Fig. 01 – Rhino to Nano Banana Interior View

Juliette Zamani Alavijeh
Caterina Lovo
Domina Lučin
Niklas Murmann
Cyrill Haas
Lisa Holbrook

CPLX

Cave Pulse Light Experiment

AR122

1:1 Interactive Architecture Prototypes

27/03/2026

1 LOCATION | **DESIGN CHALLENGE**

- Antarctica is the coldest, windiest, and driest continent on Earth

- −10°C (Coast) to −60°C (Interior)
- Katabatic winds (Wind speeds can exceed 100 km/h for days at a time)
- Blizzards (Temperature is less than 0°C and visibility is reduced to 100 m or less)

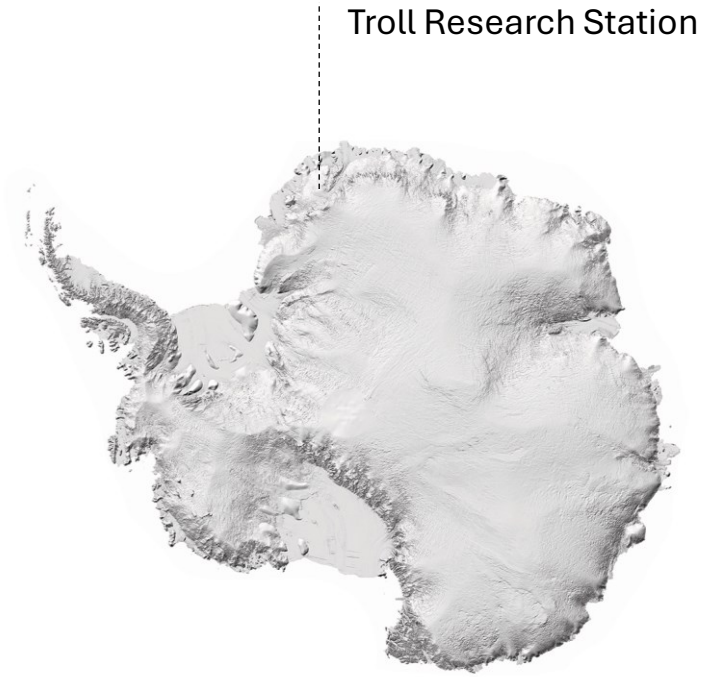


Fig. 03 – Map Terrain Antarctica

Fig. 02 – Blizzard at a station in Antartica

1 LOCATION | **DESIGN CHALLENGE**

- Approx. 50 days of polar daylight in summer
  (November to January)
- Approx. 60 days of polar night
  (May to July)

- 63% of expeditioners reported:
  - Difficulty falling asleep
  - Feeling sleepy during the daytime
  - Delayed sleep onset and offset time
  - Reduced sleep duration
  - Decreased sleep efficiency
  - Aggravated sleep fragmentation
  - And altered sleep architecture

Source: Liu et al., 2024, Nature magazine

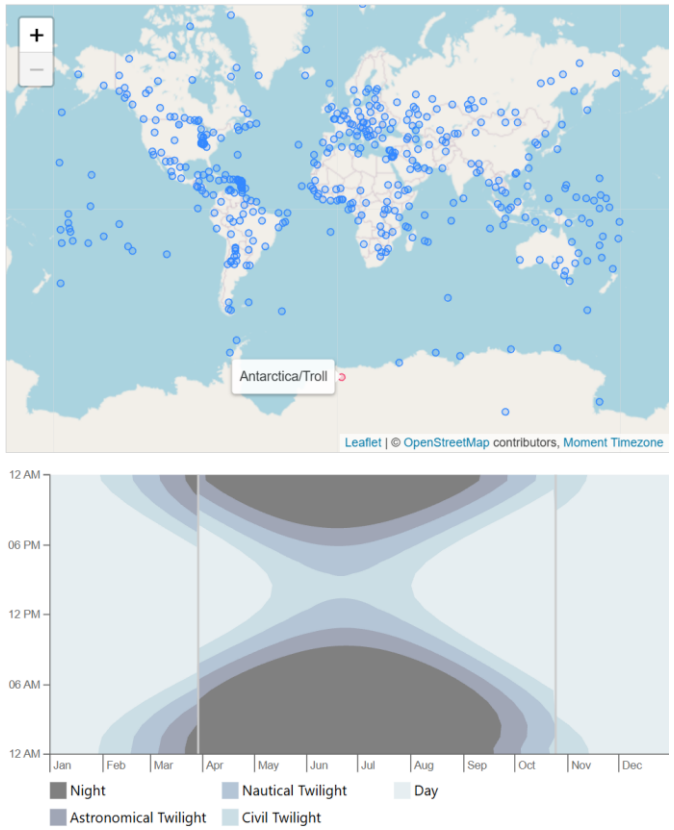


Fig. 05 – Daylight conditions – Troll station

Fig. 04 – Polar night / Aurora australis on Antarctica

INTRODUCTION

1 INTRODUCTION | **PROJECT GOAL**

CHALLENGES

Weather conditions

- Severe Temperatures
- Strong Catabatic Winds
- Minimal Precipitation
- Low Humidity

Arctic seasons

- Extreme Light Cycles

Lighting conditions

- Circadian Rhythm Disruption
- Reduced Work Efficiency
- Sleep & Mood issues

Spatial limitation

- Lack of Privacy
- Lower Productivity
- Stress and Discomfort
- Design compromises

IMPACT ON
PHYSICAL AND MENTAL HEALTH

Fig. 07 – Functional division

Fig. 06 – Troll Research Station, Norsk Polarinstitutt

Autonomy	Impact	Stimulation
Beauty	Morality	Security
Comfort	Purpose	Fitness
Community	Recognition	
Competence	Relatedness	

Fig. 08 – Fundamental needs

Fig. 06 – Troll Research Station, Norsk Polarinstitutt

Autonomy	Impact	**Stimulation**
Beauty	Morality	Security
Comfort	Purpose	**Fitness**
Community	Recognition	
Competence	Relatedness	

Fig. 08 – Fundamental needs

Fig. 06 – Troll Research Station, Norsk Polarinstitutt

1 INTRODUCTION | **PROJECT GOAL**

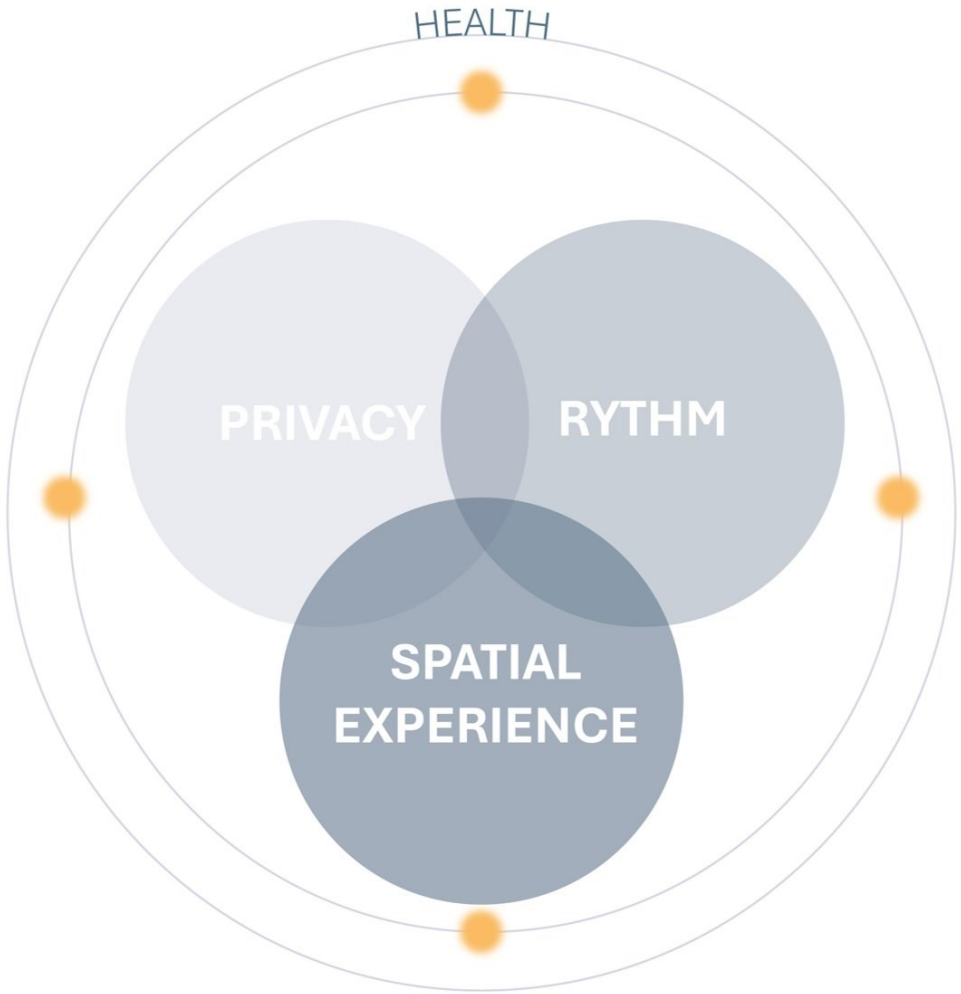


Fig. 09 – Project goals

Fig. 06 – Troll Research Station, Norsk Polarinstitutt

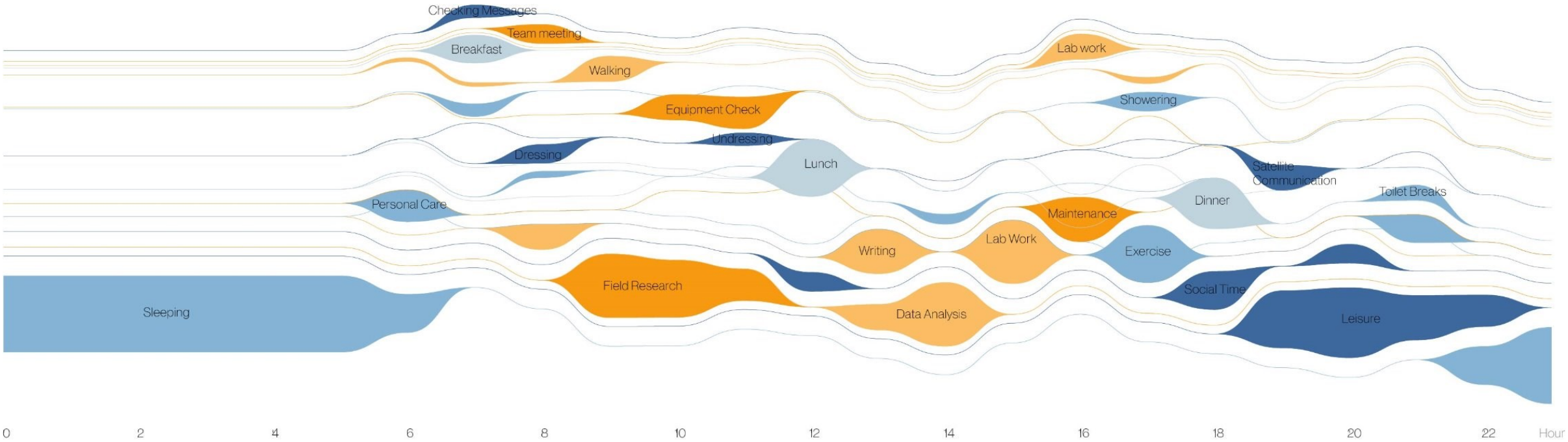
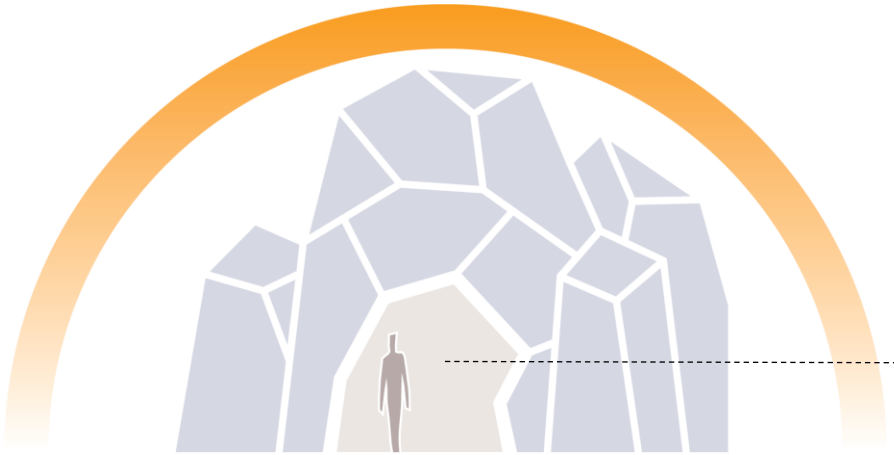


Fig. 10 – 24h activity mapping

DESIGN CONCEPT

2 DESIGN CONCEPT | **THE CAVE EXPLORATION**



CAVE | Protective space

Fig. 11 – Concept: Cave

Fig. 06 – Troll Research Station, Norsk Polarinstitutt

2 DESIGN CONCEPT | **STACKING**

- Spatial Varity
- Amplify the quality of each function
- Vertical corridor
- Reduced footprint
- Less foundation

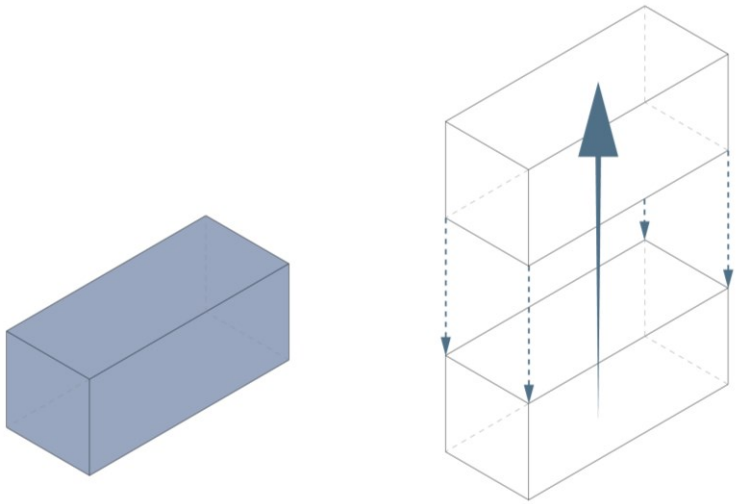


Fig. 14 – Diagram stacking

Fig. 12 – Exterior Vision, AI generated (Nano Banana)

2 DESIGN CONCEPT | **ARRAYING**

- Air Lock as a horizontal connector

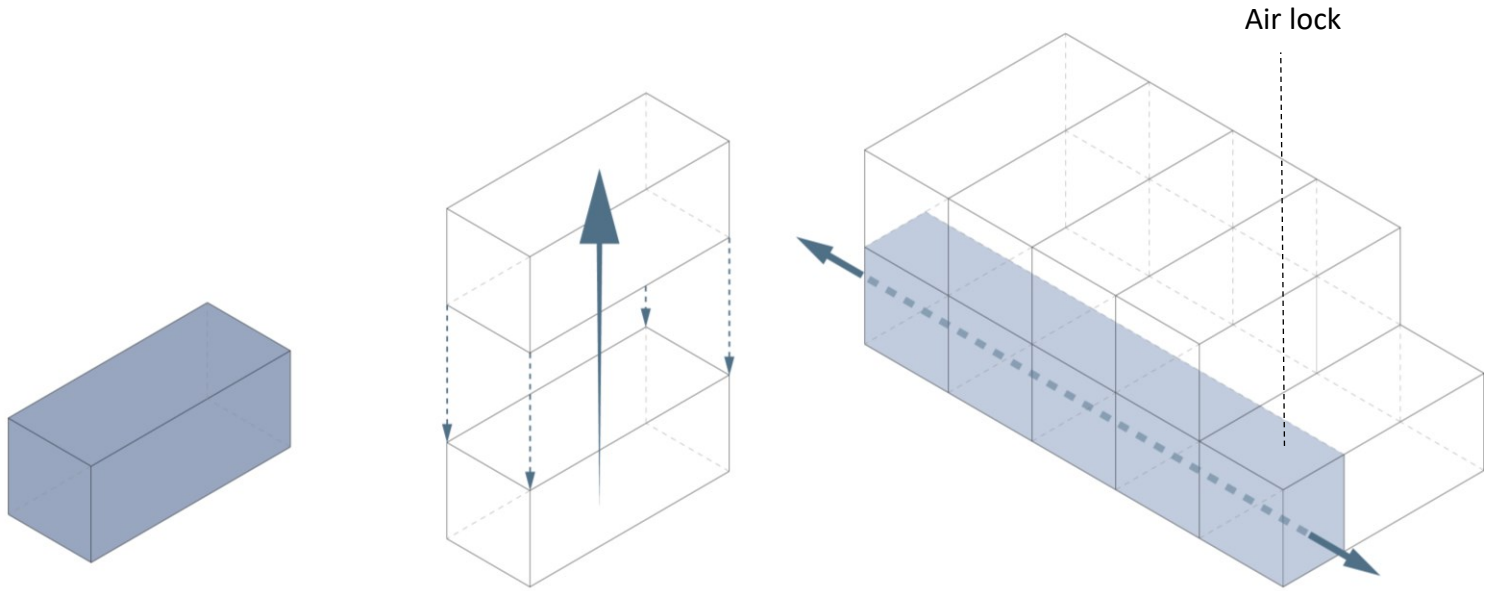
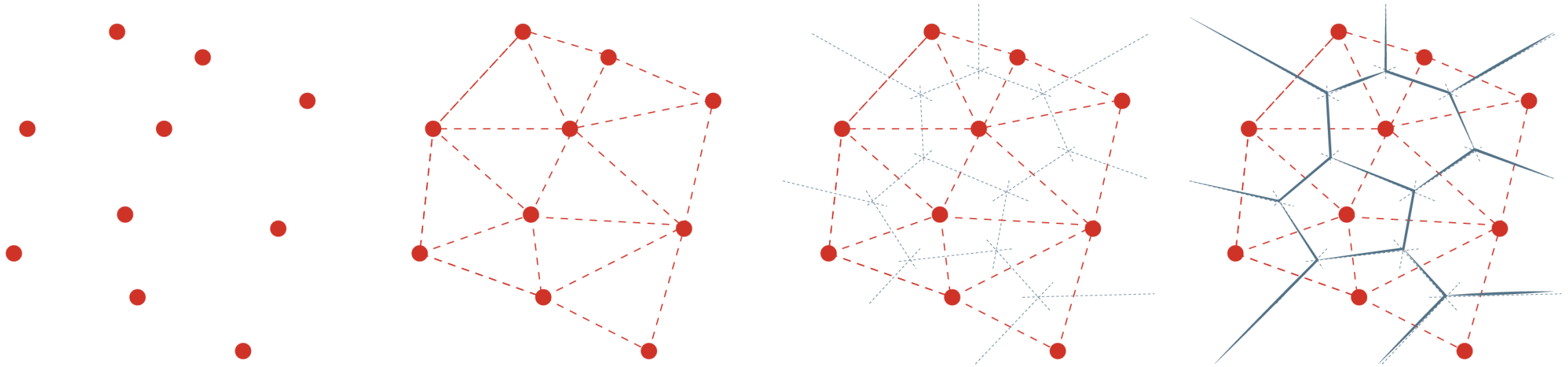


