

PROJECT MONOCEROS



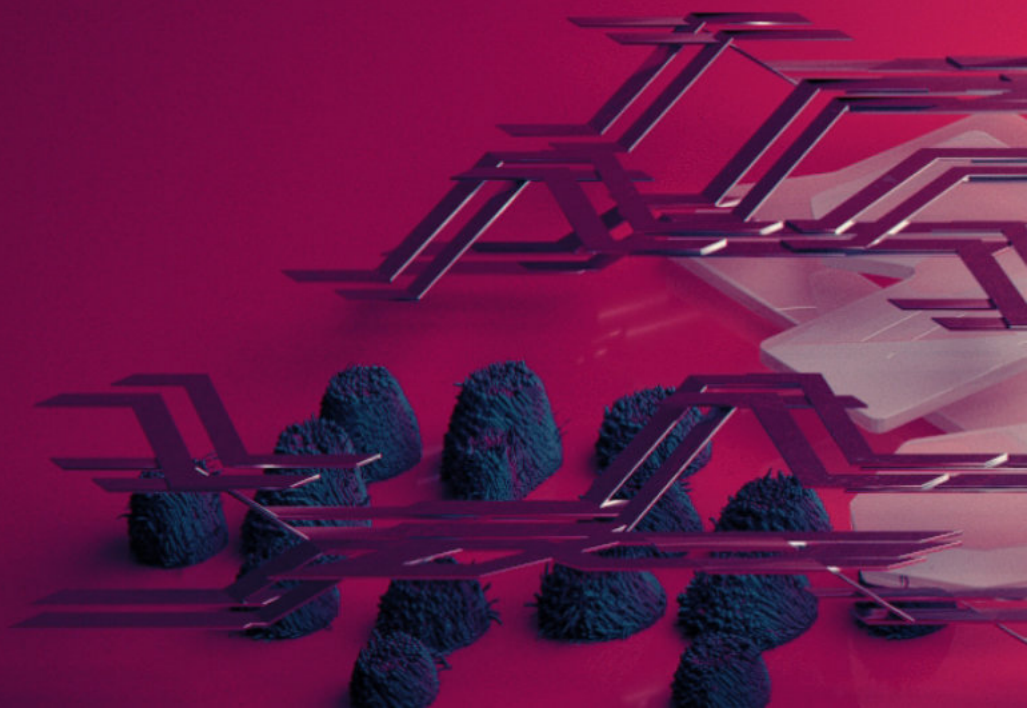
Subdigital 2021

EDITORIAL

In the summer of 2020, a group of authors Ján Pernecký, Tomáš Tholt, Ján Tóth, Michaela Krpalová, Kristýna Uhrová, Hugo Fekar, Alexander Kupko, Eva Kvaššayová, Miriam Löscher and Ľuba Ondrejkočová prepared a research project, the result of which is a proposal to adapt the mathematical method / algorithm Wave Function Collapse for architectural design purposes.

In our studio we embrace the power of computation to create architecture, design, and art. We are experts in software designed for parametric design and architecture design. We create algorithms and software solutions for complex design or architectural tasks. Algorithms are our tools, which we use to come up with unique concepts and utilize the current manufacturing technologies to their limits.

The subject of Subdigital Summer School 2020 was current and near-future big topic in computational design, prefabrication and discrete design. The early objectives were to identify the present state of the discrete design scene, as well as to learn and channel the mindset the principles of wave function algorithm while developing a comprehensive software tool that would support such design processes.





CONTENT

00 THEORETICAL INTRODUCTION

Editorial

About

01 MONOCEROS PLUGIN

Definitions and vocabulary

Inspiration from the game industry

Wfc in rust and grasshopper

New Grasshopper plug-in Monoceros

02 RESEARCH

Abstract

Contemporary tendencies in architecture

Methods

Experiments with subsets of the essential modules

Results

Conclusions

03 APPLICATIONS

Neons

Lamps and chandeliers

Playground

Housing

Hyperloop desert campus

04 PROTOTYPES

Material morphologies

05 WORKSHOPS AND EVENTS

About

People

Process

Reviews

Summary

00 THEORETICAL INTRODUCTION

INTRODUCTION TO ITERATIVE AND GROWTH METHODS

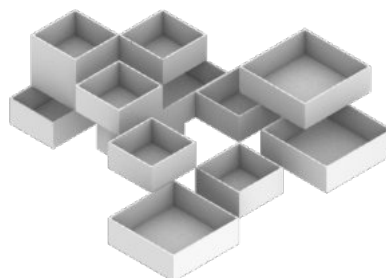
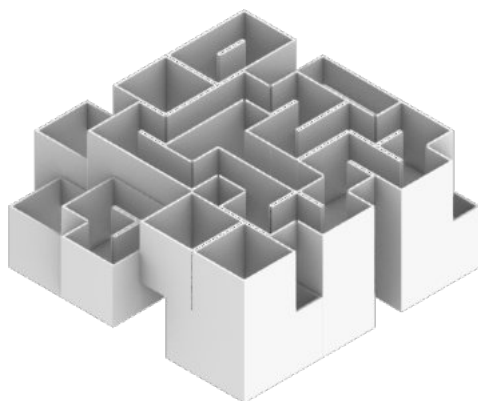
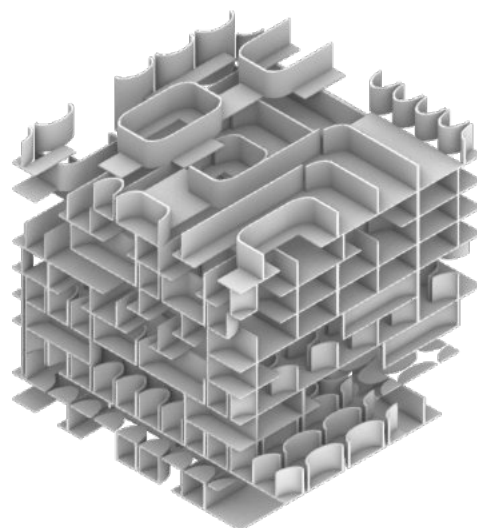
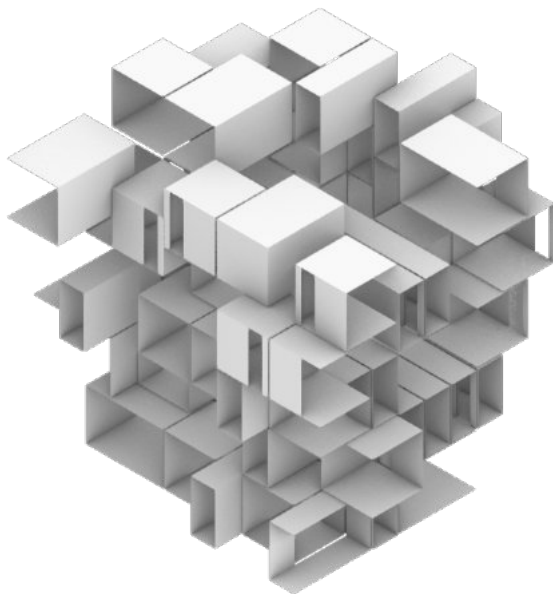
In recent years, computational design has become the main methodological discipline of most of the world's leading schools of architecture.

The key topic of this discipline is currently discrete aggregation - a method in which the assembly of a small number of distinct / unique elements creates a closed, interconnected aggregate / system.

Discrete aggregates are characterized by an organized arrangement of elements with the possibility of detailed analysis during the design process.

We developed the WFC algorithm as a full-fledged software - in the form of a free (opensource?), publicly accessible "plugin" for Rhinoceros / Grasshopper, which is the standard, widespread design software with a great range of use from parametric architecture, product design to computational art.

Our plugin is created with an emphasis on computational performance, stability and ergonomics of use close to designers and architects.



INSPIRATION FROM THE GAME INDUSTRY

Description of the algorithm

The Wave Function Collapse (WFC) is an procedural generation [0] algorithm from the constraint satisfaction problem (CSP) [1] family. Its name is inspired by quantum mechanics, but the similarity is just at the surface level - WFC is completely deterministic.

Origins of Wave Function Collapse can be traced to the demo-scene and game developer community where it is applied to lay out virtual worlds. First description of WFC is attributed to Maxim Gumin (@ExUtumno) [2], who initially used it to generate procedural images based on small samples. It was further popularized by Oskar Stålberg (@OskSta) in his EPC2018 talk about level generation in the game Bad North [3].

In its simplest form the algorithm works on a regular grid. We initially have [Figure 1]:

- A world defined by a mathematical graph [4], in our implementation represented as a regular orthogonal 3D grid.
- A set of adjacency rules describing what can occupy each tile on the grid given its neighbors.

The grid tiles are graph nodes, also called slots in WFC. Graph edges are implicitly derived from slot neighborhood: two slots are connected by an edge, if they are adjacent to each

other on the grid. Every slot contains a list of modules (e.g. images or geometry) that may reside in it. The algorithm is defined as follows:

- 1) Initialize the world to a state where every slot contains every possible module.
- 2) Repeat following steps until each slot contains exactly one module (a valid result), or some slot contains zero modules (a contradiction):
 - a) Pick a slot at random from the set of slots with the smallest-but-still-greater-than-one module count.
 - b) Randomly pick a module from the set of still available modules for that



[Figure 1]: An image from @ExUtumno's Wave Function Collapse Github project [2]. Notice how patterns in the source image (left) can be found in the generated image (right). The image was generated using an extended and thus more complicated version of WFC than what we describe here. See the Github page for more.

[0]: Wikipedia. 2020. Procedural Generation. [Online] 2020. https://en.wikipedia.org/wiki/Procedural_generation

[1]: Wikipedia. 2020. Constraint satisfaction problem. [Online] 2020. https://en.wikipedia.org/wiki/Constraint_satisfaction_problem

[2]: Github. 2020. WaveFunctionCollapse. [Online] 2020. <https://github.com/mxgmn/WaveFunctionCollapse>

[3]: Stalberg, Oskar. 2018. EPC2018 - Oskar Stalberg - Wave Function Collapse in Bad North. BUas Games. 2. 9. 2020. <https://www.youtube.com/watch?v=0bcZb-SsnrA>

[4]: Wikipedia. 2020. Graph (discrete mathematics). [Online] 2020. [https://en.wikipedia.org/wiki/Graph_\(discrete_mathematics\)](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics))

slot and remove all other possibilities, making the module count for that slot equal to one.

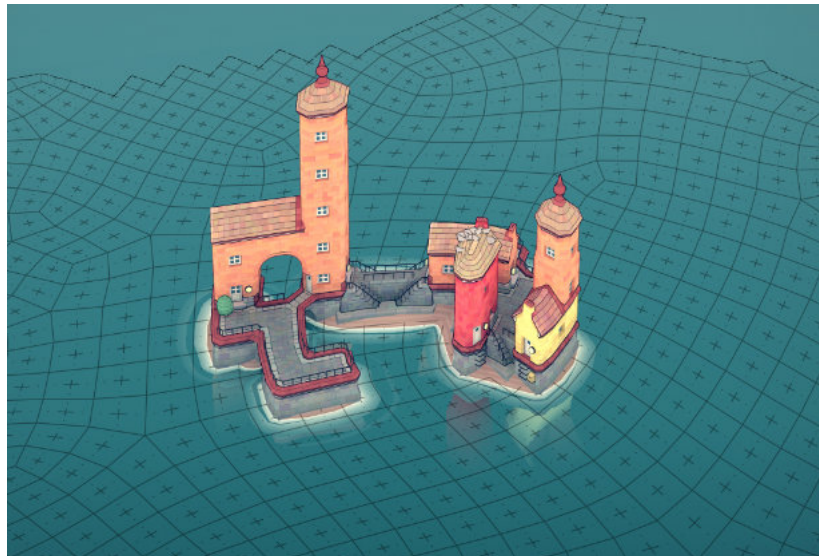
c) Remove modules from adjacent slots based on the adjacency rules. Repeat recursively for each slot modified this way.

3) If WFC generated a contradictory result, start over again with a different random state.

WFC does not necessarily always find a solution. If (and how easily) a valid solution is computed very much depends on the set of provided adjacency rules. There are sets of rules for which WFC always converges regardless of the random state, but also sets of rules for which the simulation always results in a contradictory state. Adjacency rule design is one of the more challenging parts of working with Wave Function Collapse.

WFC can be extended to work for K-

dimensional grids, or even graphs with more interesting topologies, such as the surface of arbitrary mesh geometry. More applications of the algorithm have been explored by Oskar Stålberg in [5], [6] and [7]. These discoveries culminated in the critically acclaimed town building game Townscaper [8]. [Figure 2]



[Figure 2]: Screenshot from the game Townscaper. Notice how the grid on the ground is made out of irregularly shaped polygons, but still adjacency rules can be described for this graph topology.

[5]: Stalberg, Oskar. 2020. Wave game. [Online] 2020. <http://oskarstalberg.com/game/wave/wave.html>

[6]: Stalberg, Oskar. 2020. House game. [Online] 2020. <http://www.oskarstalberg.com/game/house/Index.html>

[7]: Stalberg, Oskar. 2020. Planet game. [Online] 2020. <http://oskarstalberg.com/game/planet/planet.html>

[8]: Steam. 2020. Townscaper game. [Online] 2020. <https://store.steampowered.com/app/1291340/Townscaper/>

APPLICATIONS IN DESIGN INDUSTRY

Methodology and application

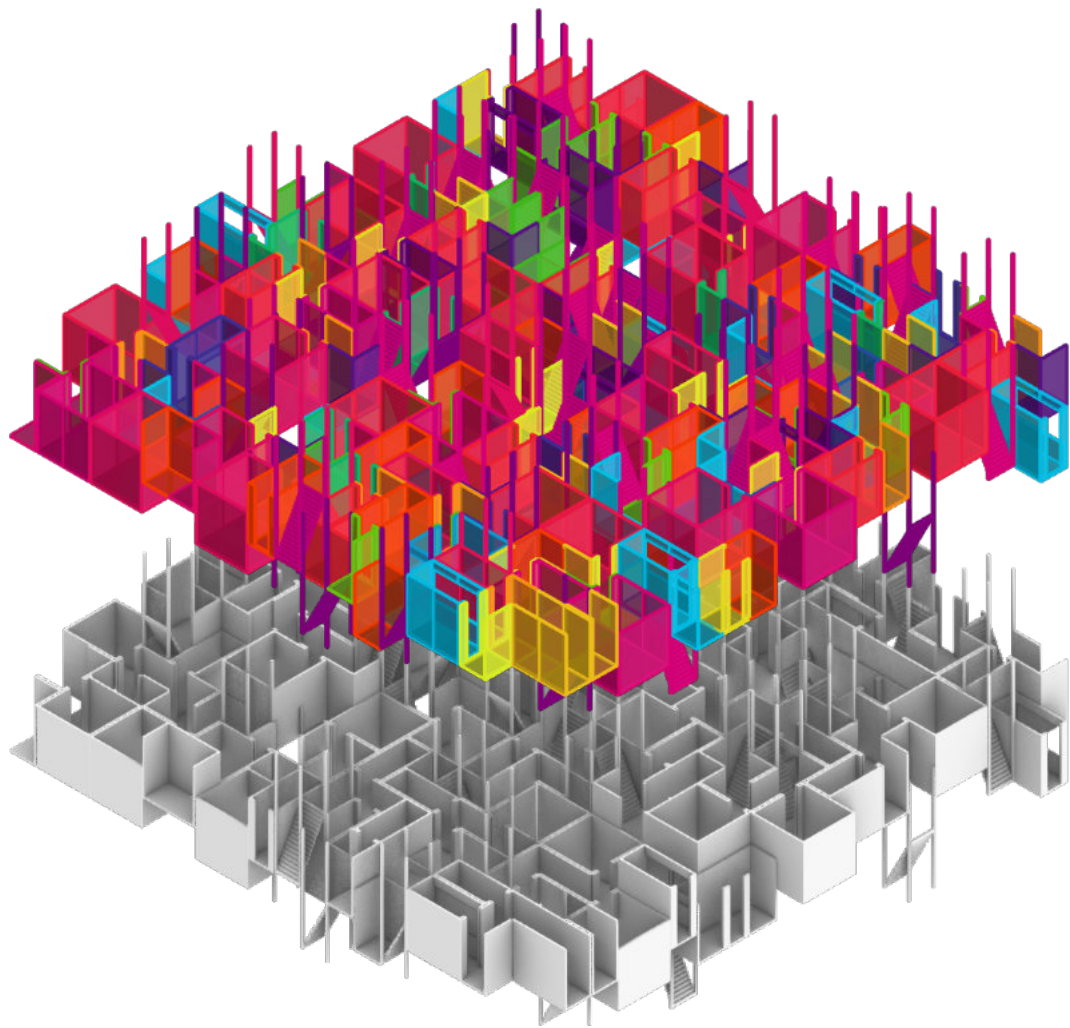
Our team seeks and creates novel, innovative solutions, new concepts and approaches for complex design or architectural tasks that we can apply in our design practice.

WCF, unlike aggregation methods, inherently creates interwoven rhizomatic structures. It almost does not rely on randomness and offers a solution rather than its simulacrum. It creates global complexity while exposing clear local rules.

WFC fills an entire segment of space with repeating prefab elements, while making sure they connect to each other properly and organize according to given rules. WFC creates diversity and complexity while maintaining the expected properties.

WFC allows building various objects in multiple scales out of a limited number of prefab element types, securing their compatibility and coherence of the resulting assembly.

With a limited number of prefab elements, it is possible to build an unlimited number of customized objects, which will have the expected functional and structural properties.



EXAMPLE WITH NUMBERS, SEED 1

Wave Function Collapse breakdown

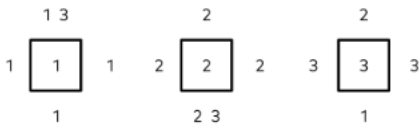
The WFC algorithm starts with Slots allowing placement of multiple listed Modules. Each Module allows only certain other Modules to be placed next to it. This is described by the adjacency Rule set. The adjacency Rules are direction-aware.

WFC follows the Rules to remove disallowed Modules from Slots adjacent to each Slot and propagates the change until each Slot allows placement of only those Modules that can be adjacent to the Modules allowed to be placed by neighboring Slots. Such state is called a Canonical world.

1 Modules



2 Rule set



3 Initial state

1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3

allowing placement of any Module

4 Observation

1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3

Choosing a random Slot and randomly picking one of the allowed Modules to be placed.

5 Propagation

1, 2, 3	1, 2, 3	2	1, 2, 3	1, 2, 3
1, 2, 3	3	3	3	1, 2, 3
1, 2, 3	1, 2, 3	1	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3

1, 2, 3	2	2	2	1, 2, 3
3	3	3	3	3
1, 2, 3	1	1	1	1, 2, 3
1, 2, 3	1, 2, 3	1	1, 2, 3	1, 2, 3
1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3

2	2	2	2	2
3	3	3	3	3
1	1	1	1	1
1, 2, 3	1	1	1	1, 2, 3
1, 2, 3	1, 2, 3	1	1, 2, 3	1, 2, 3

2	2	2	2	2
3	3	3	3	3
1	1	1	1	1
1	1	1	1	1
1, 2, 3	1	1	1	1, 2, 3

6 Result

2	2	2	2	2
3	3	3	3	3
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

2	2	2	2	2
3	3	3	3	3
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

EXAMPLE WITH MAP TILES, SEED 1

Wave Function Collapse breakdown

The WFC solver picks a random Slot and makes an "observation" - chooses a single Module from the list of Modules allowed to be placed into the Slot. This change triggers a process of eliminating disallowed Modules from the adjacent Slots, which propagates from the observed Slot outwards.

The WFC solver continues with observations until each Slot allows placement of exactly one Module. Such deterministic state of the world is the solution of the the WFC.

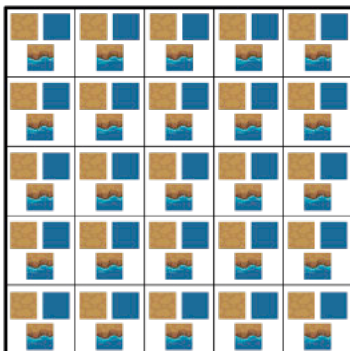
1 Modules



2 Rule set

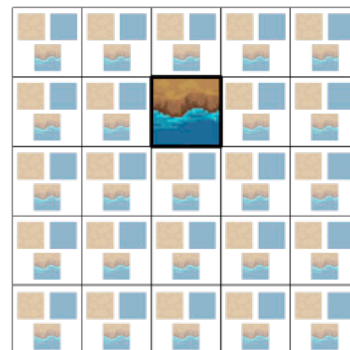


3 Initial state



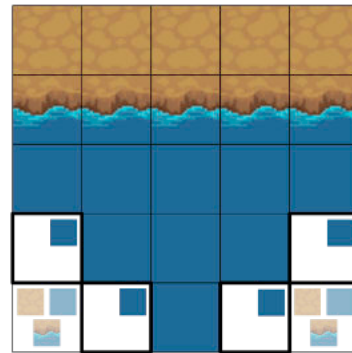
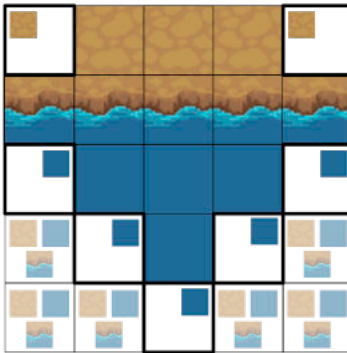
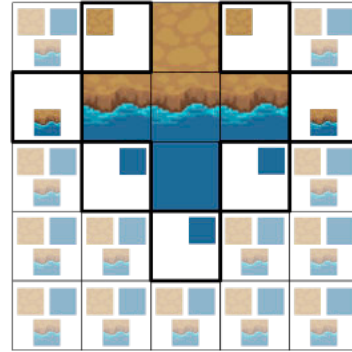
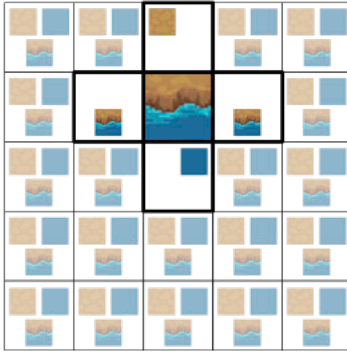
allowing placement of any Module

4 Observation

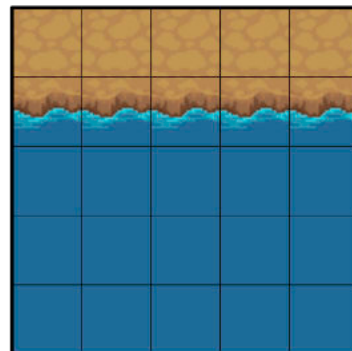
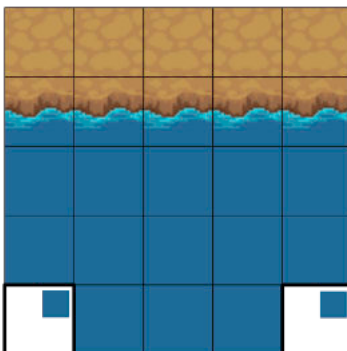


Choosing a random Slot and randomly picking one of the allowed Modules to be placed.

5 Propagation



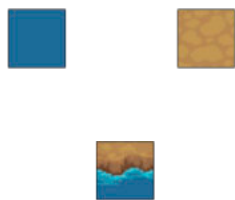
6 Result



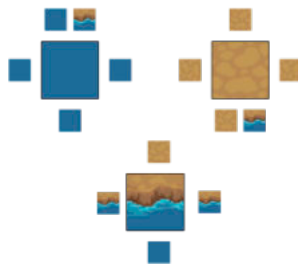
EXAMPLE WITH MAP TILES, SEED 2

Wave Function Collapse breakdown

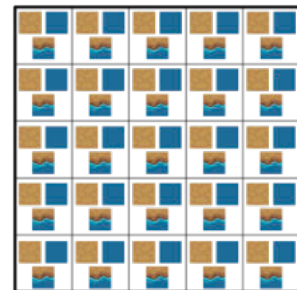
1 Modules



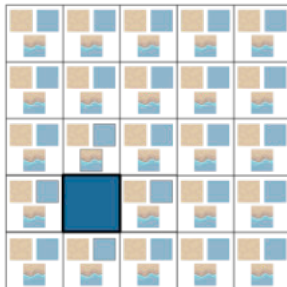
2 Rule set



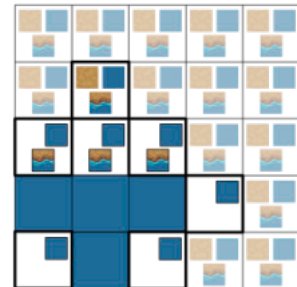
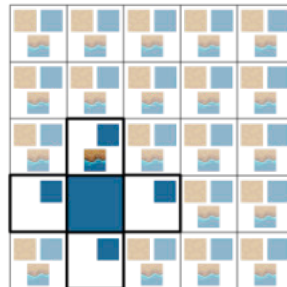
3 Initial state



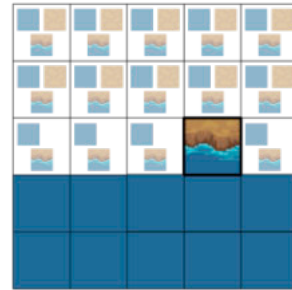
4 Observation



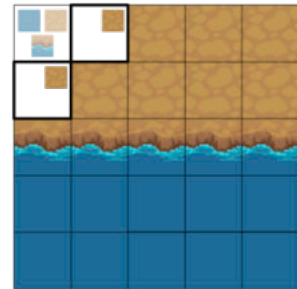
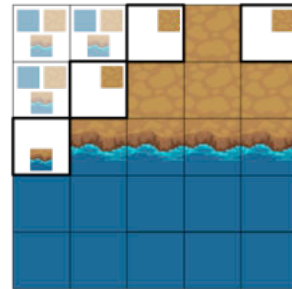
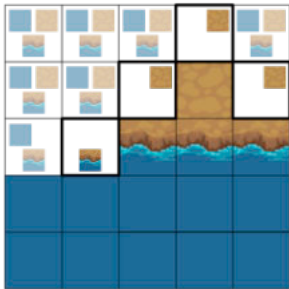
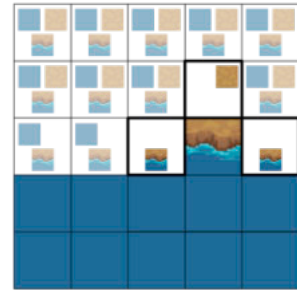
5 Propagation



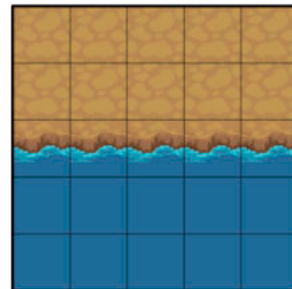
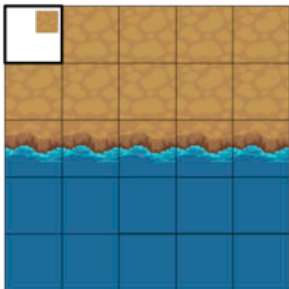
6 Observation



