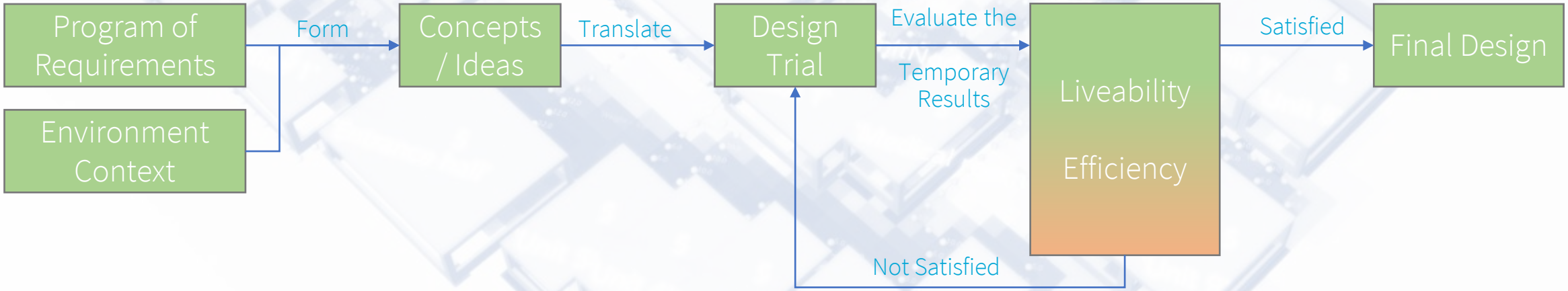
1.2 The Elderly Care Building Design

In case of the elderly care building design



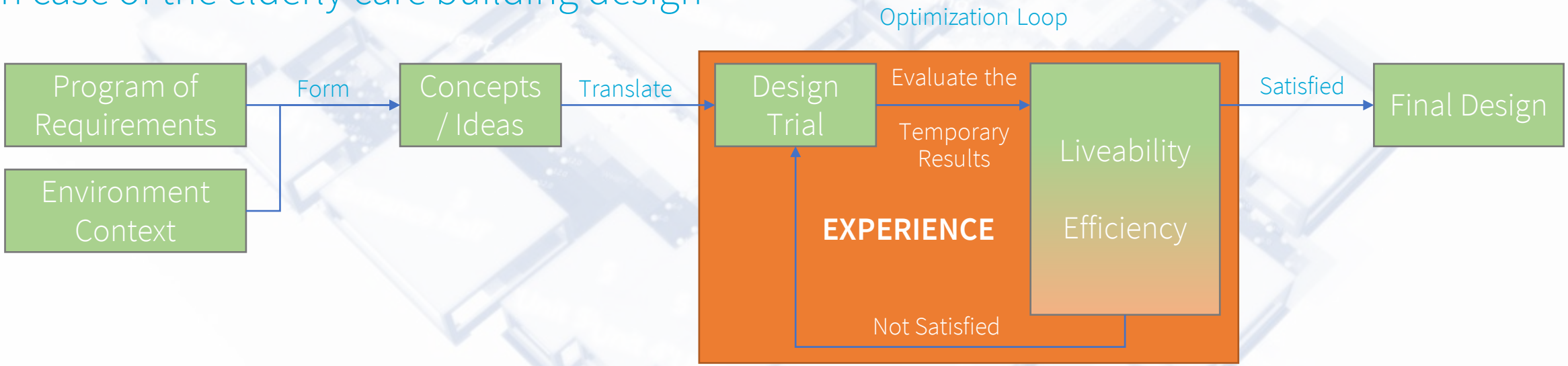
1.2 The Elderly Care Building Design

In case of the elderly care building design



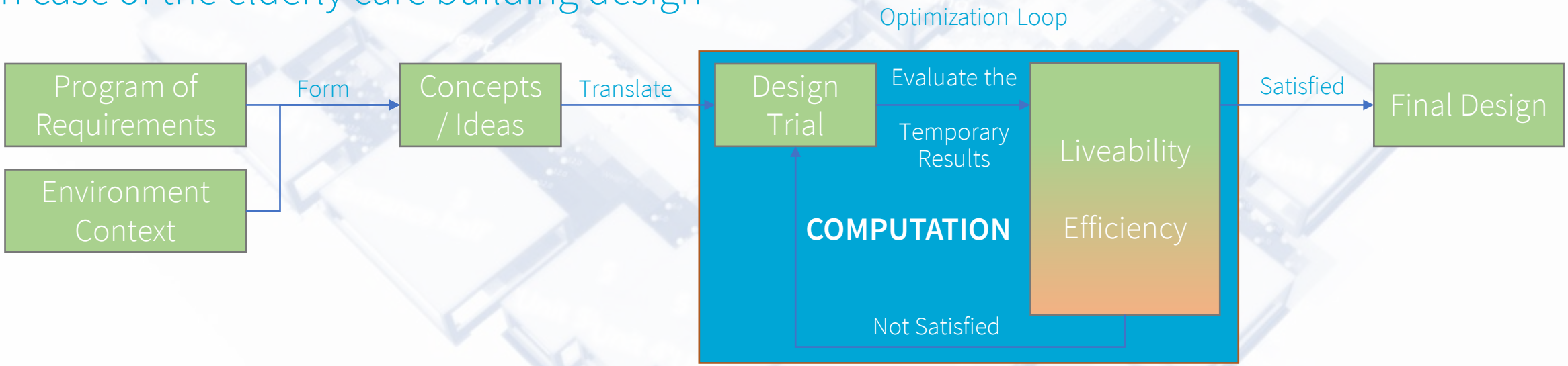
1.2 The Elderly Care Building Design

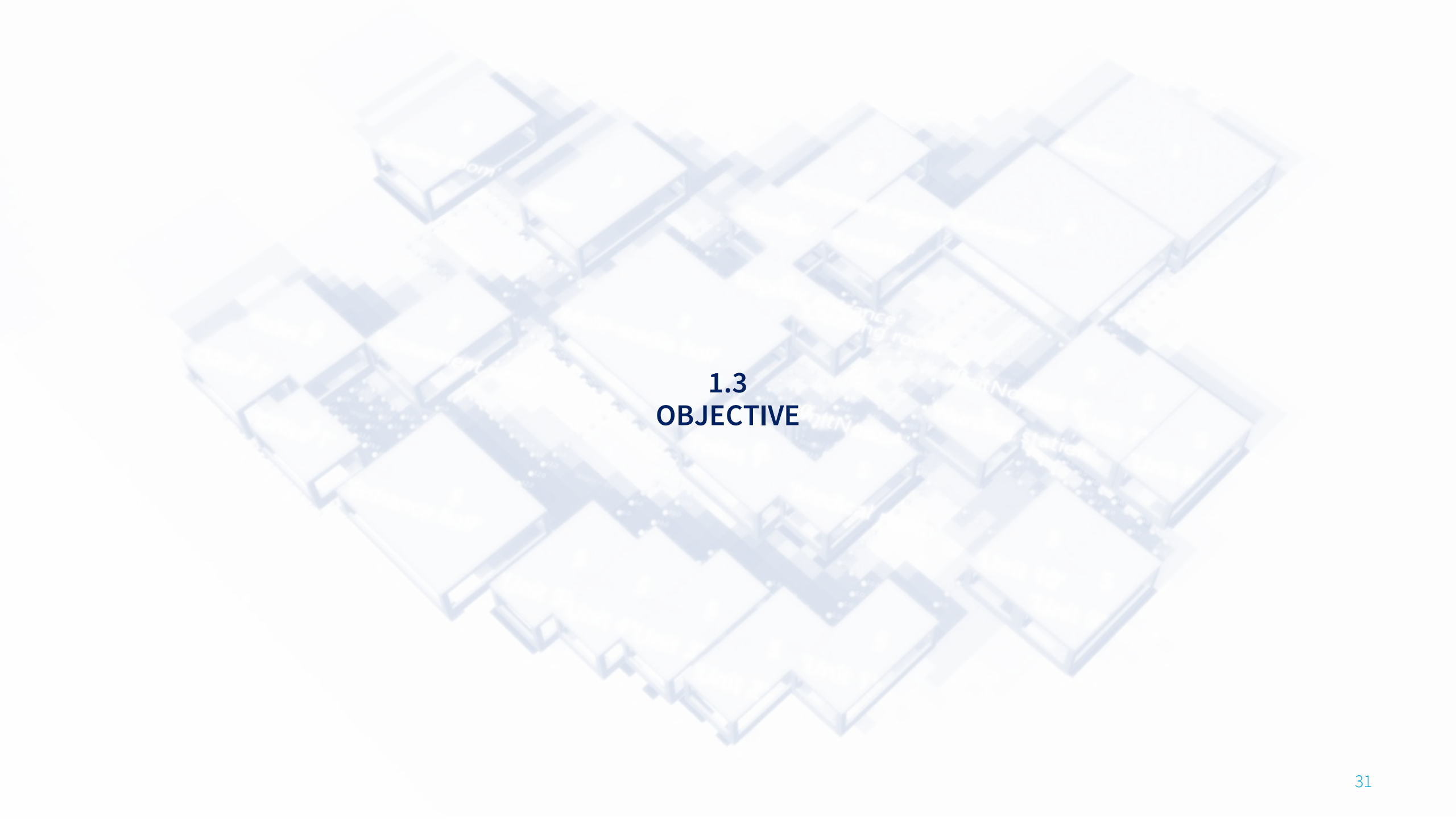
In case of the elderly care building design



1.2 The Elderly Care Building Design

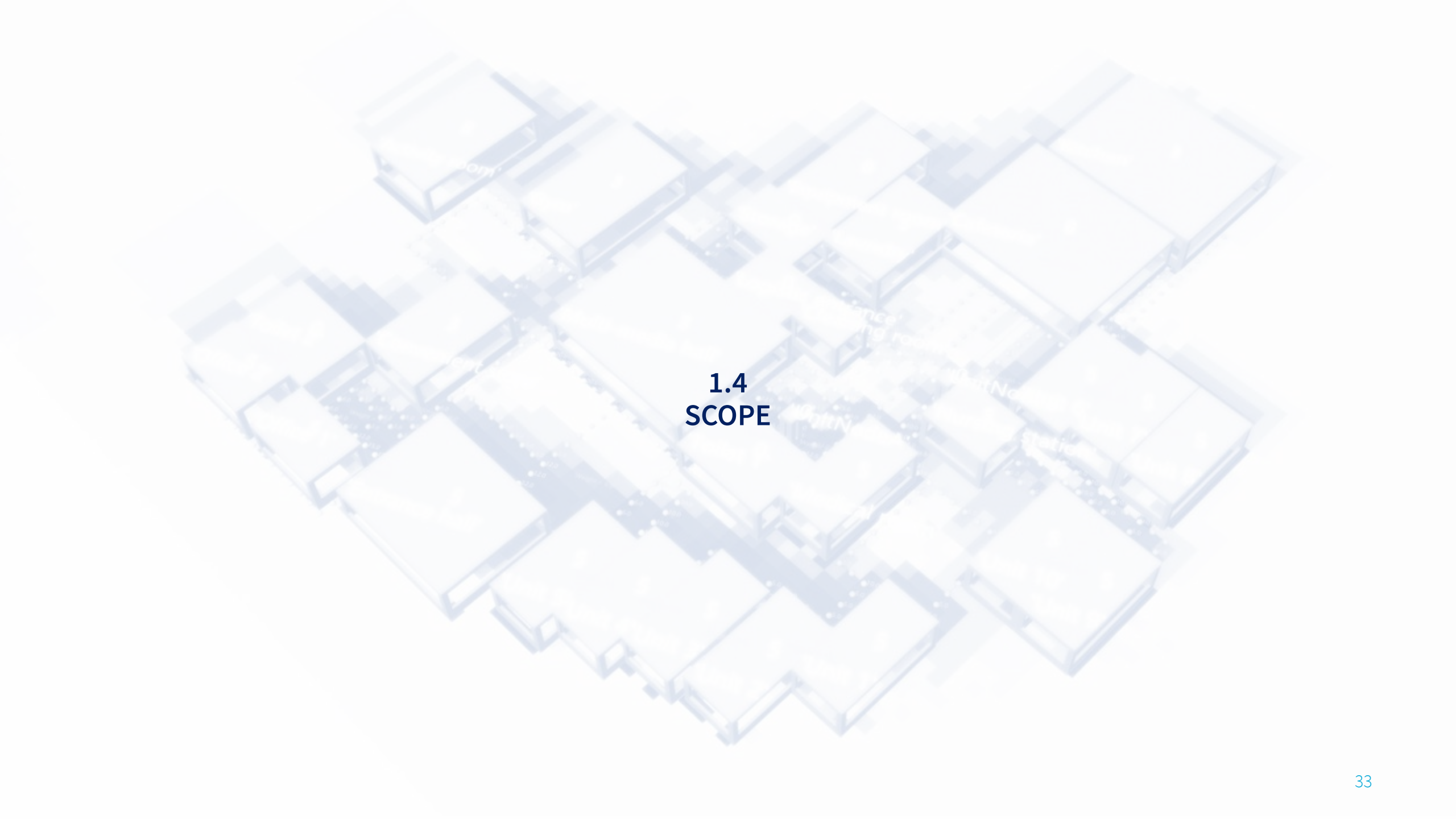
In case of the elderly care building design





1.3
OBJECTIVE

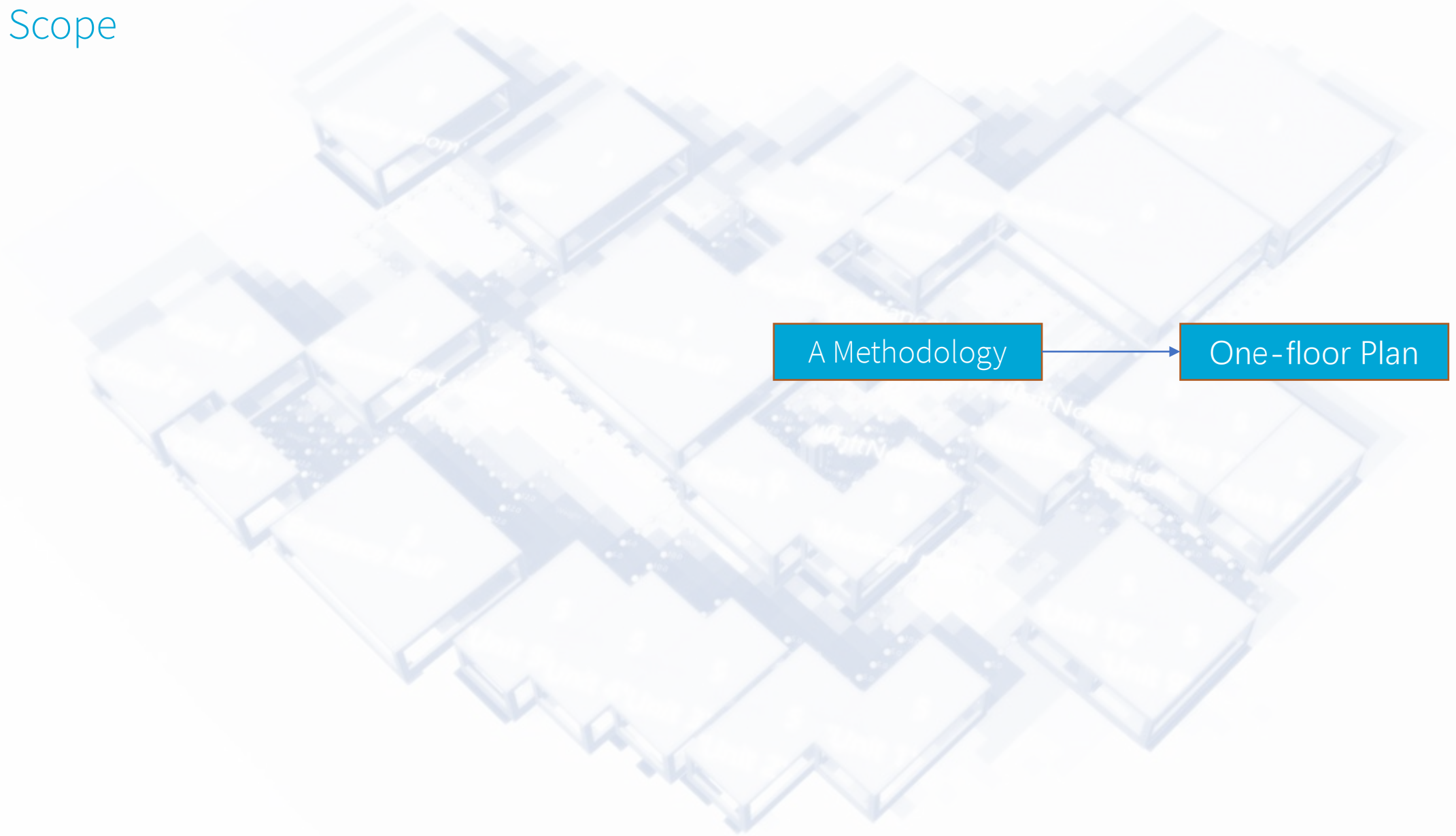
The objective of this thesis is to develop a computational method that generates one-story floor plan, that meets both circulation and daylight hour requirements of the building, which can be integrated into the architectural design process.



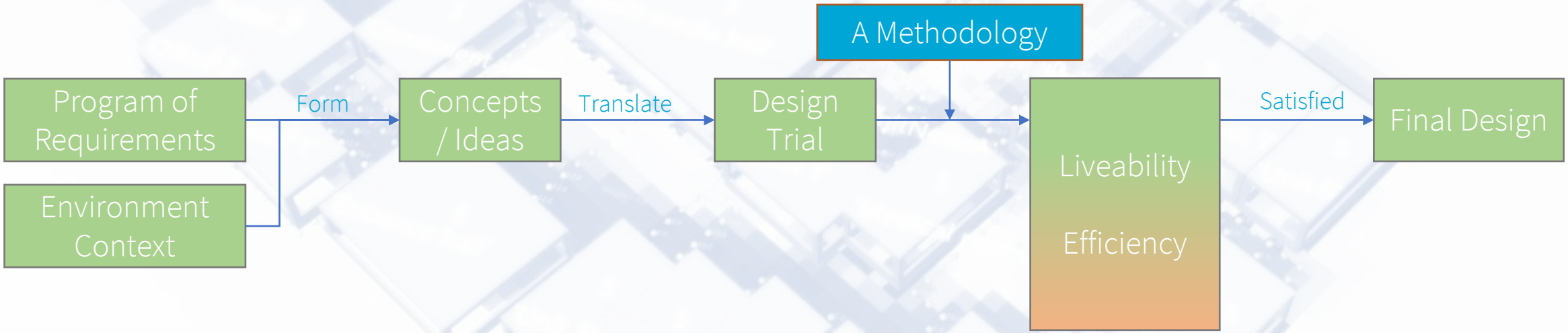
1.4
SCOPE


A Methodology

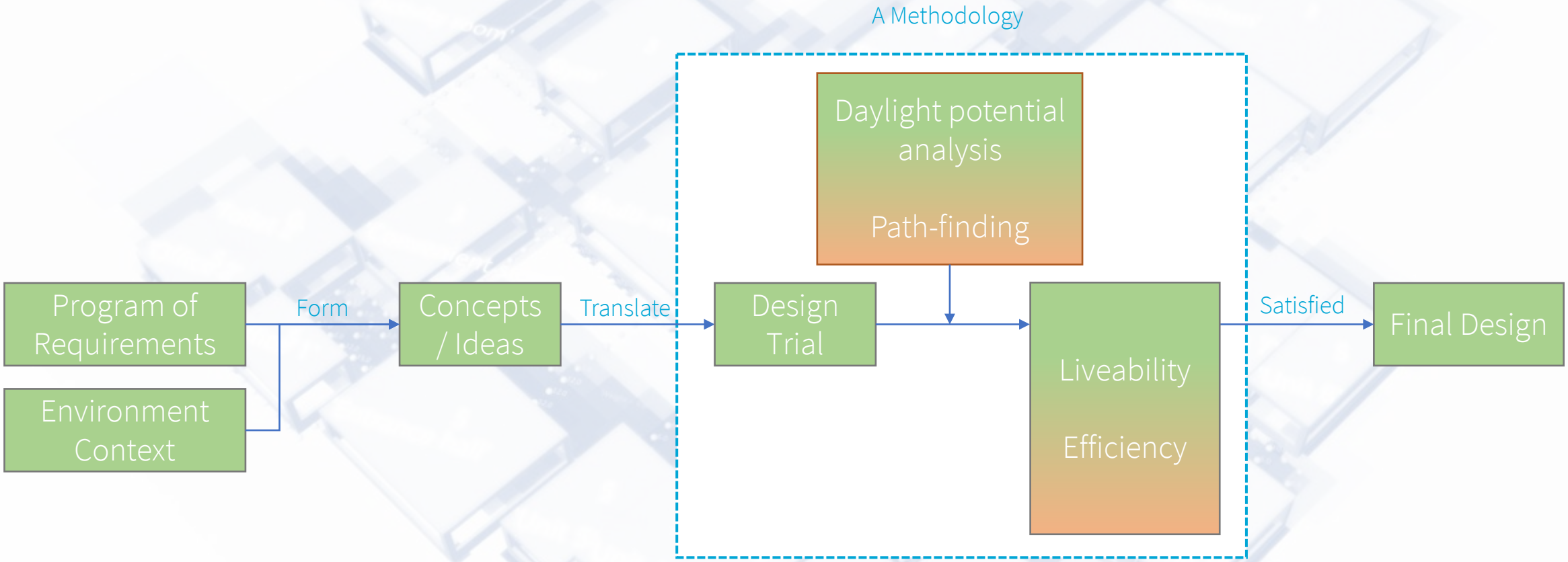
1.4 Scope



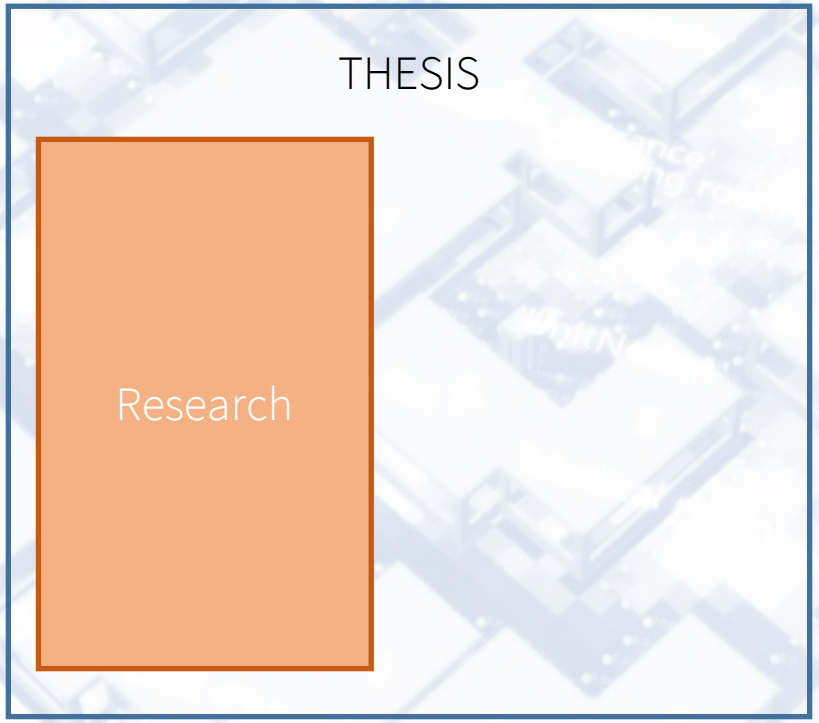
1.4 Scope


A Methodology
Program of Requirements
Environment Context
Form
Concepts / Ideas
Translate
Design Trial
Liveability
Efficiency
Satisfied
Final Design

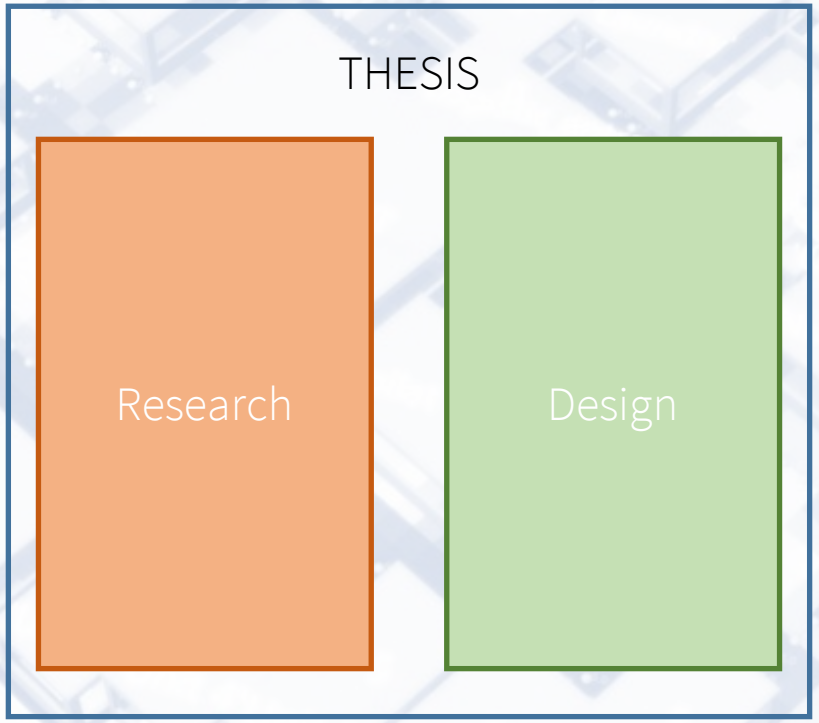
1.4 Scope


A Methodology
Daylight potential analysis
Path-finding
Program of Requirements
Environment Context
Form
Concepts / Ideas
Translate
Design Trial
Liveability
Efficiency
Satisfied
Final Design

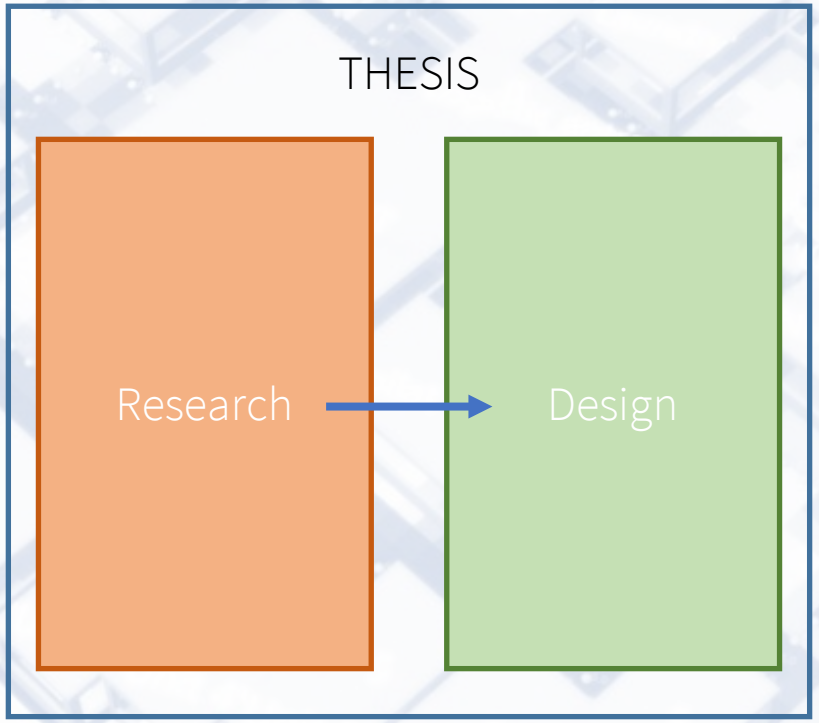
1.4 Scope



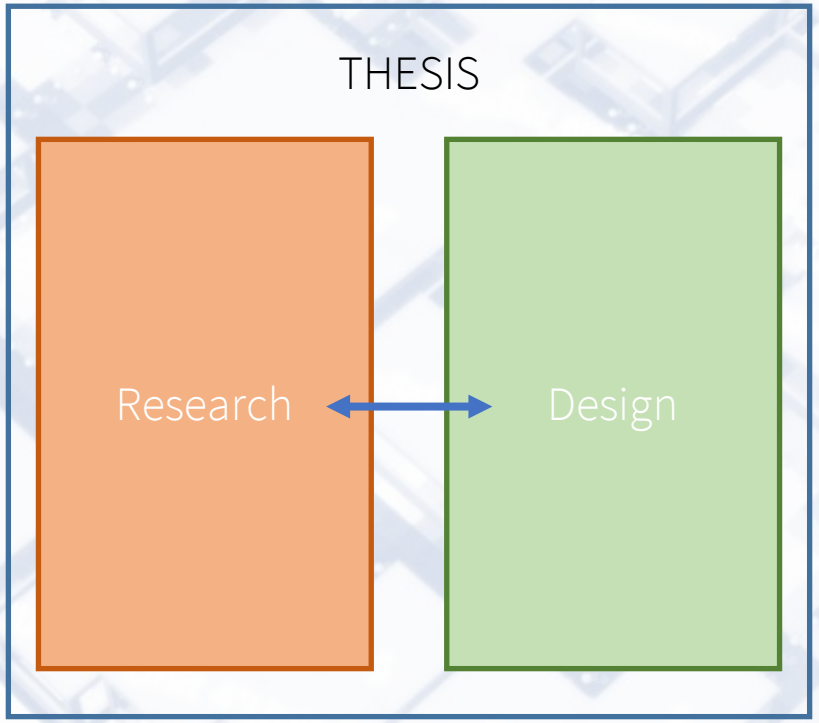
1.4 Scope



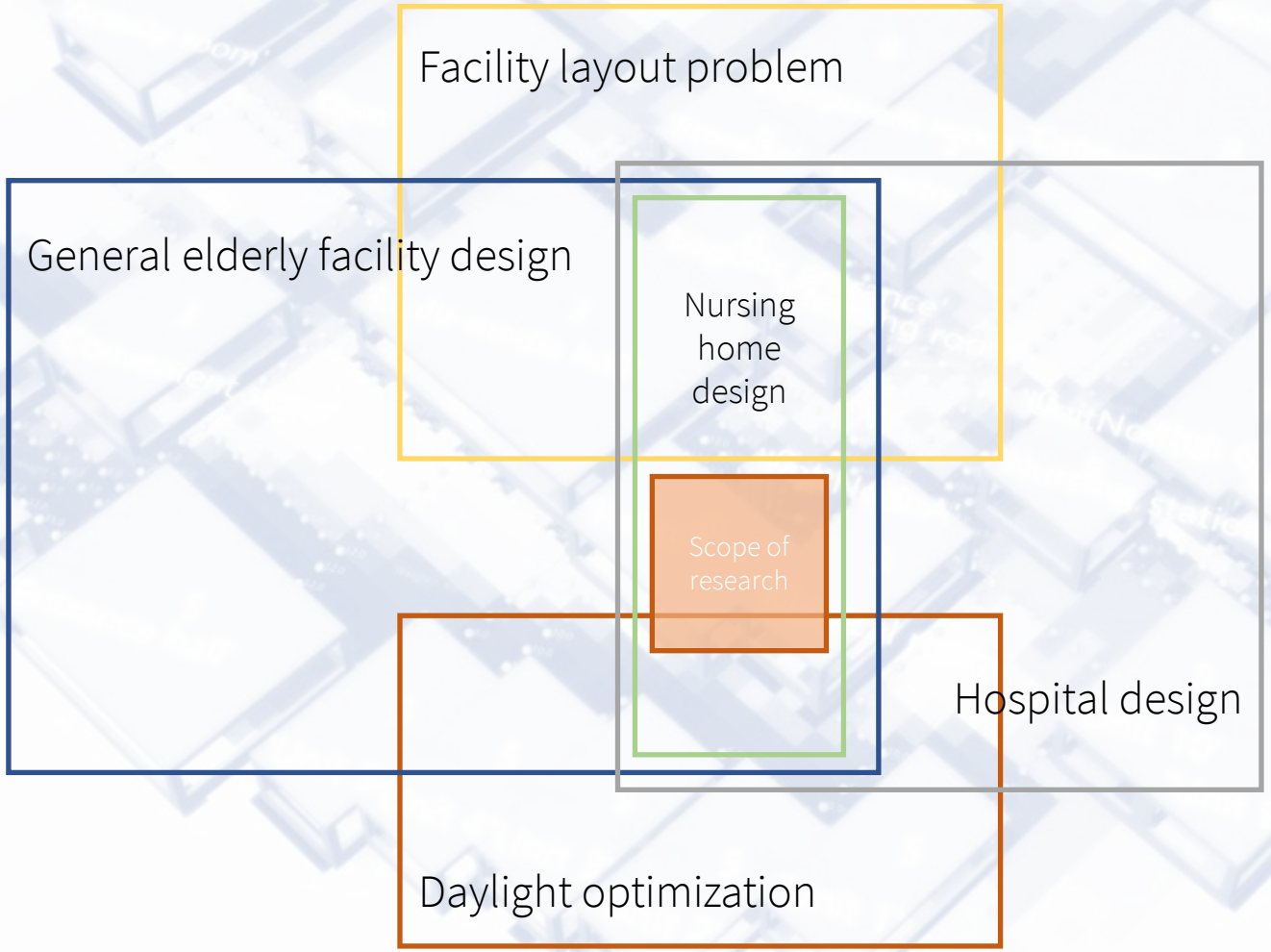
1.4 Scope



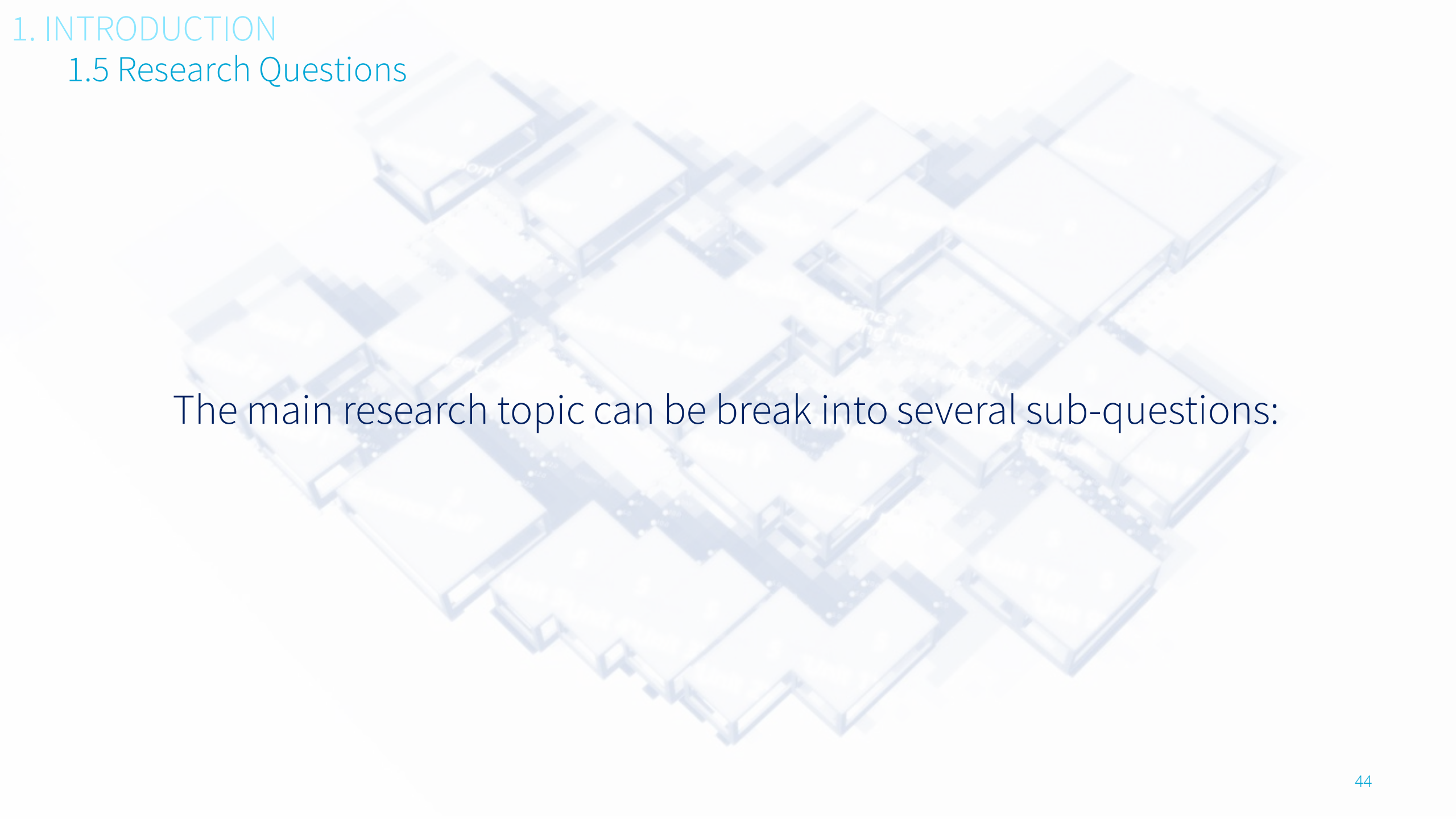
1.4 Scope



1.4 Scope

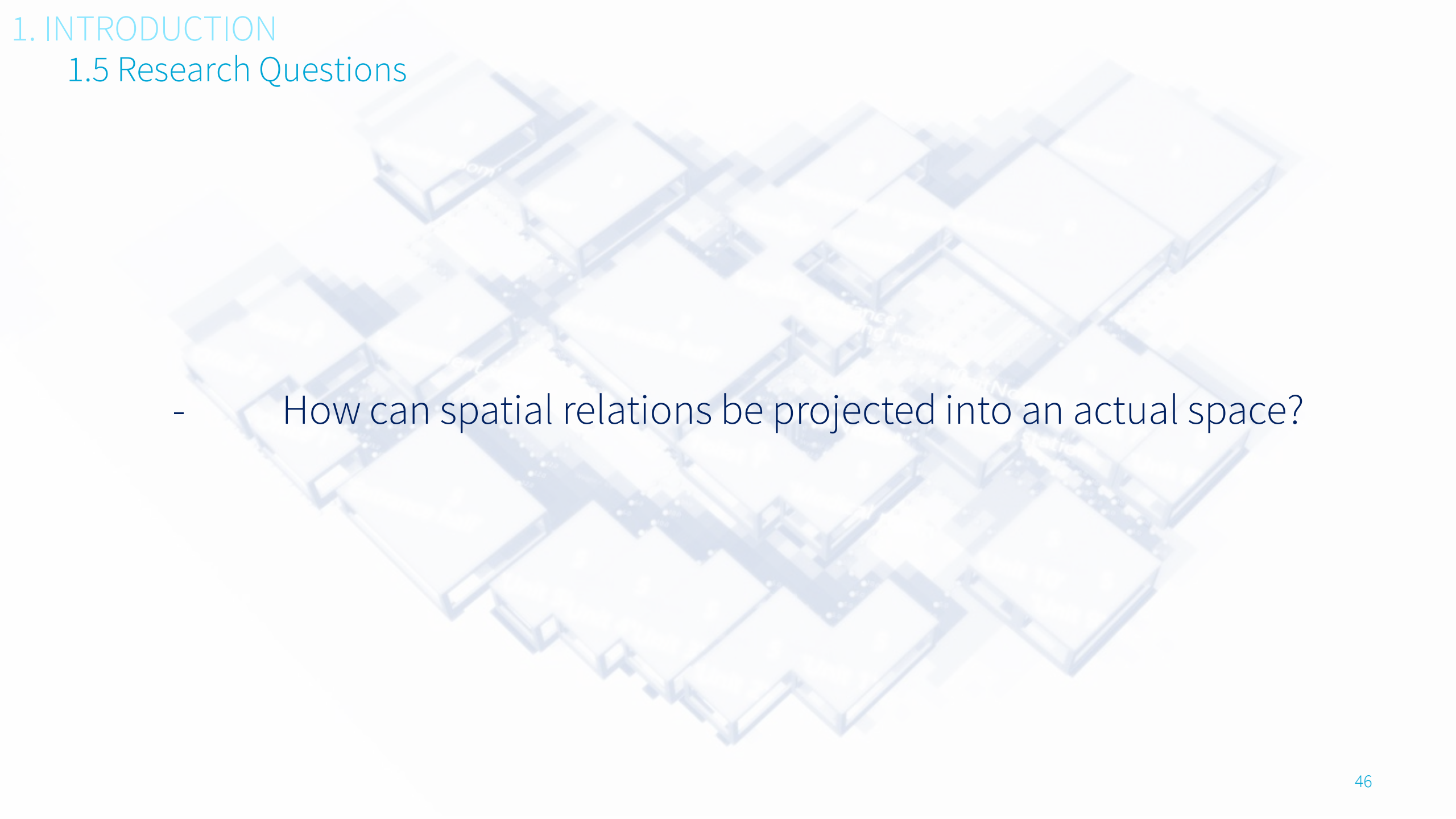

Facility layout problem
General elderly facility design
Nursing home design
Scope of research
Hospital design
Daylight optimization

1.5
RESEARCH QUESTIONS



The main research topic can be break into several sub-questions:

- How should the program of requirements be modelled into spatial relations?



- How can spatial relations be projected into an actual space?