Fig. 14 – Diagram stacking & arraying

Fig. 12 – Exterior Vision, AI generated (Nano Banana)

METHODS



1) Seed distribution within the design domain (adaptive node definition)

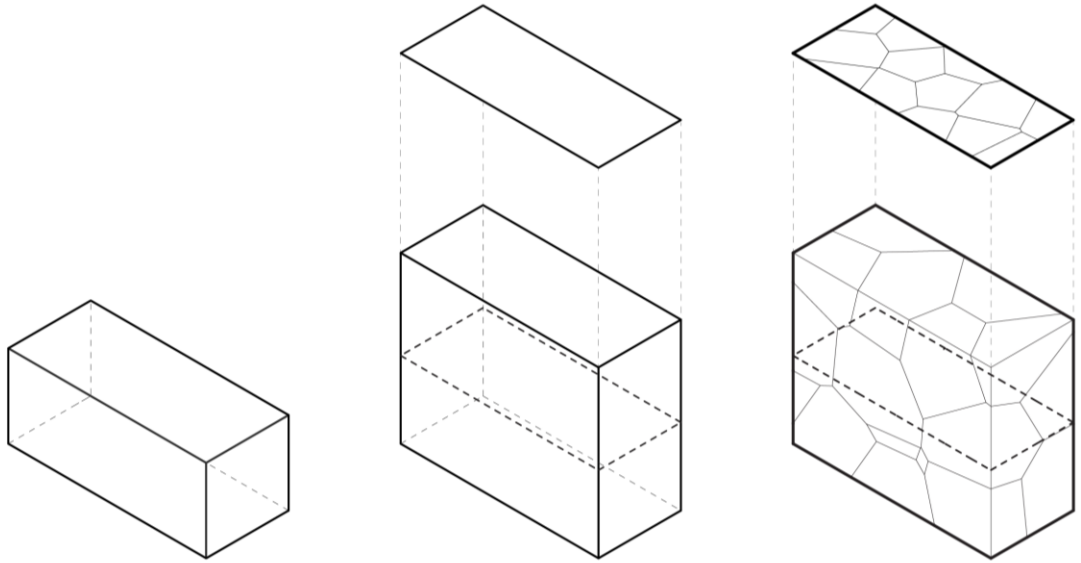
2) Delaunay triangulation

3) Bisector line identification

4) Voronoi tessellation - (2d polygons)

Fig. 13 – Voronoi principle

3 METHODS | **TOOLBOX**



Create container

Stack and add floor

Apply voronoi for zoning

Fig. 14 – Diagram form finding script

Fig. 12 – Exterior Vision, AI generated (Nano Banana)

AR122 1:1 Interactive Architecture Prototypes

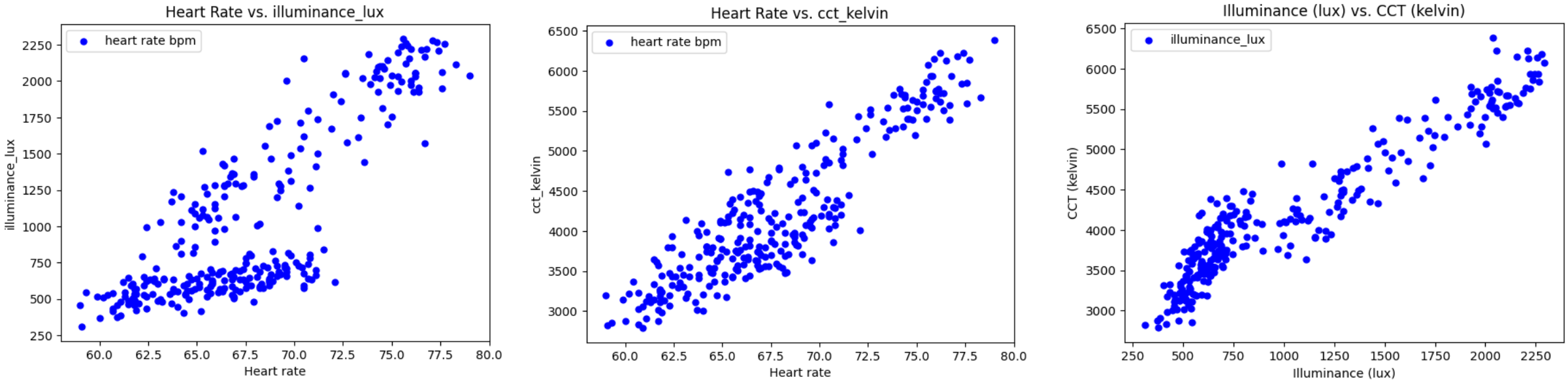


Fig. 15 – DATA output

3 METHODS | **AI**

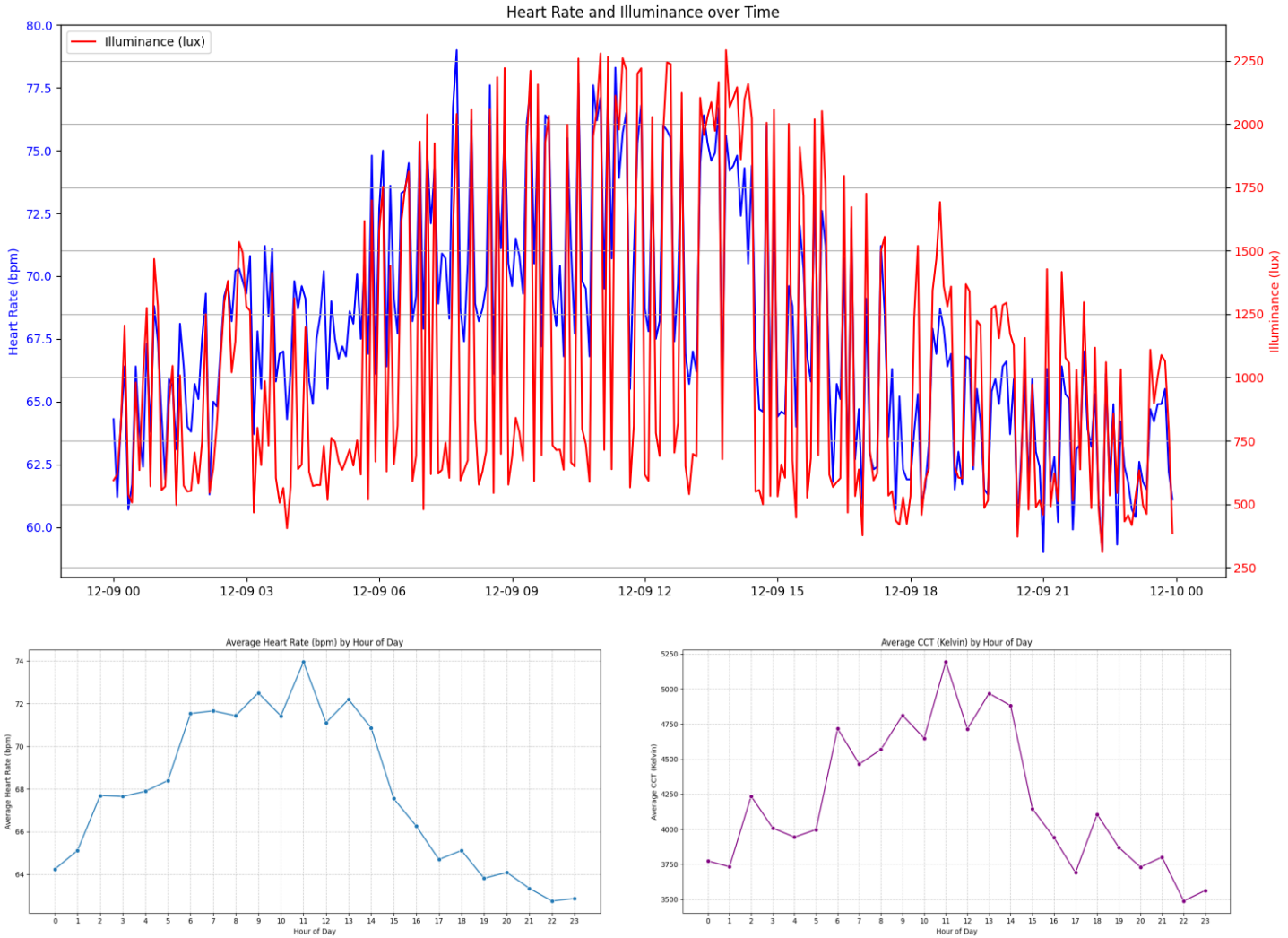


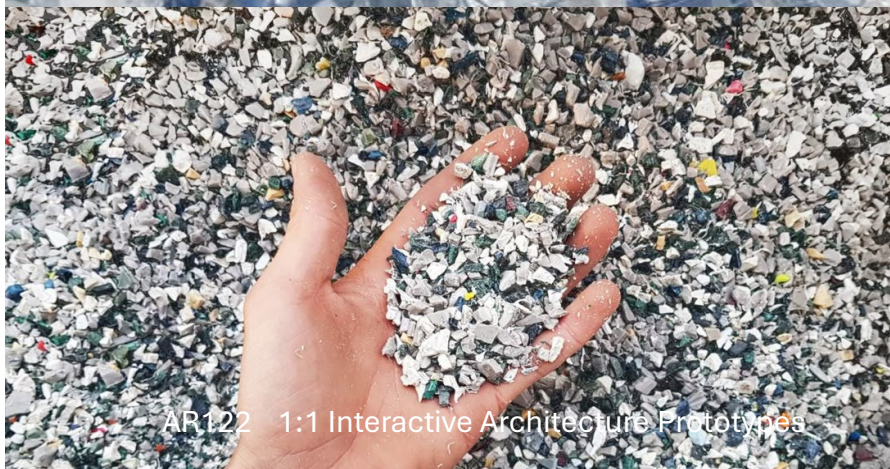
Fig. 16 – DATA output

Fig. 12 – Exterior Vision, AI generated (Nano Banana)

3 METHODS | **MATERIAL**

Raw materials
- **Bottles** (PET)
- **Food packaging** (ABS, PET, PLA, PP)
- Food waste (more useful for growing plants)
- Kerosine containers (contamination issue)

Fig. 13 – Waste Products



3 METHODS | **D2RP**

- The project is focused on creating and building with artificial intelligence (AI)-supported
- Design-to-Robotic-Production and Operation (D2RP&O)

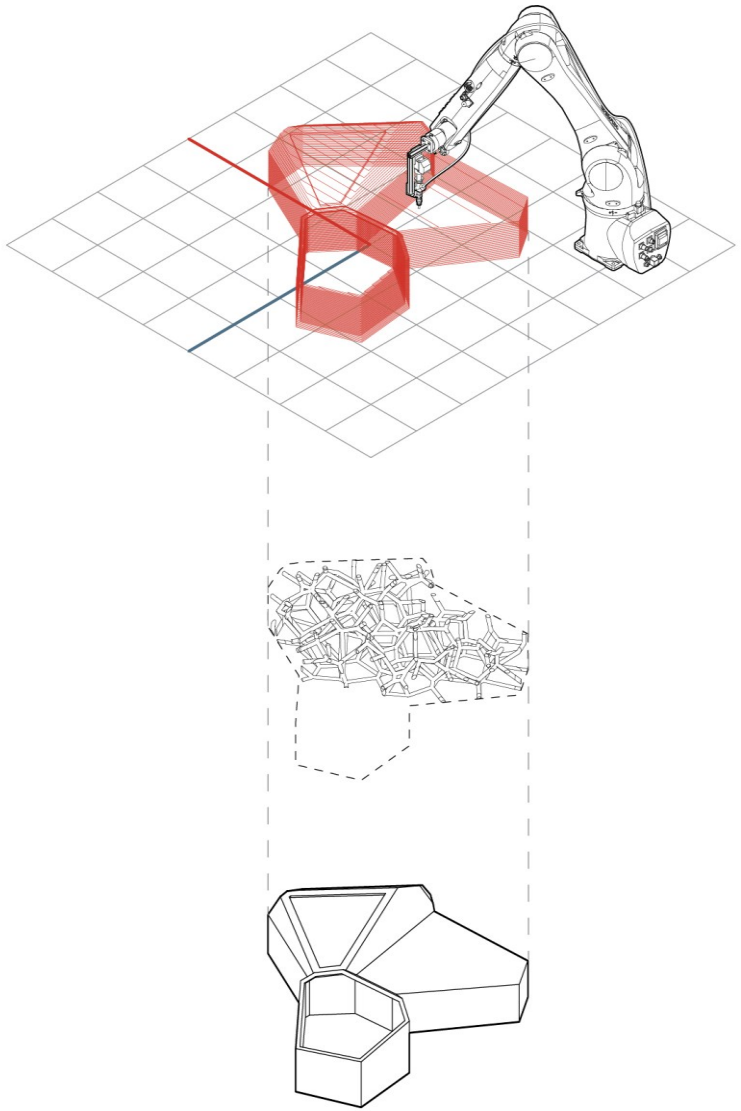


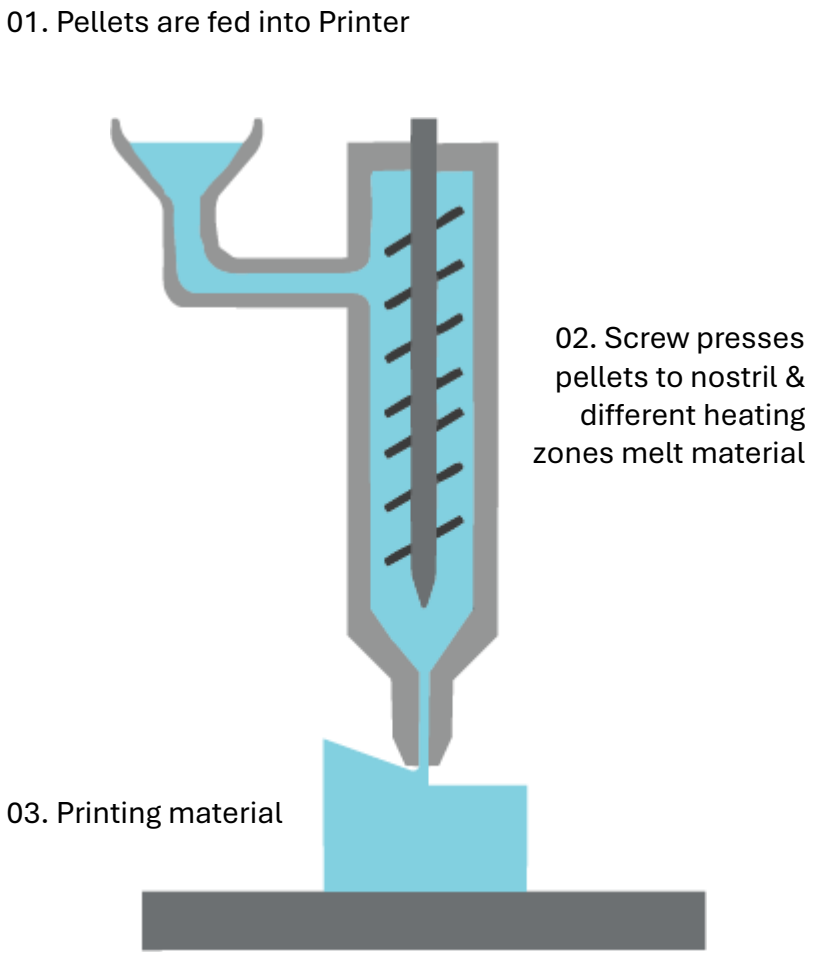
Fig. 15 – Schematic Print Process Fragment

Fig. 14 – KUKA Robot Additive Manufacture

3 METHODS | **EXTRUTION TECHNIQUES**

Extrusion techniques
- Fused Filament Fabrication (FFF/FDM) -> Filament or pellets
- **Fused particle or pellet fabrication** -> shredded or pelletized waste

Fig. 16 – Fused particle fabrication scheme



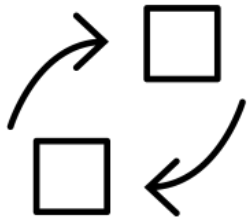
Step 1

Bluetooth
The Polar H10 sensor transmits live physiological data using Bluetooth Low Energy (BLE)

