



JULIUS ÜBERALL

Surename	Überall
First name	Julius
Date of Birth	28.03.1999
Nationality	German
Languages	German (Native), English (Full professional proficiency)
E-Mail	juliusueberall@gmx.de
Phone	+49 162 2803990
Address	Am Fährweg 1 41468 Neuss Germany

EDUCATION

2021 - September 2022

Architectural Computation (M.Sc.)

The Bartlett School of Architecture (University College London, UCL), United Kingdom

Scholarship

German Academic Scholarship Foundation

2019

Architecture (Study abroad)

Virginia Polytechnic Institute and State University, VA, USA

GPA: 4,0

Language certification

IELTS: 7,0

2017 - 2021

Architecture and Interior Design (B.A.)

University of Applied Sciences Düsseldorf (Hochschule Düsseldorf, HSD), Germany

Dean's recognition for best in year

Grade: 1,4

Family Scholarship (Borr)

Kölner Gymnasial- und Stiftungsfonds

2009 - 2017

Abitur (High school graduation)

Quirinus-Gymnasium Neuss, Germany

Grade: 2,4

WORK

2019 - 2021

Co-Founder

YÜ - Words become jewellery, Düsseldorf, Germany

www.yubyyou.com

After coding related design research in the second year of my undergraduate degree at the University of Düsseldorf (Germany) I conceptualized and implemented a deterministic algorithm which transforms letters into jewellery. Together with my deans research assistant I teamed up, developed the concept further and founded the brand 'YÜ'.

SKILLS

Rhinoceros



Grasshopper



C#



Python



Processing.js



HTML5,CSS,



JavaScript



Adobe CS



Cinema 4D



Blender



Microsoft Office



3D Printing



ABB Robotics



March 2020 - July 2020

Student assistant

Dean Prof. Dipl. -Ing. Judith Reitz / Fundamentals of Architecture

University of Applied Science Düsseldorf, Germany

February 2019 - March 2019

Volunteering

Design build 2019 - Primary School Santiguah, Guinea

Social Impact Studio, University of Applied Sciences Düsseldorf

November 2018 - April 2019

Student assistant / Event-management

Prof. Dipl. -Ing. Moritz Fleischmann / SDMA, Symposium for digital methods in architecture

University of Applied Sciences Düsseldorf

ADDITIONAL COURSES

AI For Everyone

Coursera, 2021

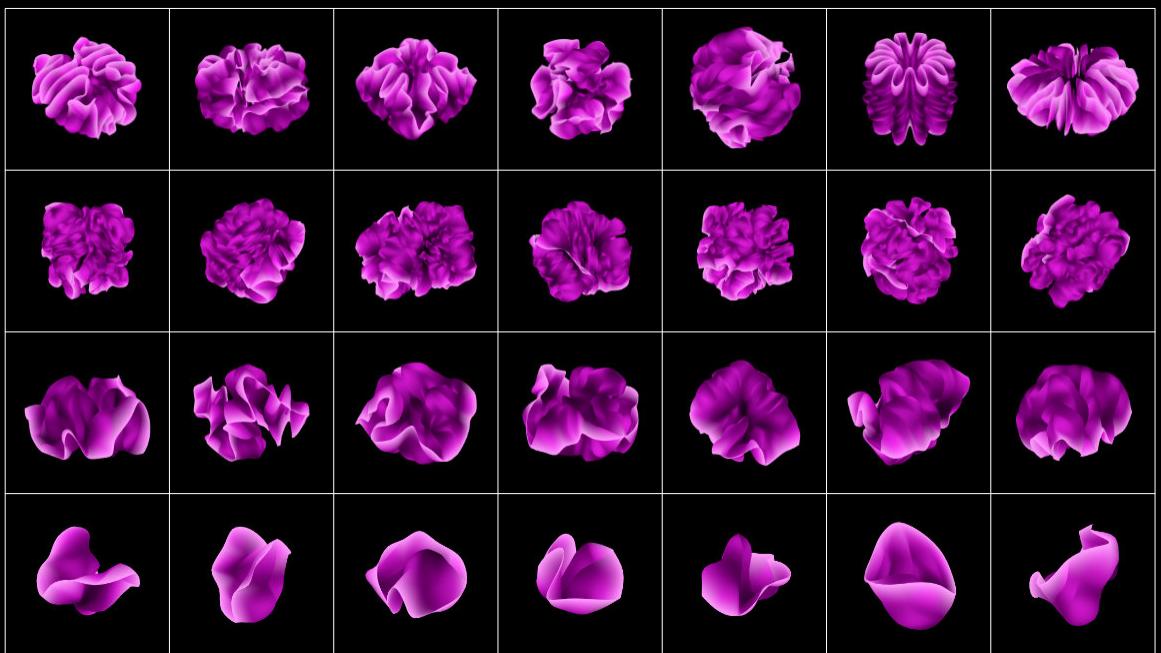
Artificial Intelligence Demystified

German Academic Scholarship Foundation, 2021

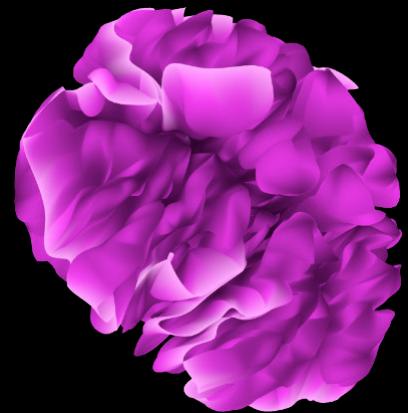
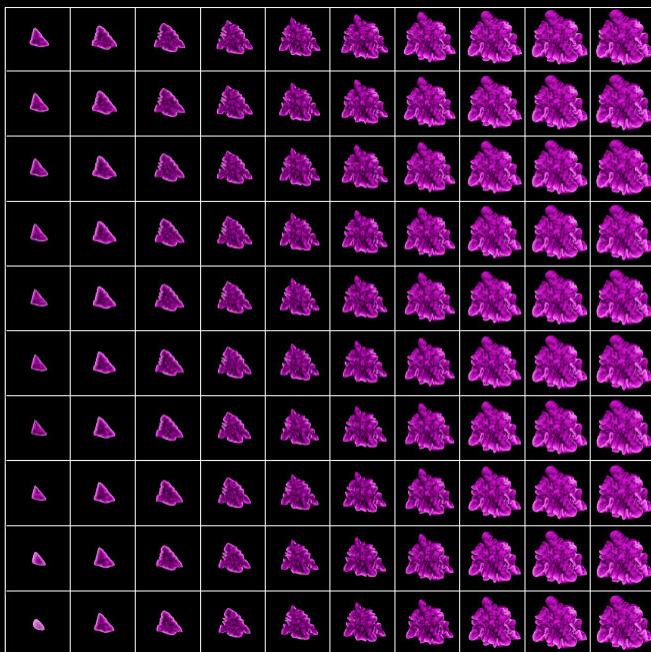
2021 Complete Python Bootcamp

Udemy, 2021

- Page 4-5* **Mesh Growth**
Simulating organic shapes through algorithms
- Page 6-7* **Dwelling Configurator AR App**
Interactive architecture
- Page 8-9* **Parametric Roof**
Reactive mesh roof
- Page 10-11* **YÜ**
Start-Up for algorithm designed jewellery
- Page 12-13* **MÖ**
Final project in first year of architecture school
- Page 14-15* **Habitat B**
Extraterrestrial architecture
- Page 16-19* **B211**
Bachelor Thesis project
- Page 20-21* **Art & Structures**
Ongoing drawing and modelling exploration



GROWTH EVOLUTION



■ Boundary edge
■ New vertex
■ Old vertex

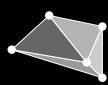
Mesh Simulation for 5000 vertices

MESH GROWTH

The project aims to create a design pipeline for users to form organic shapes through growing simulations. It should enable to form/find custom input geometries and should give the user visual feedback through real-time simulations and geometry colorings. It is developed with C# for Grasshopper.

The MeshGrowSystem class is the core of the simulation and encapsulates the two main methods to grow a given input mesh - RelaxRTree() and Mgrow(). Through a finite loop of these two methods the mesh vertices system gets continually relaxed and extended, by splitting mesh edges that become too long. To optimize computation time i implemented a R-Tree to find closest neighbours faster within the mesh vertices cloud.

Input



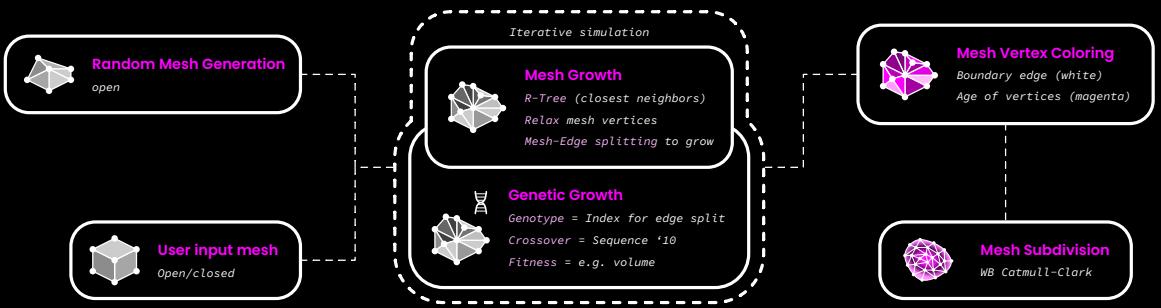
Output



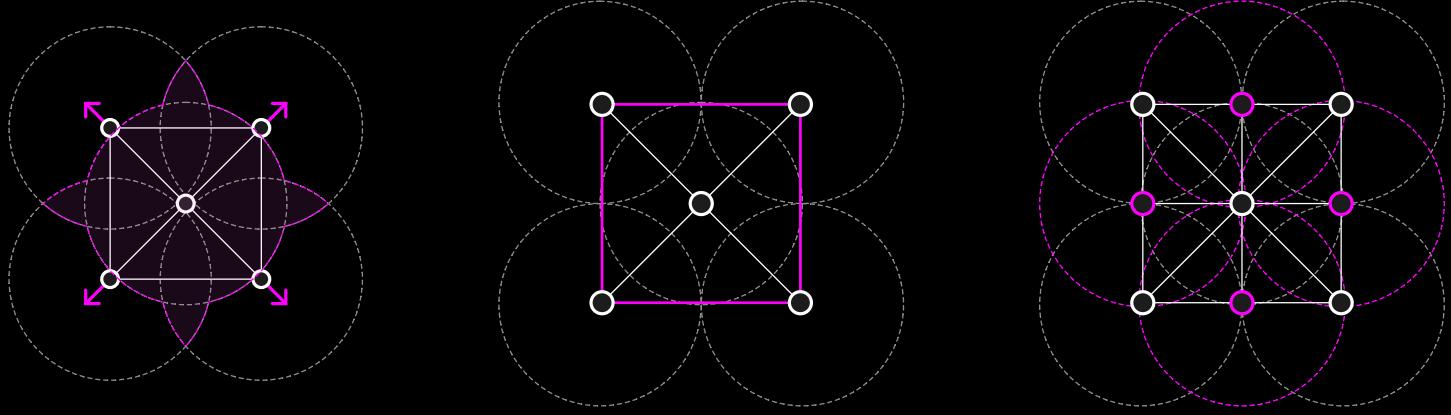
GOAL

Output

PIPELINE



GEOOMETRY GROWTH



#1 - COLLISION

If two cells are too close, they will both move away from each other - reaction detection.

