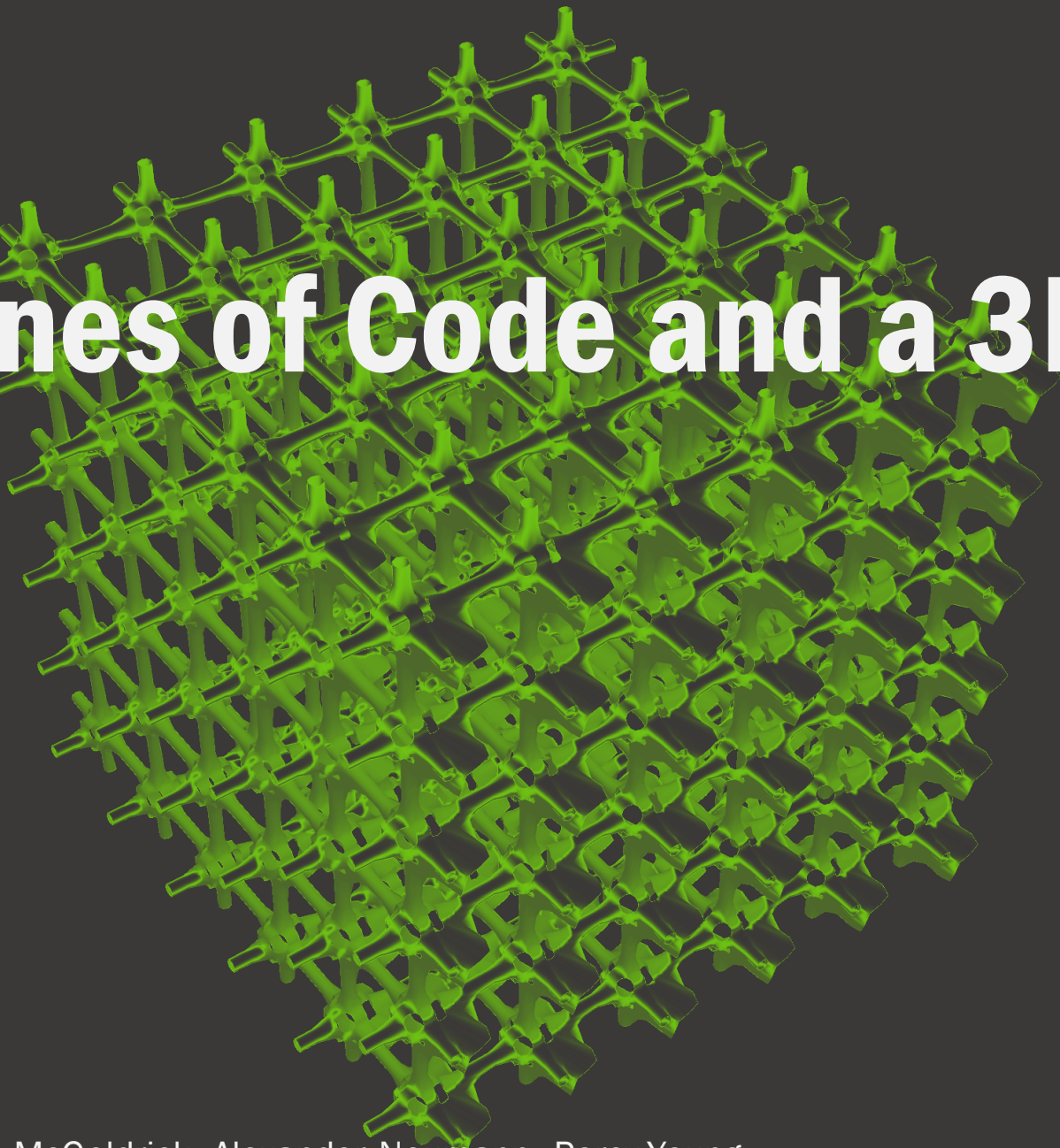
200 Lines of Code and a 3D Print



ARCH 696

Shajay Bhooshan

Caleb Derksen, Ala Ebdalla, Sinead McGoldrick, Alexander Neumann, Percy Yeung

OBJECTIVES

1) Defining functionally graded materials (with examples).

2) Creating and illustrating the workflow between Maya-code(c++)-3dp (chitubox) to create functionally graded materials.

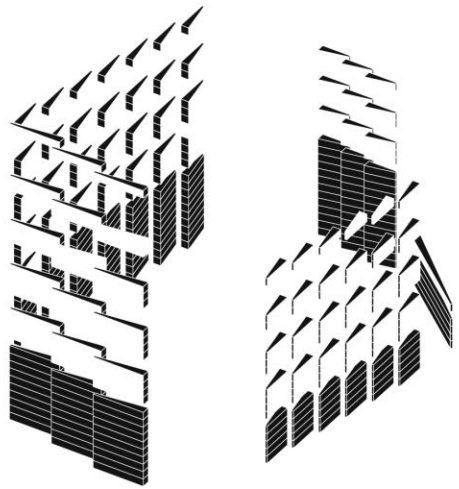
3) Creation of a catalogue of different components and their aggregation study in maya

4) Creation of a catalogue of corresponding coded aggregations (c++)

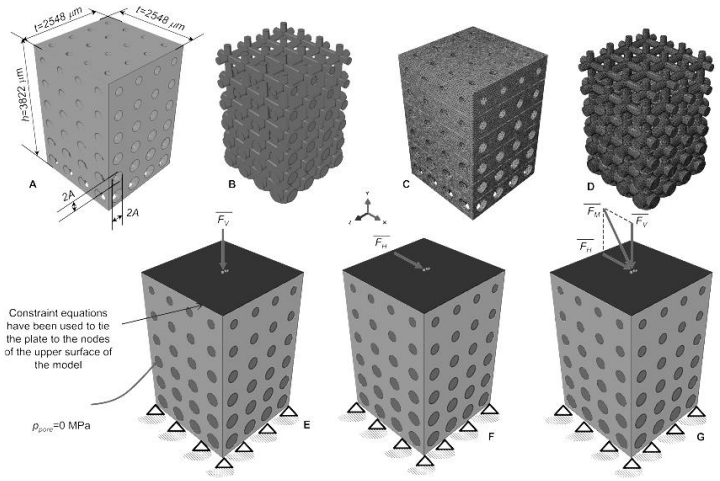
5) Exploration of a corresponding catalogue of 3D printing statistics (estimated print time, estimated material consumption, estimated supports, etc.)

WTF is an FGM

Our team defined a functionally graded material as a parametric logic structure, based in a voxel form, that can infinitely aggregated and modified to create unique forms. Small tweaks to vertex placements through scaling, transformation and rotation can result in massive, dramatic shifts when repeated.



Kit of Parts Architectural Logic



MAYA+CODE+MAYA+SLICER RELATIONSHIP

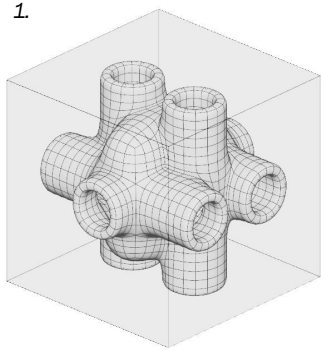
BRAIN+MAYA+CODE+MAYA+MESH REPAIR+SLICER+PRINTER

Imagination - geometry – aggregate - combine and smooth - gcode - solid

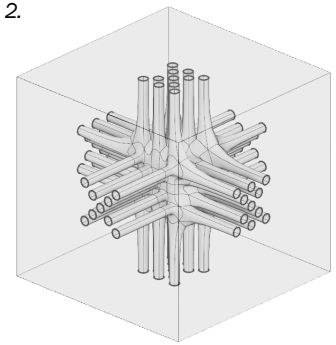
Shape variations	Transformations	Merging
Connection variations	Rotations	Subdivision
Density variations	Scaling	Boolean

Happy Accidents

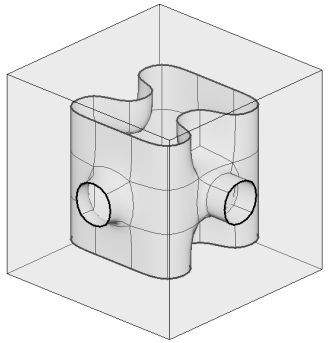
1.



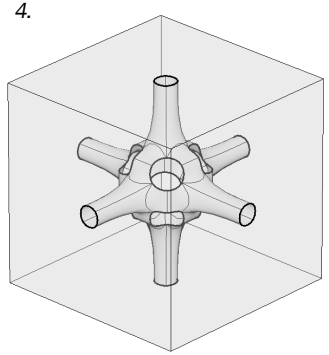
2.



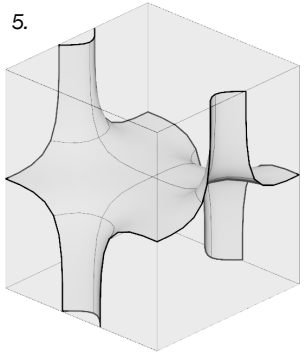
3.



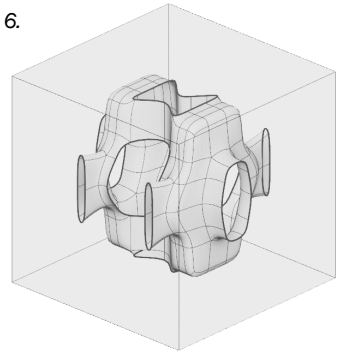
4.



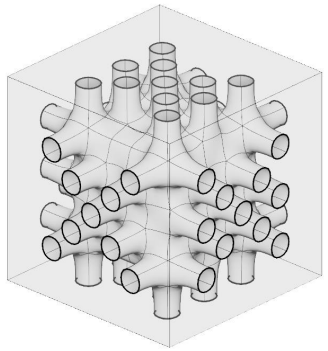
5.



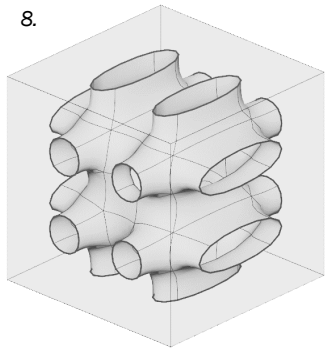
6.

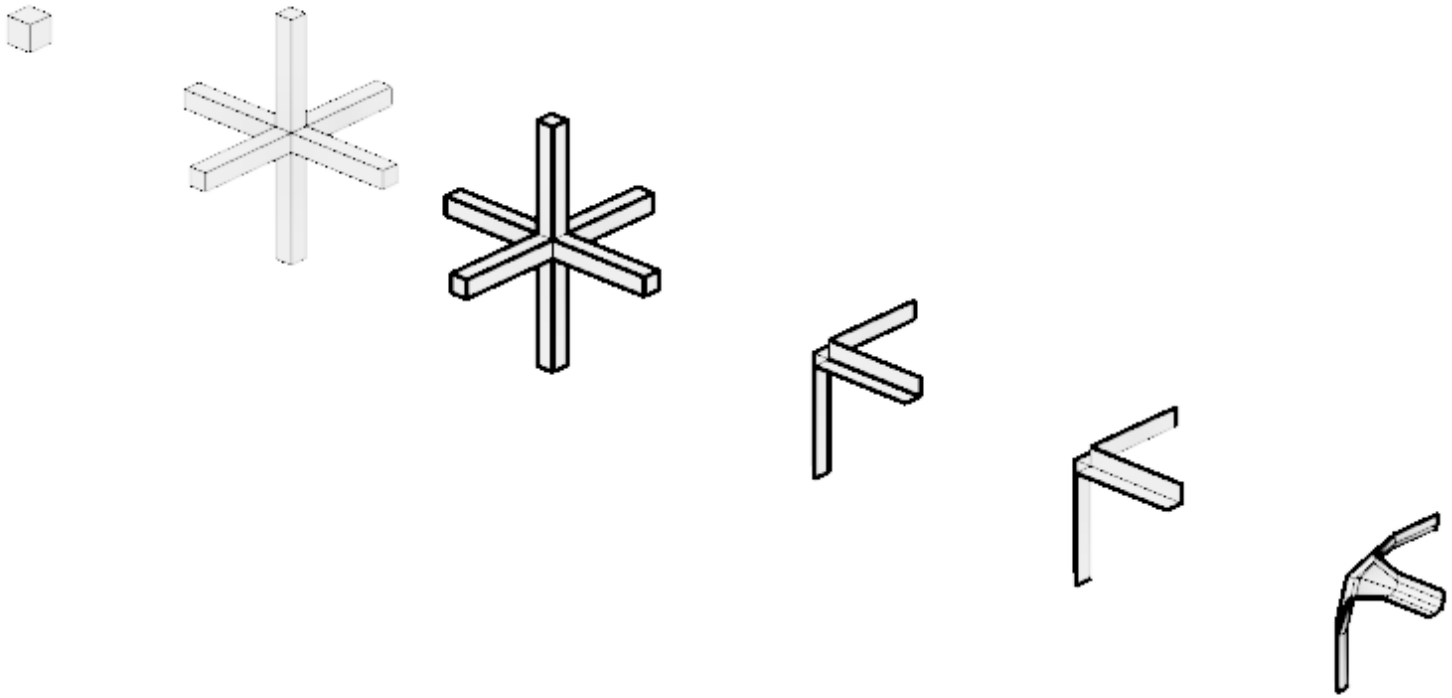


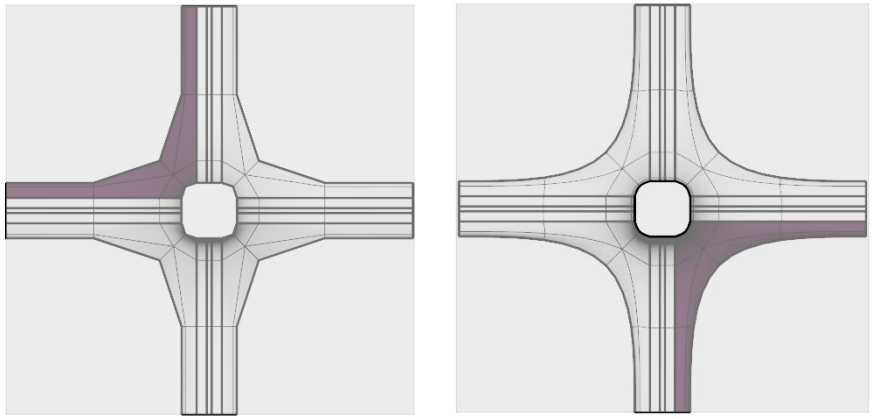
7.