Step 2

Heart rate reception
Python script receives the data, reads the heart rate, maps physiological changes to variations in color temperature (CCT)

Step 3



CCT to RGB
A data bridge streams the processed RGB data continuously

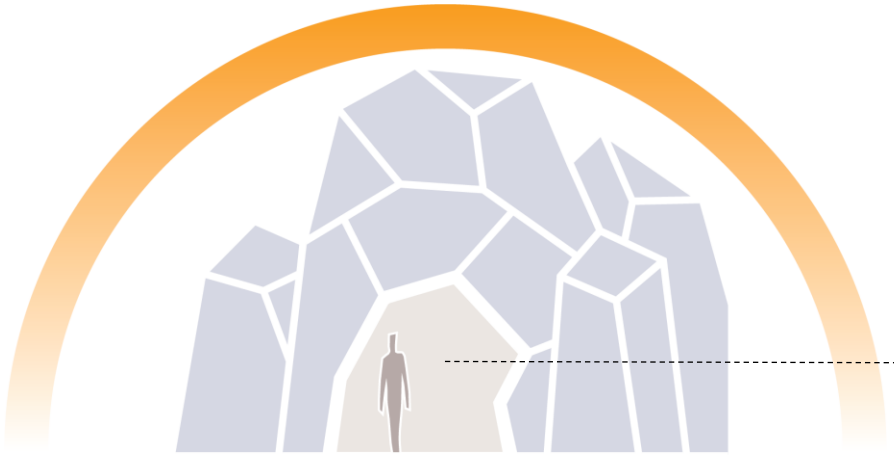
Step 4

Grasshopper
Grasshopper connects to the RGB output using a client component: GhPython

Fig. 17 – Data implementation steps, source: Python script PolarH10

DESIGN DEVELOPMENT

4 DESIGN DEVELOPMENT | **RECAP**



CAVE | Protective space

Fig. 19 – Spatial configuration

Fig. 18 – Interior perspective

4 DESIGN DEVELOPMENT | **SPATIAL CONFIGURATION**

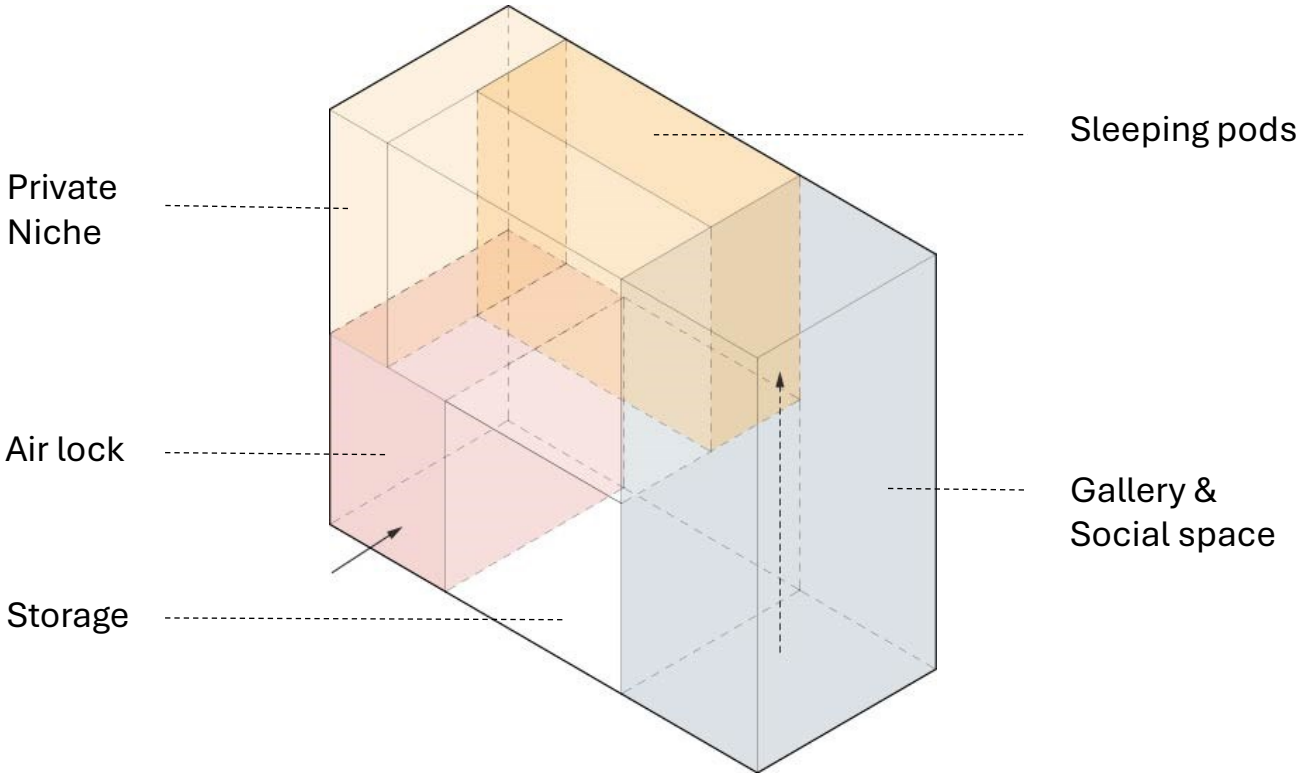
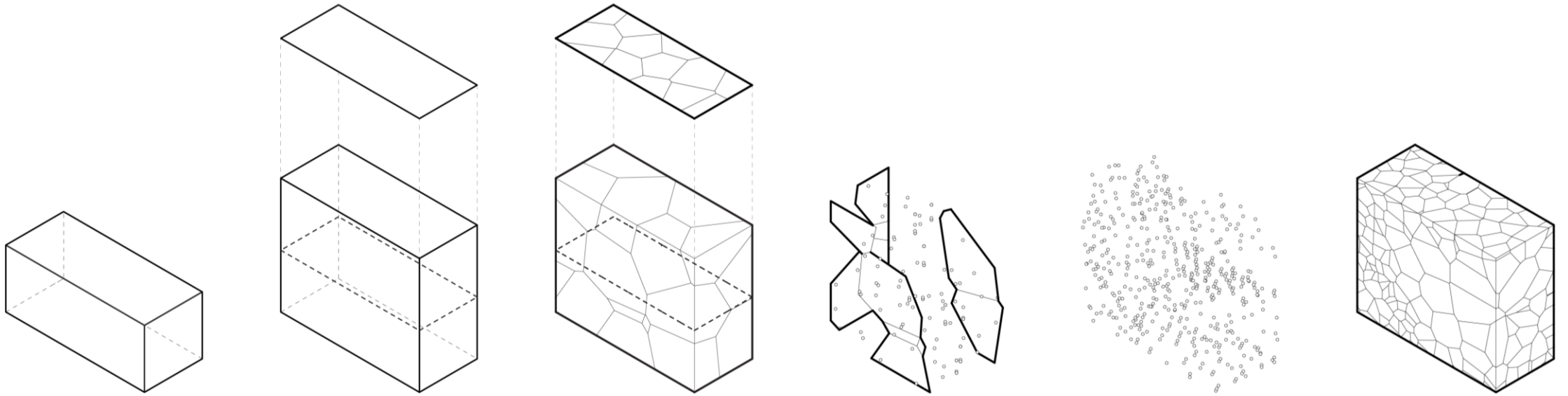


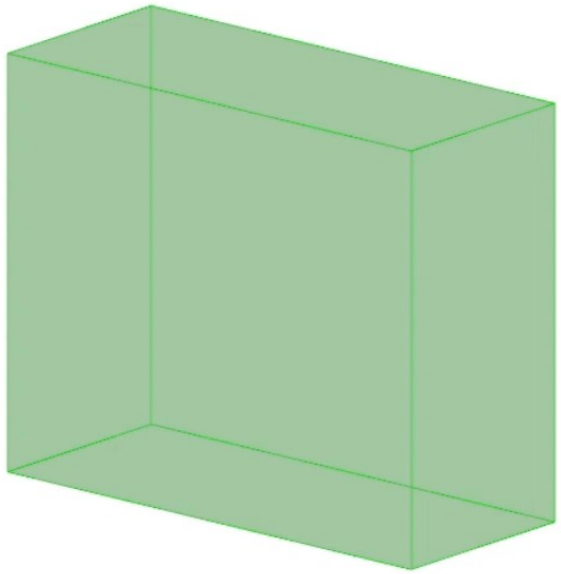
Fig. 20 – Diagram form finding script

Fig. 18 – Interior perspective



Create container | **Stack and add floor** | **Apply vonoroi for zoning** | **Pick cells for each density (S, M, L, XL) and populate** | **Adjust points manually** | **Create spacial vonoroi**