7 Propagation



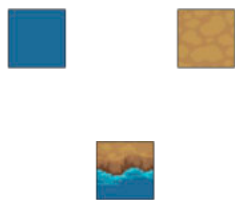
8 Result



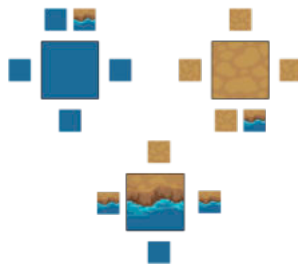
EXAMPLE WITH MAP TILES, SEED 3

Wave Function Collapse breakdown

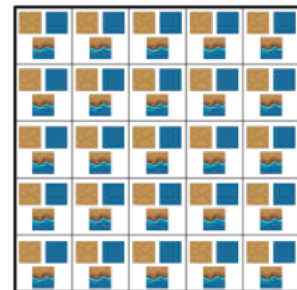
1 Modules



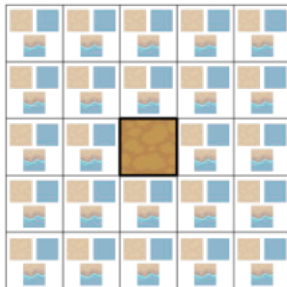
2 Rule set



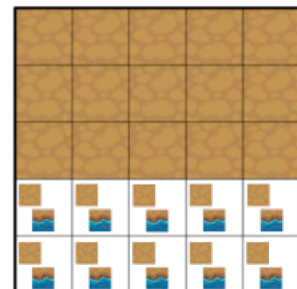
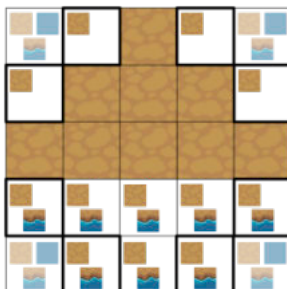
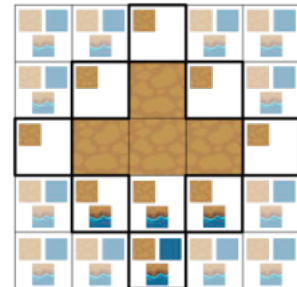
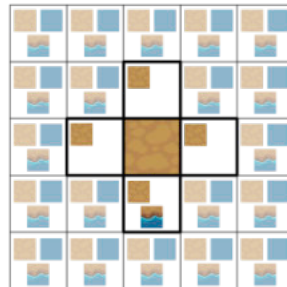
3 Initial state



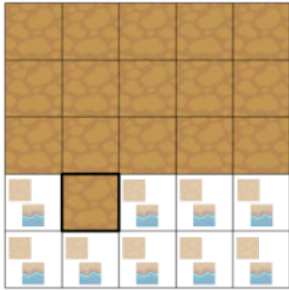
4 Observation



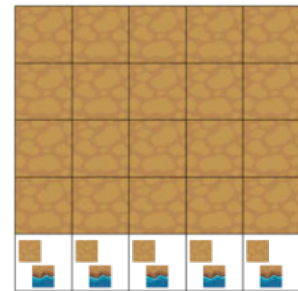
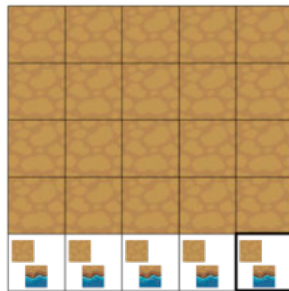
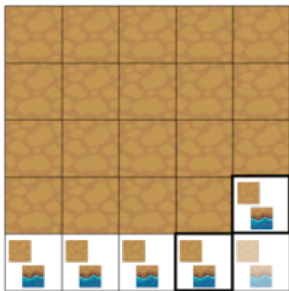
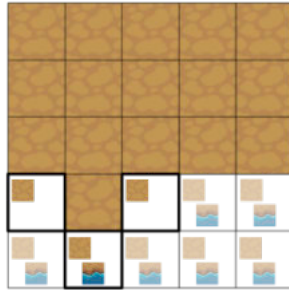
5 Propagation



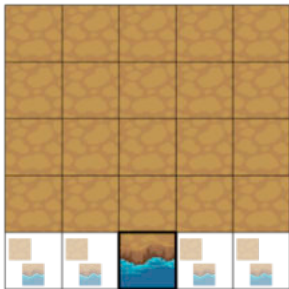
6 Observation



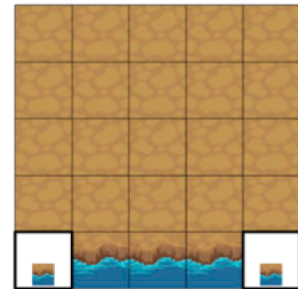
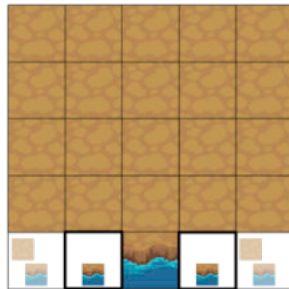
7 Propagation



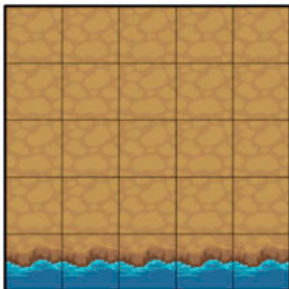
8 Observation



9 Propagation



10 Result



01 MONOCEROS PLUGIN

MONOCEROS

A Wave Function Collapse plug-in for Grasshopper by Subdigital



Monoceros is a legendary animal living in the huge mountains in the interior of India. Monoceros has the body of a horse, the head of a stag, the feet of an elephant and the tail of a boar.
from Unicorn Wiki

It is also a plug-in for Grasshopper, which is a visual programming platform for Rhinoceros 3D CAD software.

Monoceros was developed at studio Subdigital by Ján Toth and Ján Pernecký.

Monoceros is an implementation of the Wave Function Collapse (WFC) algorithm developed for game design by Maxim Gumin and extended and promoted by Oskar Stålberg with his game Townscaper.

Monoceros serves to fill the entire world with Modules, respecting the given Rules. The plug-in wraps WFC into a layer of abstraction, which makes WFC easily implemented in architectural or industrial design.

It honors the principles of WFC and Grasshopper at the same time - offering a full control of the input and output data in a Grasshopper way and their processing with a pure WFC.

Wave Function Collapse

Before we delve deeper, let's shortly explain the WFC algorithm itself. Note that Monoceros extends some concepts over the vanilla WFC, but it is very helpful to understand the original version of the algorithm first. To that end, it might also be helpful to watch the excellent explanatory talk by Oskar Stålberg at EPC2018.

Wave Function Collapse (WFC) is a procedural generation algorithm that essentially attempts to satisfy constraints on a world (see CSP for more). Its name is inspired by quantum mechanics, but the similarity is just at the surface level - WFC runs completely deterministically.

In its simplest form the algorithm works on an orthogonal grid. We initially have:

- A world defined by a mathematical graph, in our case represented as an orthogonal 3D grid.
- A set of Rules describing what can occupy each tile on the grid depending on what its neighbors are.

The grid tiles are graph nodes, also called Slots in WFC (and in Monoceros). Graph edges are implicitly

derived from Slot adjacency: two Slots are connected by an edge, if they are adjacent to each other on the grid. Every Slot contains a list of Modules that are allowed to reside in it. Conceptually, Modules could have any meaning that we assign to them, some usual meanings being that the Slots populated by them contain geometry, images, navigation meshes, or other high-level descriptions of a space.

The algorithm is defined as follows:

1. Initialize the world to a fully non-deterministic state, where every Slot allows every defined Module.

2. Repeat following steps until every Slot allows exactly one Module (a valid, deterministic result), or any Slot allows zero Modules (a contradiction):

a) Observation (Slot choice): Pick a Slot at random from the set of Slots with the smallest Module count that are still in non-deterministic state (allow more than one Module),

b) Observation (Module choice):

Randomly pick a Module from the set of still available Modules for the chosen Slot and remove all other

Modules from this Slot, making it deterministic (allowing exactly one Module),

c) Constraint Propagation: Remove Modules from neighboring Slots based on the Rules. Repeat recursively in depth-first order for each Slot modified this way.

3. If WFC generated a contradictory result, start over again with a different random state.

• **WFC does not necessarily always find a solution.** If (and how quickly) a valid solution is computed very much depends on the set of provided Rules. There are sets of Rules for which WFC always converges in just a single attempt regardless of the random state, but also sets of Rules for which the simulation always results in a contradictory world state, and also everything in between. Rule design is a challenging part of working with Wave Function Collapse.

Development notes

This repository contains the Grasshopper wrapper for the main WFC solver and comprehensive supplemental tools.

The solver itself was developed in Rust and compiled as a .dll library linked to this wrapper. The source code of the solver and a simple wrapper component for Grasshopper is in a separate repository.

The Monoceros Grasshopper plug-in is written in C# and revolves around three main data types:

1. Slot is the basic cuboid unit of a discrete world. The Slots can be embedded with Modules or their parts. Initially the Slots allow containment of multiple Modules until the WFC solver reduces the list of allowed Modules to a single Module for each Slot according to given Rules.

2. Module represents geometry wrapped into one or more cuboid cages (similar to the Slots). Modules are about to be placed into the Slots according to the given Rules.

3. Rule describes allowed neighborhood of two Modules or their parts via one of the walls of the cuboid cages - connectors.

The Monoceros plug-in offers various Grasshopper **components** (functions) for constructing and parsing the data, the solver itself and postprocessing and rendering tools.

Architecture of Monoceros Grasshopper plug-in

The core of Monoceros is a Wave Function Collapse (WFC) solver. WFC is an algorithm, that fills the entire discrete envelope with Modules with no remaining empty Slot. In case of Monoceros, the envelope is a collection of rectangular cuboid Slots, each with 6 neighbors in orthogonal directions, not taking diagonal neighbors into account.

In the original WFC algorithm, the Modules are exactly the size of a single Slot. The WFC then picks which Module should be placed into which Slot, leaving no Slot non-deterministic (with more than one Module allowed to be placed into the Slot) or empty /

contradictory (no Module allowed to be placed into the Slot). Usually, there is less Modules (Module types) than Slots, which means each Module can be placed into Slots more times or not at all.

The Monoceros implementation of WFC internally works like this too, on the outside it presents the Modules as a continuous coherent compact collection of such cuboid cages (Module Parts), each fitting into one Slot.

Like Grasshopper itself, also Monoceros revolves around data and serves for its immutable processing. Immutability means, that no existing data is being changed but rather transformed and returned as a new instance of the data. In most cases it is even possible to construct the data with valid values right away with no need to re-define already existing data.

There are three main data types: **Slots, Modules and Rules.**

Slot and Rule both reference to Module, its Part or its Connector. This reference is done only through user defined strings (for Modules and their



Parts) or integer indices (for Module Connectors). This is an intention, so that the data sets (Modules, Rules or Slots) can be replaced or shared across more Monoceros setups.

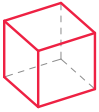
shopper and cast from and to all relevant existing data types. All Monoceros components are compatible with the existing Grasshopper data types and ready to be used with existing Grasshopper components.

Most of the Monoceros plug-in components serve for constructing, analyzing and processing data. The components try not to bring redundancy, therefore it does not do anything, that could be easily done with vanilla Grasshopper components. The three new Monoceros data types are seamlessly integrated into Gras-

Construct	Solve		Post process	
Modules	->			
Rules	->	WFC Solver	->	Materialize
Slots	->			
		Aggregate		
		Preview		

Data types

I. SLOT



Slot is a cuboid (orthogonal box) that represents the basic unit of the Monoceros rigid discrete grid. The Slots do not overlap and their position coordinates are defined in discrete numerical steps. The Slots are stacked next to each other, preferably forming a coherent continuous blob or more separate blobs that should become filled with Modules. Such blob will be called an Envelope.

I. STATES

Slot is a container that allows placement of certain Modules or their Parts. The Slot can be in several states:

- 1. Allows Nothing** is a contradictory (invalid) state, when there is no Module or its Part that can be placed inside the Slot. If the collection of Slots forming the Envelope of the solution contains one or more such Slots, the solution cannot be found and therefore such setup is invalid.
- 2. Allows one Module** (or deterministic state) is the desired state of a Slot. It

is the responsibility of the WFC Solver to bring a non-deterministic Slot into a deterministic state. Such Slot can be Materialized - a Module can be placed into the Slot.

3. Allows more Modules (or non-deterministic state) is usually an initial or intermediate state of a Slot, when it is not yet clear, which Module or its Part should be placed inside the Slot, but a list of allowed Modules is present. It is the responsibility of the WFC Solver to bring a non-deterministic Slot into a deterministic state. A non-deterministic Slot cannot be Materialized yet but it is possible to evaluate its level of entropy - a number of currently allowed Modules or their Parts stating how far from the deterministic state the Slot is.

4. Allows all Modules is a shortcut for a fully non-deterministic state, when any Module or its Part is allowed to be placed inside the Slot. In practice this state exists only for cases, when Slots are being defined before or independently from Modules and therefore it is not possible to list Modules that should be allowed by the Slot.

The collection of Slots forming an Envelope is considered Canonical if the adjacent Slots allow placement of such Modules or their Parts, that are

allowed to be neighbors by the Rules. Canonical Envelope does not have to be also deterministic. For non-deterministic Slots to form a Canonical Envelope, each allowed Module or its Part in one Slot must be allowed to be a neighbor of each allowed Module or its Part in its adjacent Slot in the given direction by the Rule set.

The Monoceros implementation of the WFC algorithm can automatically clean a Non-Canonical Envelope into Canonical, which takes a lot of responsibility from the user and enables future development of Monoceros features.

II. SLOT PROPERTIES

- **Center** of the Slot in cartesian coordinate system. The coordinate is automatically rounded so that it represents an exact center of the Slot in the discrete world coordinate system.
- **Base Plane** defining Slot's coordinate system. The discrete world coordinate system's origin and axial orientation matches that of the Base Plane. For two Slots to be compatible, their Base Planes must match.
- **Diagonal** defining Slot's dimensions in X, Y and Z directions as defined by the Base Plane. For two Slots to be

compatible, their Diagonals must match.

- **Allowed Module Names** is a list of Modules that are allowed to be placed (entire Modules or their Parts) inside the Slot. This list is empty, when the Slot is in the Allows Nothing or Allows Everything state.

- **Allows Everything** is a flag determining whether the Slot allows placement of any Module or its Part.

- **Allows Nothing** is a flag determining whether the Slot allows placement of no Module or its Part. Such Slot is invalid and prevents the WFC Solver from reducing other Slots from non-deterministic to deterministic state.

III. AUTOMATIC ENVELOPE WRAPPING

The WFC algorithm (the original one and the Monoceros implementation) work with a full regular three-dimensional box-like Envelope. Monoceros allows the user to define any set of Slots (as long as they are compatible and non-repetitive), which form any arbitrary blob.

Therefore the Monoceros WFC Solver component automatically wraps the user-defined Slots into a slightly larger bounding box and adds the

missing Slots. These Slots are dubbed Out-enabled and are pre-determined to allow only one type of a Module to be placed inside them: the Out Module. Out Module is automatically generated by the Monoceros WFC Solver, it has the same diagonal as the Envelope Slots, has just a single Part and contains no geometry. The Out Module has all connectors defined as Indifferent, therefore it is allowed to be adjacent to itself and with any other Module Indifferent in the respective direction. The Out Module and an Empty Module are distinguished so that it is possible to set a Rule for a Module Connector to be adjacent to Out Module (while not being Indifferent), which allows the Module to be placed at the boundary of the Envelope.

All boundary Slots are ensured to be surrounded by Out-enabled Slots. The Out-enabled Slots are not being displayed in the Rhinoceros viewport.

IV. MODULES AND THEIR PARTS

The Monoceros Modules may consist of more Parts. Each Part is of the size of a single Slot, therefore a larger Module occupies more Slots. For valid Modules it is ensured that the Module Parts always hold together and all of

the Module Parts are always placed into Slots in the original order. Internally, the Slots refer to Module Parts, for the user of Monoceros Grasshopper plug-in are the Module Parts inaccessible and the Module appears as the smallest unbreakable unit. Therefore it is only possible to define entire Modules to be allowed by a Slot, even when such Module consists of more Parts. Internally this means, that all Parts of the Module are allowed to be placed into a Slot. In practice this should not cause any problems or imprecisions. It is being automatically handled by the WFC Solver and offers some buffer zone for complex Module placement.

V. VIEWPORT PREVIEW AND BAKING

A Slot renders in the viewport as a wire frame box. The box is slightly smaller than the actual Slot, so that it is possible to distinguish colors of two adjacent Slots from one another.

A Slot bakes as a regular Rhino box, with dimensions matching the Slot size.

A Slot renders and bakes in different colors, indicating the level of Slot's entropy:

- **Red** - the Slot does not allow placement of any Module or its Part, therefore is empty and invalid.
- **White** - the Slot allows placement of any Module or its Part that enters the Monoceros WFC Solver.
- **Blue** - the Slot allows placement of more than one Module or its Part but the overall number of Modules is unknown to the Slot. This is a valid state for Slots defined by enumerating the allowed Modules. A Slot is blue even when all Modules are allowed, but they have been listed namely instead of using the Allow all constructor.
- **Green** - the Slot allows placement of exactly one Module Part and is ready to be Materialized. This is currently only possible for Slots processed by the Monoceros WFC Solver. Green color denotes the desired state of a Slot.
- **Grey** - the Slot allows placement of some Modules or their Parts and the overall number of Modules and their Parts is known. Bright color indicates a high level of entropy - more Modules or their Parts are (still) allowed to be placed into the Slot. Dark grey means the Slot has a lower entropy and is

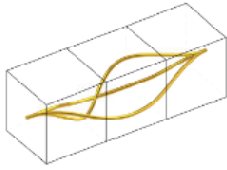
closer to a solution.

VI. SLOT CASTS TO

The Casts are meant to shorten the de-construction and re-construction of a Slot. With the following casts it is possible to use one instance of a Slot to construct a new one with the same properties by passing the Slot into individual input Slots of a Slot constructor component.

- **Point** - representing the center point of the Slot
- **Box and BRep** - representing the exact box cage of the Slot
- **Vector** - representing the Diagonal of the Slot
- **Text** (string) - returns a human-friendly report of the Slot's properties in format: Slot allows placement of XY Modules. Slot dimensions are XYZ, center is at XYZ, Base Plane is XYZ X Y.

II. MODULE



Module is a unit, which is being distributed over the specified Envelope. The main purpose of the WFC and Monoceros is to decide, which Module or its Part can be placed into which Slot, so that the adjacencies of Modules follows the specified Rule set. If this requirement is met, it means the Envelope is Canonical. If there is exactly one Module or its Part allowed to be placed into every Slot, the Envelope is Deterministic and solved.

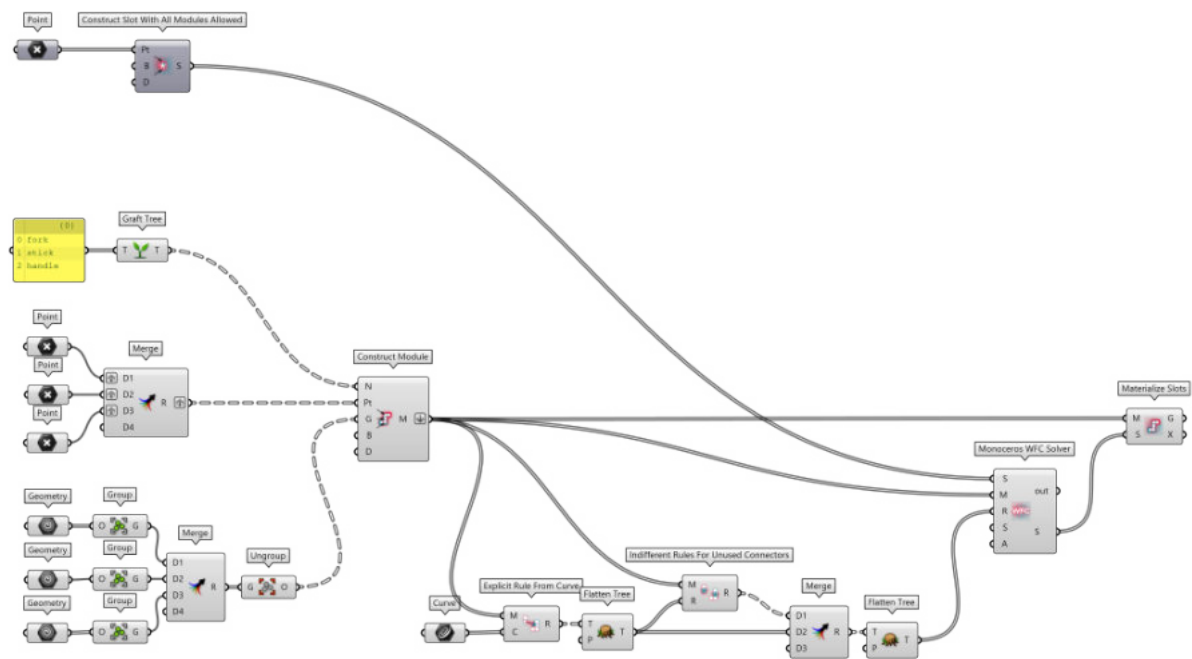
I. MONOCEROS MODULE PARTS

offers a possibility for Modules to span over (occupy) more Slots. In such case, In the original WFC algorithm, the Module occupies exactly one Slot. Monoceros the Module consists of more Parts, each of a size of a single Slot. If the Module is compact (continuous, consistent) then the Parts always hold together (this is secured by the Monoceros WFC Solver). The Module Parts are not accessible individually, the Module is rather presented as a single element.

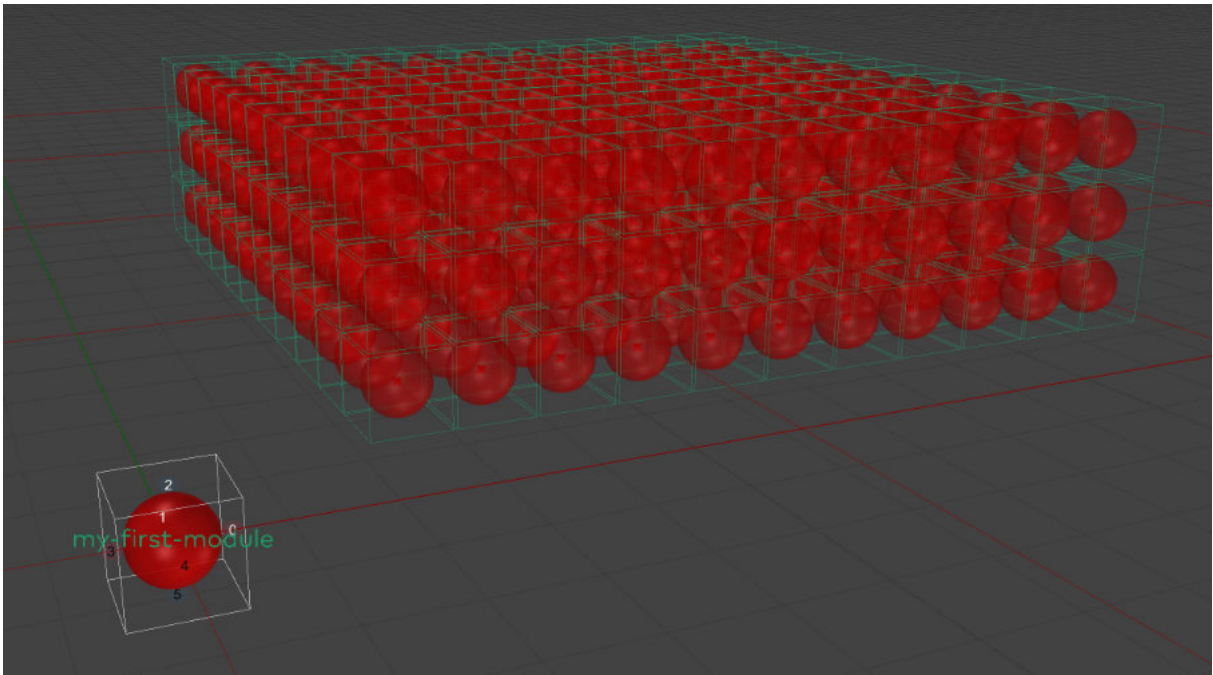
A Module can consist of a single Part or more Parts. The Parts that are entirely surrounded by other Parts in all 6 orthogonal directions, are not being presented to the user at all. Only the boundary walls of Module Parts are visible and form an orthogonal discrete unit cage of a Module.

Examples

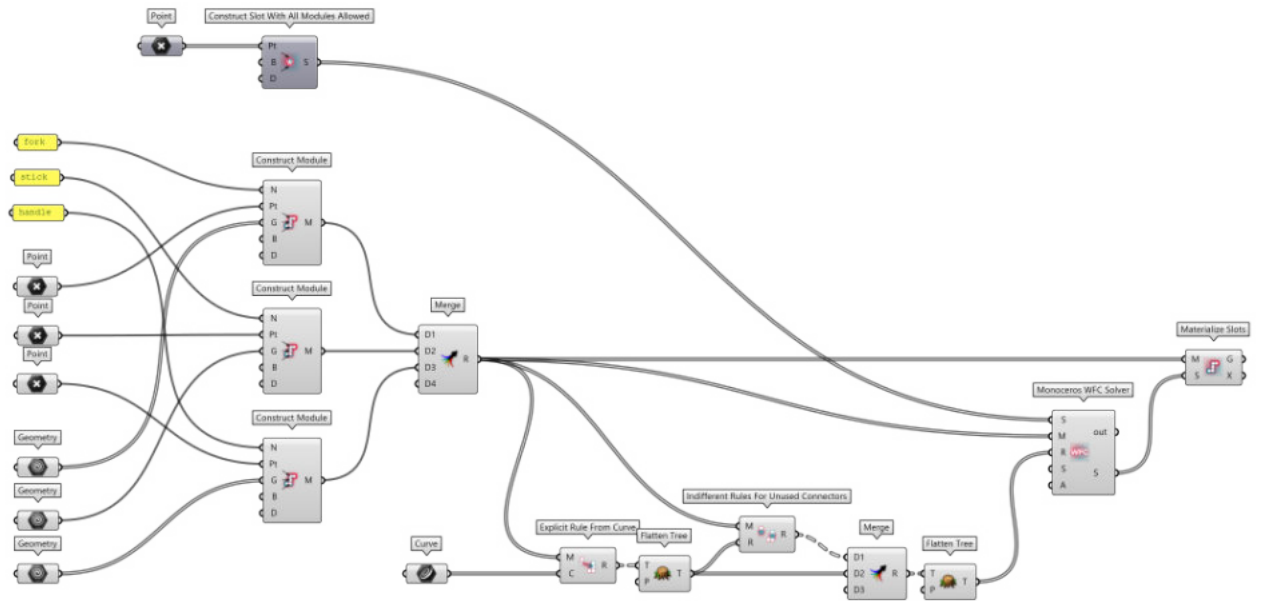
Definition



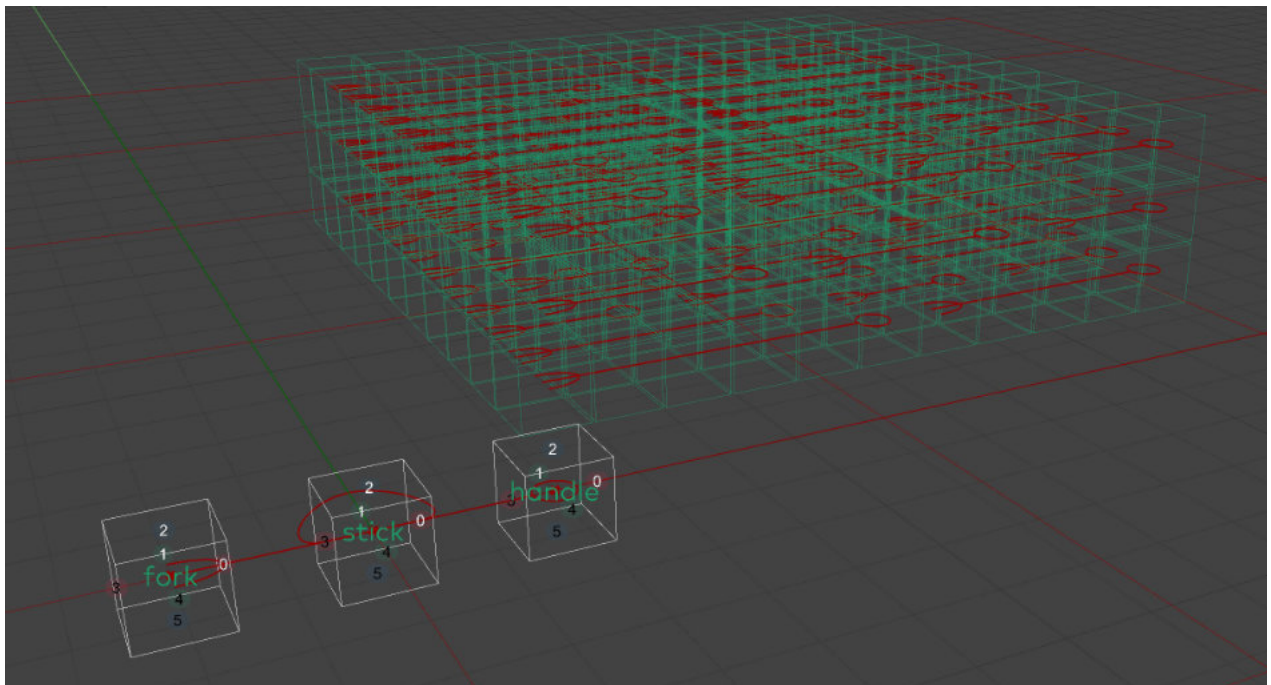
Result



Definition



Result





02 RESEARCH



OUR DEFINITIONS

Main features of the algorithm WFC

Architectural

Wave Function Collapse (WFC) fills an entire segment of space with repeating prefab elements, while making sure they connect to each other properly and organize according to given rules. WFC creates diversity and complexity while maintaining the expected properties.