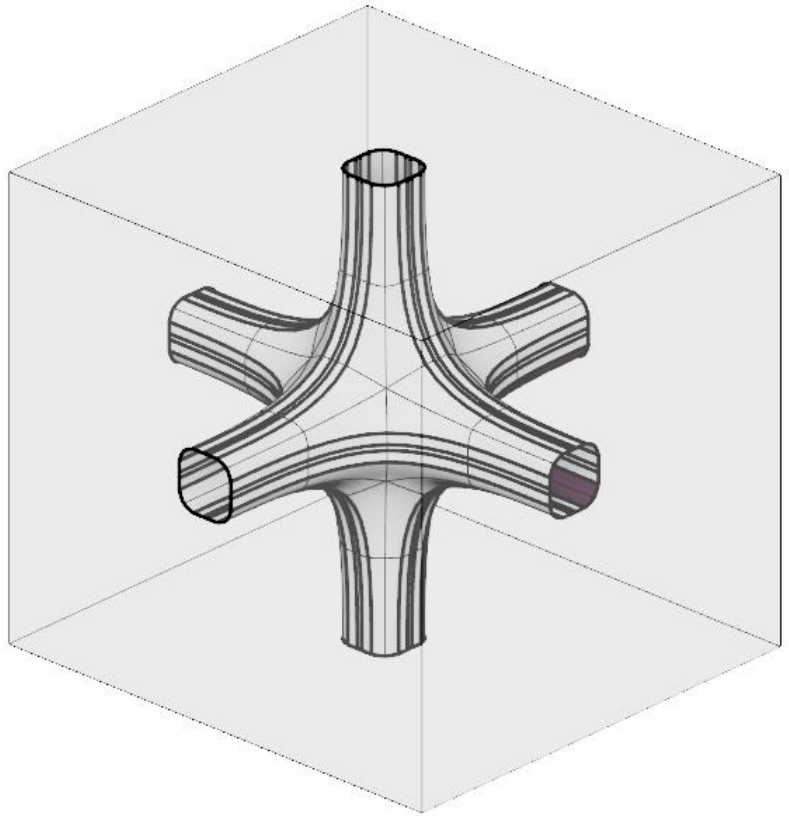


8.

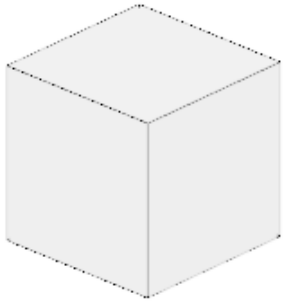


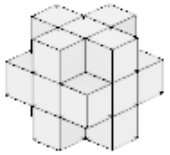


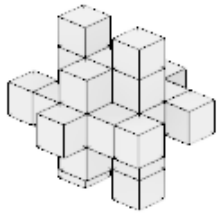


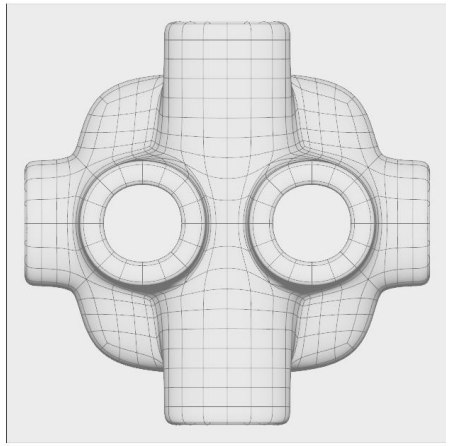


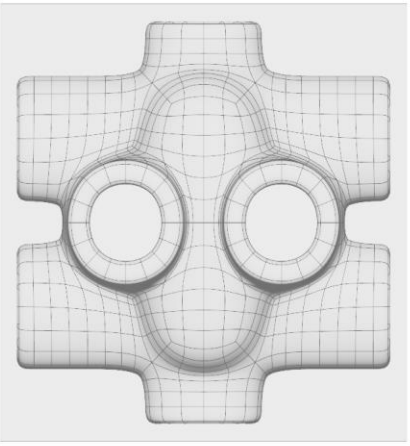
INTRO TO MAYA

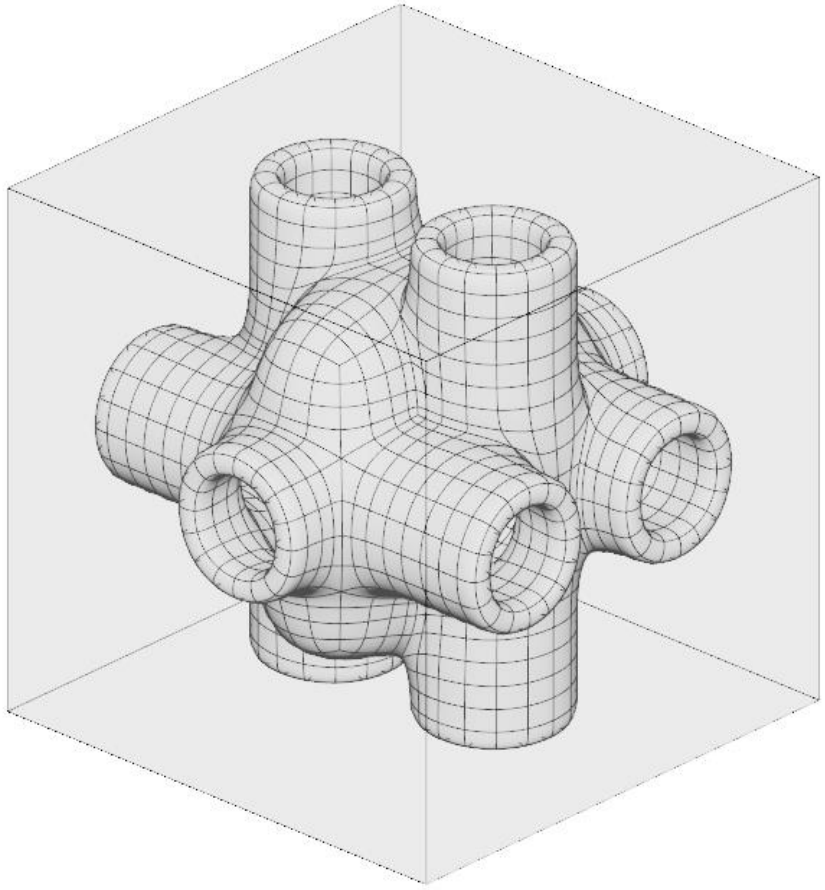










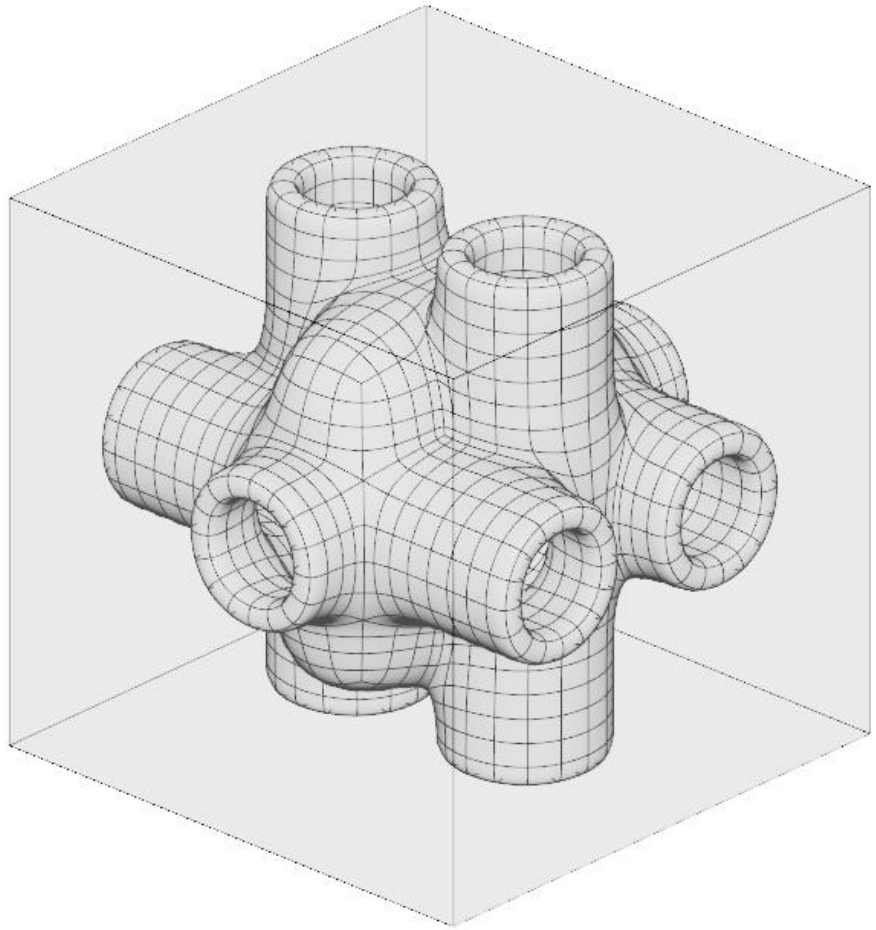


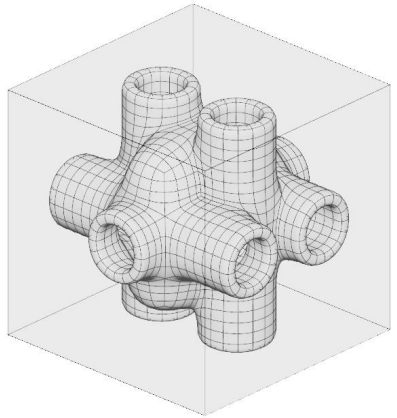
GEOMETRIC EXPLORATIONS [1]

CODED AGGREGATION [1]

Creation of a catalogue of corresponding coded aggregations (c++)

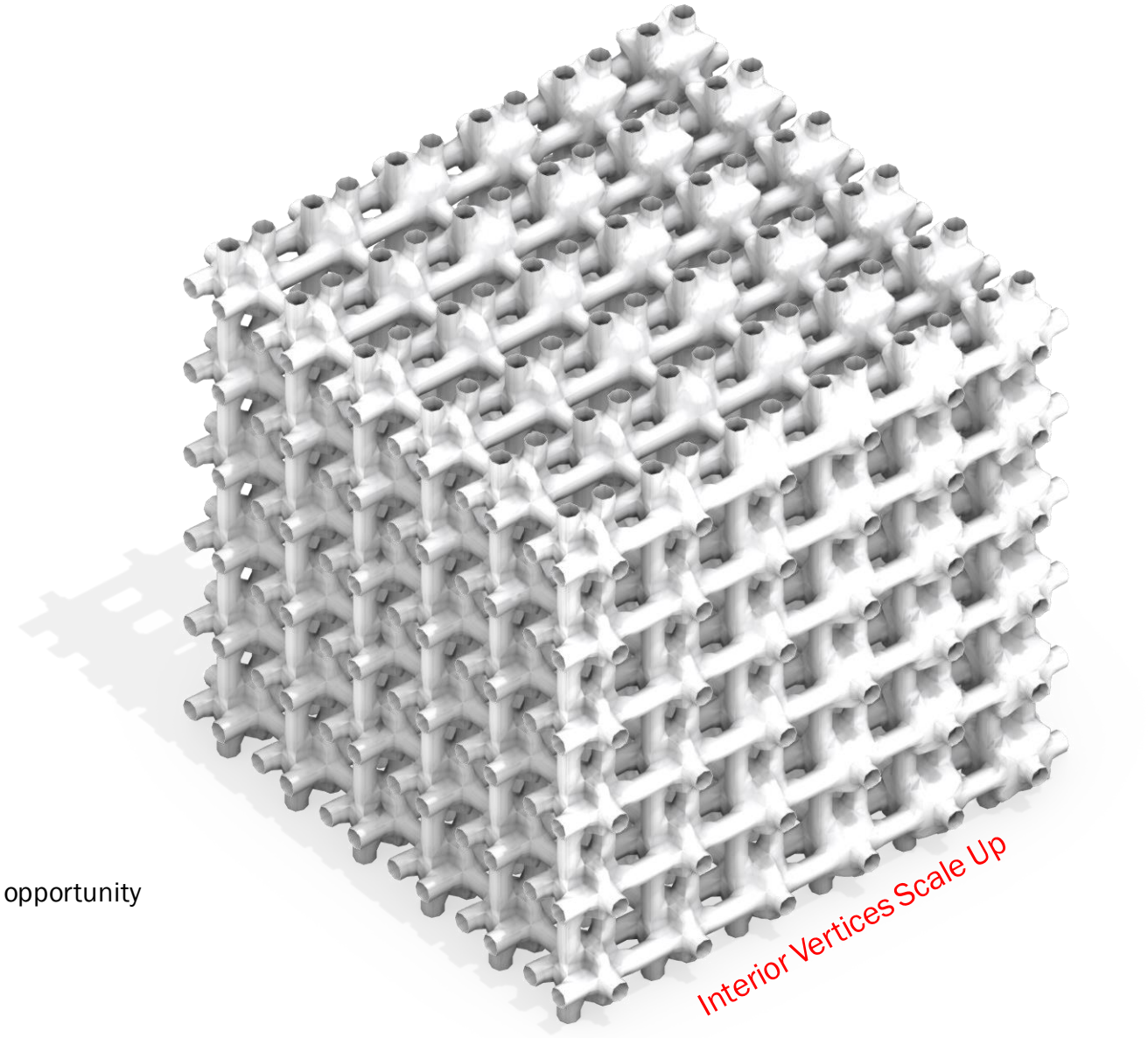
```
...cuments\GitHub\Alice2022_UoC\src\alice\VoxelFactory.h                    3
            trasnfromed
 99         int src[] = { 54, 48, 52, 49, 51, 53, 55, 50 };// cross 3d //
100         int n = sizeof(src) / sizeof(src[0]);
101         zIntArray ids(src, src + n);
102
103         // construct a gradient parameter
104         zVector u, v, w, c;
105         c = cen;
106         float maxL = diagLength;
107         float mult = ofMap(id, 0, 215, -.5, 5); ;// 4 - ofMap(cen.length
              (), 0, maxL, 0.1, 4);
108
109         //cout << mult << "--mult" << endl;
110
111         //construct the transformation matrix
112         zTransform TM;
113         TM.setIdentity();
114         u = zVector(1, 0, 0);
115         v = zVector(0, 1, 0);
116         w = zVector(0, 0, 1);
117         u.normalize(); v.normalize(); w.normalize();
118         u *= mult; v *= mult; w *= mult;
119
```