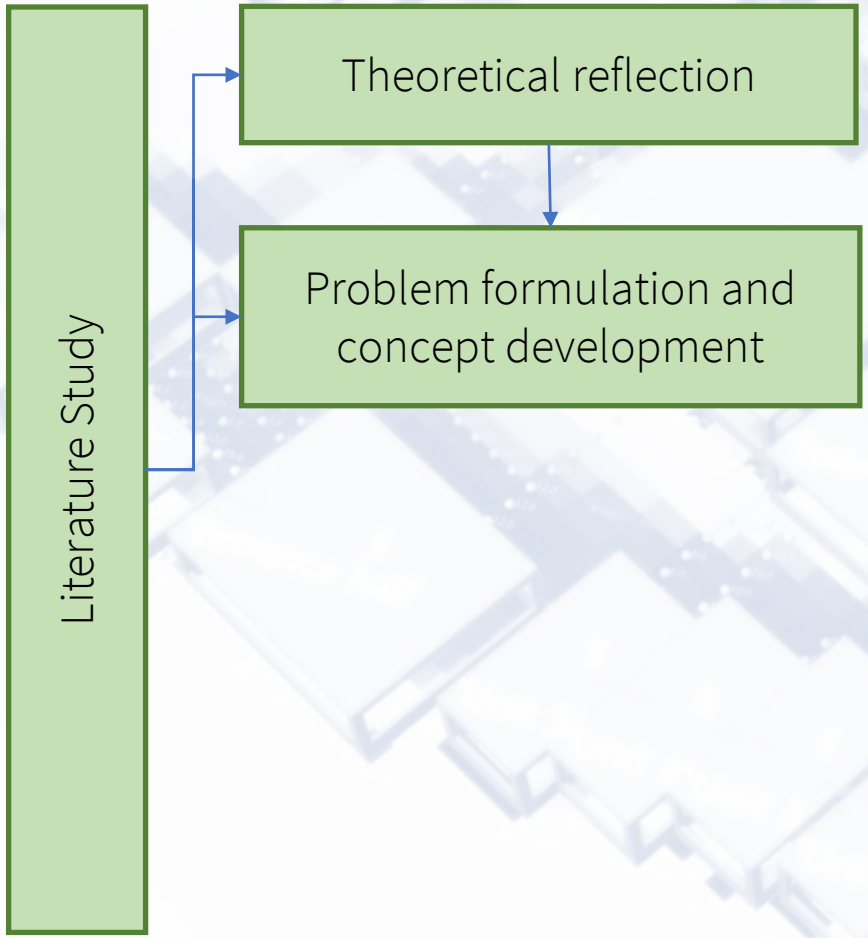
- How should the relations among evaluating circulations, walking distance and daylight hour be? In another word, what is the hierarchy?

- How should the designer interact with the computational process? What is his/her role?

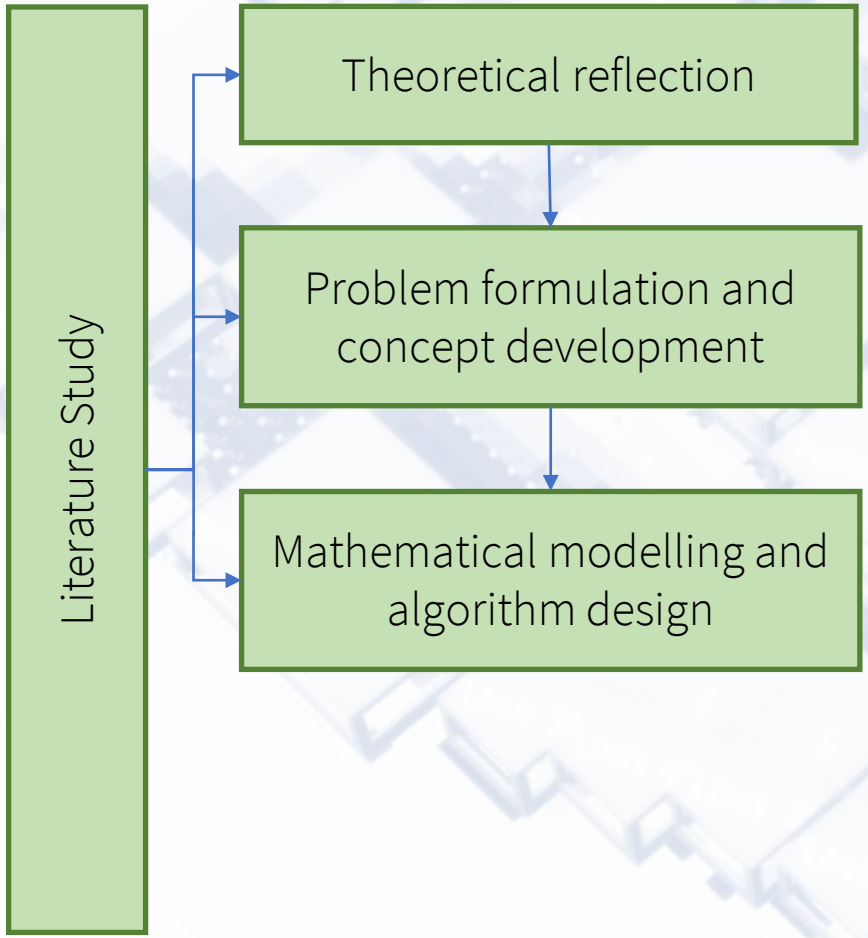
1.6
APPROACH AND METHODOLOGY

1.6 Approach And Methodology

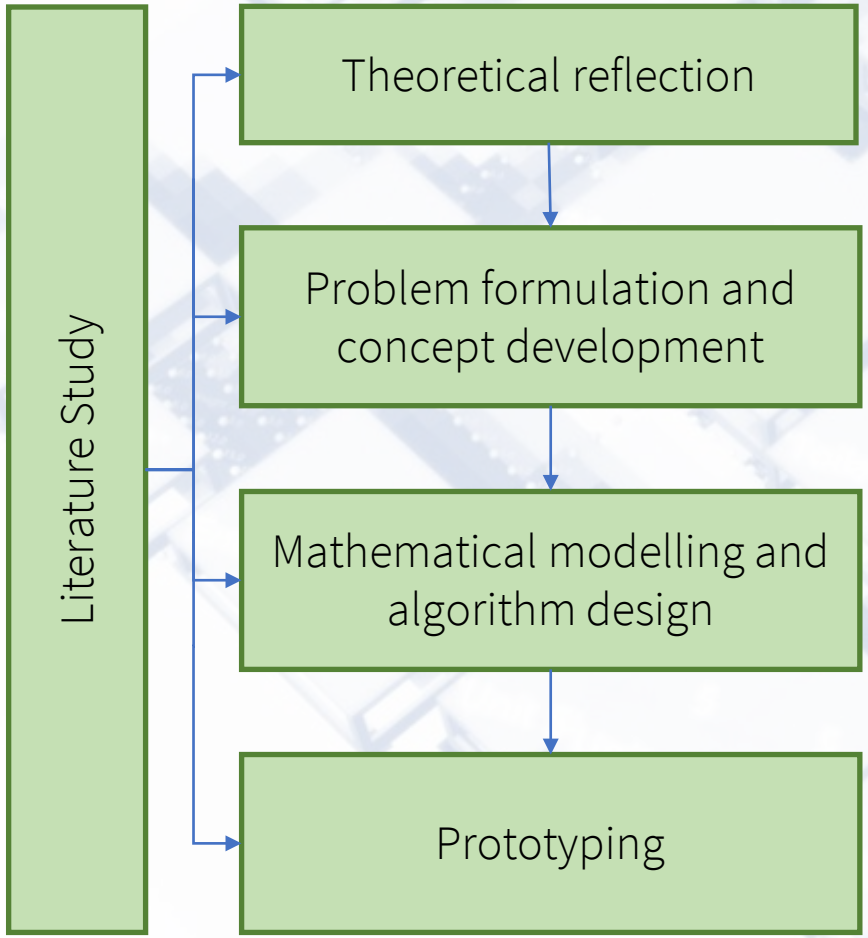
1.6 Approach And Methodology



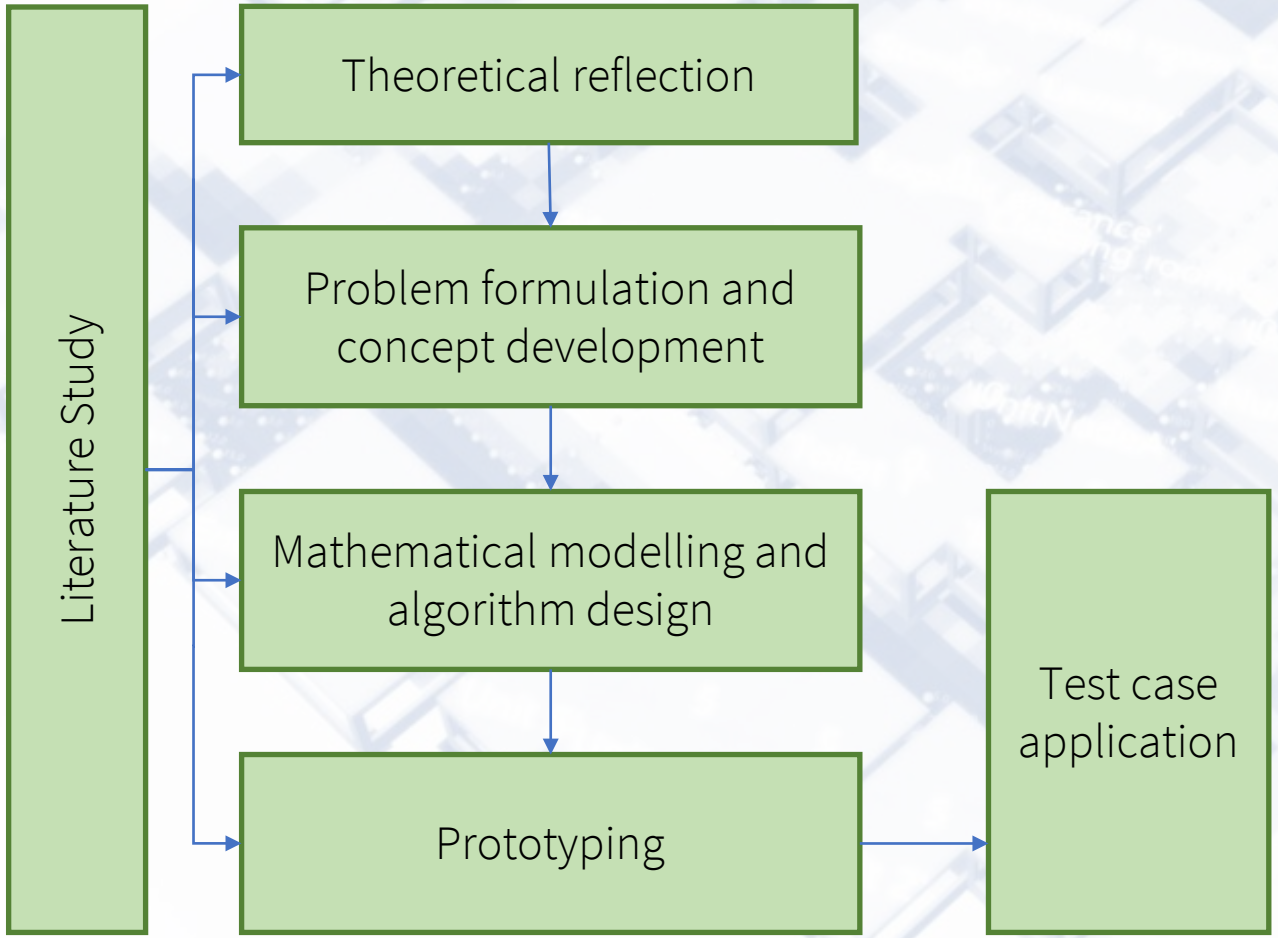
1.6 Approach And Methodology



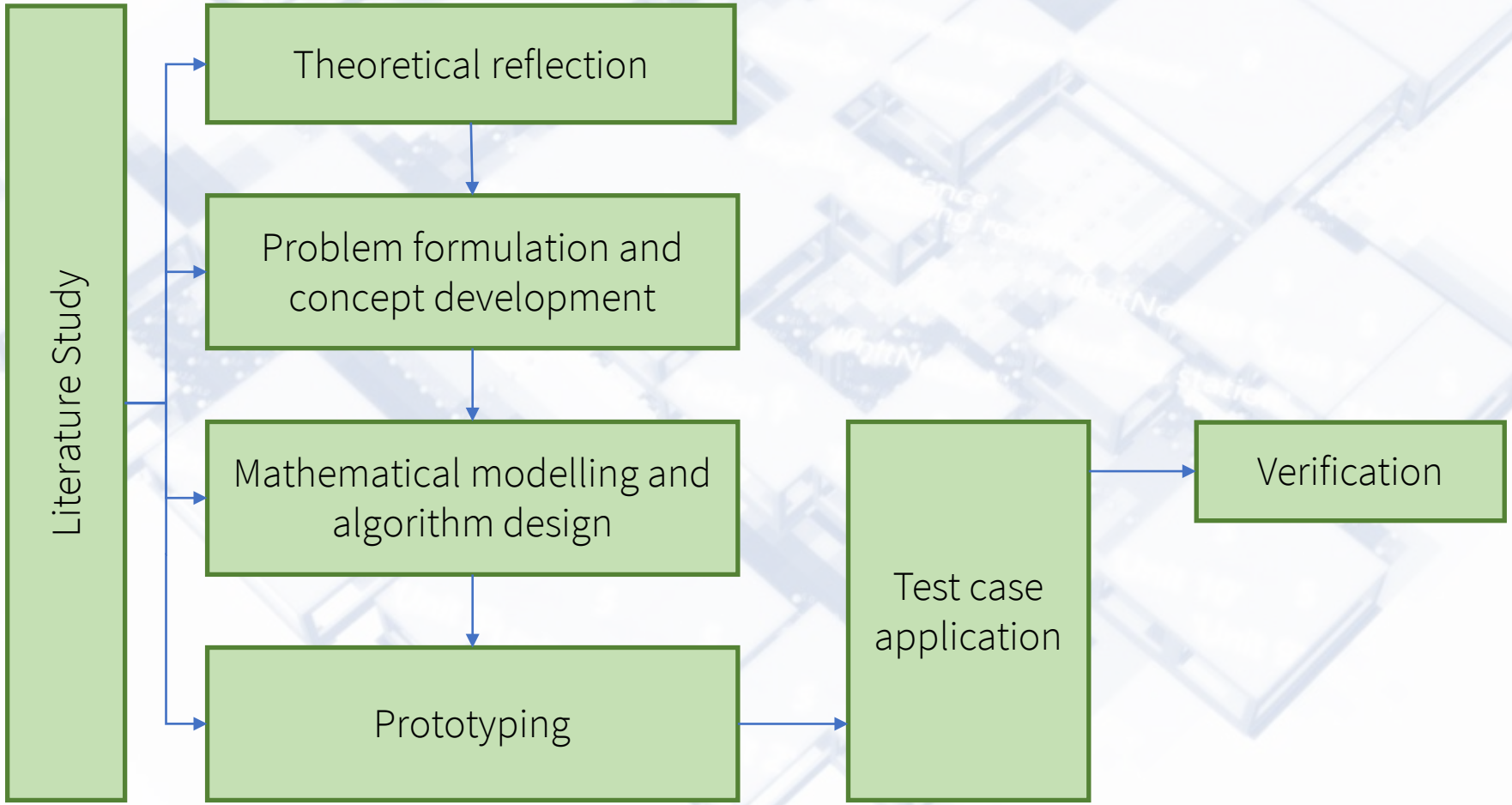
1.6 Approach And Methodology



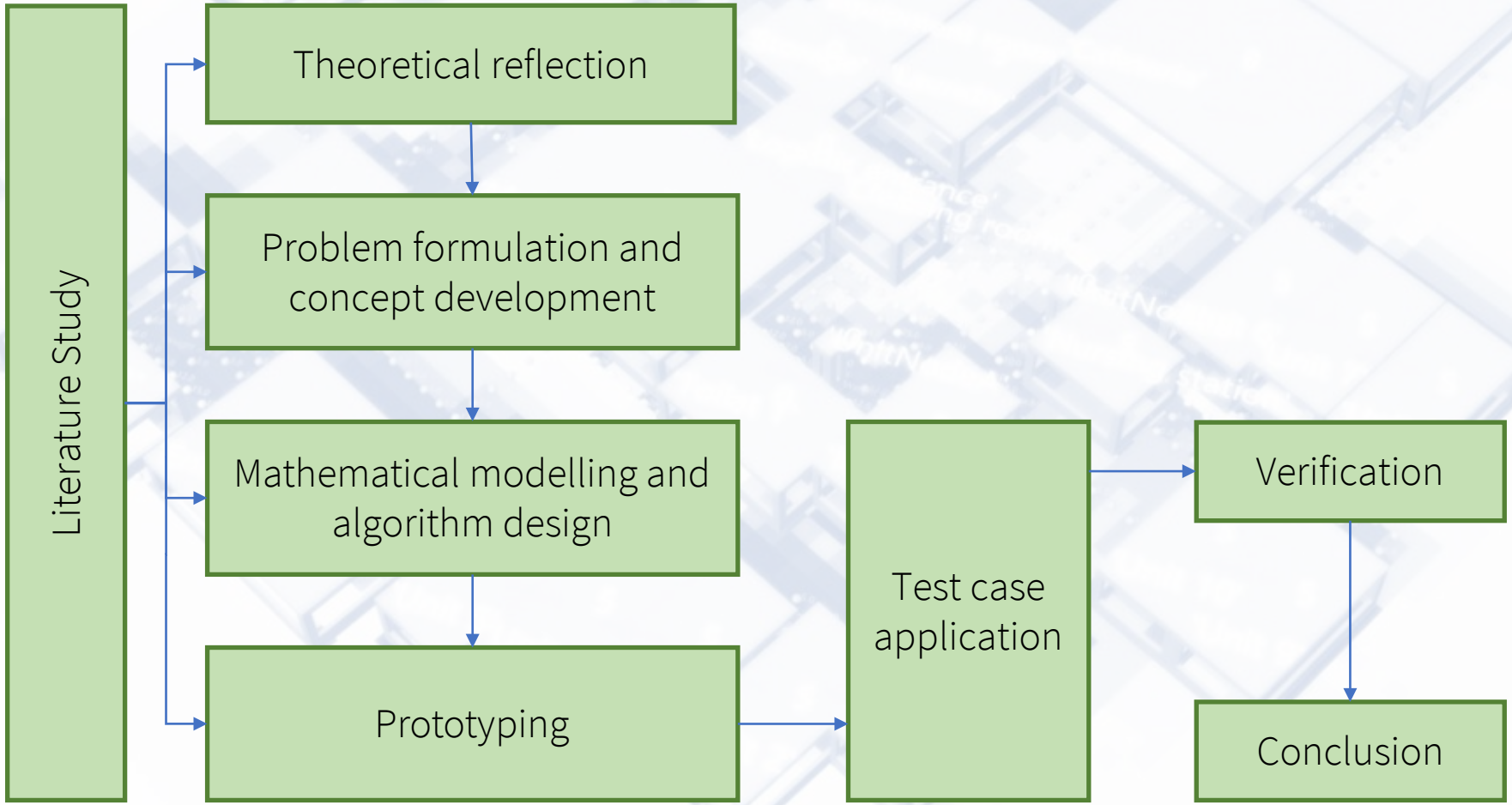
1.6 Approach And Methodology



1.6 Approach And Methodology



1.6 Approach And Methodology



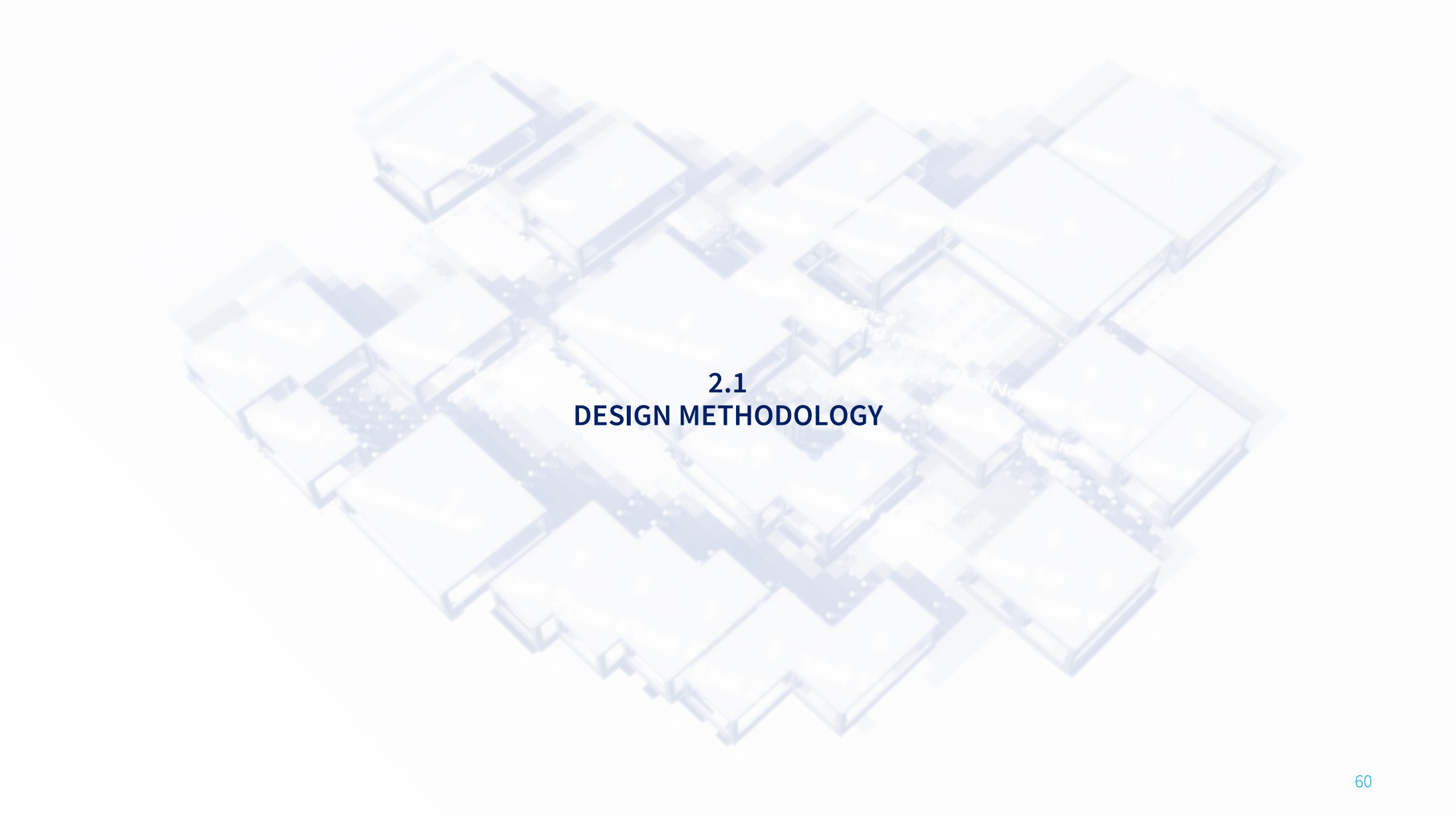
1.7
RESEARCH TOOLS AND TECHNIQUES

1.7 Research Tools And Techniques

Computational and mathematical modelling are the main tools of this research. Related tools used in this thesis are:

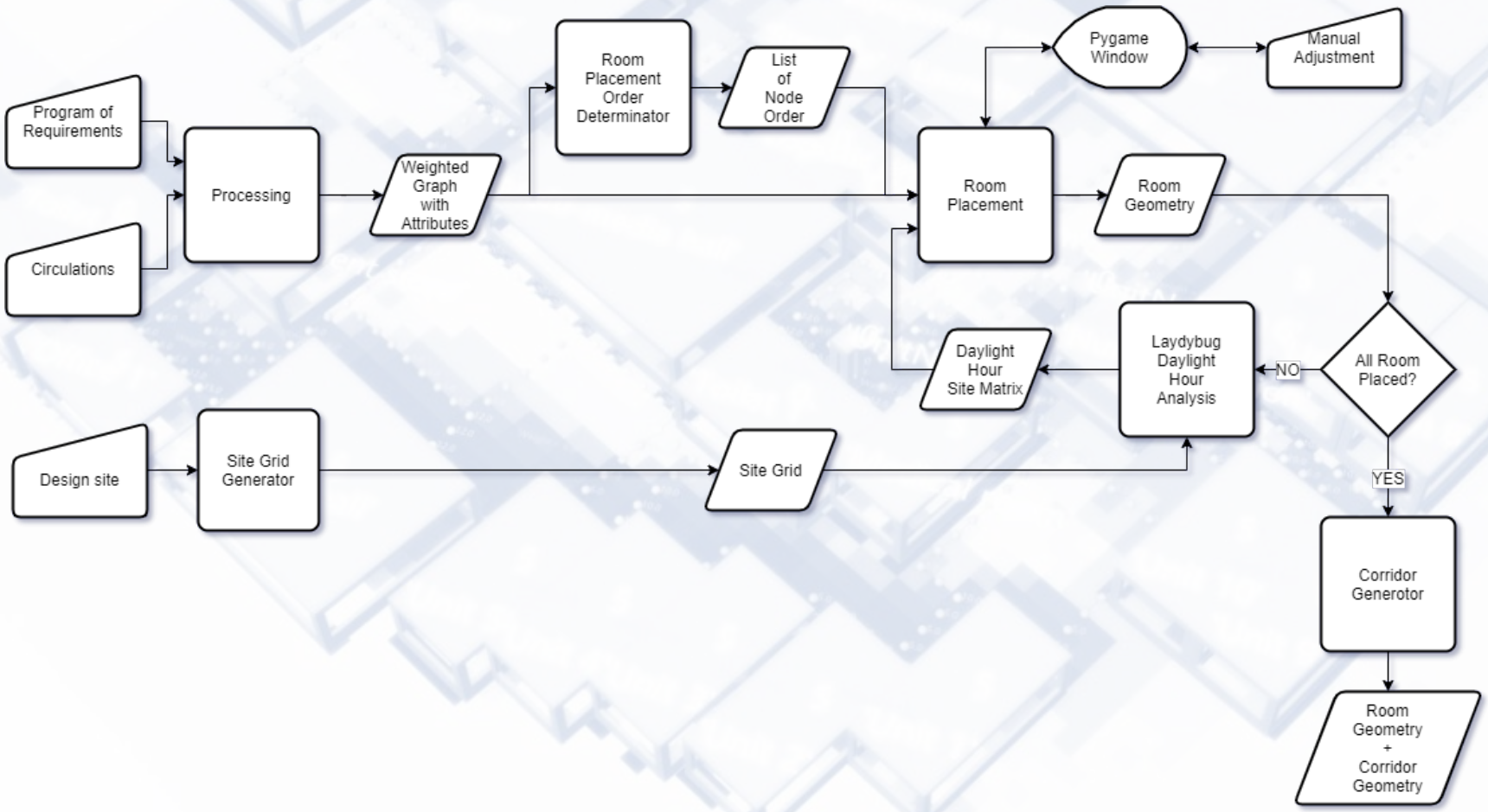
- Python 3.7 with Numpy, Networkx, Scipy, Pymunk and Pygame libraries.
- Use Rhino 6 Grasshopper as the main platform for prototyping and implementation of developed methodologies.
- Ladybug Tools (a tool that supports the evaluation of initial design options through solar radiation studies, view analyses, sunlight-hours modelling, and more. Which can be installed on grasshopper platform.) is used for daylight hour analysis. (Roudsari & Pak, 2013)
- GH_CPython component is used for running CPython in Rhino 6 grasshopper. (Abdelrahman, 2017)