Conceptual

WFC, unlike aggregation methods, inherently creates interwoven rhizomatic structures. It almost does not rely on randomness and offers a solution rather than its simulacrum. It creates global complexity while exposing clear local rules.

Practical

WFC allows building various objects in multiple scales out of a limited number of prefab element types, securing their compatibility and coherence of the resulting assembly.

Economical

With a limited number of prefab elements, it is possible to build an unlimited number of customized objects, which will have the expected functional and structural properties.

With discrete aggregation methods, we build complex structures with emergent interactions from a finite set of prefabricated building blocks. The declarative property of the methods enables us to generate a variety of candidate solutions from which we choose based on both algorithmic and subjective criteria.

Our approach to design, fabrication and assembly merges the modernist prefabrication with parametric complexity and customization. We are able to precisely control complex patterns of spaces, purposes, structures and materials to build effectively high-performance architecture of the 21st century with fabrication methods of industry 4.0.

Aim

To develop and master a meaningful materialization method of data-driven design with available fabrication methods.

Software

To develop full-fledged software tools tailored for academic and commercial needs of Subdigital.

Process

To generate a multitude of abstract and applied examples in various scales, that can be later used for partial tasks in architectural design.

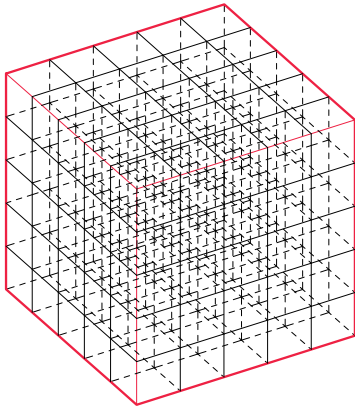
Discretization

With a limited number of prefabricated elements, it is possible to build an unlimited number of customized objects, which will have the expected functional and structural properties.

OUR VOCABULARY

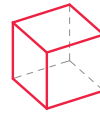
World

Block of space divided into heterogeneous orthogonal grid.



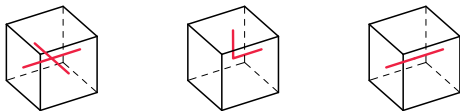
Slot

Single unit in the world grid. A slot may contain a module.



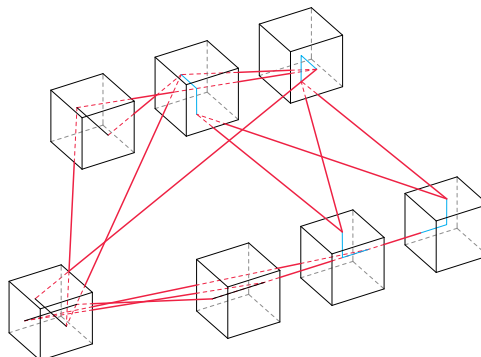
Module

Geometry that fits into a single slot. A module is about to be placed into slots.



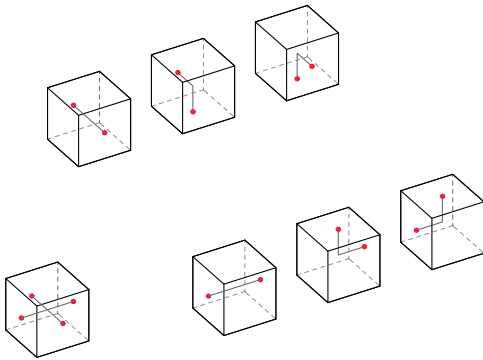
Explicit connection

Single rule explicitly allowing two modules to be neighbors in the specified direction.



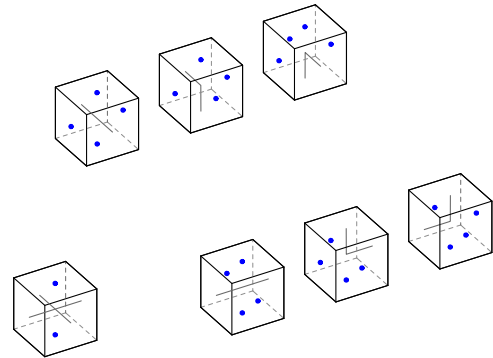
Rule set

List of allowed neighbor modules for each module in each direction.

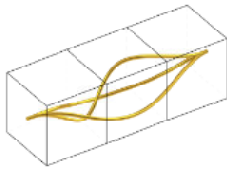


Indifferent connection

A module with no explicit connection in certain direction is indifferent in that direction and can connect to any module indifferent in the opposite direction.



Module



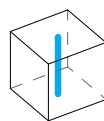
EXPERIMENTS WITH BASIC MODULES

57 essential single modules

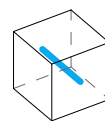
The library of shapes are abstract geometries indicating just their direction and orientation, set in regular cubic grid. The modules have 6 possible ways of aggregation, and geometries have multiple possibilities of rotations. With a goal of creating complex, peculiar solutions of given space, we came up with 56 different modules and then tested their combinations in terms of solvability.

Why some attempts are complicated and other are rigid? After some time of testing it became obvious that achieving more complicated, inter-connected 3dimensional solution of the space is dependent not only by the number of used modules within the set but also by the inner limitation of modules.

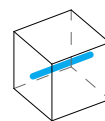
Straight shapes defined by only one of the axes, therefore with two connectors and without any other limitations and rules will fill up space with rigid, repetitive shapes. More limited modules, two or three-axed modules, with more specifically defined, exclusive connections, and the ability of one-axed modules to chain themselves are some of the aspects of results complexity.



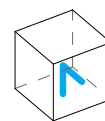
0



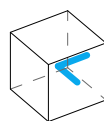
1



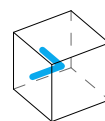
2



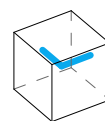
3



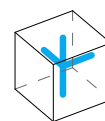
12



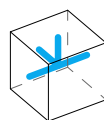
13



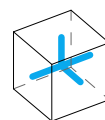
14



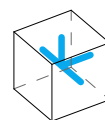
15



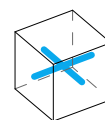
24



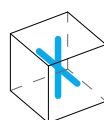
25



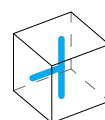
26



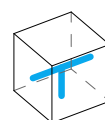
27



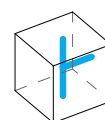
36



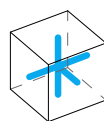
37



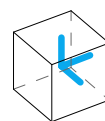
38



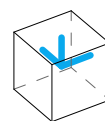
39



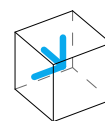
48



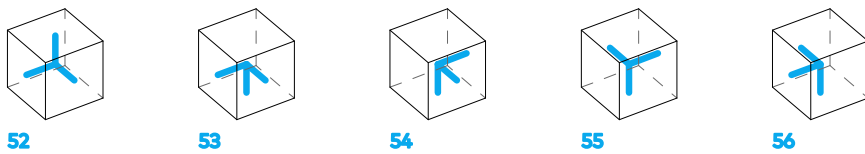
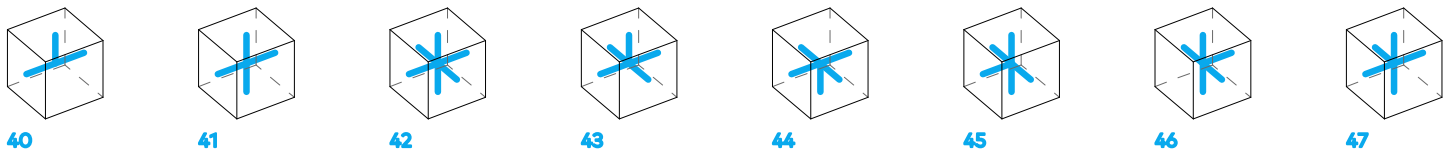
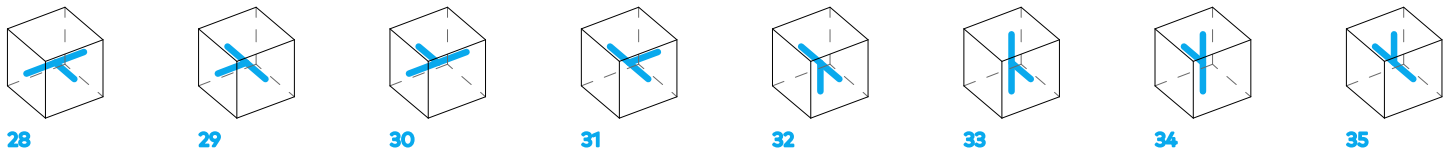
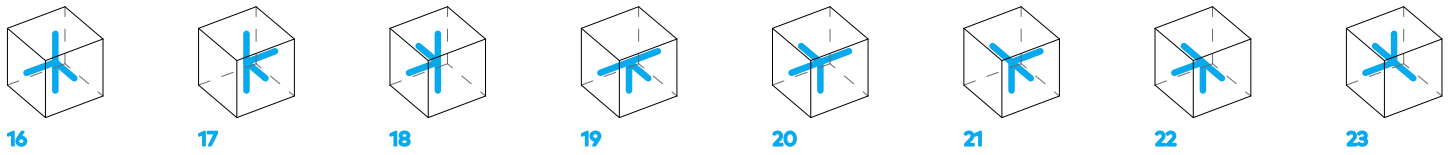
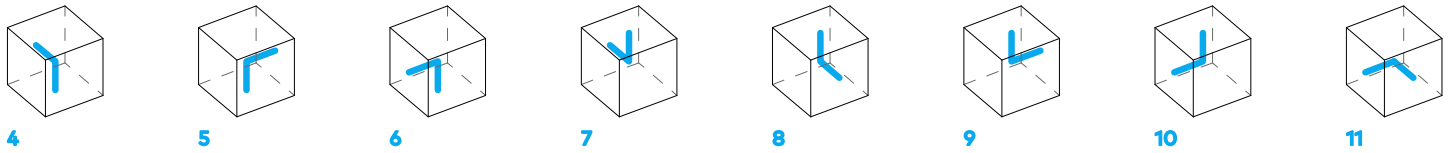
49



50



51



BASIC MODULES

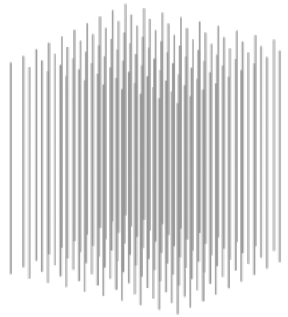
Experiments with subsets of the essential modules

1 module

World size 10x10x10 slots

Number of explicit connections 1

Number of rules

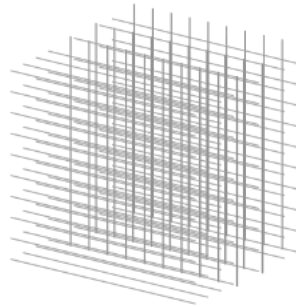
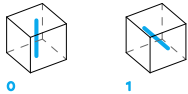


2 modules

World size 10x10x10 slots

Number of explicit connections 2

Number of rules

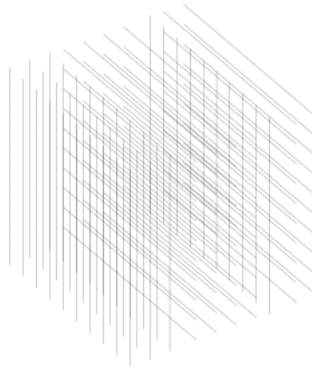
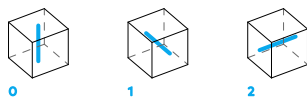


3 modules

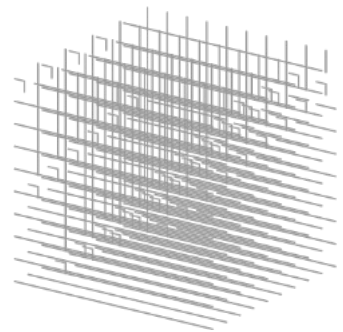
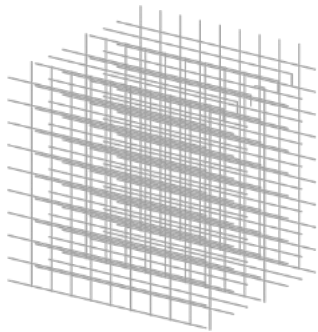
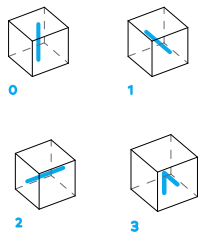
World size 10x10x10 slots

Number of explicit connections 3

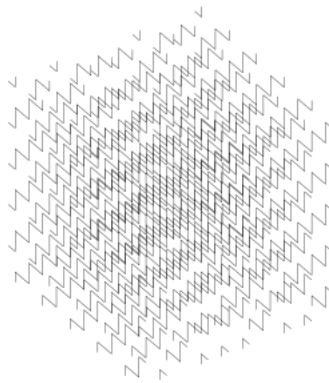
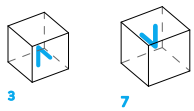
Number of rules



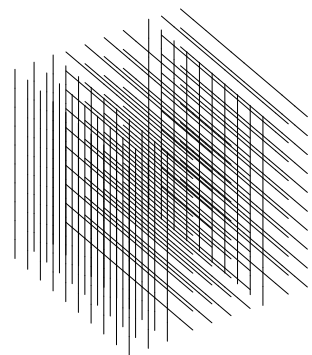
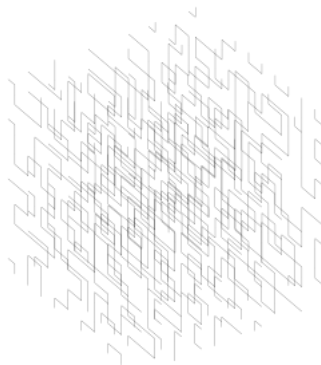
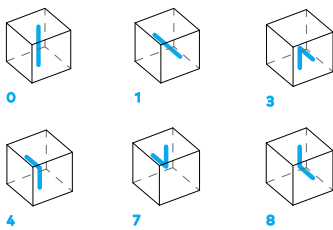
6 modules
World size 10x10x10 slots
Number of explicit connections 5
Number of rules



2 modules
World size 10x10x10 slots
Number of explicit connections 2
Number of rules



6 modules
World size 10x10x10 slots
Number of explicit connections 12
Number of rules



No empty slots

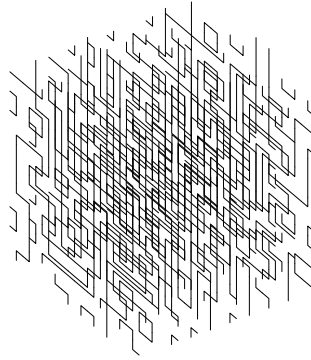
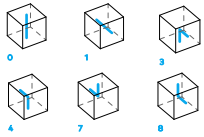
With empty slots

TEST 06_A1

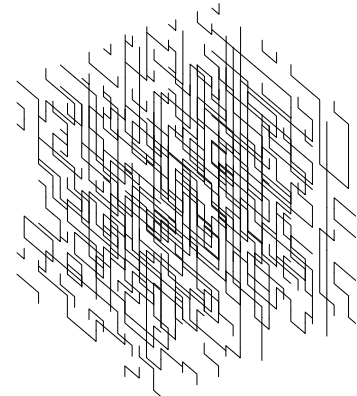
1 megamodule

5 single modules

6 components with 2 connectors



No empty modules



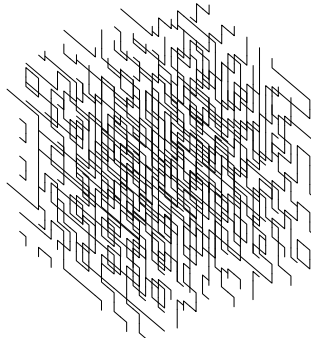
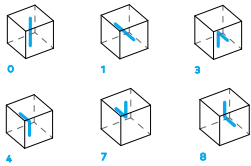
With empty modules

TEST 06_A2

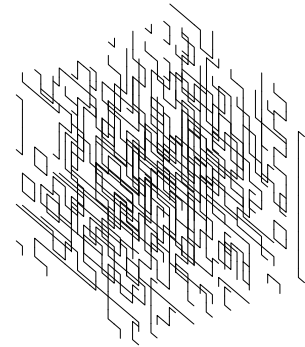
1 megamodule

5 single modules

6 components with 2 connectors



No empty modules



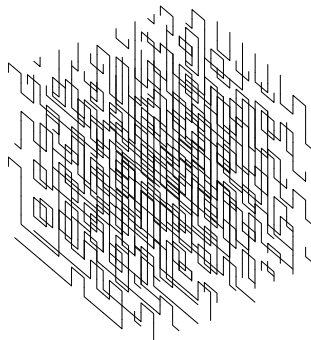
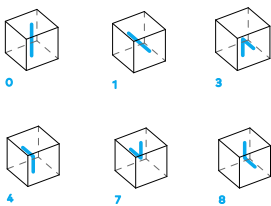
With empty modules

TEST 06_A3

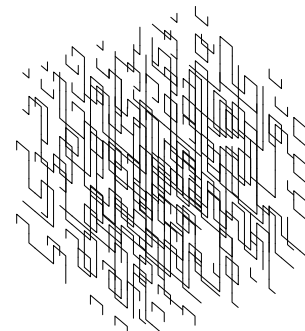
1 megamodule

4 single modules

5 components with 2 connectors

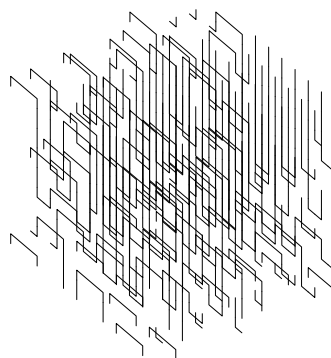
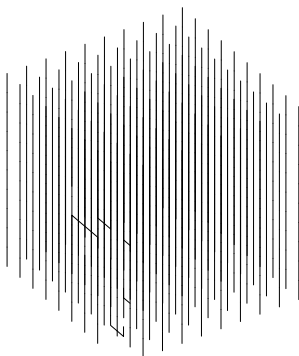
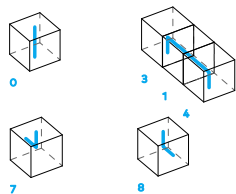


No empty modules

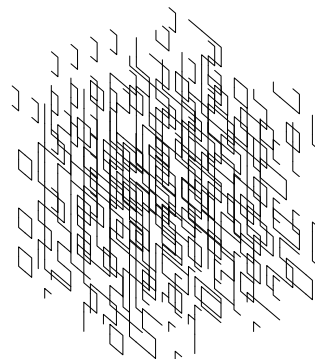
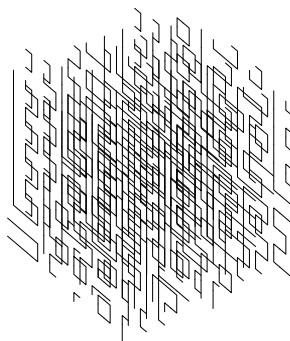
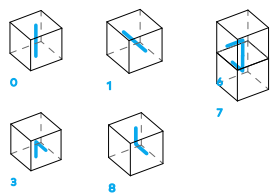


With empty modules

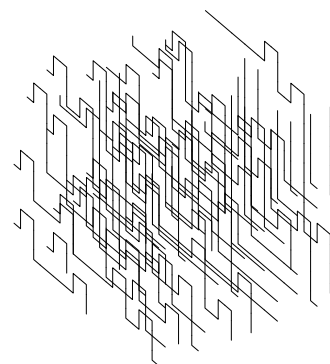
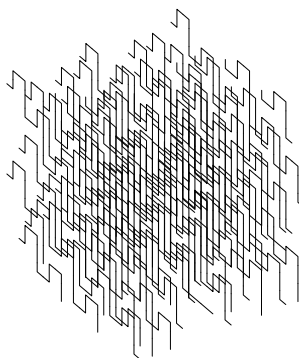
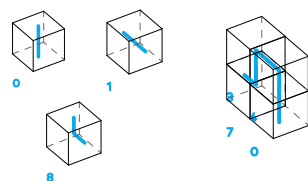
1 megamodule
3 single modules
4 components with 2 connectors



1 megamodule
4 single modules
5 components with 2 connectors

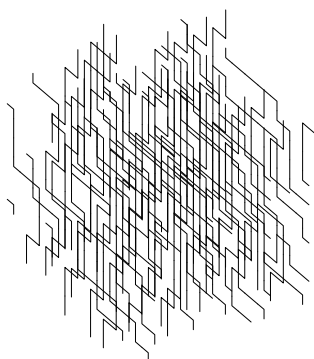
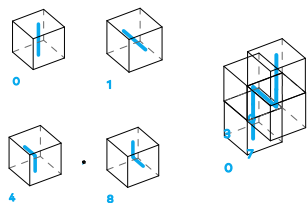


1 megamodule
3 single modules
4 components with 2 connectors



TEST 06_A7

1 megamodule
3 single modules
4 components with 2 connectors



7 modules

With empty slots

Various seeds

World size 5x5x5 slots

Number of explicit connections

Number of rules



7



8



9



10



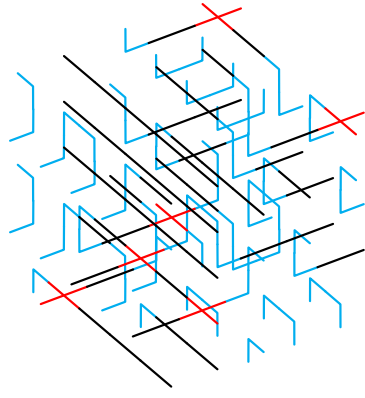
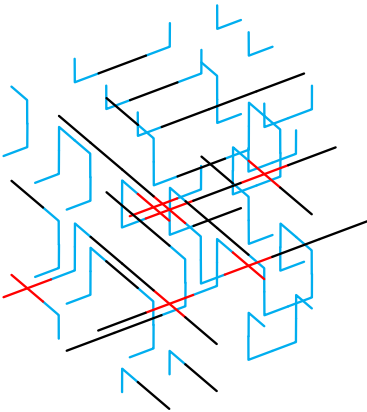
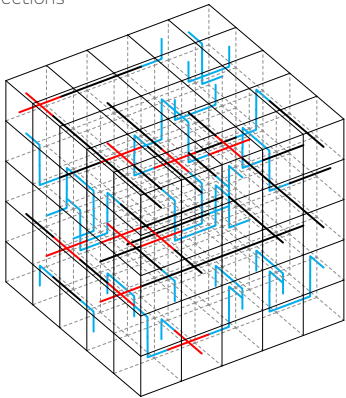
27



1



2



8 modules

With empty slots

Various seeds

World size 5x5x5 slots

Number of explicit connections

Number of rules



7



8



9



10



27



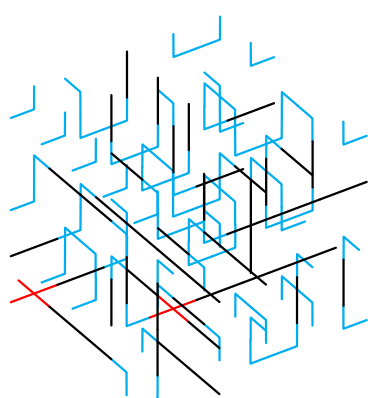
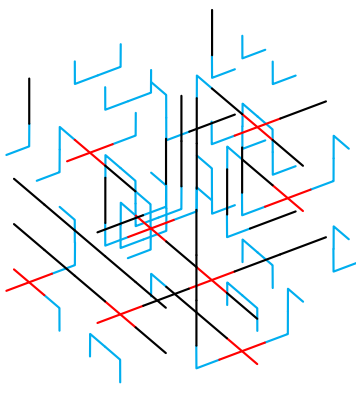
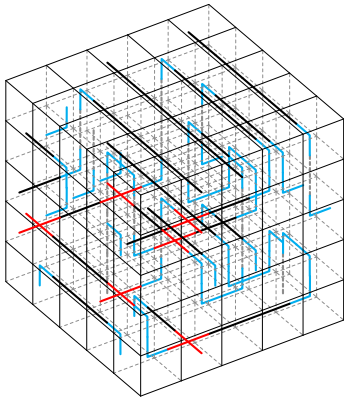
0