Fig. 21 – Diagram form finding script



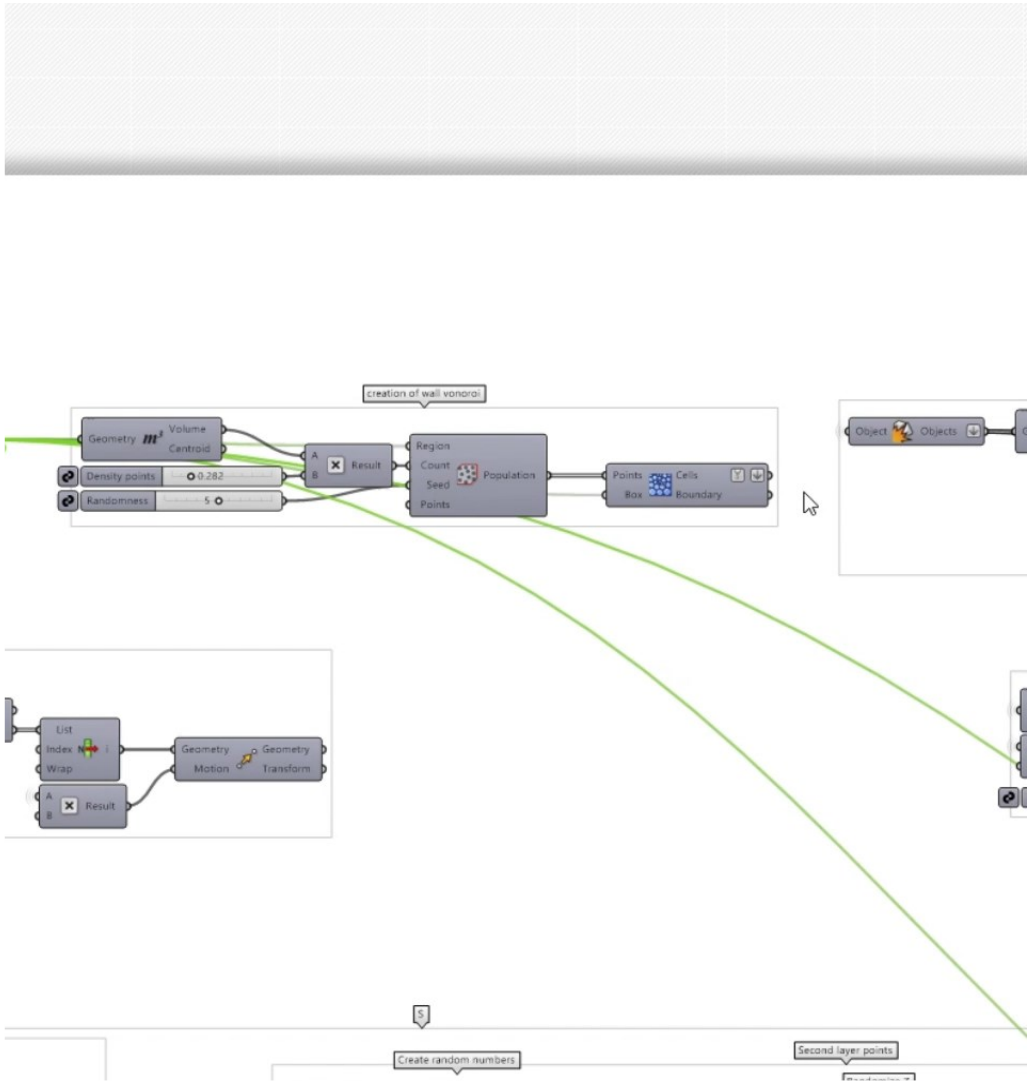


Fig. 22 – Video form finding script

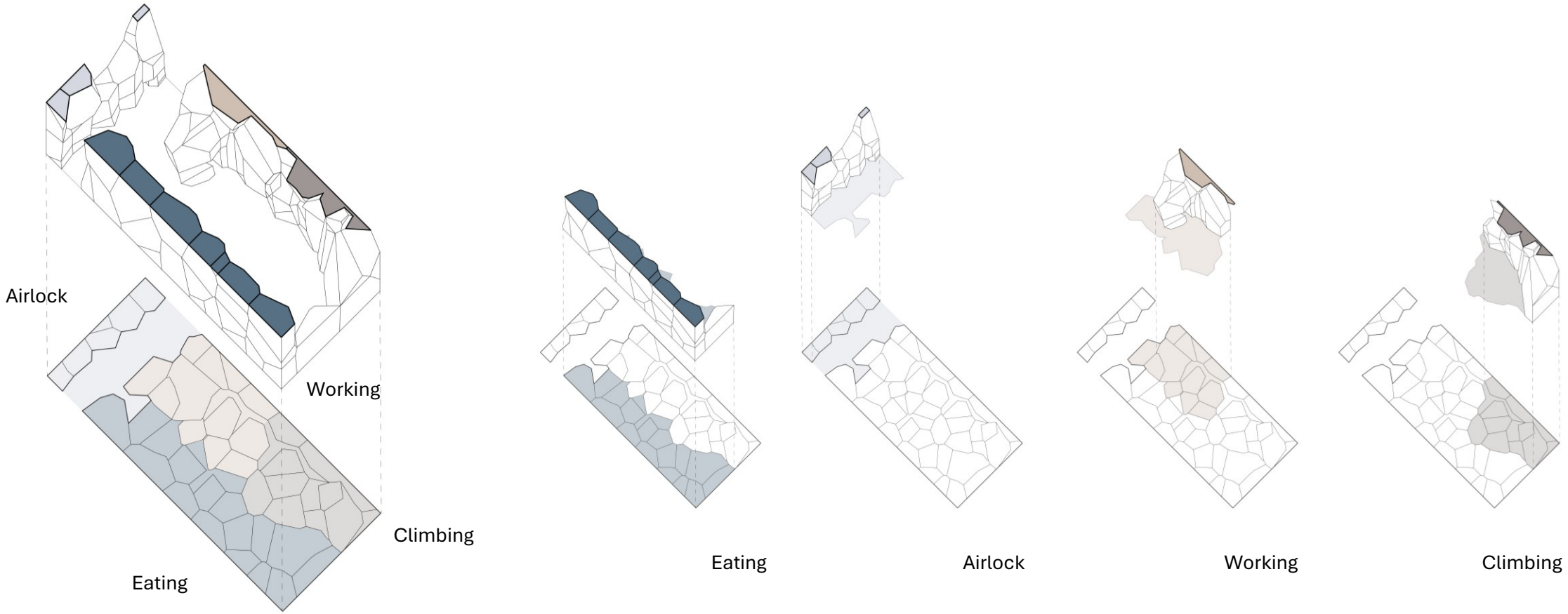


Fig. 23 – Lower Floor Spatial Configuration

Fig. 24 – Upper Floor Spatial Configuration

4 DESIGN DEVELOPMENT | **FLOOR PLANS**

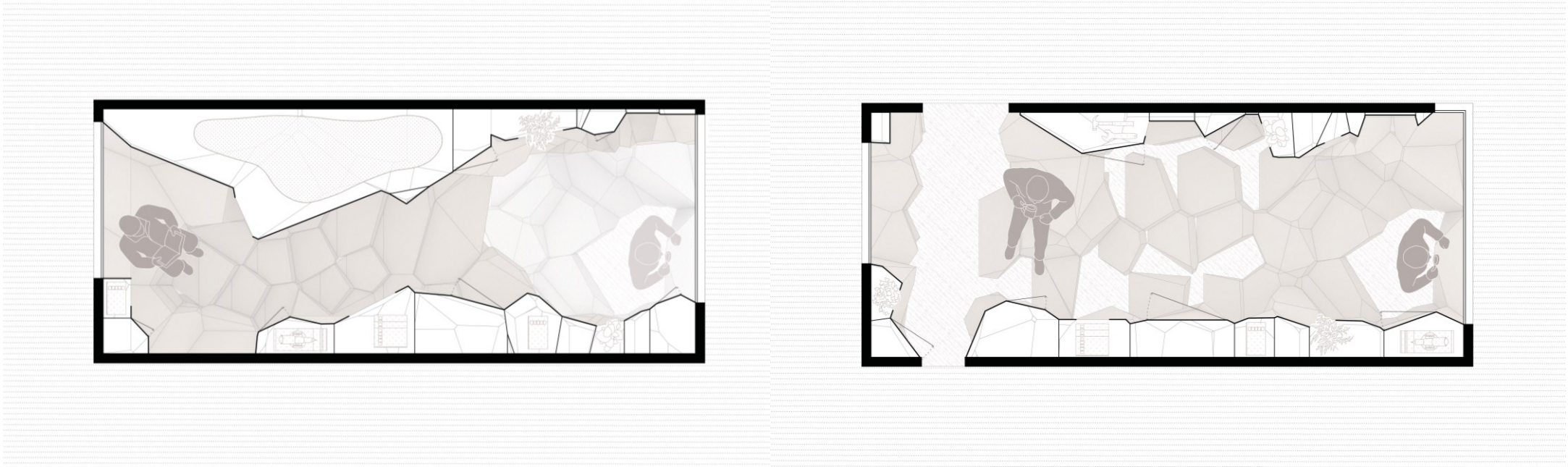


Fig. 25 – Upper and lower floor plans

4 DESIGN DEVELOPMENT | **SECTIONS**

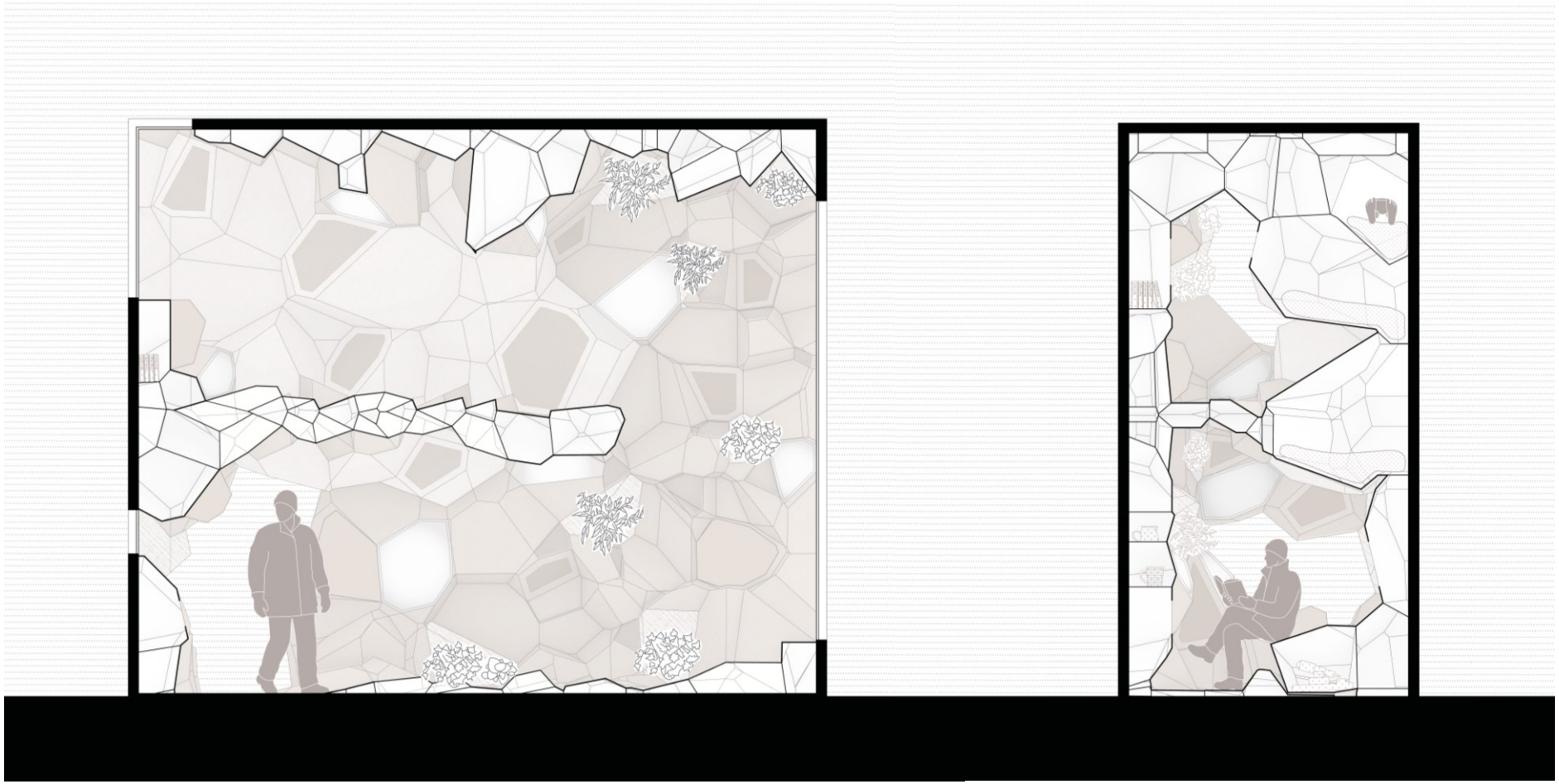


Fig. 26 – Longitudinal and cross sections

4 DESIGN DEVELOPMENT | **SPATIAL CONFIGURATION**

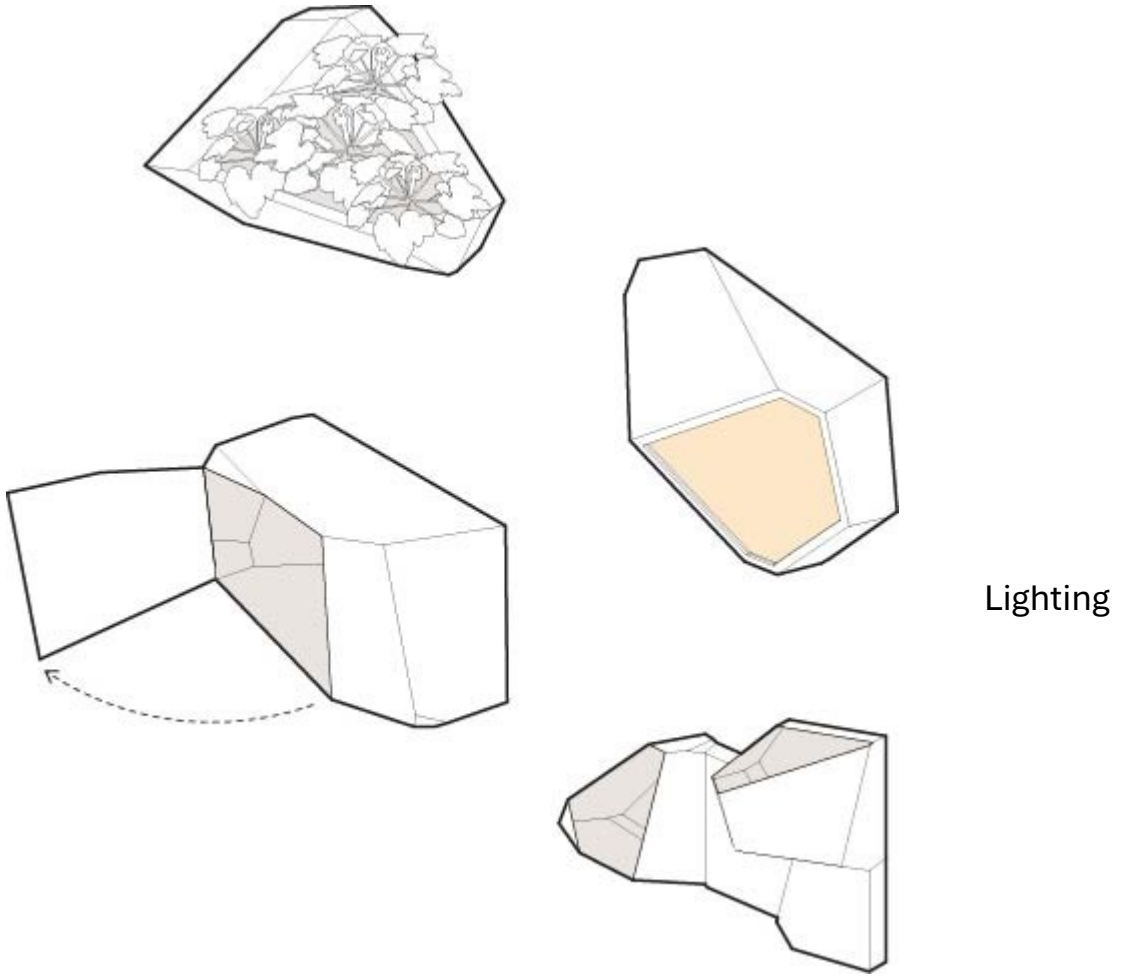


Fig. 27 – Voronoi typologies

Fig. 18 – Interior perspective

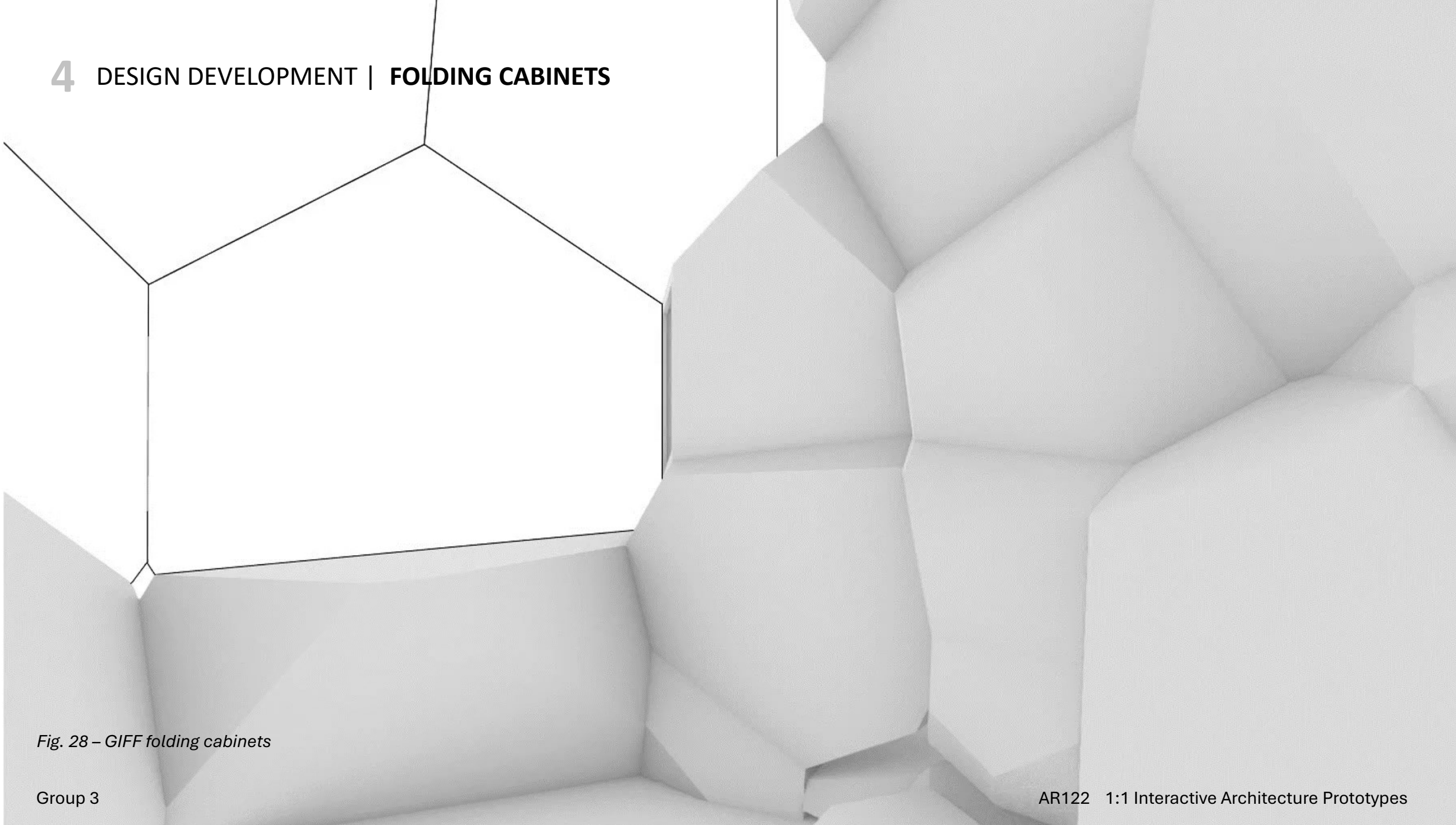


Fig. 28 – GIFF folding cabinets

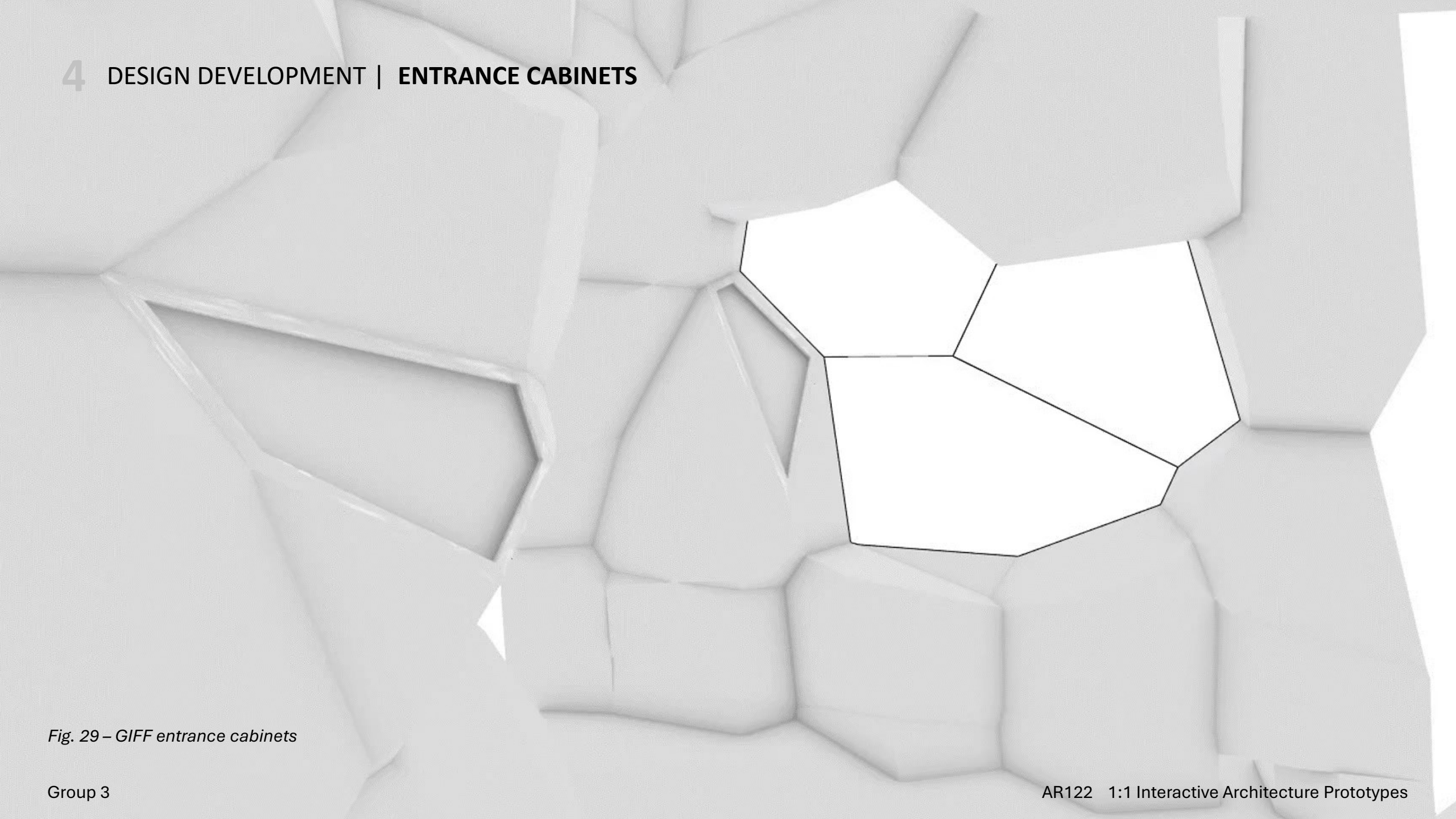


Fig. 29 – GIFF entrance cabinets