2.
RESEARCH AND METHODS

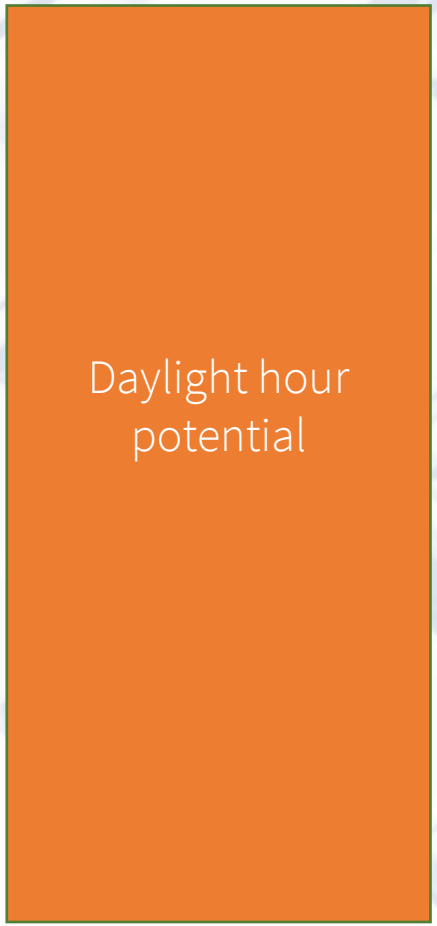


2.1
DESIGN METHODOLOGY

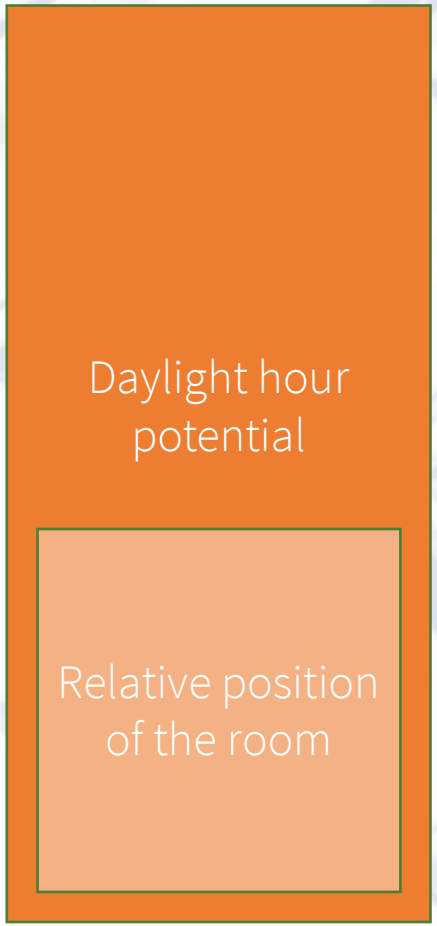
2.1 Design Methodology



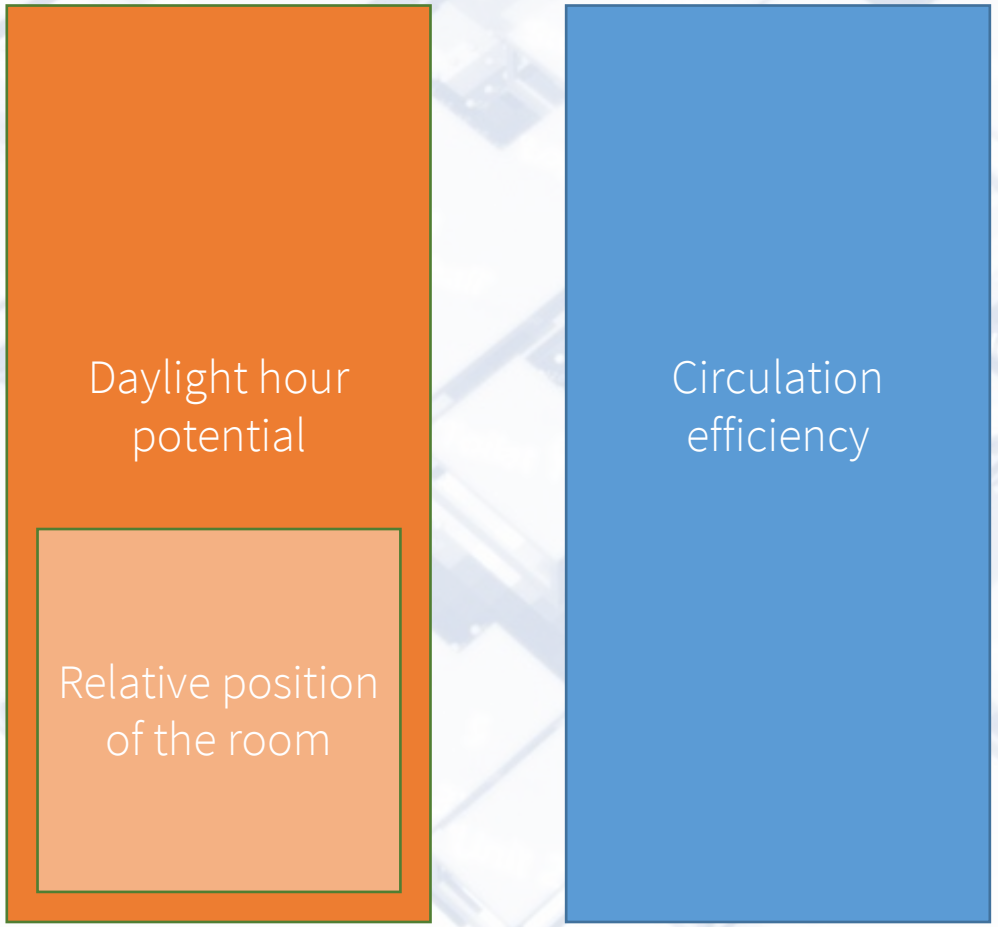
2.1 Design Methodology



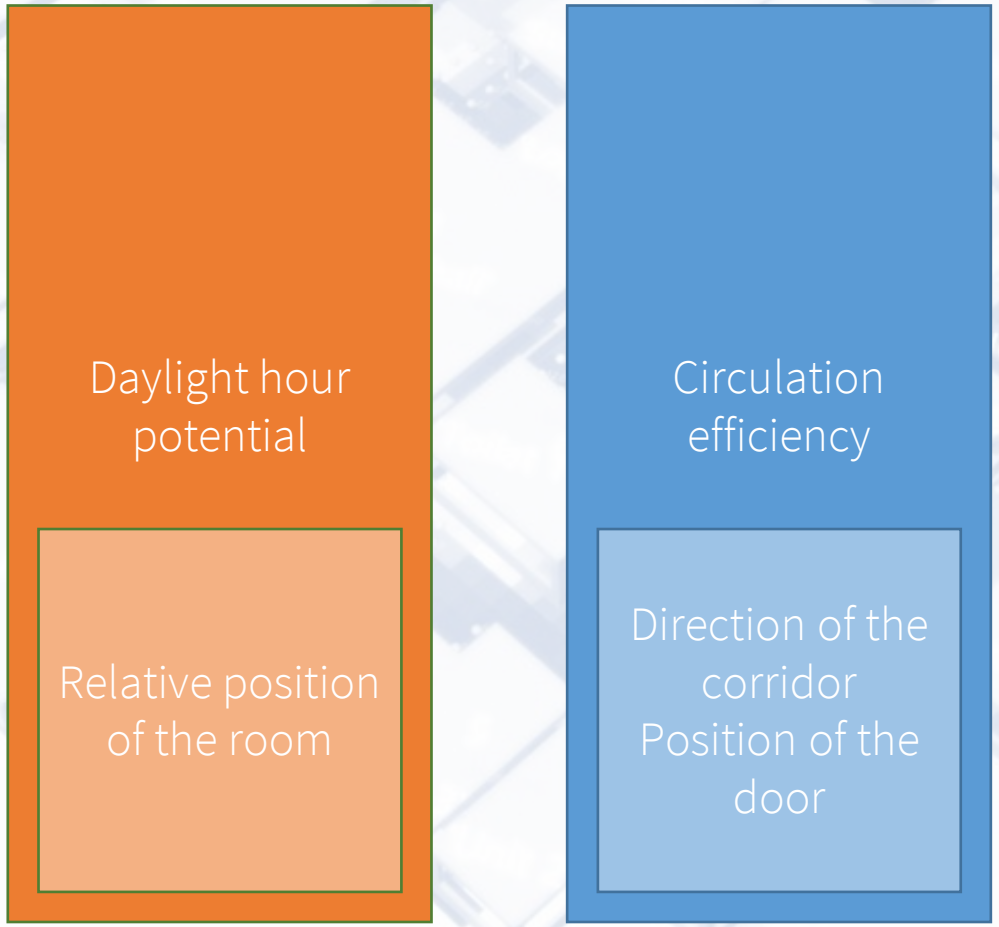
2.1 Design Methodology



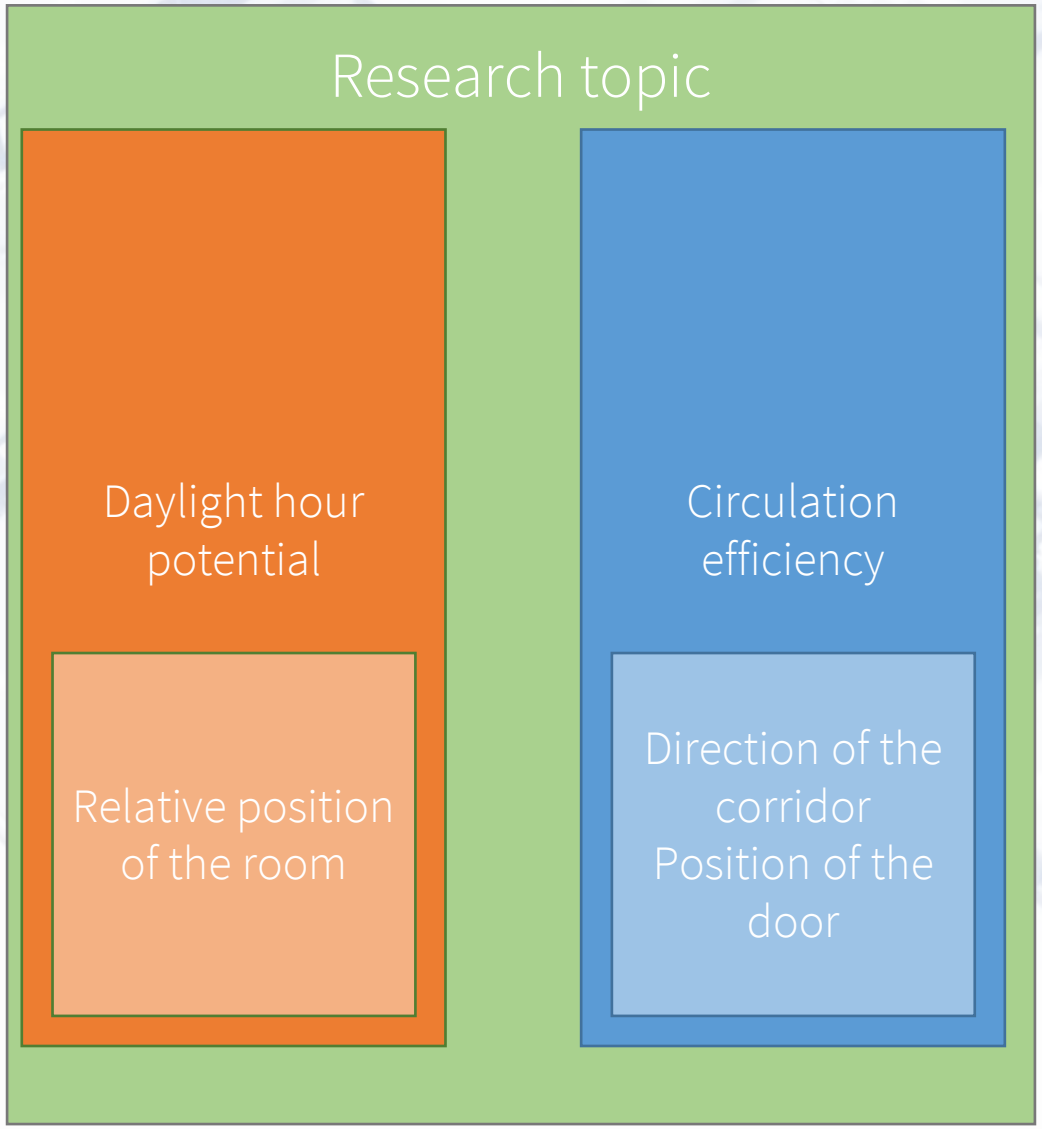
2.1 Design Methodology



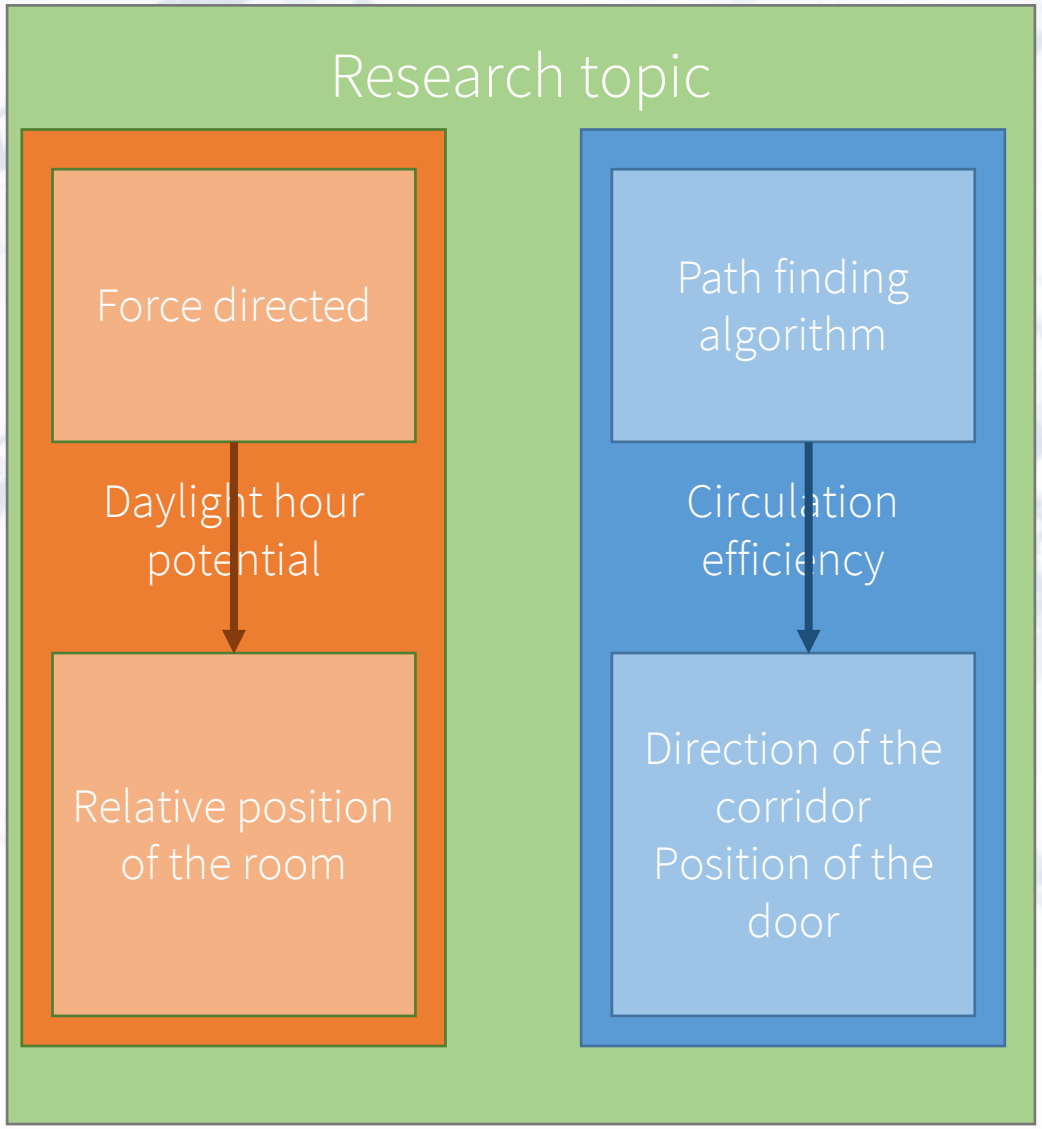
2.1 Design Methodology



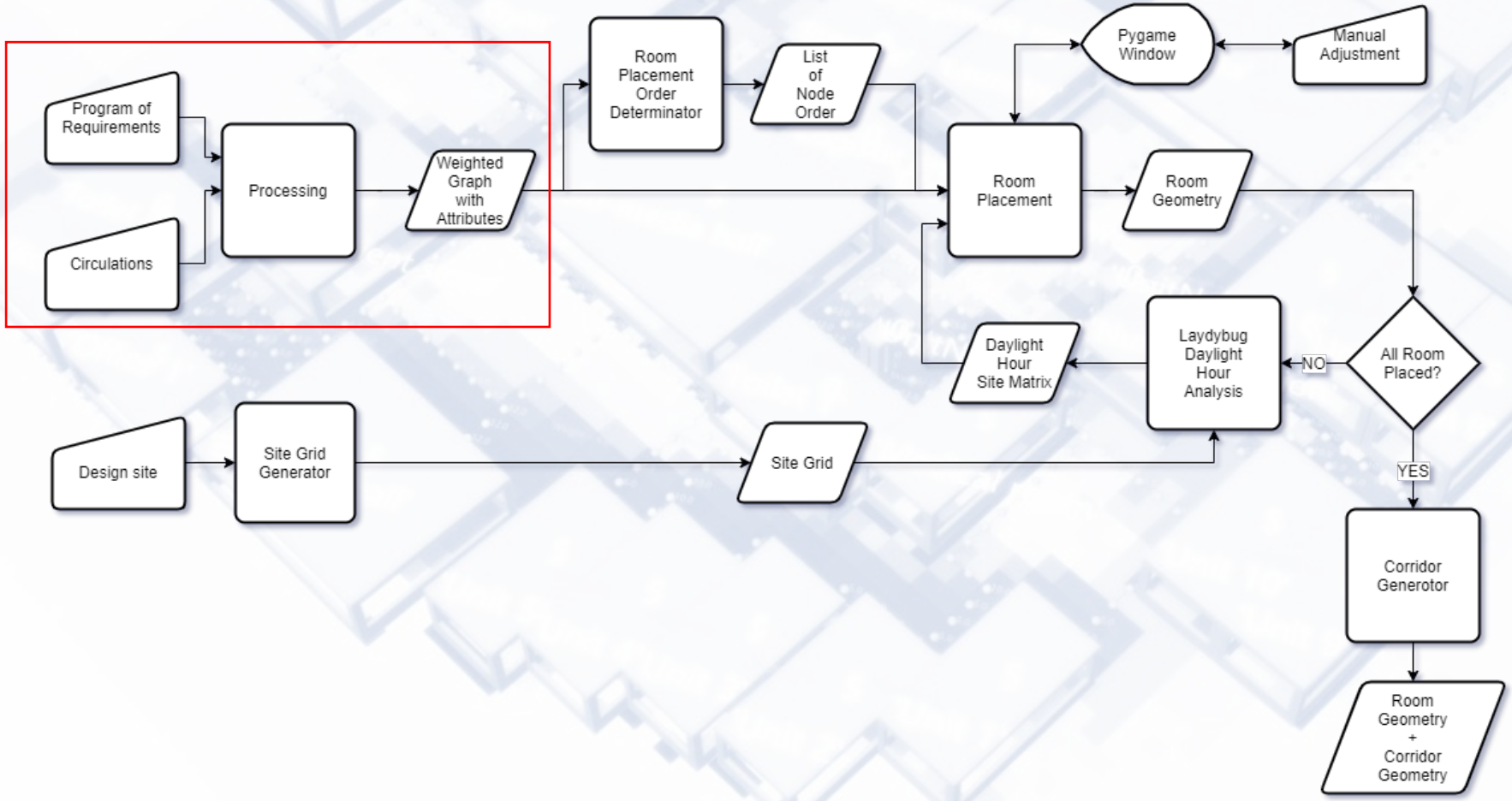
2.1 Design Methodology



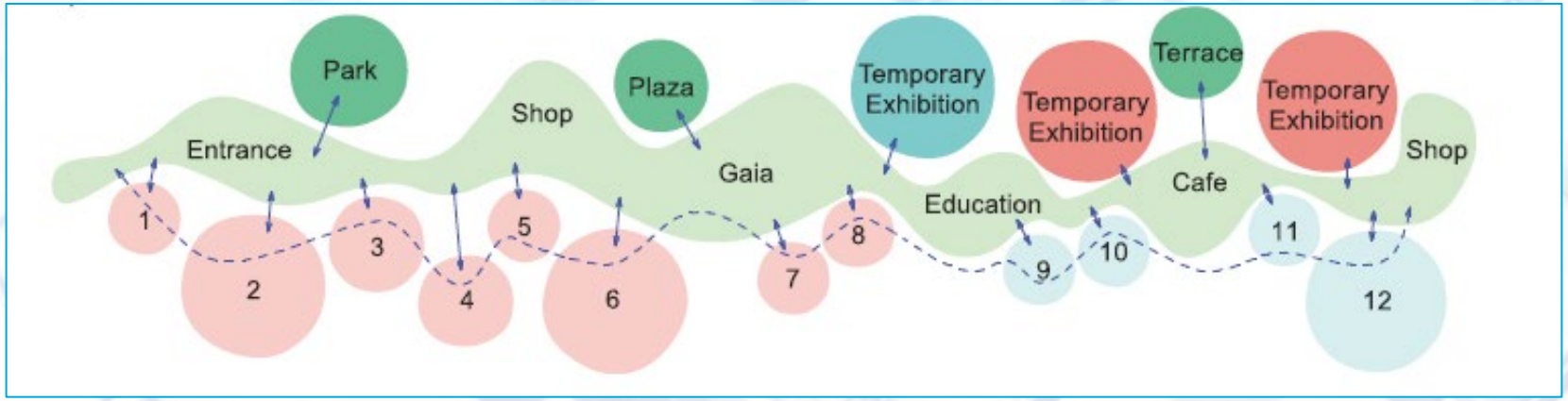
2.1 Design Methodology



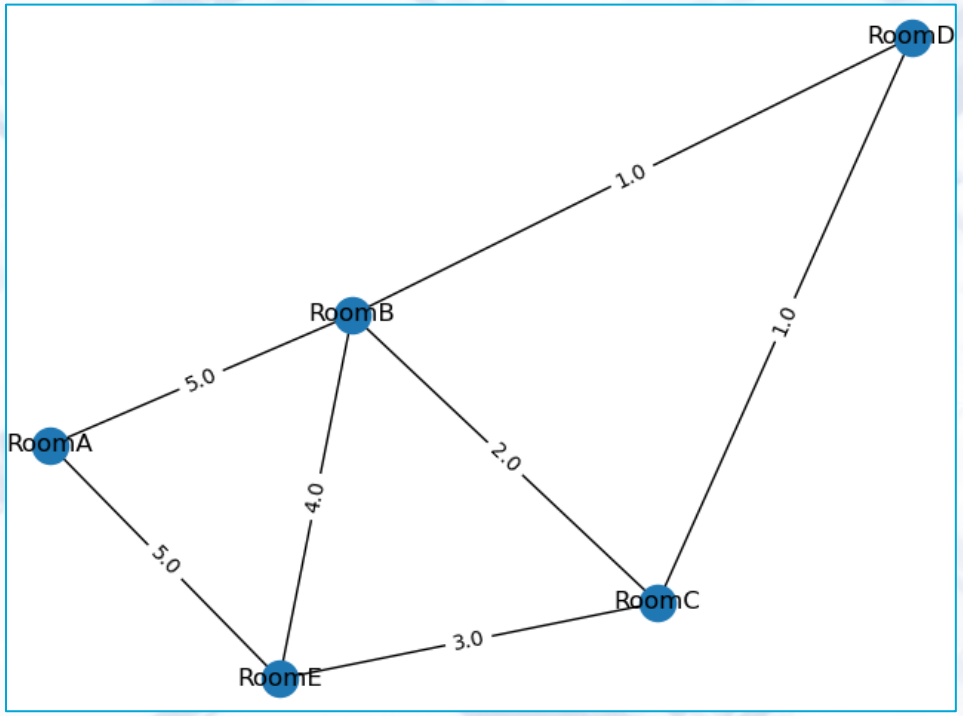
2.2 Processing of Program of Requirement



2.2 Processing of Program of Requirement



2.2 Processing of Program of Requirement



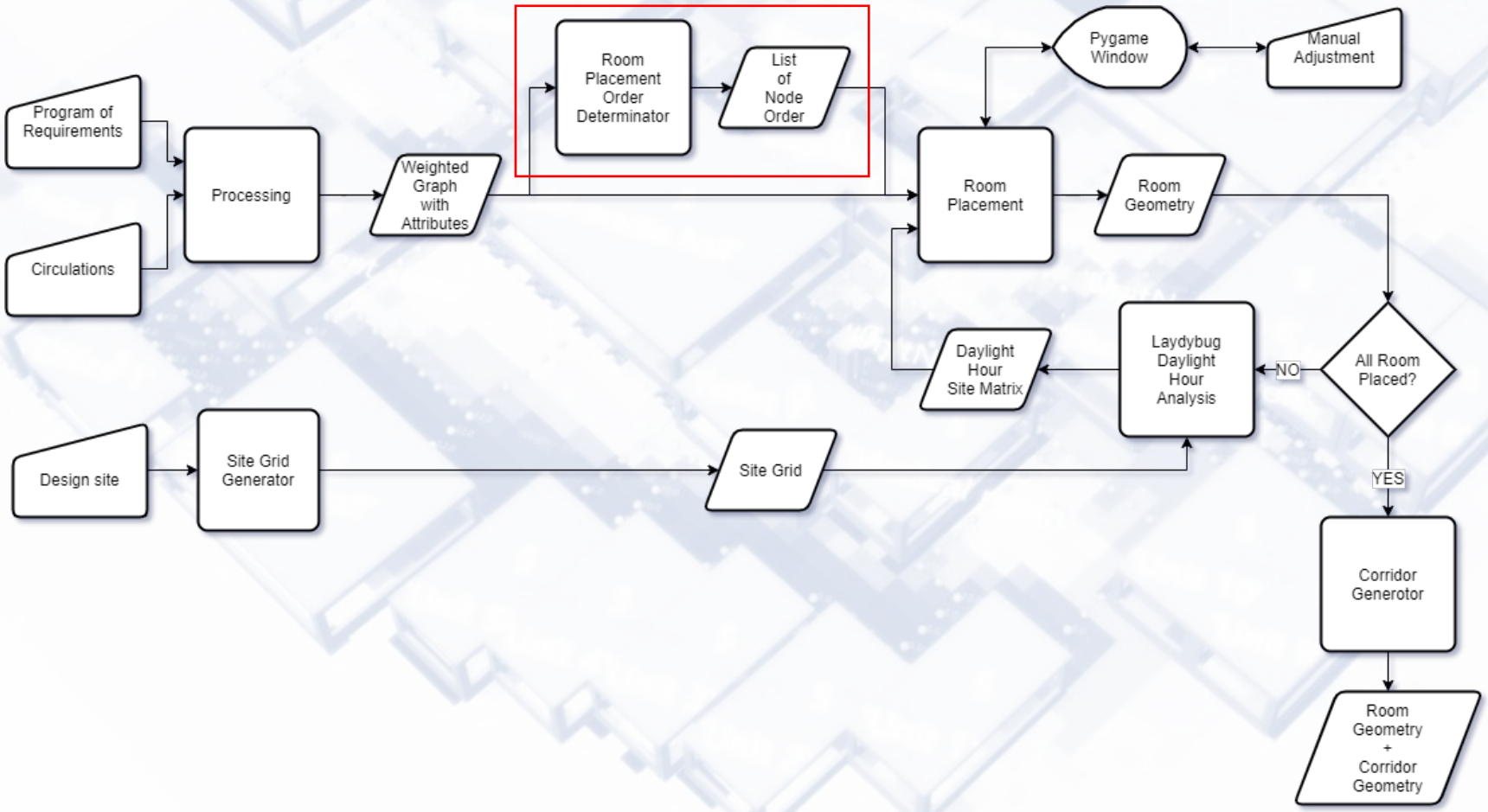
Input:

```
rooms = ["RoomA", "RoomB", "RoomC", "RoomD", "RoomE"]
sizes = [100, 50, 30, 40, 80]
lights = [8, 7, 6, 5, 4]
door_direction = ["N", "none", "none", "none", "W"]
circulations = [["RoomA", "RoomB", "RoomE", "RoomA"],
                ["RoomB", "RoomD", "RoomC", "RoomE", "RoomB"],
                ["RoomA", "RoomB", "RoomC", "RoomE", "RoomA"]]
cir_Weight = [3, 1, 2]
```

Graph generation:

```
G = por_to_graph(rooms, sizes, lights, door_direction, circulations, cir_Weight)
```

2.3 Room Placement Order



2.3 Room Placement Order

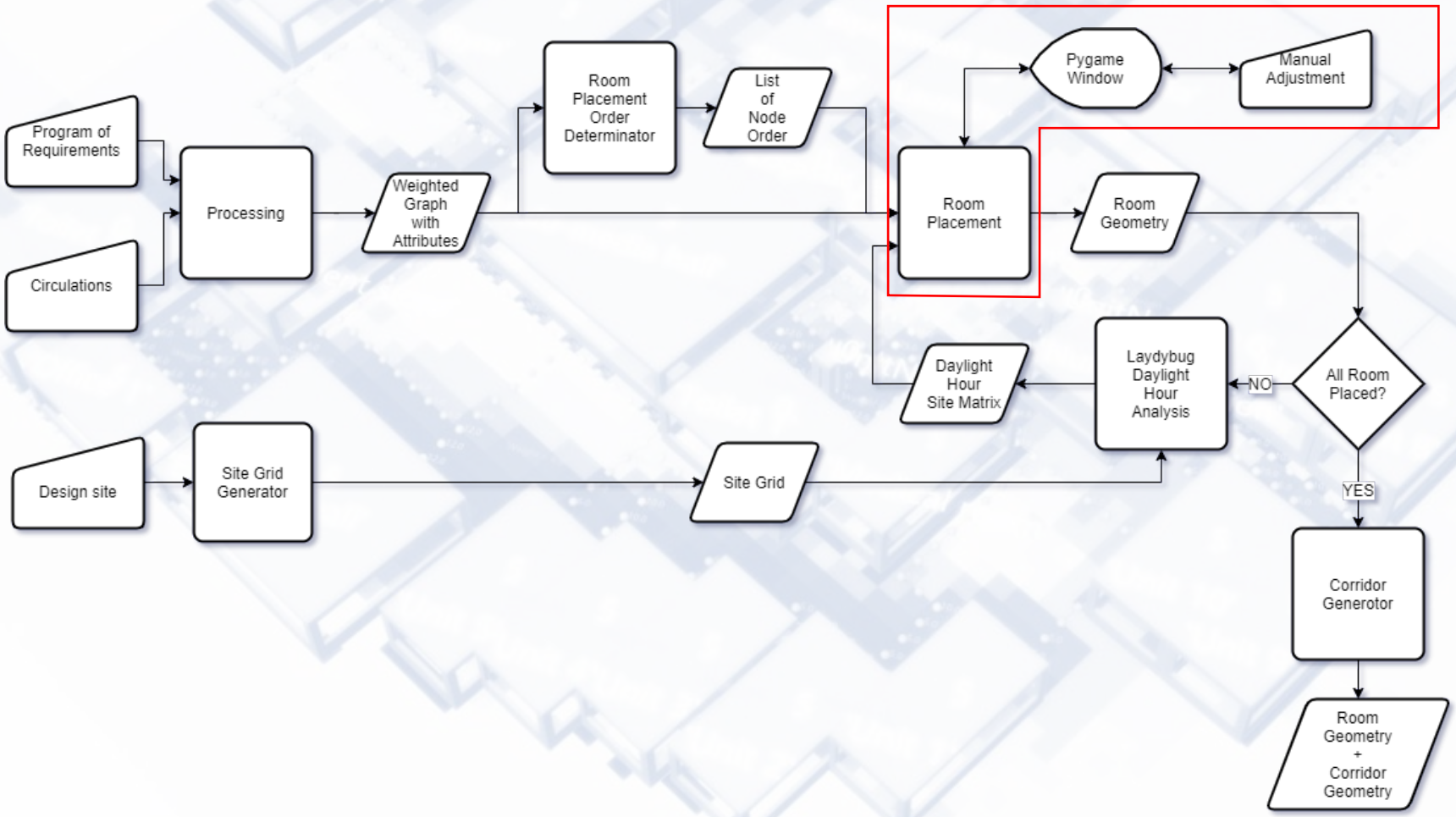
To find the order of placement:

A. First check which room has the highest total weight from connected edges, the highest gets placed first.

B. If the total weight is the same, then check which room has less degree (higher average edge weight), the least gets placed first.

C. If B is still the same, then check which room has larger size, the larger gets placed first.

2.4 Room Placement



2.4 Room Placement

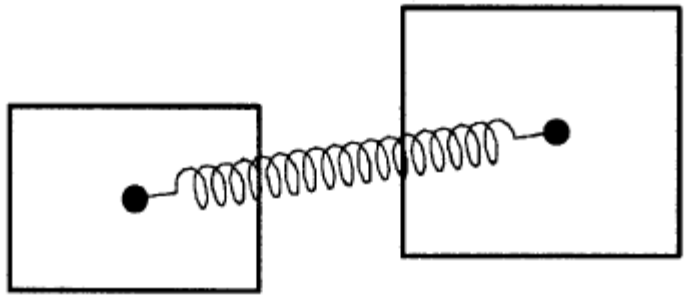


Fig. 6. Adjacency objective.

Given the graph $\Gamma = (V, E)$, $E = (V_i, V_j)$ if V_i is linked to V_j

Output a kissing-disk drawing of graph Γ

- **Do**
  - **For Each** vertex $u \in V$
    - $Resulting\_Forces = \sum \boldsymbol{Attraction\_Forces}(u) + \sum \boldsymbol{Repulsion\_Forces}(u)$
    - u=u moved by the Resulting_Forces
  - **Next**
  - Recompute Continuance_Condition:
    $\forall (i,j) \in E, x_{ij} \neq (R_i + R_j) \mp ErrorTolerance$
  - Iteration_Count=Iteration_Count+1
- **Until** (Continuance_Condition=False Or Iteration_Count>MaximumIterations)

- **Attraction_Forces=** $AF_{ij} = k_a \Delta x_{ij}$, if $(I, j) \in E$,
  - k_a = attraction strength factor,
  - $\Delta x_{ij} = Distance\ V_i\ to\ V_j\text{-}RestLength(i,j)$
  - $RestLength(i,j) = R_i + R_j$
- **Repulsion_Forces=** $RF_{ij} = \frac{k_r}{x_{ii}}$, for all (I, j) $if\ x_{ij} < RestLength(i,j)$
  - k_r = repulsion strength factor
  - $x_{ij} = Distance\ V_i\ to\ V_j$
  - $RestLength(i,j) = R_i + R_j$

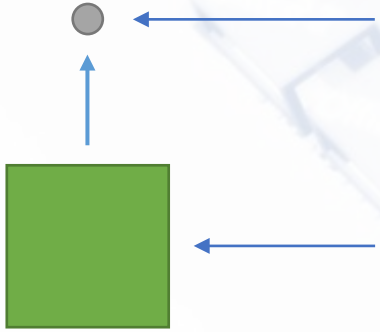
2.4 Room Placement

In room placement phase, the main aim is to configurate the position of each room. The logic is simple:

A. The site environment is analysed with ladybug, and the generated daylight hour matrix of the site is used for room placement.
B. A rectangular room in the list with given dimension or in the dimension of the two closest factors of its size is placed in the site.
C. Then check if there are already neighbours of the room placed in the space. If there is, connect the two rooms with spring with the stiffness scaled by the weight of the edge between the two rooms. It there is not, then place next room in the order.
D. Repeat B and C till all rooms in the order are placed. Designer now can use mouse to drag

2.4 Room Placement

Bodies: points



velocity, position

boundaries, collision

Bodies types:

Dynamic: react to everything
Kinematic: moves only by setting
Static: never move

2.4 Room Placement

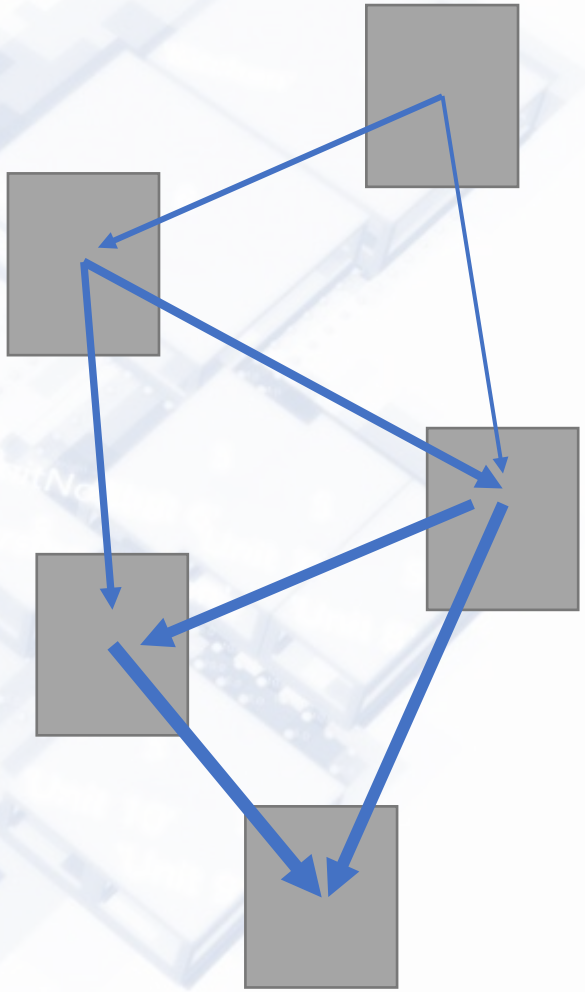
Room placing as dynamic body + spring connection:

Body placing, bottom up, left to right shifting.

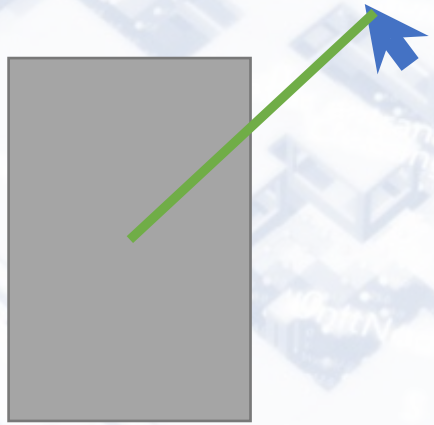
Connection check.

Spring adding:
Spring rest length = S(room_a)^1/2 +S(room_b)^1/2

Spring stiffness = edge*100

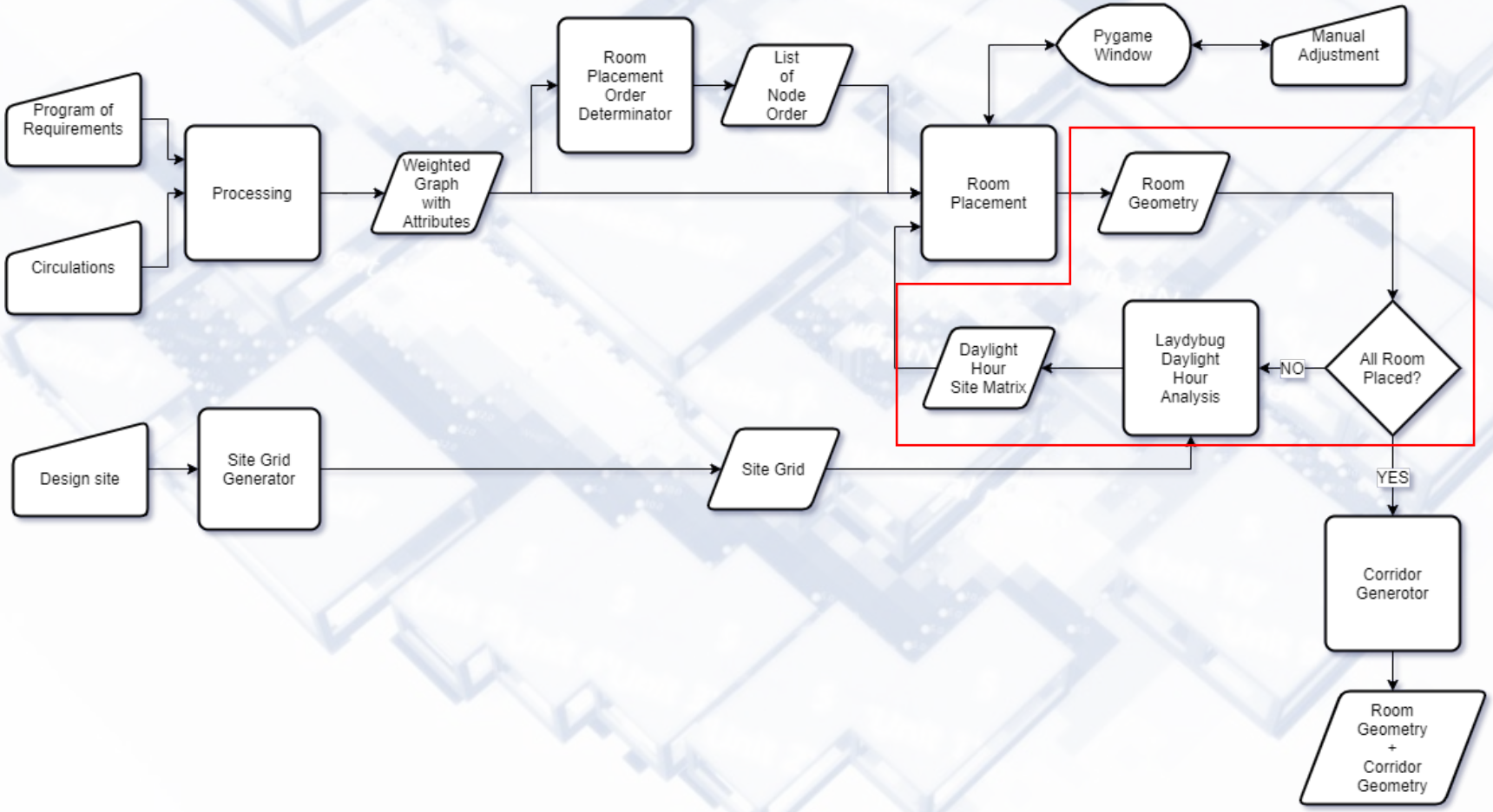


2.4 Room Placement



Also, by giving a body to mouse, it creates joints when the mouse is clicked at a shape of a body, and giving it a huge stiffness to the joint.

2.5 Daylight Hour Optimization

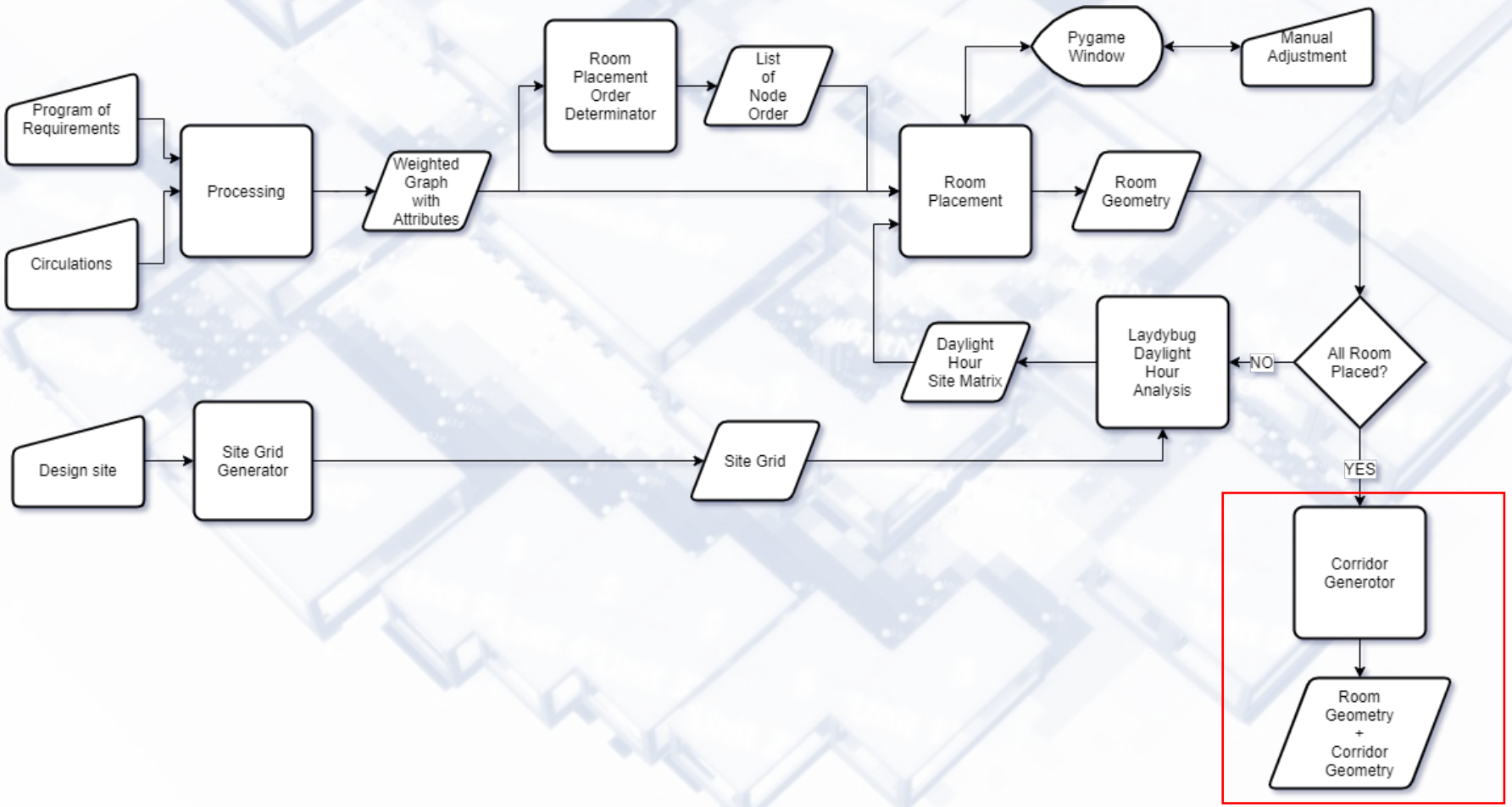


2.5 Daylight Hour Optimization

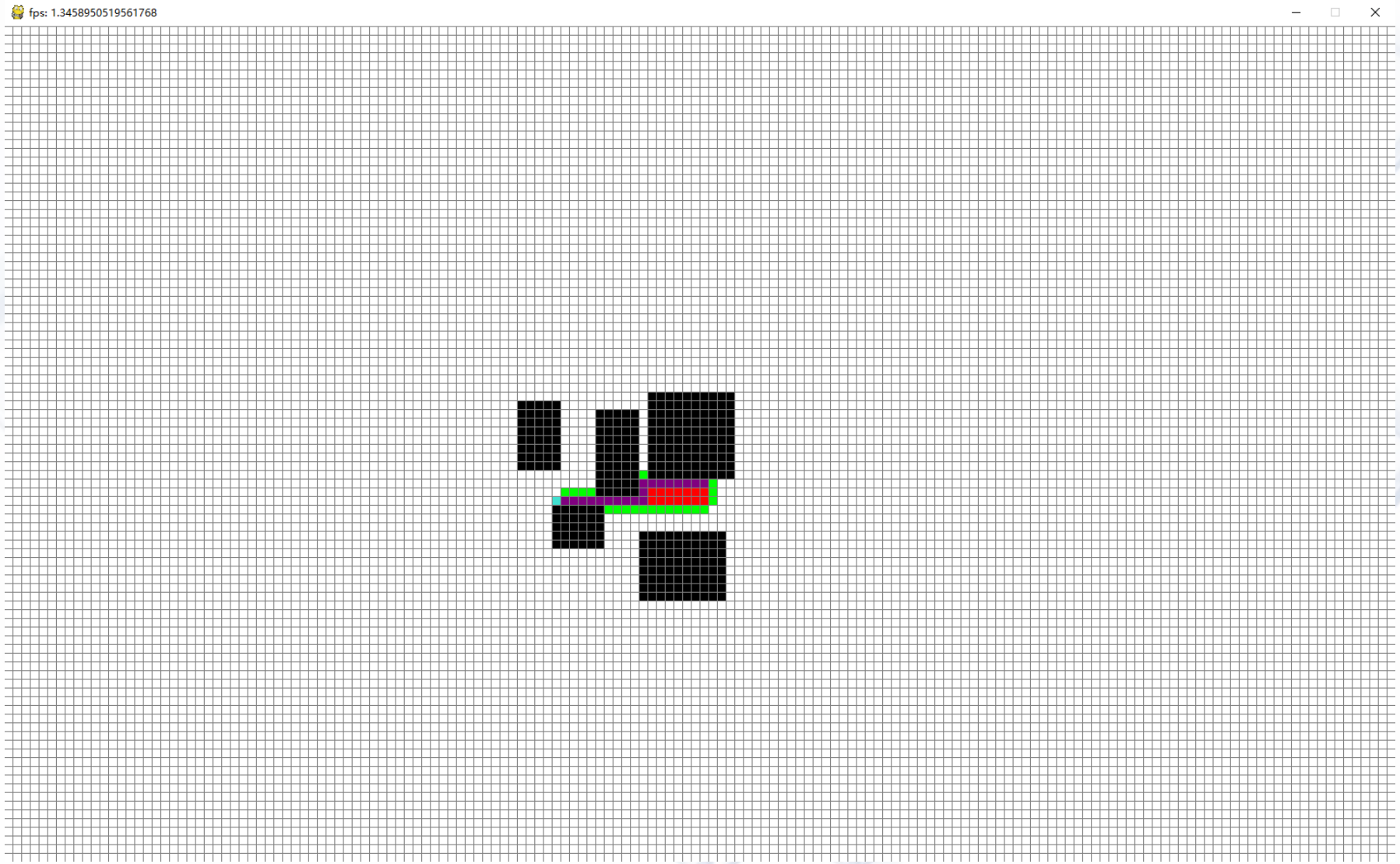
To proceed:

A. Generate first room in Rhino.
B. Analyse the daylight hour of the room, map generated results to Pymunk and Pygame space.
C. Set the analyse room as static body, and the rest of the rooms that does not meet the daylight hour requirements are squeezed out of the range. After all rooms are adjusted and relocated, generate next room in Rhino.
D. Analyse all the rooms like step B and C in the order of placement.
E. Location of rooms can be dragged with mouse after each analysis.

2.6 Path-finding



2.6 Path-finding



2.5 Path-finding

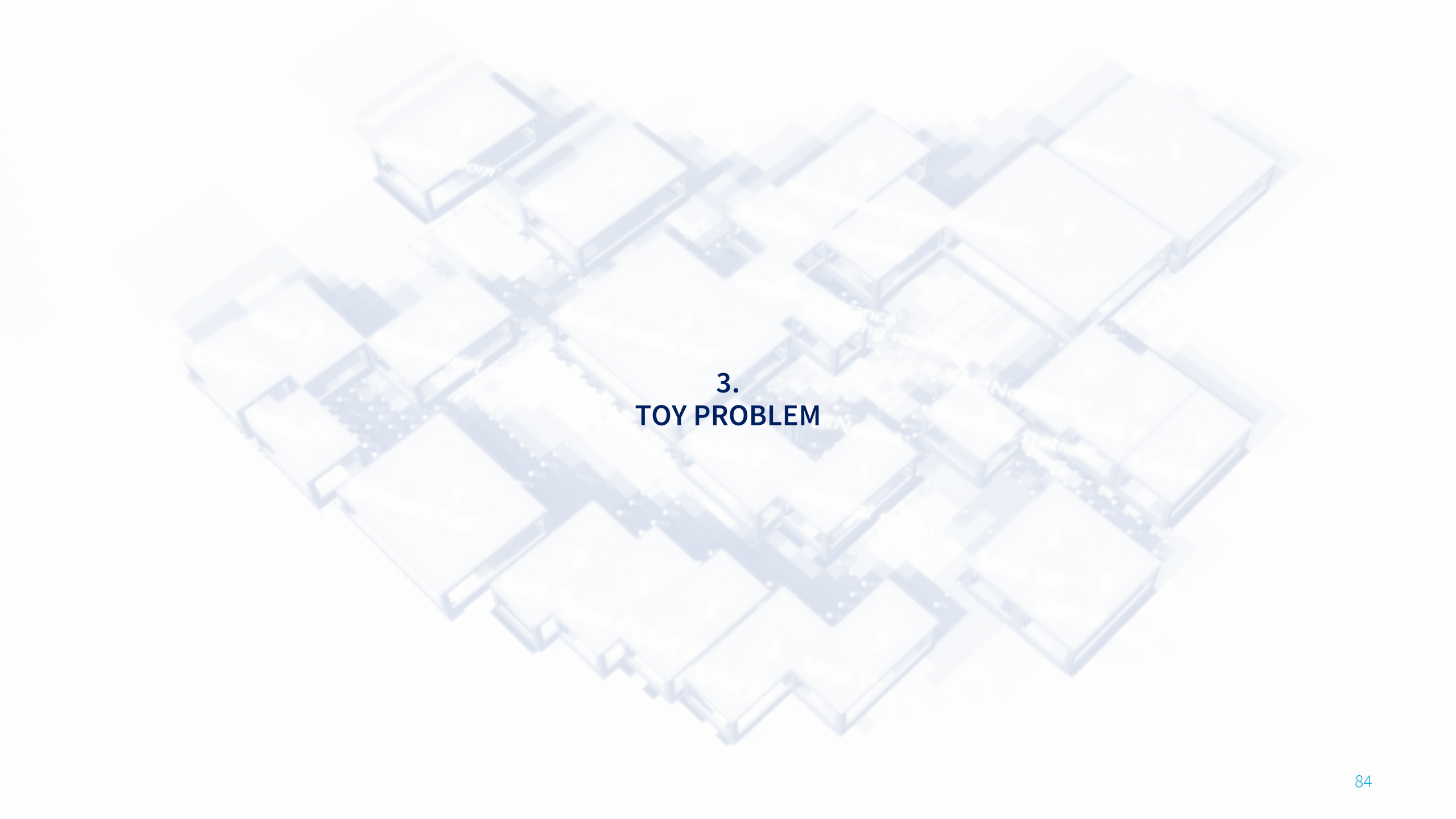
The general process is:

A. Set the grid of the space.

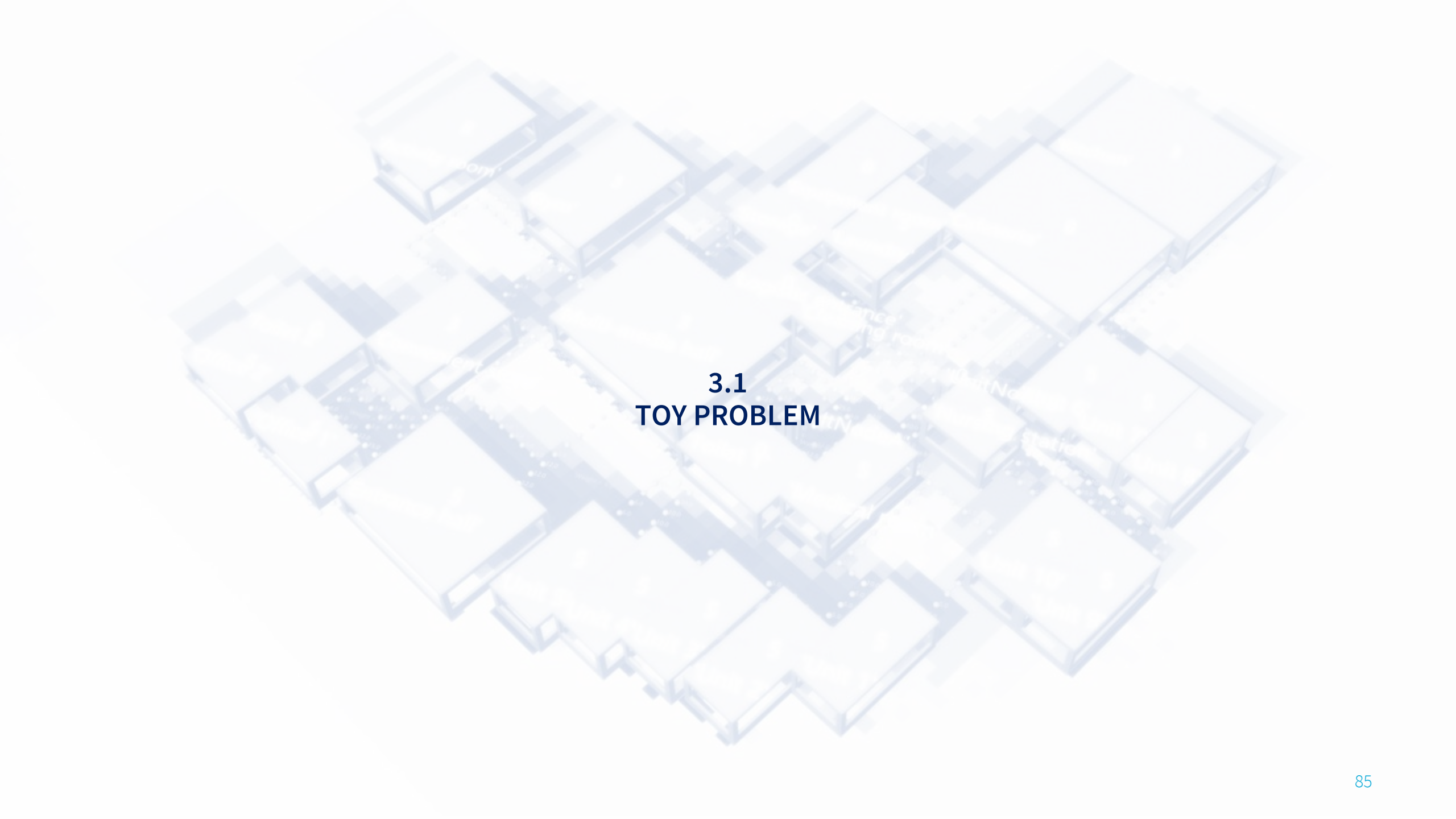
B. Locations of the doors are set on the boundary of rooms according to the program of requirements.