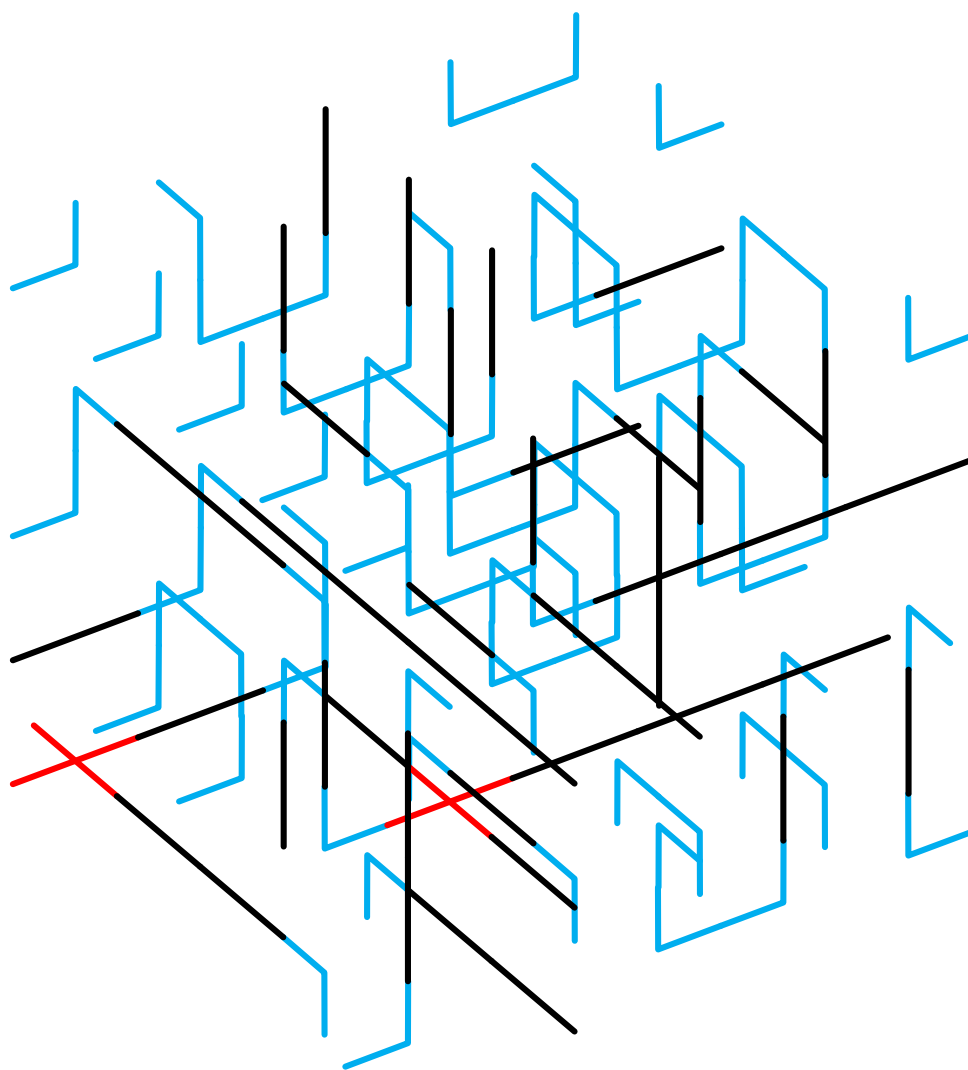
1



2



Basic singlemodules



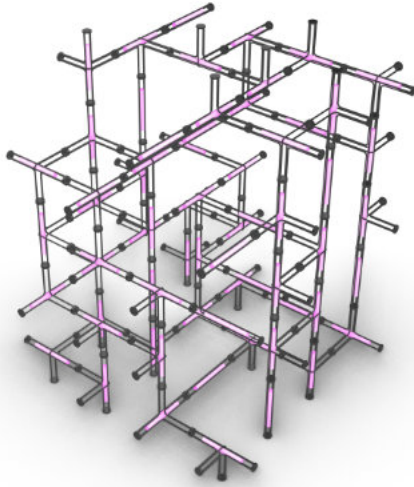


03 APPLICATIONS

WFC in real-life projects

APPLICATIONS

Testing and improving our skills and sensitivity with our toolset on real-world applications in various scales – from the scale of product design to urban planning

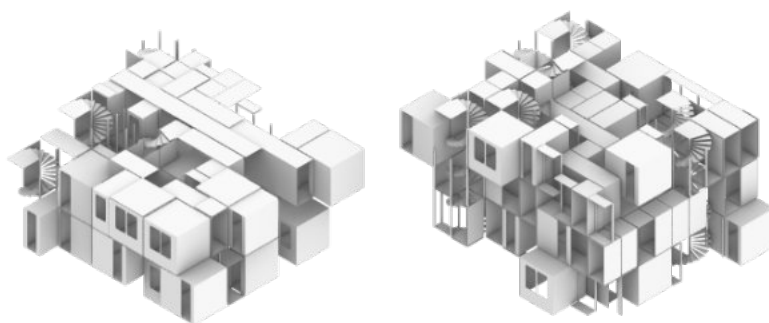


S

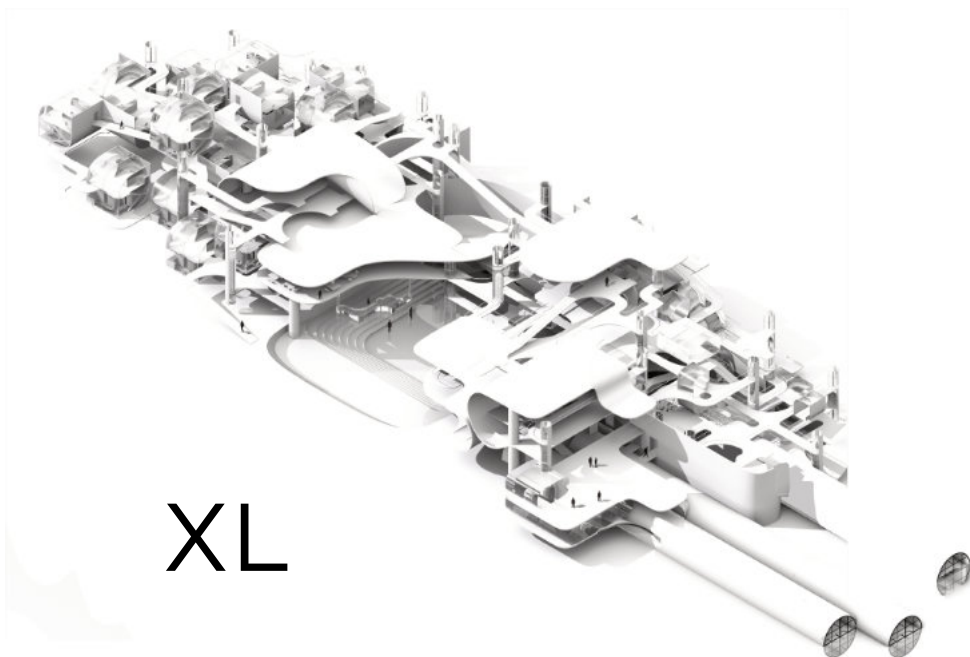


M

L



XL



NEONS

Modular light sculpture

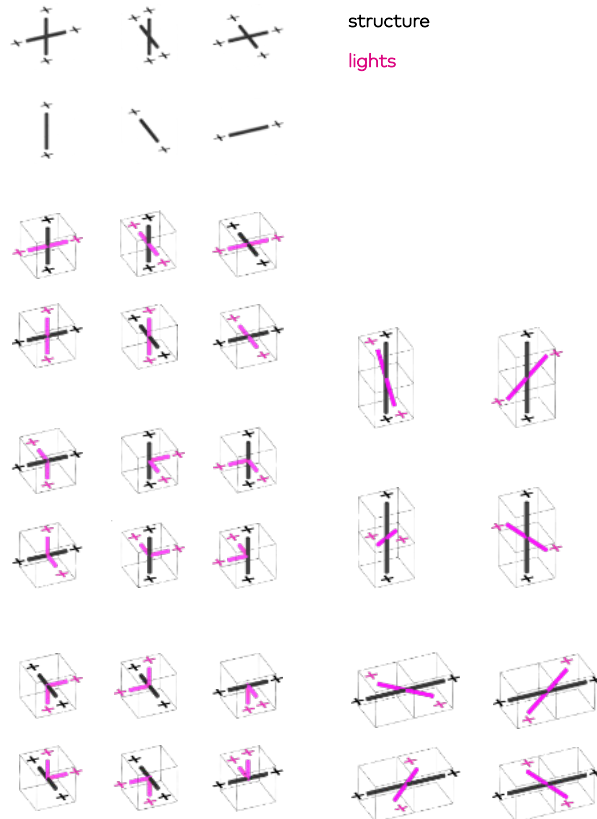
Fixing an emergent aggregate of neon tube loops into a an intricate, yet fairly regular frame.

In early stages of the development we tested the WFC solver on simple directional stick assets.

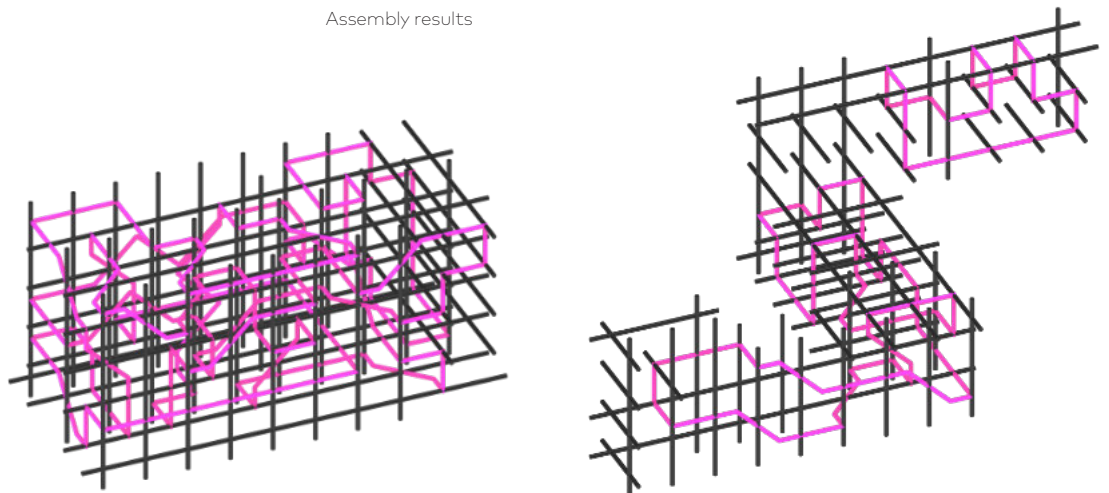
The Neons application is its direct offspring: a complete set of directional and cross-pipes were used for the structural frame together with several special modules with neon tubes. The ruleset required the neons to form closed loops, each of a different color.

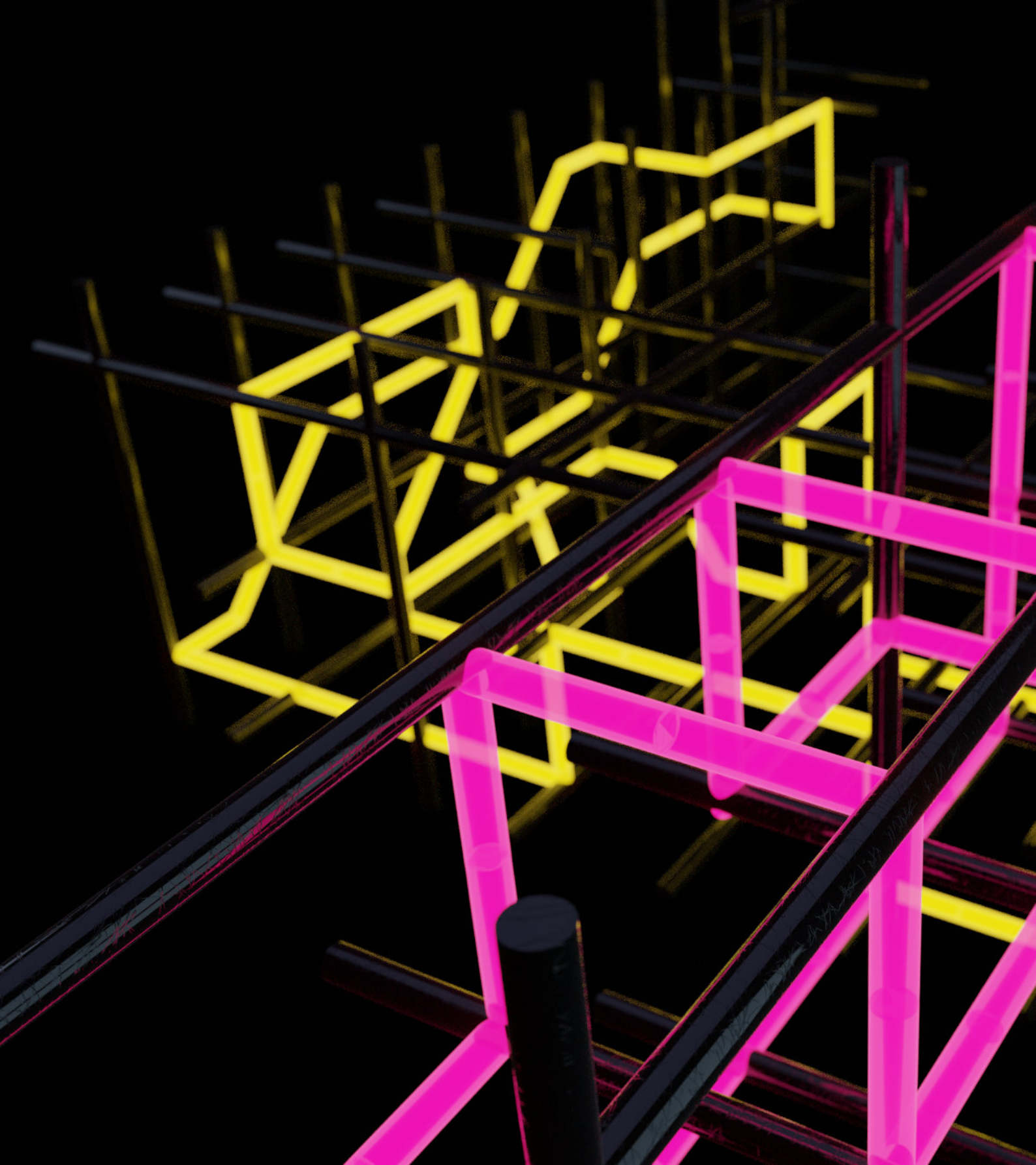
The design of a spectacular lamp with a complicated wired net of led-strings, which would be still connected through framing system providing successful materialization of the lamp.

Modules and rules



Assembly results





PIPES

Sculpture

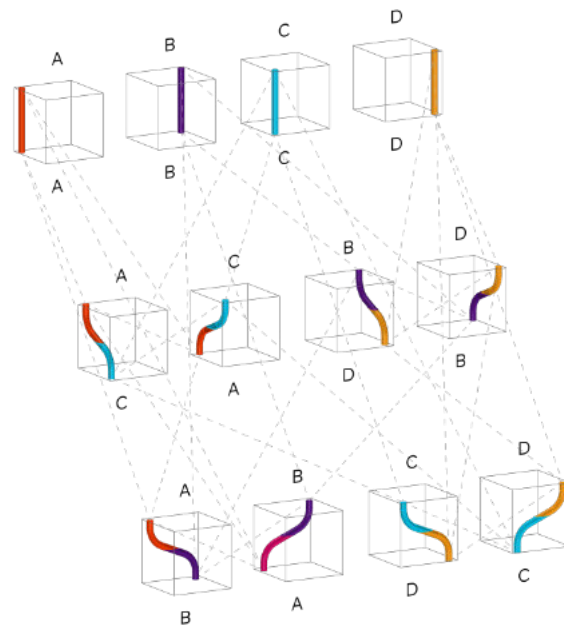
A series of continuous winding pipes should entirely fill the given spatial envelope, each starting at its vertical top and ending at its vertical bottom. The result should be simple to produce yet feature an intricate aesthetics.

Twelve modules, that can be replaced with only two different physical elements, were used. The ruleset allowed for the modules to aggregate into continuous pipes exclusively in the vertical direction. Specific modules were selected to be at the beginning and at the end of each individual aggregated pipe. The asset was used for testing the WFC solver toolset, as it showed easy-to-evaluate results and featured nontrivial ruleset.

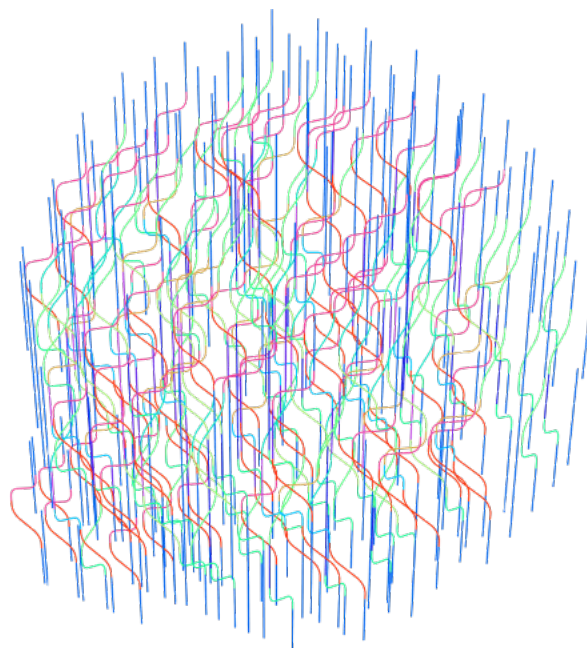
The WFC application was a remake of an older design, which was generated by a growth algorithm. The WFC fills any spatial envelope entirely, allows to define specific modules at selected areas. The asset can be extended so that the pipes intertwine more vigorously and still fill the envelope entirely. The setup and solution time is significantly shorter than in the case of the growth algorithm.

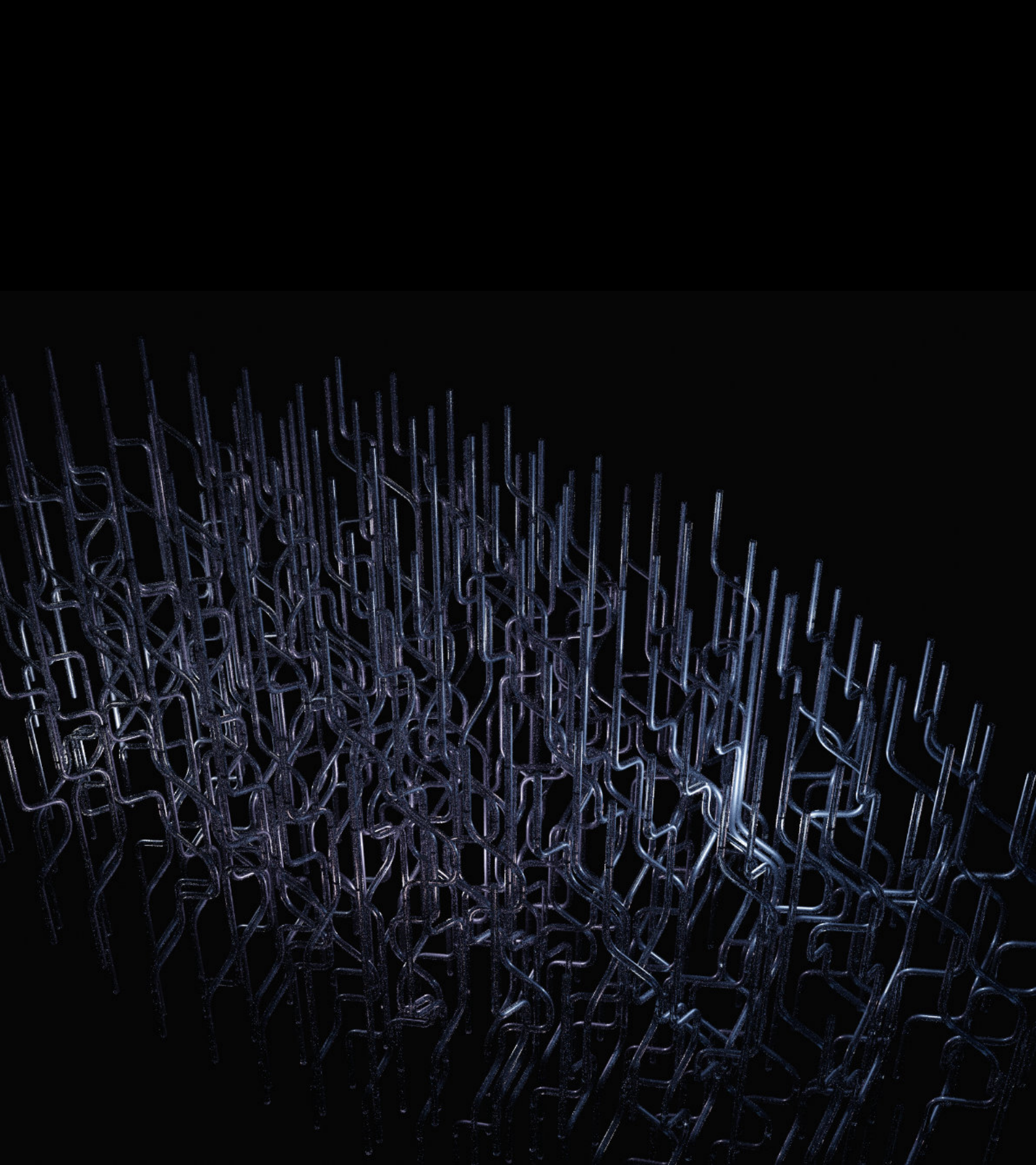
Modules and rules

Connectors: A,B,C,D



Assembly results





LALALAMP

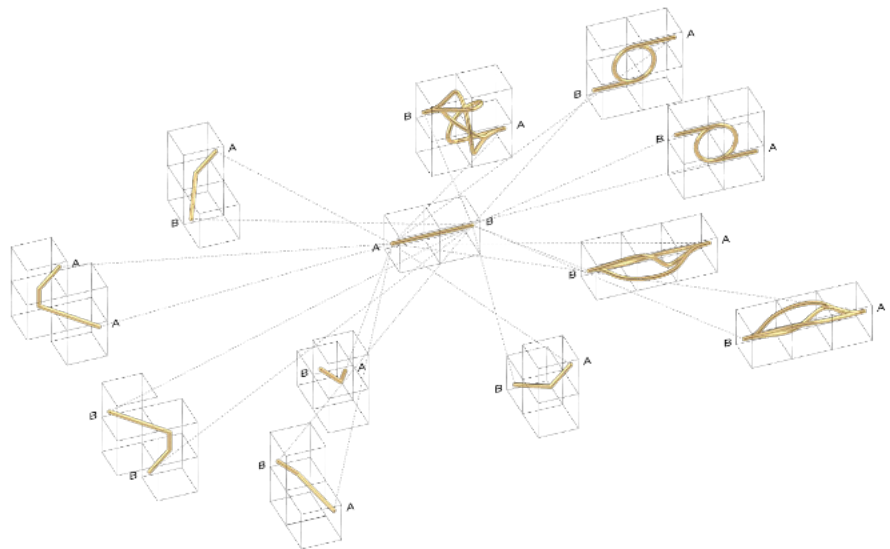
Public space light installation

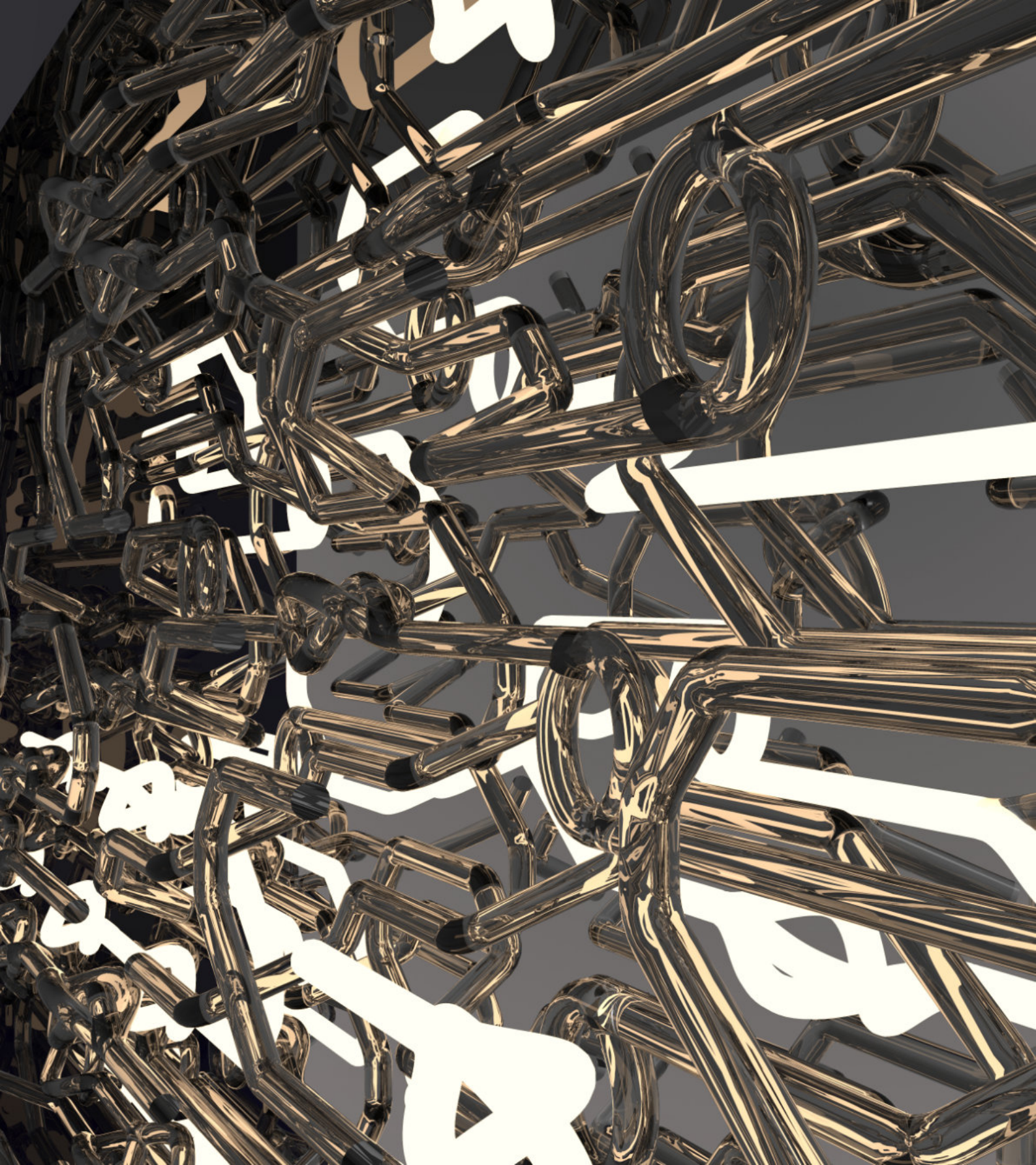
Is it possible to fill space with a branching system of closed pipes, that has a strong dynamic spatial orientation?

A simple set of modules follows carefully defined rules, that ensure the result is horizontal rather than omnidirectional, yet it creates closed pipe loops that intertwine also in the third direction.

It would be entirely impossible to achieve such density of pipes with growth algorithms or manually. The requirement of generating a closed looped system of pipes was easy to achieve with WFC.

Modules and connection rules





PLAYGROUND MAZE

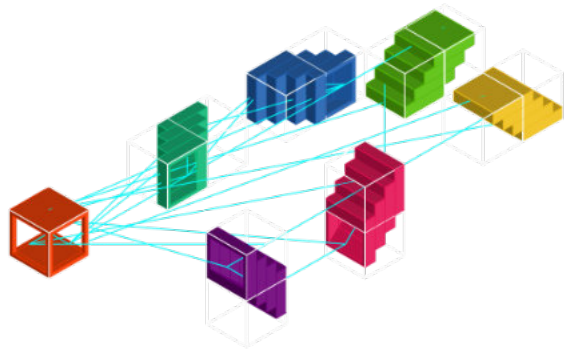
School project

This project challenges the flexibility of discrete assemblies we can generate with WFC. The aim was to create a diverse network of tunnels and stairs to become a playground for children.