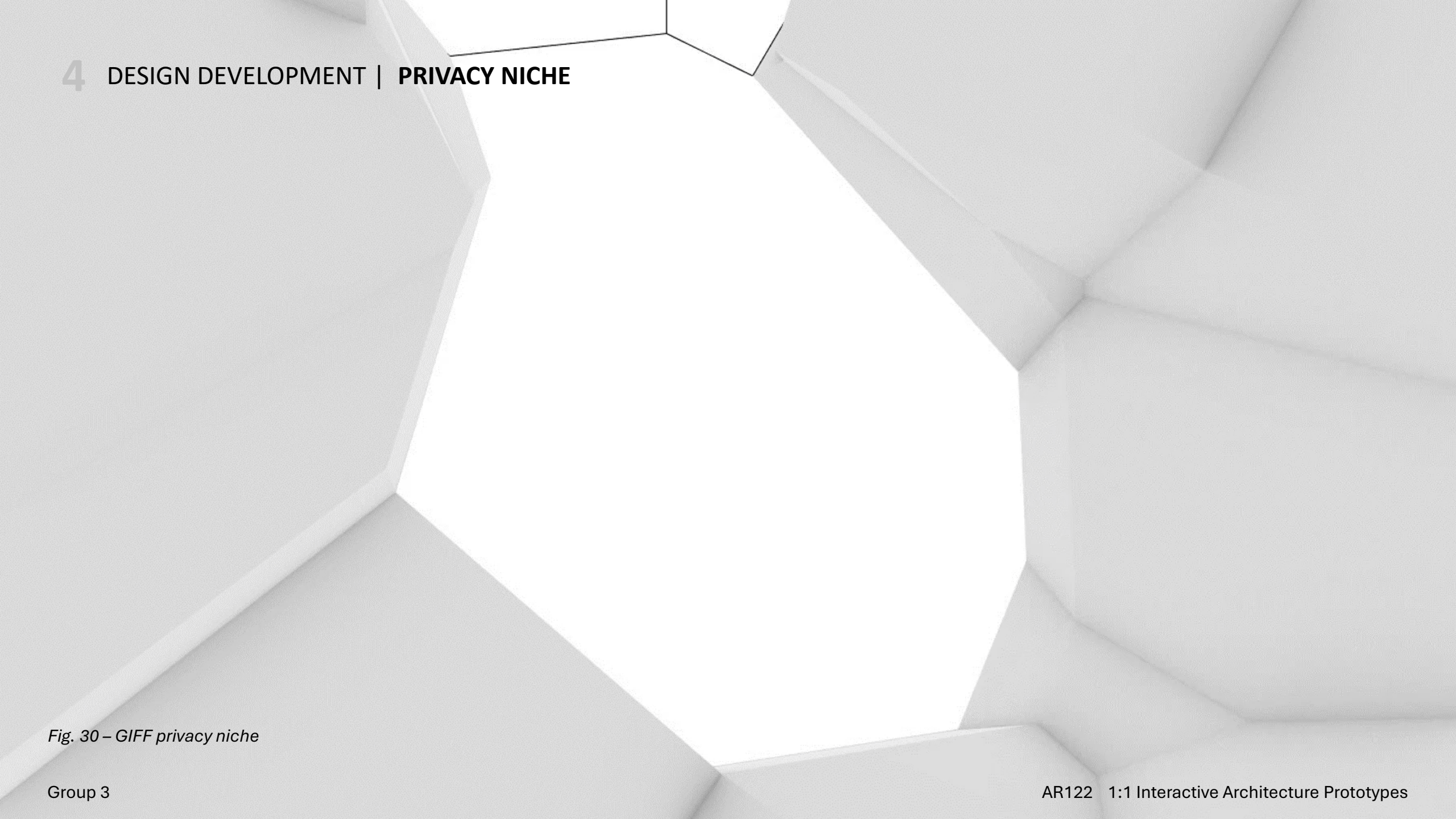


Fig. 30 – GIFF privacy niche

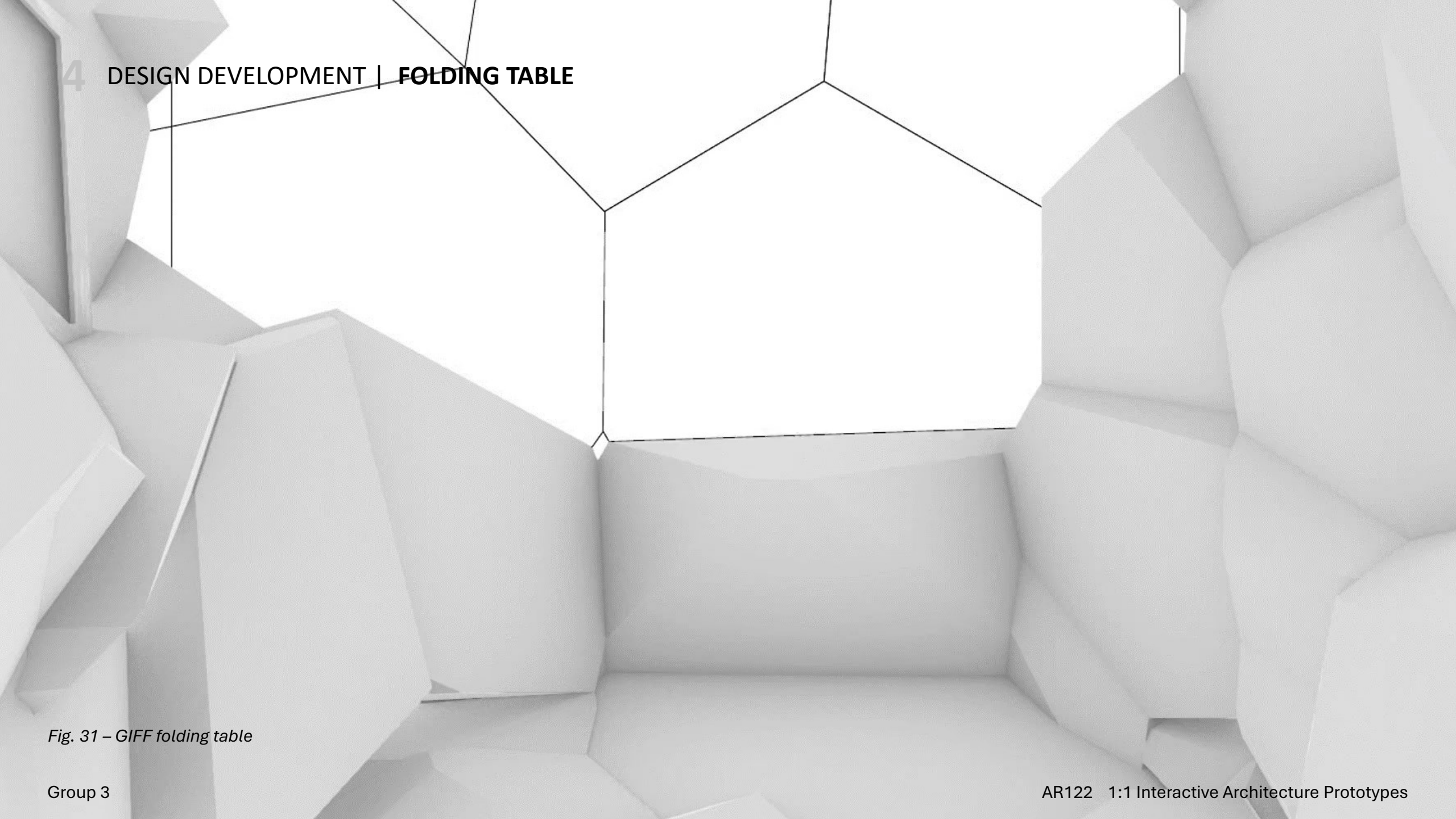


Fig. 31 – GIFF folding table

LIGHTING

5 LIGHTING | **CONCEPT**

•Simulate circadian rhythm
•Use a high CCT during daytime and working hours
•Gradually shift to low CCT in the morning and evening

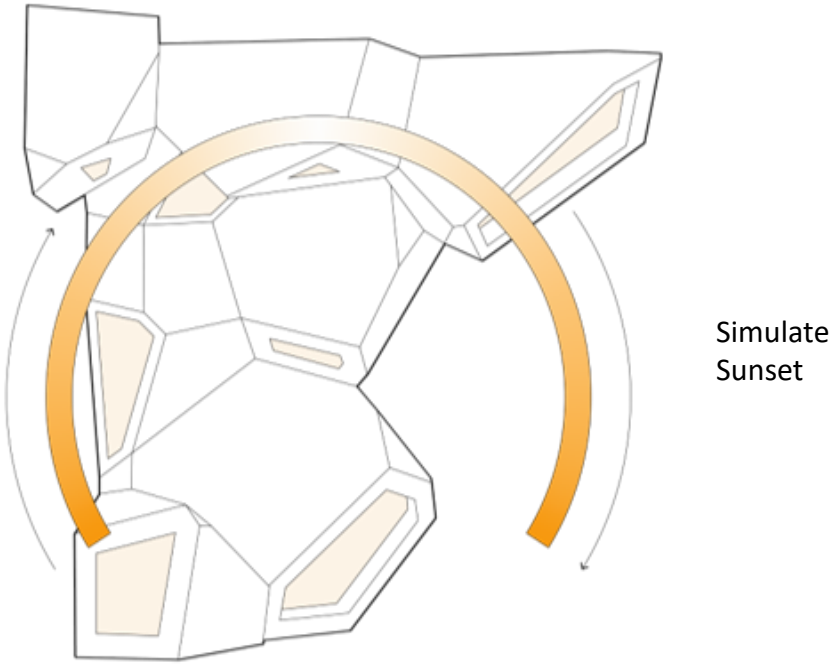


Fig. 32 – Lighting principle

Fig. 33 – Lighting principle, V-Ray Ren

5 LIGHTING | **CONCEPT**

- Implement heart rate responsive lighting
- When heart rate is lower, the system shifts to a higher CCT
  Increased Alertness
- When heart rate is elevated, the system shifts to a lower CCT
  Calm Atmosphere

Fig. 34 – Equalizing heartrates

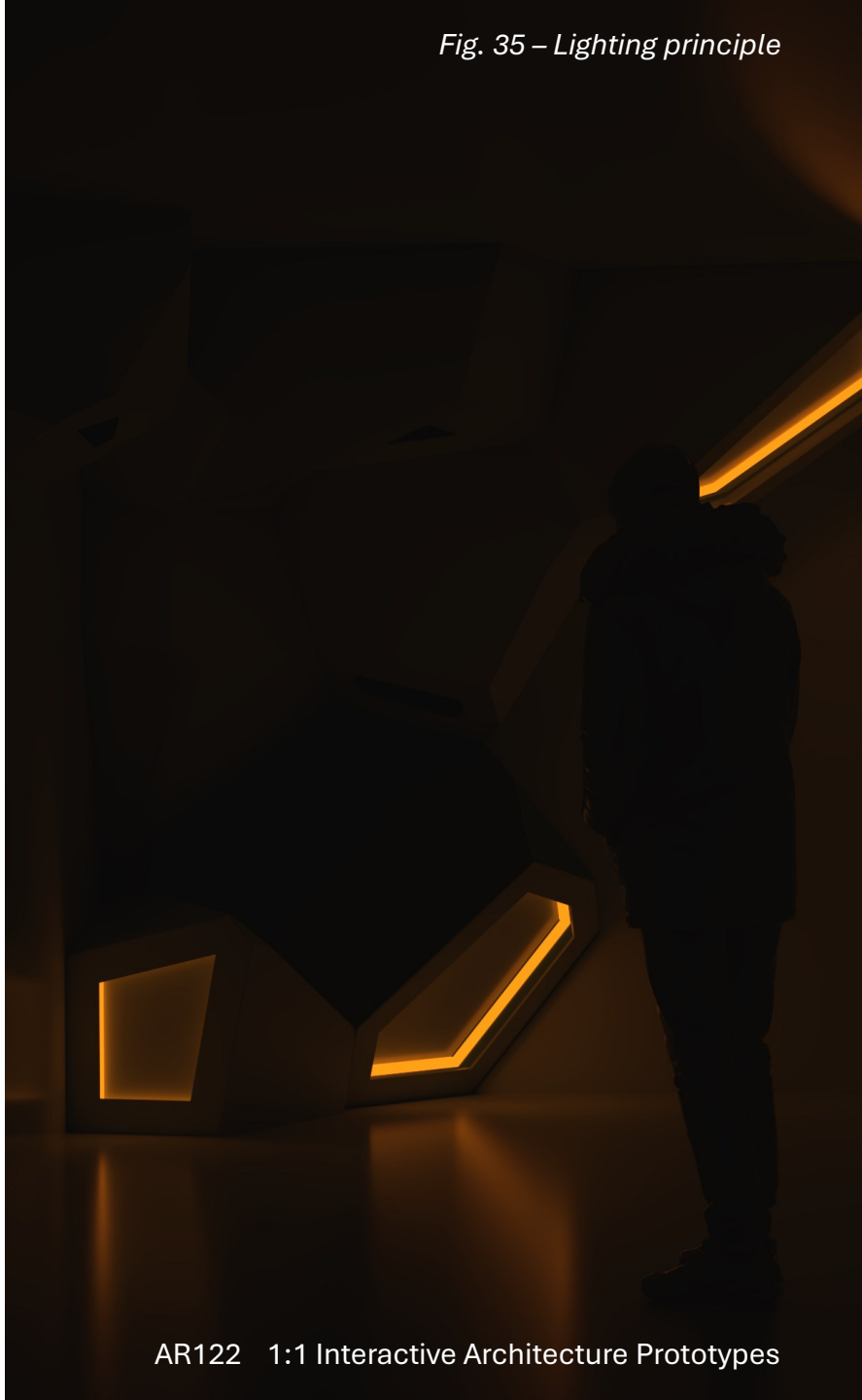
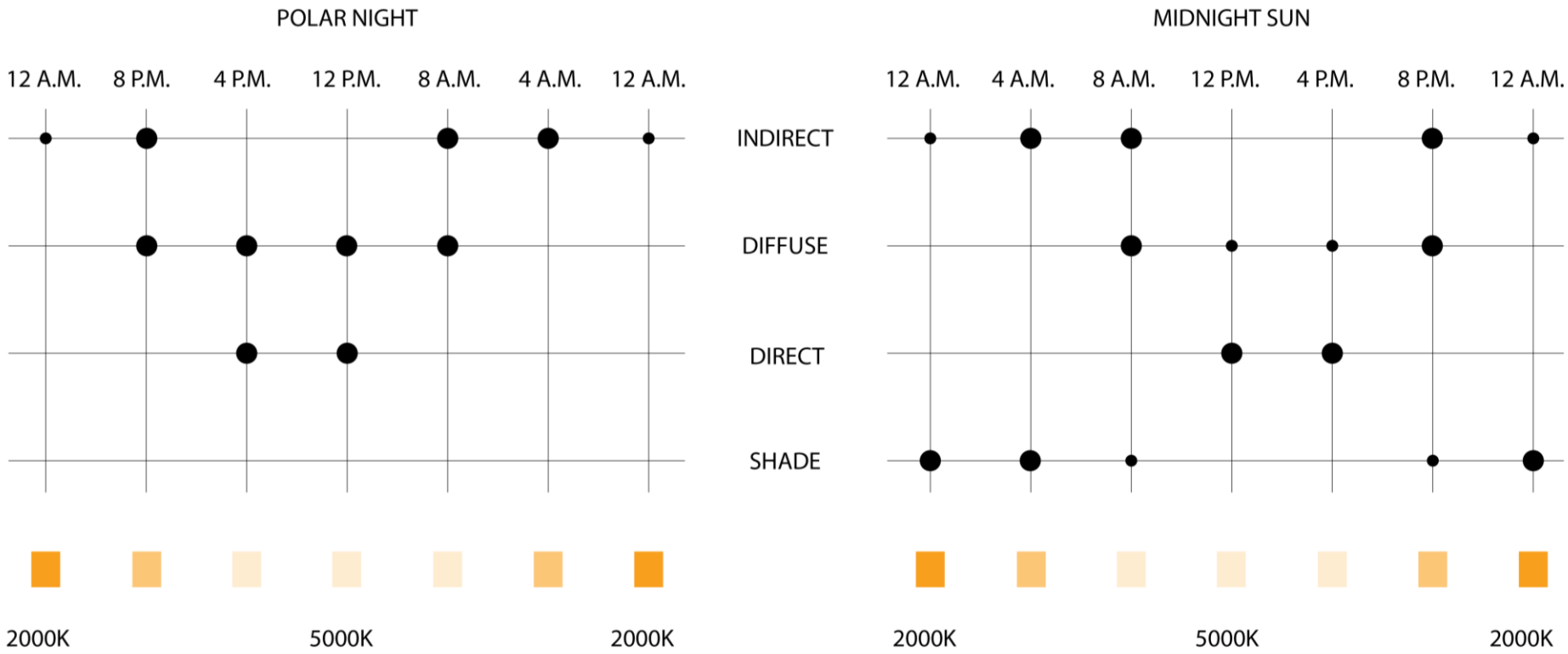