C. Paths are generated between rooms connected with edges by pathfinding algorithm, record the list of nodes of paths.

D. After all paths are generated, generate corridor shapes with the node lists in rhino software.



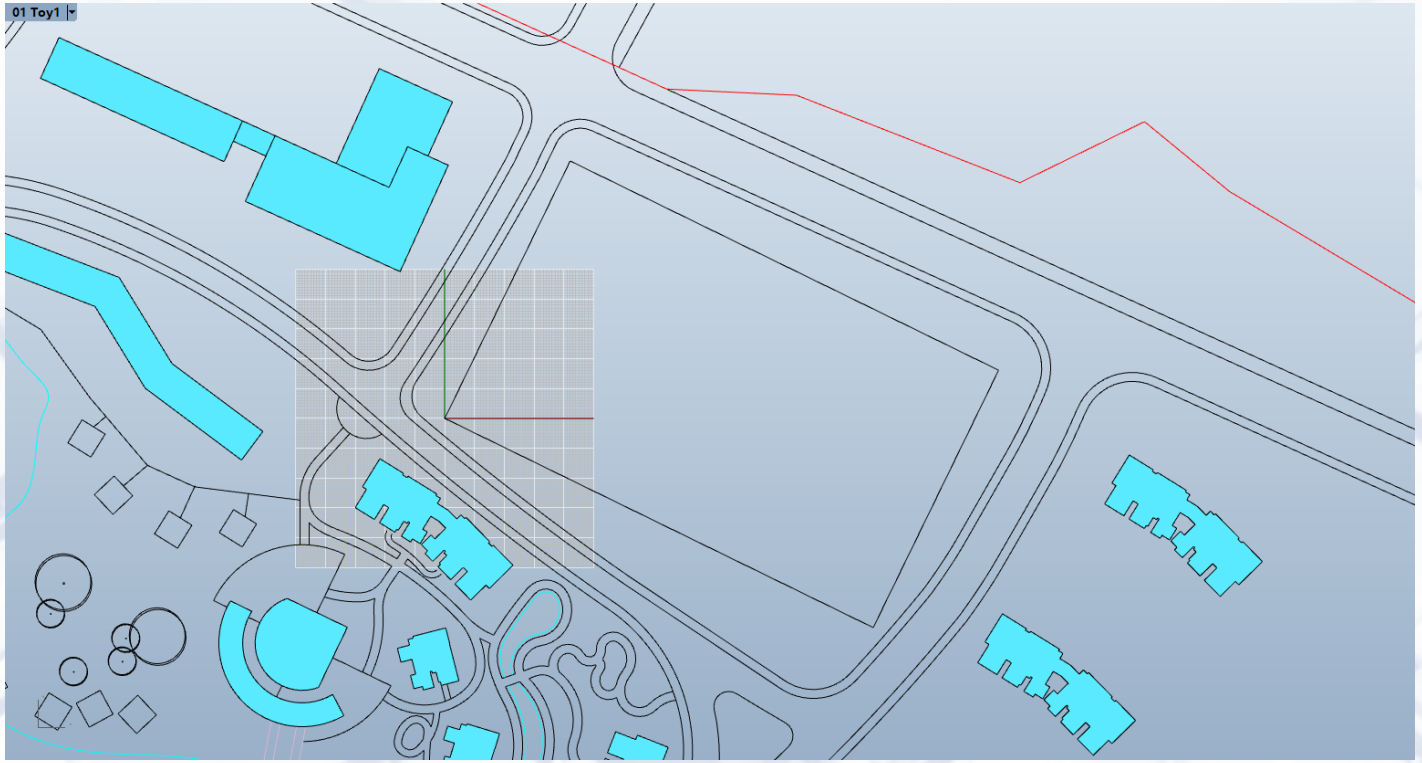
3.
TOY PROBLEM



3.1
TOY PROBLEM

3. TOY PROBLEM

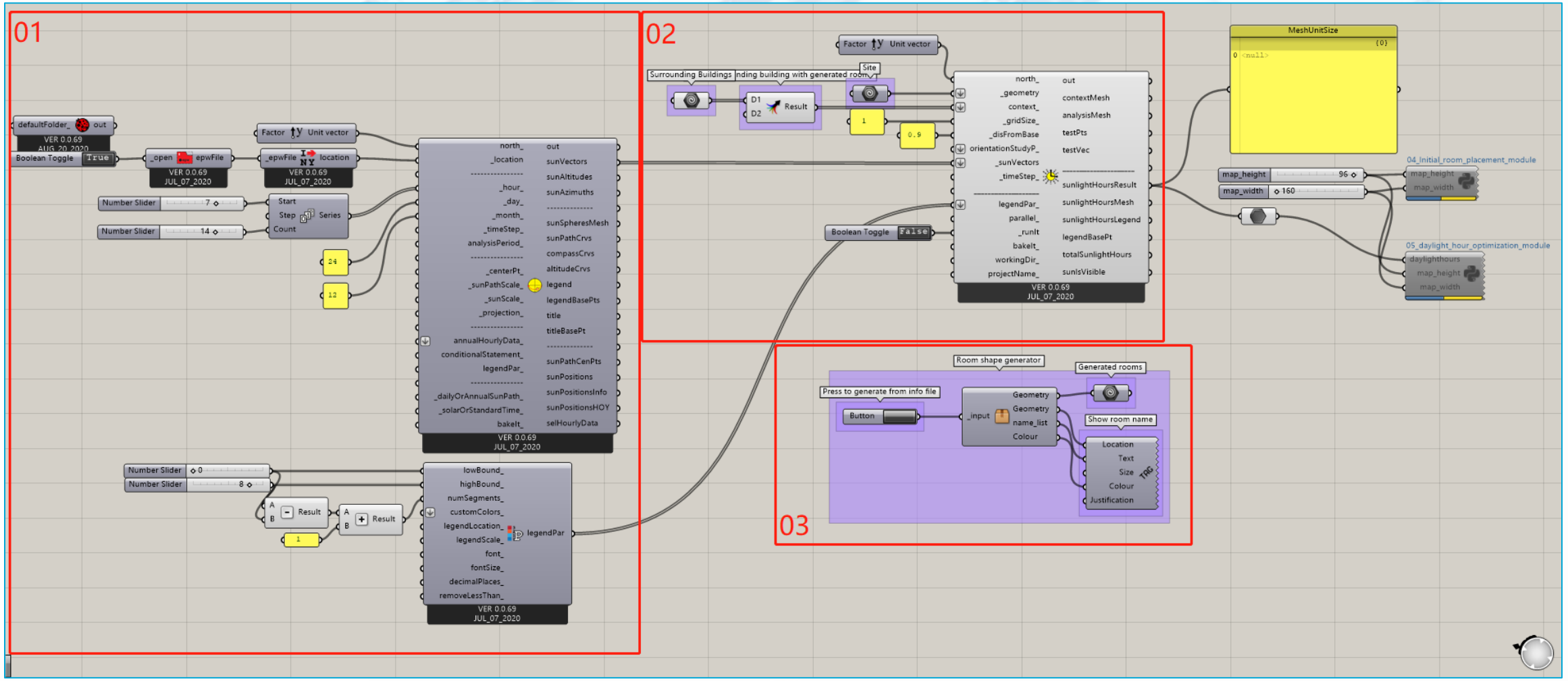
3.1 Toy Problem



In the toy problem, I need to realize an algorithm to generate a simple floor layout insists of 5 rooms and 3 circulations in this 160m x 96m design site, while daylight potentials are considered.