#2 - RELAX

If the system of cells is enough distanced, the system is relaxed and stops moving.

#3 - GROW

If two cells are too close, they will both move away from each other.

PSEUDO CODE

```

public class MeshGrowSystem{
    //PROPERTIES
    private R-Tree rTree; //R-Tree with current vertices, to find closest in vertex cloud
    private List<Point3d> rPoints; //For callback event of R-Tree search in range

    //CONSTRUCTOR
    public MeshGrowSystem(Mesh m){
        //Duplicate start-mesh(m) and store in class
    }

    //RELAXATION OF MESH VERTICES WITH R-TREE
    public void RelaxRTree(double relaxDis){
        //Update R-Tree with transformed mesh vertices
        //Create List<Vector3d> to store movement of mesh vertices before transforming
        for (//All vertices){
            //Use R-Tree to get vertices that are too close and in relaxDis
            for (//All vertices in relaxDis){
                //Calculate push-Vector3d to move out of relaxDis for each in-range vertex neighbor
            }
            //Average Sum of all push-Vector3d for each vertex
            //Add to movement List<Vector3d> for mesh vertices
        }
        for (//All vertices){
            //Transform each vertex with movement List<Vector3d>
        }
    }

    //CALLBACK EVENT FOR R-TREE SEARCH
    private void RTreeEventHandler(object sender, RTreeEventArgs args){
        //Add found vertex within relaxDis to rPoints
    }

    //MESH GROWTH THROUGH MESH-EDGE SPLITTING
    public void MGrow(double relaxDis){
        if (//Current count of verticies < //Maximum amount of vertices){
            //Create lookup list for edges to split List<VertexPair>
            for (//All mesh edges){
                if (//Mesh edge length > relaxDis){
                    //Store vertex pair for edge to split in lookup list
                }
            }
            for (//All vertex pairs of edges to split){
                //Get correct index of edge with vertex pair
                //Split edge with received index
            }
        }
    }
}

```



HOUSING APP

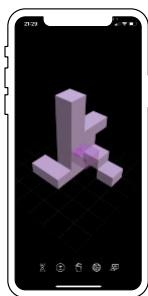
Reimagine Housing.

This is the latest coding group project that we developed with a group of four students. Our goal was to speed up the client-architect design process, by developing an interactive AR App. The architect would pre-design a set of architectural components and the client would be able to manually adjust and aggregate those in a virtual space, which could be accessed with any given iOS device.

Our team investigated different domains to automate specific design operations to make the interaction for the user as easy and playful as possible.

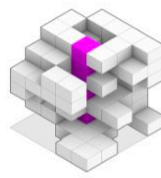
I implemented the front- and backend of the app and merged the results of my team members. We used C# in Unity combined with multiple AR packages to access the ARKit framework of apple devices.