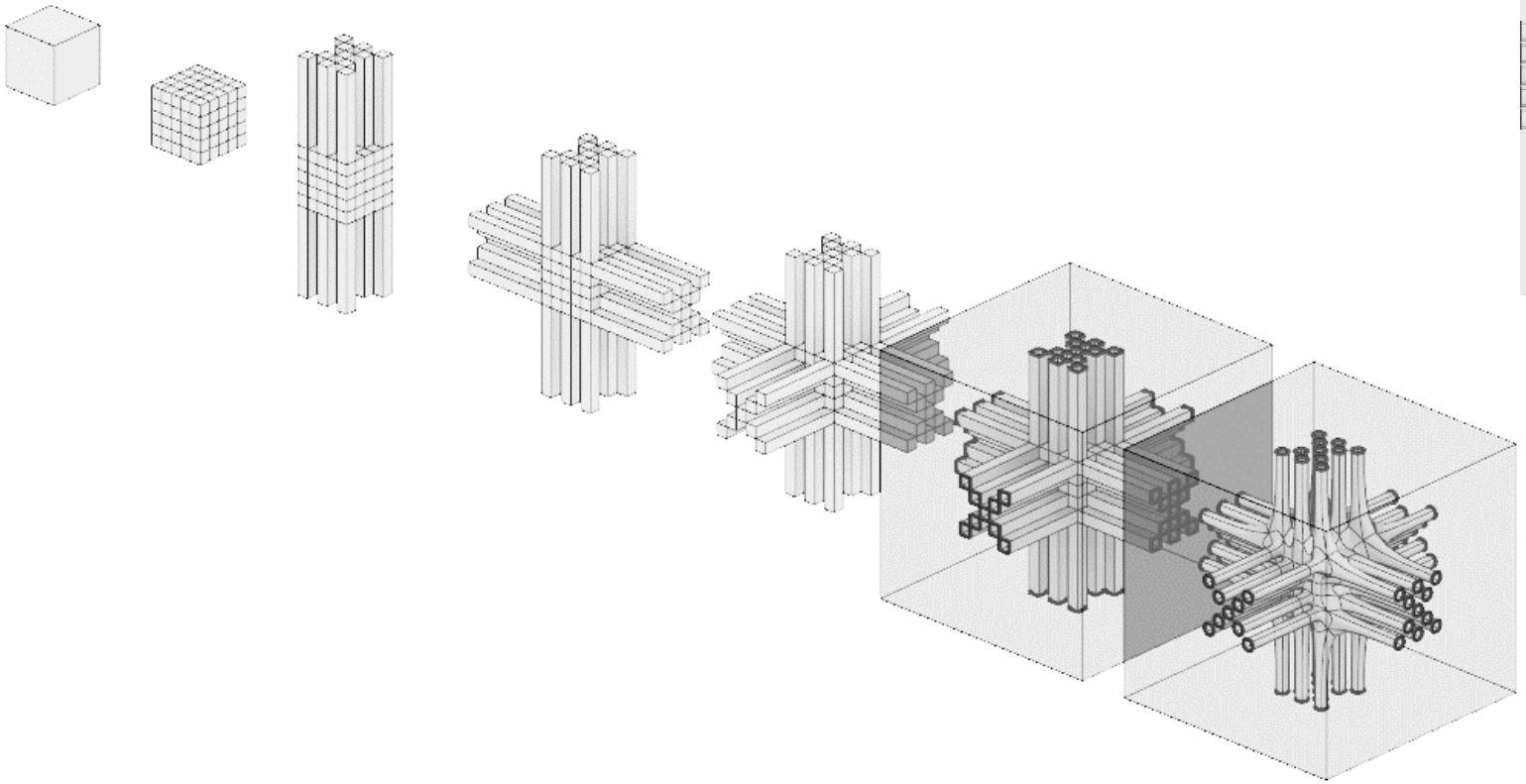


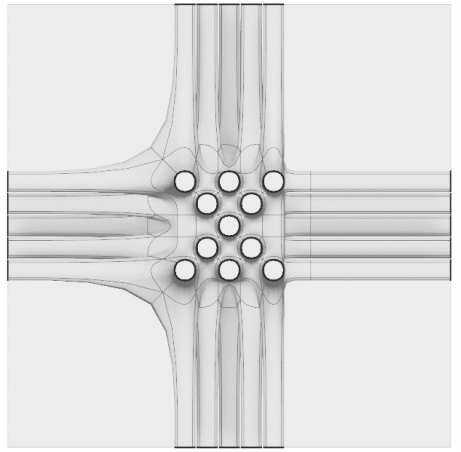


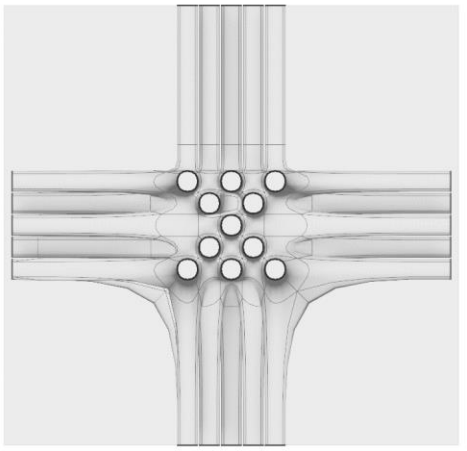
CONNECTION NODE BASE

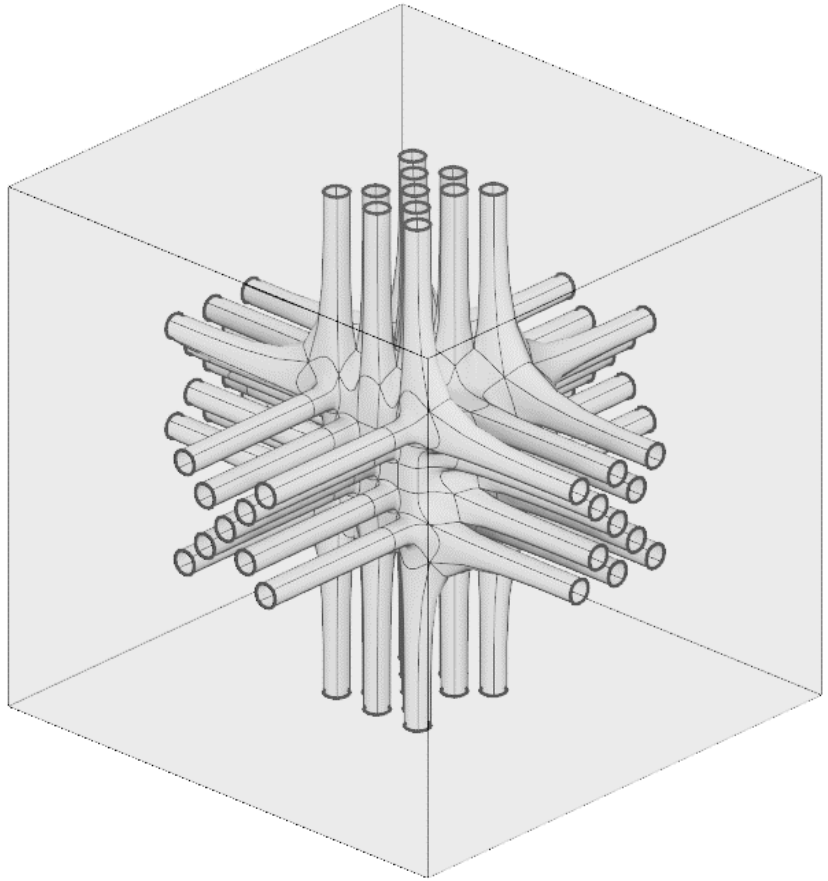
Each side of model offers unique aggregation opportunity









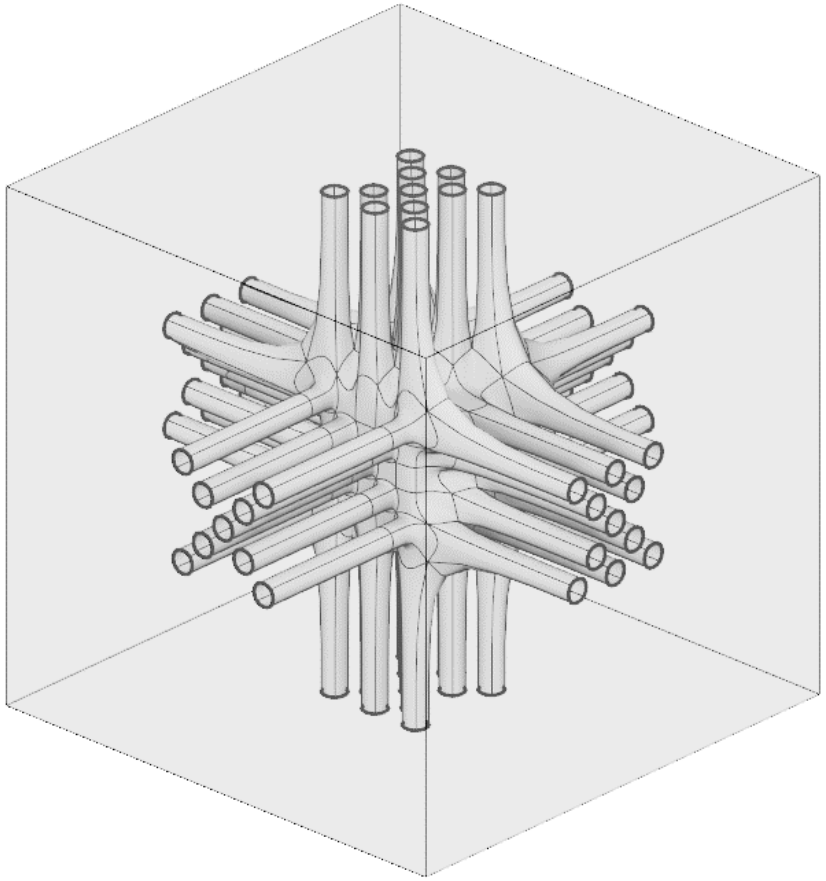


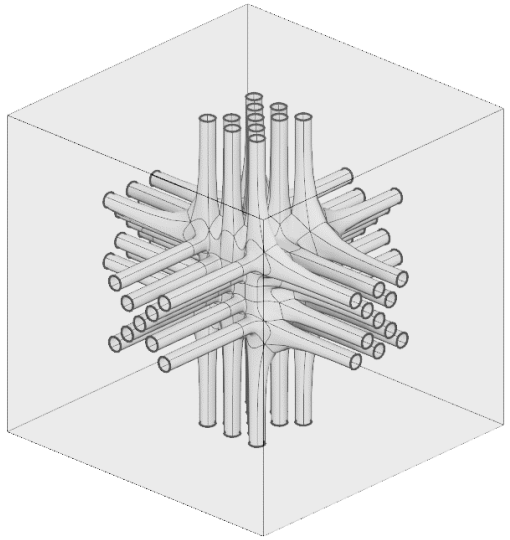
GEOMETRIC EXPLORATIONS [2]

CODED AGGREGATION [2]

Creation of a catalogue of corresponding coded aggregations (c++)