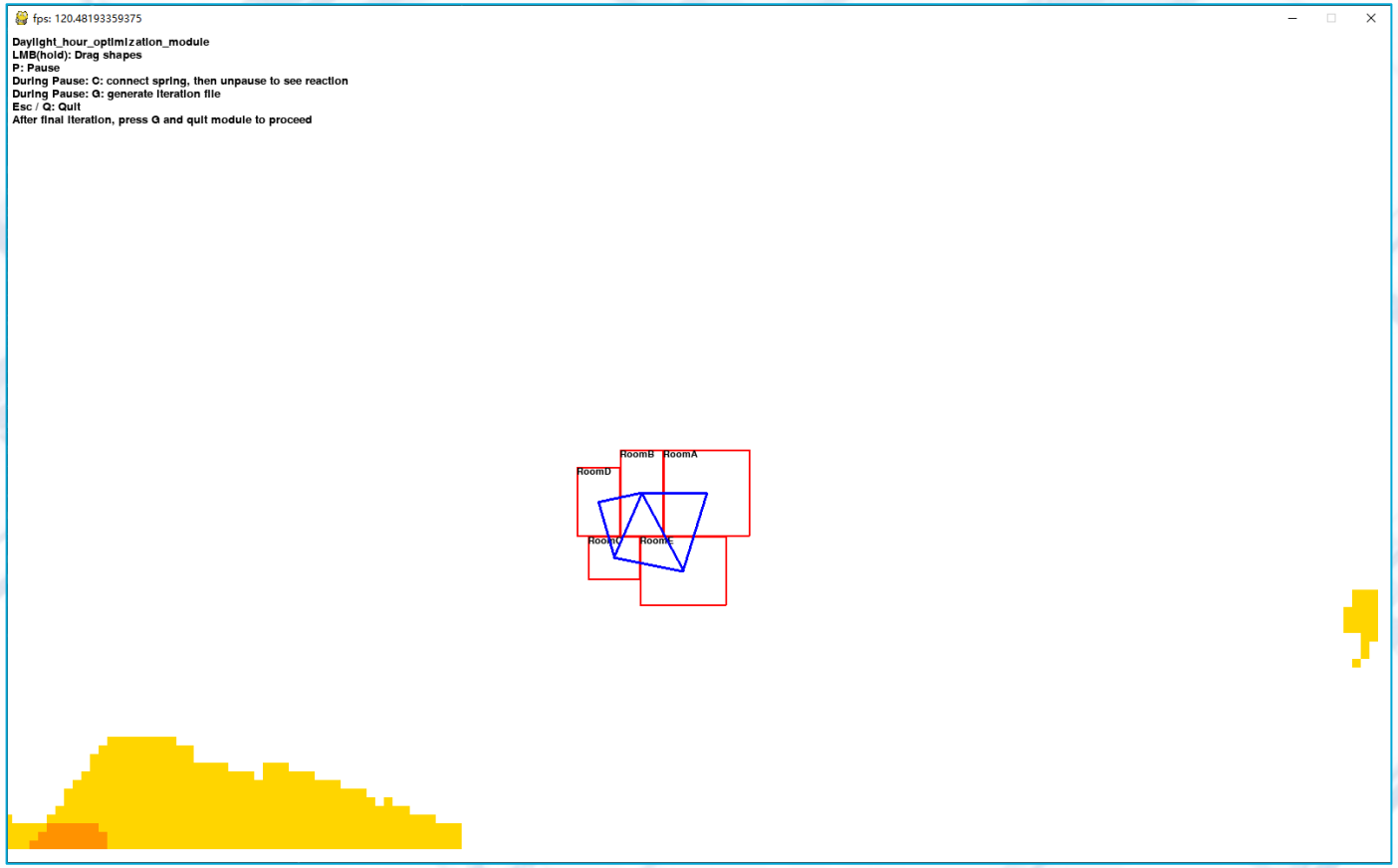
3.1 Toy Problem

Grasshopper layout



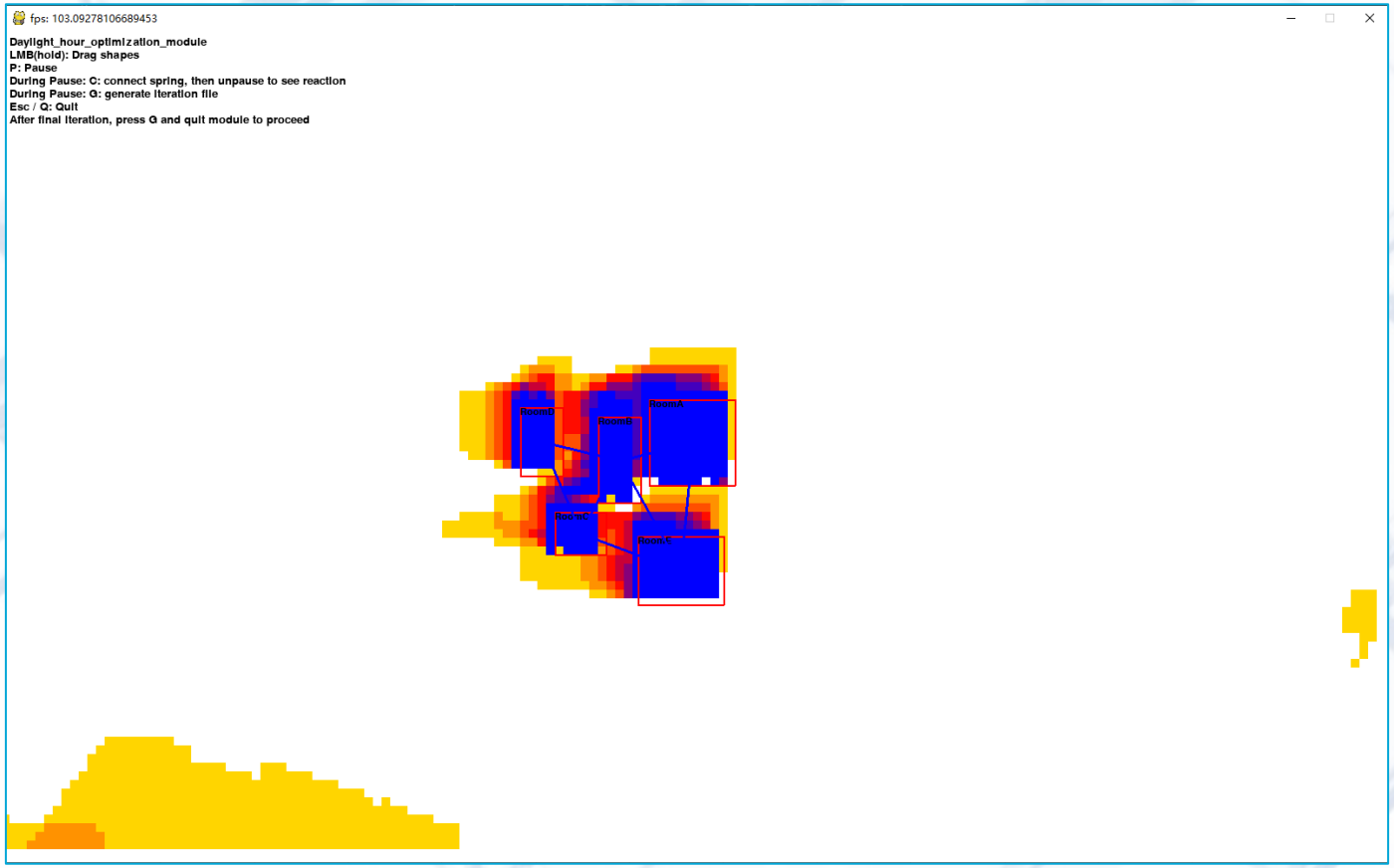
3.1 Toy Problem

Result of room placement



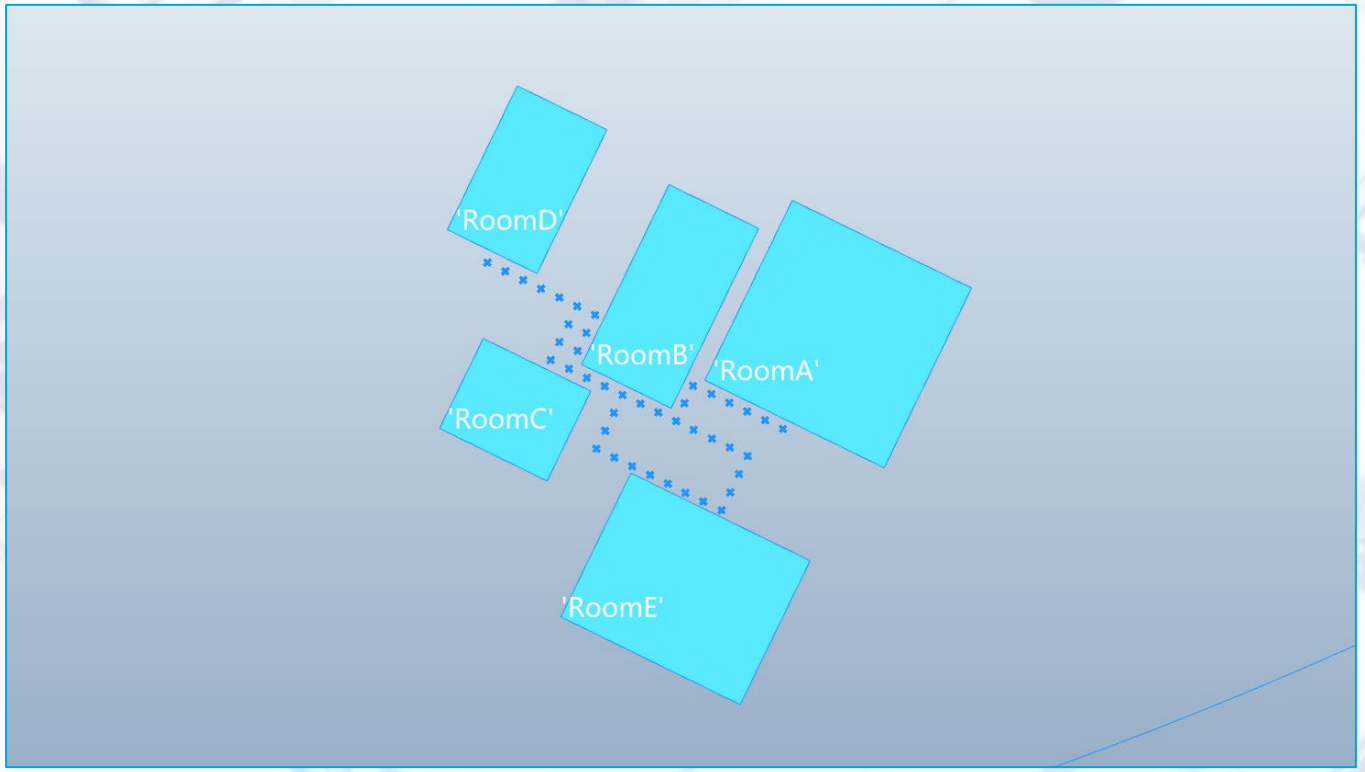
3.1 Toy Problem

Result of daylight hour optimization



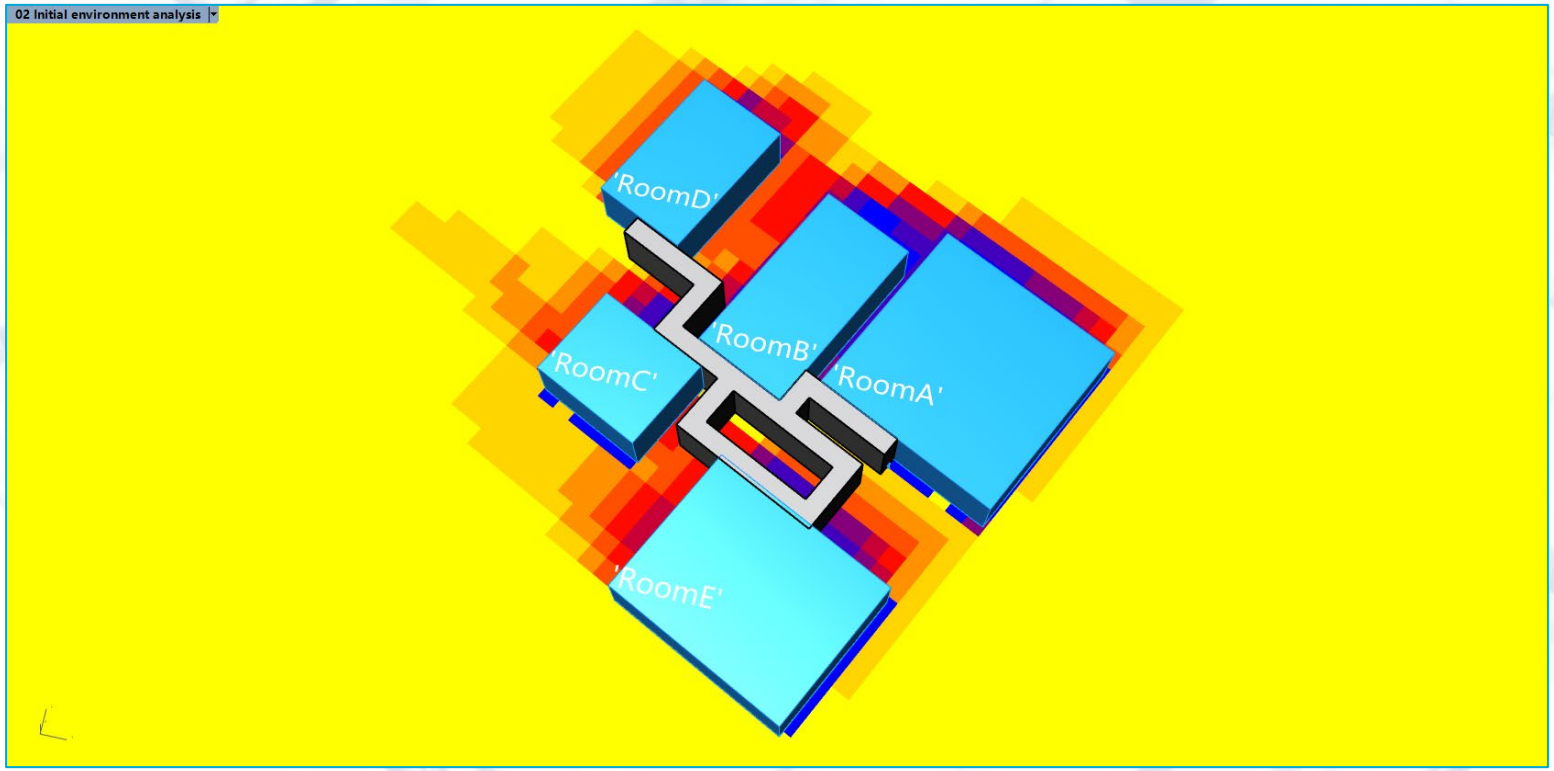
3.1 Toy Problem

Result of path-finding



3.1 Toy Problem

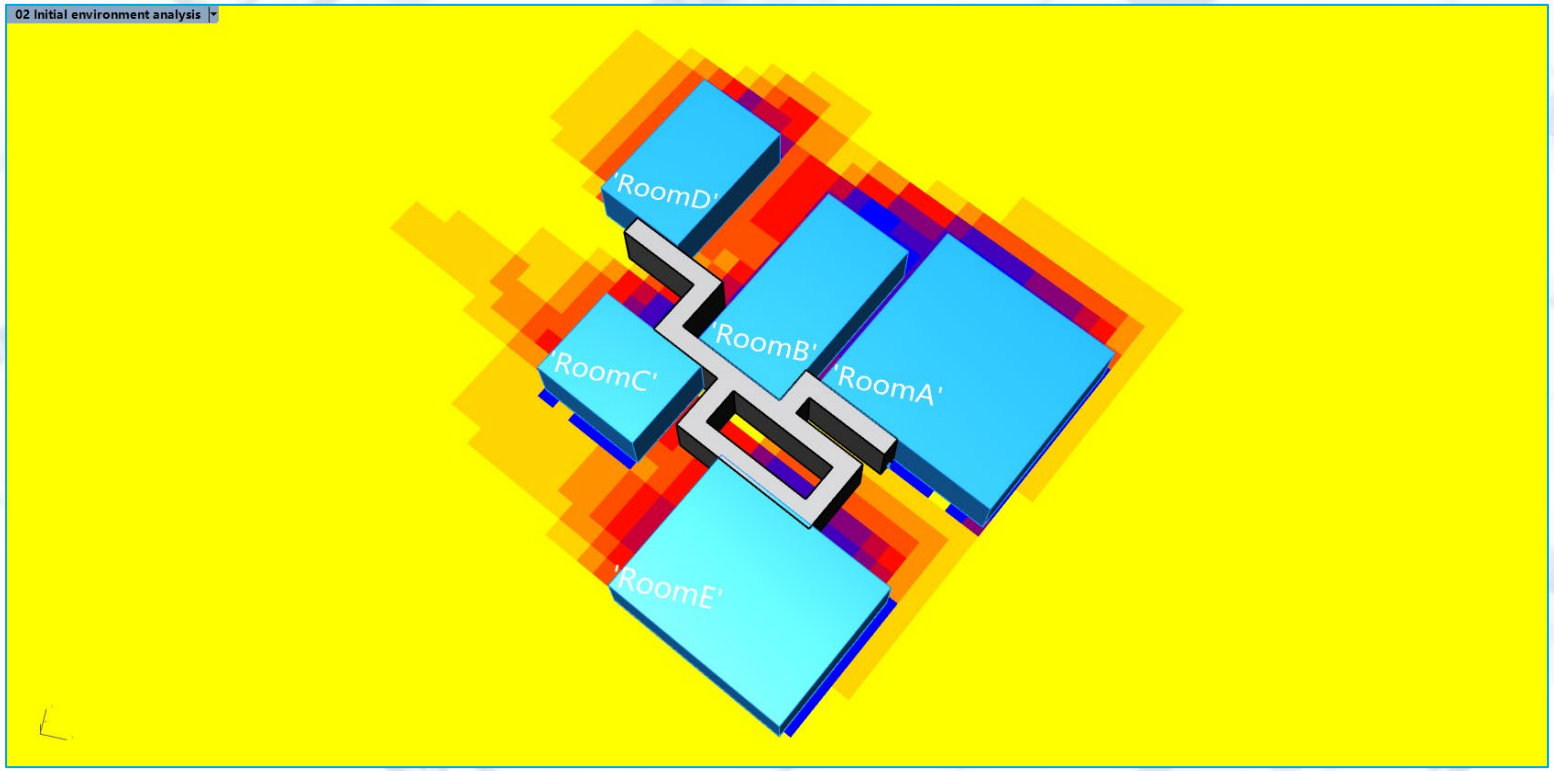
Result of corridor generation



3.2
PRELIMINARY RESULTS

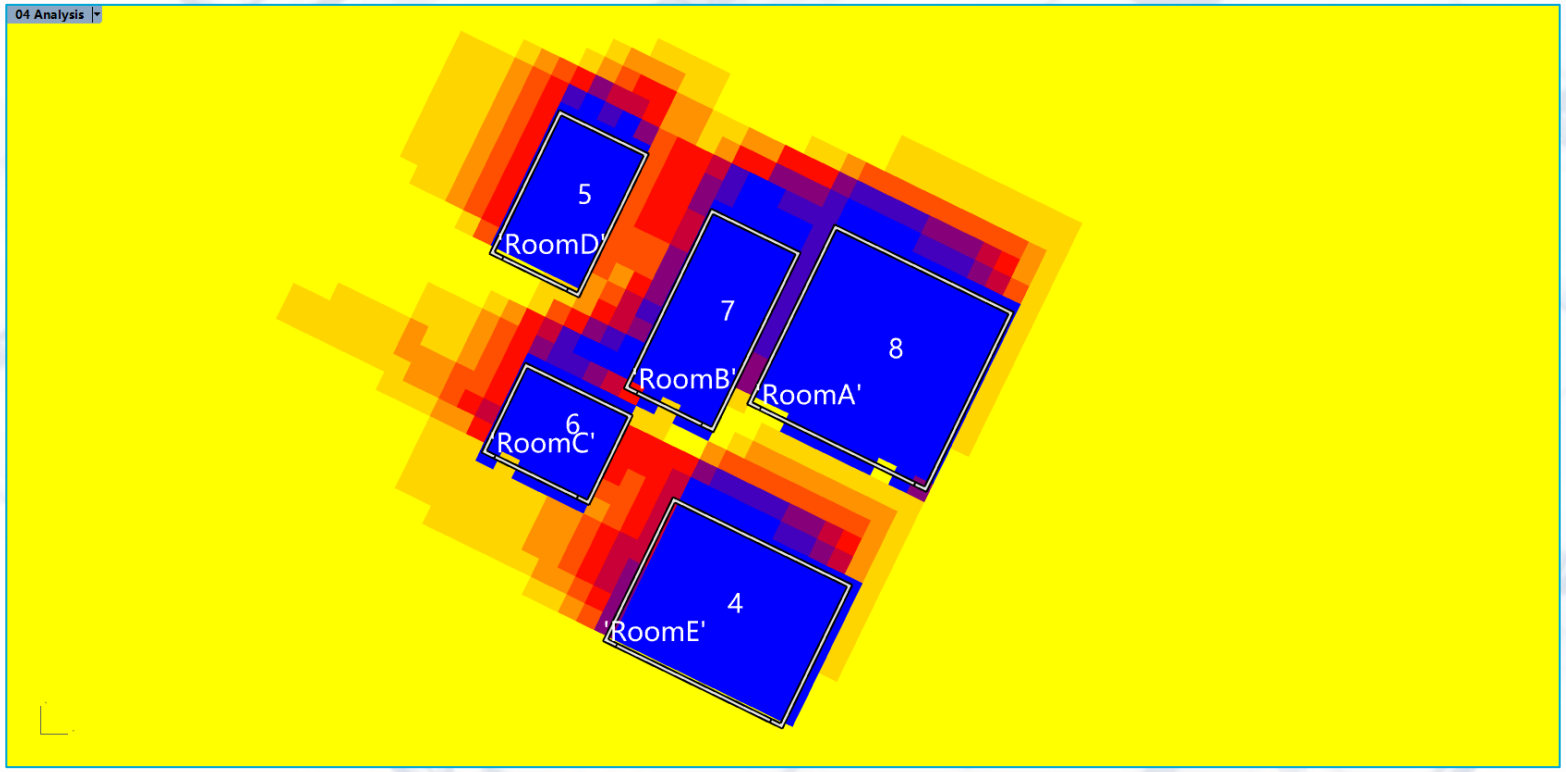
3.2 Preliminary Results

Result of corridor generation



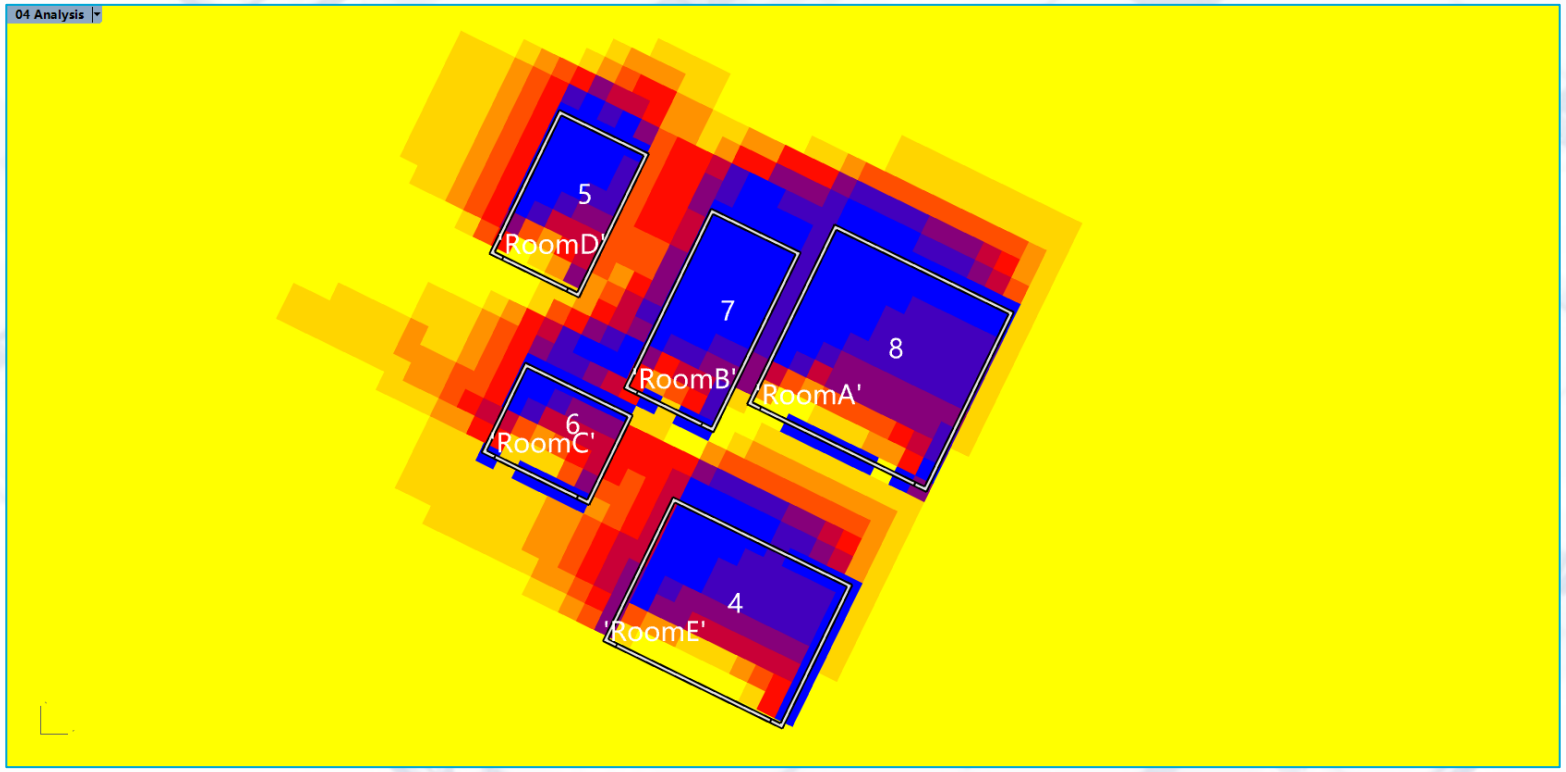
3.2 Preliminary Results

Result of corridor generation



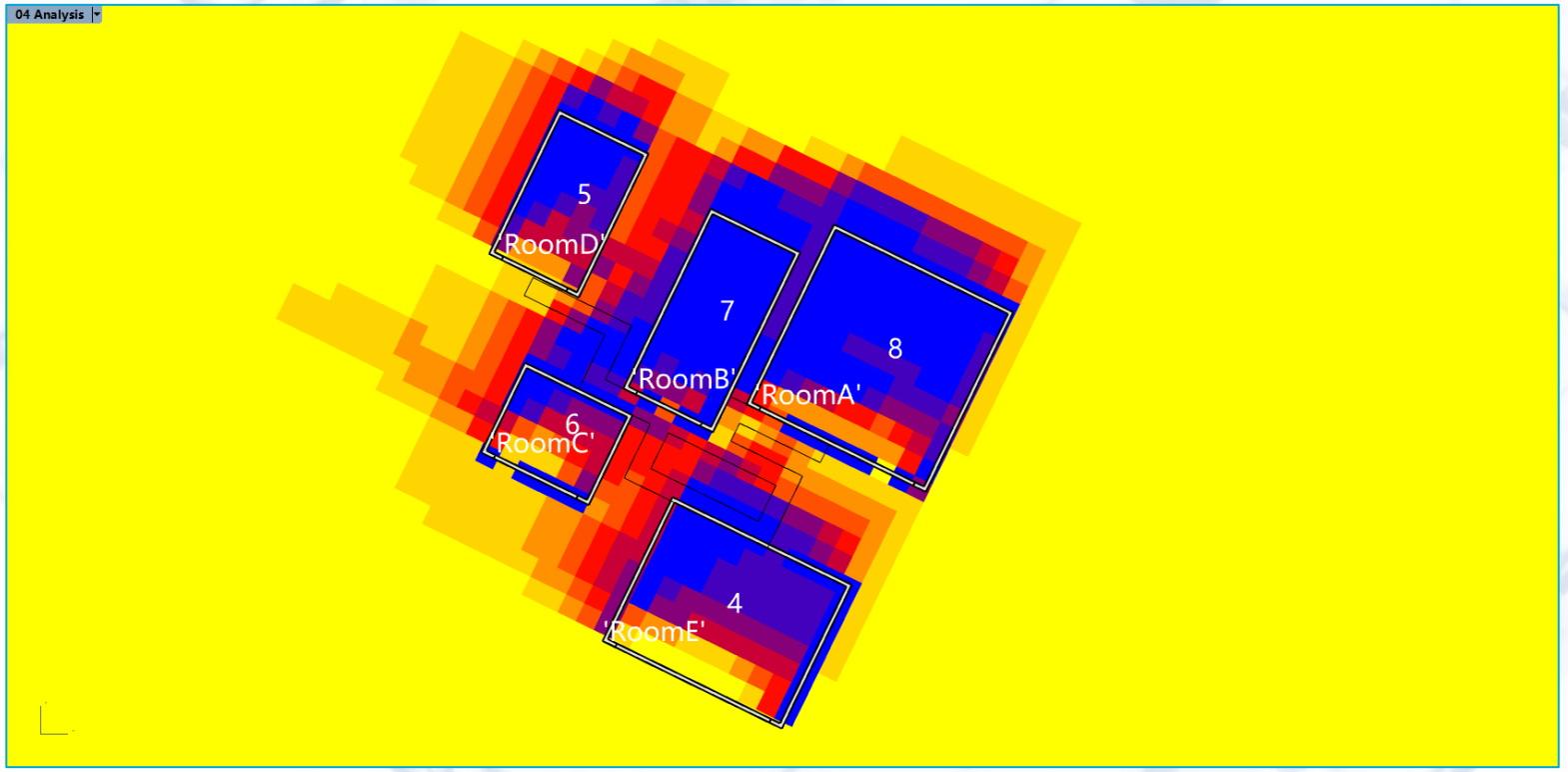
3.2 Preliminary Results

Result of corridor generation



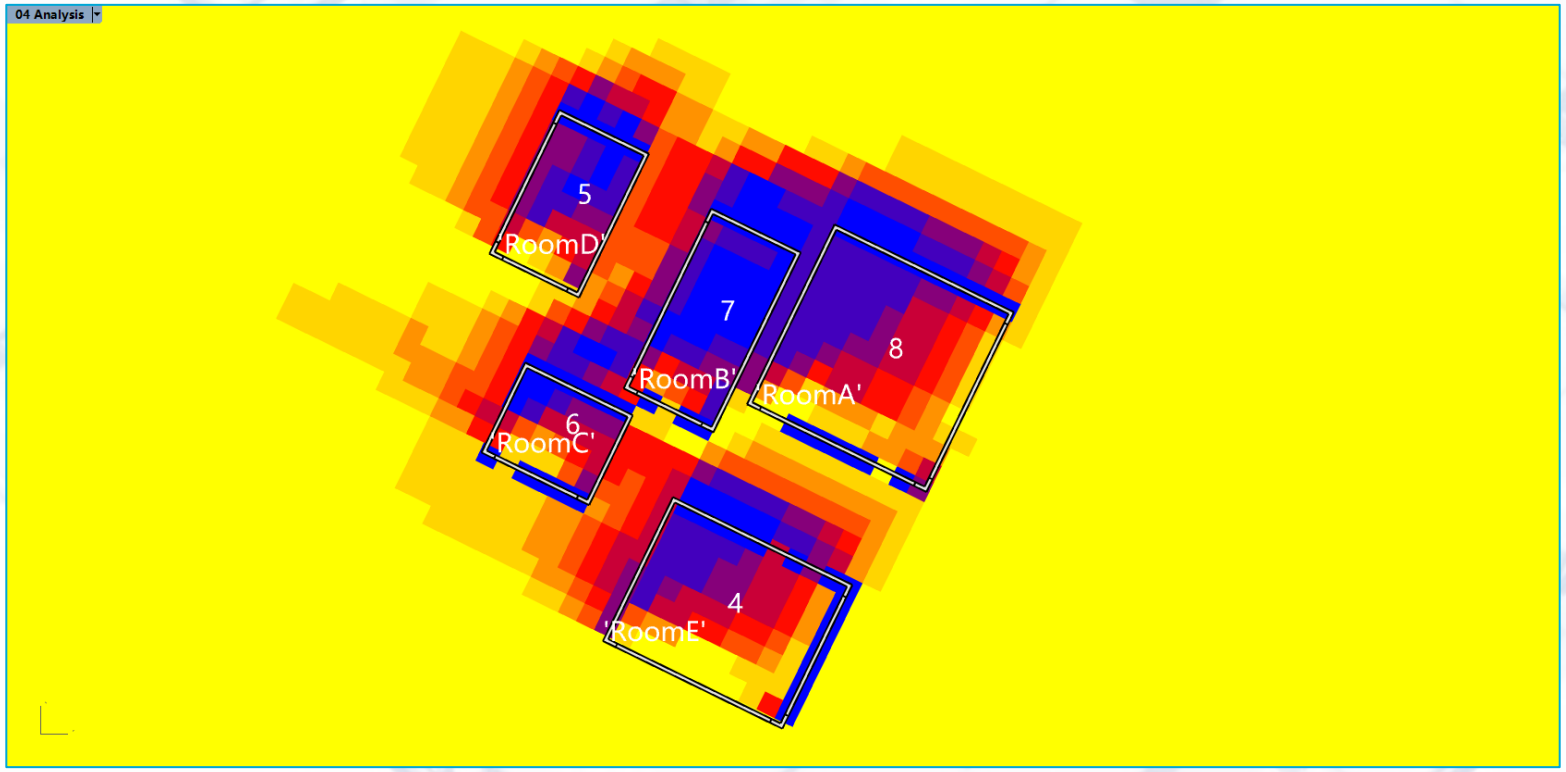
3.2 Preliminary Results

Result of corridor generation



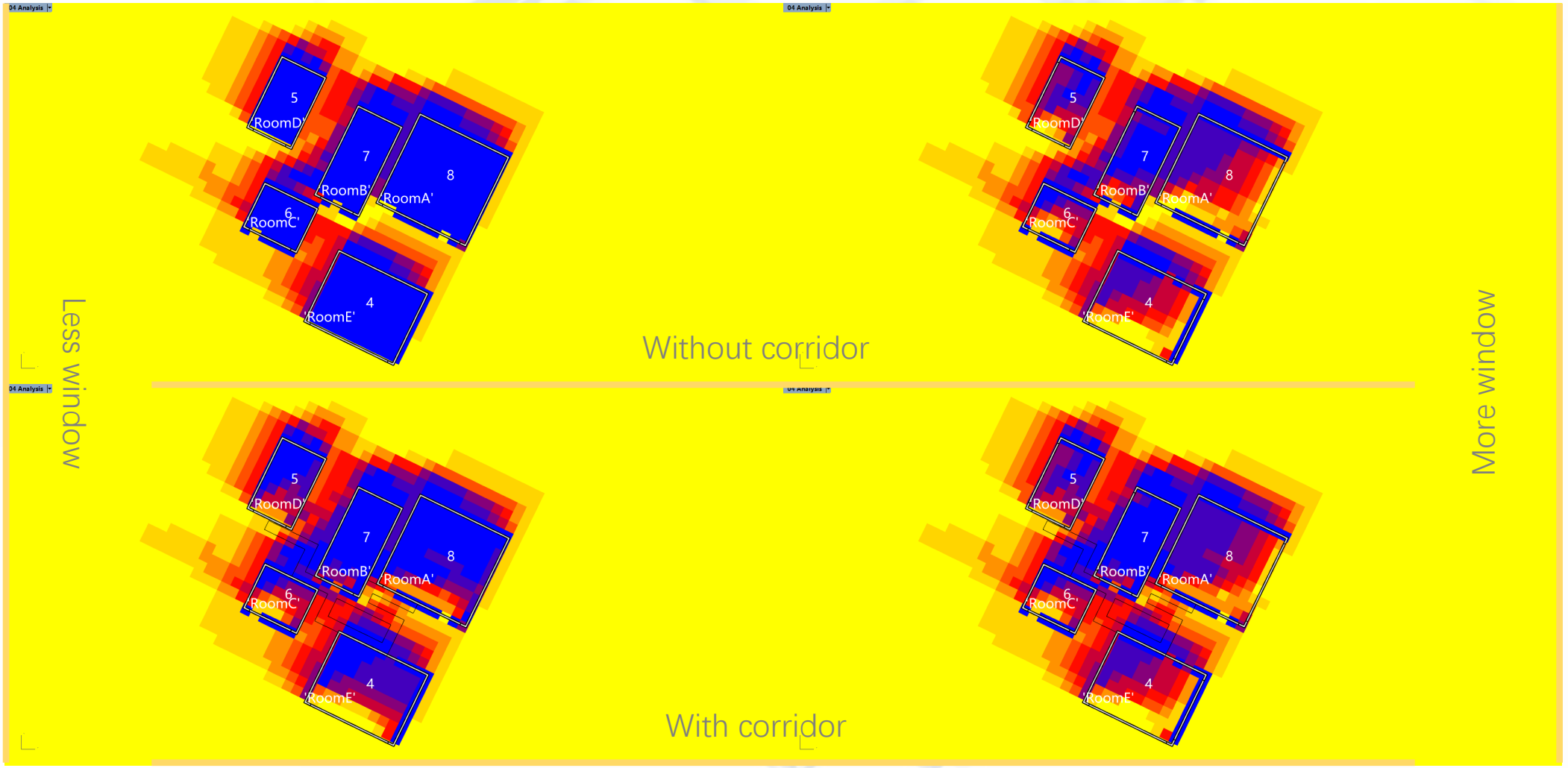
3.2 Preliminary Results

Result of corridor generation

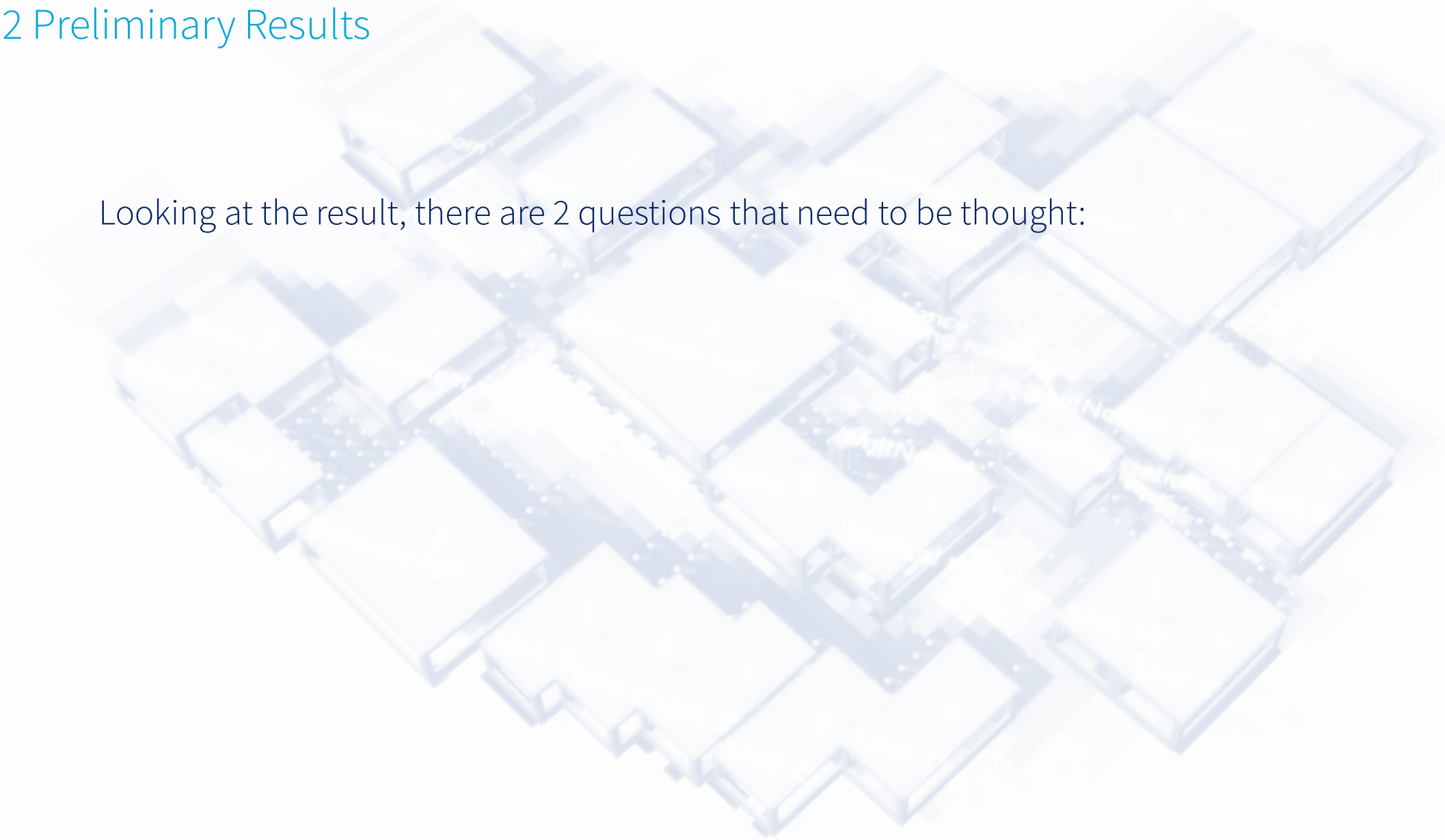


3.2 Preliminary Results

Result of corridor generation



3.2 Preliminary Results



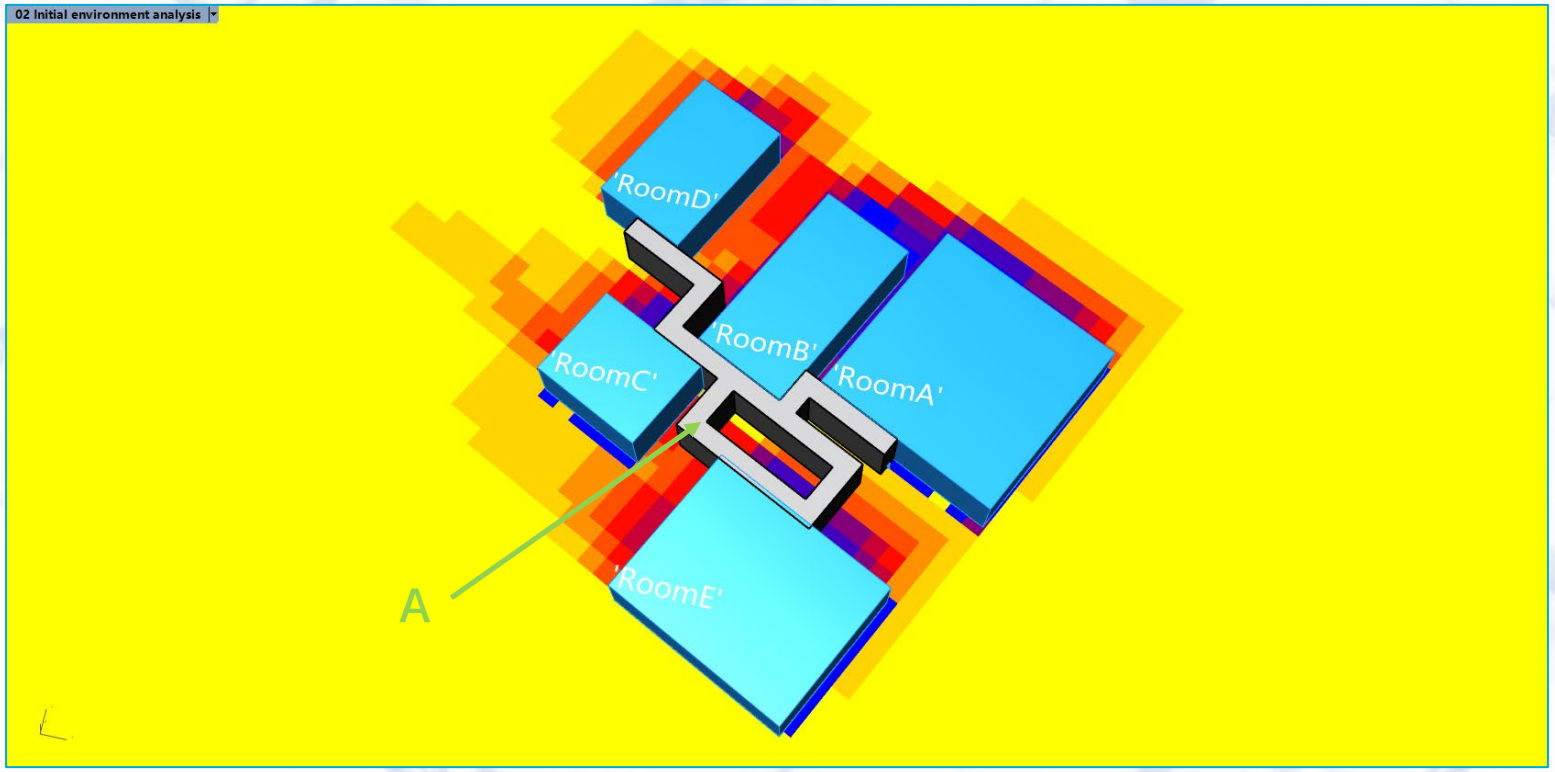
Looking at the result, there are 2 questions that need to be thought:

3.2 Preliminary Results

1. Is the order of placement previously determined representative to the needs towards certain requirements?
The order of placement previously determined generally took the aspect of ‘node importance’ in a graph, then place the room from south to north. The direction of placement is purely from the consideration of putting down the shape without conflict. But if the designer is not aware of the process, the first impression might create more costs more for designers to reach a satisfying result.

2. What forms of information should be provided?
The placement method right now involves human intervention. So, there is a need of the information actively update to the designer, when he/she is dragging the mouse.

3.2 Preliminary Results



4.
IMPROVEMENTS

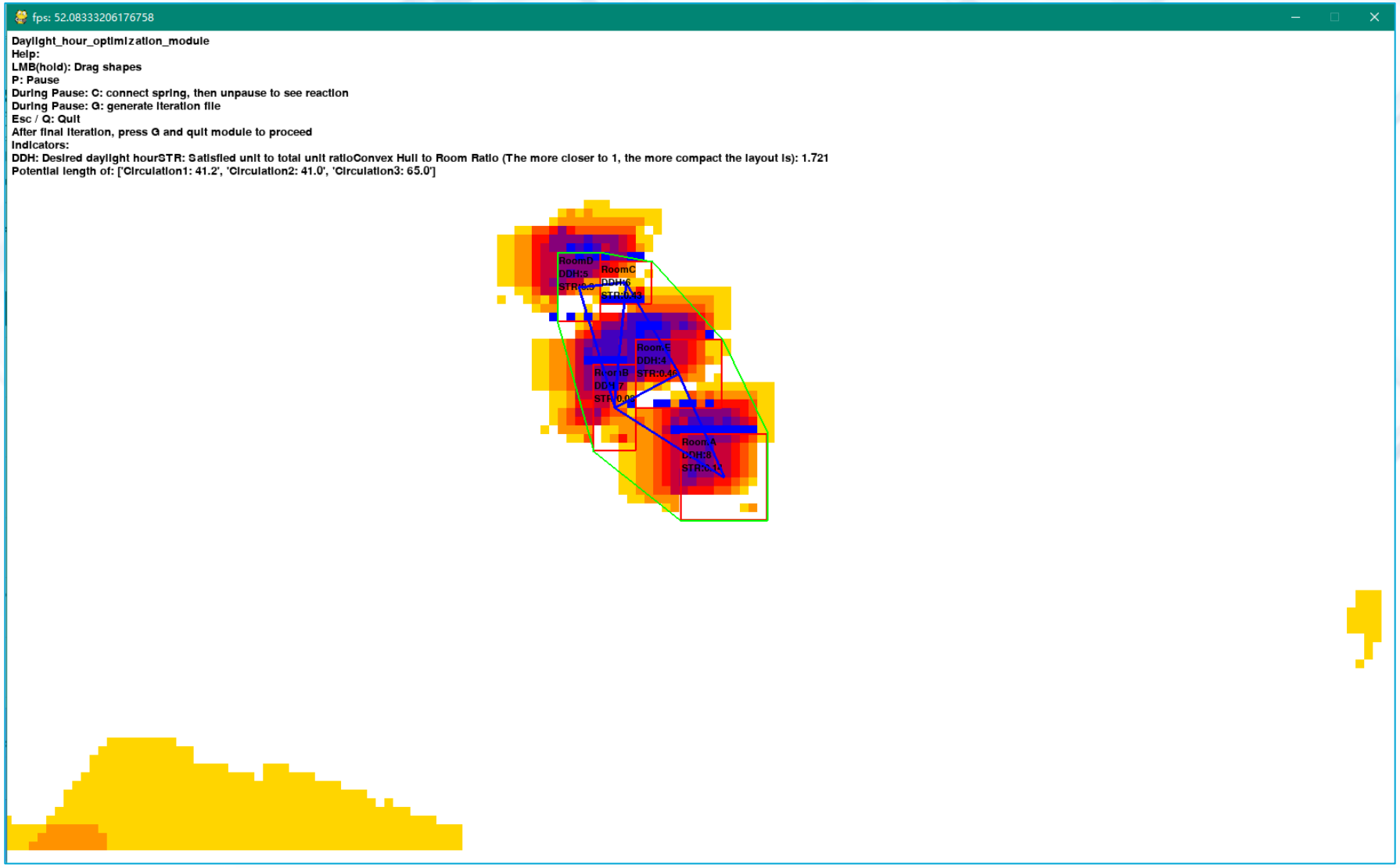
4.1 Optimization Indicators

Convex Hull to Room Ratio: It is the ratio between the size of the convex hull of the corner of all rooms and the total size of the rooms. The closer it is to 1, the more compact the layout is. The calculation of the convex hull is done by using Scipy Python library.

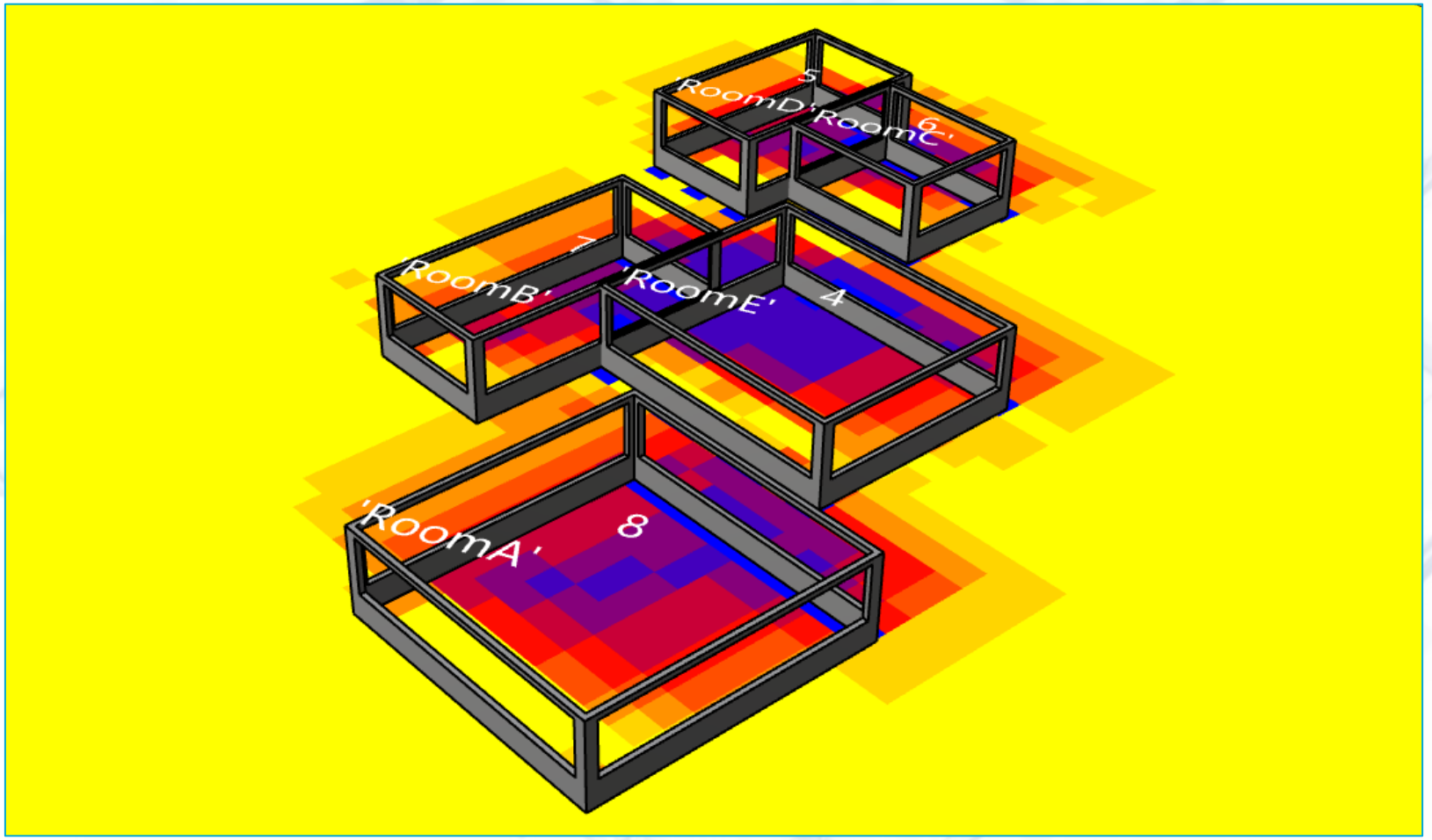
Potential length of each circulation: It is calculated by the Manhattan distance among all doors of rooms within the circulation. Therefore, it is only a very rough estimation, does not represent the final path length.

Satisfied unit to total unit ratio: It is the ratio of the number of grid unit that satisfied the desired daylight hour to the total grid unit of one room.

4.1 Optimization Indicators



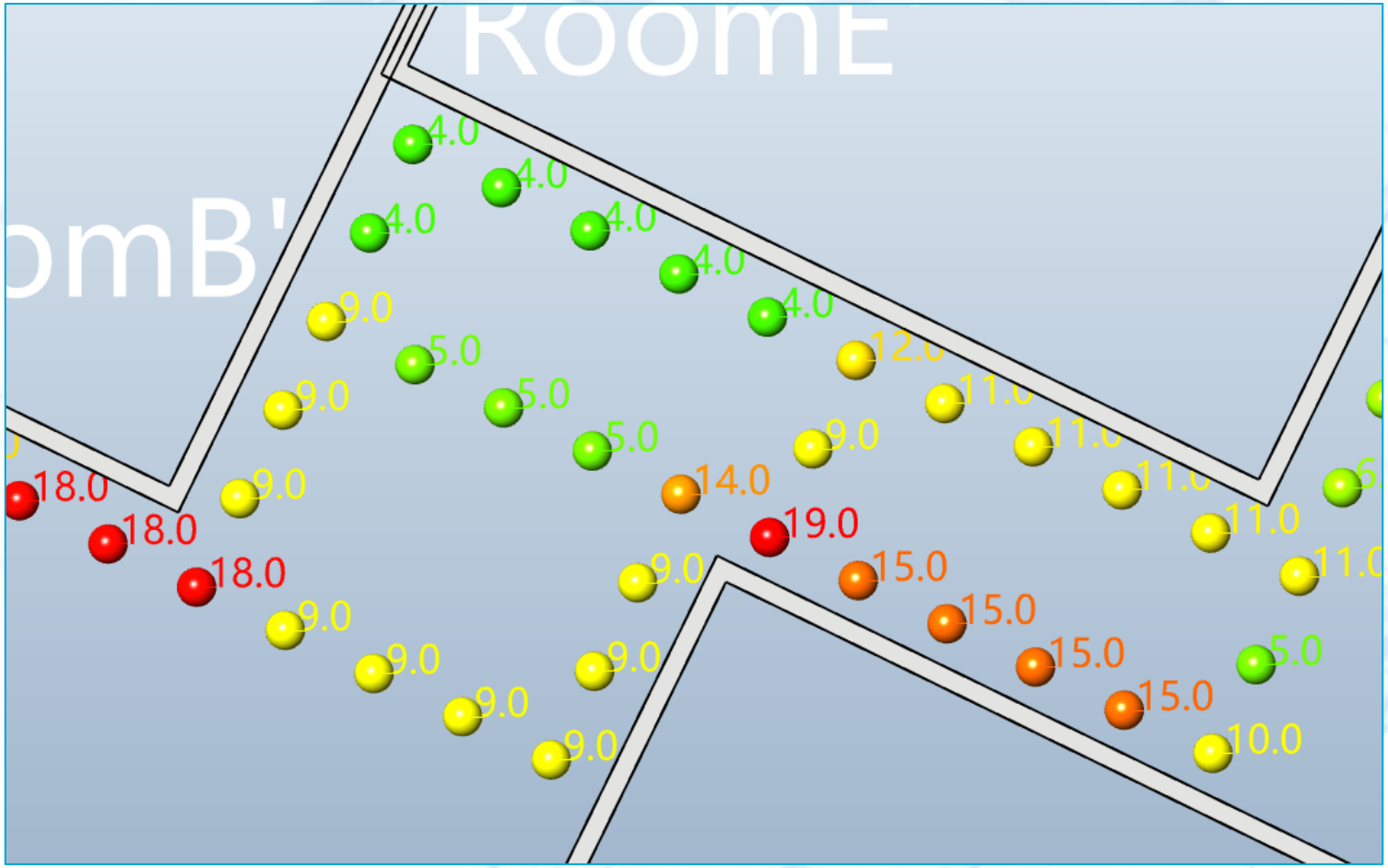
4.2 Object For Daylight Potential Analysis



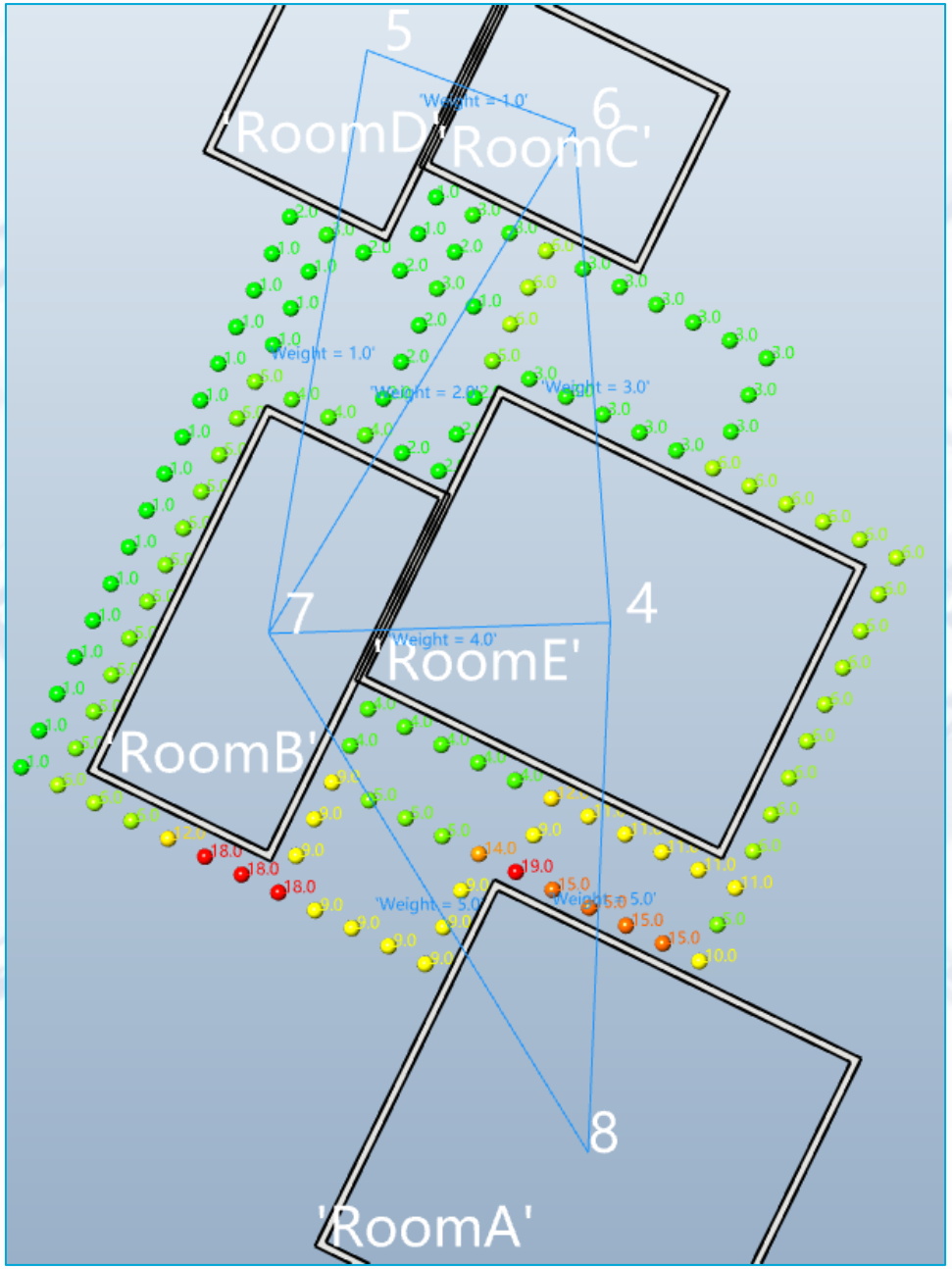
4.3 Corridor Selection

```
path = eval(infolist[0])
x_list = []
y_list = []
weight_list = []

path_array = np.zeros([96,160])

for i in range(len(path)):
    for j in range(len(path[i])):
        x = int(path[i][j][0])
        y = int(path[i][j][1])
        weight = path[i][j][2]
        path_array[y][x] += weight

x_list = []
y_list = []
weight_list = []
for i in range(96):
    for j in range(160):
        if path_array[i][j] != 0:
            x_list.append(j)
            y_list.append(96 - i)
            weight_list.append(path_array[i][j])
```

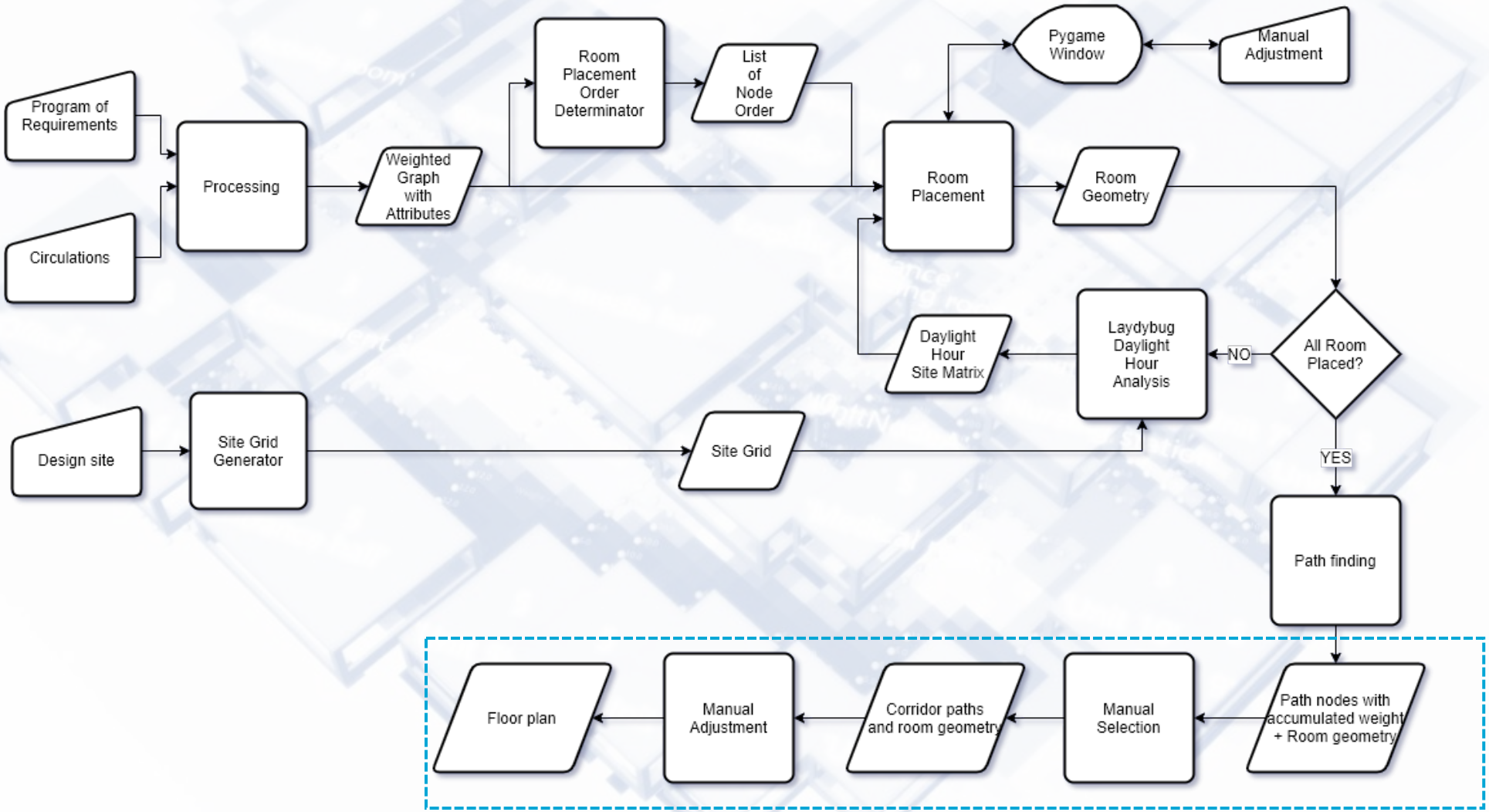
4.3 Corridor Selection

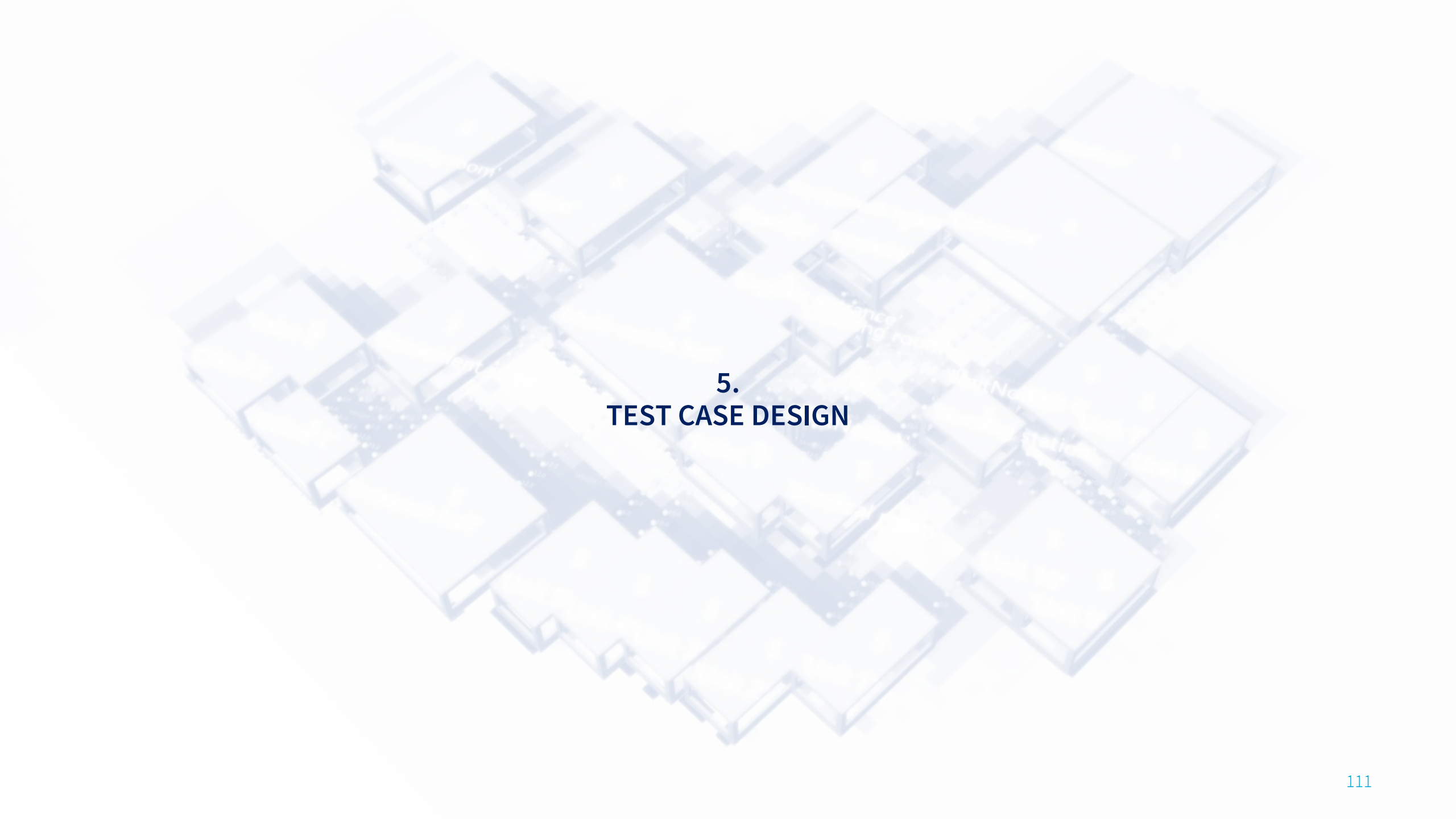


4.3 Corridor Selection



4.3 Corridor Selection

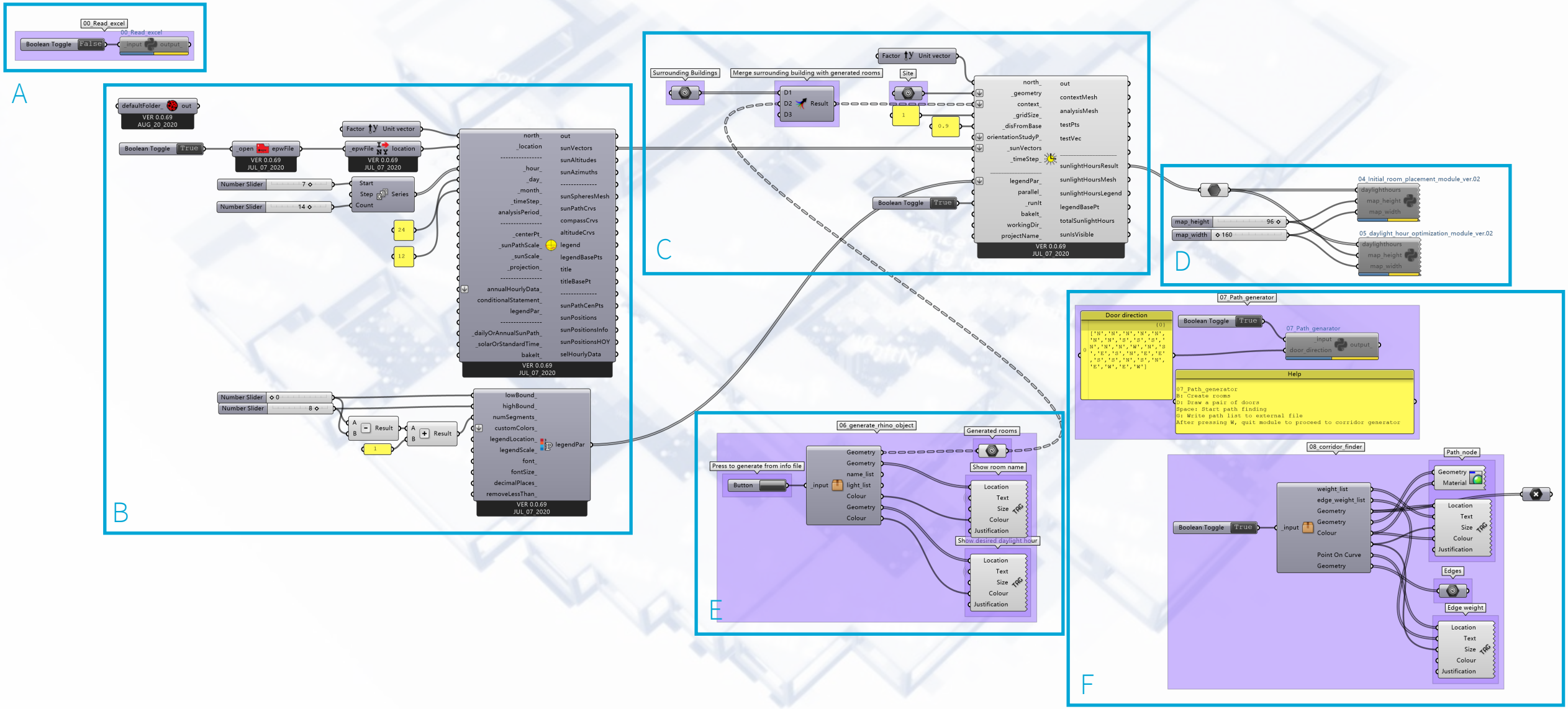




5.
TEST CASE DESIGN

5. Test Case Design

Grasshopper layout



Program of requirement

Space	Size	Quantity	Space	Size	Quantity
Entrance hall	80	1	**Recreation:**		
Living:			Activity room	60	1
Living units	32	10	Gym	60	1
Medication:			Multi-media hall	120	1
Nursing station	20	1	**Administration:**		
Medical room	32	1	Offices	24	2
Public service:			**Others:**		
Cafeteria	120	1	Toilet	35	2
Kitchen	120	1	Equipment room	50	1
Laundry	40	1	Storage	24	1
Cleaning room	12	1	Logistic entrance	20	1
Convenient store	40	1			

Program of requirement into excel info

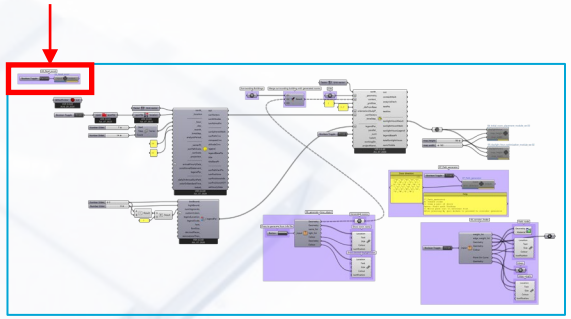
	A	B	C	D
1	**Space**	**Size**	**Daylight hour**	**Door direction**
2	Entrance hall	80	5	S
3	Unit 1	32	5	N
4	Unit 2	32	5	N
5	Unit 3	32	5	N
6	Unit 4	32	5	N
7	Unit 5	32	5	N
8	UnitNodeA	1	0	none
9	Unit 6	32	5	N
10	Unit 7	32	5	N
11	Unit 8	32	5	N
12	Unit 9	32	5	N
13	Unit 10	32	5	N
14	UnitNodeA	1	0	none
15	Nursing station	20	5	S
16	Medical room	32	5	S
17	Cafeteria	120	8	N
18	Kitchen	120	3	W
19	Laundry	40	3	E
20	Cleaning room	12	3	E
21	Convenient store	40	3	W
22	Activity room	60	8	N
23	Gym	60	3	E
24	Multi-media hall	120	3	W
25	Office 1	24	5	N
26	Office 2	24	5	N
27	Toilet 1	35	0	S
28	Toilet 2	35	0	S
29	Equipment room	50	0	S
30	Storage	24	0	S
31	Logistic entrance	20	0	S
32				
33				
34				
35				
36				