FRONTEND FLOW



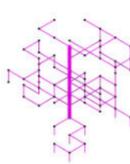
User Seed

Augmented Reality

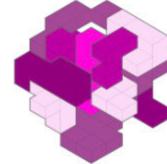


Aggregation

Cellular Automata
Genetic Algorithm



Graph



Clustering

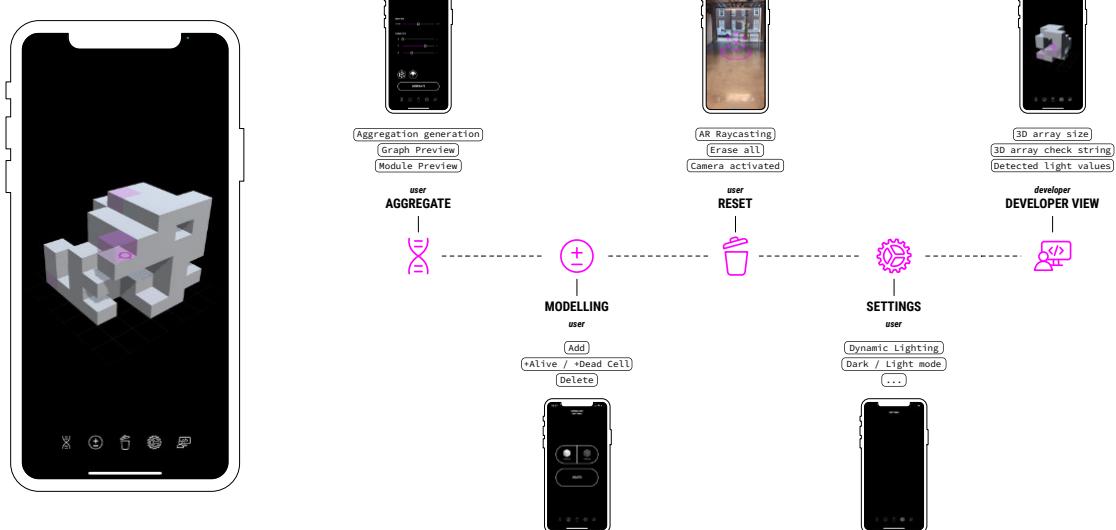
K-Means Clustering



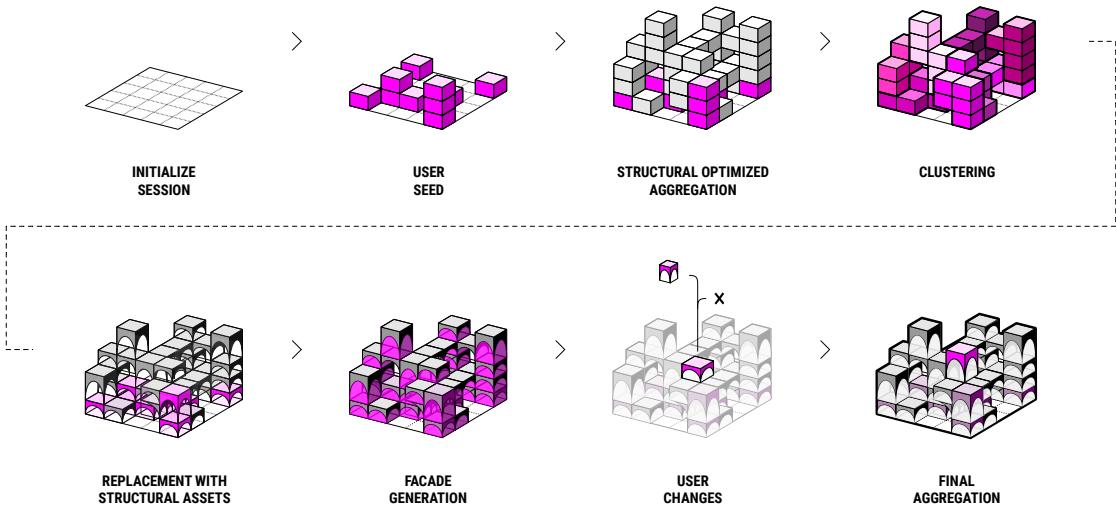
Modules

3D Graphic Statics

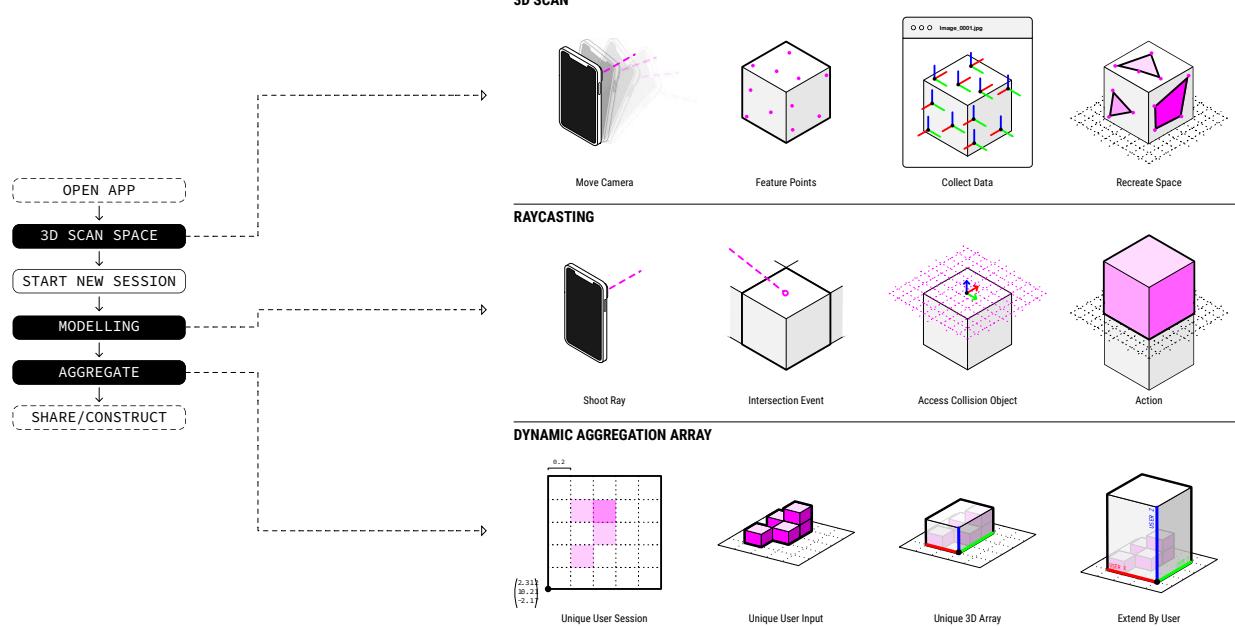
USER INTERFACE

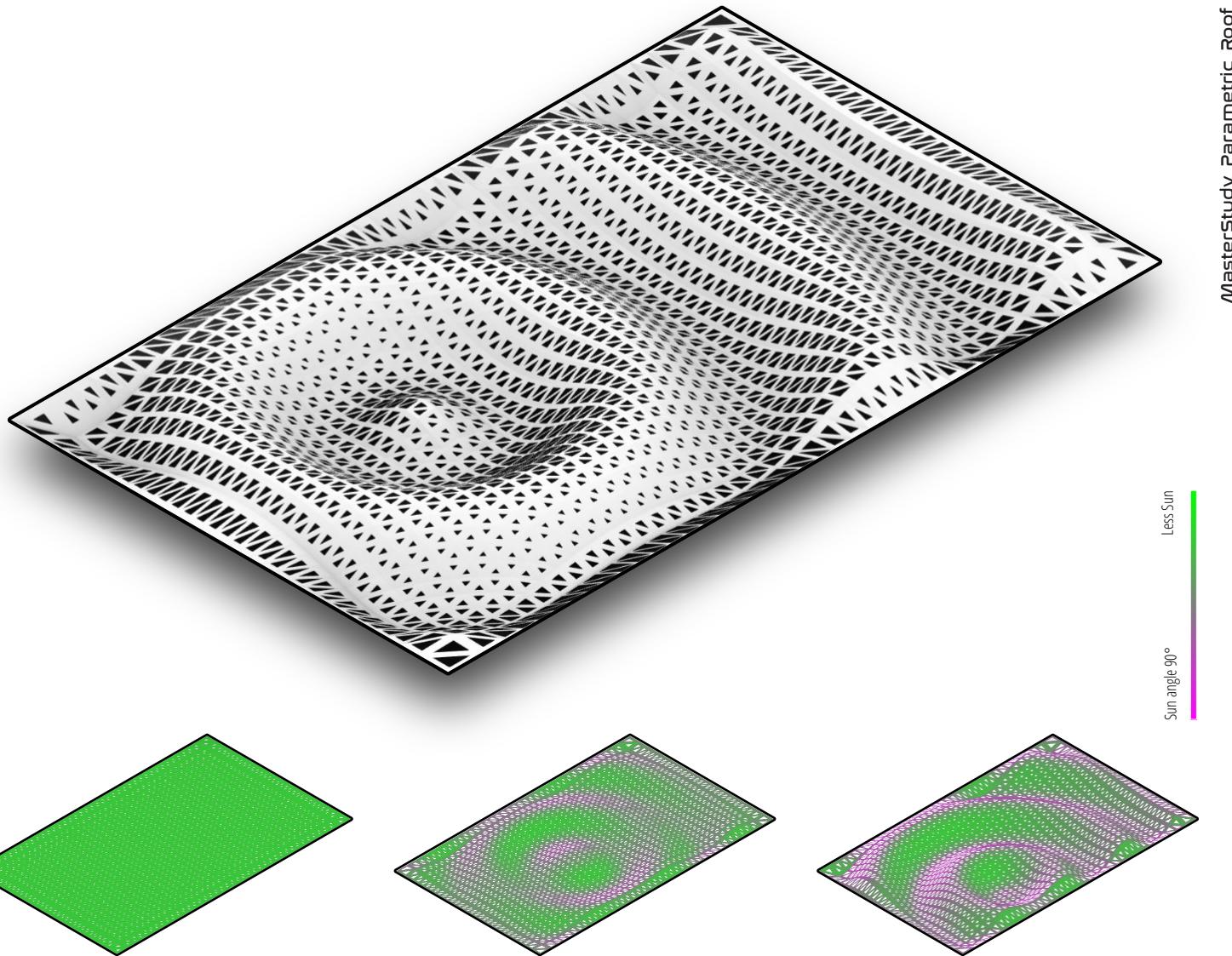


BACKEND

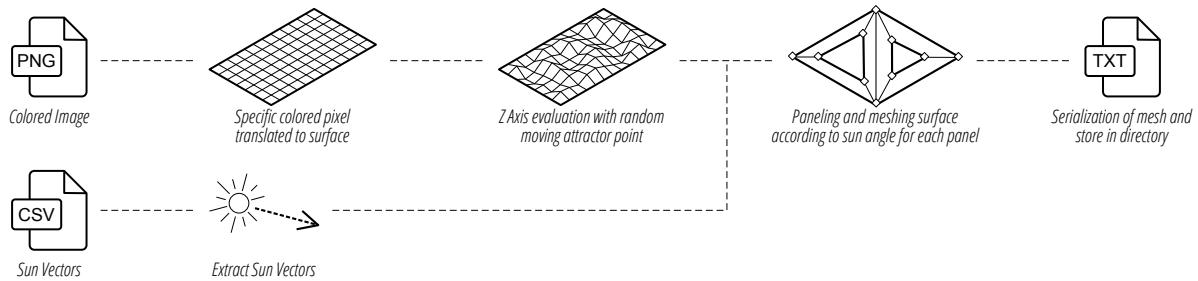


AR INTERACTIVITY LOGIC





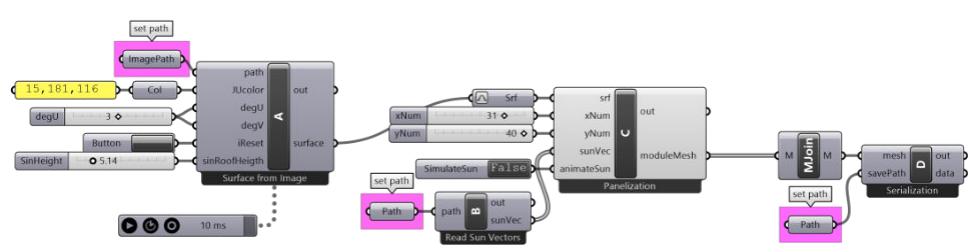
C# WORKFLOW



PARAMETRIC ROOF

This C# script is the latest and one of the most advanced coding projects I have done so far. It was the final project in my first term during my M.Sc. in Architectural Computation. The Grasshopper script consists of almost 100% out of custom-written components which use the Rhino API and some additional libraries to create a parametric animated roof out of panels.

As visualized in the diagram above, I generated a basic surface from an image using a specific target color to calculate it from the colored pixel. I also extract the sun vectors for London in June from a CSV file to integrate these vectors into the parametric system. Together with a random walker, I evaluated the basic 2D surface into 3D and meshed and colored each panel according to its relation to the sun.



PANELIZATION - C# extract

Component C in Grasshopper Script

RunScript

```
//Component Message
Component.Message = "Panelization";
//Create Roof Class
assignmentRoof = new Roof(srf, xNum, yNum);
//Call roof division function
assignmentRoof.DividesSurfaceIntoPointGrid();
//Create Triangular Mesh from surface
assignmentRoof.TriangularMeshRoof();
//Extract Triangular Face edges
assignmentRoof.DrawTriFaceEdges();
//Calculate Triangular Face Planes
assignmentRoof.CalculateTriFacePlanes();
//Animate sun
if (animateSun){
    //Select Sun vector (Select new sun vector if animated)
    sunVectorSelector = sunVectorSelector + 1;
}
//If not animated stick to first static sun Vector
else{
    sunVectorSelector = 0;
}
//Select new Sun Vector
currentSunVector = sunVec[sunVectorSelector];
//Create Parametric Modules with sunVector angle
assignmentRoof.ParametricModule(currentSunVector);

//OUTPUT
moduleMesh = assignmentRoof.moduleMeshes;
```

Additional Code

```
//ANIMATION VARIABLE - ONLY ONCE
Vector3d currentSunVector;
int sunVectorSelector = 0;
//DISPLAY REASONS
Roof assignmentRoof;
```

```
//<<< MODULE CLASS >>>
public class Module{

    //++++ PROPERTIES +++
    //module Number
    int moduleNumber;
    //Module Plane
    Plane plane;
    //List of Vertices
    List<Point3d> faceVertices;
    //Module Edge PolylineCurve
    PolylineCurve faceEdge;
    Polyline faceEdgePolyline;
    //Centerpoint of face
    Point3d centerPoint;
    //Module opening vertices movement lines
    List<Line> openingVerticesMovementLines;
    //Triangle Offset
    //points
    List<Point3d> offsetPoints;
    //Polyline
    Polyline offsetLine;
    //Mesh vertices List
    List<Point3d> meshVertices;
    //Final Mesh
    public Mesh moduleMesh;
    //Current Sun Vector London
    Vector3d sunVector;
    //Angle how Sun hits the module
    //0=90degree hit, the greater the less sunlight
    public double sunAngle;
    //Color for mesh (depending on sun angle)
    public Rhino.Display.DisplayMaterial panelMaterial;

    //++++ CONSTRUCTOR +++
    //Store data in object and calculate sun angle hit of module
    public Module(Plane _plane, List<Point3d> _faceVertices,
        int _moduleNumber, PolylineCurve _faceEdge,
        Vector3d _sunVector){
        //Store important module data in object
        plane = _plane;
        faceVertices = _faceVertices;
        moduleNumber = _moduleNumber;
        faceEdge = _faceEdge;
        sunVector = _sunVector;
    }

    //Sun Vector relation to module normal
    //Invert panel normal
    Vector3d invertedNormal = plane.Normal * -1;
    //Angle in Degrees (0==90 degrees hit of the panel, if larger
    //number less and less direct sun, more than 90 == no sun at all)
    sunAngle = RhinoMath.ToDegrees(Vector3d.Vector
        Angle(invertedNormal, sunVector));
}
```

```
//++++ METHODS +++
//#1 Custom offset edge for module opening (depending on panel triangle
//shape)
public void OffsetModule(double remapHighBound){
    //Centerpoint
    //Create basic face edge polyline instead of polyline curve (initial idea was
    //to use the offset method of the polylinecurve class)
    List<Point3d> closedPolylineVerticesList = new List<Point3d>(face
    Vertices);
    closedPolylineVerticesList.Add(faceVertices[0]);
    faceEdgePolyline = new Polyline(closedPolylineVerticesList);
    //Calculate center of face
    centerPoint = faceEdgePolyline.CenterPoint();
    offsetPoints = new List<Point3d>();
    //Move Vertices
    //Instantiate List for corner center lines(opening vertices
    //movement lines)
    openingVerticesMovementLines = new List<Line>();
    //Instantiate List for offsetted points

    //Create movement lines for opening vertices of module
    for(int i = 0; i < faceVertices.Count;i++){
        //Create movement line for checking/display purposes
        openingVerticesMovementLines.Add(new Line(centerPoint,
        faceVertices[i]));
        //measure length of each line
        double lineLength = openingVerticesMovementLines[i].Length;
        //remap value in length unit
        double remapedValue = (sunAngle - 0) * (lineLength - 0) /
        (remapHighBound - 0);
        //Get Unit vector
        Vector3d vertexMoveVector = new Vector3d(centerPoint -
        faceVertices[i]);
        vertexMoveVector.Unitize();
        //Create movement vector with remaped value
        Vector3d finalVertexMove = vertexMoveVector * remapedValue;
        offsetPoints.Add(faceVertices[i] + finalVertexMove);
    }

    //Prepare list for closed polyline generation
    List<Point3d> closedPolylinePoints = new List<Point3d>(offsetPoints);
    closedPolylinePoints.Add(offsetPoints[0]);
    //create polyline from new vertices
    offsetLine = new Polyline(closedPolylinePoints);
}

//#2 Mesh parametric modules with opening
public void CreateMesh(double remapHighBound){
    //Instantiate Mesh
    moduleMesh = new Mesh();
    //Create one List with all vertices
    meshVertices = new List<Point3d>(faceVertices);
    meshVertices.AddRange(offsetPoints);
    //Store Vertices in mesh
    moduleMesh.Vertices.AddVertices(meshVertices);
    //Loop through vertices to create faces for each module
    for(int i = 0; i < faceVertices.Count;i++){
        //Calculate Index pattern for face generation
        if(i < faceVertices.Count - 1){
            int a = i;
            int b = i + faceVertices.Count;
            int c = i + faceVertices.Count + 1;
            int d = i + 1;
            //Create face
            moduleMesh.Faces.AddFace(a, b, c, d);
        }
        //Last face needs to be generated differently cause of the vertices
        //order in the list
        else{
            int a = i;
            int b = i + faceVertices.Count;
            int c = faceVertices.Count;
            int d = 0;
            //Create face
            moduleMesh.Faces.AddFace(a, b, c, d);
        }
    }

    //Compute Normals and Compact
    moduleMesh.Normals.ComputeNormals();
    moduleMesh.Compact();

    //Calculate Color for mesh
    //Remap angle to RGB color value
    double rValue = (sunAngle - 0) * (255 - 0) /
    (remapHighBound - 0);
    //RGB value formula for material creation
    int rValue = Convert.ToInt32(255 - colorValue);
    int gValue = Convert.ToInt32(colorValue);
    int bValue = Convert.ToInt32(255 - colorValue);
    //Instantiate Display Material
    panelMaterial = new Rhino.Display.DisplayMaterial(System.
    Drawing.Color.FromArgb(rValue, gValue, bValue));
}
```

```
//<<< ROOF CLASS >>>
public class Roof{
    //++++ PROPERTIES +++
    //Base Surface
    Surface roofBaseSurface;
    //Panel points on surface
    List<Point3d> roofPanelGrid;
    //Basic Roof mesh
    Mesh basicRoofMesh;
    //Amount of rectangular grid cells in XY (will be divided into 2
    //triangular faces)
    double xNum;
    double yNum;
    //Amount of Points in XY
    int xNumPoints;
    int yNumPoints;
    //List of Lists for all face vertices grouped
    List<List<Point3d>> groupedFaceVertices;
    //Polylines for wireframe of faces
    List<PolylineCurve> triangularFaceEdges;
    //Triangular face planes
    List<Plane> triangularFacePlanes;
    //Parametric Module List
    List<Module> modules;
    //List of all extracted Module meshes for visualization purposes
    public List<Mesh> moduleMeshes;
    //List of all extracted Module mesh COLORS
    public List<Rhino.Display.DisplayMaterial> moduleColors;
    //Largest angle
    double largestSunAngle;

    //++++ CONSTRUCTOR +++
    public Roof(Surface _srf, int _xNum, int _yNum){
        //store base surface into class
        this.roofBaseSurface = _srf;
        //Instantiate basic mesh
        basicRoofMesh = new Mesh();
        //Store amount of panels & Points for XY inside class
        xNum = _xNum;
        yNum = _yNum;
        xNumPoints = _xNum + 1;
        yNumPoints = _yNum + 1;
    }

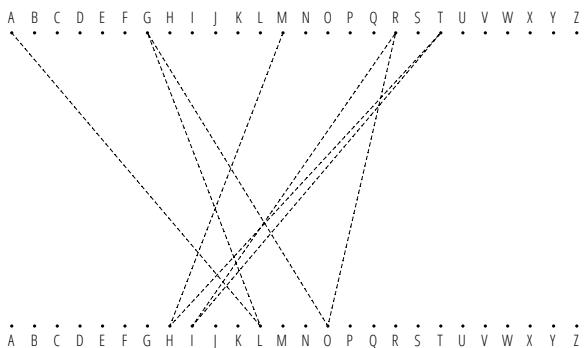
    //++++ METHODS +++
    //#1 Divide Base surface into Point Grid to panelize it
    public void DivideSurfaceIntoPointGrid(){
        //Instantiate Point grid
        roofPanelGrid = new List<Point3d>();
        //Calculate parameter steps according to amount of panels
        //X
        double xParameterStep = 1 / xNum;
        //Y
        double yParameterStep = 1 / yNum;
        //Generate points with correct parameter steps size
        for(int i = 0; i < xNumPoints;j++){
            for(int j = 0; j < yNumPoints;j++){
                //Generate and store points
                roofPanelGrid.Add(new Point3d(roofBaseSurface.
                PointAt(xParameterStep * i, yParameterStep * j)));
            }
        }

        //#2 Mesh triangulate surface
        public void TriangularMeshRoof(){
            //Apply UV points from surface to mesh vertices
            basicRoofMesh.Vertices.AddVertices(roofPanelGrid);
            //Instantiate List of Lists for grouped Face Vertices
            groupedFaceVertices = new List<List<Point3d>>();
            //Create Faces while looping through vertices
            for(int i = 0; i < xNumPoints - 1;j++){
                for(int j = 0; j < yNumPoints - 1; j++){
                    //Calculate vertex indices
                    int a = i * yNumPoints + j;
                    int b = i * yNumPoints + (j + 1);
                    int c = (i + 1) * yNumPoints + (j + 1);
                    int d = (i + 1) * yNumPoints + j;
                    //Create Faces
                    basicRoofMesh.Faces.AddFace(a, b, c);
                    basicRoofMesh.Faces.AddFace(a, c, d);
                    //Store Face points in List of List
                    groupedFaceVertices.Add(new List<Point3d> { roofPanelGrid[a],
                    roofPanelGrid[b], roofPanelGrid[c] });
                    groupedFaceVertices.Add(new List<Point3d> { roofPanelGrid[a],
                    roofPanelGrid[c], roofPanelGrid[d] });
                }
            }

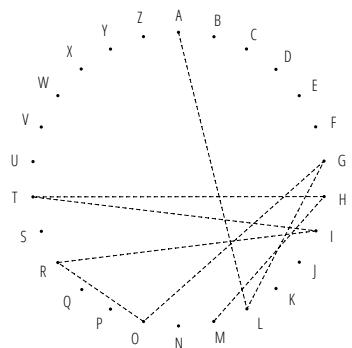
            //Compute Normals and Compact
            basicRoofMesh.Normals.ComputeNormals();
            basicRoofMesh.Compact();
        }
    }

    (...)
```