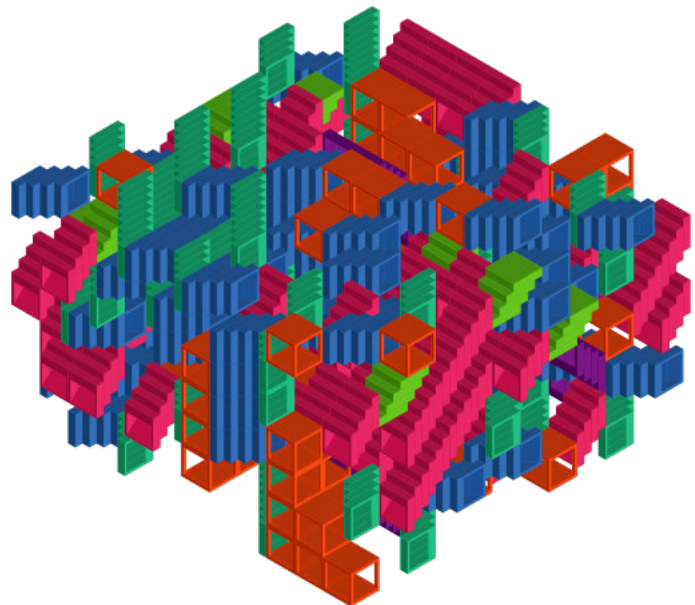
A single basic "communication" element was set with have many directional aggregations, with being full (stairs) or dull (tunnel). Later, a "house" element was added to the ruleset.

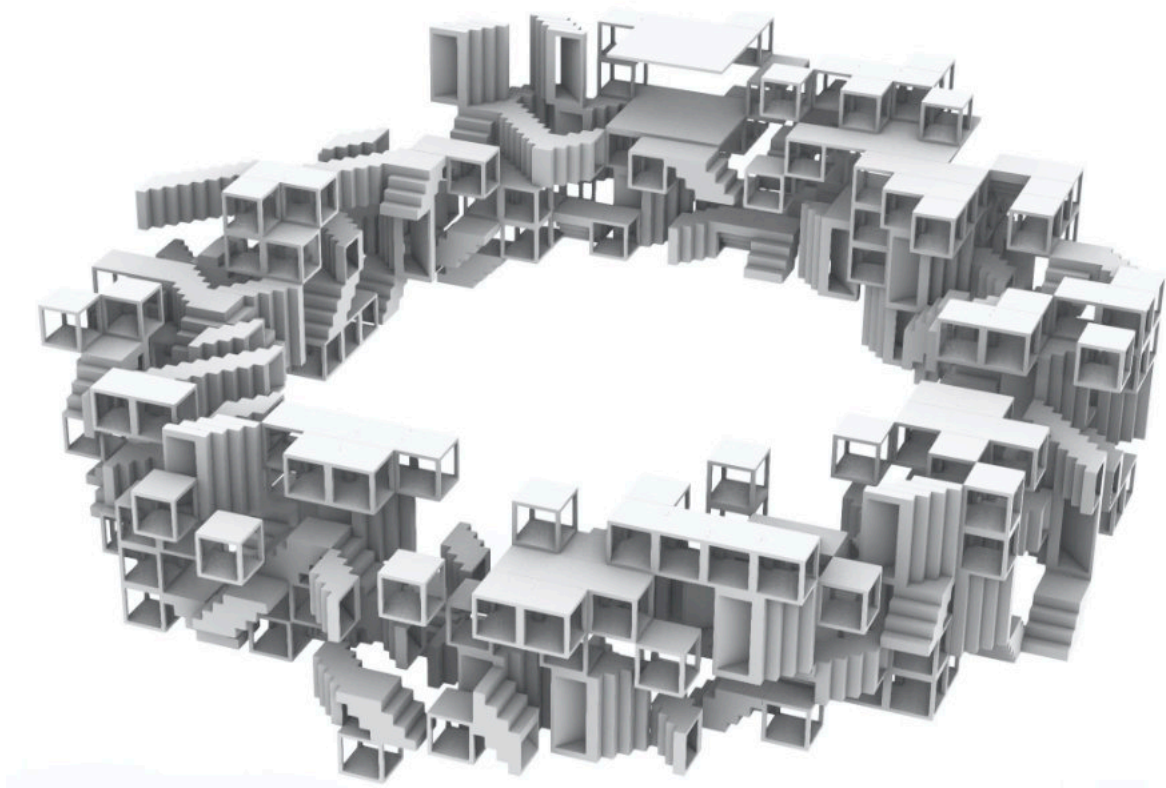
All pieces assembled together precisely fill an arbitrary space with tunnels and staircases fluently interconnected into a playful maze.

Modules and connection rules

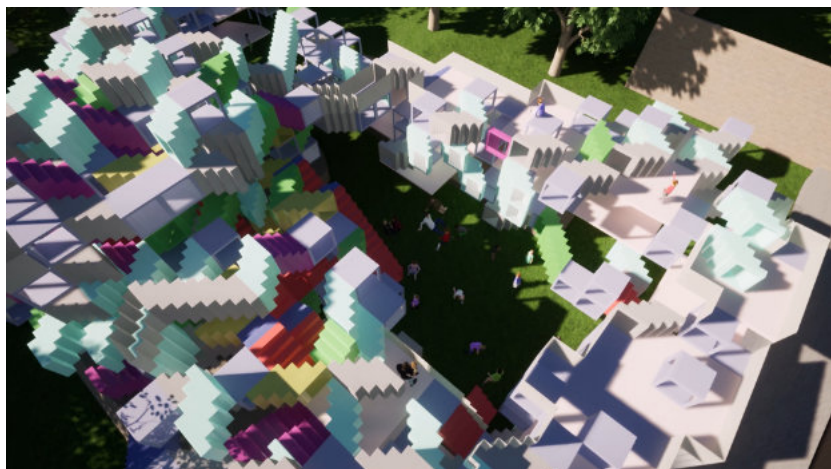


Assembly results









HOUSING

A series of architectonic exercises

The housing industry is battling overpopulation in cities, and pressure of global housing crisis.

The prediction of near future seems to shift from 3D-printing back to prefabrication and discrete design.

How to automatize housing design process? The objective of this project is to create functional, generic housing system that is flexible, adaptable on given spatial and social conditions.

The housing industry is battling overpopulation in cities, and pressure of global housing crisis.

Such system should be ready for prefabrication and repetition of mass production, while maintaining the uniqueness and peculiarity of every resulting design.

Our building system is defined by finite number of repeating basic elements, and the relationship between them.

In this case modules are representing horizontal and vertical communications, rooms and facades and the relationship are taken from basic common building typologies.

The resulting system is generating site (space) -specific designs with limitless number of variations.

Depending on the given space, the system can combine rules and modules into valid single houses, apartment blocks or into superstructures.

The strong side of discrete design and our WFC-generated housing is in its plurality and the ability to create connected unique spaces using a finite number of parts while preserving the given relationships between them.

If the rules and relationships are correct, every result is valid.



Source modules

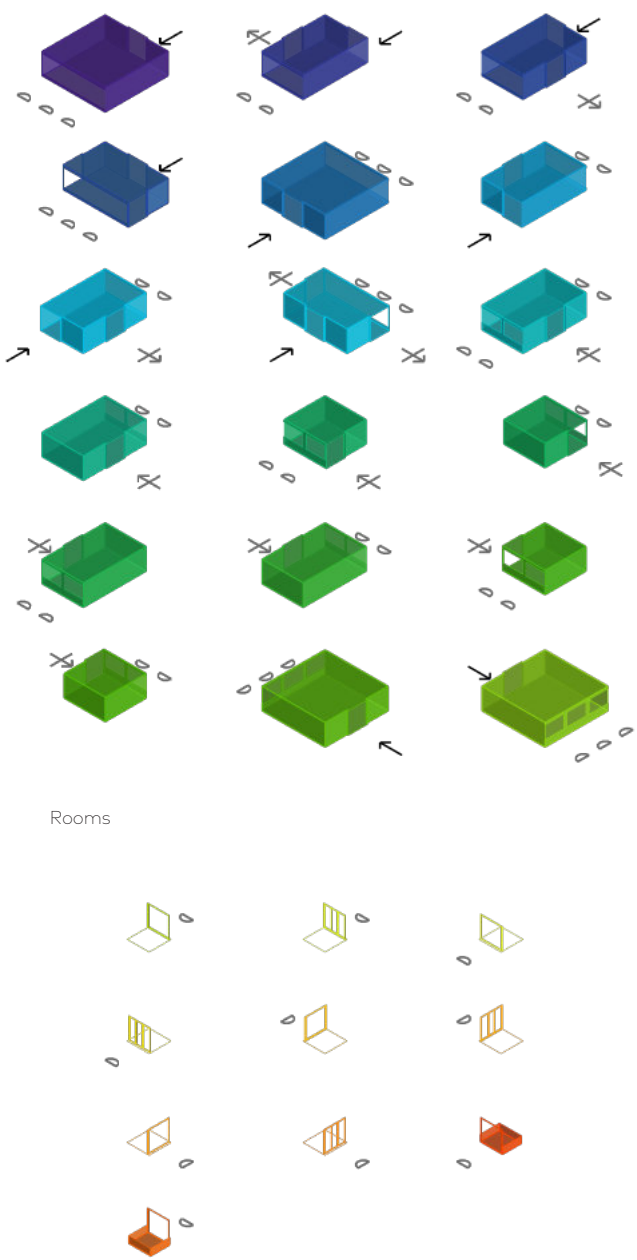
43 mega-modules
consisting of 175 single modules

Main hallway modules

Connecting hallway modules

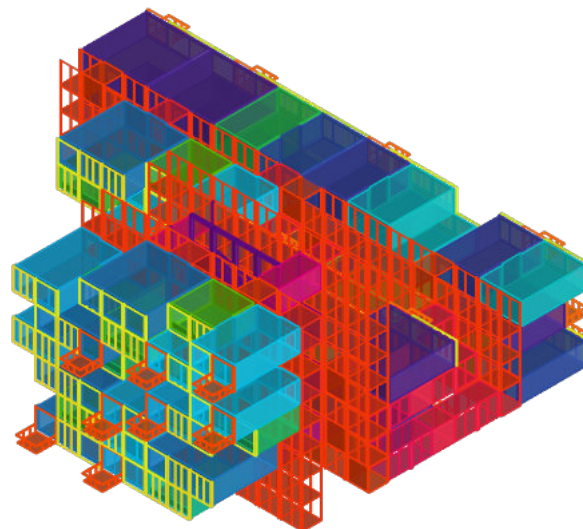
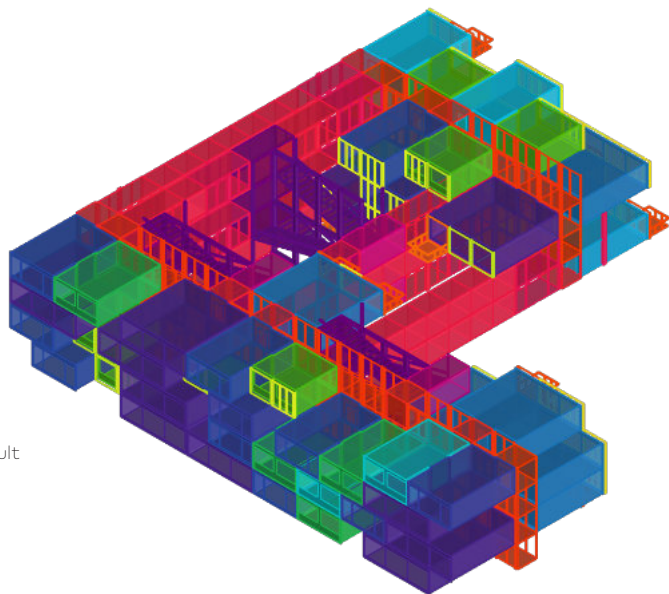
Staircase landings modules

Stairs modules

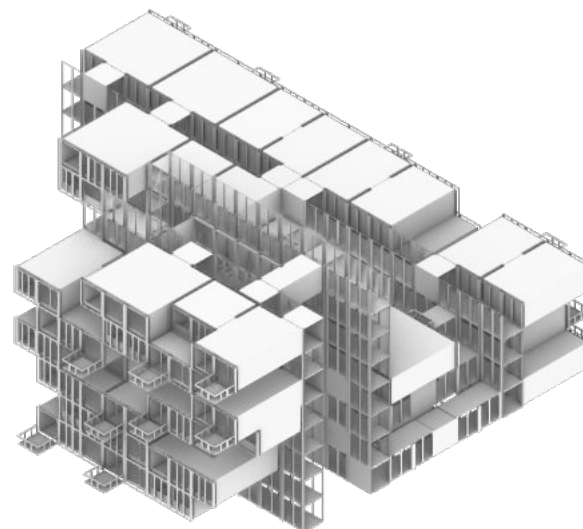
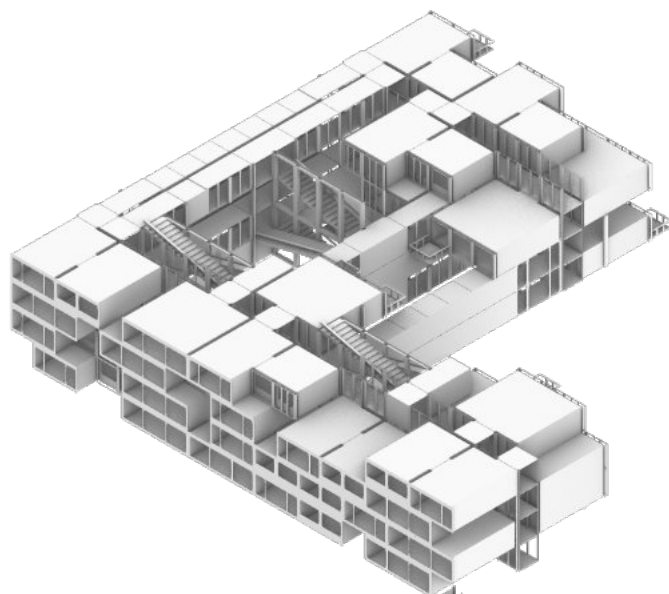


Rooms

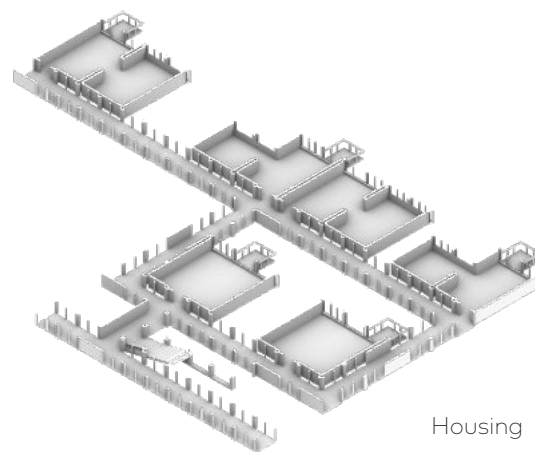
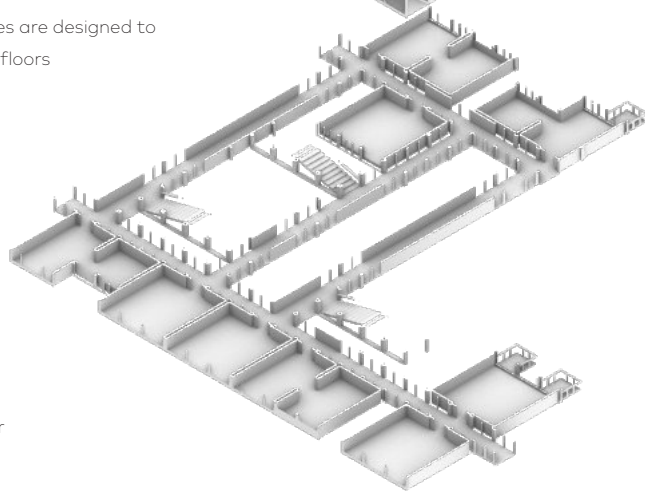
Facade modules



Result



Modules are designed to join by floors



1st floor

The process of finding and developing a functional system originated in our established set of abstract set of modules and its translation into architectural language.

The basic set of modules and rules were defined by geometries set in cubic grid with 6 possible directions of aggregation. In the world of architecture that means 4 cardinal directions plus two vertical directions – the above and below. In this regular cubic world, our tests had proven that too many modules with too complex, undefinable rules would have to repeat to achieve the variety of elements essential to construct valid architectonic space. Our solution was the creation of various clusters of modules with constant internal rules.

This optimization created the definition of "megamodules". In this way we achieved the ability to define and gain control over different sized spaces that are required to create a building. Differently sized and shaped "megamodules" were then given external connection rules and qualities of various architectonic elements.

In this case we defined a set of "megamodules" which doubled as set of elements required to create a horizontal and vertical communication within the building. These we have

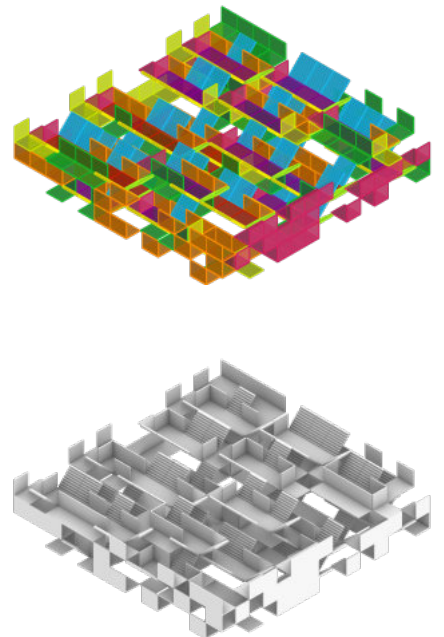
called "hallway megamodules" and "stairs megamodules" and got various types of connectors, some with the ability to connect to each other and create a continual path within a building and some with the ability to connect the rooms.

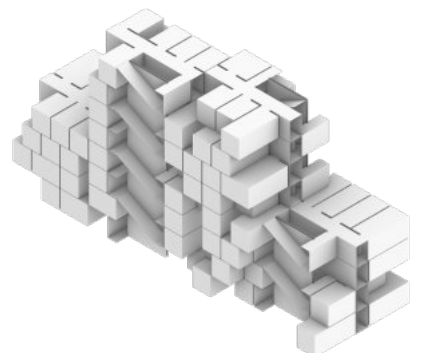
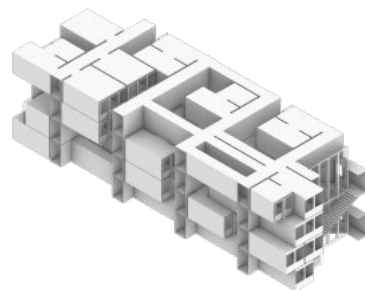
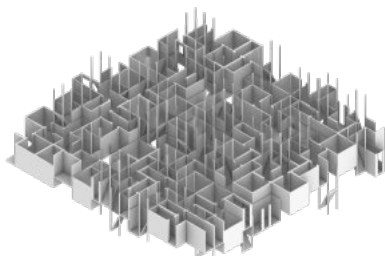
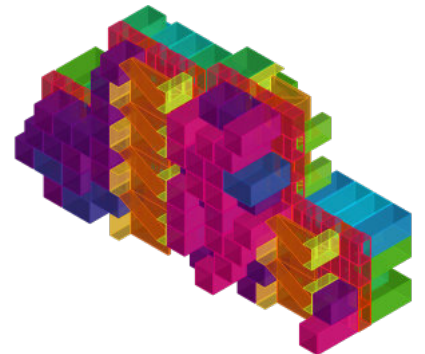
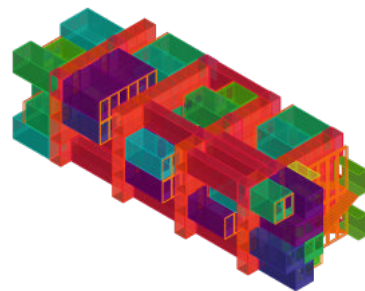
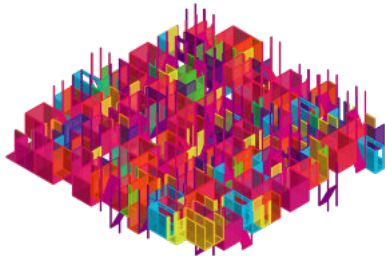
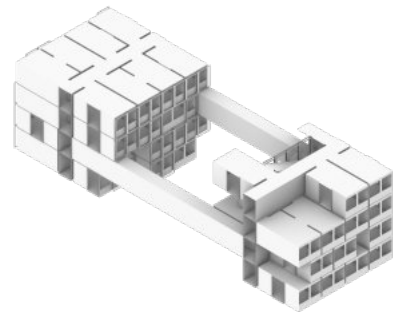
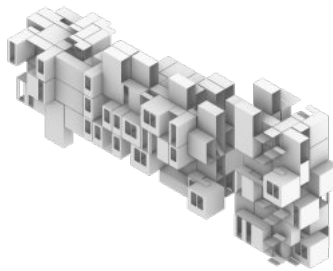
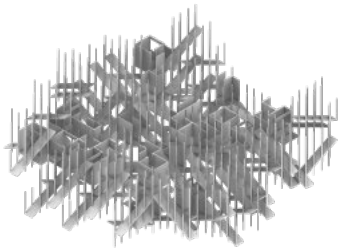
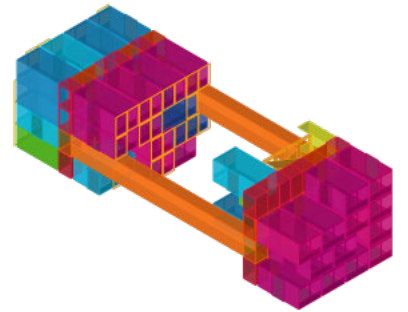
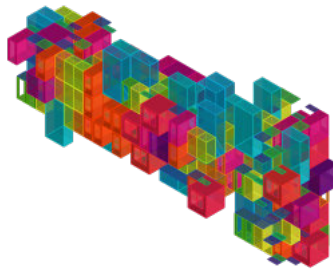
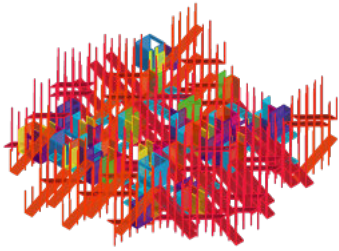
The set that follow these communication megamodules, the "room megamodules" and "façade megamodules" have another types of rules and connections that helped us control and define the interior and exterior of our system and the relationships within them.

By this variety of architectonically defined "megamodules" we have proven the relevancy and applicability of our wfc algorithm in the design process of scale of architecture.

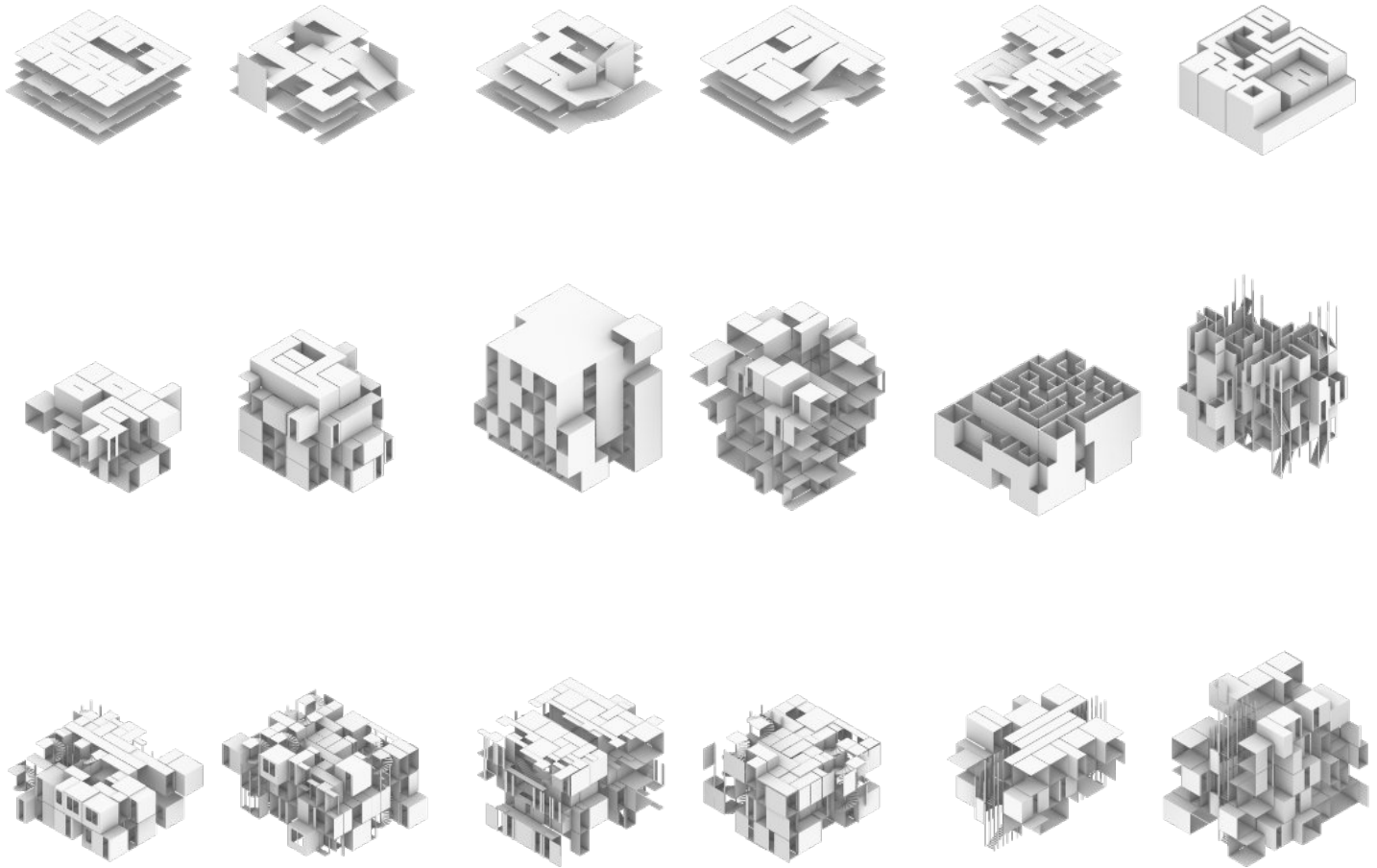


A basic example of a floor plans of a result generated by WFC

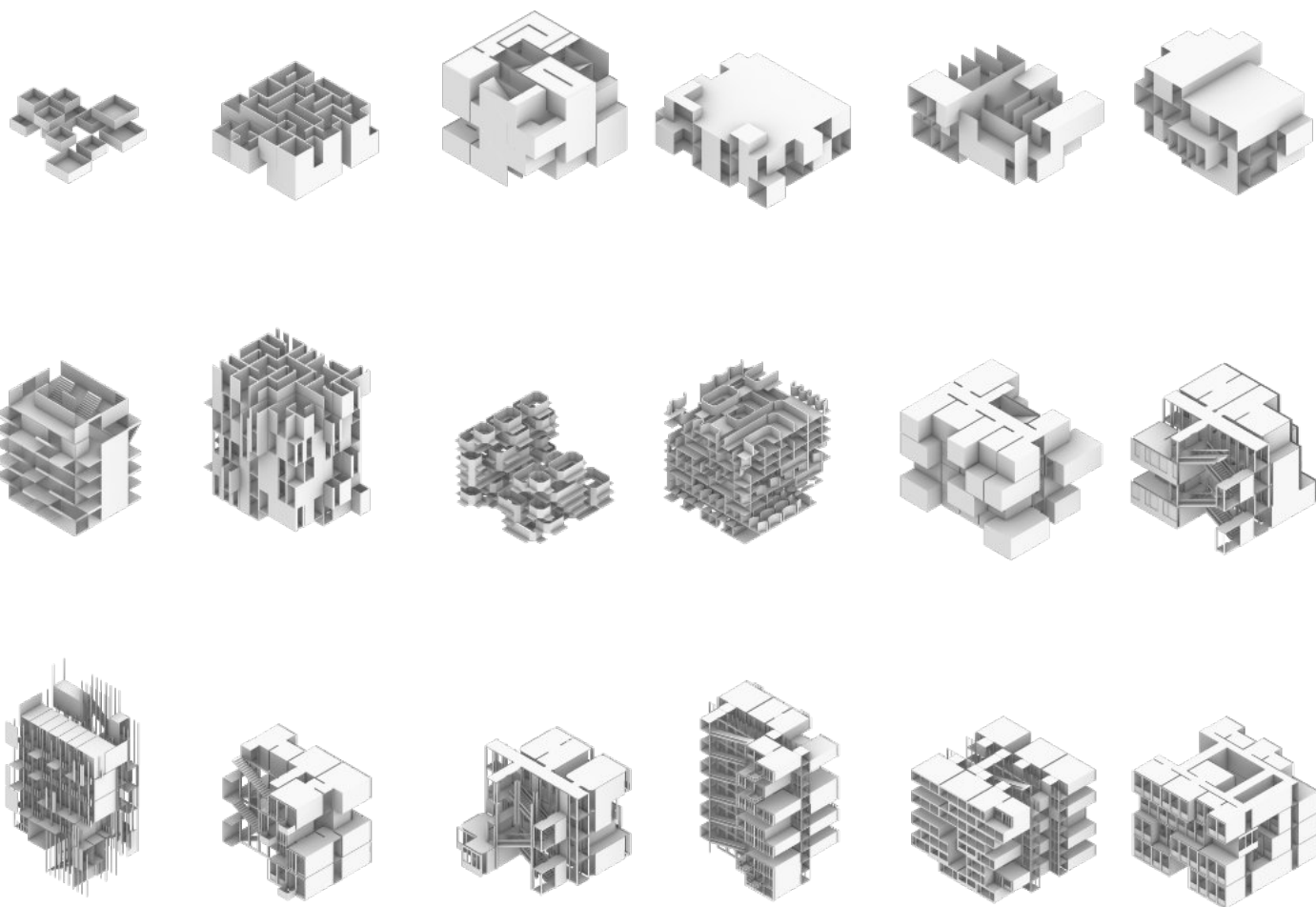




Variations



Different seeds and space definitions



HYPERLOOP DESERT CAMPUS

Conceptual proposal for competition

Project by Miriam Löscher and Eva Kvaššayová

How to fill specific spatial envelopes with a given set of interconnected elements?