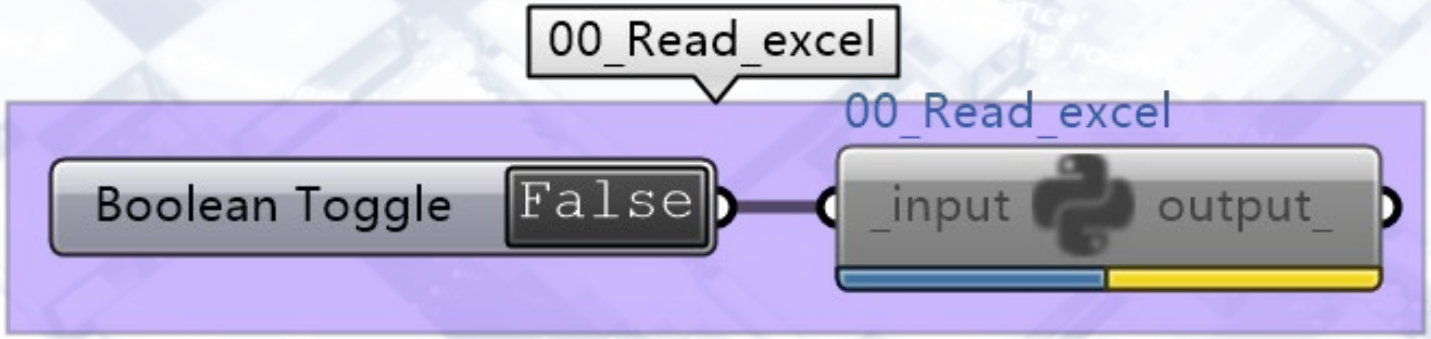
PoR | Circulation | Sheet1 | Simplified

	A	B	C	D	E	F	G	H	I
1	Circulation 1	Circulation 2	Circulation 3	Circulation 4	Circulation 5	Circulation 6	Circulation 7	Circulation 8	Circulation 9
2	**UnitGroup1**	**UnitGroup2**	**UnitNodeA**	**UnitNodeB**	**Cleaner**	**Medical**	**Administrativ**	**Food**	**Logistic**
3	5	5	3	3	4	4	3	2	1
4	Unit 1	Unit 6	UnitNodeA	UnitNodeA	Cleaning room	Nursing station	Entrance hall	Kitchen	Logistic entrance
5	Unit 2	Unit 7	Toilet 1	Toilet 1	UnitNodeA	Medical room	Office 1	Cafeteria	Cleaning room
6	Unit 3	Unit 8	Convenient store	Convenient store	UnitNodeB	UnitNodeA	Office 2	UnitNodeA	Equipment room
7	Unit 4	Unit 9	Activity room	Activity room	Laundry	UnitNodeB	Toilet 2	UnitNodeB	Storage
8	Unit 5	Unit 10	Gym	Gym	Cleaning room	Nursing station	Entrance hall	Kitchen	Logistic entrance
9	UnitNodeA	UnitNodeB	Multi-media hall	Multi-media hall					
10	Unit 1	Unit 6	UnitNodeA	UnitNodeA					
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									

PoR | Circulation | Sheet1 | Simplified

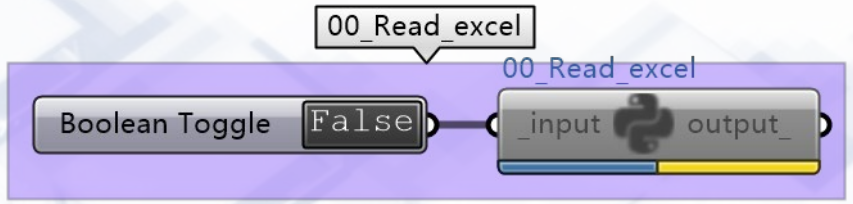
00_Read_excel

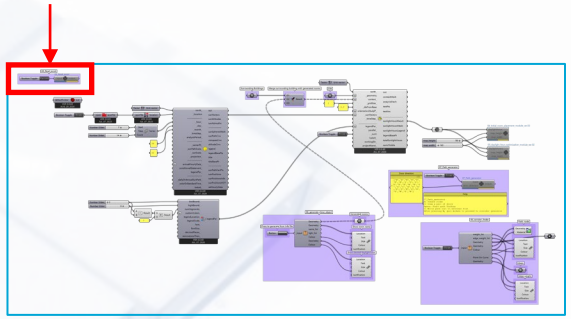




5. Test Case Design

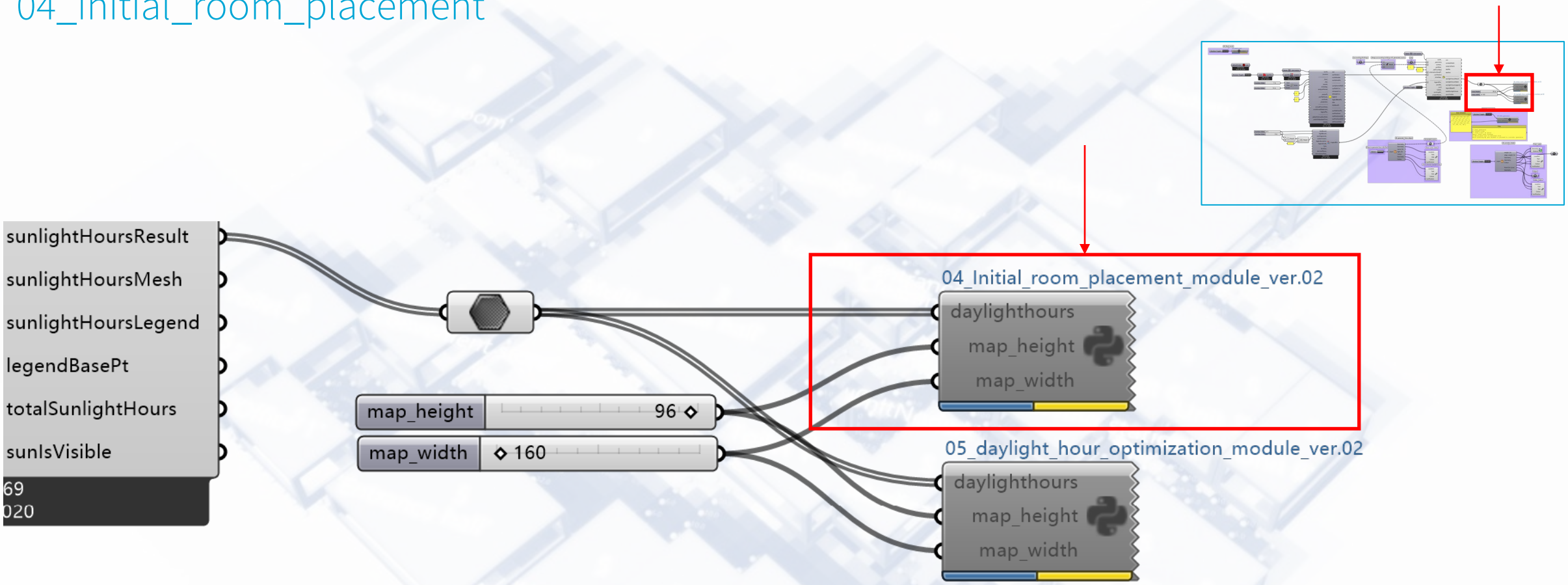
00_Read_excel



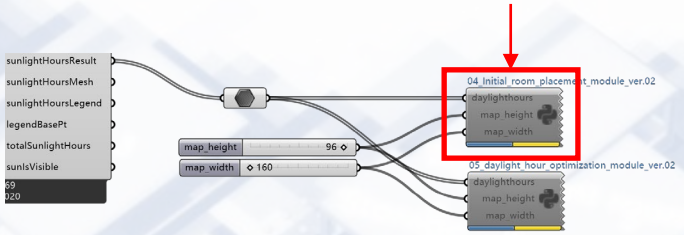


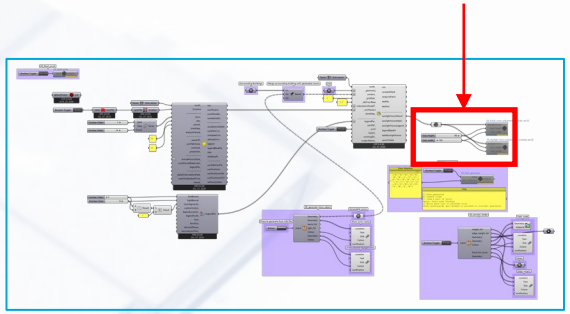
```
['Entrance hall', 'Unit 1', 'Unit 2', 'Unit 3', 'Unit 4', 'Unit 5', 'UnitNodeA', 'Unit 6', 'Unit 7',
'Unit 8', 'Unit 9', 'Unit 10', 'UnitNodeB', 'Nursing station', 'Medical room', 'Cafeteria',
'Kitchen', 'Laundry', 'Cleaning room', 'Convenient store', 'Activity room', 'Gym', 'Multi-media
hall', 'Office 1', 'Office 2', 'Toilet 1', 'Toilet 2', 'Equipment room', 'Storage', 'Logistic
entrance']
[80, 32, 32, 32, 32, 32, 1, 32, 32, 32, 32, 32, 1, 20, 32, 120, 120, 40, 12, 40, 60, 60, 120, 24, 24,
35, 35, 50, 24, 20]
[5, 5, 5, 5, 5, 5, 0, 5, 5, 5, 5, 5, 0, 5, 5, 8, 3, 3, 3, 3, 8, 3, 3, 5, 5, 0, 0, 0, 0, 0]
['S', 'N', 'N', 'N', 'N', 'N', 'none', 'N', 'N', 'N', 'N', 'N', 'none', 'S', 'S', 'N', 'W', 'E', 'E',
'W', 'N', 'E', 'W', 'N', 'N', 'S', 'S', 'S', 'S', 'S']
[['Unit 1', 'Unit 2', 'Unit 3', 'Unit 4', 'Unit 5', 'UnitNodeA', 'Unit 1'], ['Unit 6', 'Unit 7',
'Unit 8', 'Unit 9', 'Unit 10', 'UnitNodeB', 'Unit 6'], ['UnitNodeA', 'Toilet 1', 'Entrance hall',
'Convenient store', 'Activity room', 'Gym', 'Multi-media hall', 'UnitNodeA'], ['UnitNodeA', 'Toilet
1', 'Entrance hall', 'Convenient store', 'Activity room', 'Gym', 'Multi-media hall', 'UnitNodeA'],
['Cleaning room', 'UnitNodeA', 'UnitNodeB', 'Laundry', 'Cleaning room'], ['Nursing station', 'Medical
room', 'UnitNodeA', 'UnitNodeB', 'Nursing station'], ['Entrance hall', 'Office 1', 'Office 2',
'Toilet 2', 'Entrance hall'], ['Kitchen', 'Cafeteria', 'UnitNodeA', 'UnitNodeB', 'Kitchen'],
['Logistic entrance', 'Cleaning room', 'Equipment room', 'Storage', 'Logistic entrance']]
[5, 5, 3, 3, 4, 4, 3, 2, 1]
```

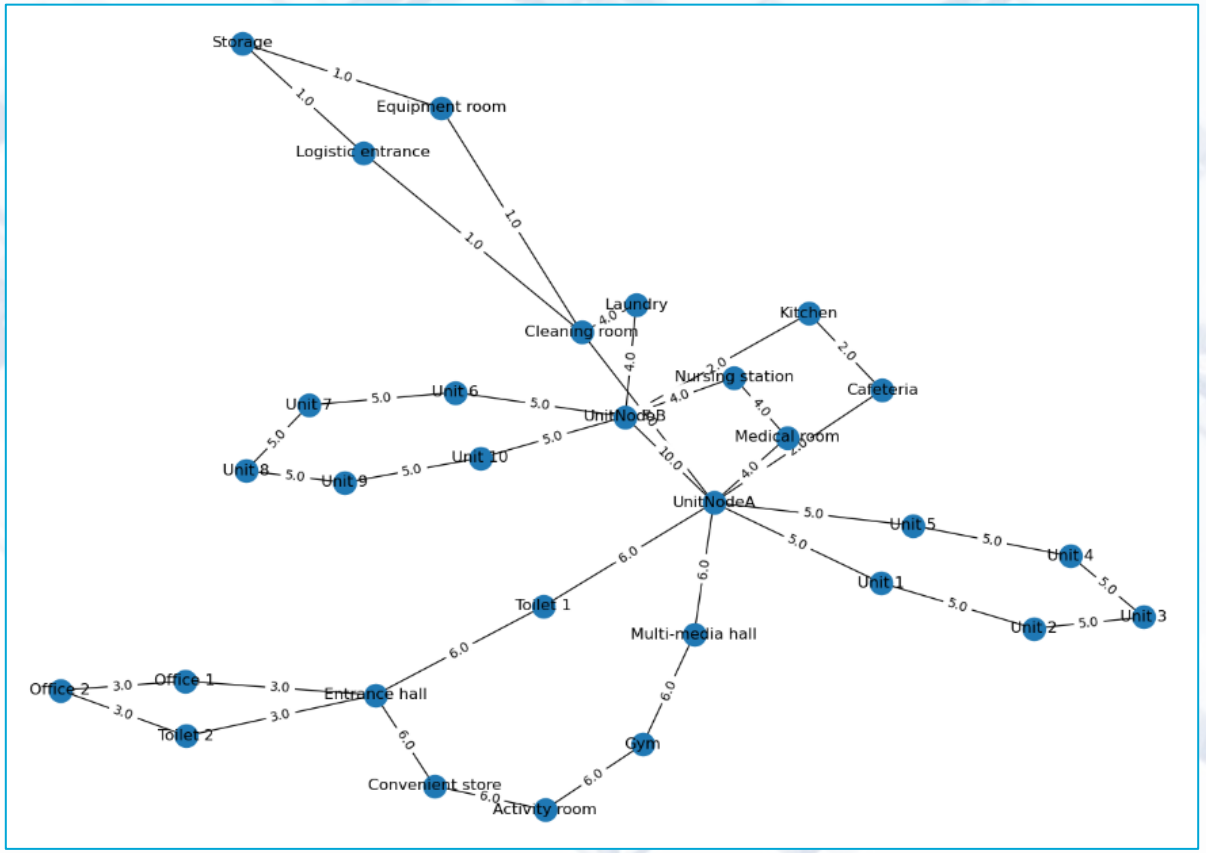
04_initial_room_placement



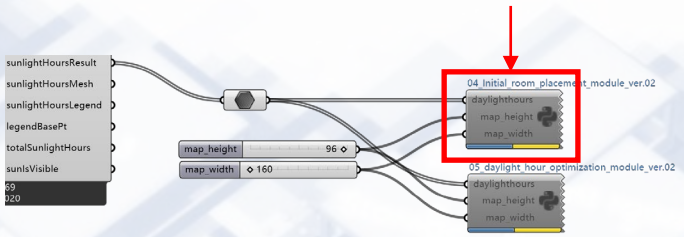
04_initial_room_placement

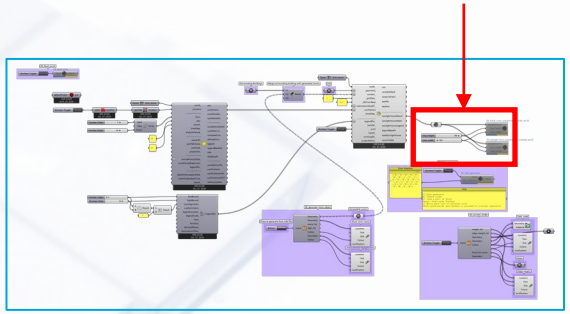






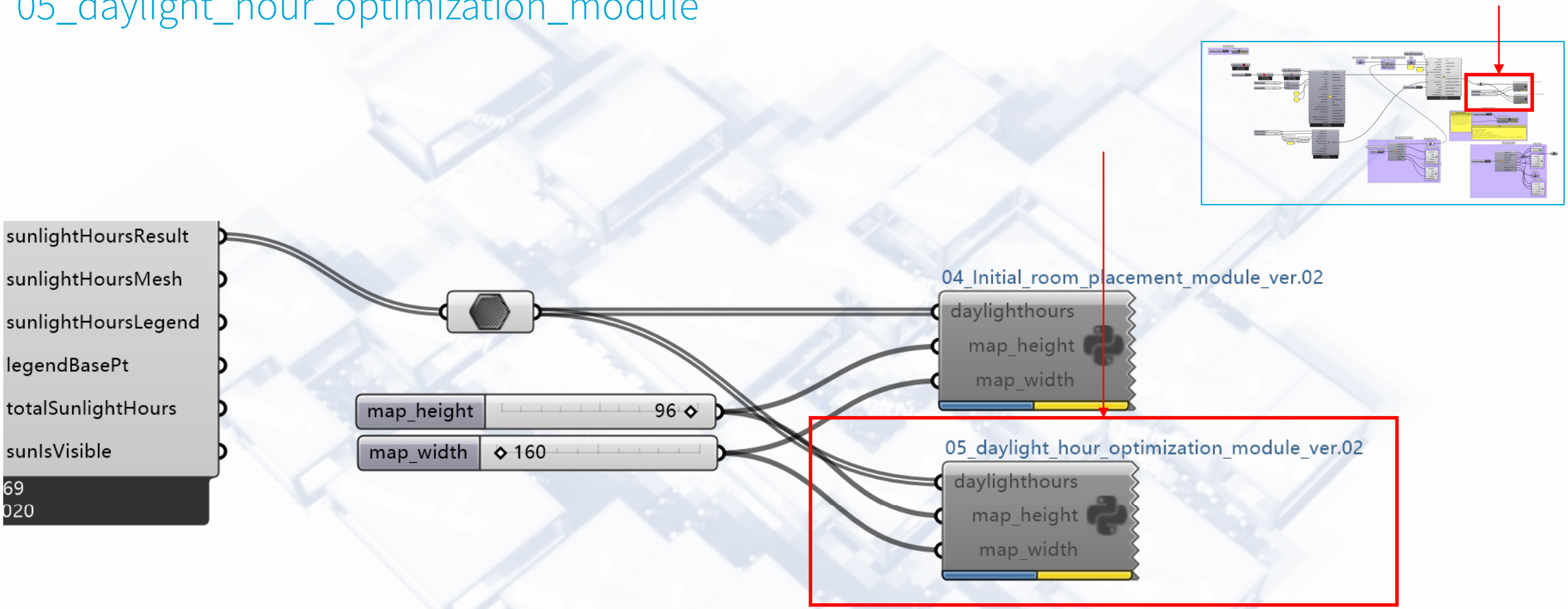
4_5_info_exchange



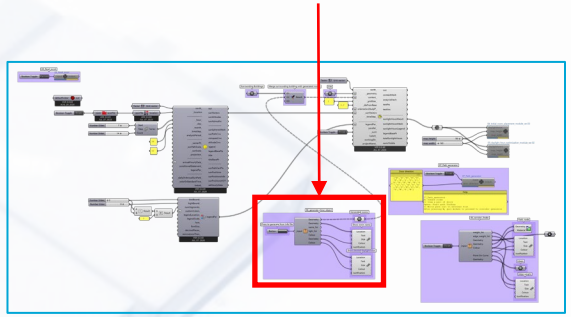


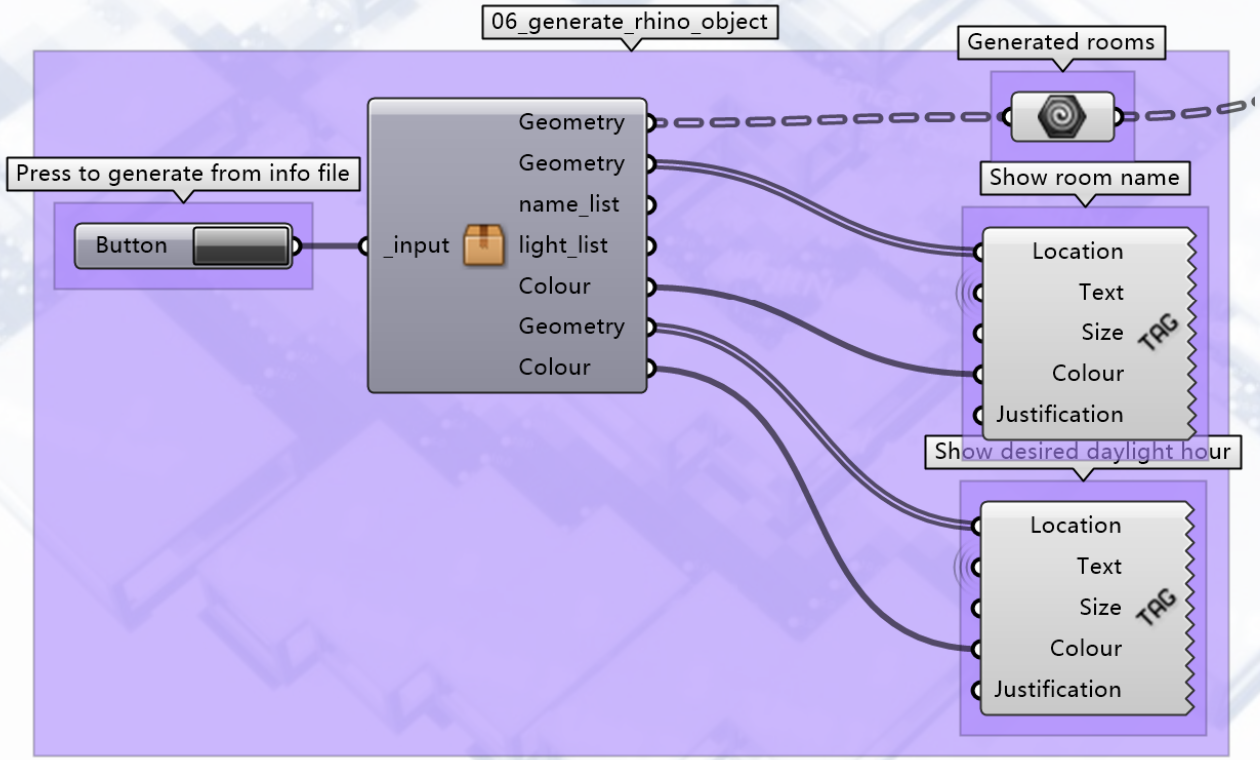
```
30
[(748, 869), (675, 869), (844, 840), (683, 824), (928, 816), (967, 802), (1006, 785), (1049, 777),
(742, 760), (1088, 731), (949, 689), (845, 655), (1003, 655), (948, 649), (615, 645), (1154, 611),
(1114, 611), (726, 605), (869, 585), (913, 579), (1030, 572), (998, 561), (835, 545), (1055, 503),
(1134, 495), (889, 495), (1093, 495), (837, 475), (969, 419), (977, 316)]
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0]
[23, 24, 0, 26, 5, 4, 3, 2, 19, 1, 25, 22, 14, 6, 20, 10, 11, 21, 29, 18, 13, 12, 28, 7, 9, 17, 8,
27, 15, 16]
['Entrance hall', 'Unit 1', 'Unit 2', 'Unit 3', 'Unit 4', 'Unit 5', 'UnitNodeA', 'Unit 6', 'Unit 7',
'Unit 8', 'Unit 9', 'Unit 10', 'UnitNodeB', 'Nursing station', 'Medical room', 'Cafeteria',
'Kitchen', 'Laundry', 'Cleaning room', 'Convenient store', 'Activity room', 'Gym', 'Multi-media
hall', 'Office 1', 'Office 2', 'Toilet 1', 'Toilet 2', 'Equipment room', 'Storage', 'Logistic
entrance']
[80, 32, 32, 32, 32, 32, 1, 32, 32, 32, 32, 32, 1, 20, 32, 120, 120, 40, 12, 40, 60, 60, 120, 24, 24,
35, 35, 50, 24, 20]
[5, 5, 5, 5, 5, 5, 0, 5, 5, 5, 5, 5, 0, 5, 5, 8, 3, 3, 3, 3, 8, 3, 3, 5, 5, 0, 0, 0, 0, 0]
['S', 'N', 'N', 'N', 'N', 'N', 'none', 'N', 'N', 'N', 'N', 'N', 'none', 'S', 'S', 'N', 'W', 'E', 'E',
'W', 'N', 'E', 'W', 'N', 'N', 'S', 'S', 'S', 'S', 'S']
[['Unit 1', 'Unit 2', 'Unit 3', 'Unit 4', 'Unit 5', 'UnitNodeA', 'Unit 1'], ['Unit 6', 'Unit 7',
'Unit 8', 'Unit 9', 'Unit 10', 'UnitNodeB', 'Unit 6'], ['UnitNodeA', 'Toilet 1', 'Entrance hall',
'Convenient store', 'Activity room', 'Gym', 'Multi-media hall', 'UnitNodeA'], ['UnitNodeA', 'Toilet
1', 'Entrance hall', 'Convenient store', 'Activity room', 'Gym', 'Multi-media hall', 'UnitNodeA'],
['Cleaning room', 'UnitNodeA', 'UnitNodeB', 'Laundry', 'Cleaning room'], ['Nursing station', 'Medical
room', 'UnitNodeA', 'UnitNodeB', 'Nursing station'], ['Entrance hall', 'Office 1', 'Office 2',
'Toilet 2', 'Entrance hall'], ['Kitchen', 'Cafeteria', 'UnitNodeA', 'UnitNodeB', 'Kitchen'],
['Logistic entrance', 'Cleaning room', 'Equipment room', 'Storage', 'Logistic entrance']]
[5, 5, 3, 3, 4, 4, 3, 2, 1]
```