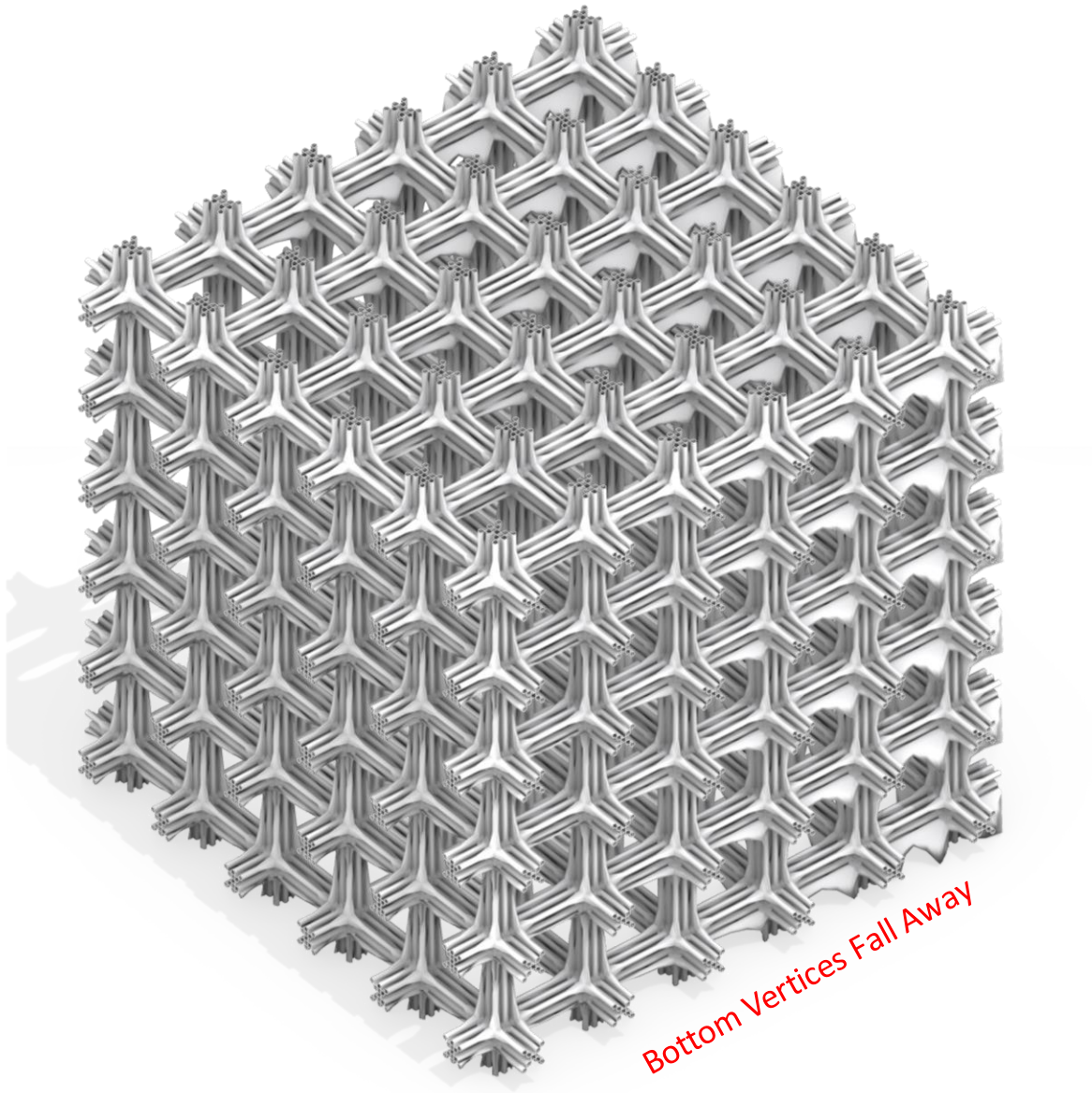
```
...e\Desktop\Alice2022_UoC-main\src\alice\VoxelFactory.h                    3
          trasnfromed
 99       int src[] = { 0, 1, 2, 3 };// cross 3d
100       int n = sizeof(src) / sizeof(src[0]);
101       zIntArray ids(src, src + n);
102
103       // construct a gradient parameter
104       zVector u, v, w, c;
105       c = cen;
106       float maxL = diagLength;
107       float mult = ofMap(id, 0, 215, -0.5, 4); ;// 4 - ofMap(cen.length ↵
            (), 0, maxL, 0.1, 4);
108
109       //cout << mult << "--mult" << endl;
110
111       //construct the transformation matrix
112       zTransform TM;
113       TM.setIdentity();
114       u = zVector(1, 0, 0);
115       v = zVector(0, 1, 0);
116       w = zVector(0, 0, 1);
117       u.normalize(); v.normalize(); w.normalize();
118       u *= mult; v *= mult; w *= mult;
```

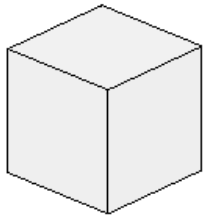


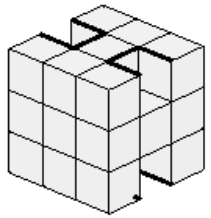


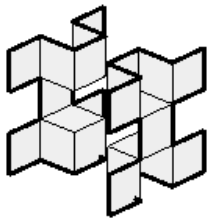
CONNECTION NODE BASE

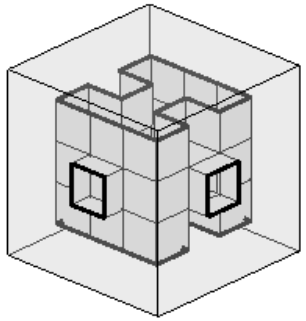
Geometry is similar on each side

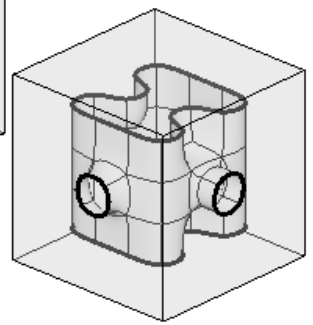


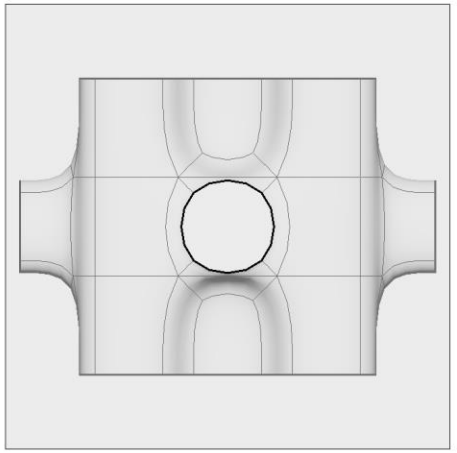


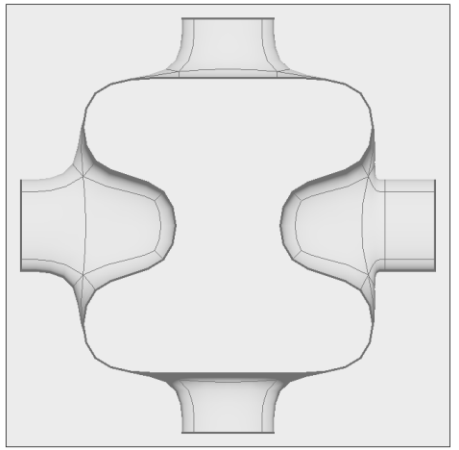


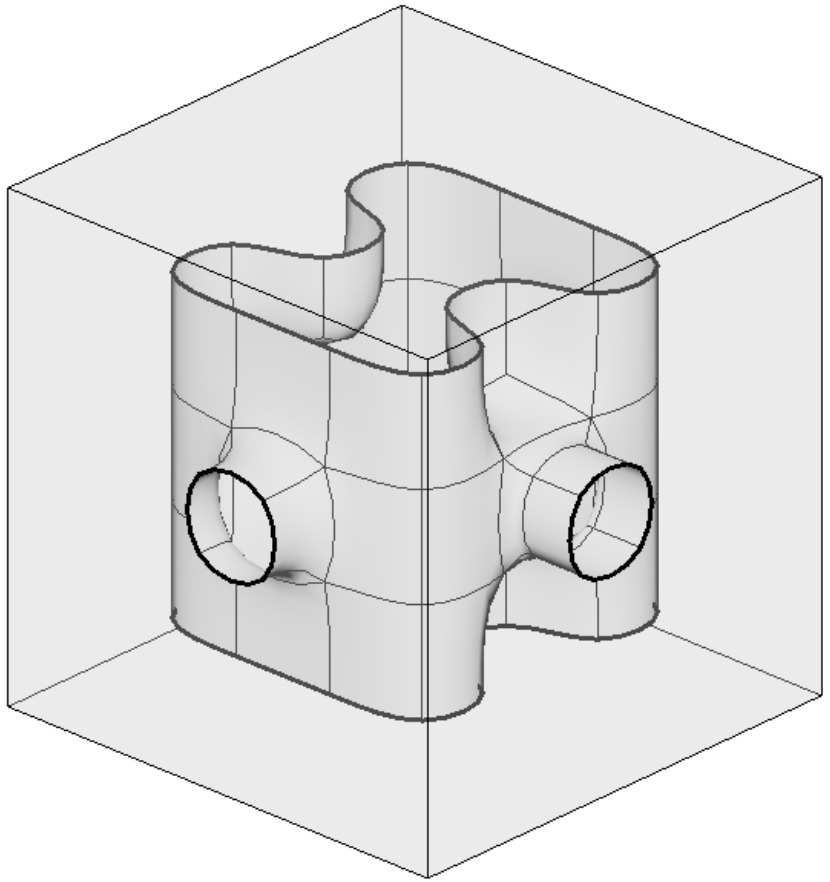










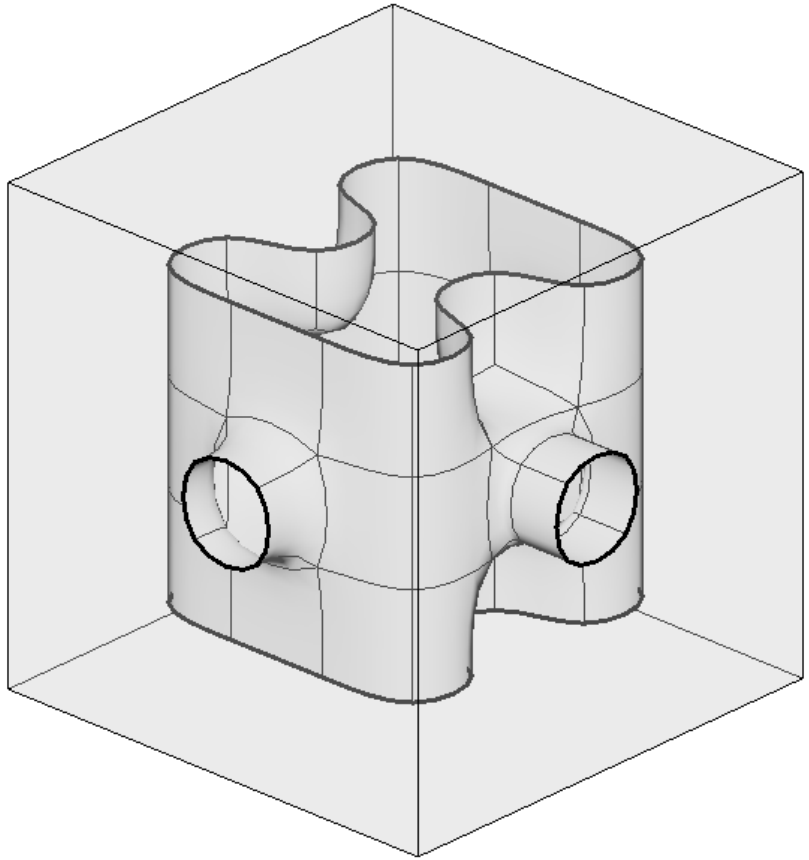


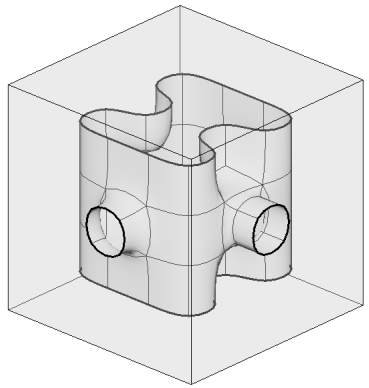
GEOMETRIC EXPLORATIONS [3]

CODED AGGREGATION [3]

Creation of a catalogue of corresponding coded aggregations (c++)