Fig. 35 – Lighting principle

5 LIGHTING | CONCEPT



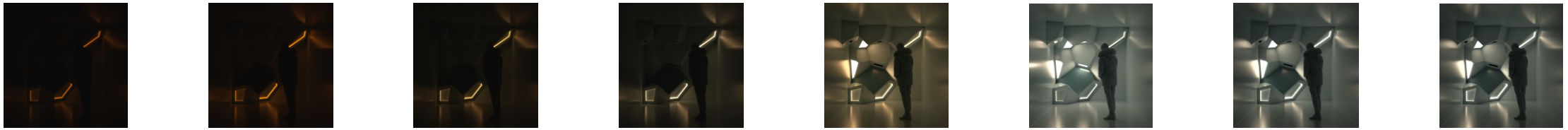


Fig. 36 – CCT adjustment based on activity mapping

5 LIGHTING | **GH SCRIPT**

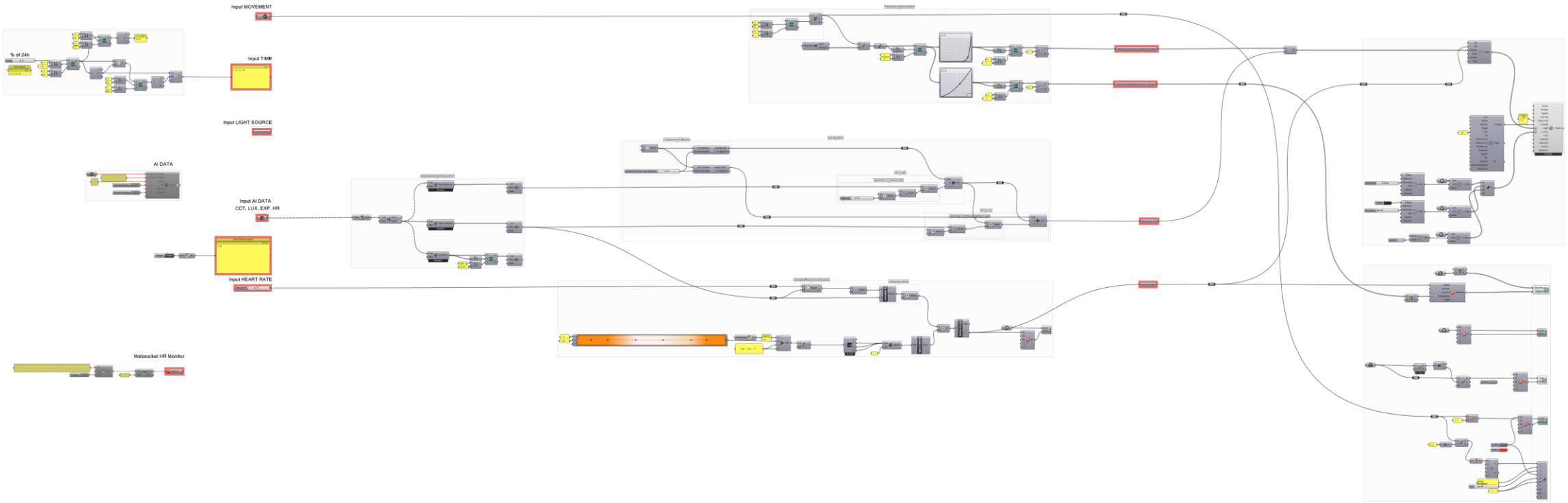


Fig. 37 – Lighting Script Grasshopper - Overview

5 LIGHTING | **GH SCRIPT**

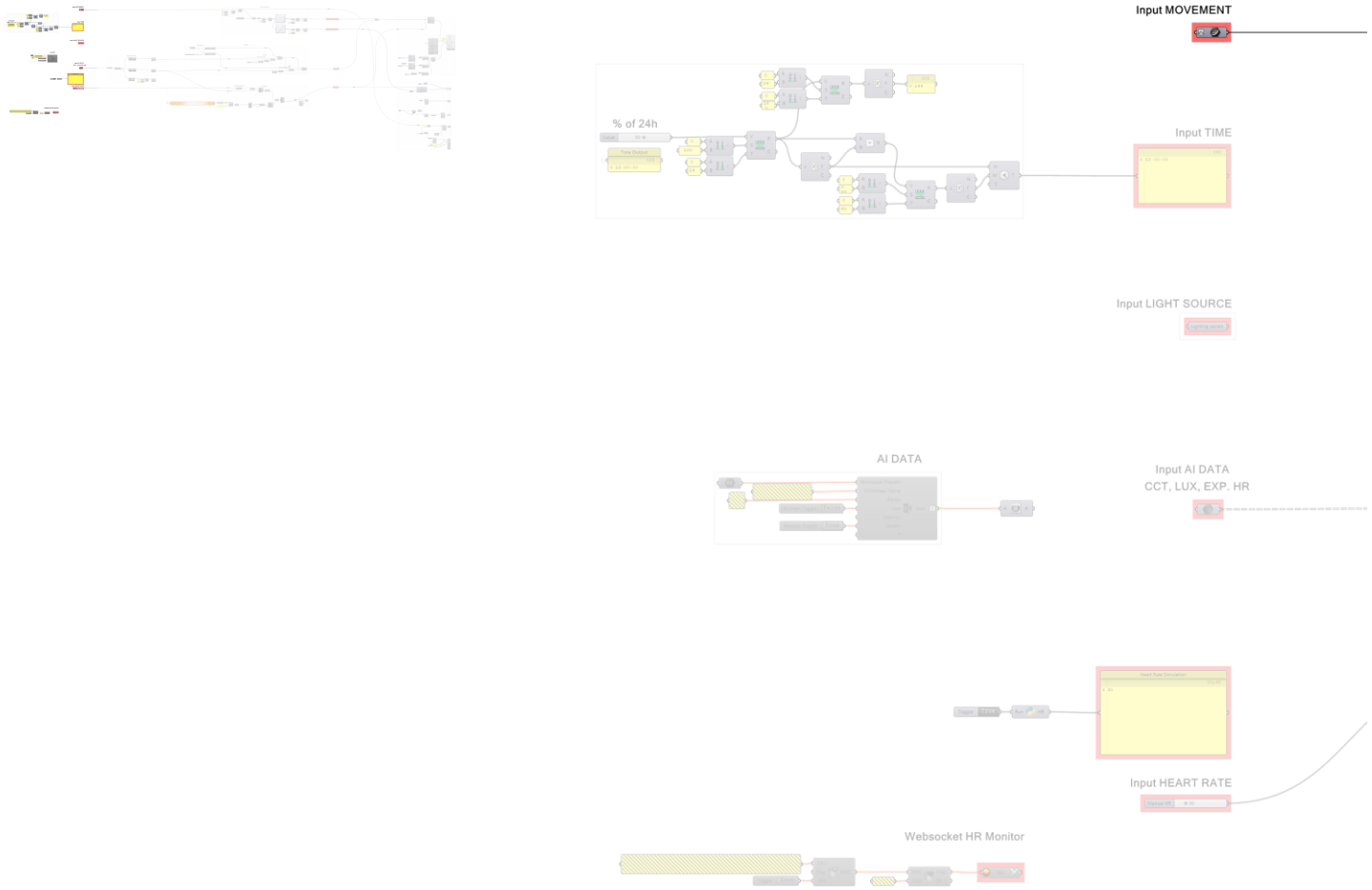


Fig. 38 – Lighting Script Grasshopper – Input Parameters

5 LIGHTING | **GH SCRIPT**

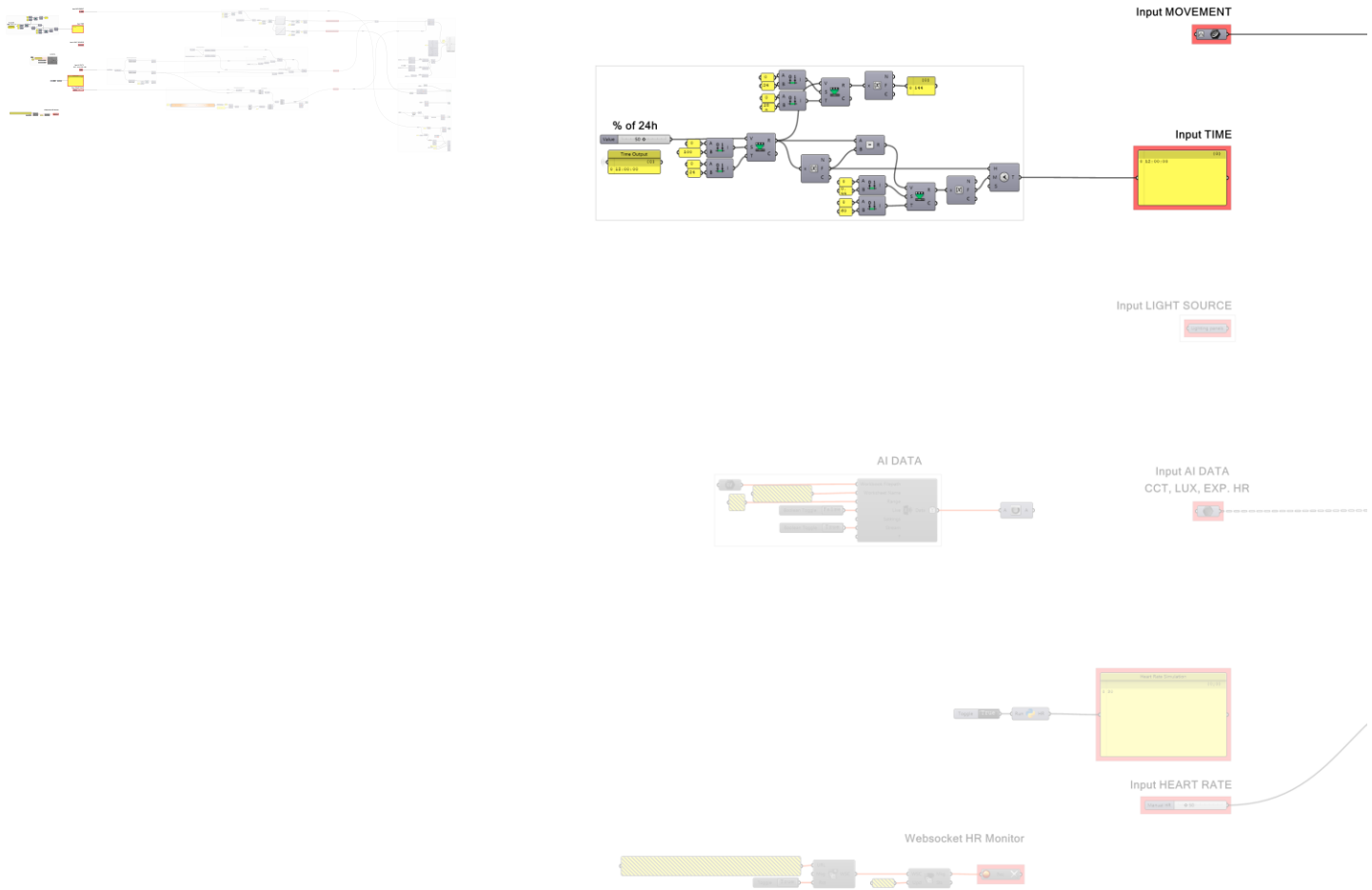


Fig. 38 – Lighting Script Grasshopper – Input Parameters

5 LIGHTING | **GH SCRIPT**

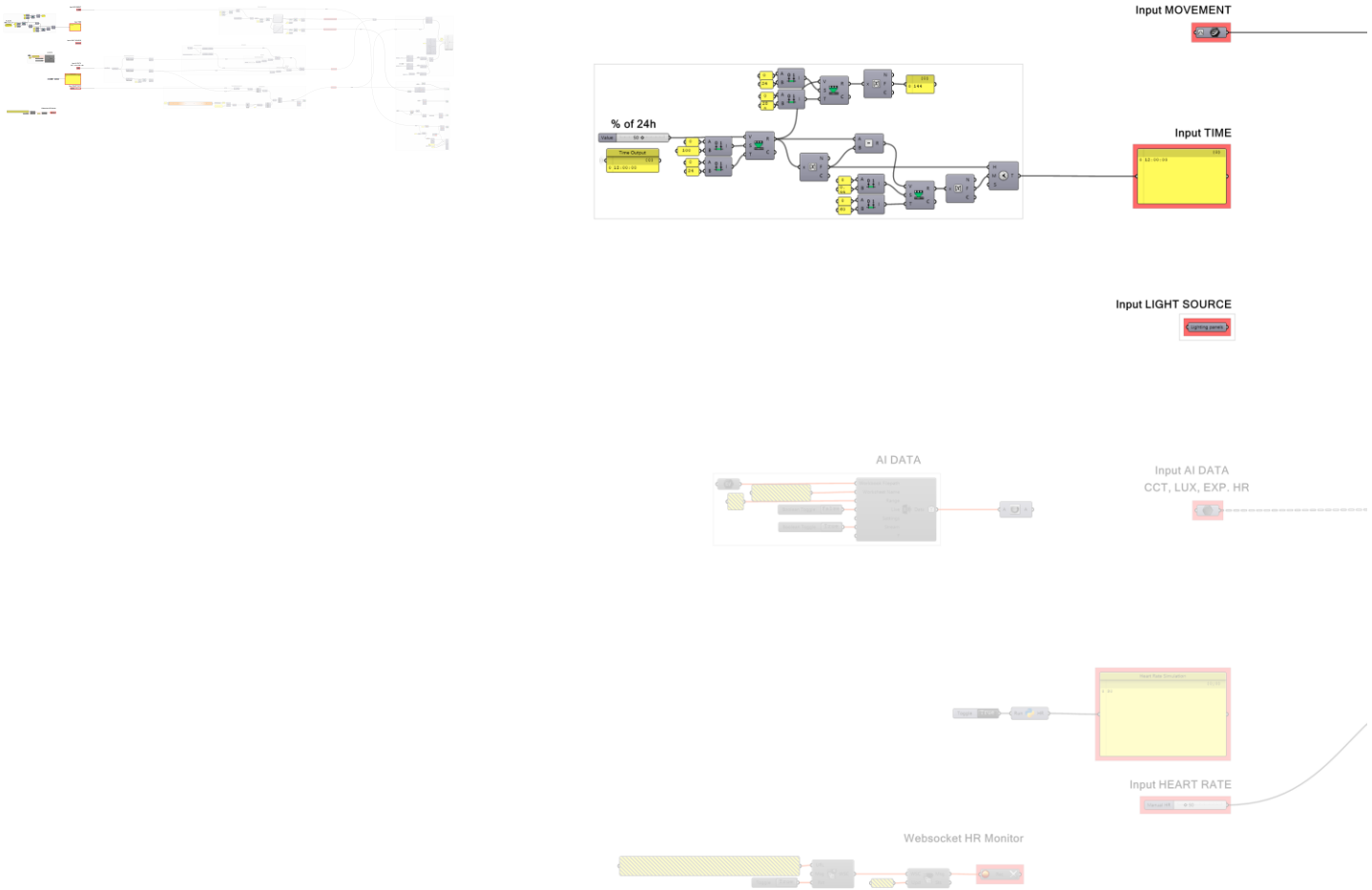


Fig. 38 – Lighting Script Grasshopper – Input Parameters

5 LIGHTING | **GH SCRIPT**

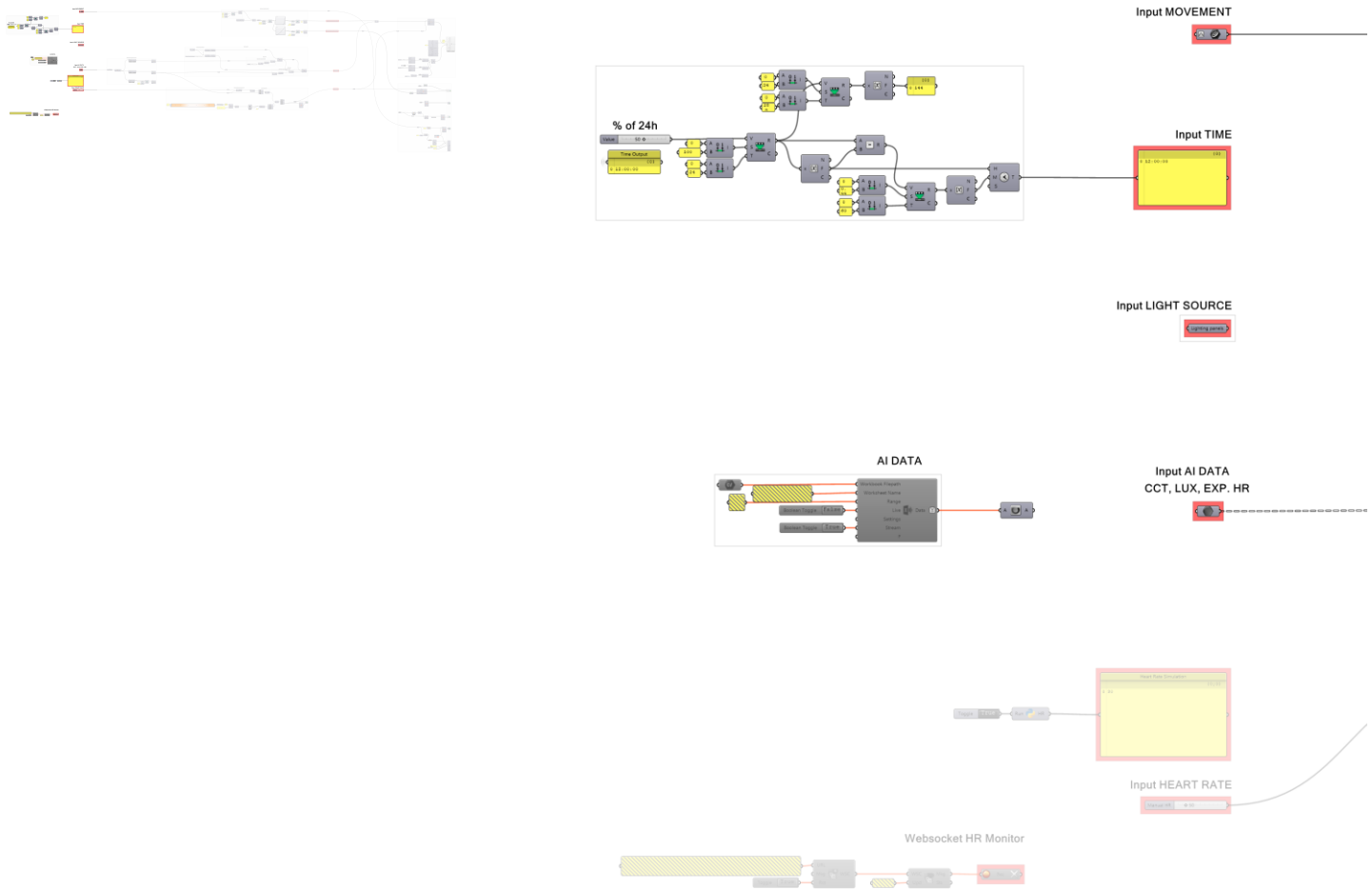
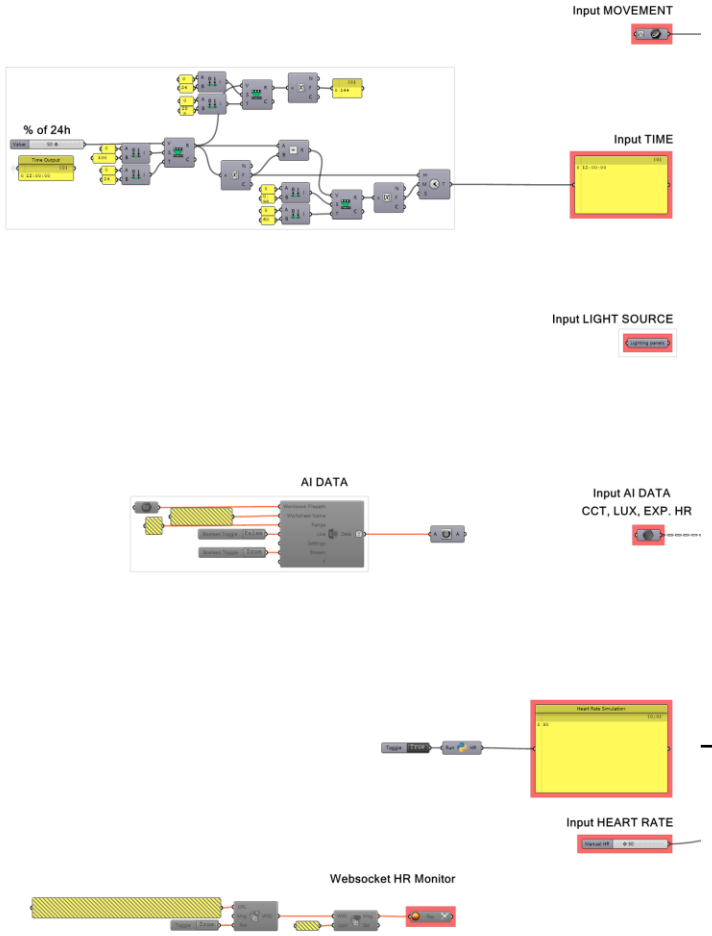
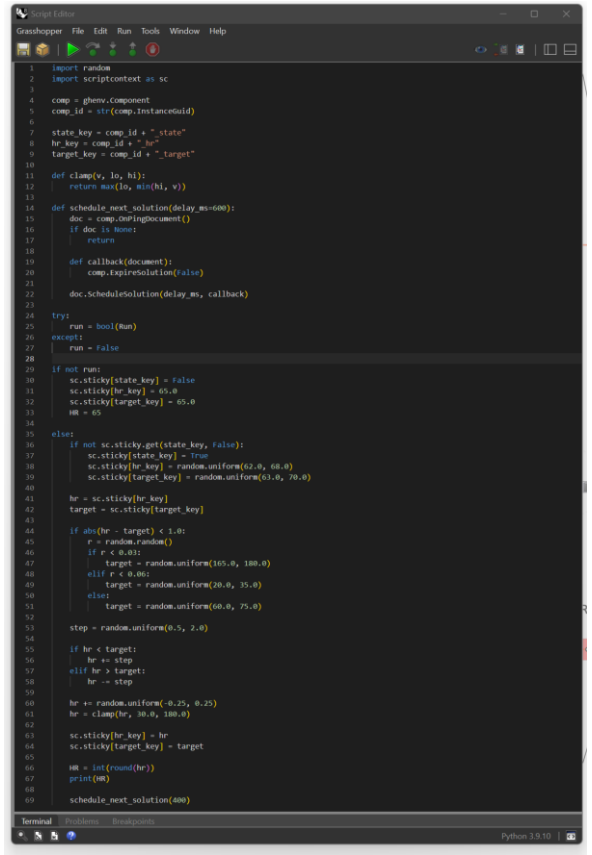