Cross-scale design of the campus / small-scale masterplanning consisting of various program types (in large scale) and separated networks of infrastructure assembled with various types of facilities and amenities (in a small scale).

We were able to choose from a multitude of program organizations and then populate them with smaller inner components. By setting strict rules we could precisely control the density of horizontal and vertical circulation, empty and filled spaces.

And thus also the permeability of the structure and movement of people. The algorithm helped us find solutions, which wouldn't be possible to find by traditional ways of design. Separated networks of infrastructure assembled with various types of facilities and amenities (in a small scale).

Programme scheme

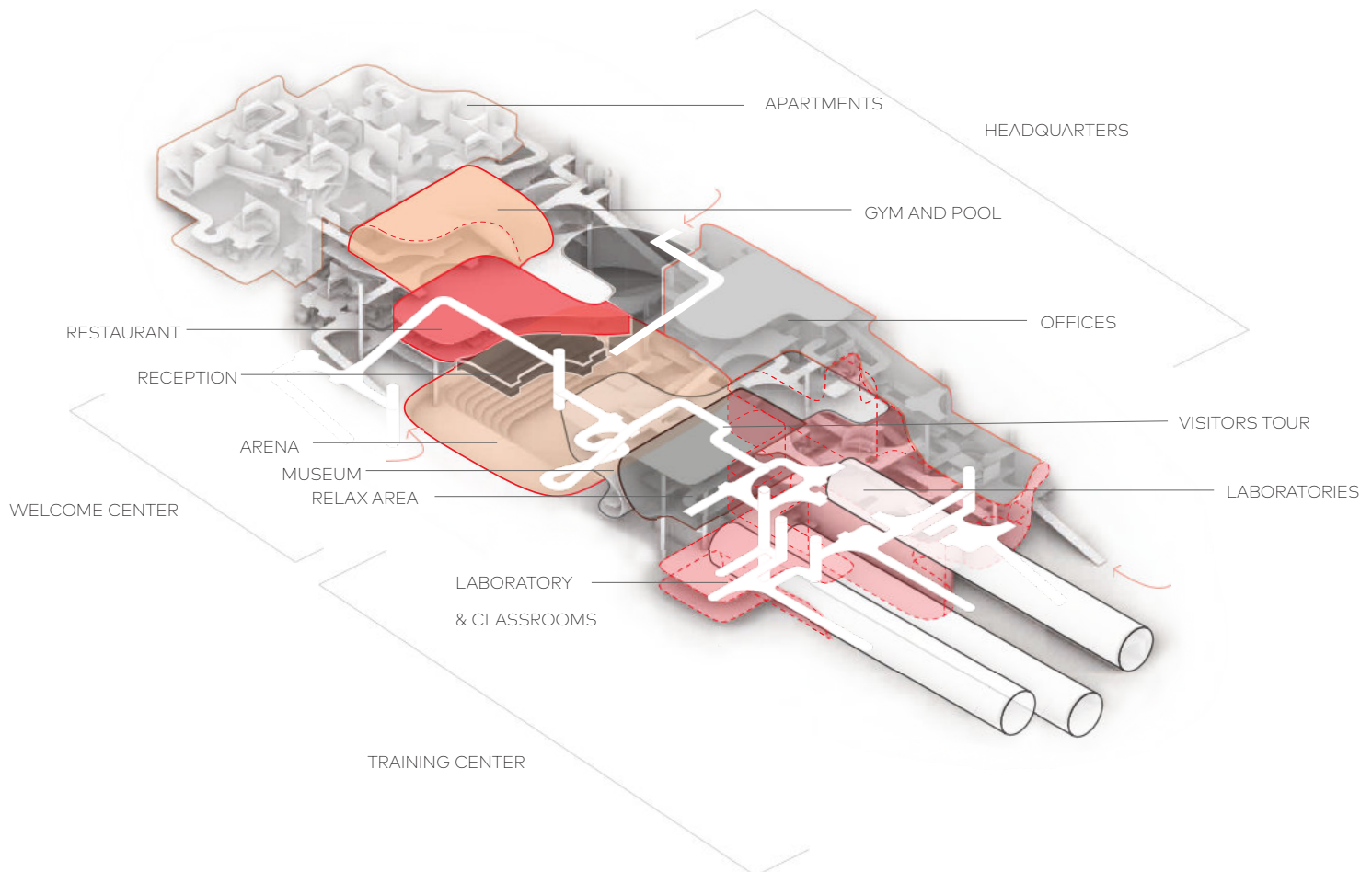
The hyperloop campus is going to be first of its species, basically a flagship. therefore it needs to breathe with future and mediate the feeling from the novelty of hyperloop itself to those, who come to visit this center. one of the most up-to-date digital design methods is using the wave function collapse - an algorithm,

which creates assemblies from given elements according to their defined adjacencies.

in first step we set voxel volumes based on space requests from brief, designed their logical connections and let the algorithm assemble it for us.

secondly we designed the communica-

tions and cells and let the space of former designed voxels be filled with those. a spectacular net of communications was created, where visitors can enter all accessible areas and enjoy the view of mountains and desert as well as the inner organization.



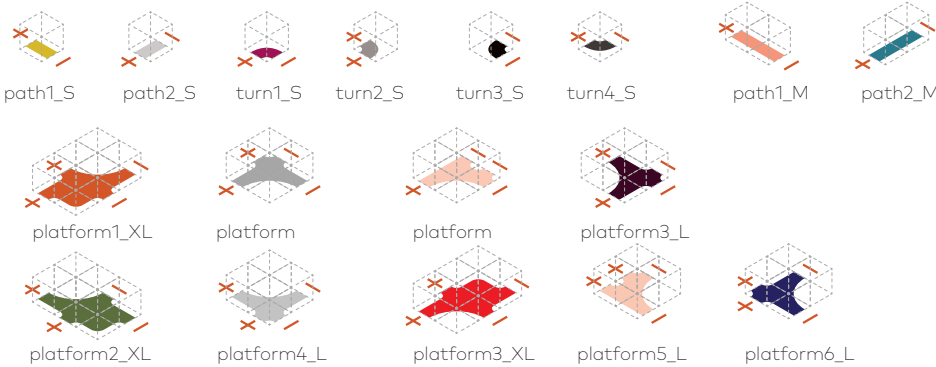
Elements

Digitally developed autofill of the space - wave function collapse algorithm - automatically fills the space with the Modules of specific size (s, m, l, xl; smallest module - 3x3x3m) with given set of the rules - marked as a + or - (side of the voxel with THE PLUS

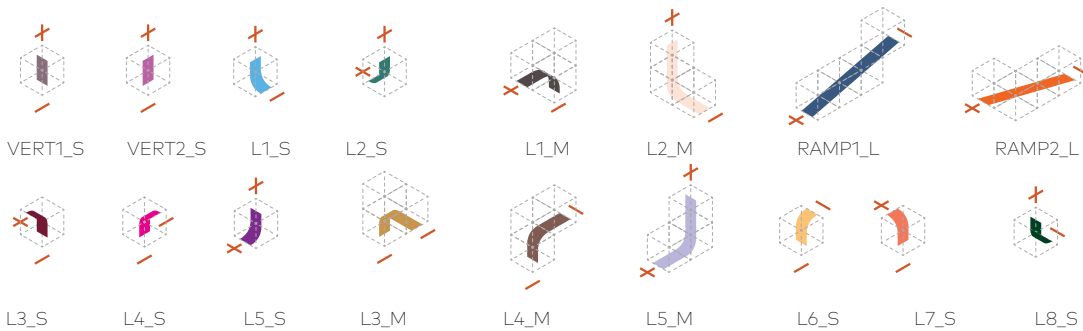
SIGN (+) CAN CONNECT WITH THE side of the another VOXEL WITH THE MINUS (-) in the same axis.) it allows us to precisely control the density of the Communication, empty and filled spaces - therefore, ensure the proper permeability of the structure.

COMMUNICATION

HORIZONTAL MODULES
paths, halls, courtyards

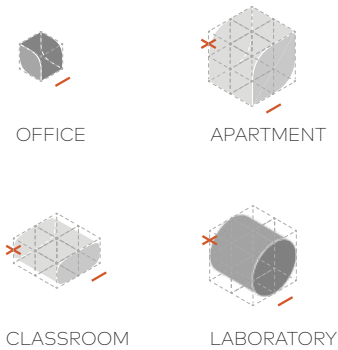


VERTICAL MODULES
Elevators, ramps, stacks



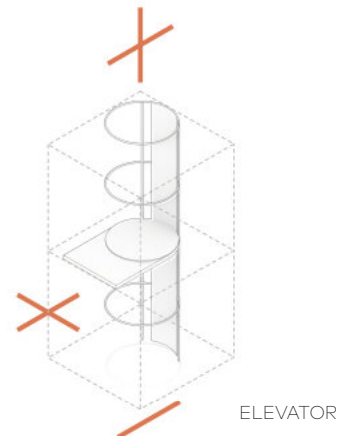
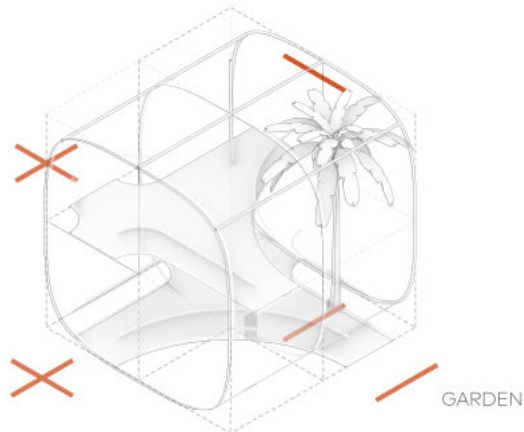
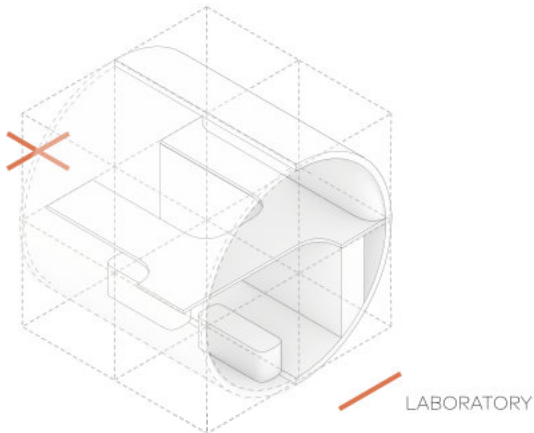
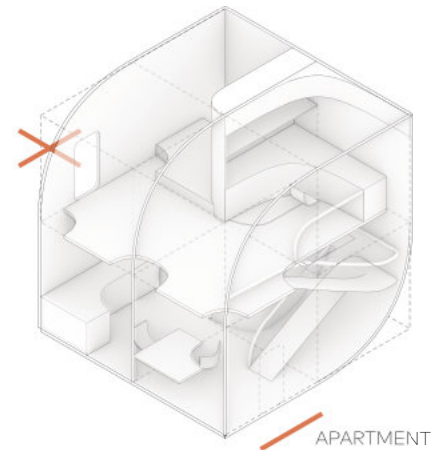
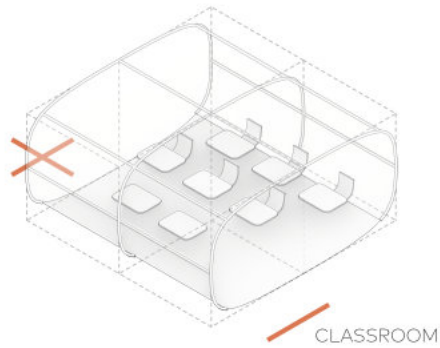
FACILITIES AND AMENITIES

cells with specified functions

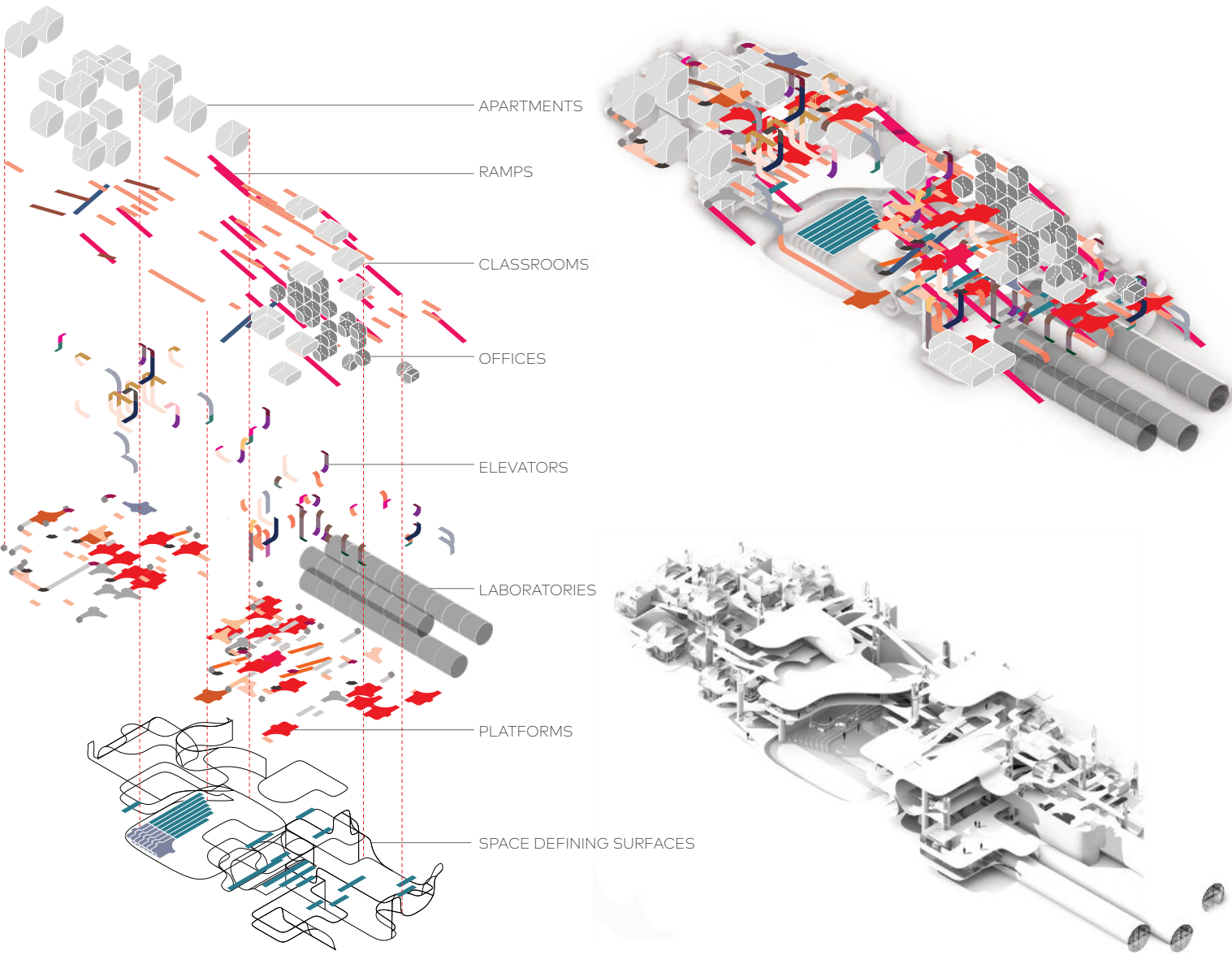


Elements - specific cells

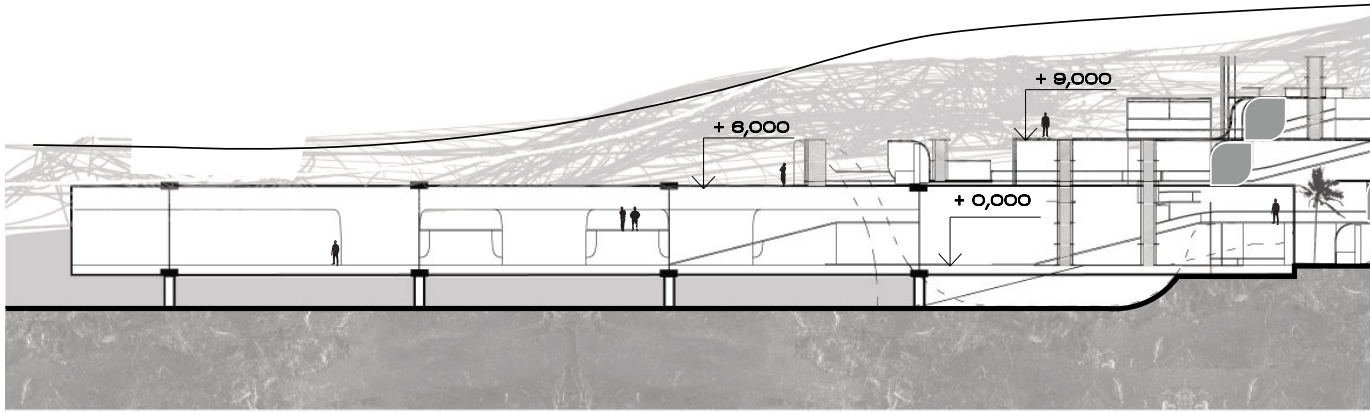
We set six main functional categories of cells, out of which some have modular variations of furnishing and size (basic module: 3x3x3 m). These are evenly distributed among the set space and bound with many paths and platform, so that the daily routine walk to office, relax facilities and back into the apartment becomes an adventurous route through hidden desert paradise.



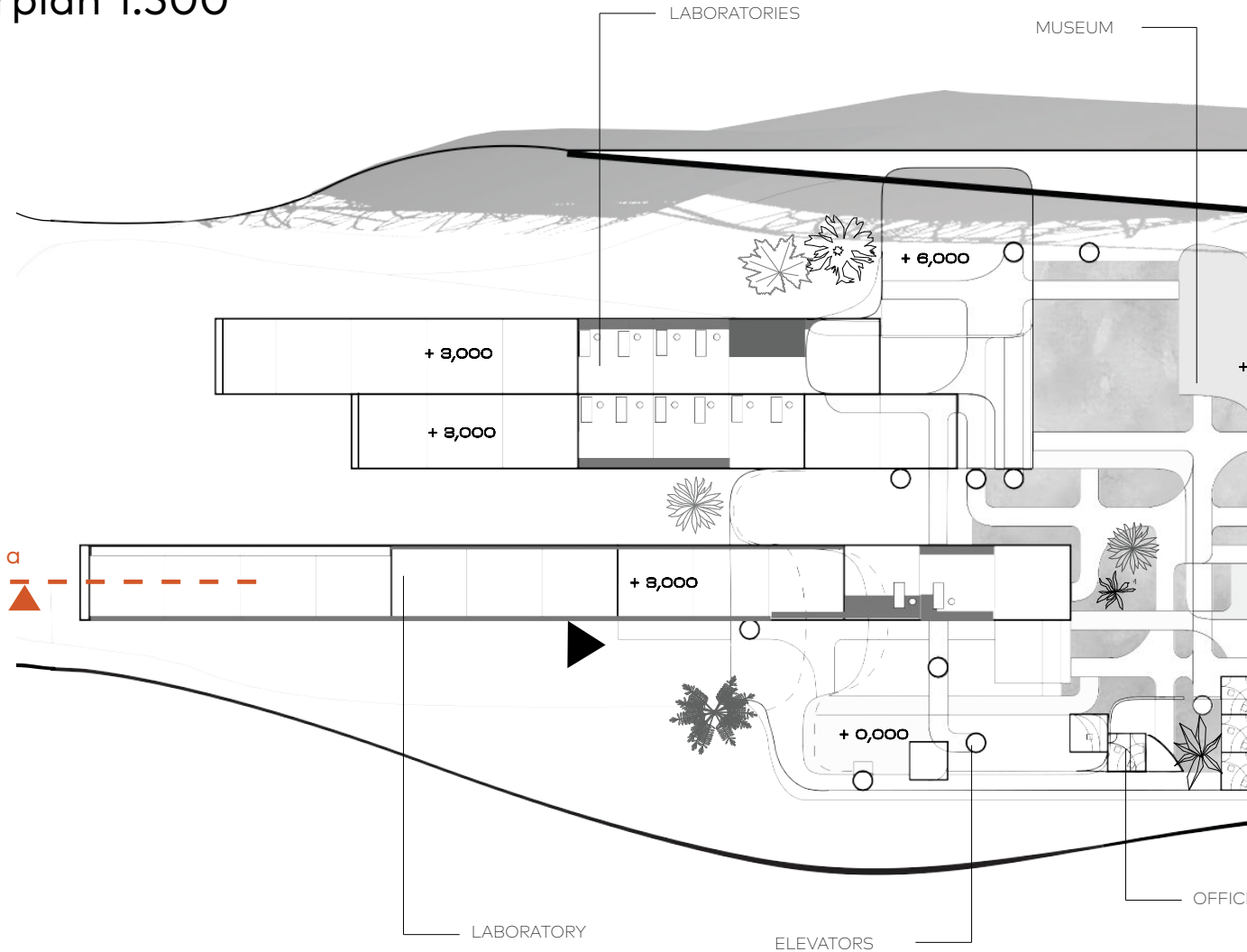
Exploded and assembled isometry with modules

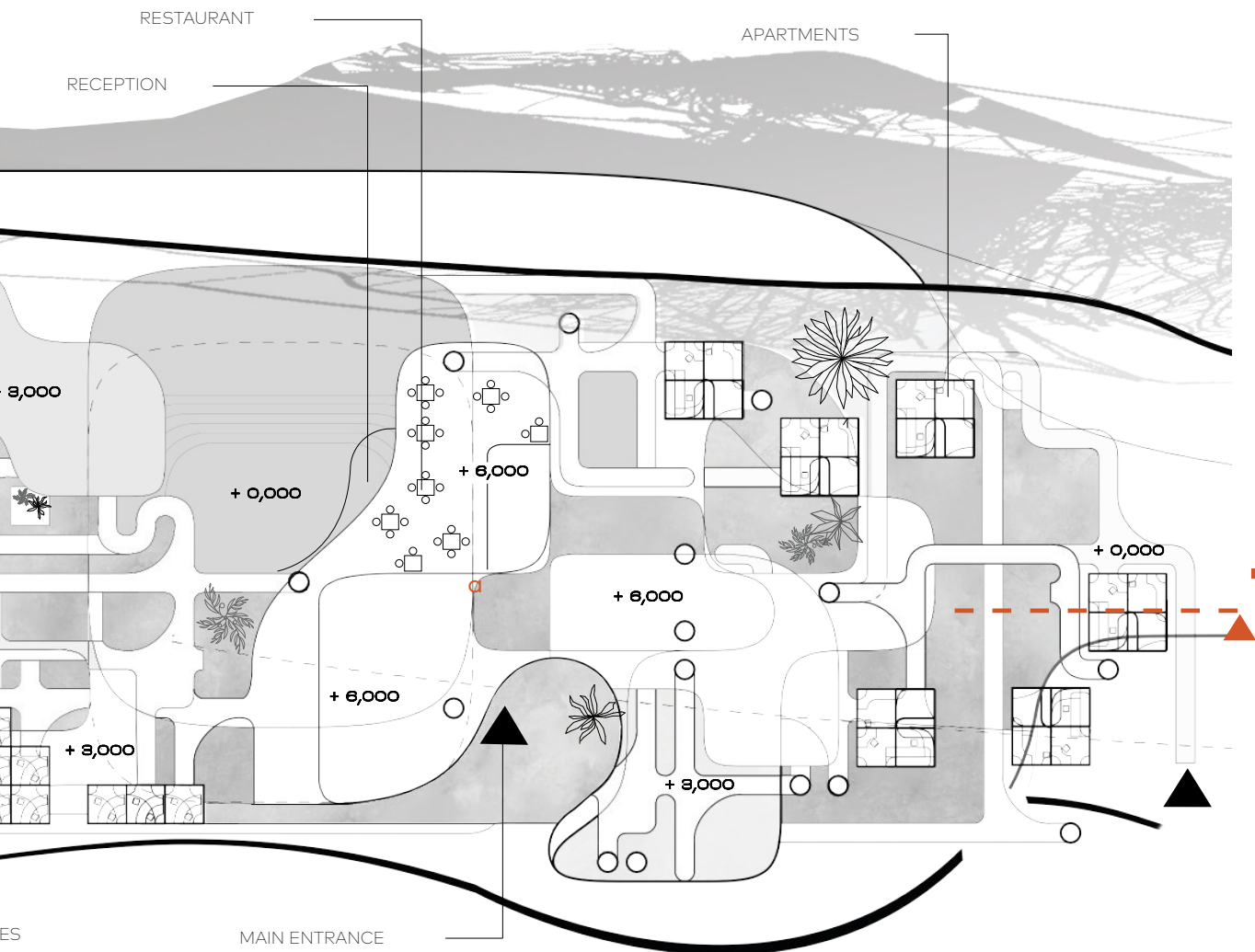
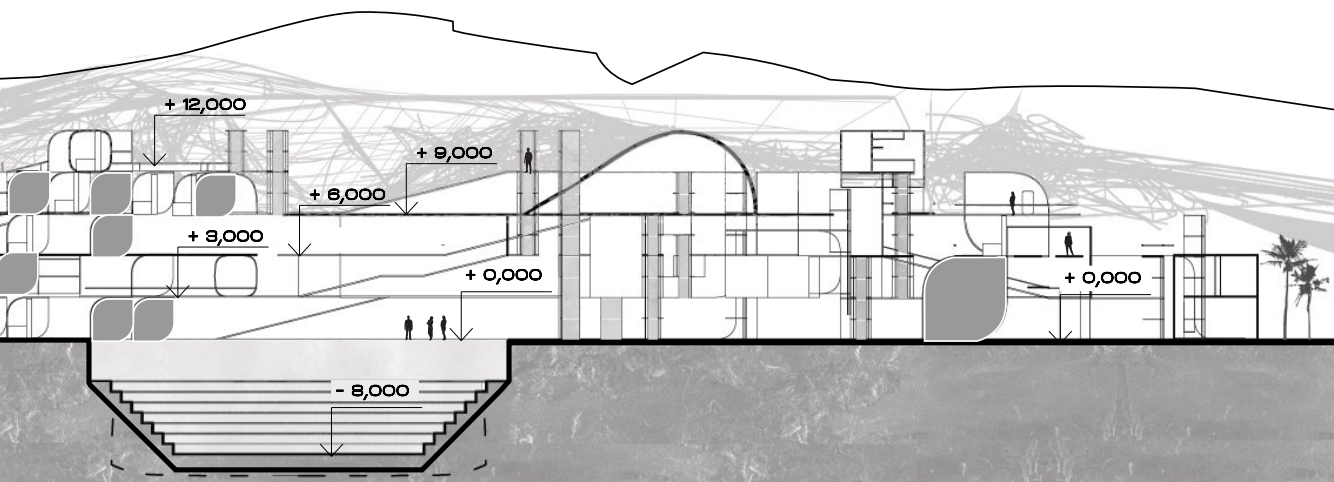