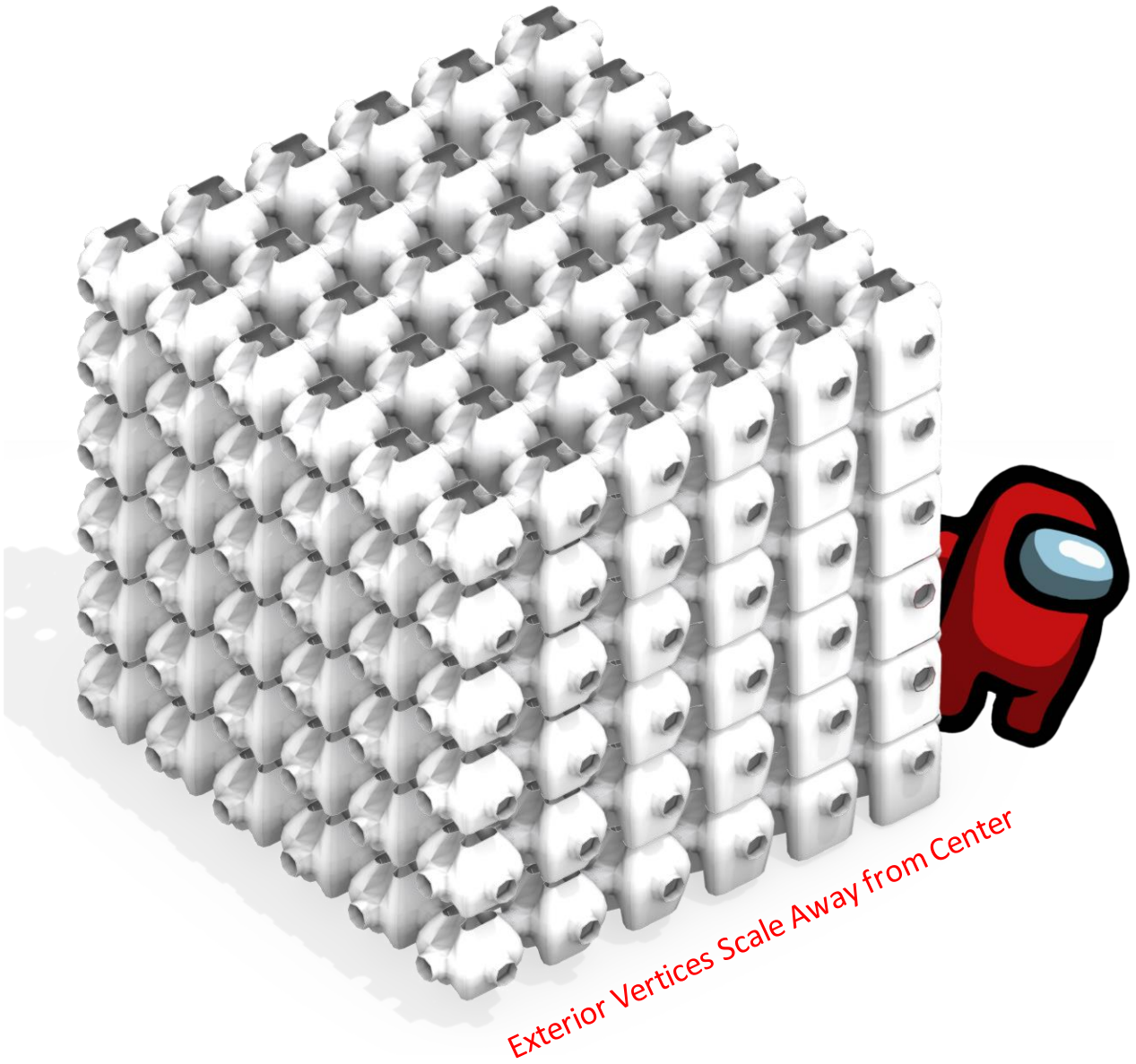
```
...e\Desktop\Alice2022_UoC-main\src\alice\VoxelFactory.h                    3
            trasnfromed
 99         int src[] = { 21, 18, 24, 27 };// cross 3d
100         int n = sizeof(src) / sizeof(src[0]);
101         zIntArray ids(src, src + n);
102
103         // construct a gradient parameter
104         zVector u, v, w, c;
105         c = cen;
106         float maxL = diagLength;
107         float mult = ofMap(id, 0, 215, -0.5, 2); ;// 4 - ofMap(cen.length ↵
              (), 0, maxL, 0.1, 4);
108
109         //cout << mult << "--mult" << endl;
110
111         //construct the transformation matrix
112         zTransform TM;
113         TM.setIdentity();
```

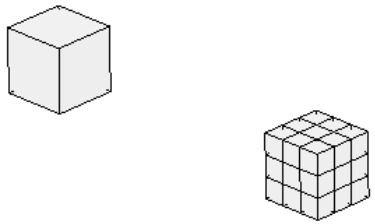


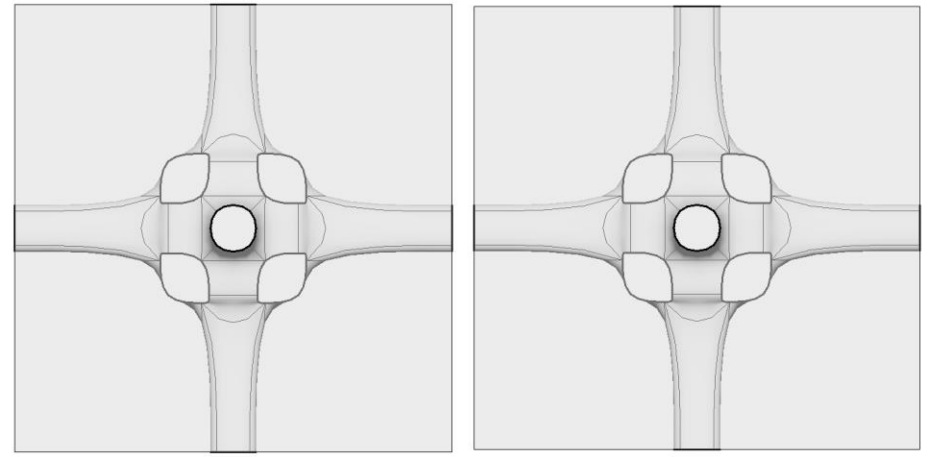


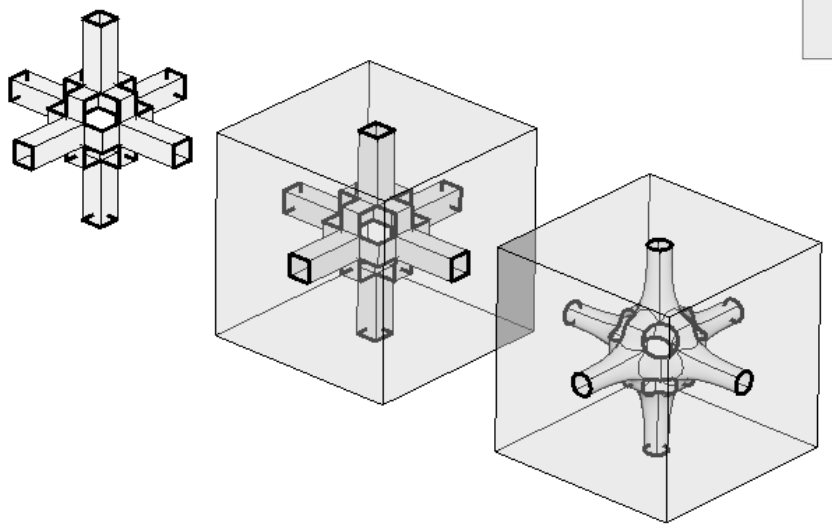
CONNECTION NODE BASE

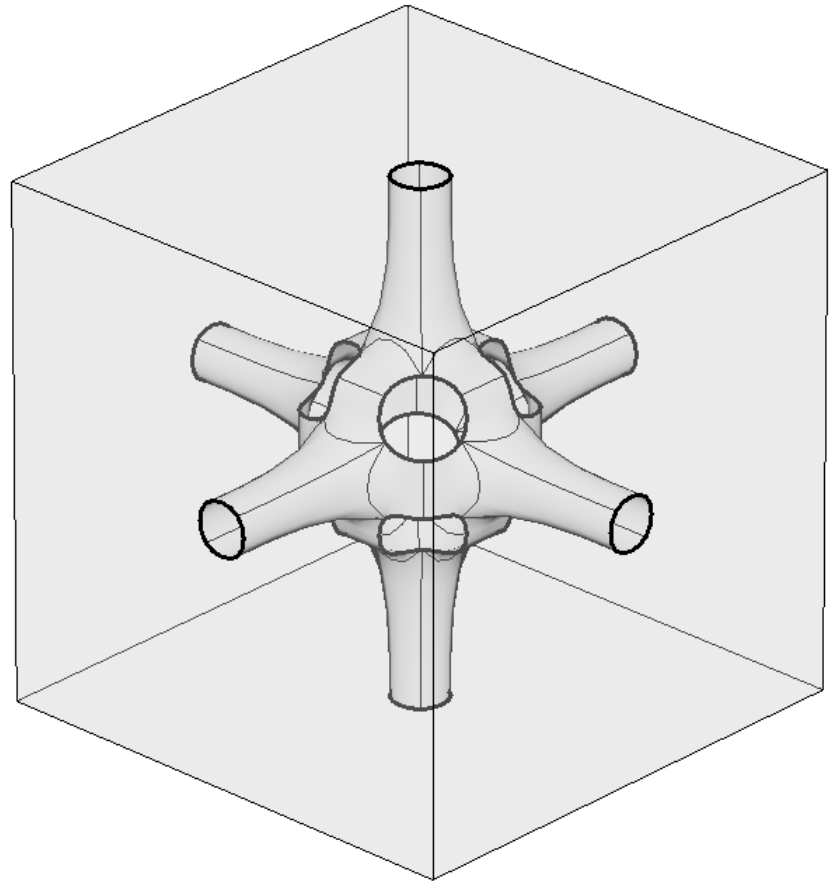
Geometry begins to resemble a language of doors, windows, and walls









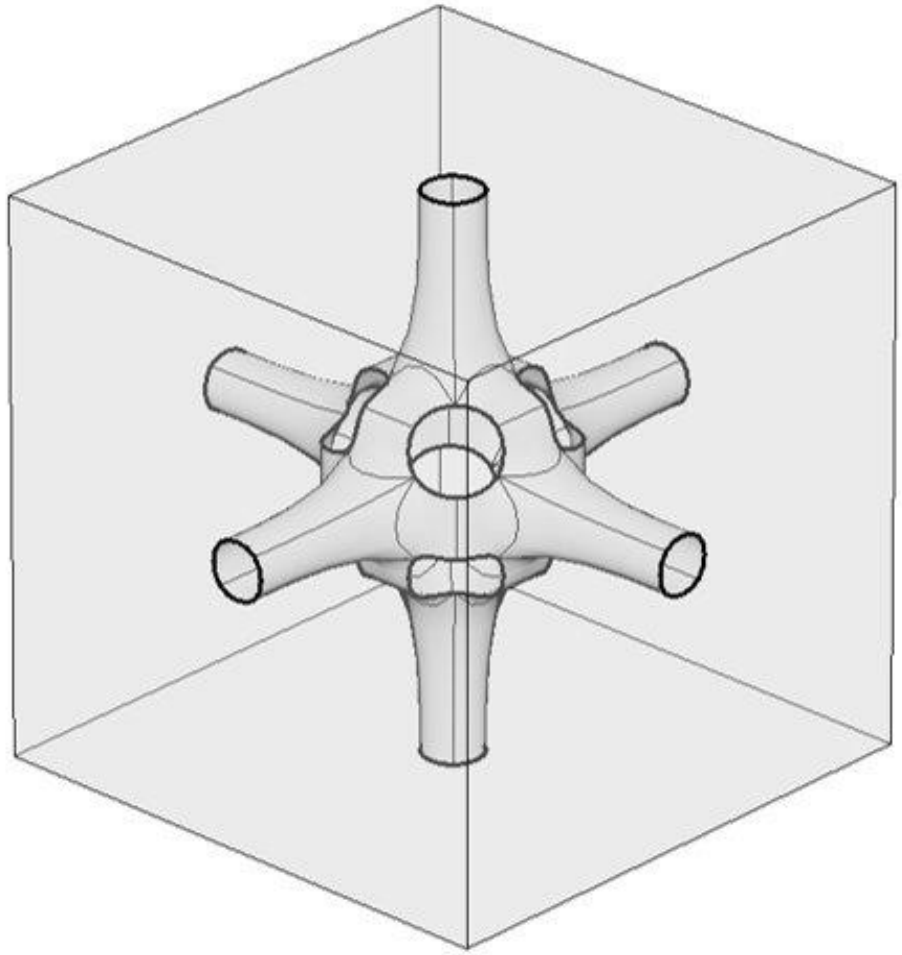


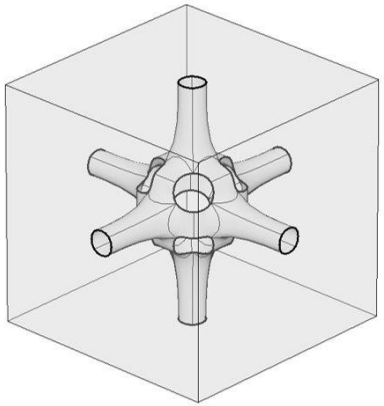
GEOMETRIC EXPLORATIONS [4]

CODED AGGREGATION [4]

Creation of a catalogue of corresponding coded aggregations (c++)