73 more lines for this Component



"Algorithm"

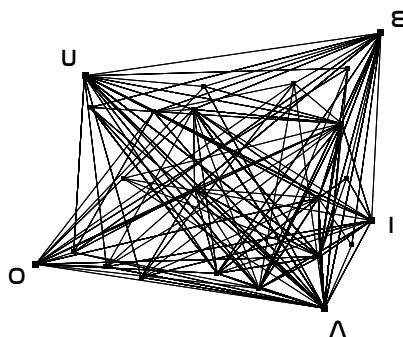


"Algorithm"

YÜ Words Become Jewellery

The U.E.D. algorithm was originally written in Processing.js to transform books into large scaled megastructures.

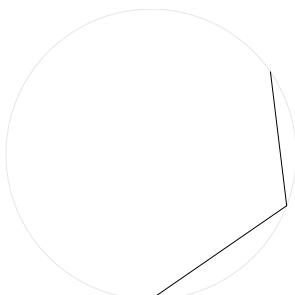
The algorithm's principle is extraordinarily simple and therefore easy to understand. Everything starts with an entered word, sentence or paragraph in latin characters. The whole input is going to get split up into individual letters. You get an unique and deterministic sequence which will be used to draw its related pattern. The algorithm drafts a continuous line following the inputs sequence. Result is a specific web, capturing the word's letter connections.



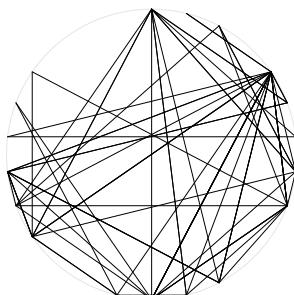
'Ein namhafter Wissenschaftler [...] hielt einmal einen öffentlichen Vortrag über Astronomie. Er schilderte, wie die Erde um die Sonne und die Sonne ihrerseits um den Mittelpunkt einer riesigen Ansammlung von Sternen kreist, die wir unsere Galaxis nennen. Als der Vortrag beendet war, stand hinten im Saal eine kleine alte Dame auf und erklärte: "Was Sie uns da erzählt haben, stimmt alles nicht. In Wirklichkeit ist die Welt eine flache Scheibe, die von einer Riesenschildkröte auf dem Rücken getragen wird." Mit einem überlegenen Lächeln hielt der Wissenschaftler ihr entgegen: "Und worauf steht die Schildkröte?" - "Sehr schlau, junger Mann", pariert die alte Dame. "Ich werd's Ihnen sagen: Da stehen lauter Schildkröten aufeinander." [...]'

Extract - Translated with U.E.D. below
A Brief History of Time (German),
Stephen Hawking

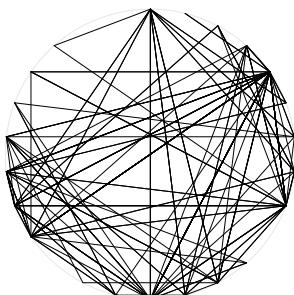
1 Word



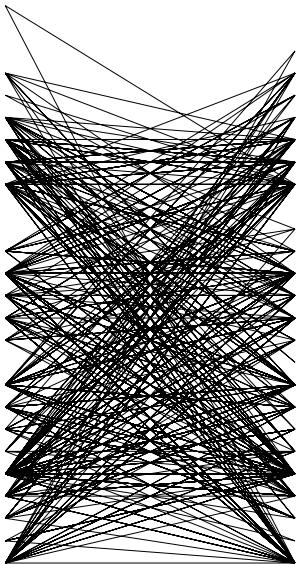
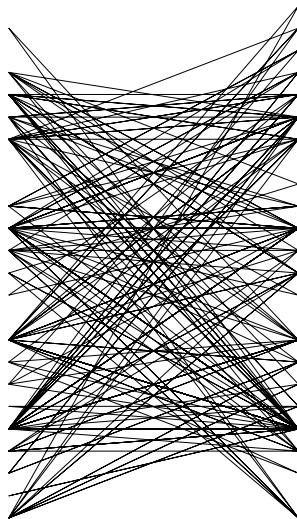
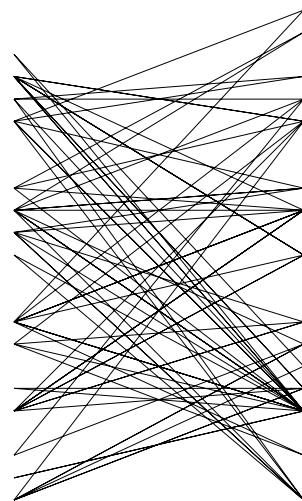
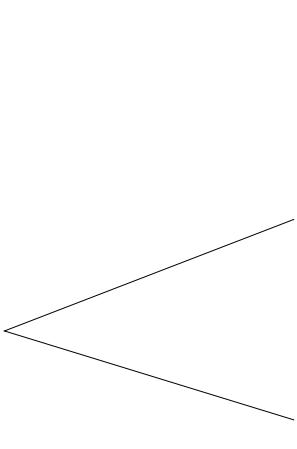
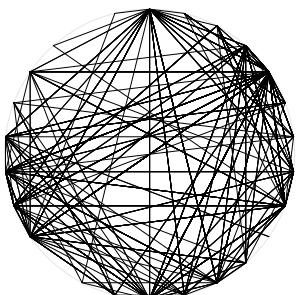
1 Sentence



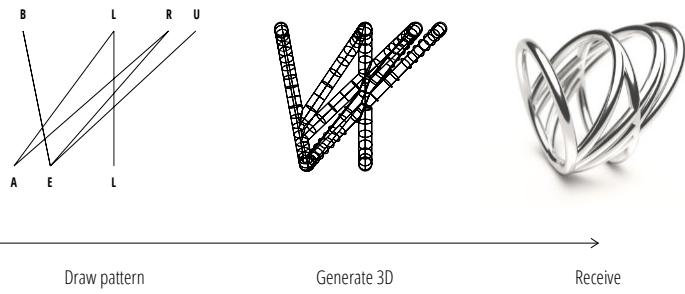
2 Sentences



1 Paragraph



UEBERALL



The idea to create jewellery and to reprogram the U.E.D. algorithm, evolved on the night before christmas in 2018.

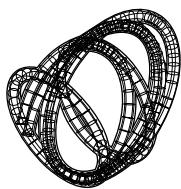
In May 2019 I founded the start-up YÜ with my dean's research assistant to focus on developing a high-tech and unprecedeted brand that enables customers to individualize and shape the whole product in the friendliest way possible.