Fig. 38 – Lighting Script Grasshopper – Input Parameters

5 LIGHTING | **GH SCRIPT**

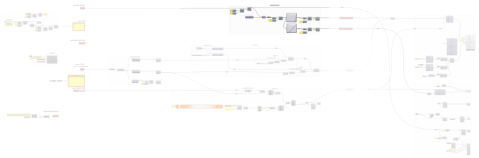




```
import random
import scriptcontext as sc

comp = ghenv.Component
comp_id = str(comp.InstanceGuid)

state_key = comp_id + "_state"
hr_key = comp_id + "_hr"
target_key = comp_id + "_target"

def clamp(v, lo, hi):
    return max(lo, min(hi, v))

def schedule_next_solution(delay_ms=600):
    doc = comp.OnPingDocument()
    if doc is None:
        return

    def callback(document):
        comp.ExpireSolution(False)

    doc.ScheduleSolution(delay_ms, callback)

try:
    run = bool(Run)
except:
    run = False

if not run:
    sc.sticky[state_key] = False
    sc.sticky[hr_key] = 65.0
    sc.sticky[target_key] = 65.0
    HR = 65

else:
    if not sc.sticky.get(state_key, False):
        sc.sticky[state_key] = True
        sc.sticky[hr_key] = random.uniform(62.0, 68.0)
        sc.sticky[target_key] = random.uniform(63.0, 70.0)

    hr = sc.sticky[hr_key]
    target = sc.sticky[target_key]

    if abs(hr - target) < 1.0:
        r = random.random()
        if r < 0.03:
            target = random.uniform(165.0, 180.0)
        elif r < 0.06:
            target = random.uniform(20.0, 35.0)
        else:
            target = random.uniform(60.0, 75.0)

    step = random.uniform(0.5, 2.0)

    if hr < target:
        hr += step
    elif hr > target:
        hr -= step

    hr += random.uniform(-0.25, 0.25)
    hr = clamp(hr, 30.0, 180.0)

    sc.sticky[hr_key] = hr
    sc.sticky[target_key] = target

    HR = int(round(hr))
    print(HR)

    schedule_next_solution(600)
```

Fig. 38 & 39– Lighting Script Grasshopper – Input Parameters, Python Script HR Simulation

5 LIGHTING | **GH SCRIPT**



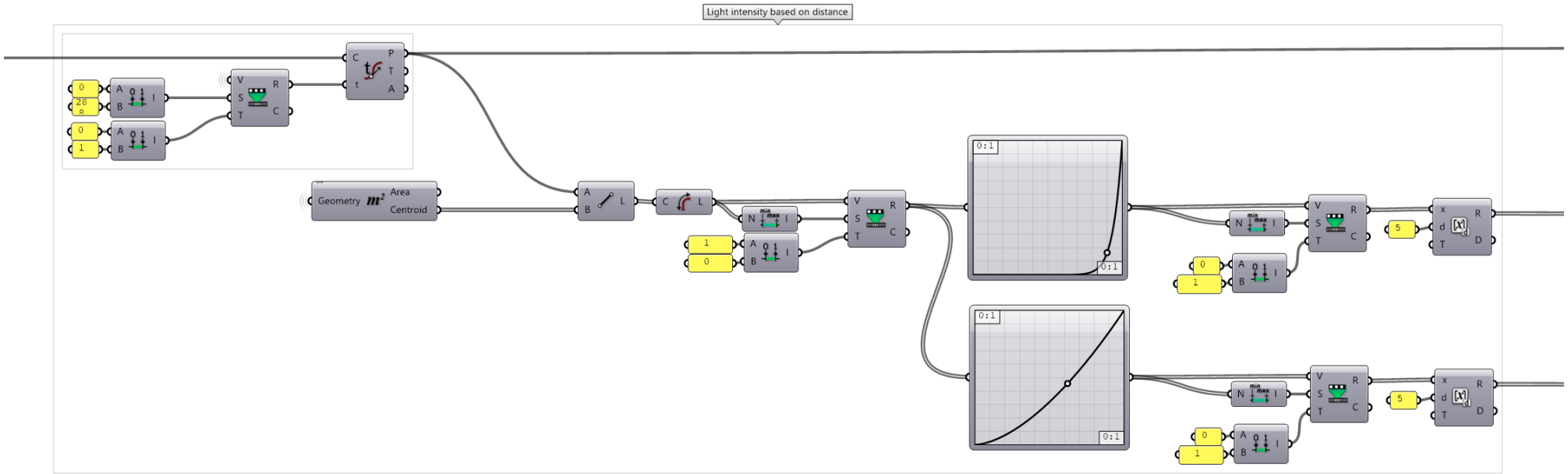
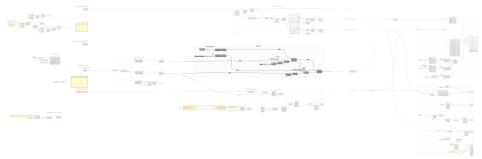


Fig. 40 – Lighting Script Grasshopper – Proximity amplification

5 LIGHTING | **GH SCRIPT**



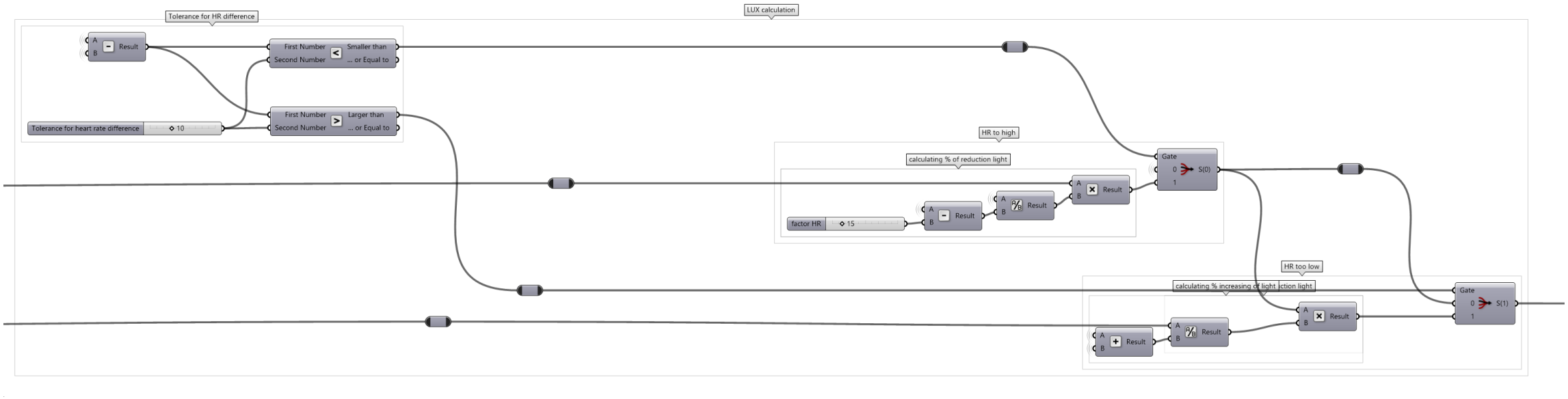
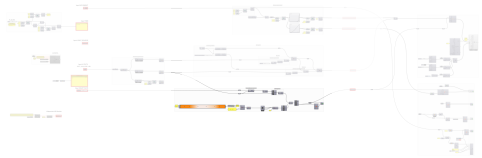


Fig. 41 – Lighting Script Grasshopper – Calculation tolerance threshold

5 LIGHTING | **GH SCRIPT**



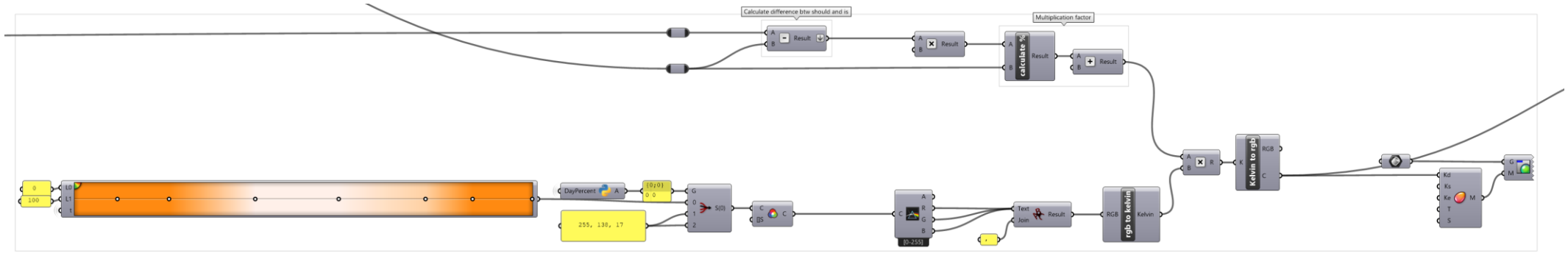
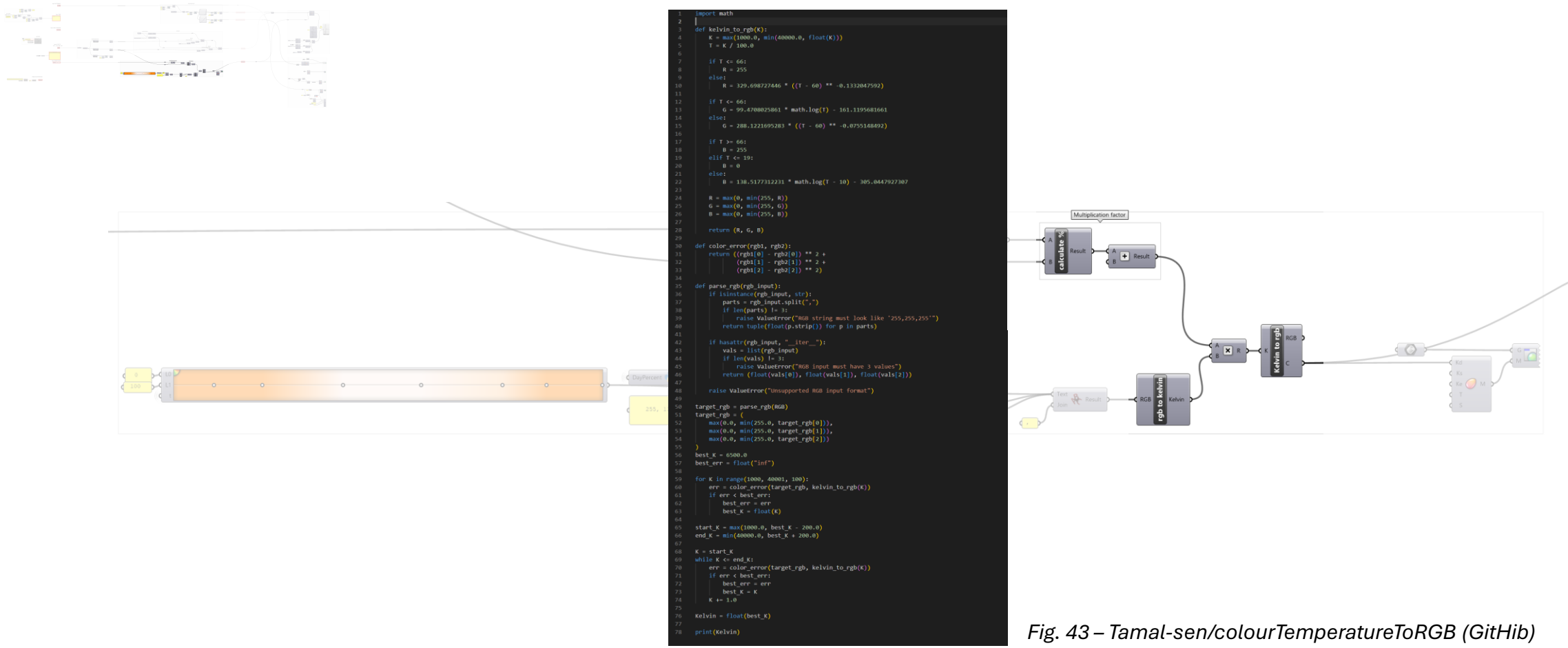


Fig. 42 – Lighting Script Grasshopper - Heart Rate–Driven colour temperature adjustment

5 LIGHTING | **GH SCRIPT**

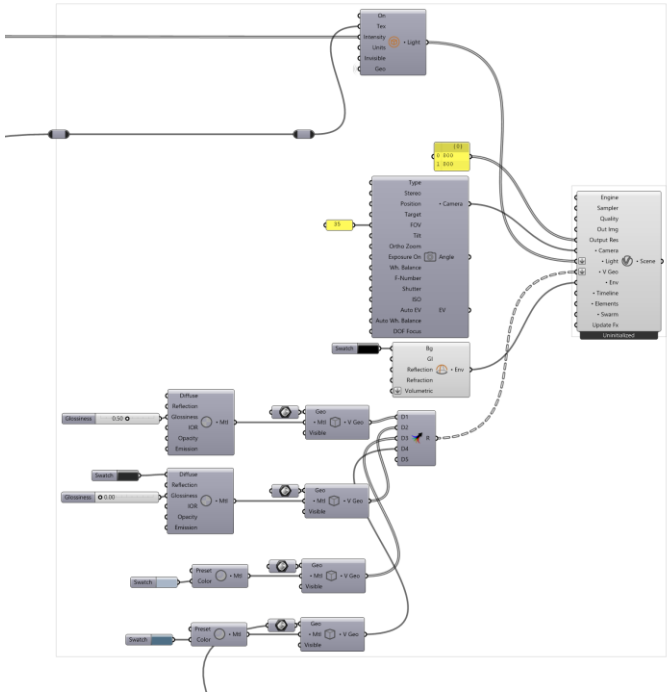


```python
import math

def kelvin_to_rgb(K):
    K = max(1000.0, min(40000.0, float(K)))
    T = K / 100.0

    if T <= 66:
        R = 255
    else:
        R = 329.698727446 * ((T - 60) ** -0.1332047592)

    if T <= 66:
        G = 99.4708025861 * math.log(T) - 161.1195681661
    else:
        G = 288.1221695283 * ((T - 60) ** -0.0755148492)

    if T >= 66:
        B = 255
    elif T <= 19:
        B = 0
    else:
        B = 138.5177312231 * math.log(T - 10) - 305.0447927307

    R = max(0, min(255, R))
    G = max(0, min(255, G))
    B = max(0, min(255, B))

    return (R, G, B)

def color_error(rgb1, rgb2):
    return ((rgb1[0] - rgb2[0]) ** 2 +
            (rgb1[1] - rgb2[1]) ** 2 +
            (rgb1[2] - rgb2[2]) ** 2)

def parse_rgb(rgb_input):
    if isinstance(rgb_input, str):
        parts = rgb_input.split(",")
        if len(parts) != 3:
            raise ValueError("RGB string must look like '255,255,255'")
        return tuple(float(p.strip()) for p in parts)

    if hasattr(rgb_input, "__iter__"):
        vals = list(rgb_input)
        if len(vals) != 3:
            raise ValueError("RGB input must have 3 values")
        return (float(vals[0]), float(vals[1]), float(vals[2]))

    raise ValueError("Unsupported RGB input format")

target_rgb = parse_rgb(RGB)
target_rgb = (
    max(0.0, min(255.0, target_rgb[0])),
    max(0.0, min(255.0, target_rgb[1])),
    max(0.0, min(255.0, target_rgb[2]))
)
best_K = 6500.0
best_err = float("inf")

for K in range(1000, 40001, 100):
    err = color_error(target_rgb, kelvin_to_rgb(K))
    if err < best_err:
        best_err = err
        best_K = float(K)

start_K = max(1000.0, best_K - 200.0)
end_K = min(40000.0, best_K + 200.0)

K = start_K
while K <= end_K:
    err = color_error(target_rgb, kelvin_to_rgb(K))
    if err < best_err:
        best_err = err
        best_K = K
    K += 1.0

Kelvin = float(best_K)

print(Kelvin)
```