Longitudinal section A-A 1:500



Floorplan 1:500





HYDROGEN ISLAND

Urban strategy for independent future

Diploma project by Jiri Vitek

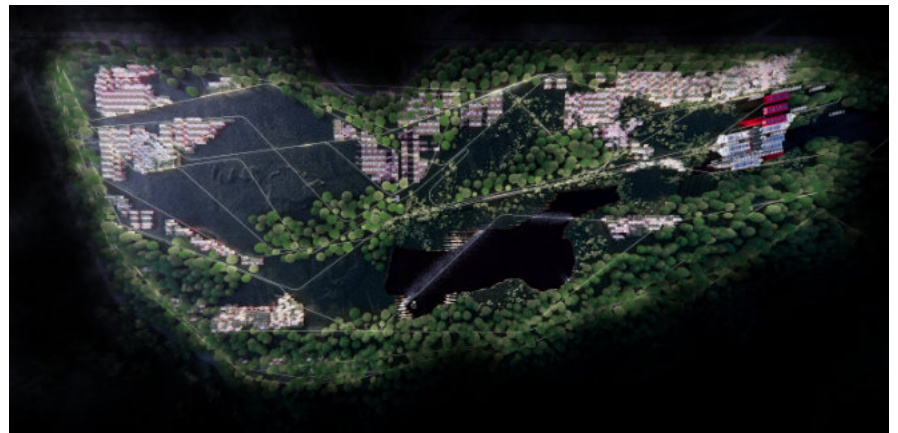
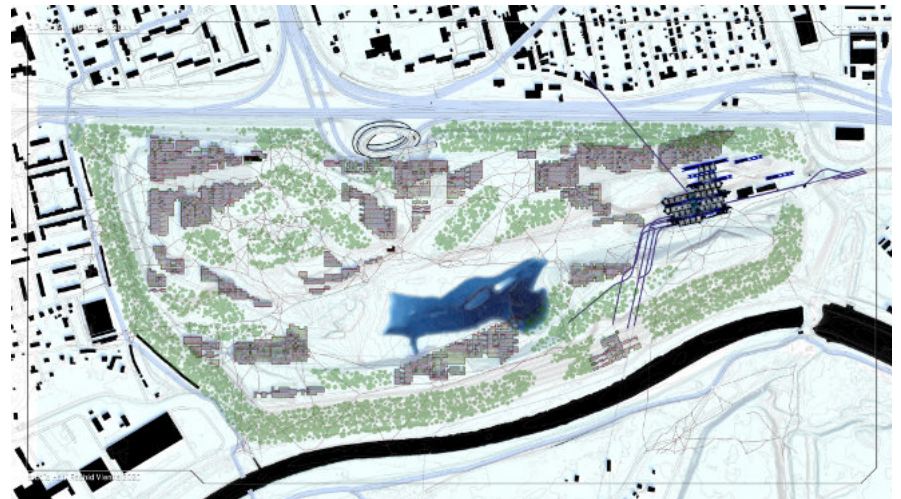
Studio Hani Rashid, Vienna 2020

The project hydrogen island has aim to develop strategies and methodologies for urbanisation of the abandoned and forgotten landscapes using current technologies and focusing on topologies qualities, multiplicities and smooth and stripped spaces.

For bottom up generative systems were used existing pathways, points and datascares interpreting into ground and solar tissue.

Such a defined mass needs to be carefully translated into architectural use, when instead of just slicing we can push it further via topological system such as Wave Function Collapse - Monoceros by Subdigital.

Masterplan

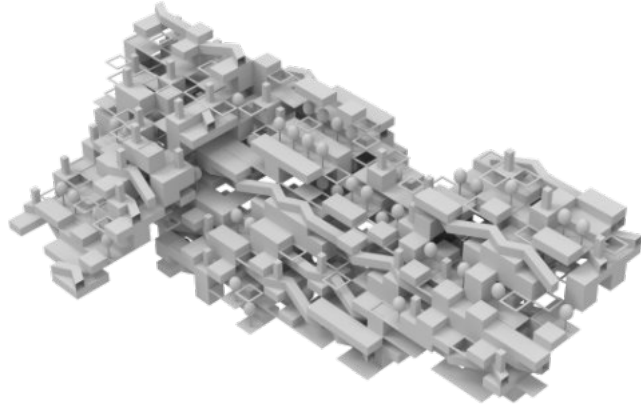




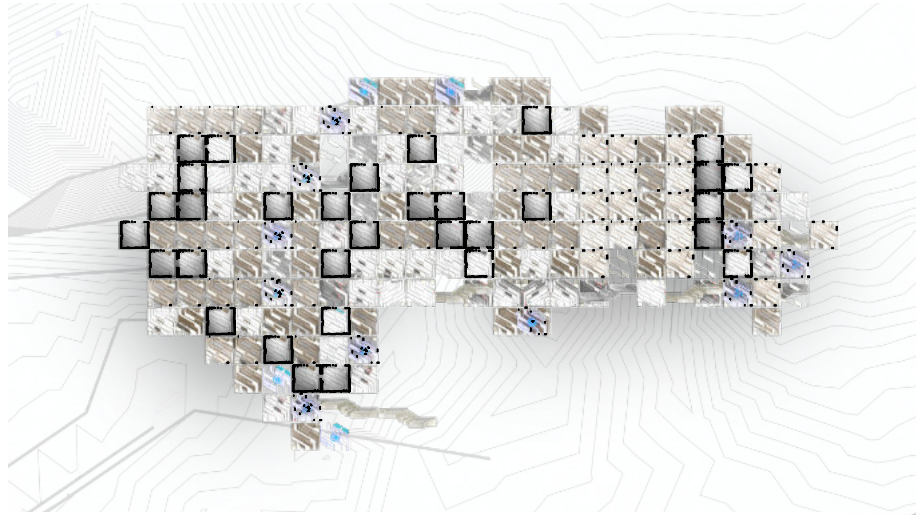
Clusters

Given mass was understood as space for habitats / architectural program _ such as residential units, public space units, gardens, ateliers, work spaces, corridors and very importantly energy production units contains H2 store facility, solar panels and wind turbos as well as recycling grey water facility. In this synergy effect each cluster could be independently existing and be fully sustainable. Right definition of connectors and play with algorithms end up in very nice, differentiated, rich and contextual results.

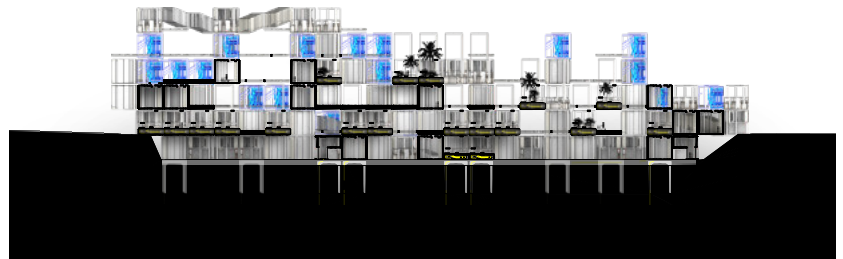
Simplified geometry of one cluster



Plan of production geometry of the cluster



Section of production geometry of the cluster

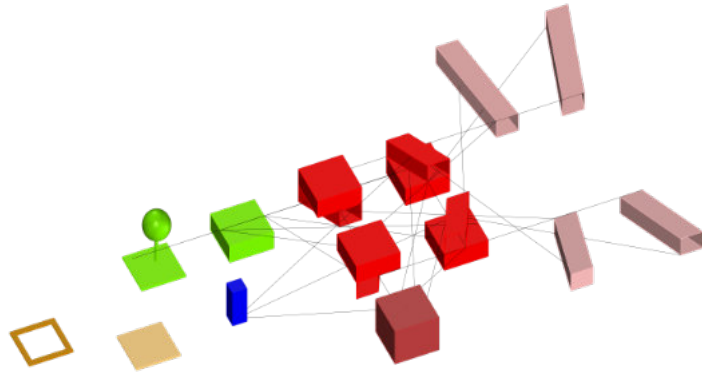




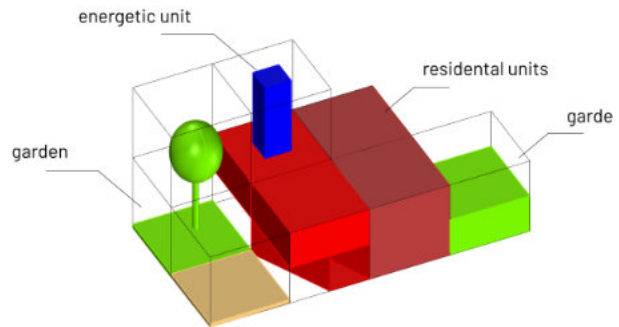
Components and architectural Language

The language is coming from structural understanding of basic frame units and its possibilities of variations. It's quite intelligent and it's able to become an energetic facility or interior spaces in residential units. Relations between parts and whole is considered a field allowing transition between private and public spaces. Such strategies generate a rich and very structural environment.

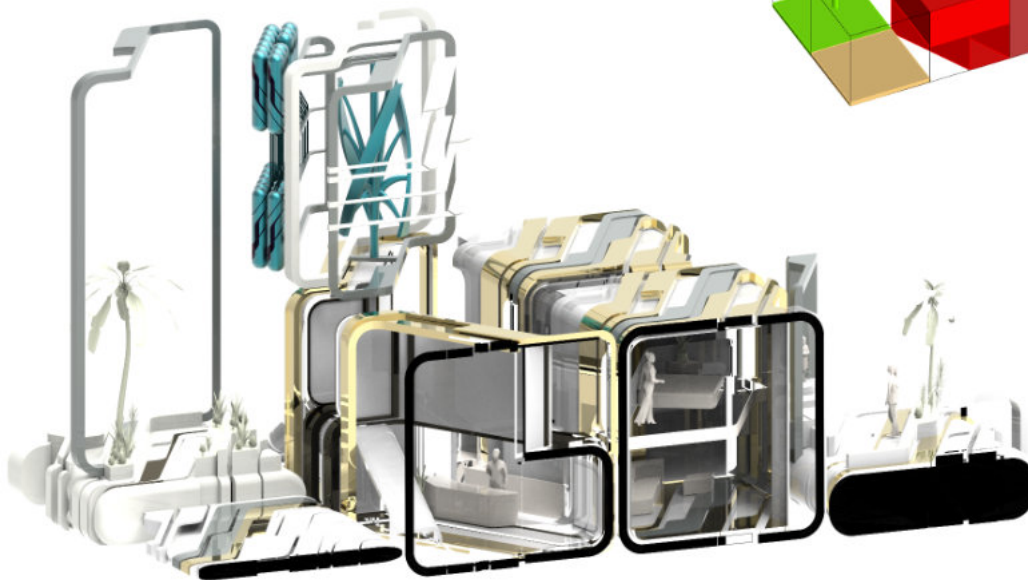
Modules and connection rules



Assembly results - sustainable cluster of units



Simplified geometry



Production geometry

RESIDENTIAL UNIT 01

With hallway



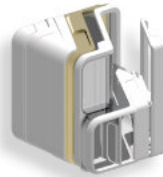
perspective



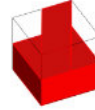
simplified geometry

RESIDENTIAL UNIT 02

With hallway



perspective



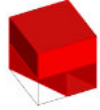
simplified geometry

RESIDENTIAL UNIT 03

With hallway



perspective



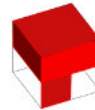
simplified geometry

RESIDENTIAL UNIT 04

With hallway



perspective



simplified geometry

RESIDENTIAL UNIT 05



perspective



simplified geometry

PLATFORM



perspective

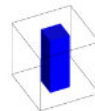


simplified geometry

ENERGETIC UNIT



perspective

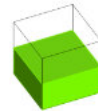


simplified geometry

SMALL GARDEN

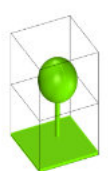


perspective



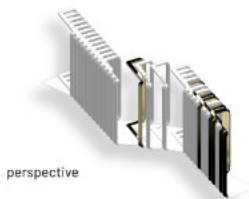
simplified geometry

LARGE GARDEN

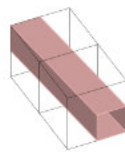


simplified geometry

STAIRCASE



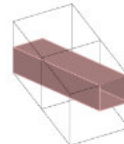
perspective



simplified geometry



perspective



simplified geometry

WFCITY

Urban and housing strategies

Authors: Jakob Dostert, Adam Varga, Petra Sochůrková

Studio Architecture III, UMPRUM 2020/2021

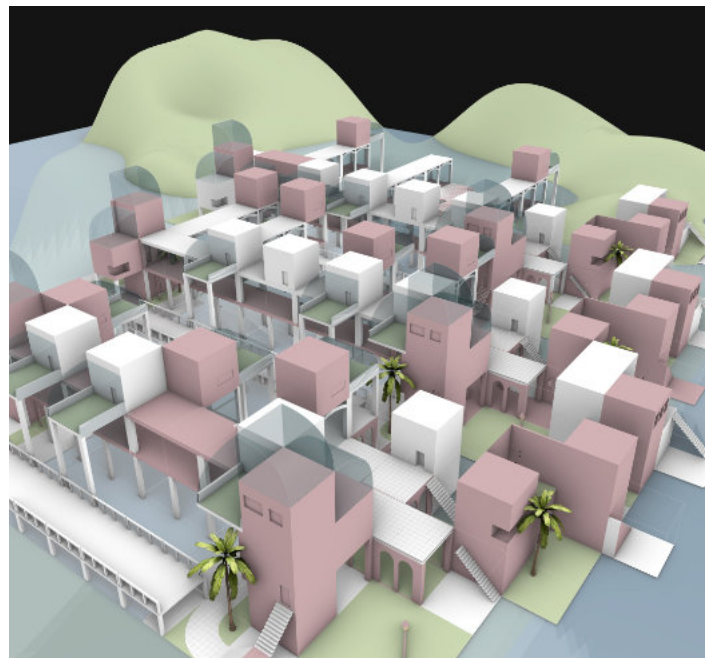
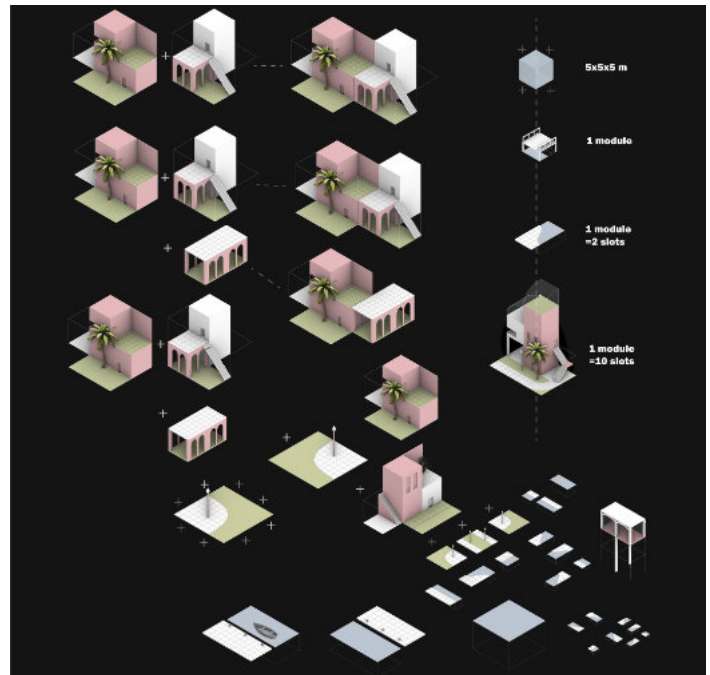
The project is exploring possibilities of discrete assemblies via Monoceros. The exploration was conducted on three urban/housing scenarios - LAND, COAST and WATER as a reference to an imminent danger of rising seawater levels.

LAND

The land scenario of the WFCity explores the possibility to cover as much as possible of a given landscape. It is achieved to cover more of the terrain over time by adding more modules and rules with every evolutionary step. Connected terraces can be used for gardening, streets connect the land to the water, where the land scenario unifies with the coast.

COAST

Lagoons, bays and shallow waters are all alive. So are the spaces above them, rising on the columns. COAST WFCity scenario encourages to imagine urban schemes, organized by an algorithm, on the borders between water and land.



WATERcity

Project explores possible housing typologies based entirely on water. It consists of several branches of designs for different functions. The main goal is to connect the city with the water seamlessly and create typology, which can be applied on different input conditions. This can be achieved thanks to WFC, which can create endless possible aggregates that are applicable on any given state of input landscape.



VIRTUAL DORMITORY

Game

Authors: Kateřina Suchánková, Hugo Fekar,
Eva Kvaššayová

Studio Architecture III, UMPRUM 2020/2021

Project speculates about shifted role of computational architect, analyzes principles of warehousing - one of the last industries to see major change wrought by 21st century technology - and reapp-
lies them to challenge „architectural flexibility“ of the space.

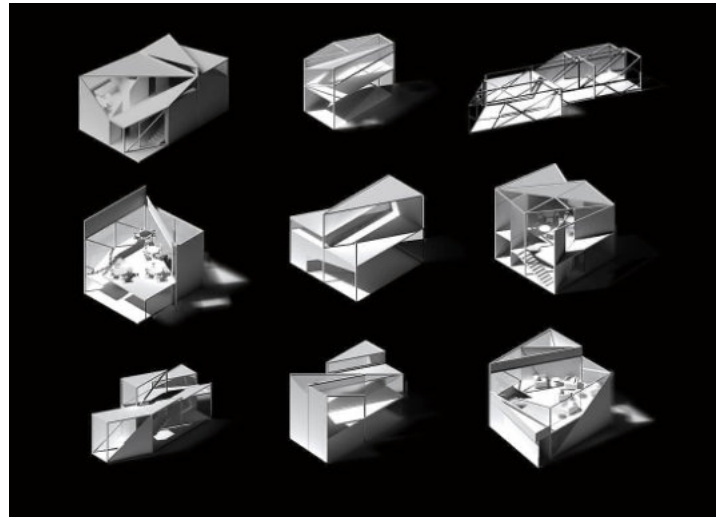
Student housing experiment, where time and seasonability play the key role in architectural displacement, similar to the logistics of the commercial warehouse (goods are shifted throughout warehouse based on season) offers solution, which disrupt physical rigidity of the space and provides personalized user experience.

Spatial inefficiency of the standard dormitory lies with the inability of the spaces to respond to the real-time requirements, inadaptability to the actual needs, comfort and expectations of the users.

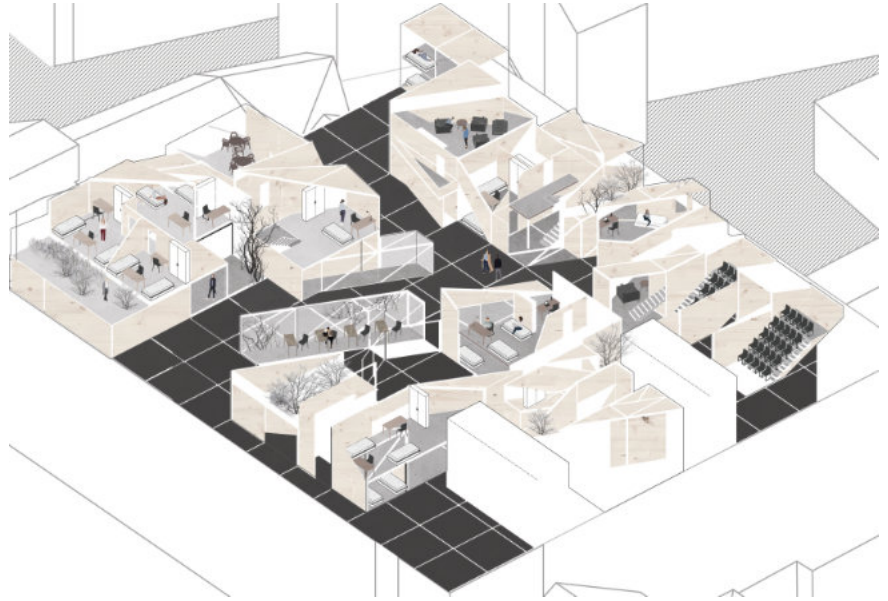
Built environment of the dormitory is not considering, if the capacity of the spaces needs to be fully fulfilled or not.

But through constant remodelling and reprogramming, brought by digital revolution, building could change on a monthly, weekly or daily basis and could achieve greater efficiency of the spaces.

Reorganization of the dormitory is fully automated thanks to the predictive algo-



rhythm – following traffic and occupancy of the users. System consists of specific amount of flexible elements, which interact with each other, depending on the speed and direction of the movement. Given specific starting point, element is able to find the most efficient way to its destination. It is able to stop on specific place for specific amount of time, even it can have a delay before start (in order to avoid collisions). The result presents new unexpected spatial consequences and creates fast space layout cycles.



04 PROTOTYPES

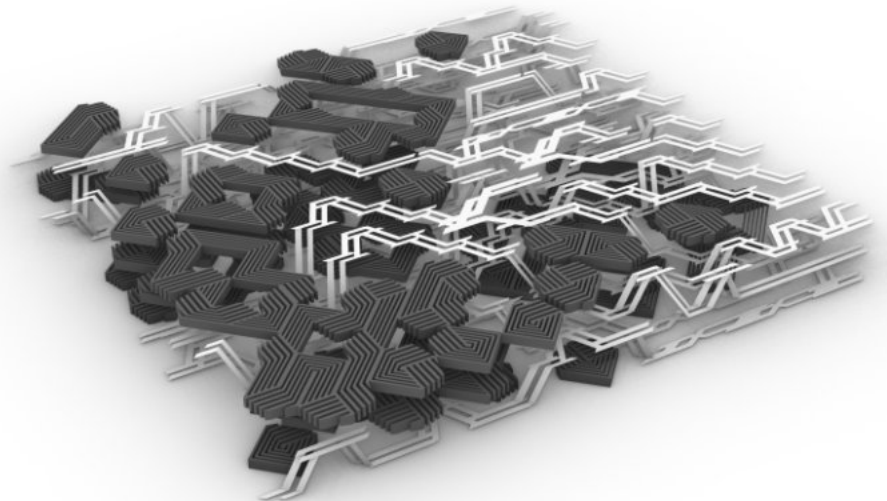
Fabrication experiments

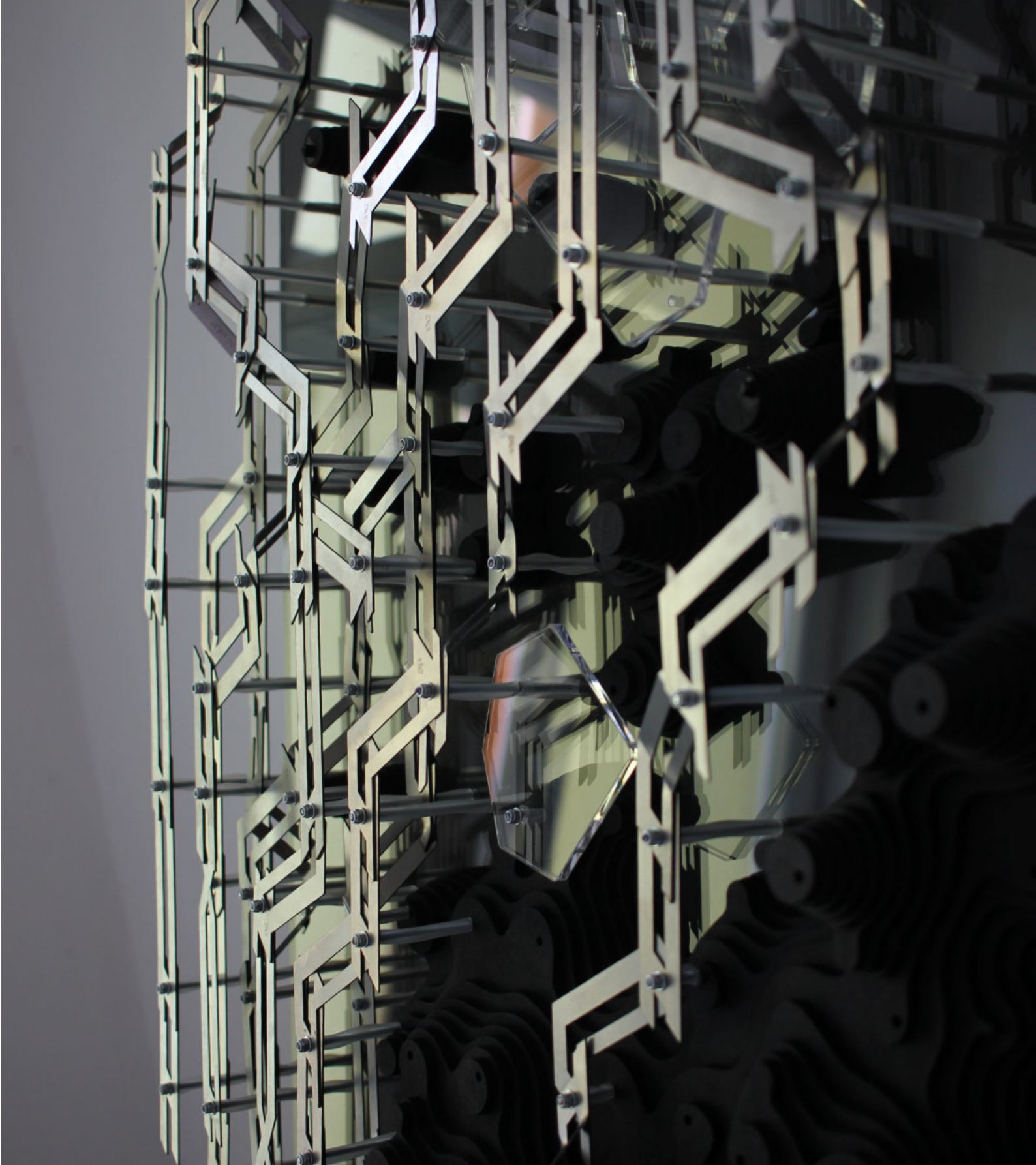
MATERIAL MORPHOLOGIES

Aggregation formal study

Internet is full of images. Creating an appealing render of unreal, digital geometries that stays within the borders of computer screens is most of the time not convincing enough to reach the wider audience outside our community. Our interest in Subdigital is therefore to transform the digital results of our design to the real world, to bridge between digital and physical.