I reprogrammed and extended the algorithm in Grasshopper and Python to make it accessable for a web plug-in. We designed a website and integrated the algorithm into our web system.

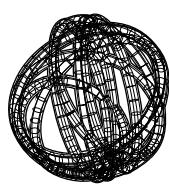
Today we are on the edge to go public with our entire work and to make a first real life test. Have a look at our website to see the latest progress.

www.yubyyou.com

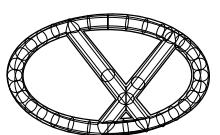
RING



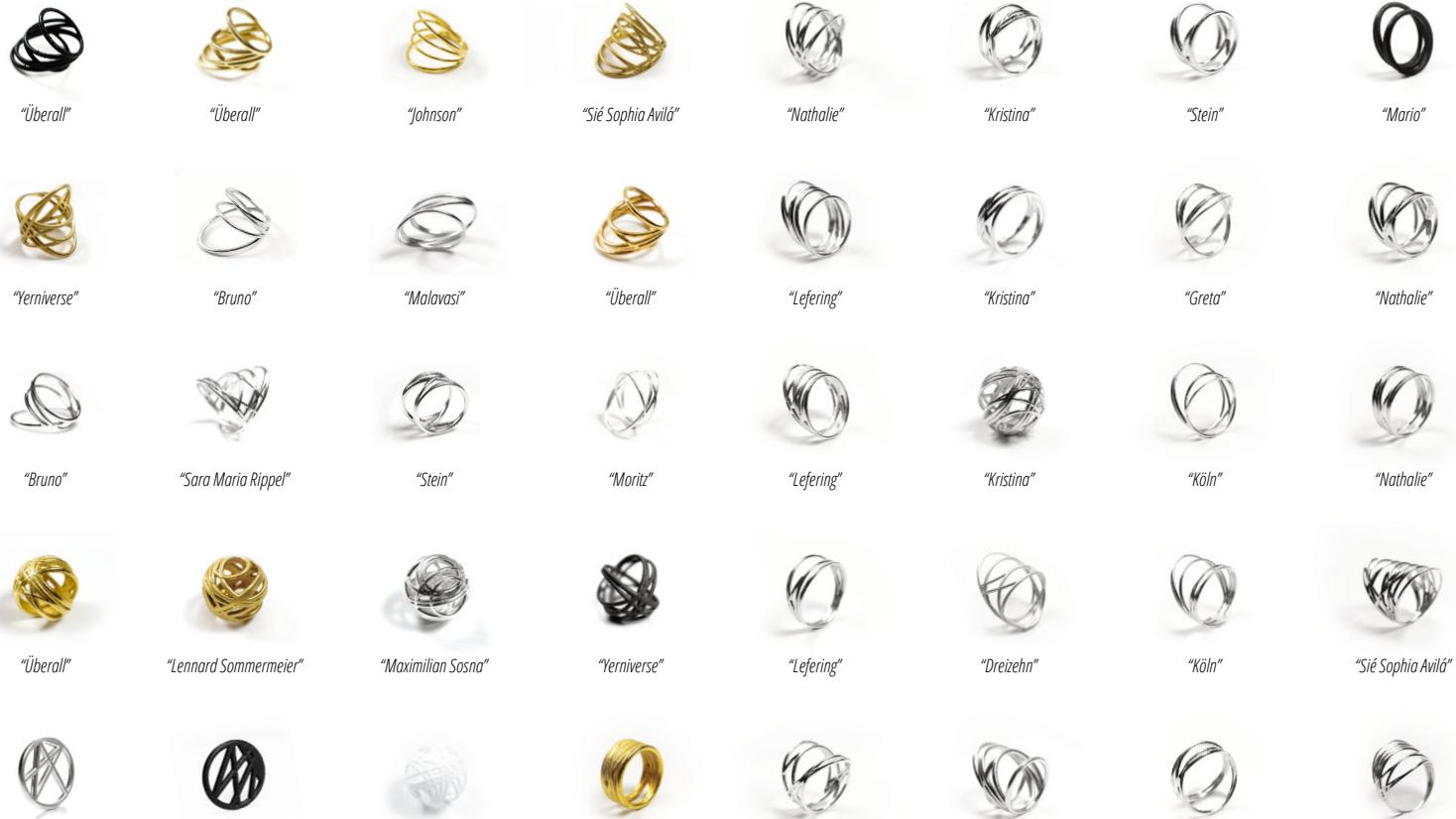
CHARM



AMULET



Down below you can see the 40 first jewellery pieces we made for customers in 2019. All of them have a unique and personal meaning at its core.

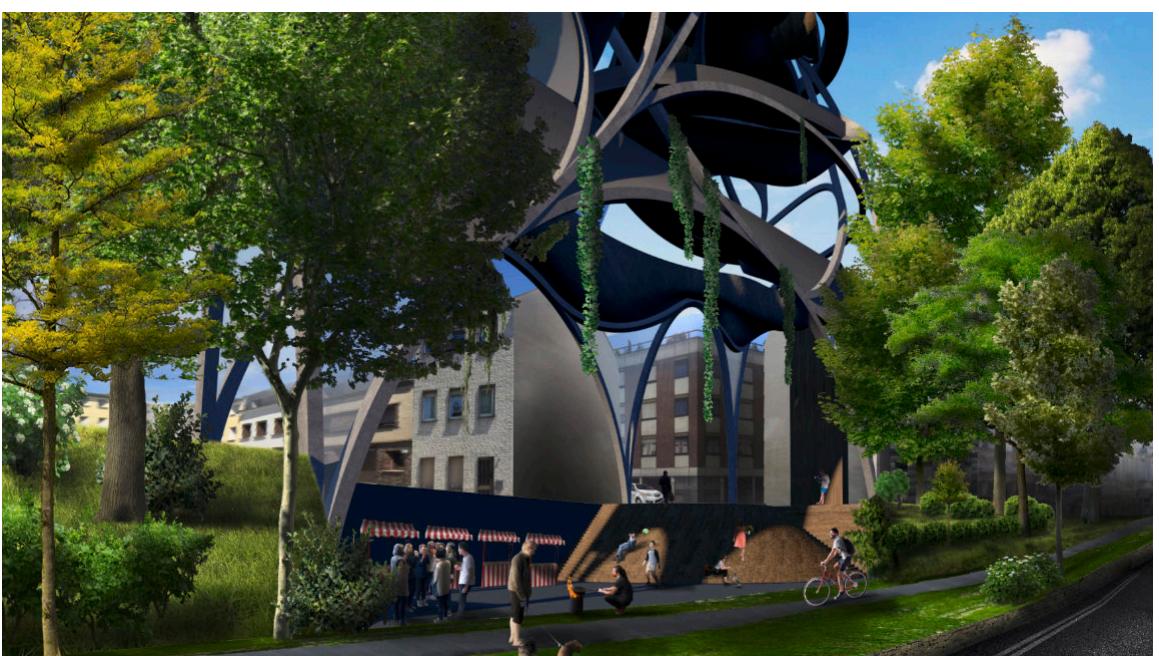


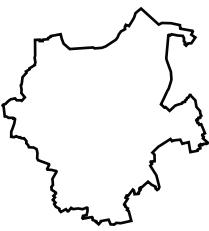


MÖ

Humans always had a deep connection to the natural environment.

In the 21st century climate change and other ecology related topics made mankind realize the importance of this extraordinary environment. Rather than making cities grey and dead, humans tried to integrate nature again into the big city life. Some concrete structured buildings grew like big trees in the heart of the cities. They achieved a direct coexistence between humans and nature. People were able to enjoy the sunset next to a couple of trees and flowers, 60m above the ground. A peaceful and calm place next to the fast and chaotic - a home to a few ecological pioneers.

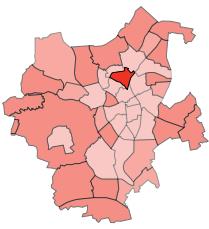




Mönchengladbach DE



Districts



Demography



Population density



Building height



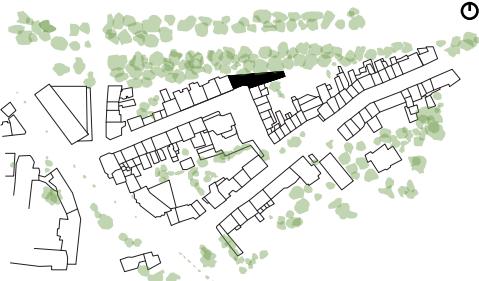
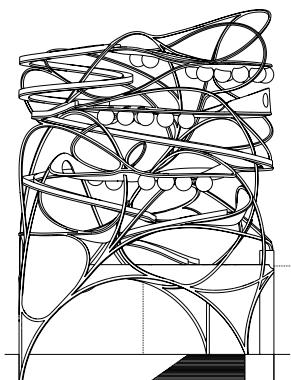
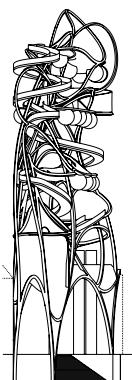
Property



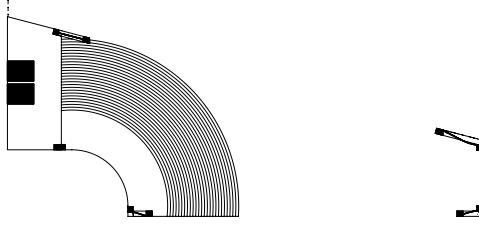
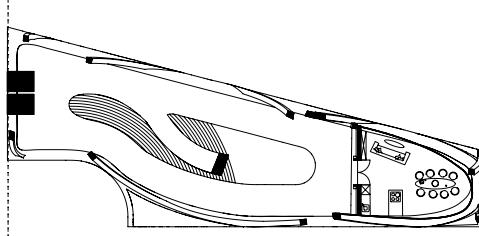
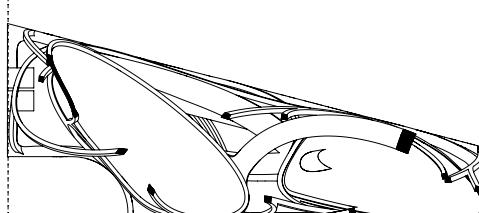
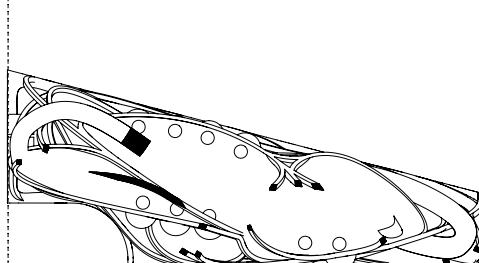
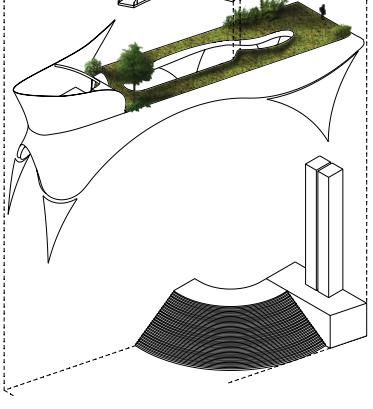
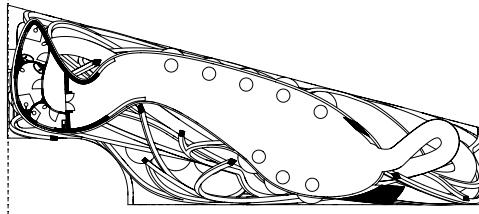
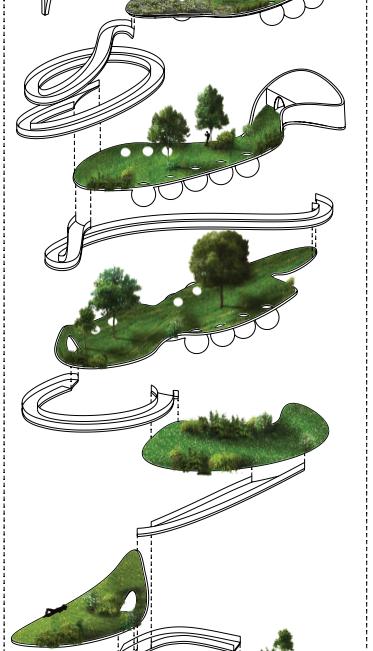
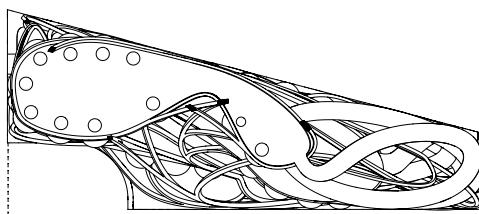
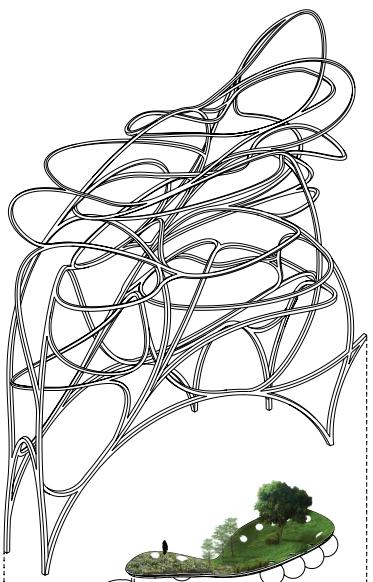
Street grid