Fig. 43 – Tamal-sen/colourTemperatureToRGB (GitHib)

Fig. 42 – Lighting Script Grasshopper - Heart Rate–Driven colour Temperature Adjustment

5 LIGHTING | **GH SCRIPT**



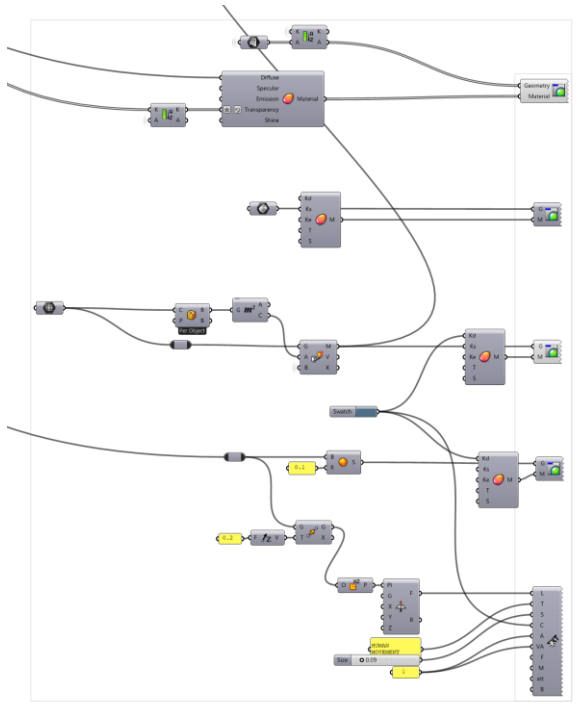


Fig. 44 – Overview Lighting Script Grasshopper - Visualisation using Chaos V-Ray & Rhino Preview

5 LIGHTING | **GH SCRIPT**

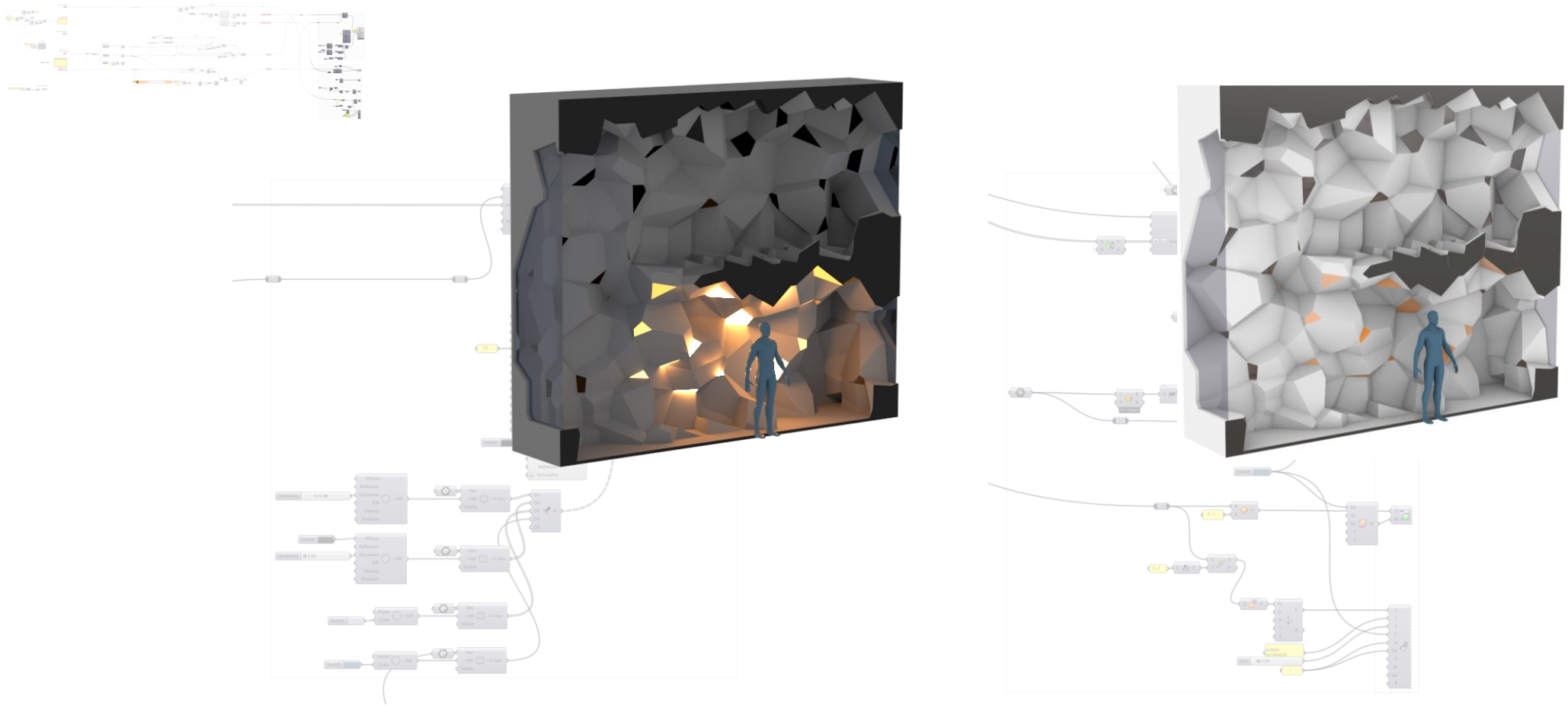


Fig. 45 – Chaos V-Ray & Rhino Preview

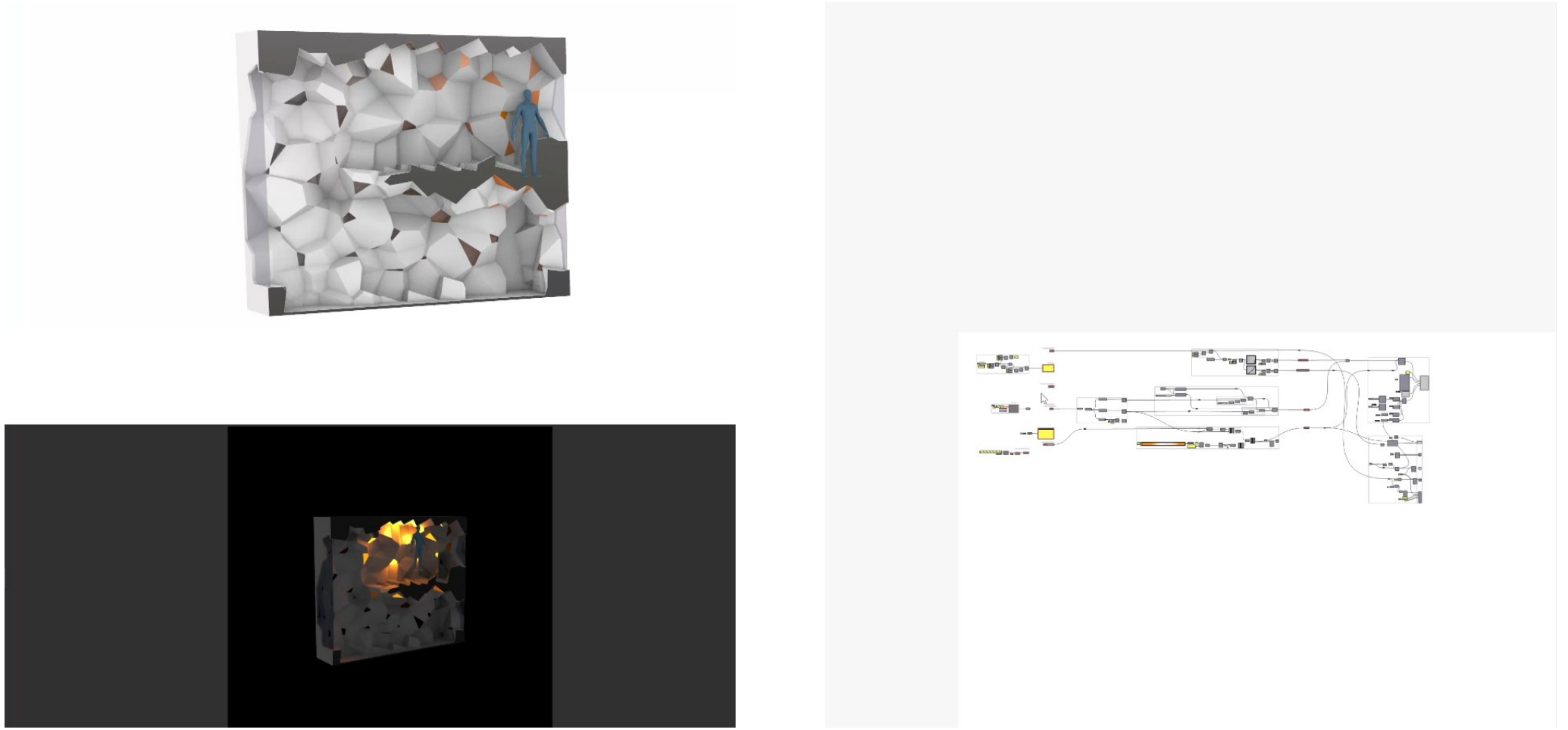


Fig. 46 – Lighting Script Grasshopper – Lighting script demonstration

CPLX

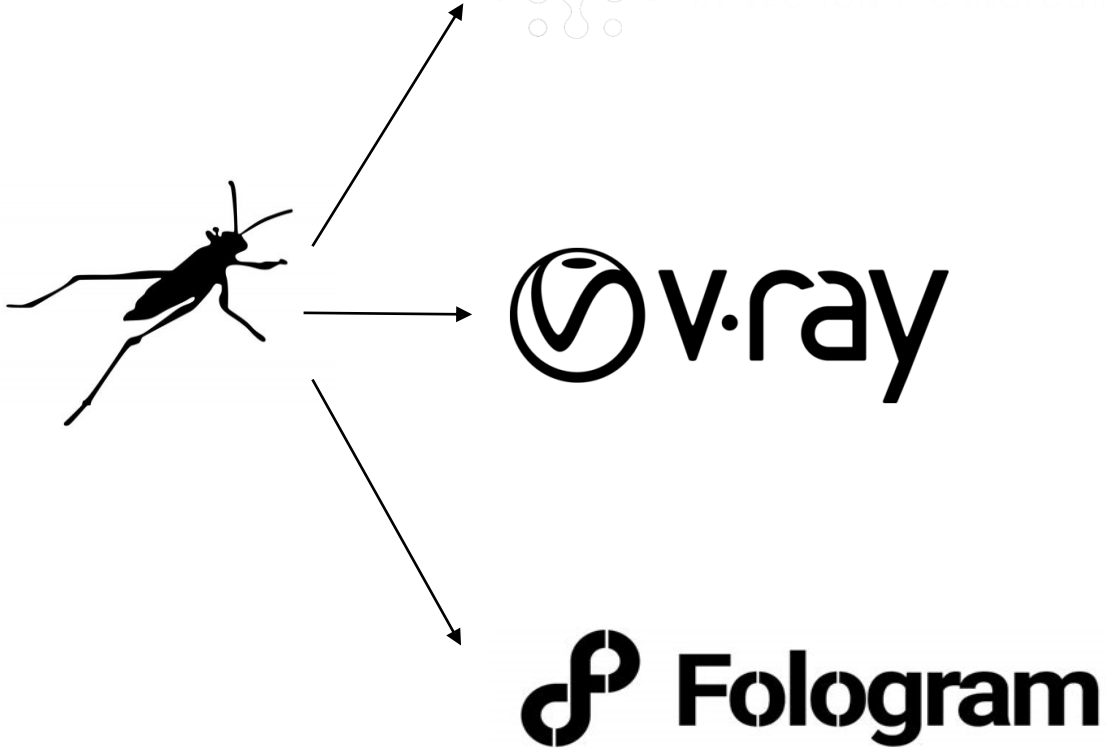
Cave Pulse Light Experiment

Group 3

AR122 1:1 Interactive Architecture Prototypes

LIMITATION

LIMITATIONS | LIGHT VISUALIZATION



+

UNREAL ENGINE

V-RAY

Fologram

ISSUES

No option for light preview, only material display

No live connection for lighting

No option for light preview, only material display

Fig. XX – Limitations in light visualization tools

CPLX

Cave Pulse Light Experiment

THE GOAL

IS

To design

a **human-centered** interior

for the

TROLL Norwegian Research Station in Antarctica

using

6m x 2.5m shipping **container** modules

THE NECESSITY

IS

To improve HEALTH by

- **Mitigating Chronic Stress**
caused by
Extreme Isolation

- **Aligning Circadian Disruption**
caused by
Lack of natural day-night cycles

- **Prioritizing Experience-First Design**

THE STRATEGY

IS

To use

- **Voronoi-based geometry**
as a
Reconfigurable Design Logic

- **AI-assisted lighting**
with an
Adaptive Pulse system

- **Circular Fabrication**
to
Convert plastic waste into 3D-printing filament

THE OUTPUT

IS

To provide

- **Cave interior** supporting sleeping, working, socializing
with
Biophilic elements and integrated climbing

- **Interactive Lighting**
with
heart rate and circadian monitoring

- **The Prototype**
of a
3D-printed cell fragment

Q&A

REFERENCES

9 REFERENCES

•Fig. 01,Rhino to Nano Banana Interior View, April 2026
https://gemini.google.com/
•Fig. 02, Blizzard at a station in Antartica, April 2026
https://www.antarctica.gov.au/about-antarctica/weather-and-climate/weather/
•Fig. 03, Map Terrain Antarctica, April 2026
https://news.osu.edu/a-terrain-map-that-shows-antarctica-in-stunning-detail/
•Fig. 04, Polar night / Aurora australis on Antarctica, April 2026
https://tinyurl.com/3fahry7b
•Fig. 05, Daylight conditions, Troll station, April 2026
Diagram produced by the authors
•Fig. 06, Troll Research Station, Norsk Polarinstitutt, April 2026
https://npolar.no/en/tone/
•Fig. 07, Functional division, April 2026
Diagram produced by the authors
•Fig. 08, Fundamental needs, April 2026
Diagram produced by the authors
•Fig. 09, Project goals, April 2026
Diagram produced by the authors
•Fig. 10, 24h activity mapping, February 2026
Diagram produced by the authors
•Fig. 11, Concept: Cave, April 2026
Diagram produced by the authors
•Fig. 12, Exterior Vision, AI generated (Nano Banana), April 2026
https://gemini.google.com/
•Fig. 13, Voronoi principle, April 2026
Diagram produced by the authors
•Fig. 14, KUKA Robot Additive Manufacture
https://www.kuka.com/it-it/settori/banca-dati-di-soluzioni/2025/09/caracol-case-study
•Fig. 15, DATA output / Schematic Print Process Fragment, April 2026
Diagram produced by the authors
•Fig. 16, DATA output / Fused particle fabrication scheme, April 2026
Diagram produced by the authors
•Fig. 17, Data implementation steps, source: Python script PolarH10, April 2026
Diagram or script output produced by the authors
•Fig. 18, Daylight Conditions, Troll Station / Interior perspective, April 2026
Figure number used inconsistently in the presentation
•Fig. 19, Spatial configuration, April 2026
Diagram produced by the authors
•Fig. 20, Diagram form finding script, April 2026
Script output or diagram produced by the authors
•Fig. 21, Diagram form finding script, April 2026
Script output or diagram produced by the authors
•Fig. 22, Video form finding script, April 2026
Script output or diagram produced by the authors
•Fig. 23, Lower Floor Spatial Configura Script output produced by the authors
Script output or diagram produced by the authors
•Fig. 24, Upper Floor Spatial Configuration, April 2026
Diagram produced by the authors
•Fig. 25, Upper and lower floor plans, April 2026
Drawing produced by the authors
•Fig. 26, Longitudinal and cross sections, April 2026
Drawing produced by the authors
•Fig. 27, Voronoi typologies, April 2026
Diagram produced by the authors
•Fig. 28, GIFF folding cabinets, April 2026
Diagram or visualization produced by the authors
•Fig. 29, GIFF entrance cabinets, April 2026
Diagram or visualization produced by the authors
•Fig. 30, GIFF privacy niche, April 2026
Diagram or visualization produced by the authors
•Fig. 31, GIFF folding table, April 2026
Diagram or visualization produced by the authors
•Fig. 32, Lighting principle, April 2026
Diagram produced by the authors
•Fig. 33, Lighting principle, V-Ray Rendering, March 2026
Diagram produced by the authors
•Fig. 34, Equalizing heartrates, April 2026
Diagram produced by the authors
•Fig. 35, Lighting principle, April 2026
Diagram produced by the authors
•Fig. 36, CCT adjustment based on activity mapping, April 2026
Diagram produced by the authors
•Fig. 37, Lighting Script Grasshopper, Overview, April 2026
Script output produced by the authors
•Fig. 38, Lighting Script Grasshopper, Input Parameters, April 2026
Script output produced by the authors
•Fig. 39, Python Script HR Simulation, April 2026
Script output produced by the authors
•Fig. 40, Lighting Script Grasshopper, Proximity amplification, April 2026
Script output produced by the authors
•Fig. 41, Lighting Script Grasshopper, Calculation tolerance threshold, April 2026
Script output produced by the authors
•Fig. 42, Lighting Script Grasshopper, Heart Rate-Driven colour temperature adjustment, April 2026
Script output produced by the authors
•Fig. 43, Tamal-sen / colourTemperatureToRGB (GitHib), April 2026
External source or adapted reference
•Fig. 44, Overview Lighting Script Grasshopper, Visualisation (Chaos V-Ray & Rhino Preview), April 2026
Visualization produced by the authors
•Fig. 45, Chaos V-Ray & Rhino Preview, April 2026
Visualization produced by the authors
•Fig. 46, Lighting Script Grasshopper, Lighting script demonstration, April 2026
Script output produced by the authors