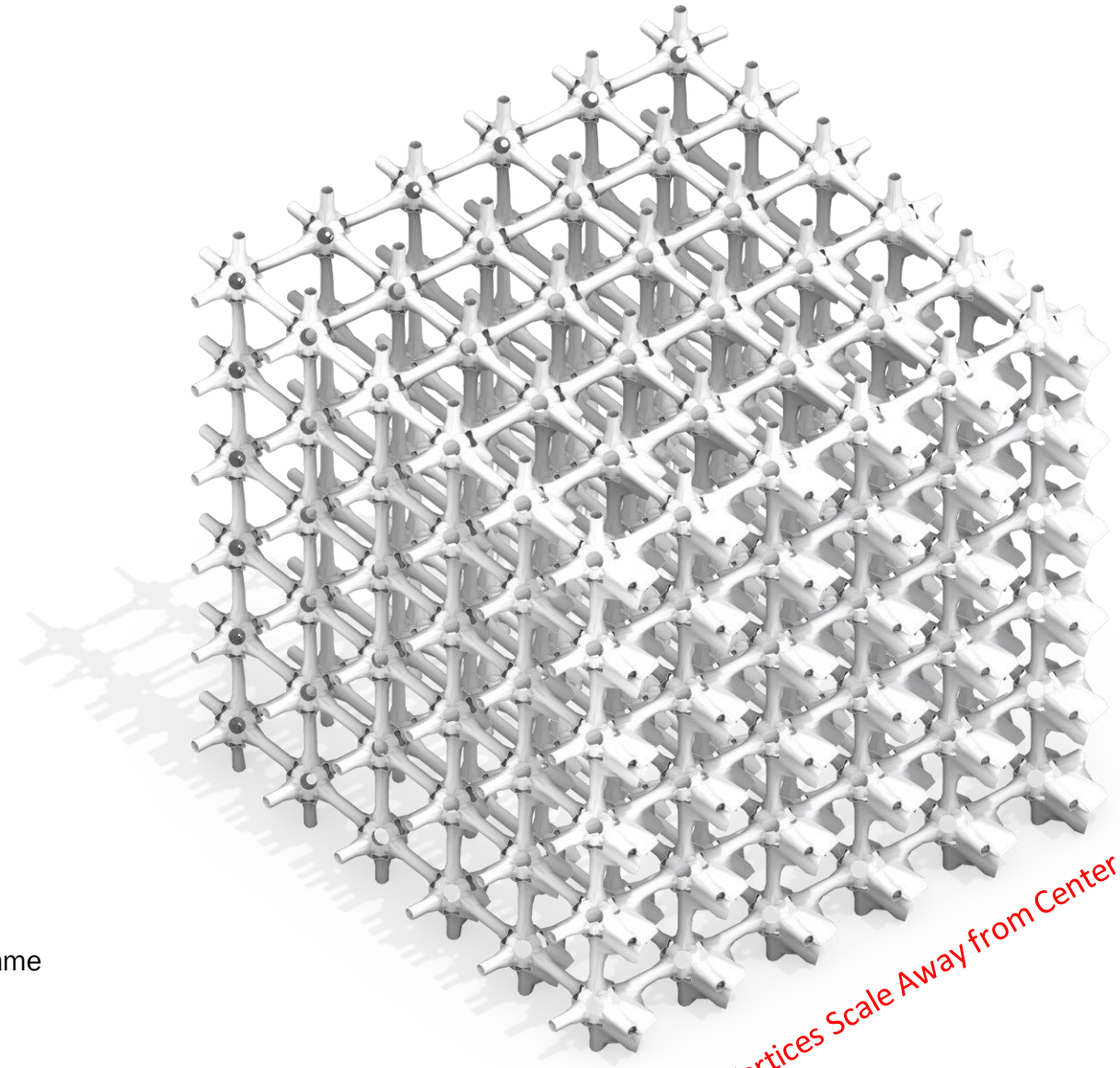
```
...e\Desktop\Alice2022_UoC-main\src\alice\VoxelFactory.h                    3
          trasnfromed
 99       int src[] = {3,0,37,36,45,2,4,5,41,39,38,1};// cross 3d
100       int n = sizeof(src) / sizeof(src[0]);
101       zIntArray ids(src, src + n);
102
103       // construct a gradient parameter
104       zVector u, v, w, c;
105       c = cen;
106       float maxL = diagLength;
107       float mult = ofMap(id, 0, 215, 0.1, 3); ;// 4 - ofMap(cen.length  ↵
          (), 0, maxL, 0.1, 4);
108
109       //cout << mult << "--mult" << endl;
110
111       //construct the transformation matrix
112       zTransform TM;
113       TM.setIdentity();
114       u = zVector(1, 0, 0);
115       v = zVector(0, 1, 0);
116       w = zVector(0, 0, 1);
117       u.normalize(); v.normalize(); w.normalize();
118       u *= mult; v *= mult; w *= mult;
119
```