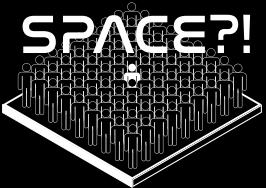
Parks



Studio_1_Year_MÖ



I I



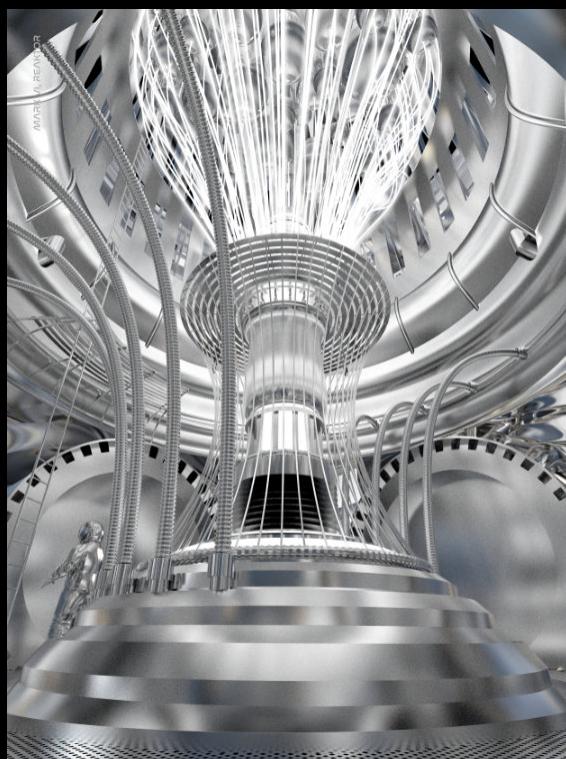
HABITAT B

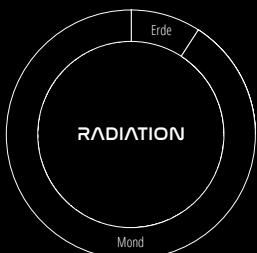
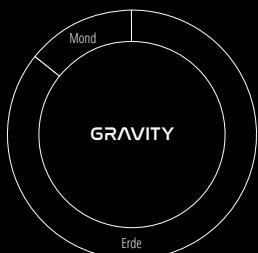
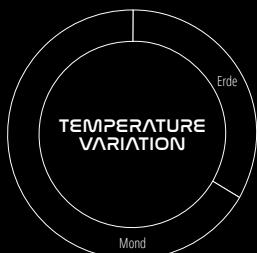
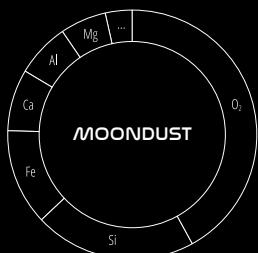
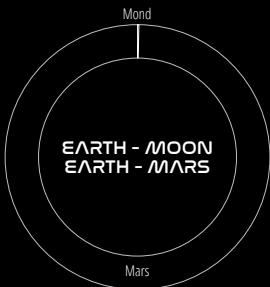
This is a studio project that I initiated myself to work on for half a year with my professor.

Main purpose of this extraordinary field of study was to explore architecture beyond earth. After a lot of research I focused on our moon - earth's closest neighbour. I discovered the R.O.X.Y. reactor from Airbus, which can turn moon dust into a variety of resources without a lot of energy. I designed two robots for two different phases on the moon. A small one to replicate itself and to grow manufacturing time. And a big one, as you can see on the right, to produce enough resources for space colonization.

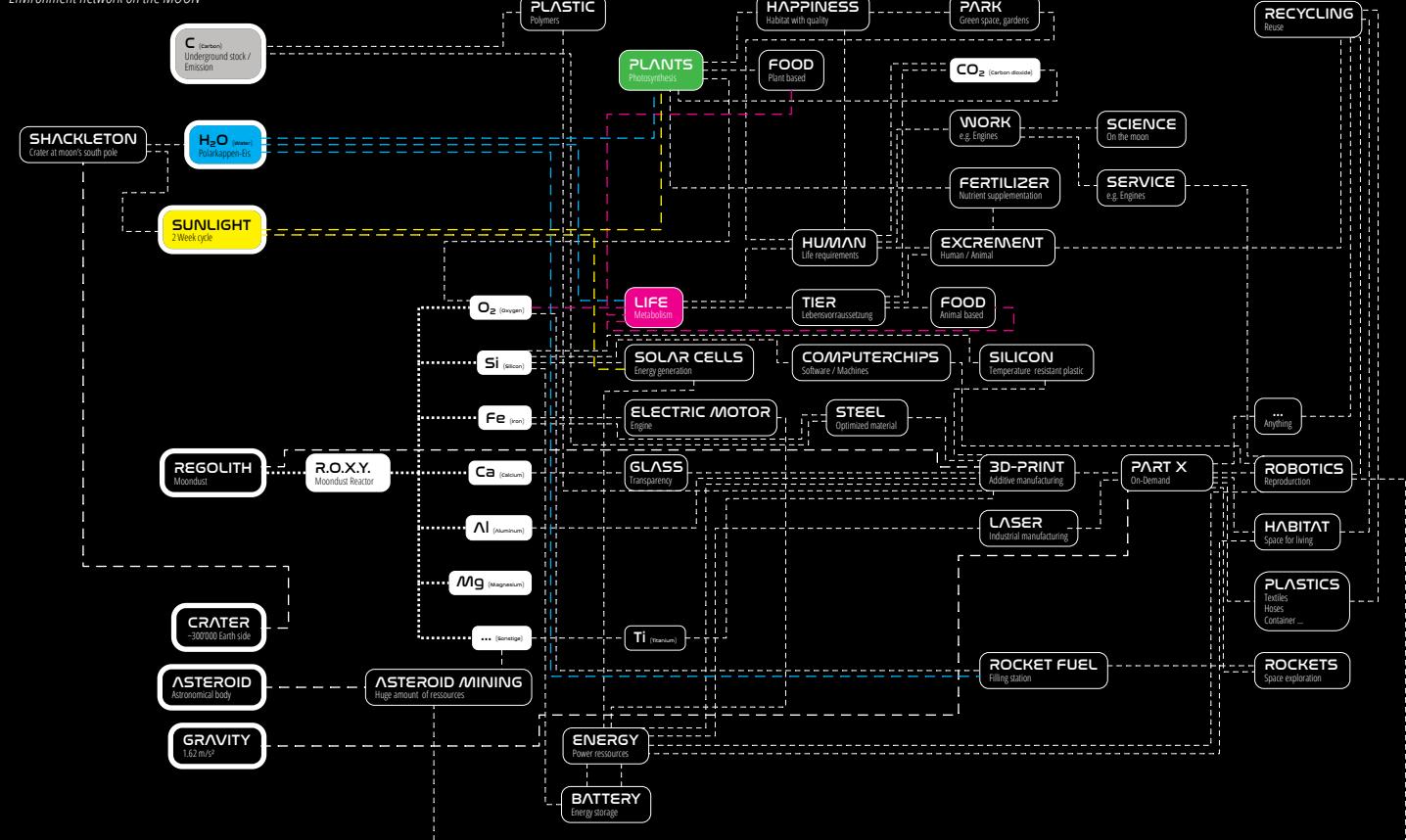
With the following link you can enter my VR art exhibition related to the space architecture project.