05_daylight_hour_optimization_module



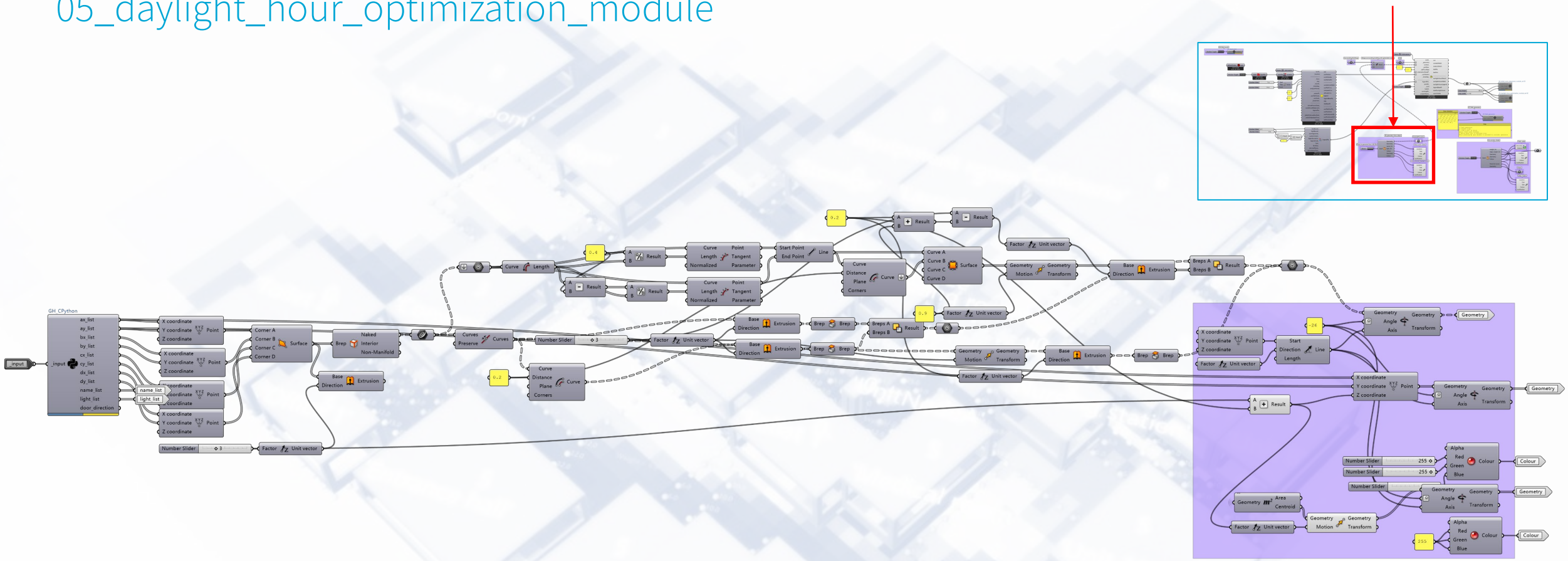
05_daylight_hour_optimization_module



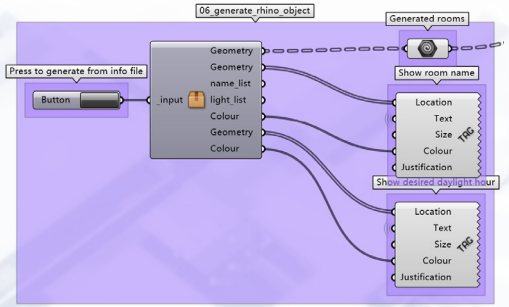


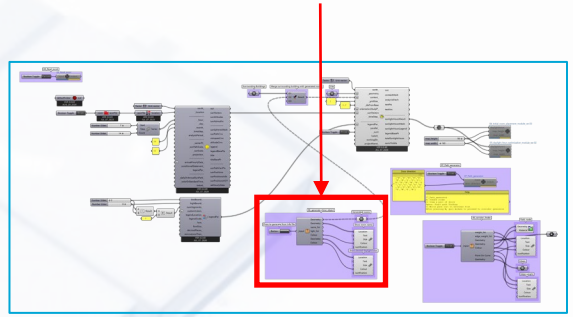
5. Test Case Design

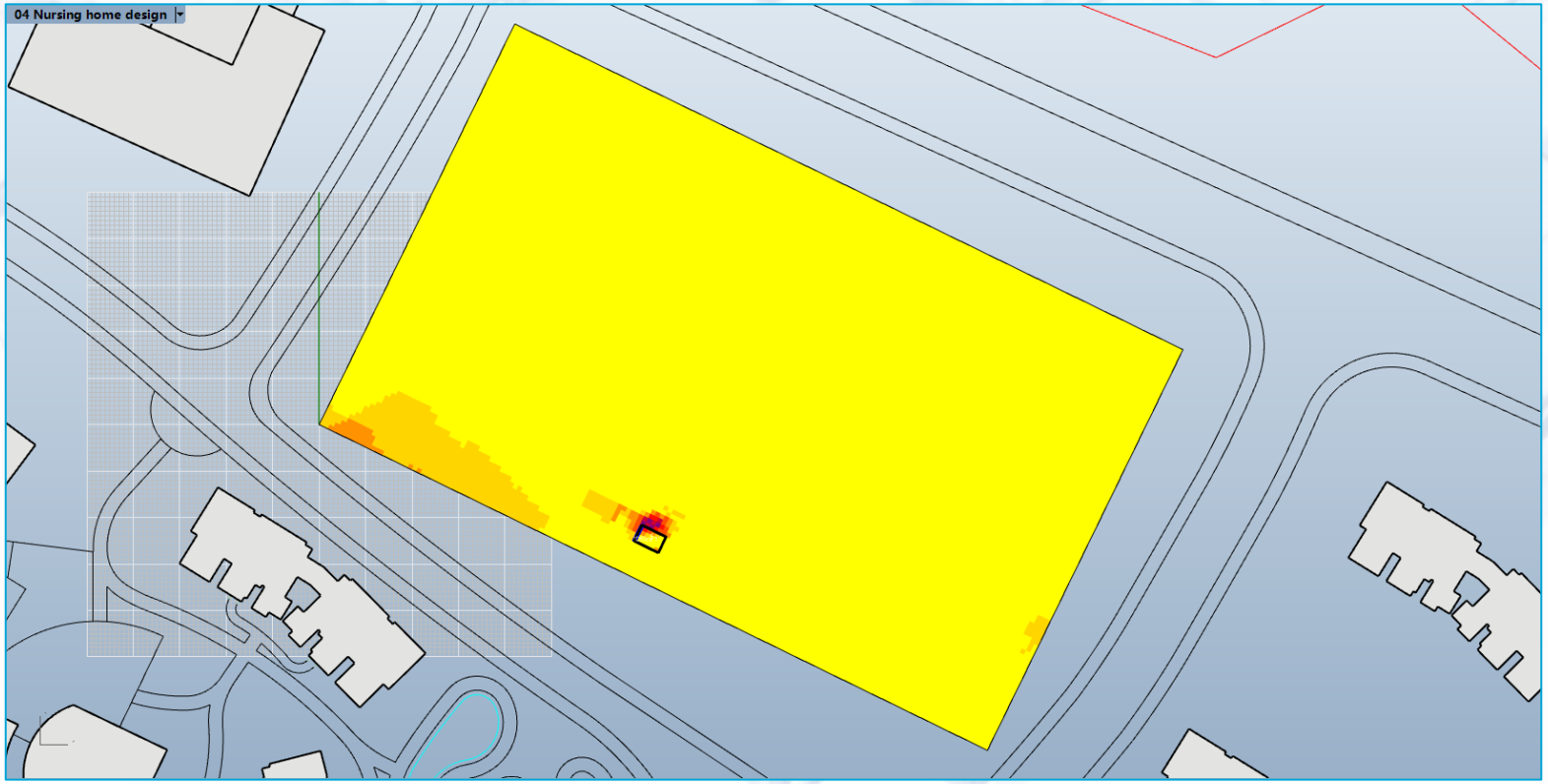
05_daylight_hour_optimization_module



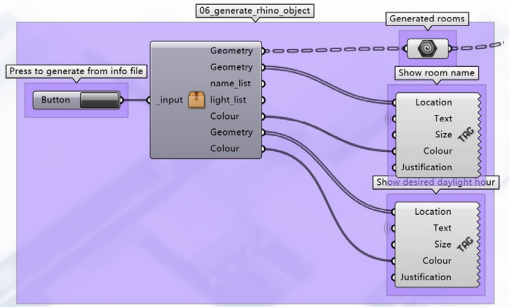
05_daylight_hour_optimization_module

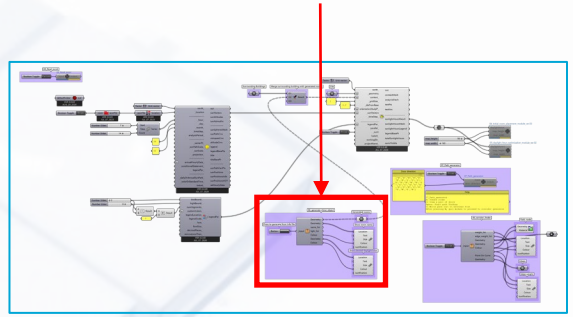


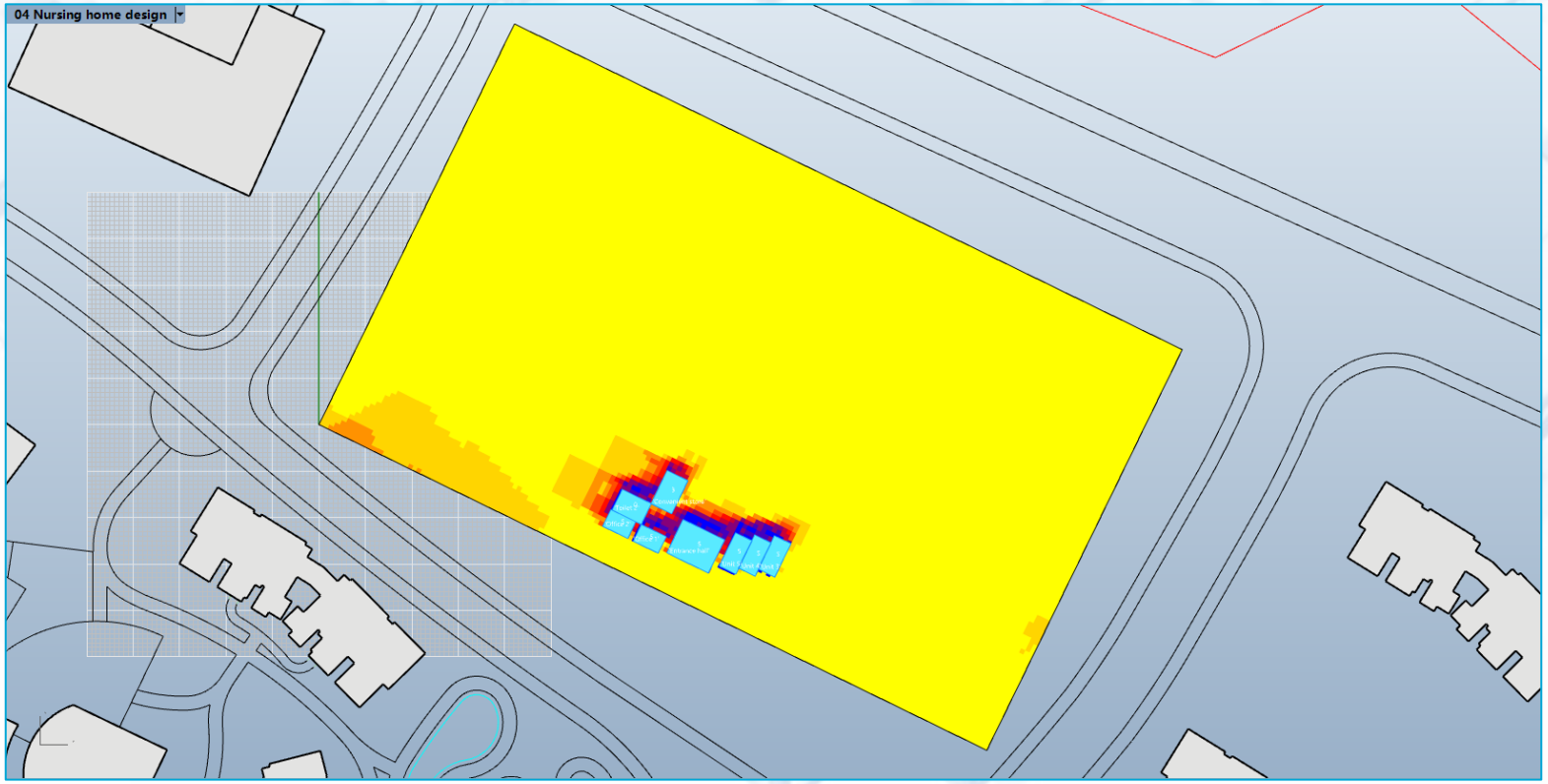




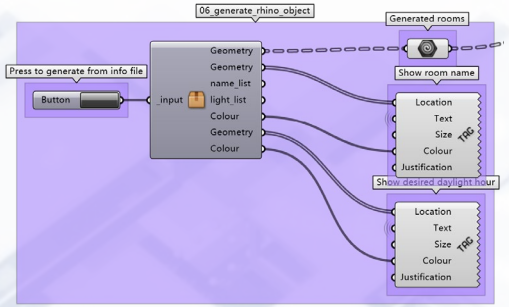
05_daylight_hour_optimization_module

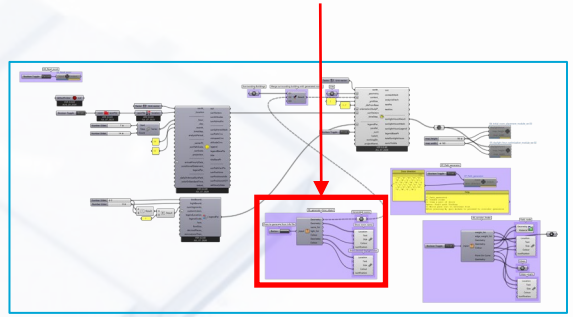


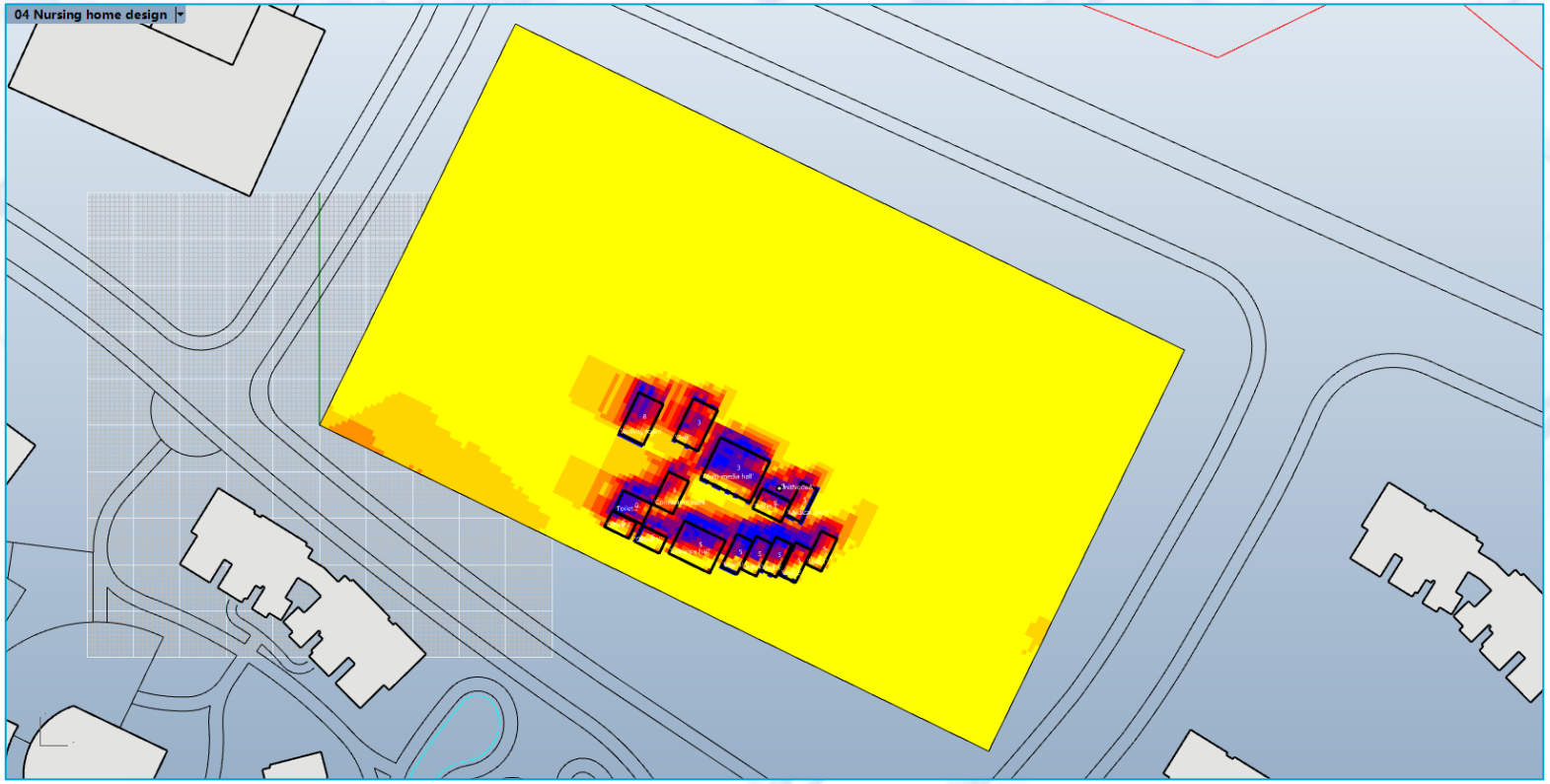




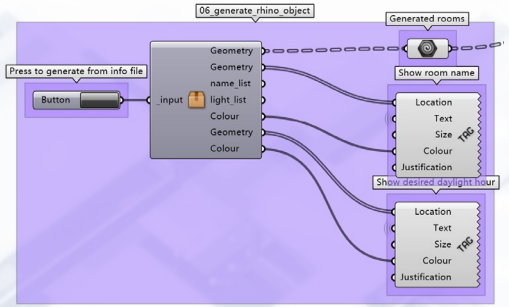
05_daylight_hour_optimization_module

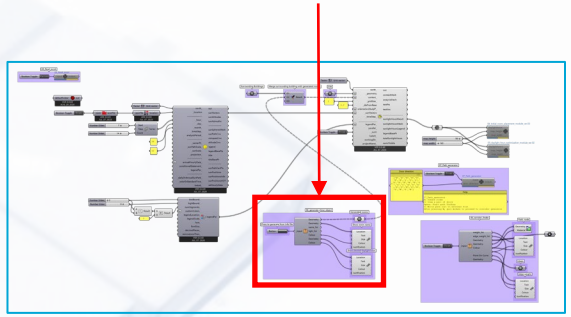


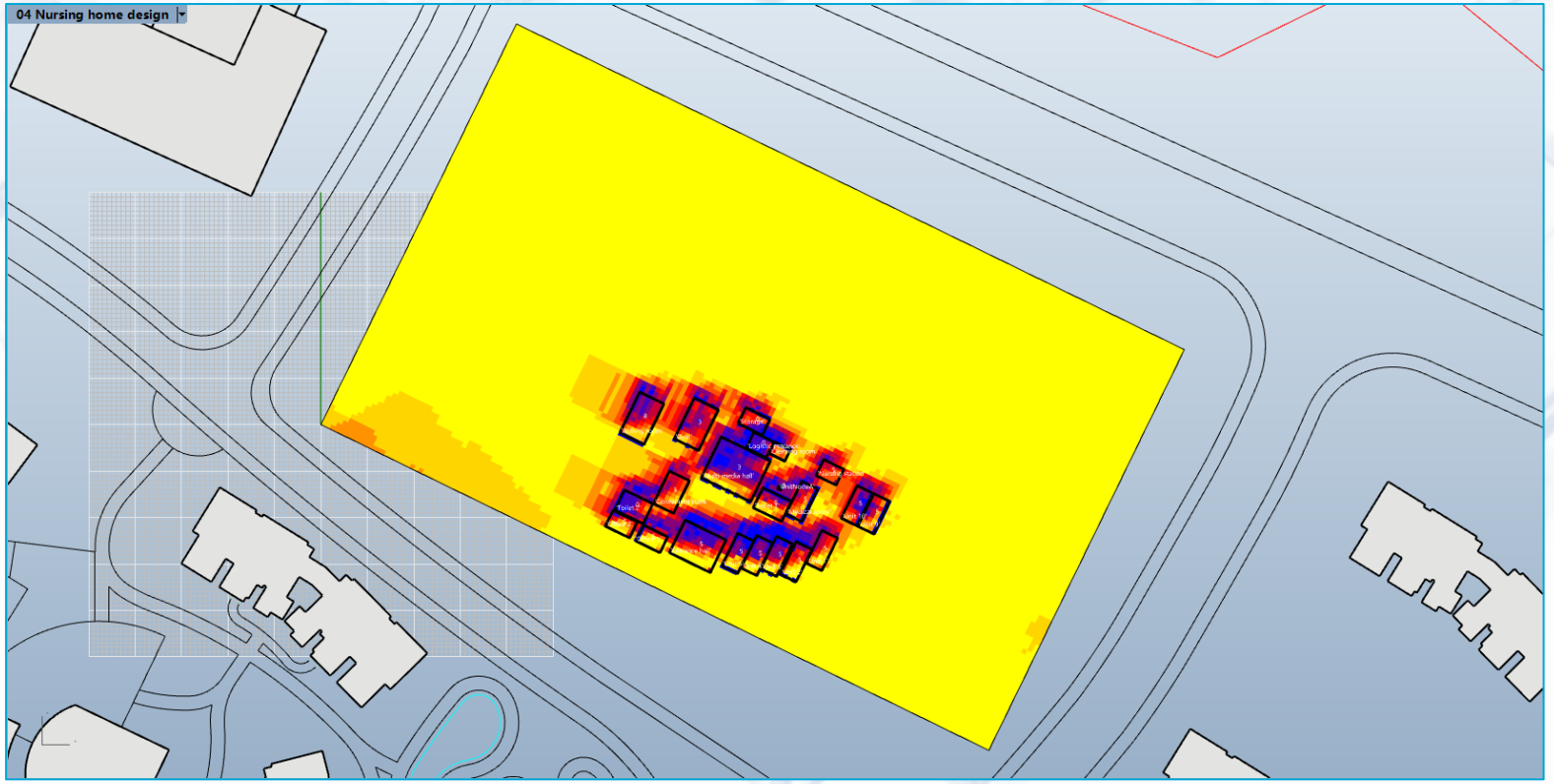




05_daylight_hour_optimization_module

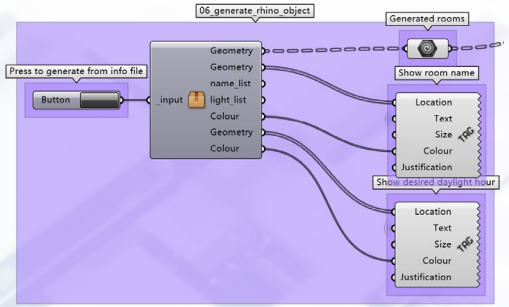


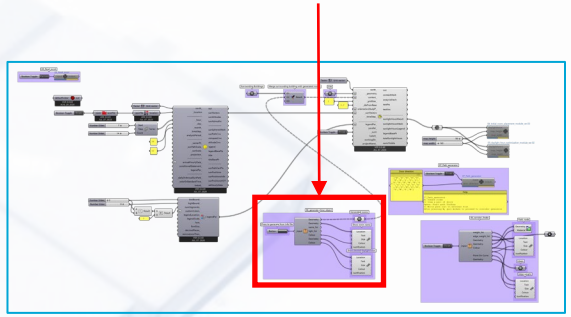


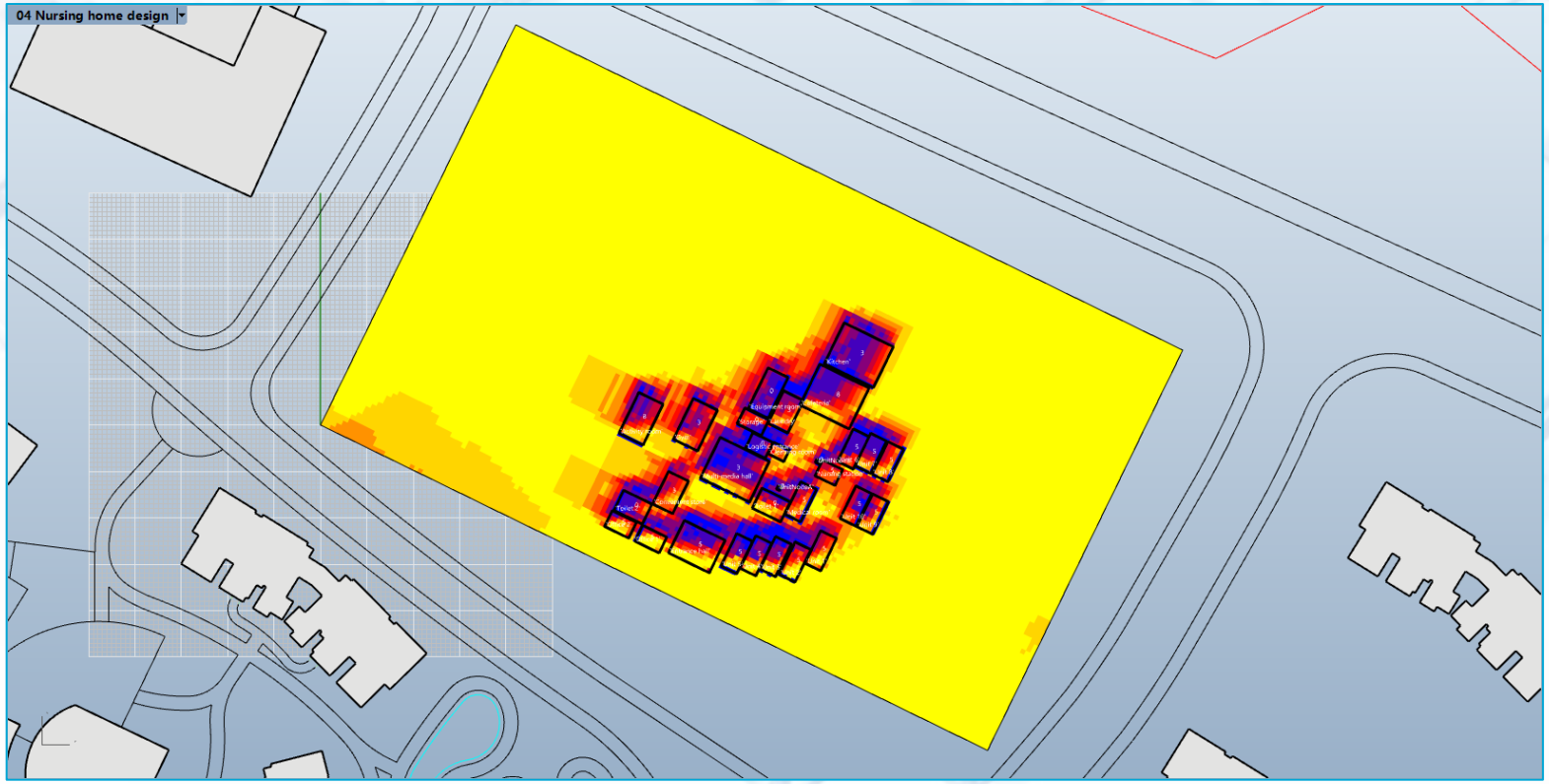


5. Test Case Design

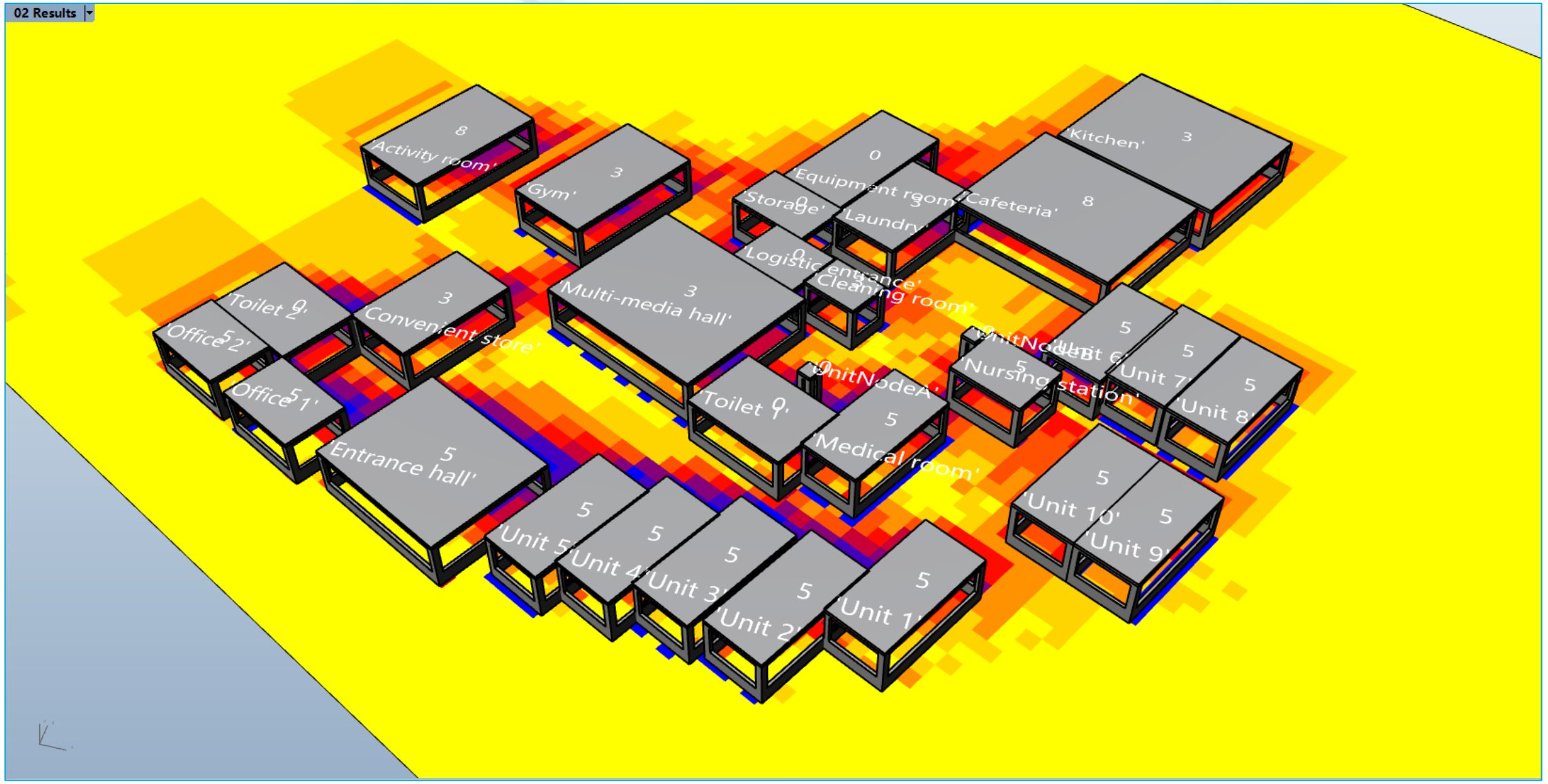
05_daylight_hour_optimization_module



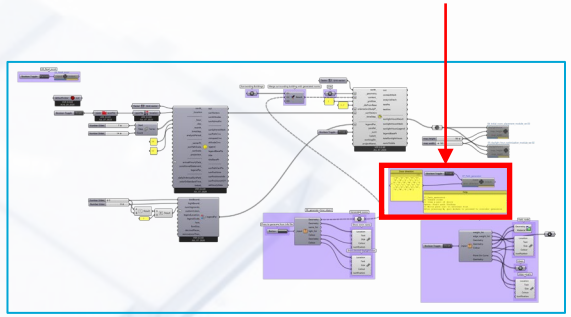


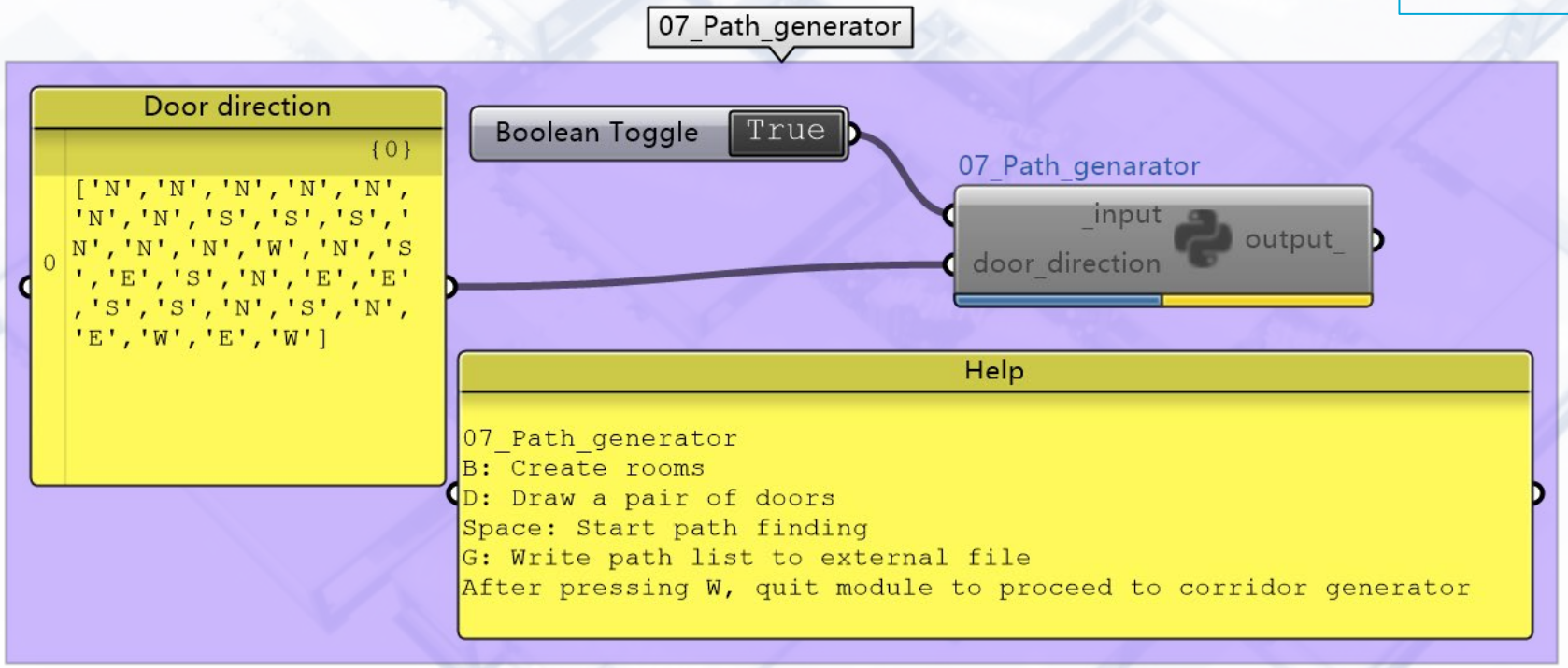


05_daylight_hour_optimization_module



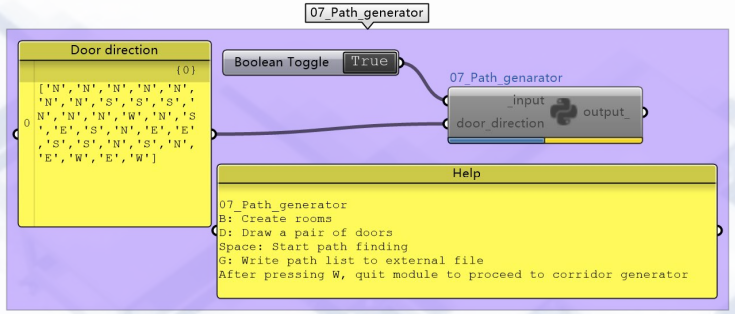
7_path_info_exchange

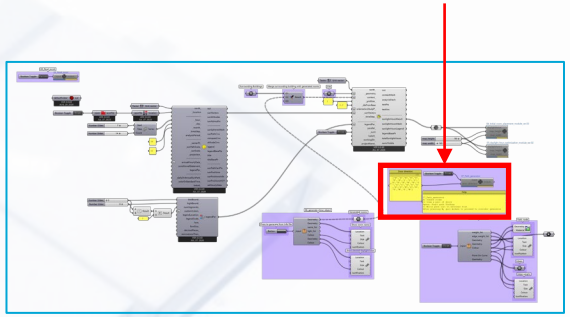




5. Test Case Design

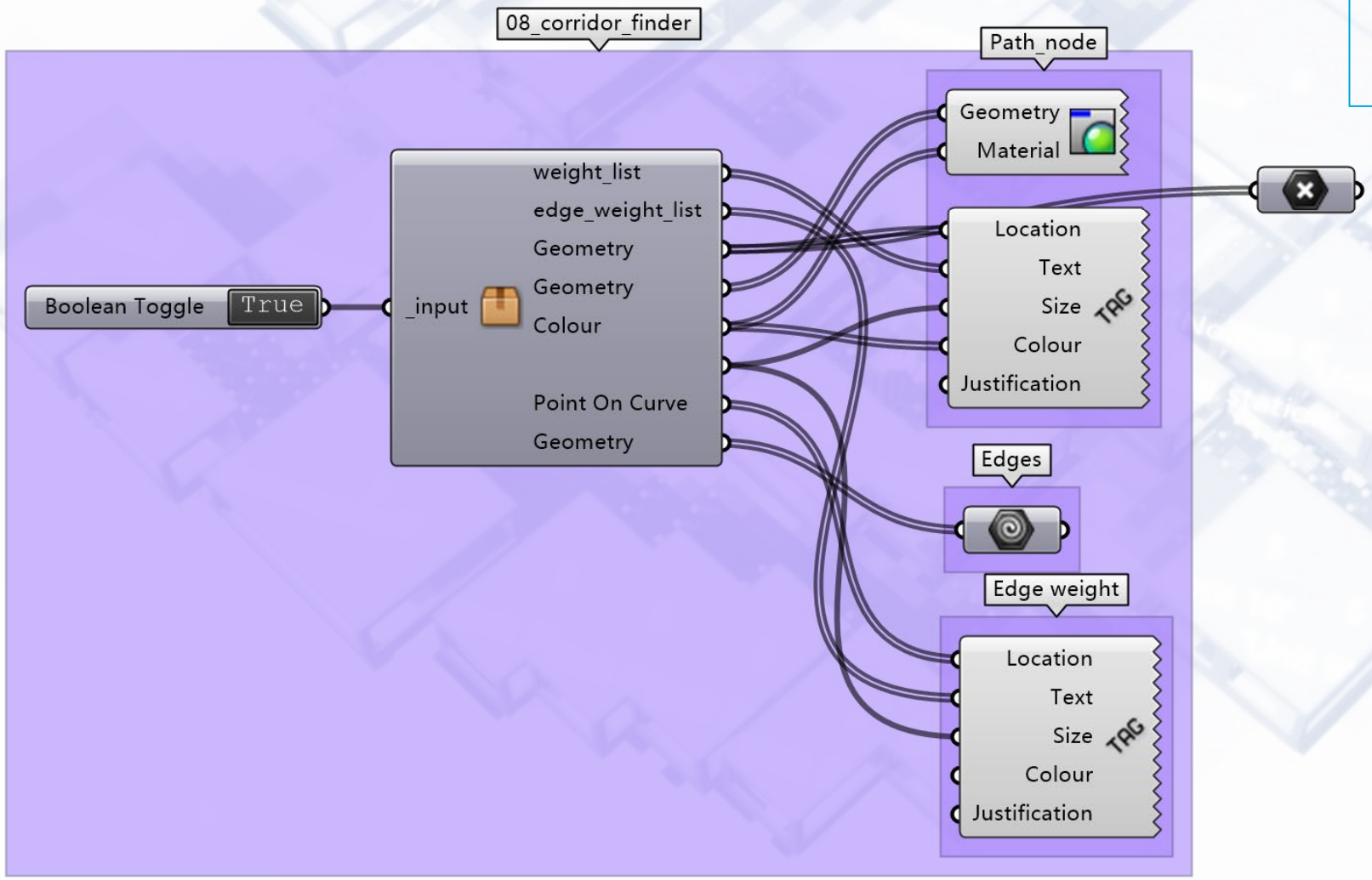
7_path_info_exchange

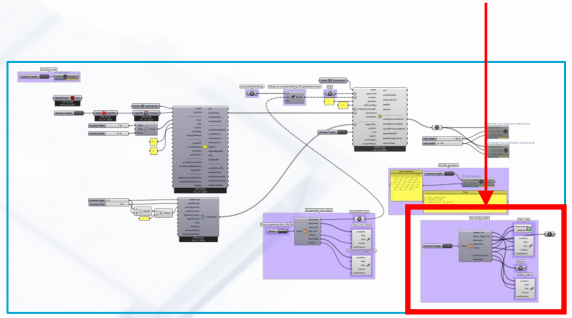




```
[[(67, 88, 3.0), (68, 88, 3.0), (69, 88, 3.0), (70, 88, 3.0), (71, 88, 3.0), (72, 88, 3.0), (73, 88,
3.0), (74, 88, 3.0), (75, 83, 3.0), (75, 88, 3.0), (76, 83, 3.0), (76, 88, 3.0), (77, 83, 3.0), (77,
84, 3.0), (77, 85, 3.0), (77, 86, 3.0), (77, 87, 3.0), (77, 88, 3.0)], [(68, 88, 3.0), (69, 88, 3.0),
(70, 88, 3.0), (71, 88, 3.0), (72, 88, 3.0), (73, 88, 3.0), (74, 83, 3.0), (74, 88, 3.0), (75, 83,
3.0), (75, 88, 3.0), (76, 83, 3.0), (76, 88, 3.0), (77, 83, 3.0), (77, 84, 3.0), (77, 85, 3.0), (77,
86, 3.0), (77, 87, 3.0), (77, 88, 3.0)], [(74, 80, 3.0), (74, 81, 3.0), (74, 82, 3.0), (75, 80, 3.0),
(76, 79, 3.0), (76, 80, 3.0), (77, 79, 3.0), (78, 79, 3.0), (79, 79, 3.0), (80, 79, 3.0), (81, 79,
3.0), (82, 79, 3.0), (83, 79, 3.0), (84, 79, 3.0)], [(74, 83, 3.0), (75, 83, 3.0), (76, 83, 3.0),
(77, 83, 3.0), (78, 79, 3.0), (78, 80, 3.0), (78, 81, 3.0), (78, 82, 3.0), (78, 83, 3.0), (79, 79,
3.0), (80, 79, 3.0), (81, 79, 3.0), (82, 79, 3.0), (83, 79, 3.0)], [(71, 81, 3.0), (72, 81, 3.0),
(73, 81, 3.0), (74, 81, 3.0), (75, 81, 3.0), (76, 81, 3.0), (77, 81, 3.0), (78, 79, 3.0), (78, 80,
3.0), (78, 81, 3.0), (79, 79, 3.0), (80, 79, 3.0), (81, 79, 3.0), (82, 79, 3.0), (83, 79, 3.0)],
[(71, 80, 3.0), (72, 80, 3.0), (73, 80, 3.0), (74, 80, 3.0), (75, 80, 3.0), (76, 79, 3.0), (76, 80,
3.0), (77, 79, 3.0), (78, 79, 3.0), (79, 79, 3.0), (80, 79, 3.0), (81, 79, 3.0), (82, 79, 3.0), (83,
79, 3.0), (84, 79, 3.0)], [(68, 88, 3.0), (69, 88, 3.0), (70, 88, 3.0), (71, 81, 3.0), (71, 82, 3.0),
(71, 83, 3.0), (71, 88, 3.0), (72, 83, 3.0), (72, 88, 3.0), (73, 83, 3.0), (73, 88, 3.0), (74, 83,
3.0), (74, 88, 3.0), (75, 83, 3.0), (75, 88, 3.0), (76, 83, 3.0), (76, 88, 3.0), (77, 83, 3.0), (77,
84, 3.0), (77, 85, 3.0), (77, 86, 3.0), (77, 87, 3.0), (77, 88, 3.0)], [(67, 88, 3.0), (68, 88, 3.0),
(69, 88, 3.0), (70, 88, 3.0), (71, 82, 3.0), (71, 83, 3.0), (71, 88, 3.0), (72, 83, 3.0), (72, 88,
3.0), (73, 83, 3.0), (73, 88, 3.0), (74, 83, 3.0), (74, 88, 3.0), (75, 83, 3.0), (75, 88, 3.0), (76,
83, 3.0), (76, 88, 3.0), (77, 83, 3.0), (77, 84, 3.0), (77, 85, 3.0), (77, 86, 3.0), (77, 87, 3.0),
(77, 88, 3.0)], [(92, 75, 5.0), (93, 75, 5.0), (94, 75, 5.0), (95, 75, 5.0), (96, 75, 5.0)], [(92,
76, 5.0), (93, 75, 5.0), (93, 76, 5.0), (94, 75, 5.0), (95, 75, 5.0)], [(96, 73, 5.0), (96, 74, 5.0),
(97, 73, 5.0), (98, 73, 5.0), (99, 73, 5.0), (100, 73, 5.0)], [(96, 75, 5.0), (97, 73, 5.0), (97, 74,
```