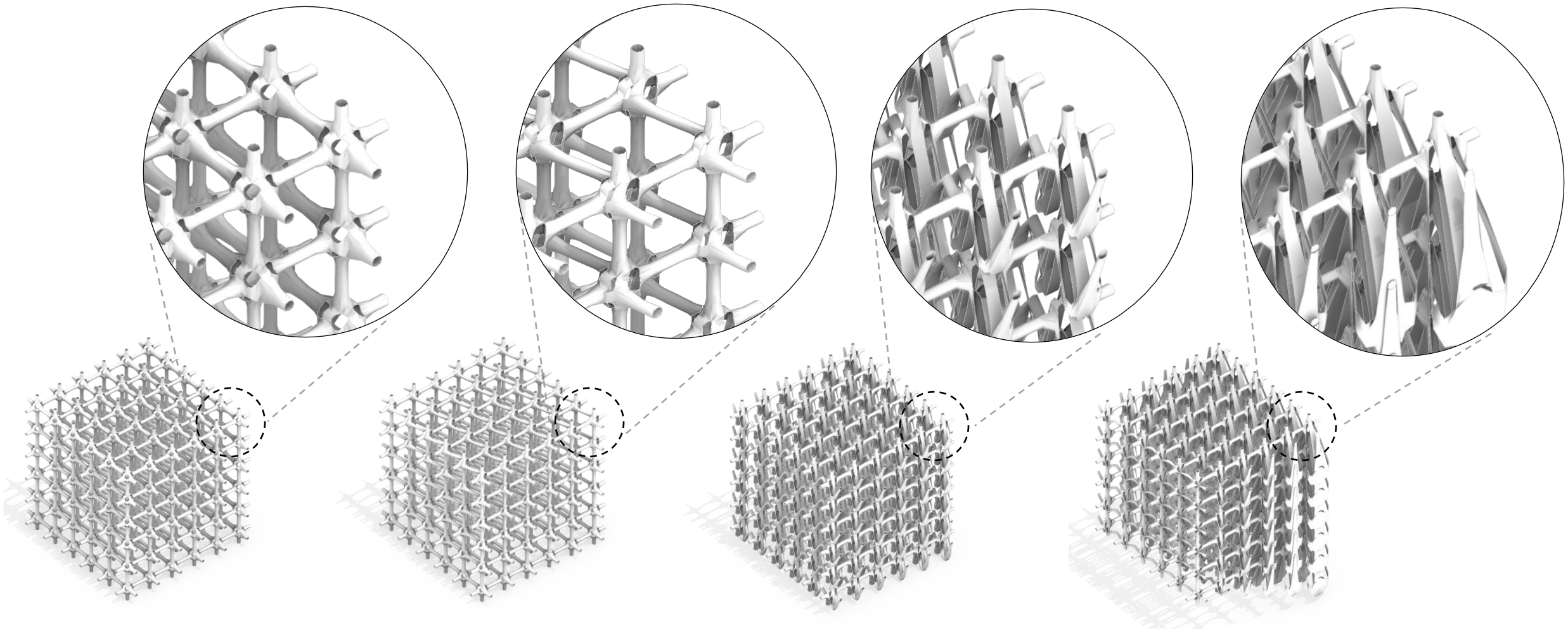


CONNECTION NODE BASE

Geometry begins to become a space frame



EXPLORING ROTATION AND SCALE

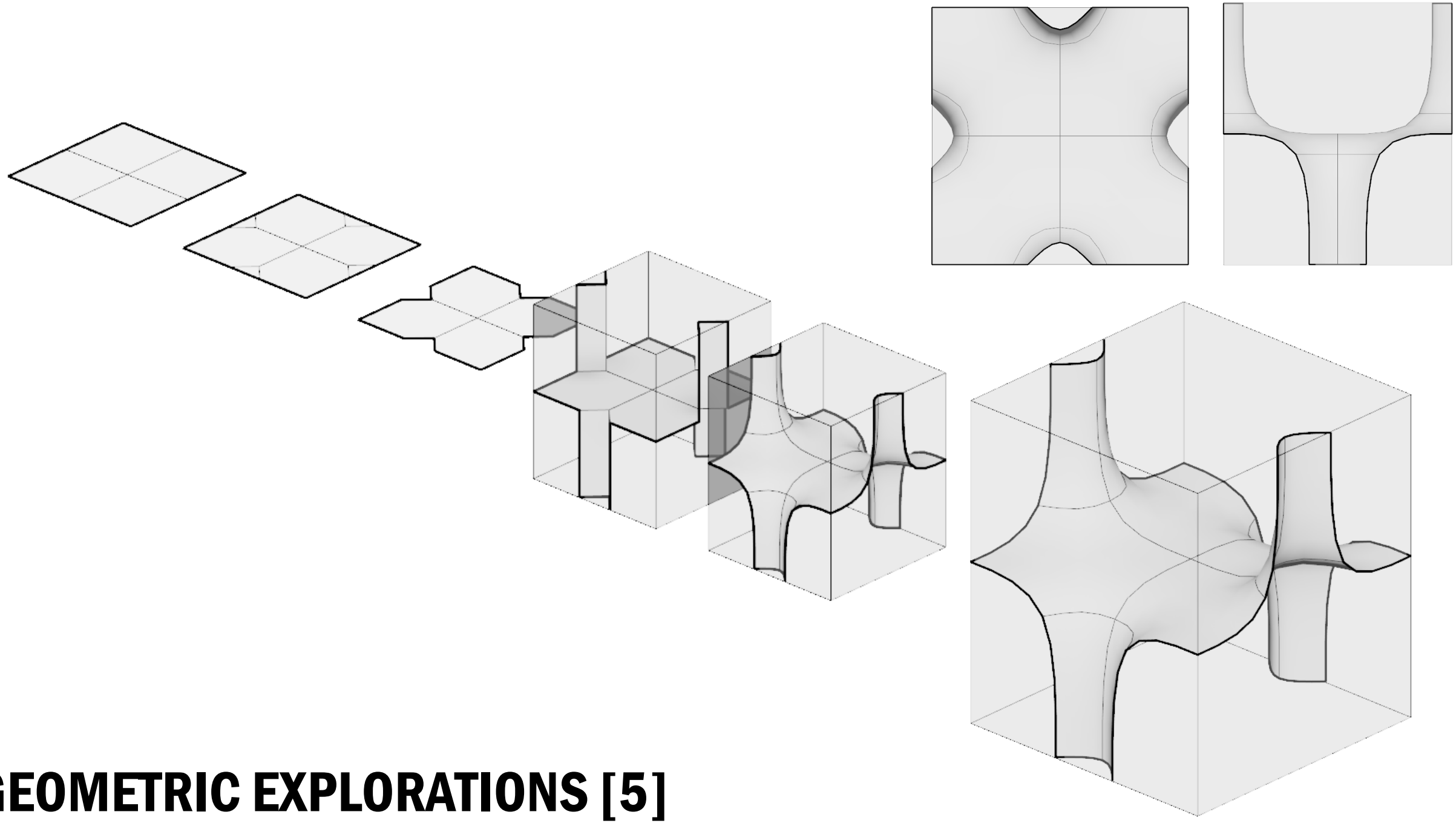


Iteration 4.1

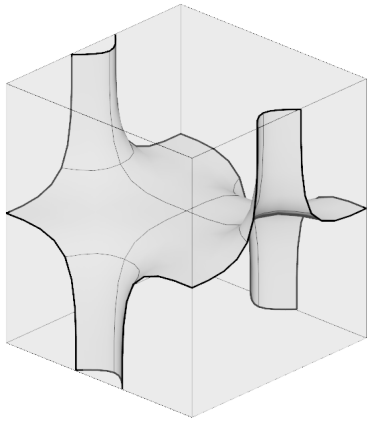
Iteration 4.2

Iteration 4.3

Iteration 4.4

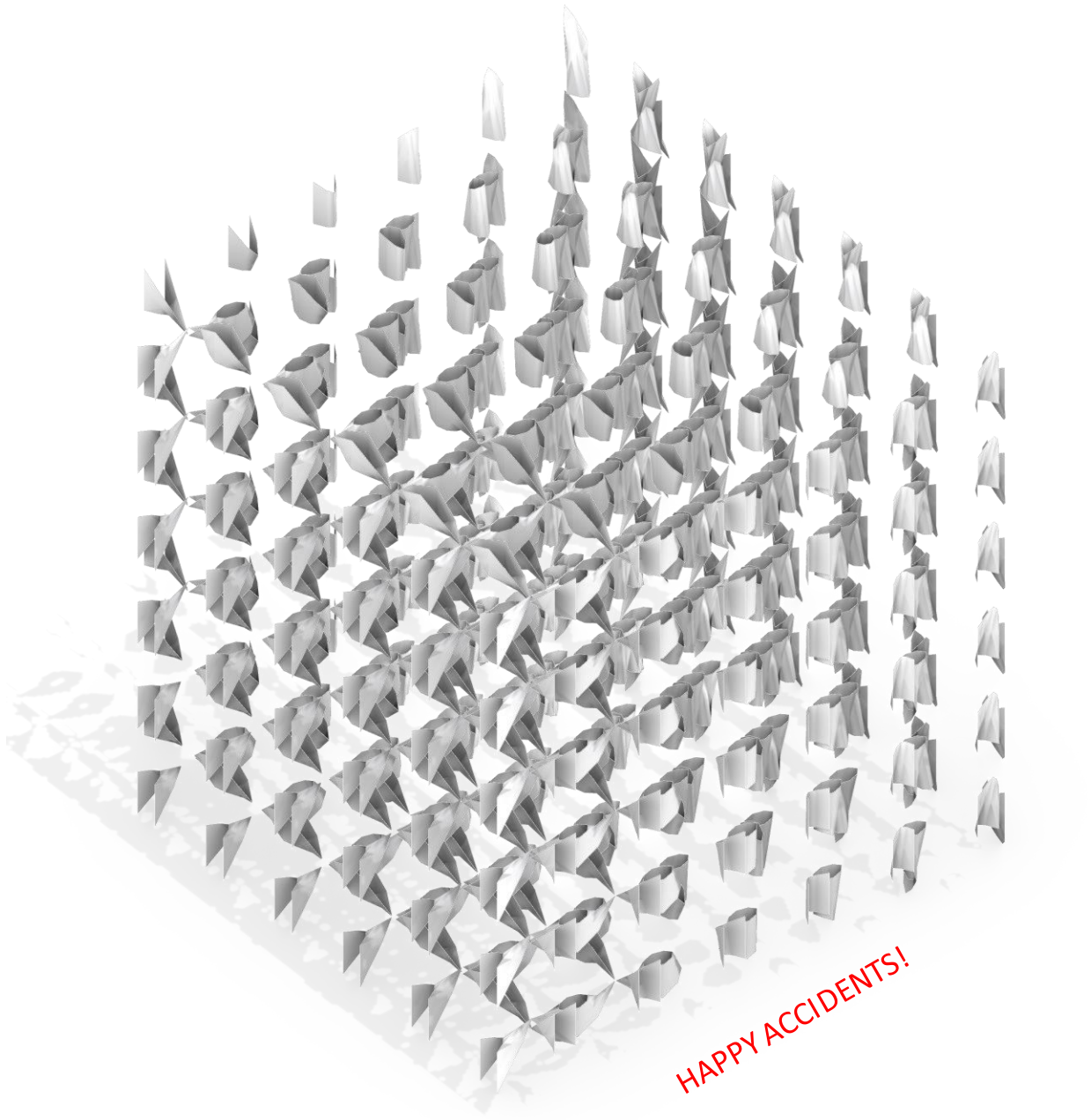


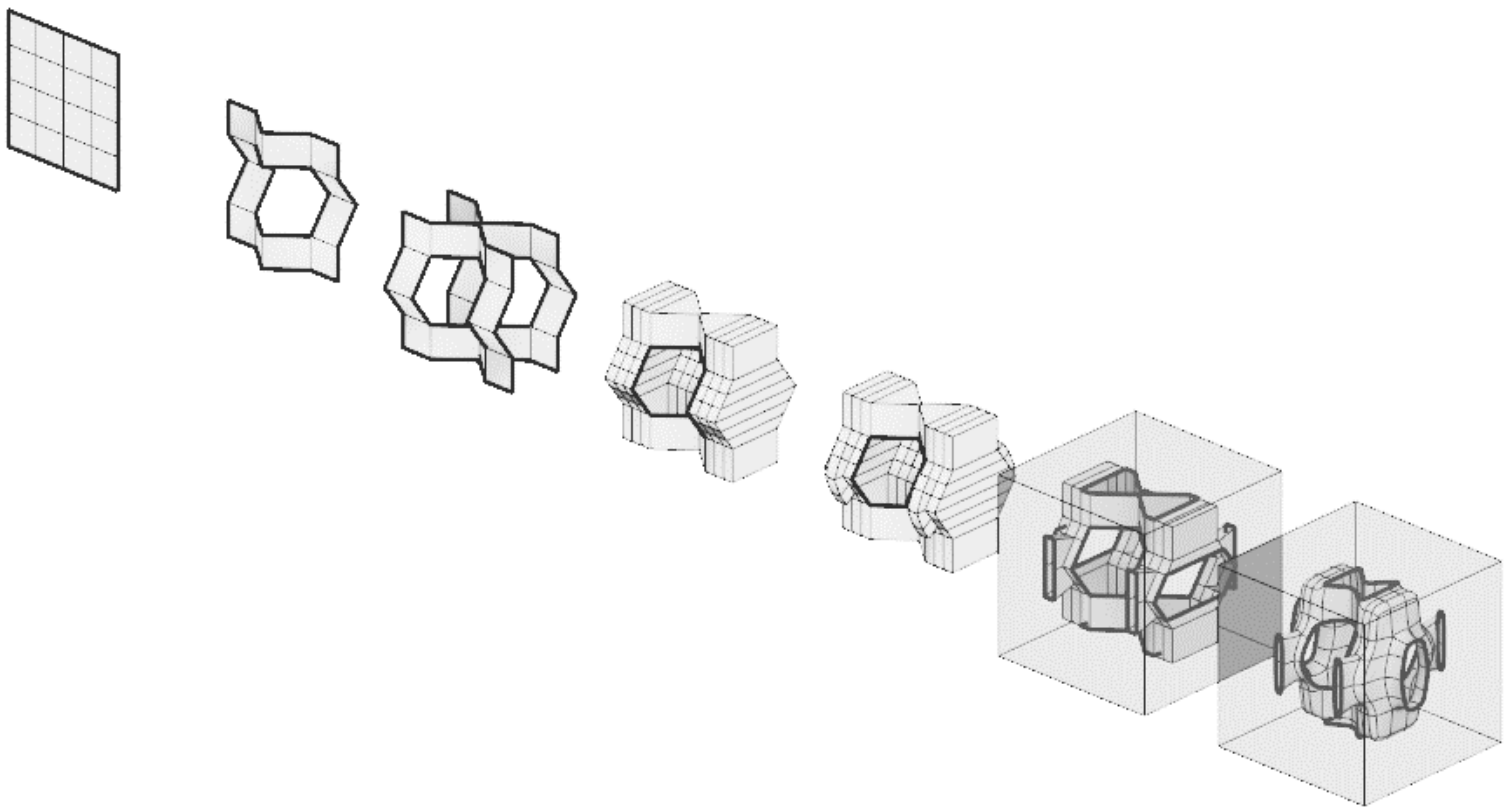
GEOMETRIC EXPLORATIONS [5]

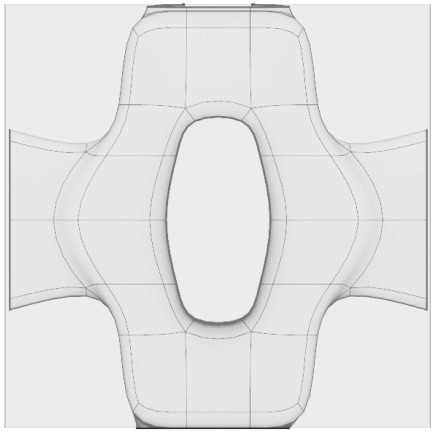


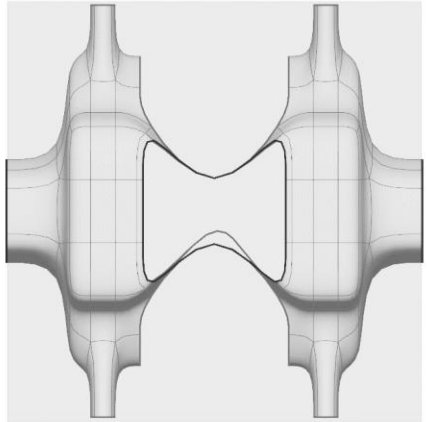
TPMS BASE

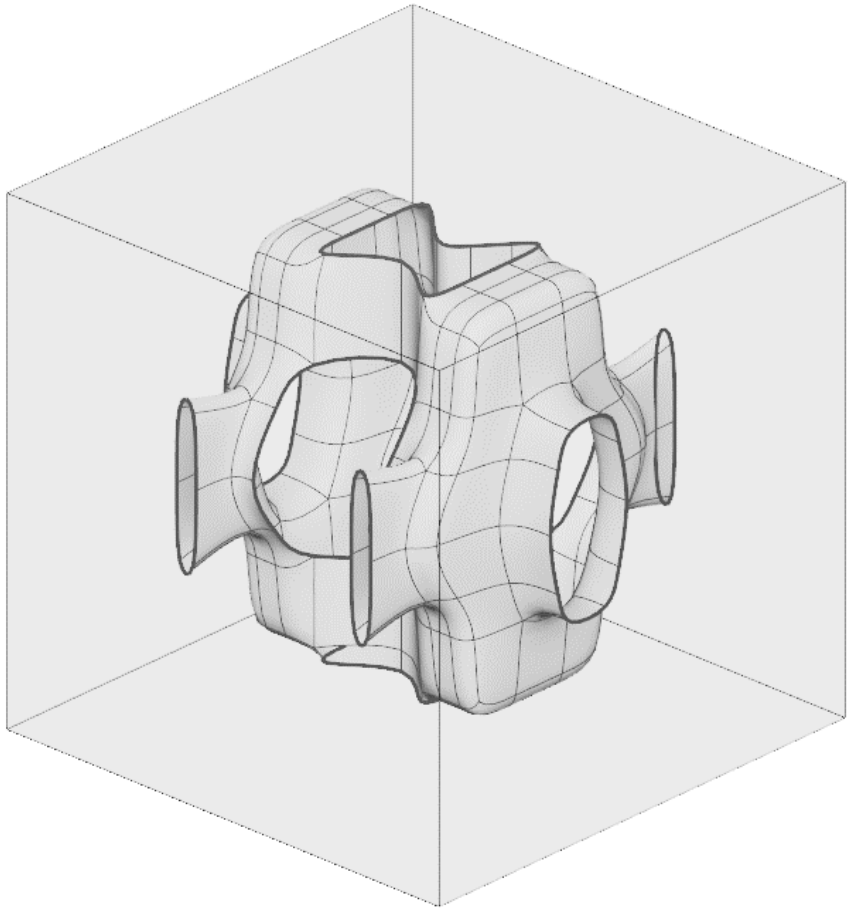
An attempt to create a triply periodic minimal surface becomes a horrifying monster



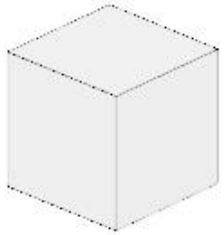
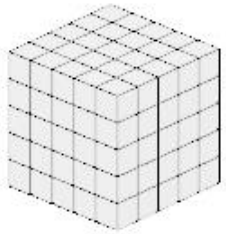
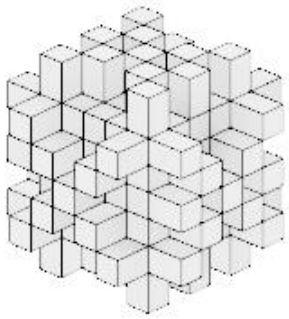
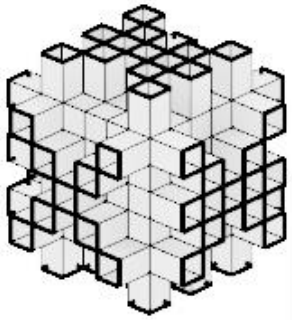
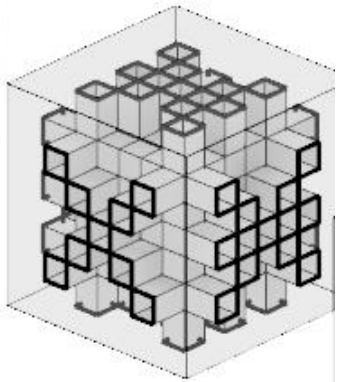
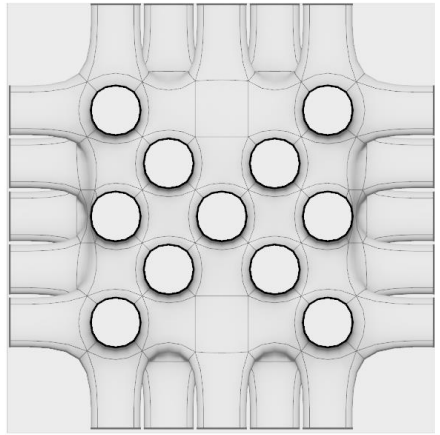
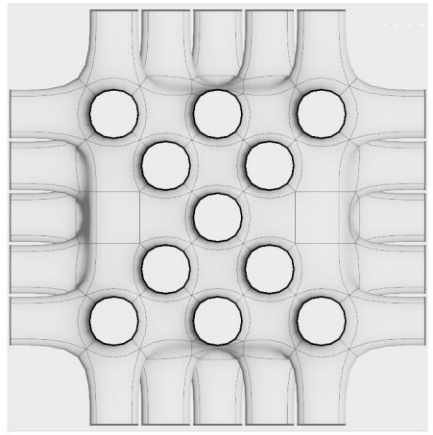
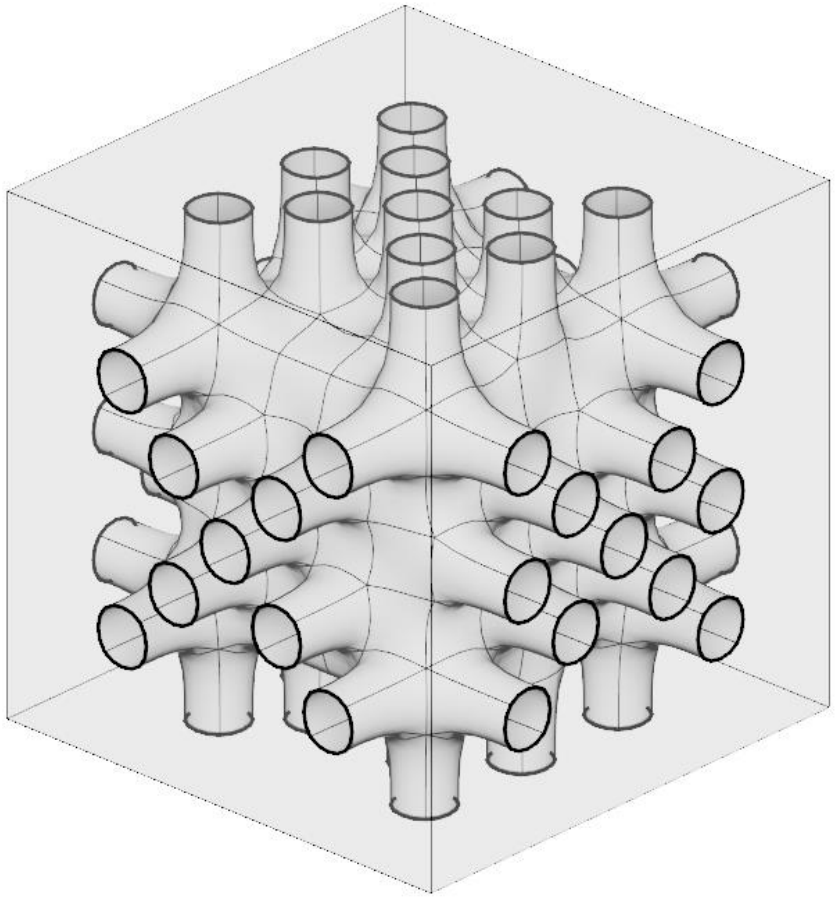