<https://hub.link/P4mDdt2>

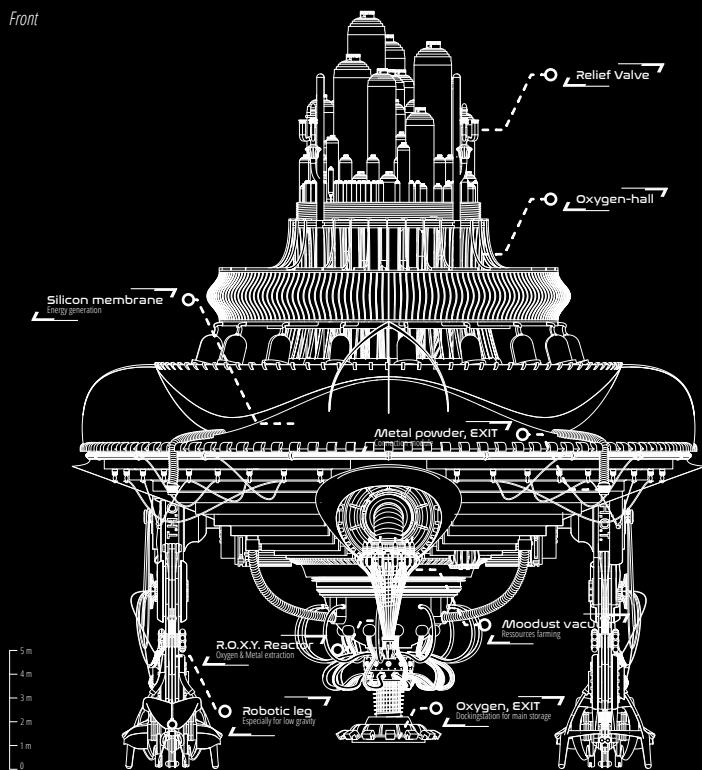




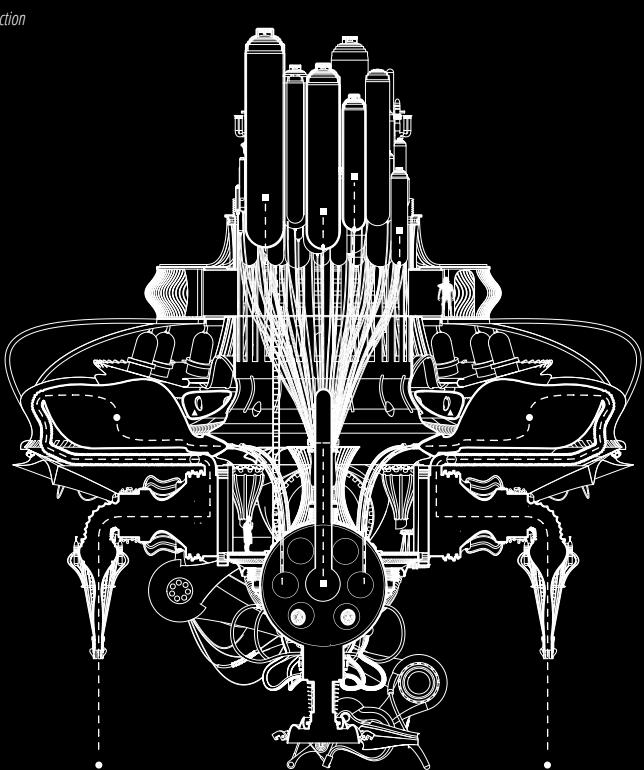
Environment network on the MOON



Front



Section



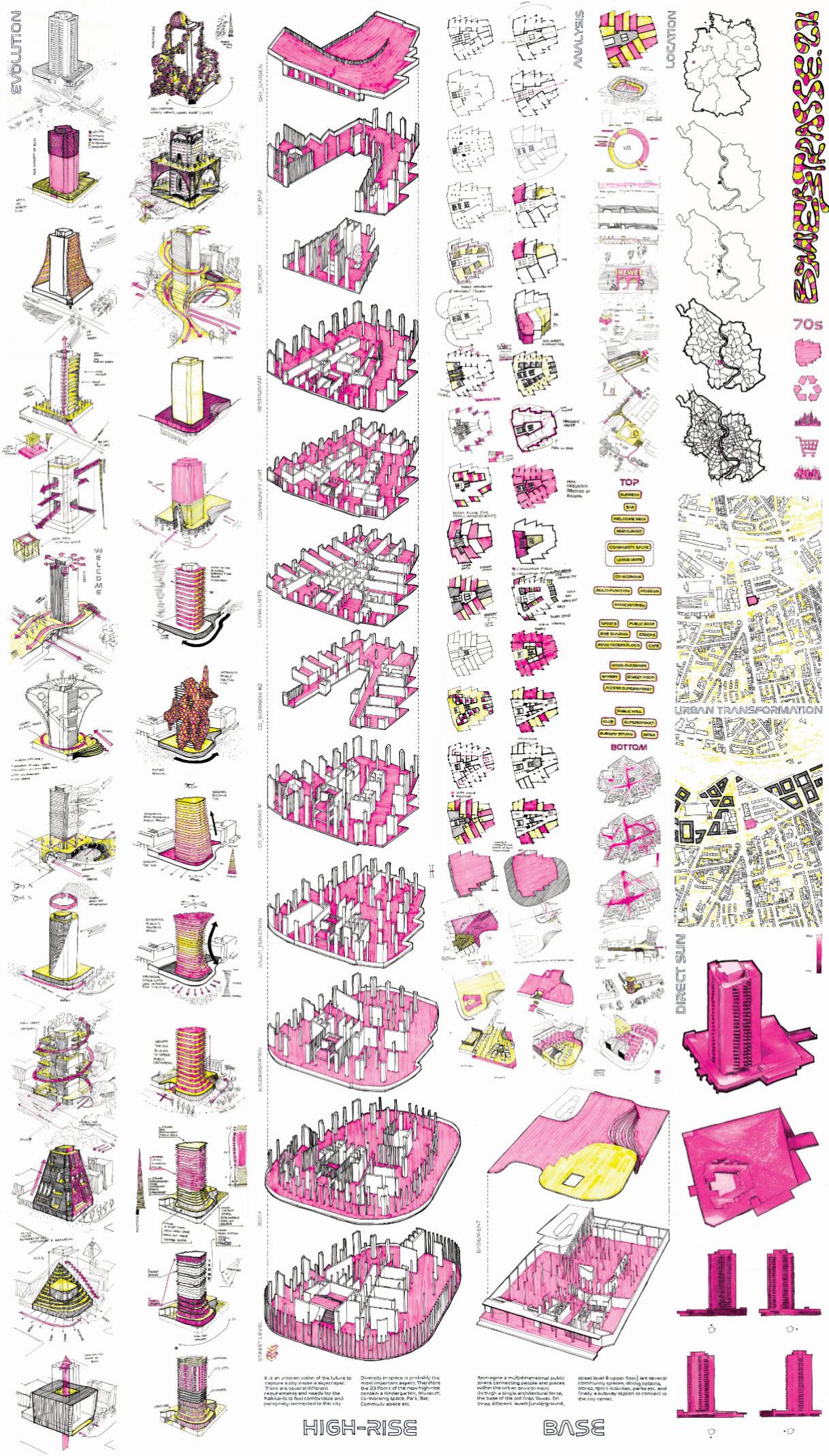


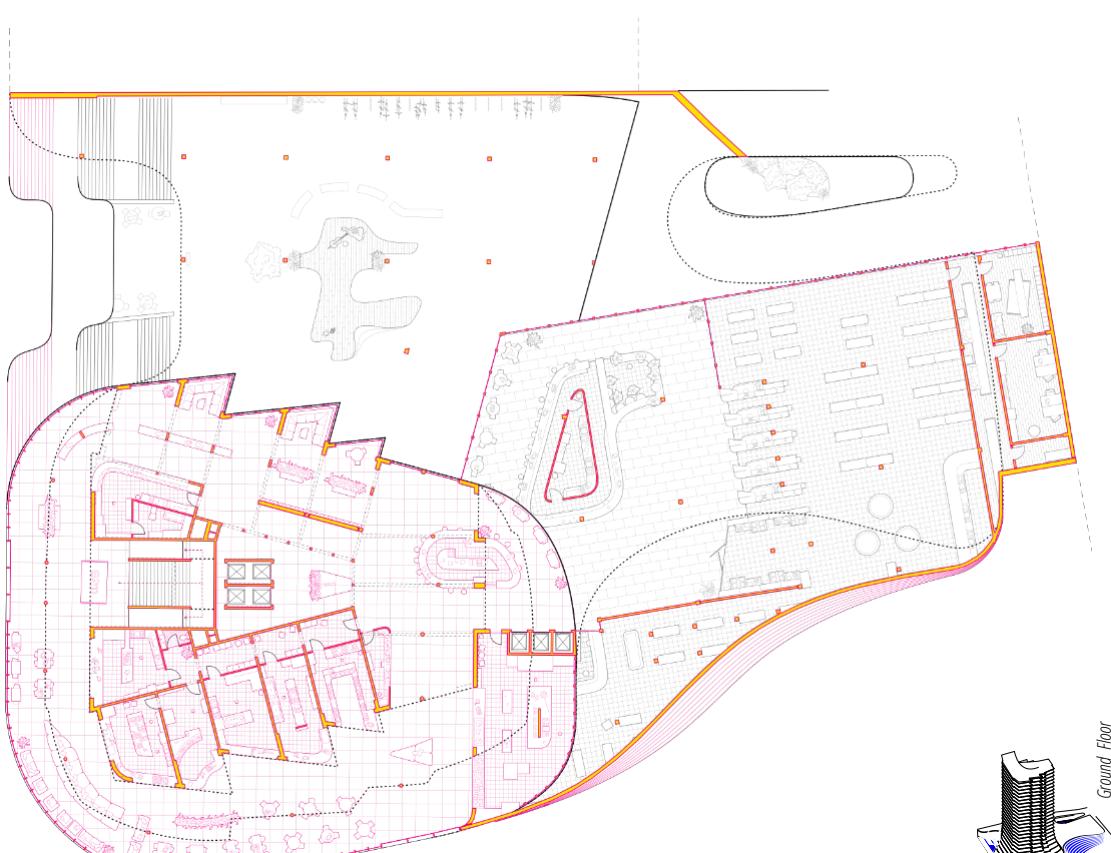
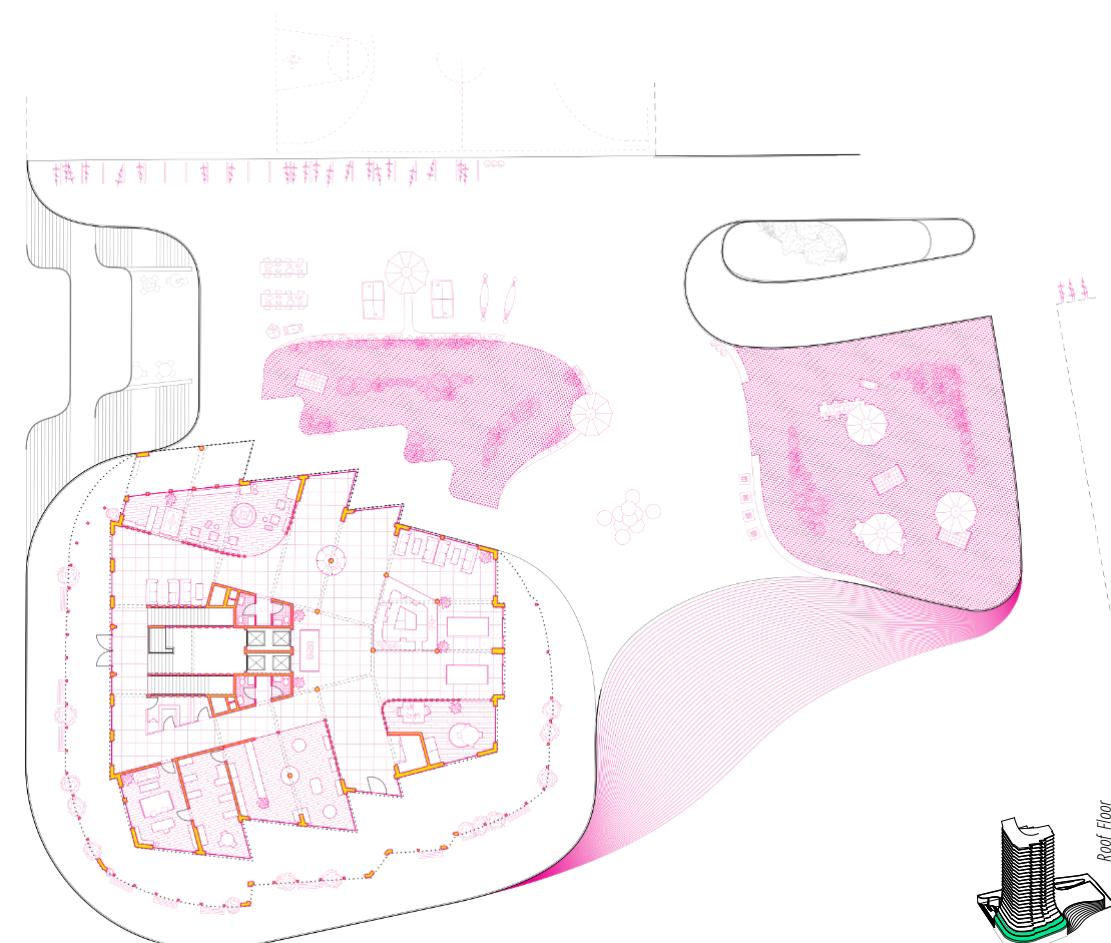
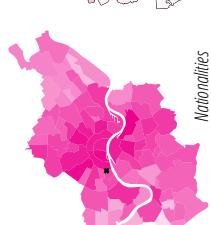
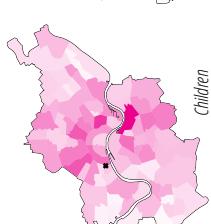
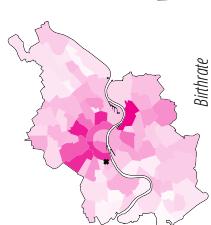
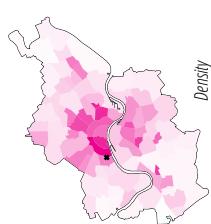
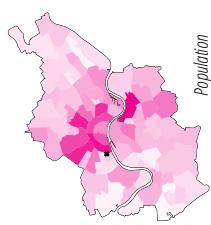
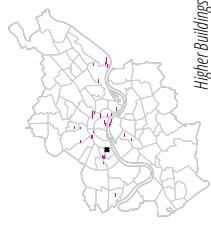
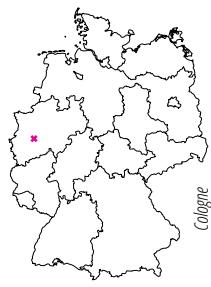
B211