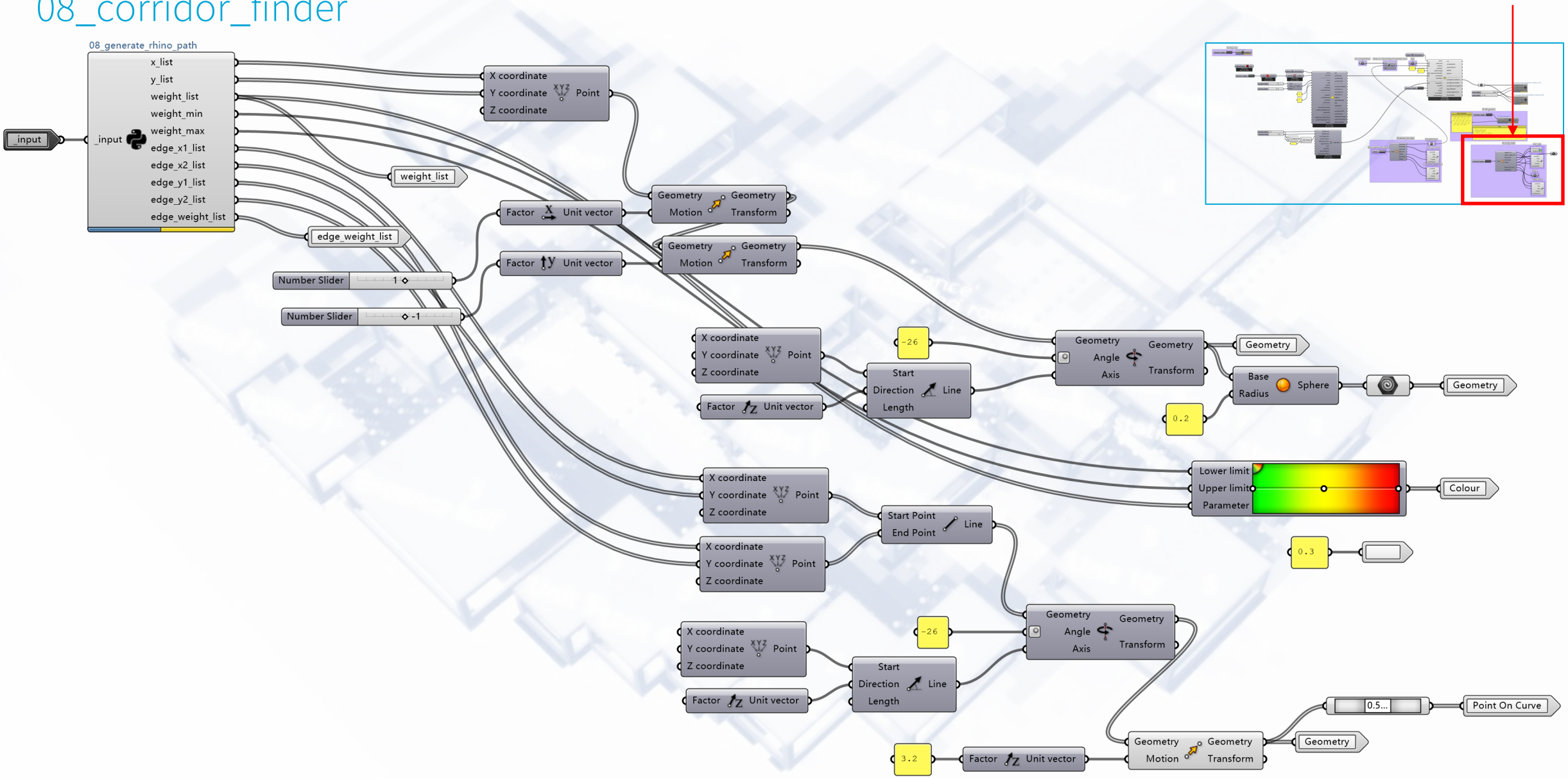
08_corridor_finder





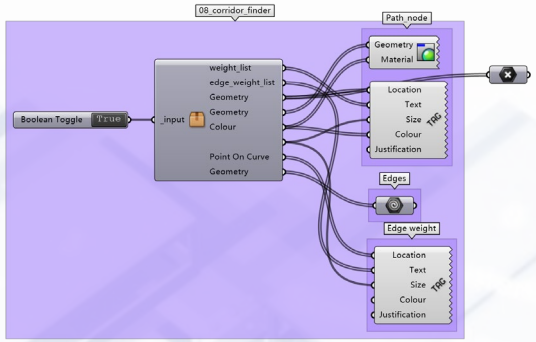
5. Test Case Design

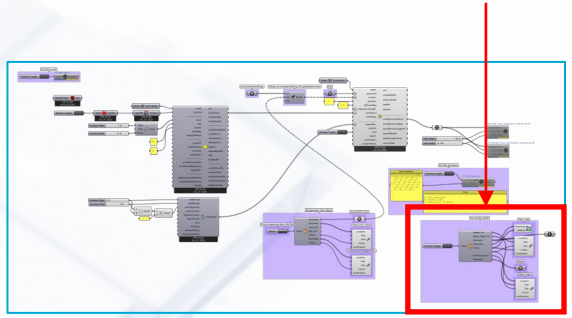
08_corridor_finder

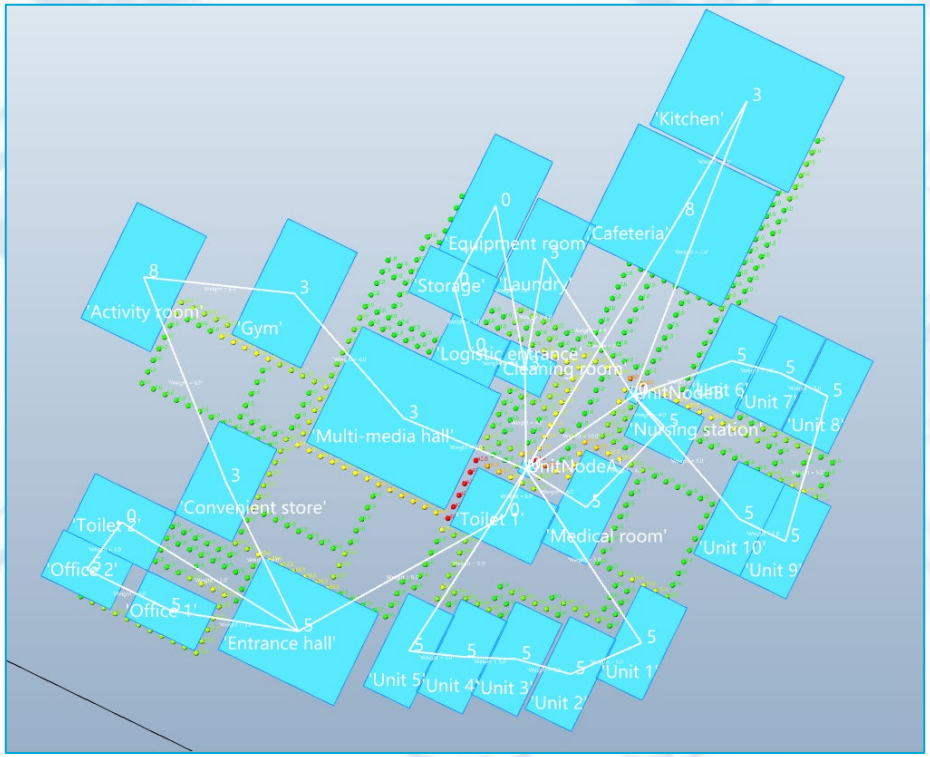


5. Test Case Design

08_corridor_finder

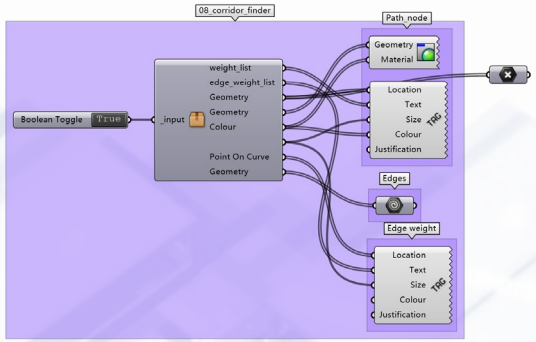


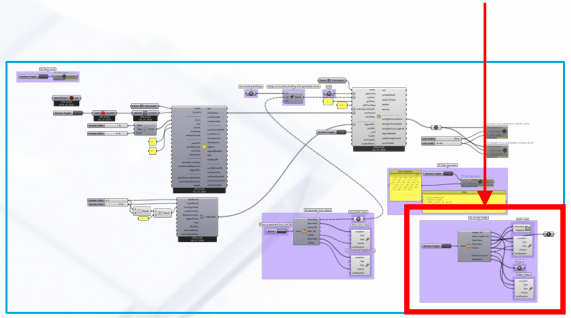


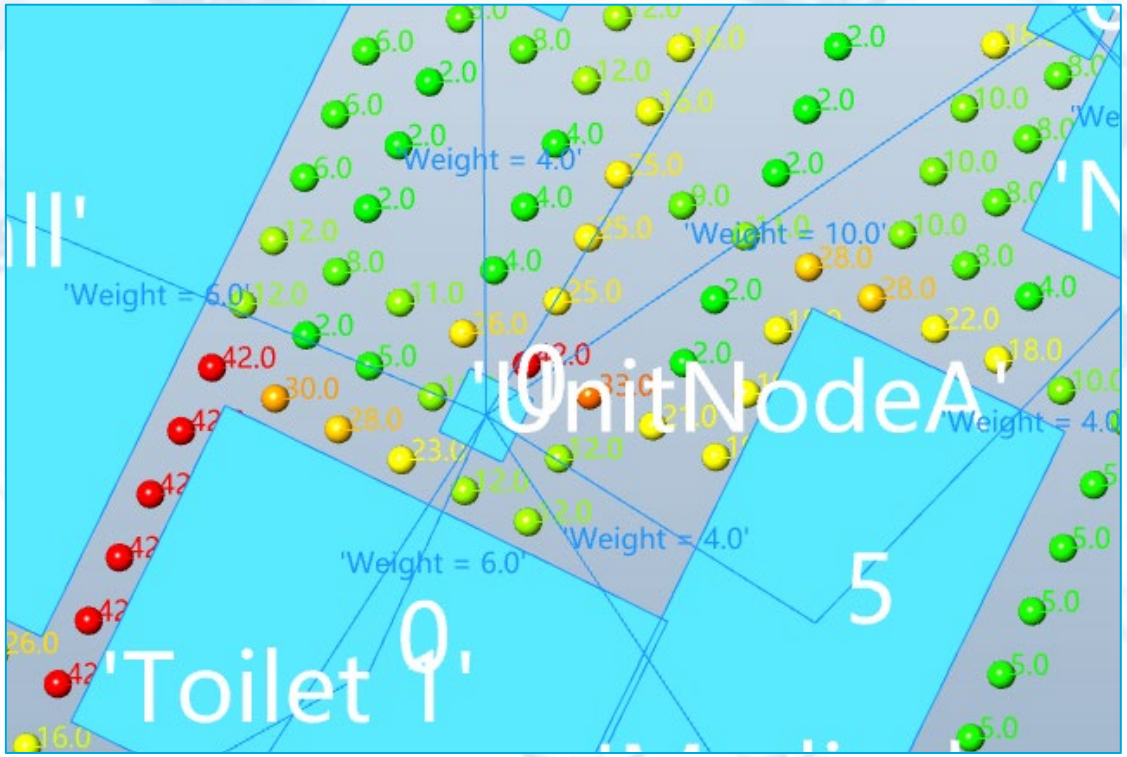


5. Test Case Design

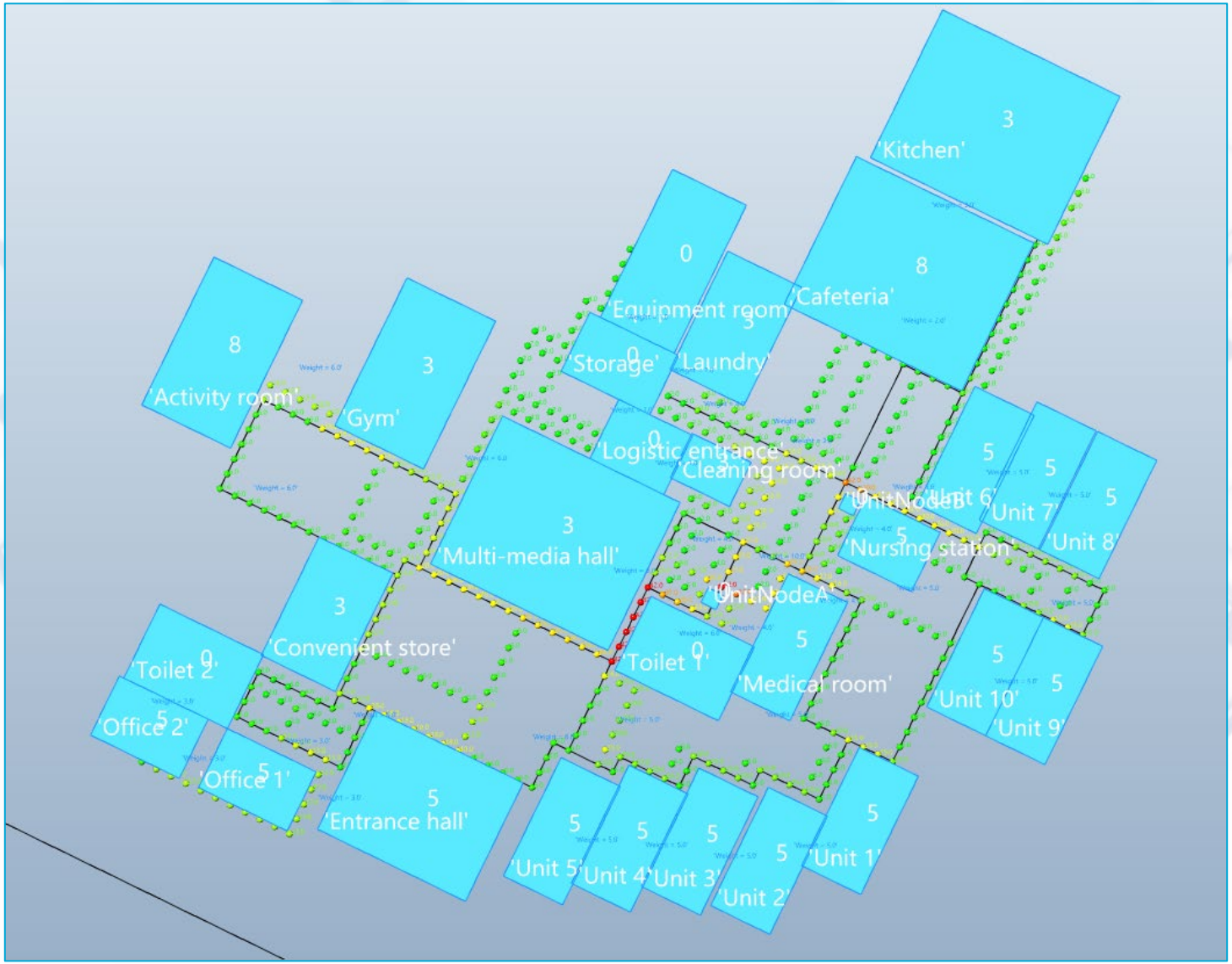
08_corridor_finder



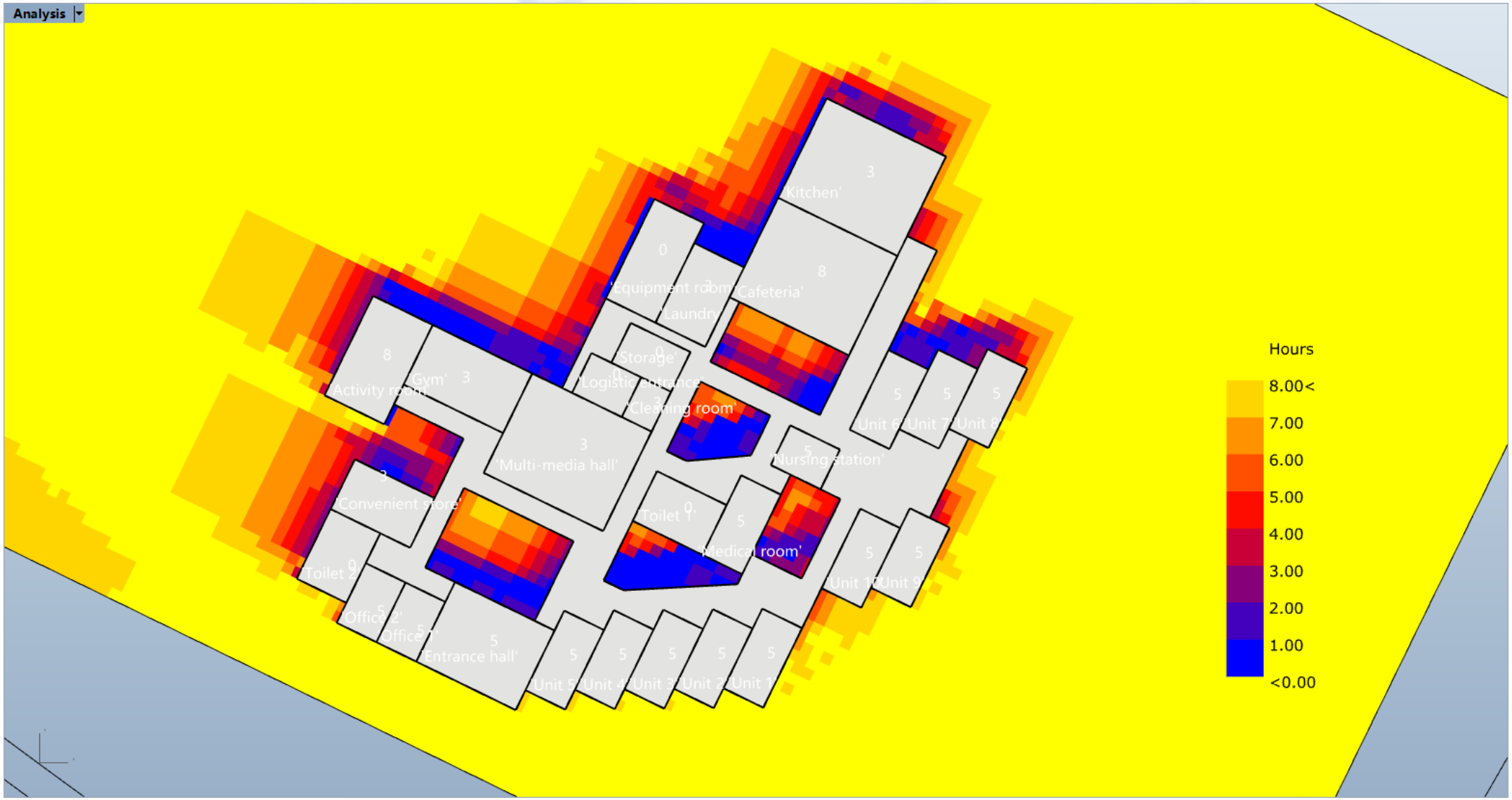


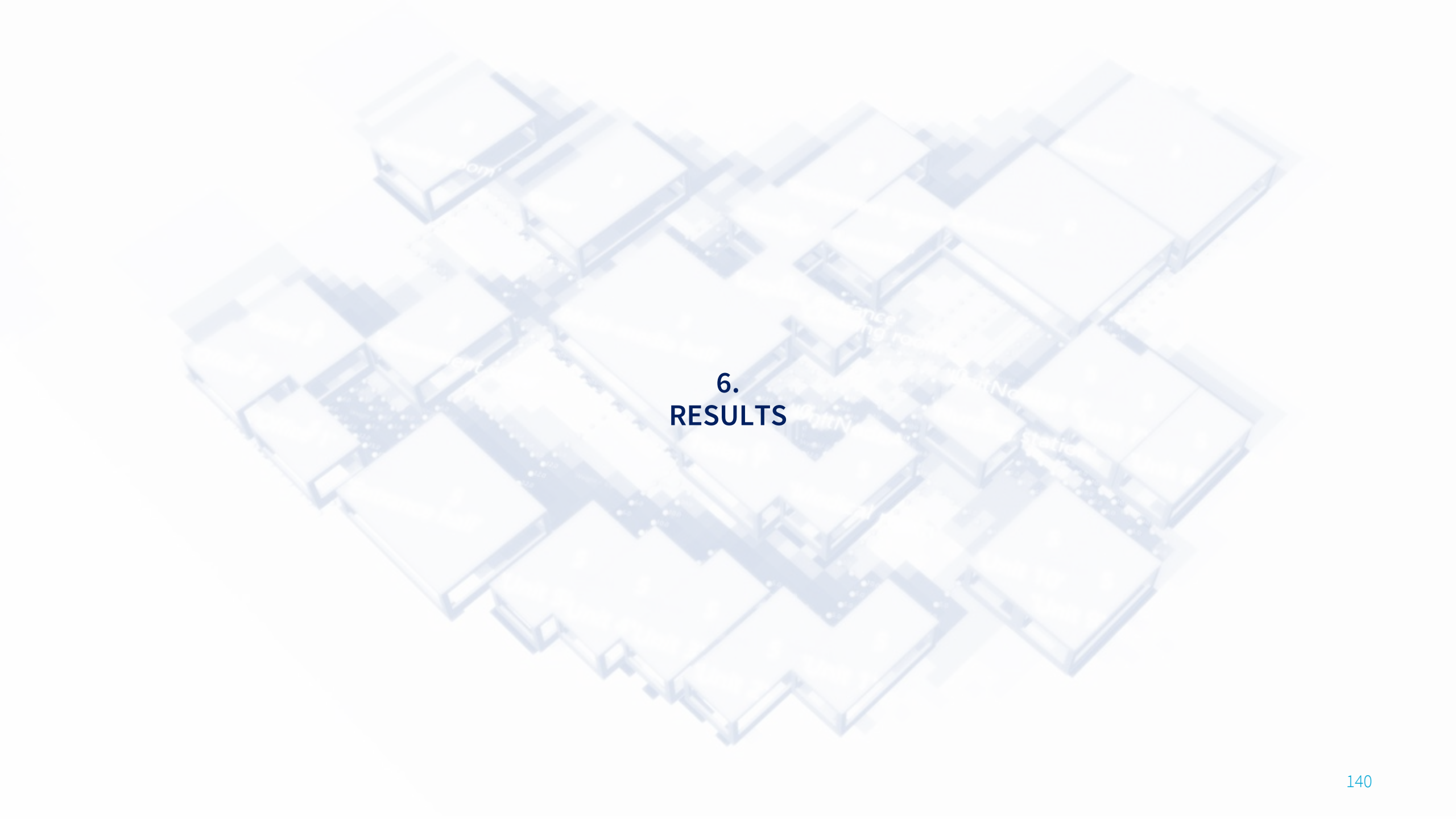


09_manual_path_selection



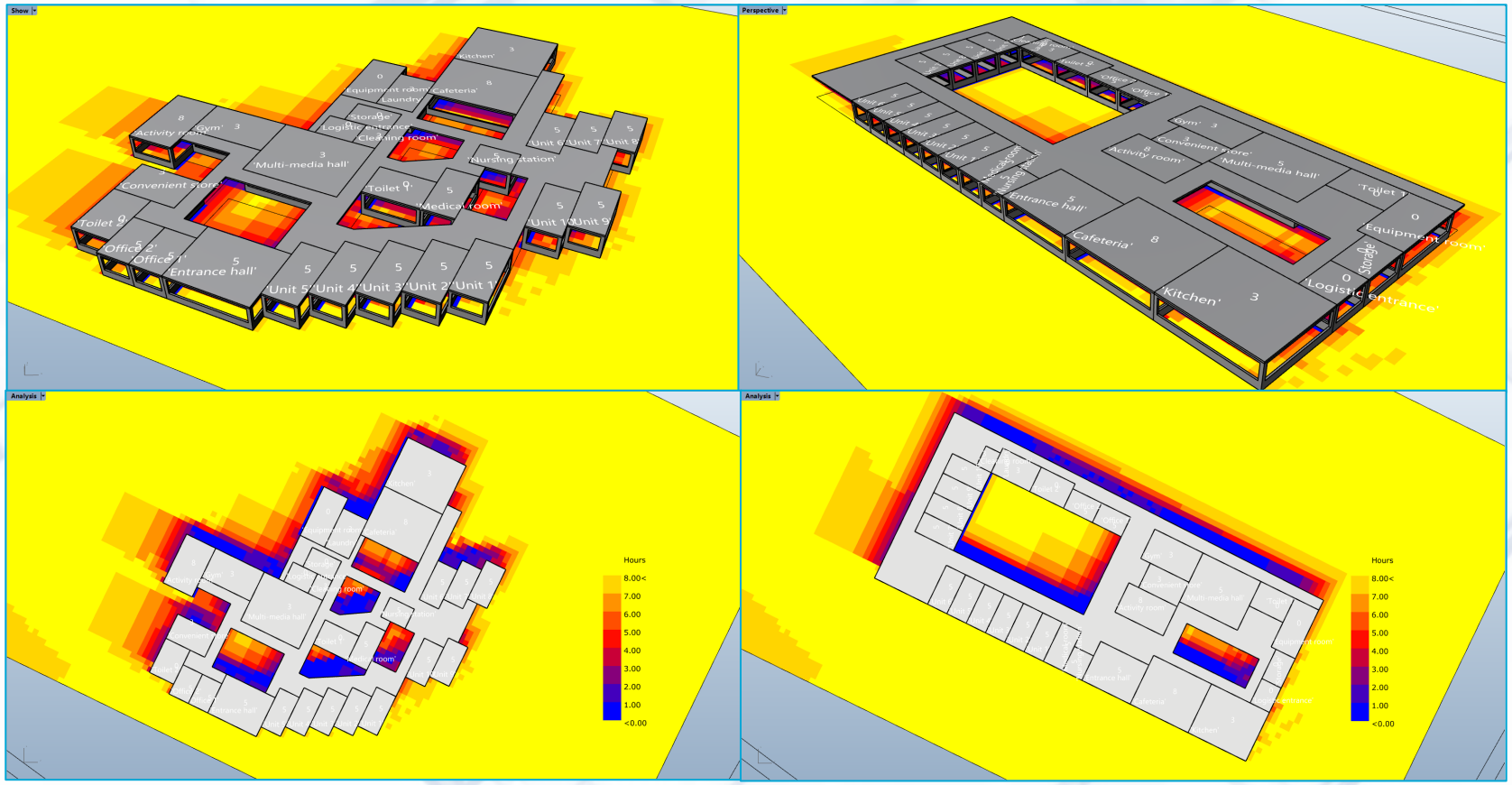
09_manually_adjusted_result



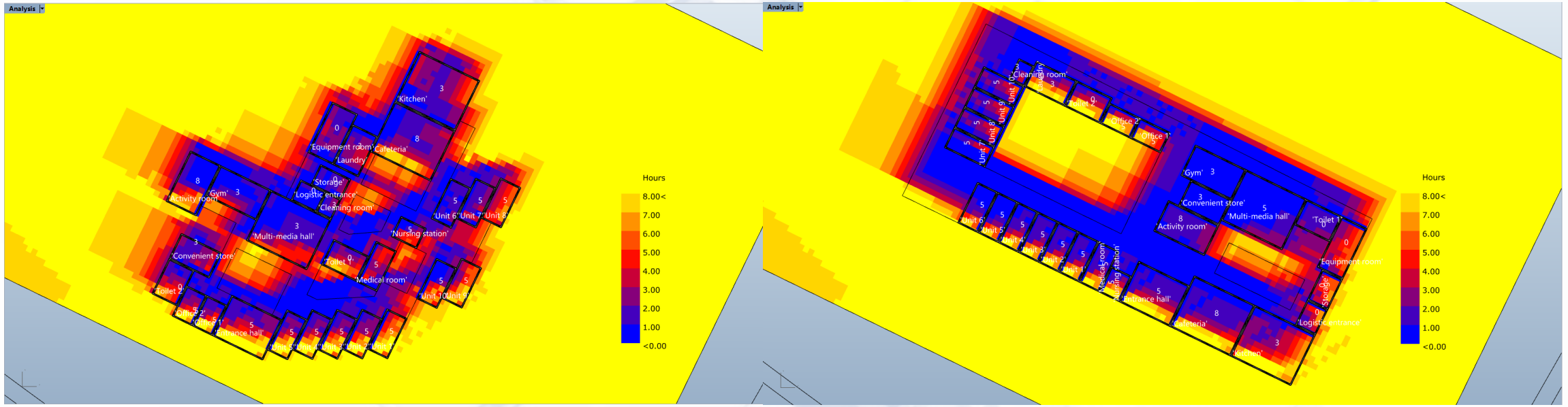


6.
RESULTS

6.1 Evaluation And Comparison With A Manually Designed Result



6.1 Evaluation And Comparison With A Manually Designed Result



6.1 Evaluation And Comparison With A Manually Designed Result

	Designed by the method	**Designed by hand**
Area of convex hull	2583 m^2	2580 m^2
Area of corridor	568 m^2	1084 m^2
Length of complete elderly activity circulation	164.05 m	184.3 m
Length of cleaning staff circulation	103.70 m	137.73 m
Length of medical staff circulation	85.92 m	109.67 m
Length of administrative staff circulation	16.75 m	79.09 m
Length of logistic circulation	33.75 m	163.28 m
Length of food delivery circulation (to each unit)	126.37 m	155.64 m

6.1 Evaluation And Comparison With A Manually Designed Result

6.2 Summary

Comparing the two designs, both ways have their advantages. Manual design technique of designer is more trained, its result appears more regular and convincing. Program-distribution-wise, it is more reasonable, while this method brings more efficiency to the building, brings interesting daylight hour conditions. Some problems this computational method bring can be foreseeable solved. For example, by altering interacting behaviour with the module, the distribution of different program can be regulated to a certain extent, by adding new features to the module, like ‘drawing courtyard’ during the interaction, can create more space in the floor plan for desired daylight accessibility. There are still some problems left, which will be discussed in the next section.

7.
DISCUSSIONS

7.2 Future Work (Limitation And Open Problems)

The first limitation of this methodology is that while this thesis involves optimization theory, graph theory, force-directed methods, computational geometry and daylight hour analysis in some way, the detailed discussion of each subject falls out of the scope of the project. So, the substitutional methods being used for specific purposes might be inaccurate.

7.2 Future Work (Limitation And Open Problems)

Second, the current workflow will cause a problem, that when generating corridors after the daylight hour analysis, the generated corridor is very possible to block the sun and change the daylight condition of the site. This requires a different workflow or method to maybe generate corridors while placing rooms for daylight analysis.

7.2 Future Work (Limitation And Open Problems)

Third, this methodology for now only produces 2D plans, room shapes are set to rectangles, not flexible as real life, the room dimensions are not flexible. A more adaptive room shape generation module can be worked in future.

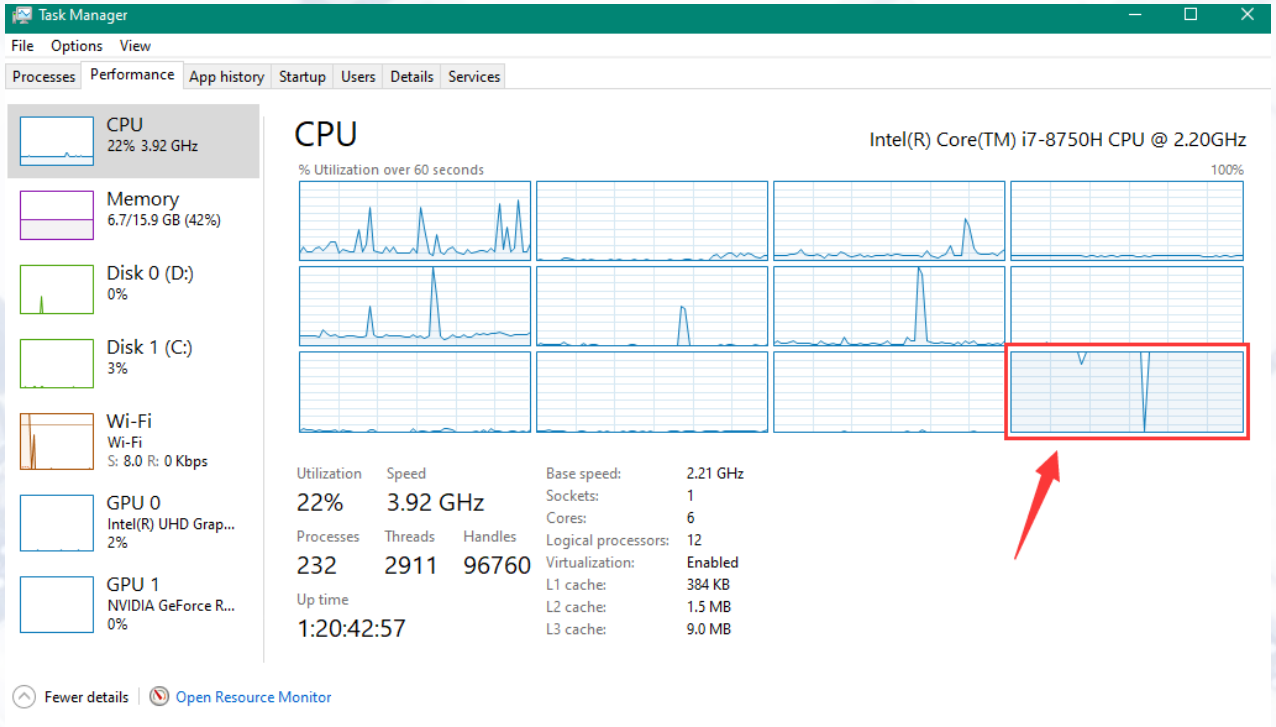
7.2 Future Work (Limitation And Open Problems)

Fourth, in the pathfinding algorithm, the door direction and position are pre-assigned to each room and manually adjusted before the pathfinding process. Therefore, door positions in the results might not be the optimal ones, potentially making the layout and paths not the most optimal ones. The door-position finding module can be added in future for finding optimal door position in a room.

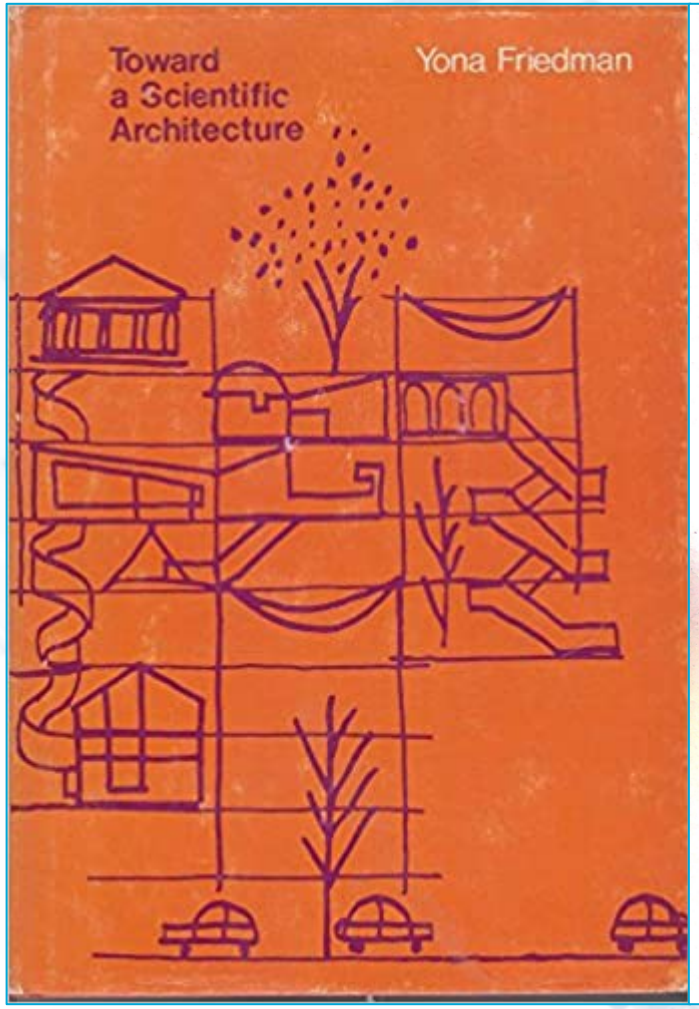
7.2 Future Work (Limitation And Open Problems)

Fifth, the iteration process of the daylight hour analysis is done manually, which is rather labour intensive. A better workflow might be possible by adding looping components to grasshopper. And, the current process requires human intervention, it can be good if a fully automatic option can be offered, in case of a large number of rooms. It requires the update of the algorithm on the room placement method, where more mathematics can kick in. And, current indicators are generally single-room there is a lack of an overall daylight hour evaluation of the whole floor plan

7.2 Future Work (Limitation And Open Problems)



Sixth, following the fifth, the module is developed in Python 3, with 2d physic engine Pymunk, 2d game library Pygame. One thing it results is that, when the number of rooms and spring joints among rooms goes up, the general CPU occupation stays low while single-core occupation goes rocket high, and the frame rate drops significantly, for example, during the nursing home design, it is around 3 – 7 fps. So more efficient alternative platform or tools shall be sought in future developments.



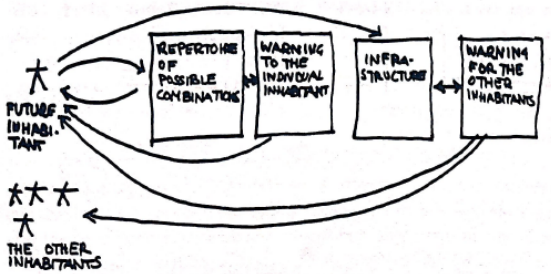


Figure 28

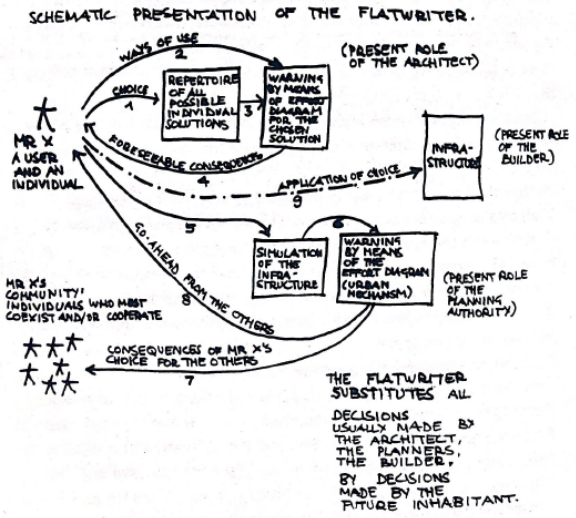


Figure 29

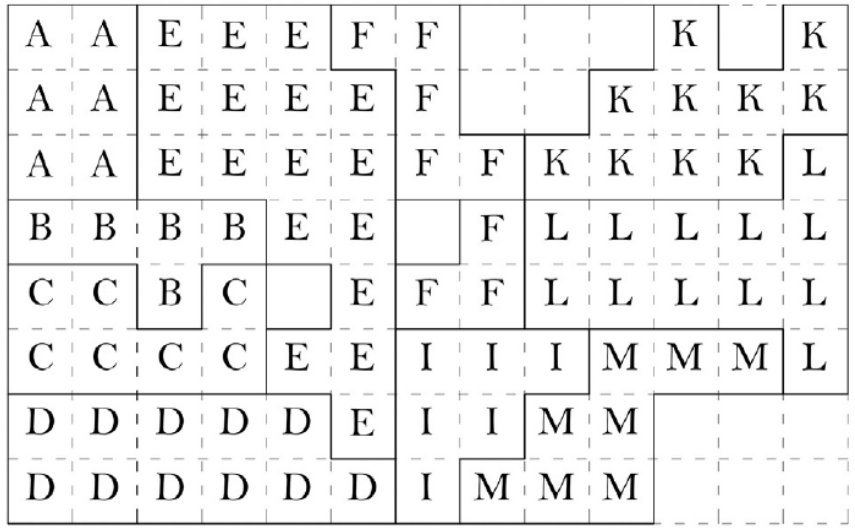


Fig. 5 A layout generated using Quadratic Assignment (adapted from: Montreuil, 1990).

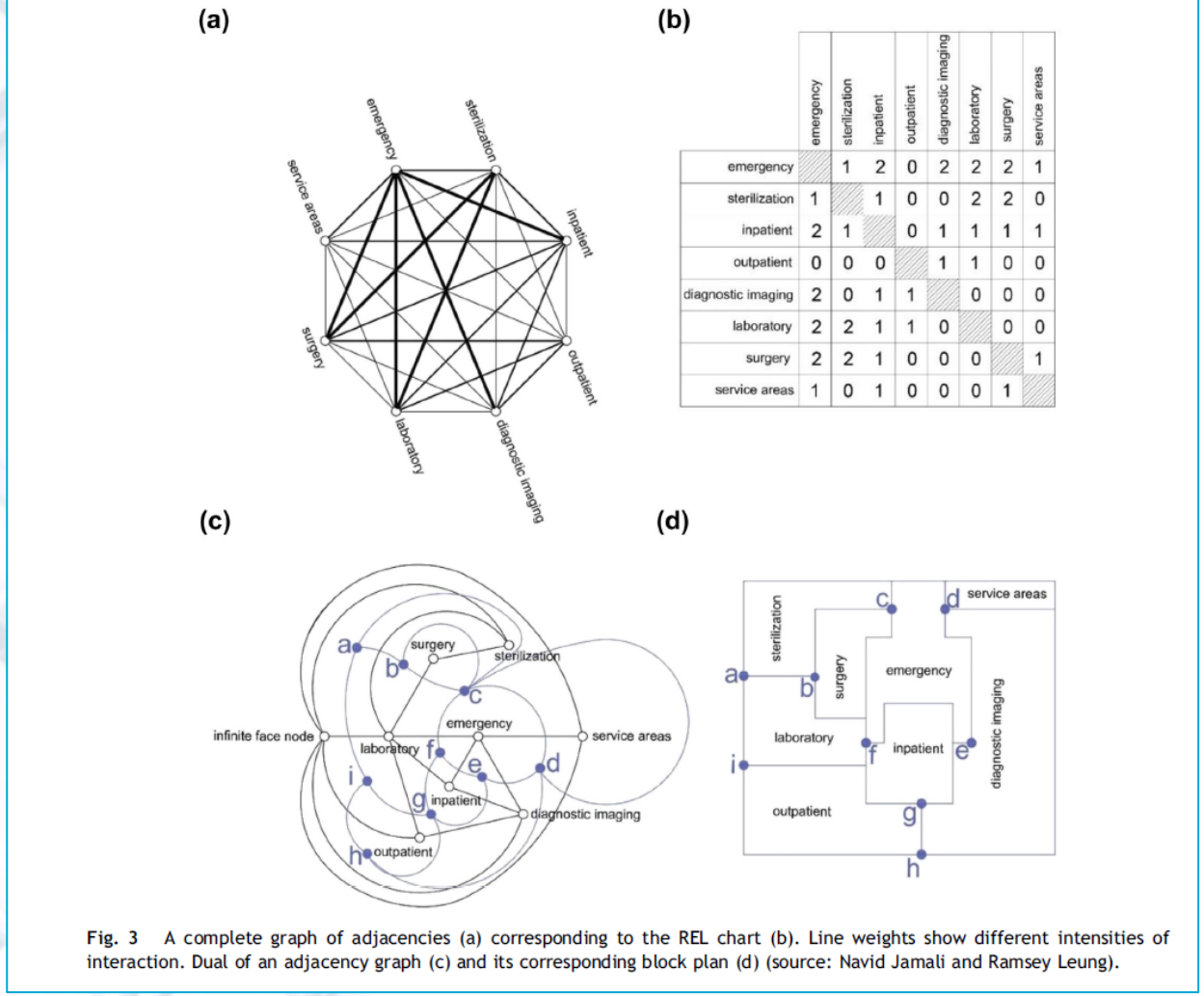
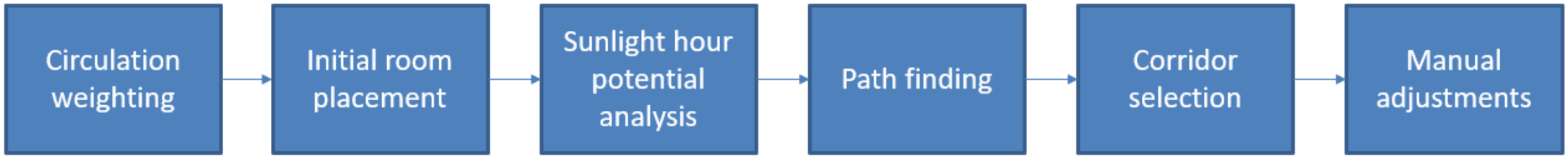
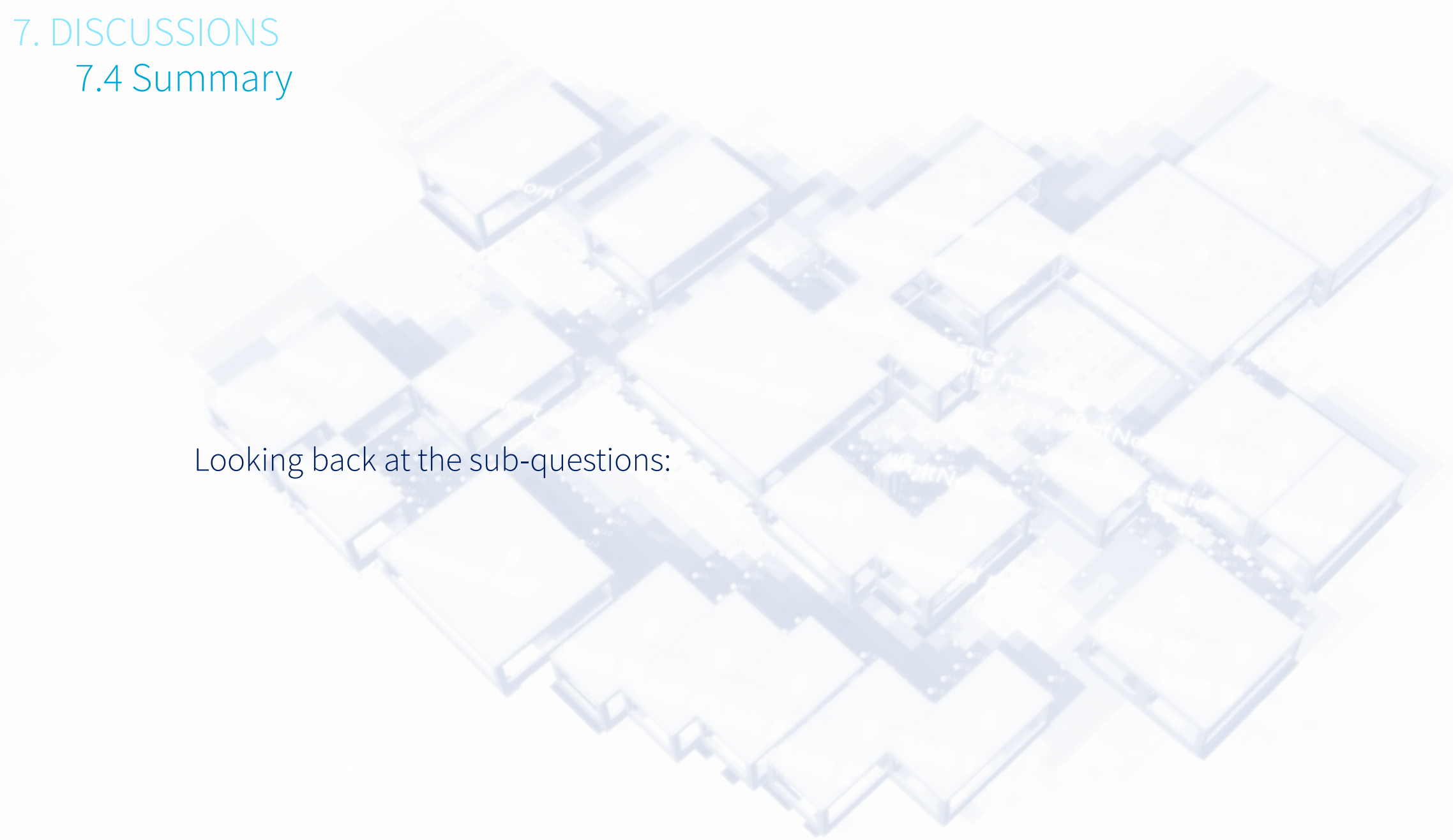


Fig. 3 A complete graph of adjacencies (a) corresponding to the REL chart (b). Line weights show different intensities of interaction. Dual of an adjacency graph (c) and its corresponding block plan (d) (source: Navid Jamali and Ramsey Leung).

7.3 Discussion



7.4 Summary



Looking back at the sub-questions:

7.4 Summary

- How should the program of requirements be modelled into spatial relations?

As illustrations like bubble diagrams have been widely used and accepted by designers, the graph can be a good and intuitive carrier of architectural information. In a design that considers circulation efficiency, circulation relations can be represented by edges of the graph.

7.4 Summary

- How can spatial relations be projected into an actual space?

Force-directed method can be an intuitive method to visualize spatial relations and project it into an actual space.

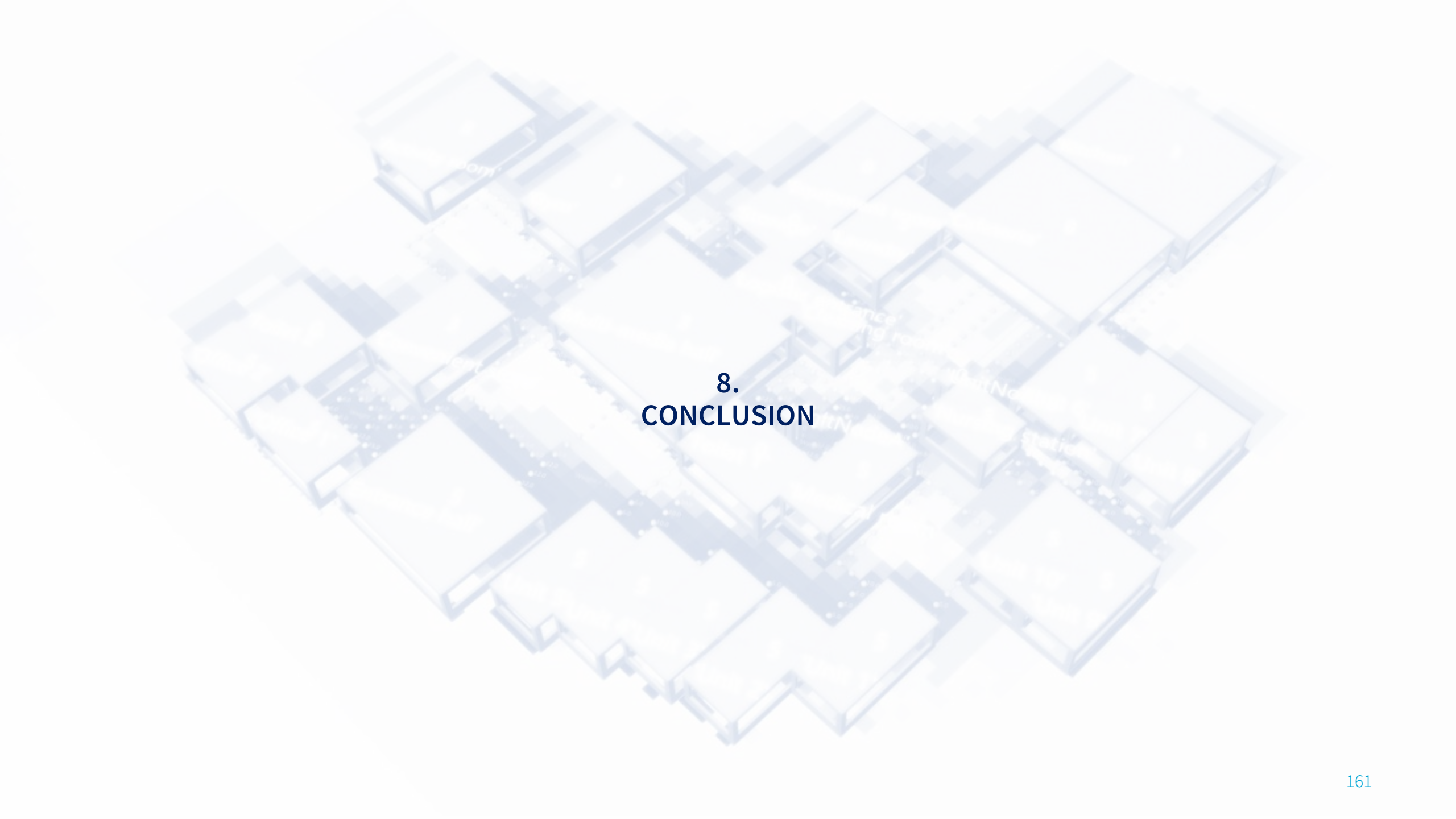
7.4 Summary

- How should the relations among evaluating circulations, walking distance and daylight hour be? In another word, what is the hierarchy?

In my design, I put circulation evaluation in front of the daylight hour analysis, then after daylight hour analysis, I generated potential walking nodes with pathfinding algorithm. The result showed some disadvantages, but that is mainly due to the execution of room placement order, which needs to be improved. Daylight hour analysis should be more integrated into the previous process.

7.4 Summary

- How should designers interact with the computational process? What is his/her role? After having a full understanding of how the process work, designers can actively intervene the computation progress with his/her own thinking/consideration for a more humanity-wise ideal result, and like Friedman's vision, determining a convincing result at the end.



8.
CONCLUSION

This methodology from a certain extent can be a supplementary method to the early stage of an architecture design workflow. It can help designers make fast decisions and evaluations at the early stage of design without investing too much into it.

While being aware of its limitations, it is applicable for designs with the need of daylight hour optimization, and it is sufficient for designs with the needs for circulation efficiency to a certain extent.

The ideology behind it can be a framework for the future development of such methodologies involving multi-criteria space configuration. Besides working with this methodology on the floor plan level, this method also has the potential of working on a larger scale, where the general layout of a set of buildings is required to consider the daylight.

THANK YOU

LINCHENG JIANG · NOVEMBER 2020

Reference

Abdelrahman, M. (2017). GH_CPython: CPython plugin for grasshopper. https://doi.org/10.5281/zenodo.888148

API Reference—Pymunk 5.7.0 documentation. (n.d.). Retrieved September 28, 2020, from http://www.pymunk.org/en/latest/pymunk.html

Chaillou, S. (2019, July 9). AI & Architecture. Retrieved from https://towardsdatascience.com/ai-architecture-f9d78c6958e0.

China National Committee on Ageing (2007). Research Report of China's Aging Population Developing Trend Prediction. Chinese Women's Movement(02),15-18

Dormans, J., & Bakkes, S. (2011). Generating Missions and Spaces for Adaptable Play Experiences. IEEE Transactions on Computational Intelligence and AI in Games, 3(3), 216–228. doi: 10.1109/tciaig.2011.2149523

Egor, G., Sven, S., Martin, D., & Reinhard, K. (2020). Computer-aided approach to public buildings floor plan generation. Magnetizing Floor Plan Generator. Procedia Manufacturing, 44, 132–139. https://doi.org/10.1016/j.promfg.2020.02.214

Flores Villa, L. M., Unwin, J., & Raynham, P. (2017, September 17). The effect of daylight on the elderly population [Proceedings paper]. Lux Europa 2017: European Lighting Conference - Lighting for modern society. Proceedings of the Lux Europa 2017 European Lighting Conference - Lighting for Modern Society; Lighting engineering society of Slovenia. http://www.sdr.si/sl/index.html

Linden, R. V. D., Lopes, R., & Bidarra, R. (2014). Procedural Generation of Dungeons. IEEE Transactions on Computational Intelligence and AI in Games, 6(1), 78–89. doi: 10.1109/tciaig.2013.2290371

Niemann, M. (2015). Constructive Generation Methods for Dungeons (dissertation).

Nisztuk, M., & Myszkowski, P. B. (2019). Hybrid Evolutionary Algorithm applied to Automated Floor Plan Generation. International Journal of Architectural Computing, 17(3), 260–283. doi: 10.1177/1478077119832982

Nourian, P. (2016). Configraphics: Graph Theoretical Methods for Design and Analysis of Spatial Configurations [Delft University of Technology]. https://doi.org/10.7480/abe.2016.14

Nourian, P, (2019) How to write a thesis? A Generative Design Graduate Studio Guidebook. (2019). Delft University of Technology, Faculty of Architecture and Built Environment, Department of Architectural Engineering and Technology. DOI: 10.6084/m9.figshare.11987328

Roudsari, M. S., & Pak, M. (2013). Ladybug: A parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design. Proceedings of the 13th International IBPSA Conference Held in Lyon, France Aug 25–30th, 3128-3135.

Ruscica, T. (n.d.). A* Pathfinding Visualization Tutorial—Python A* Path Finding Tutorial. Retrieved September 28, 2020, from https://www.youtube.com/watch?v=JtiK0DOeI4A&t=3818s

Simon, J. (2019). Evolving Floorplans. Retrieved January 8, 2020, from https://www.joelsimon.net/evo_floorplans.html.

World Health Organization. (1995). WHOQOL-100. Geneva, Switzerland: World Health Organization.

Zhou, Y. (2018). Yang lao she shi jian zhu she ji xiang jie = Design and interpretation of elderly care facility. Beijing: China Architecture & Building Press.