It seemed difficult, if not impossible to fabricate and materialize a complex and compelling aggregated structure using conventional methods. Current parametric design tools are creating spectacular results on the computer screen. Not all these geometries are possible to recreate with the limits of physics laws and fabrication methods. Most of those design tools required use of voxelization, a brute force of geometry, to turn the data into matter. It mostly required time consuming process of 3D printing the complex objects, or tedious man-power driven work, assembling the multitude of unique components.





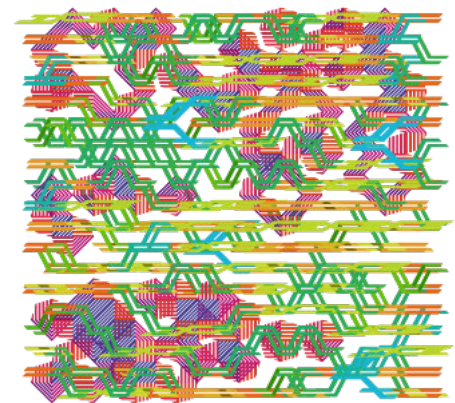
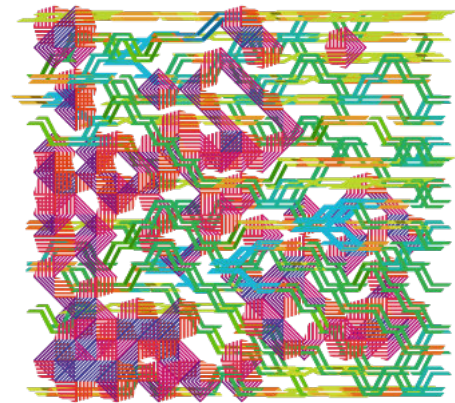
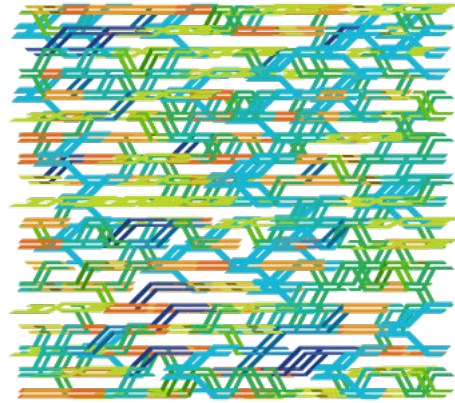
Formal study

What if we created a method that would grant us the possibility of easier fabrication without losing the complexity?

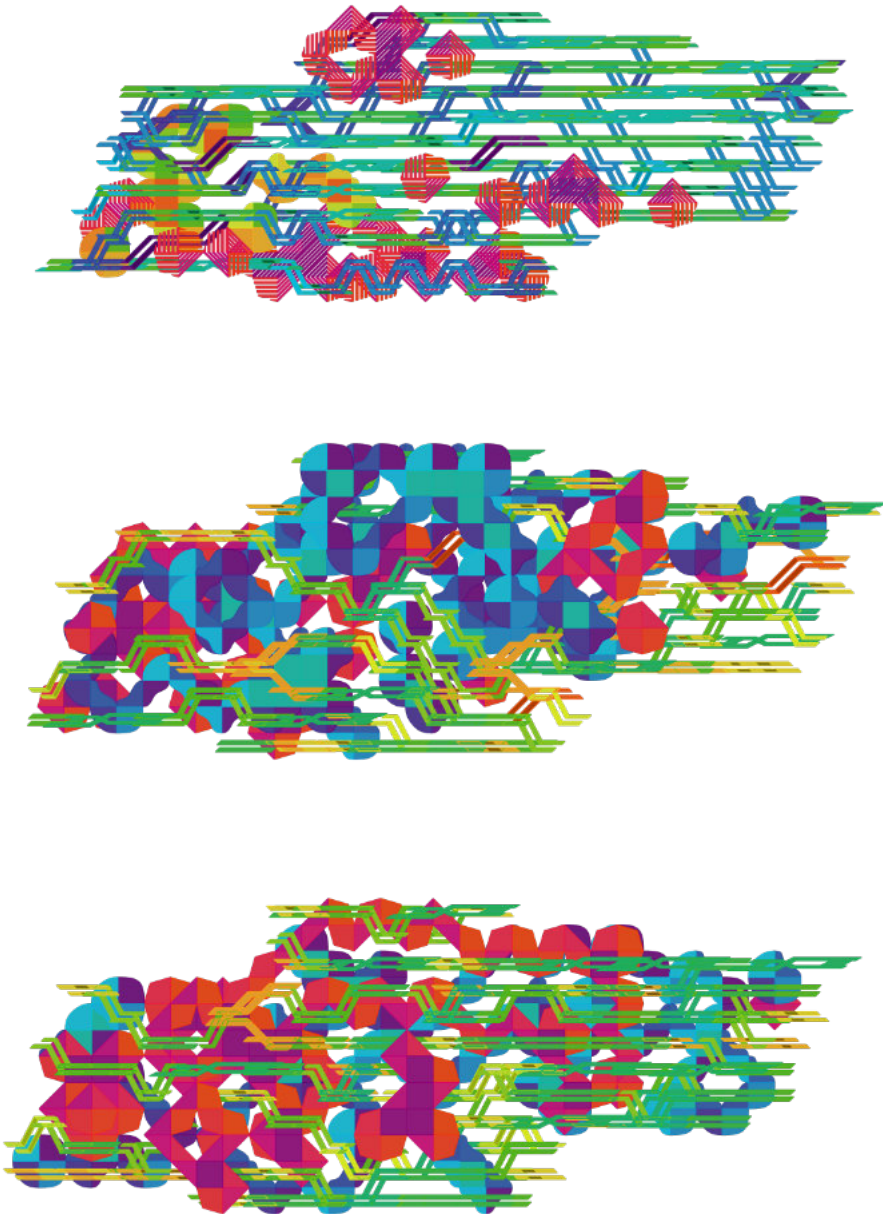
Previously available tools were often limited to define the geometry within the limits of one material. It often takes some level of post-production, to define a coherent object with multiple materials.

How to design a system of multiple weaved aggregated structures?

Aim of this case study was to apply the Wave Function Collapse (an algorithm, at that time we were developing an implementation of this algorithm into our computational tools) in design prepared for fabrication, test the detailing of connections between elements, and to test material possibilities. Our implementation of the WFC algorithm helps us turn data into an aggregate of well-defined discrete elements of several types that connect to each other in a prescribed way. The elements are designed with digital pre-fabrication in mind, they never overlap and never fall into a wrong place.



Variations



We defined one type of connectors for the elements of one material type. The rules defined within one type ensure, that the modules of the same type have some kind of connection or relationship between one another. Various rules are defining how these one-material elements are aggregable within the system. Together having three materials means having three types of connectors with own rules. Another type was then also defined due to inter-material interactions. Indifference and exclusion rules between the units of different materials ensure their coexistence and intertwined character of the structure.

WFC system of Material Morphologies is therefore based of two main ideas – rules of connections and rules of indifference and exclusion. Precisely defined connections between units of one material not only arrange a structure of predictable technical detail for physical assembly of the structure, but also creates a "flow" of material in the structure.

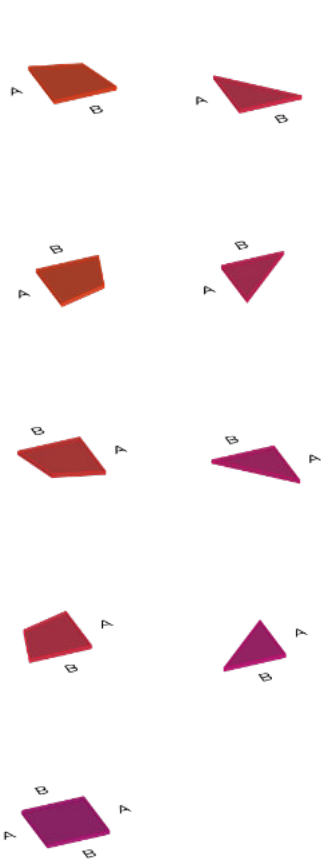
After all the necessary information is assigned, rules and connectivity definitions necessary for our tool using the WFC algorithm will fill a defined space. Using the design principle based on local connections, not global appearance allowed us to unify the construction details without having to compromise the formal complexity.

Modules and connection rules

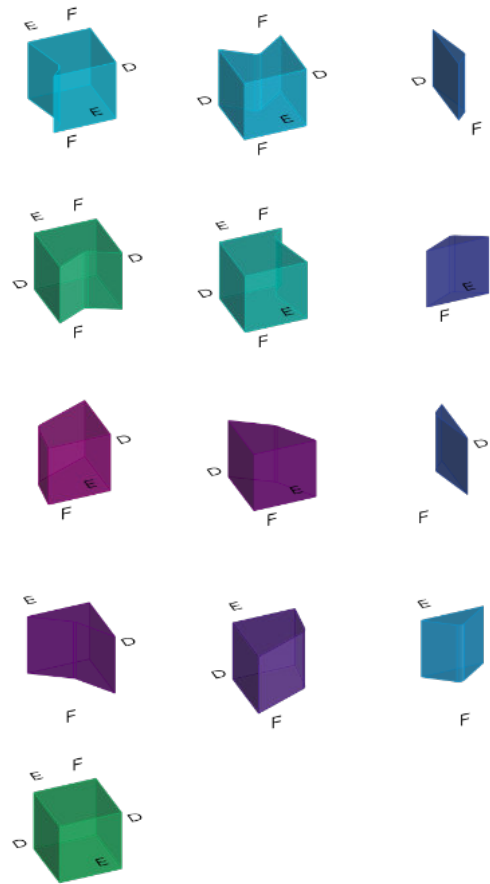
The name Material Morphologies refers to the complex landscape of several units made of a few of materials. One material creates one layer of landscape, multiple layers coexist and intertwine, forming one object. The aggregated structure creates coherent flow of materials with changing spatial characteristics. We evaluated the structure by aesthetic and spatial criteria, considering it to be a proto-architectural object.

We have developed a design-oriented version of the Wave Function Collapse aggregation method that can be later used for partial tasks in architectural design. A software tool that can be used to generate a multitude of abstract and applied examples in various scales.

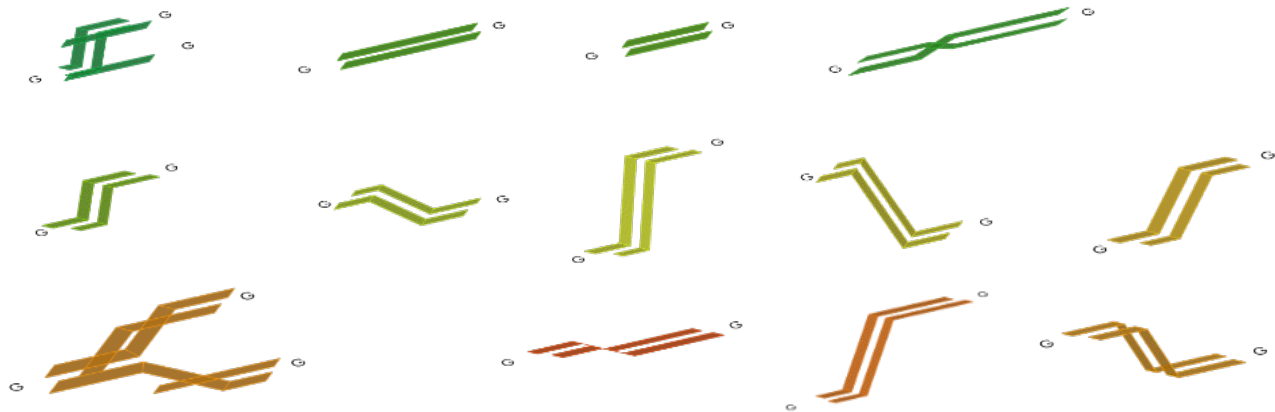
PLEXI



WOOD

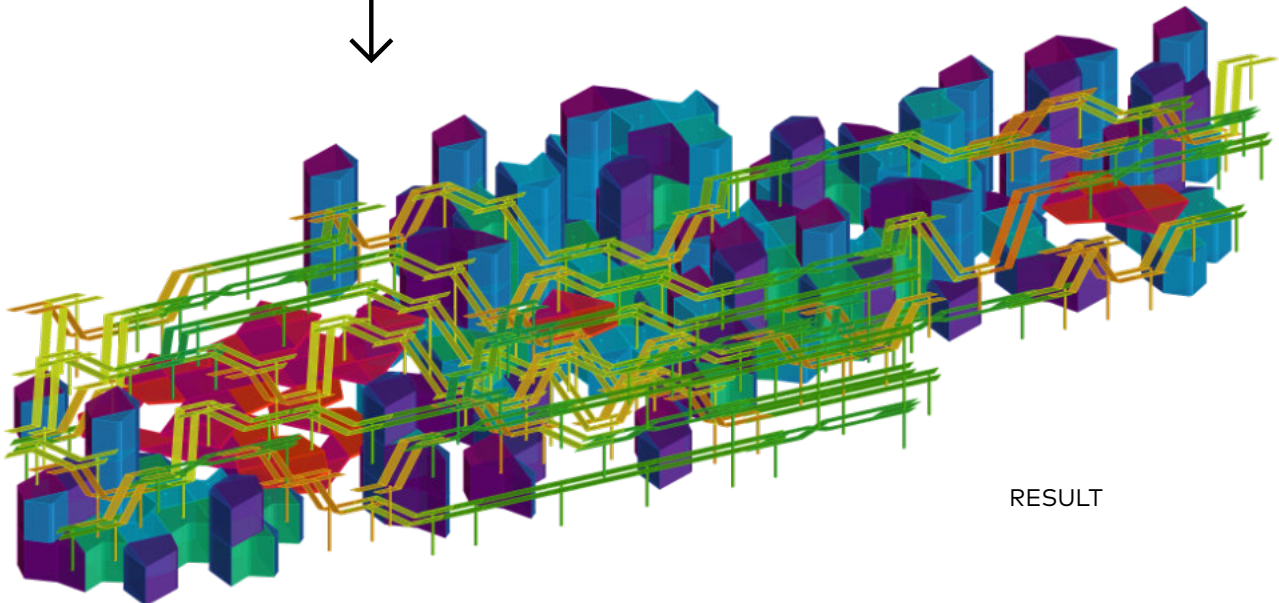
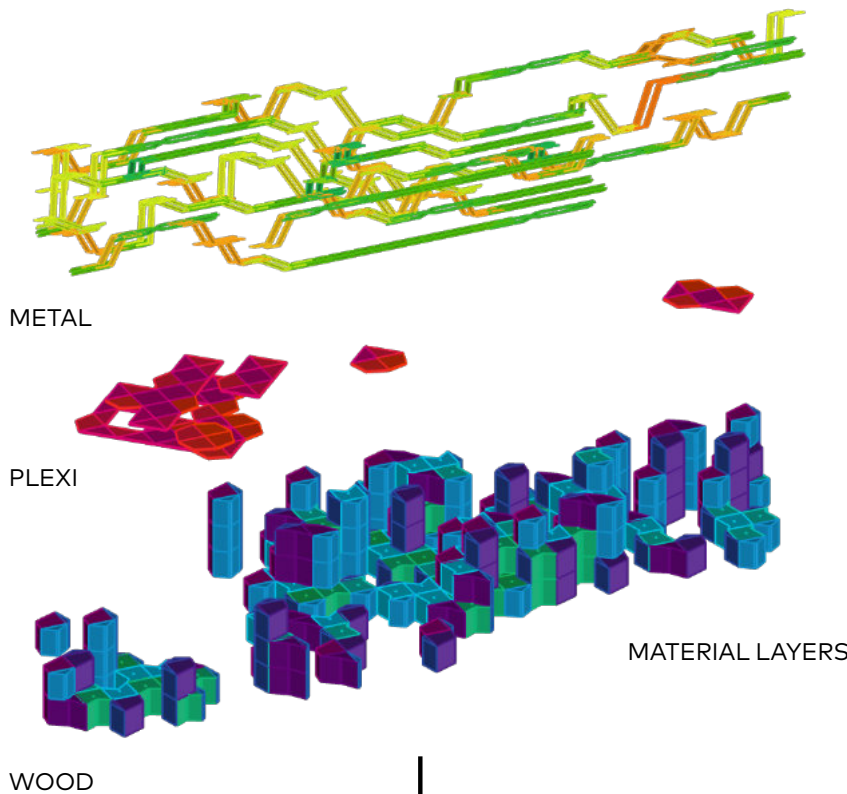


METAL



Result

What if we created a system that is defined by finite number of repeating modules of different materials with its own inner rules of connectivity within itself and in between these materials?



Materialization

The whole WFC system follows and obeys the laws of a rasterized 3D space. Discreet elements were designed with the intent to behave and fit within this kind of space as well. Of course, that does not necessarily mean to stay trapped within the borders of boxes with the design of discreet elements entirely. The inevitable requirement for the algorithm to successfully compute the result are the locations of the connectors. It is the location between two slot faces of the invisible rasterized space that holds all the information, all the rules and relationships necessary for assembling pieces together.

Exactly this had proven to be not a limitation, but an advantage while fabricating and assembling physical model. Not all slot faces need a physical connection. Some connections stay hidden within the coherent discreet element that spans across multiple slots of the space. Some places of connections, by the pre-defined rules of indifference can stay empty. The remaining places of contact are consequently not necessary creating a regular rasterized grid, but still are regular enough to be

easily located and assure quick and effortless assembly of the elements in physical model.

This way the finished object can stay organic and fluent and remain the order and inner organization necessary for creating a documentation and following materialization of the digital model.

How to turn the digital image into a physical object?



Exhibition

This project was a part of the Studio Architecture III Vysoká škola uměleckoprůmyslová v Praze exhibition at Designblok, Prague International Design Festival 2020. Many thanks to our team!

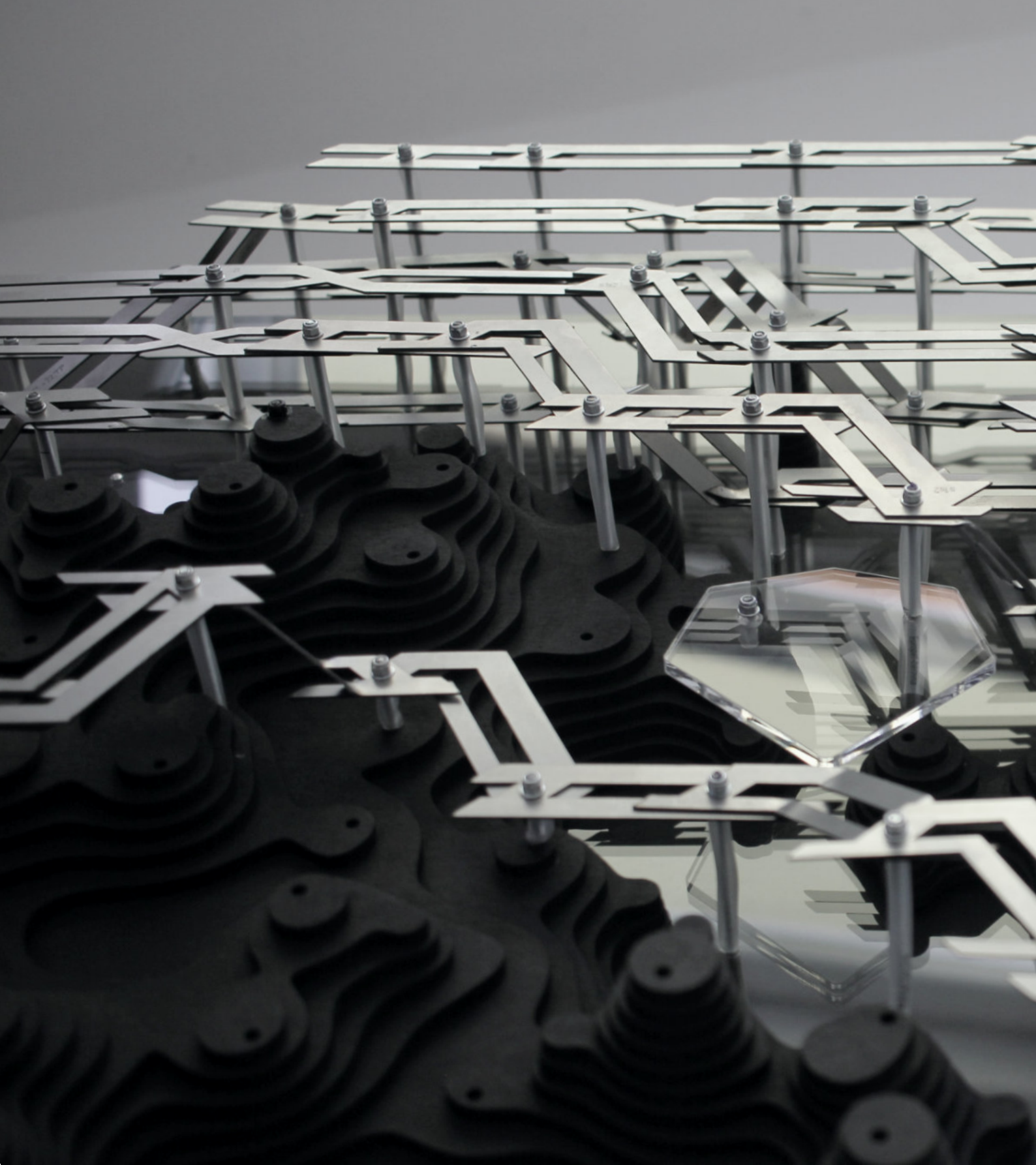
Mimi Löscher (FAD STU, Die Angewandte), Suulsak Sk, Eva Kvaššayová (UMPRUM), Ľuba Ondrejkočová (FAD STU), Hugo Fekar (UMPRUM), Ján Pernecký (UMPRUM / Subdigital), Tomáš Tholt (FAD STU / Subdigital), Janči Tóth (Subdigital), Michaela Krpalová (AFAD / Subdigital), Kristyna Uhrova (Subdigital).

Great work on digital fabrication of components was done by 4from media, sro and Colab Slovakia.











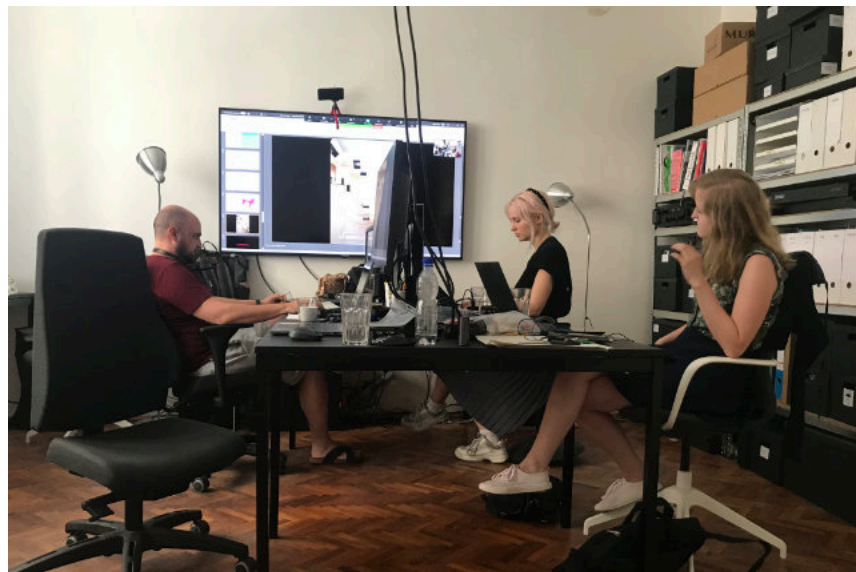
05 WORKSHOPS AND EVENTS

SUBDIGITAL SUMMER SCHOOL 2020

How it started!

In the summer of 2020, a group of authors Ján Pernecký, Tomáš Tholt, Ján Tóth, Michaela Krpalová, Kristýna Uhrová, Hugo Fekar, Alexander Kupko, Eva Kvaščayová, Miriam Löscher and Ľuba Ondrejkočová prepared a research project, the result of which is a proposal to adapt the mathematical method / algorithm Wave Function Collapse for architectural design purposes.

The subject of Subdigital Summer School 2020 was current and near-future big topic in computational design, prefabrication and discrete design. The early objectives were to identify the present state of the discrete design scene, as well as to learn and channel the mindset the principles of wave function algorithm while developing a comprehensive software tool that would support such design processes.



Summer school desk crits

Within the span of summer months, we consulted our progress with guest specialists of various backgrounds. With their help we acquired an insight from theoretical and academical point of view, trough the view of software developers. We met and discussed with technologists and professionals in computational design, digital fabrication and robotics as well as end-user specialists, on the for interdisciplinary collaborations between art and tech community, and the UX professionals.



Guest reviewers

Imro Vaško/UMPRUM

Shota Tsikoliya / UMPRUM

Martin Kossuth / MORE

Aldo Sollazzo / Noumena / IAAC

Michal Hladký / EHMKK 2013

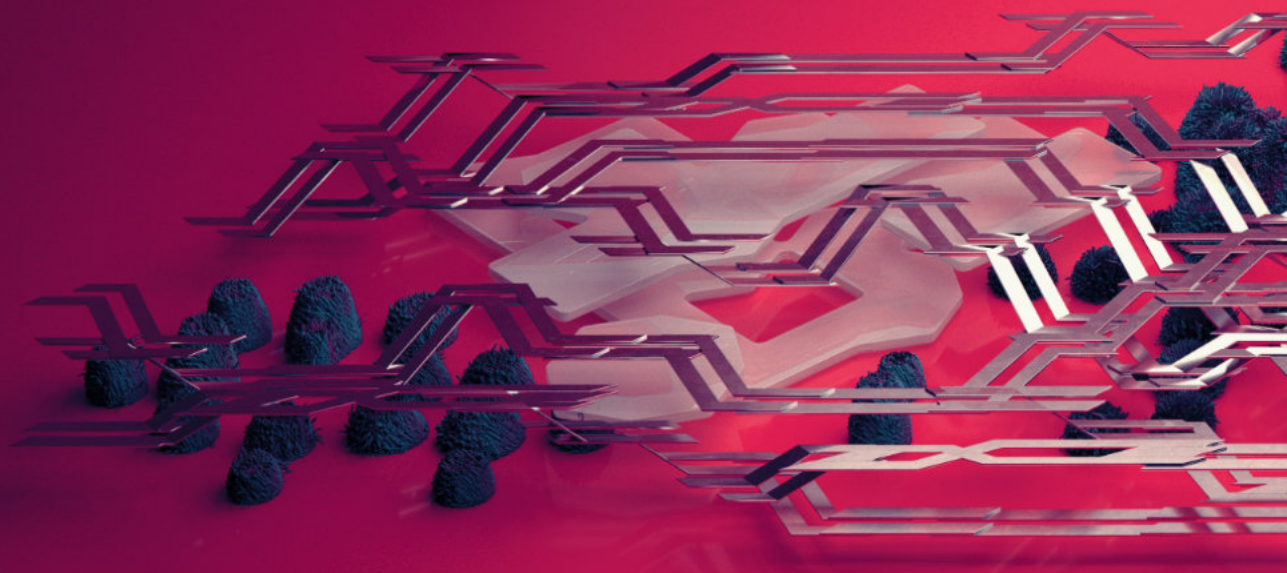
Lucia Dubačová / Sensorium

Martin Uhrík / DataLab

Roman Hajtmanek / DataLab

Alessio Erioli / Co-de-iT





Subdigital 2021