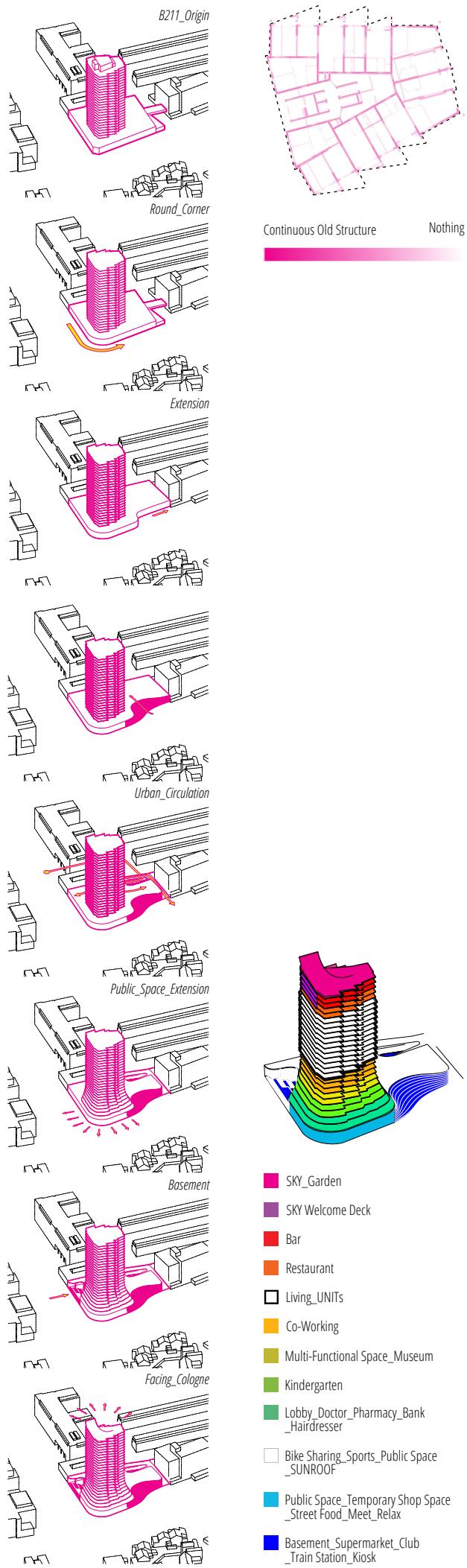
The vertical city, a transgenerational utopia full of fantasy and harmony. A place characterized by freedom and versatility that would not be possible for an average apartment block. Light and dark, quiet and loud, green and gray. A space that metamorphoses through its high density and contrasts into a colorful place shaped by its inhabitants. From a basement with subway station, club and grocery, to a sky garden with sundeck and café. The vertical city as an architectural manifesto for a social future full of diversity and acceptance. A place of community and togetherness.

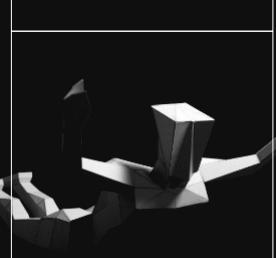
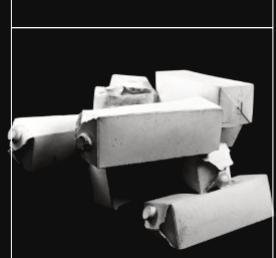
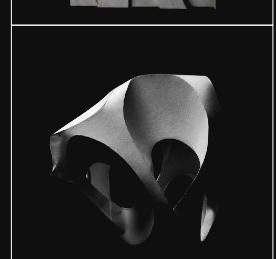
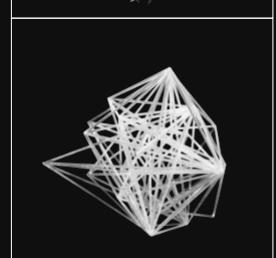
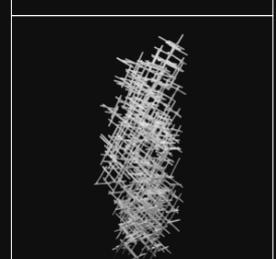
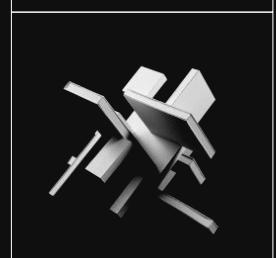
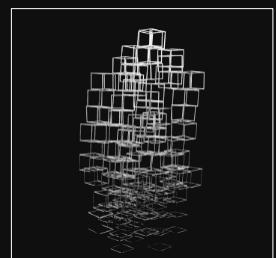
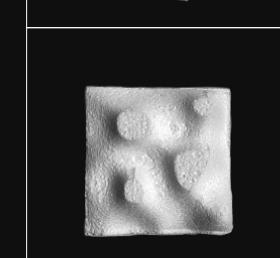
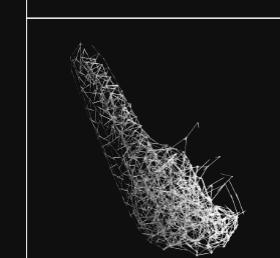
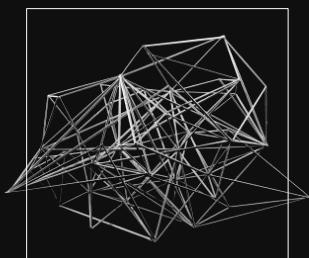
This Bachelor Thesis project marked the ending of my undergraduate degree in architecture and interior design in Düsseldorf, Germany in 2021.

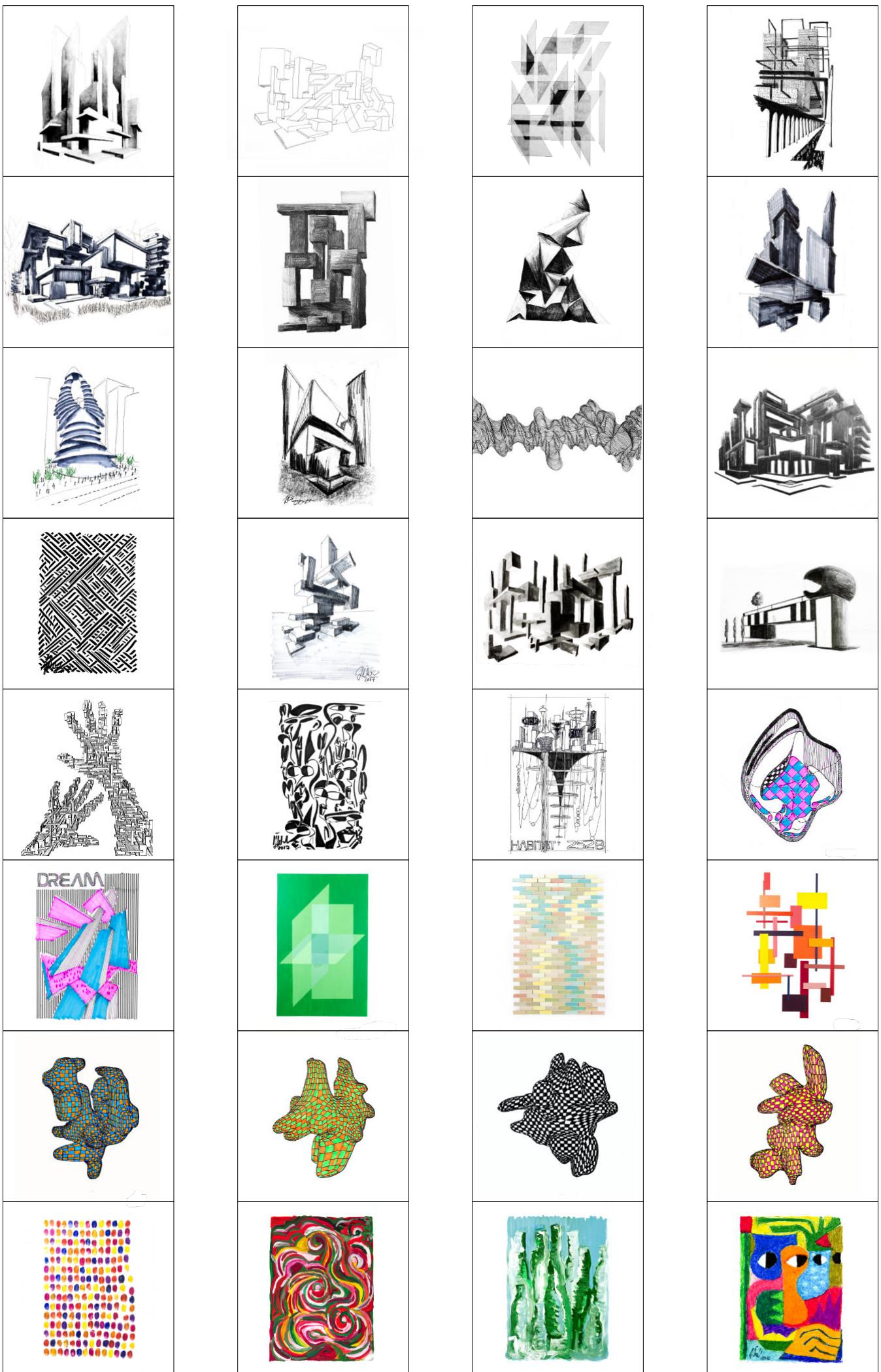












A handwritten signature in black ink, appearing to read "Julius Überall".