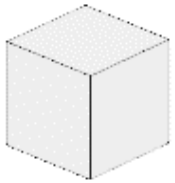


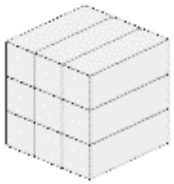


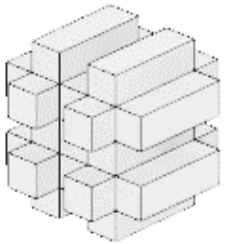
GEOMETRIC EXPLORATIONS [6]

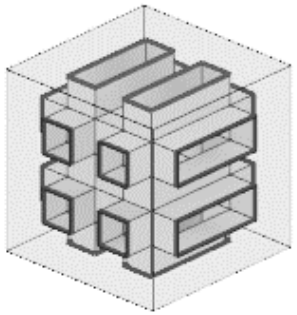









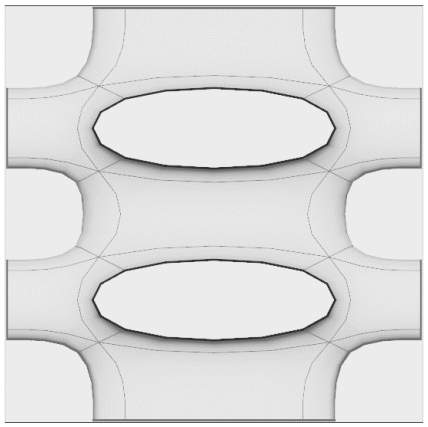
GEOMETRIC EXPLORATIONS [7]

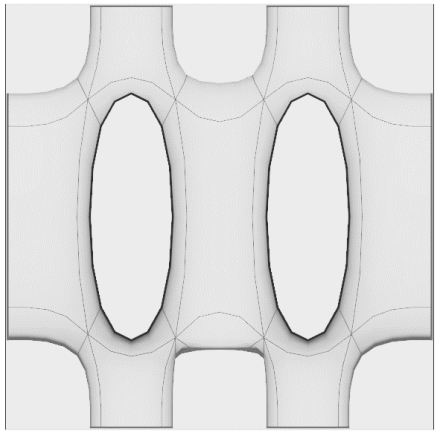


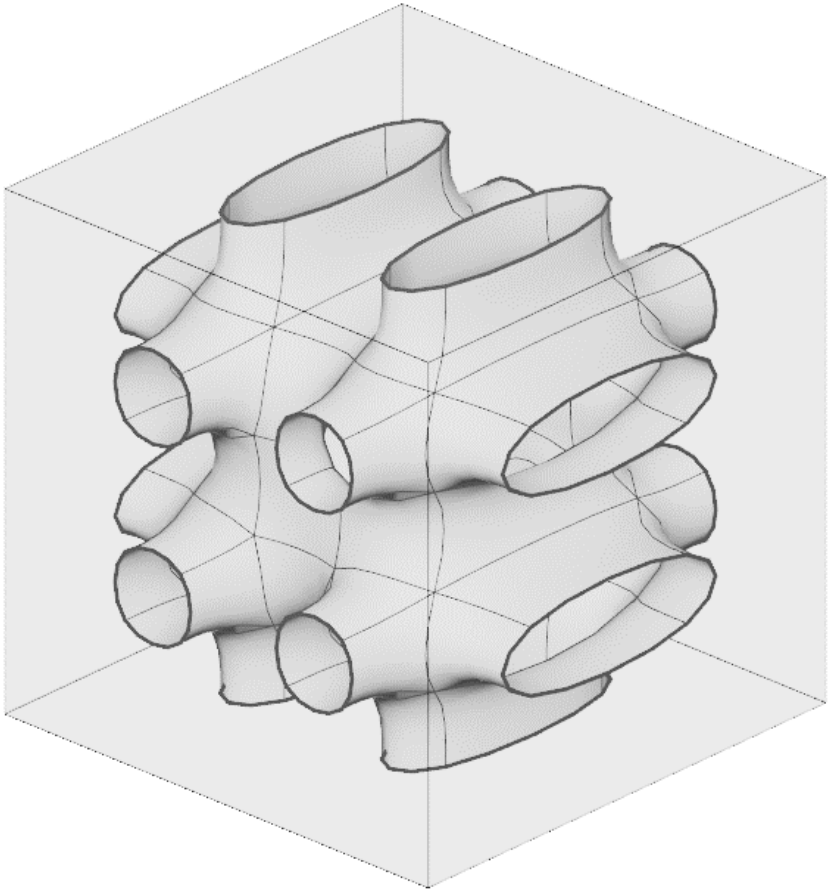




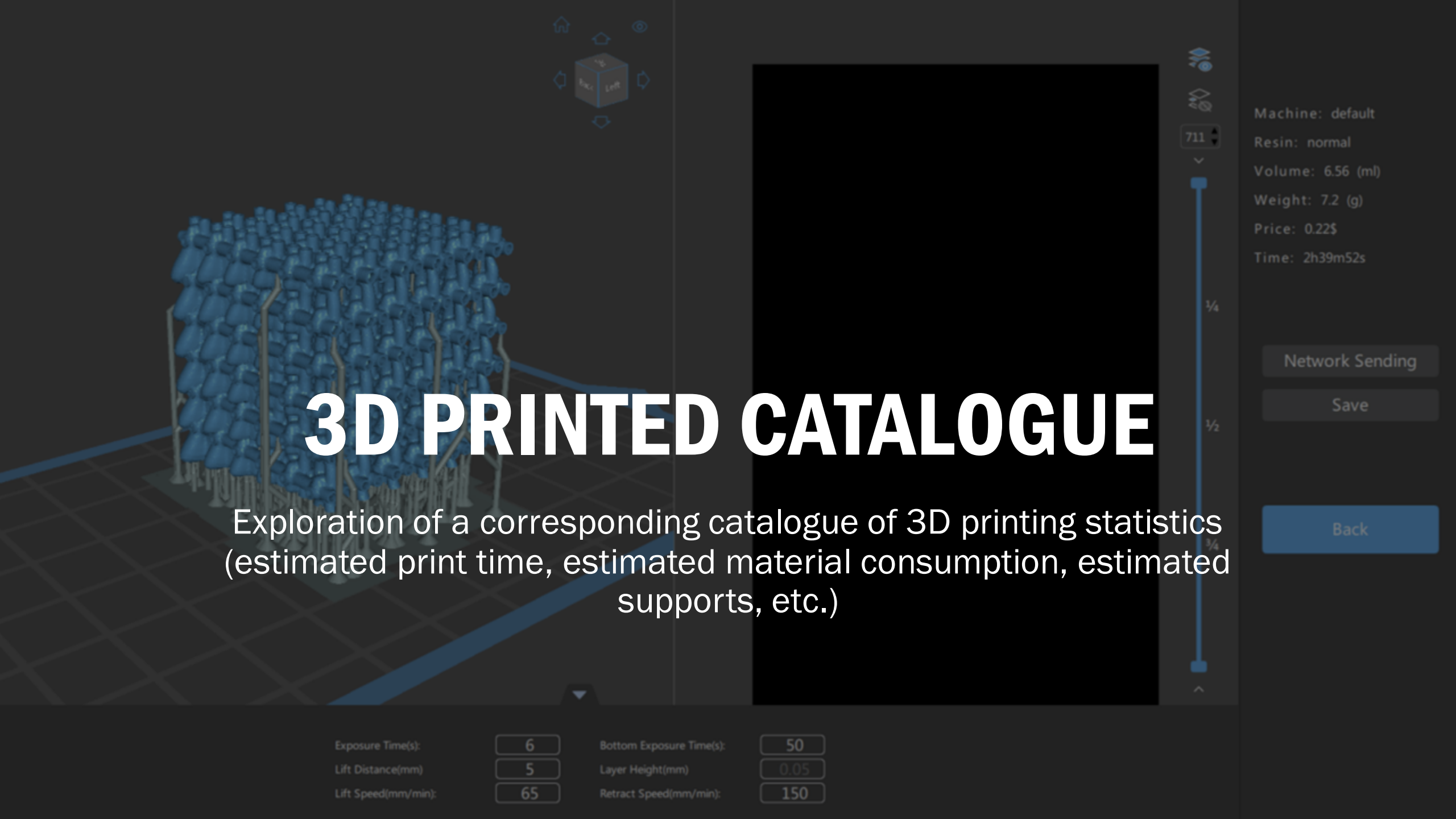






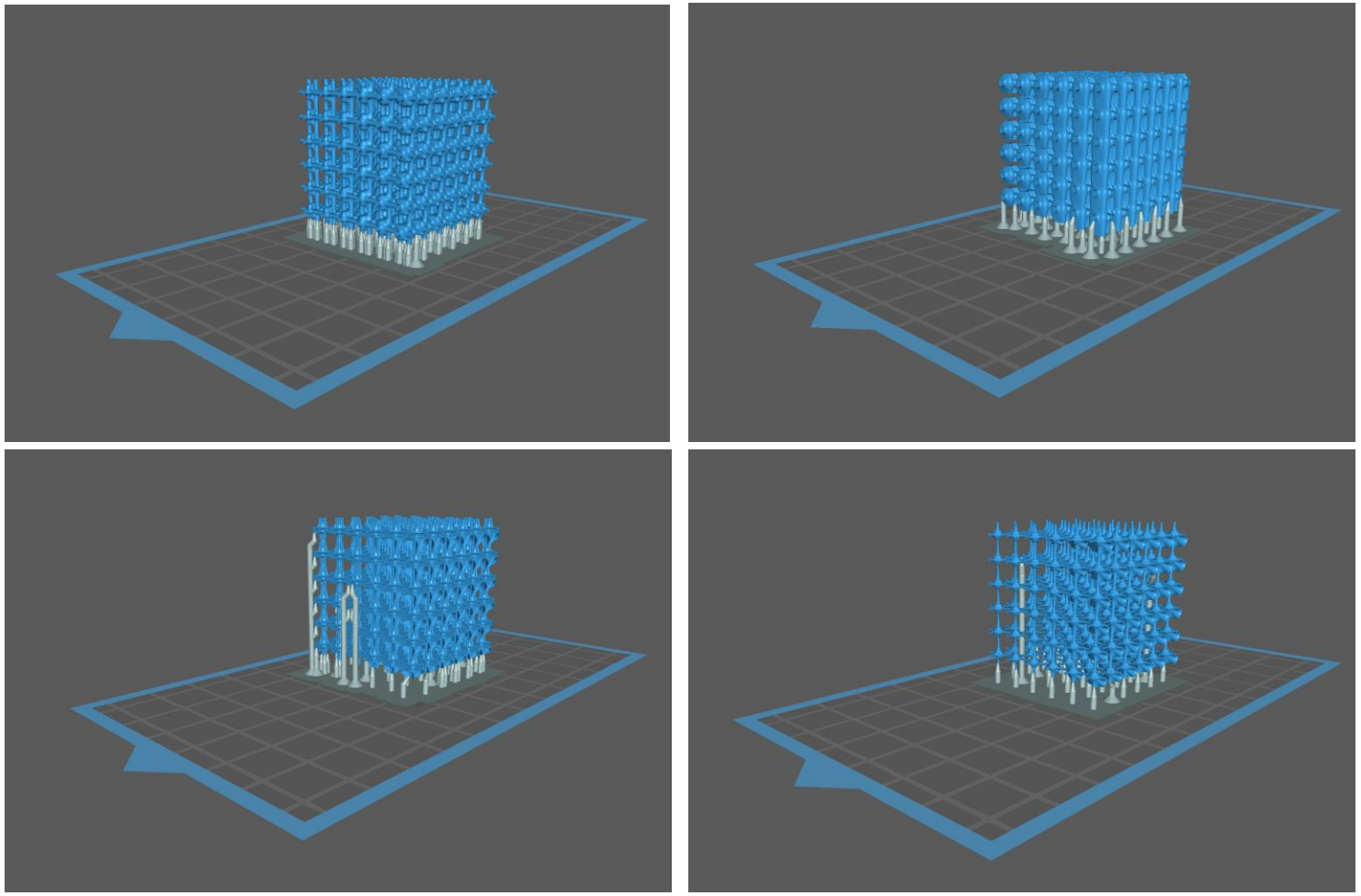


GEOMETRIC EXPLORATIONS [8]



3D PRINTED CATALOGUE

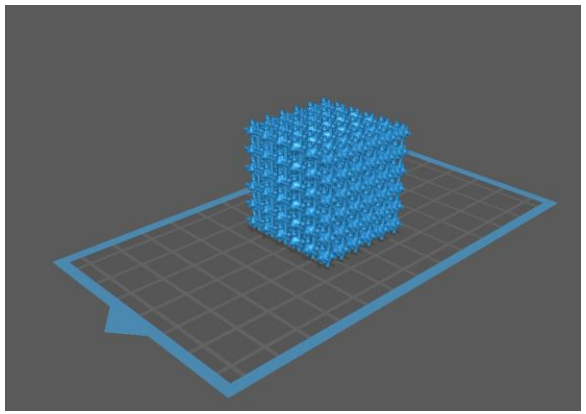
Exploration of a corresponding catalogue of 3D printing statistics (estimated print time, estimated material consumption, estimated supports, etc.)

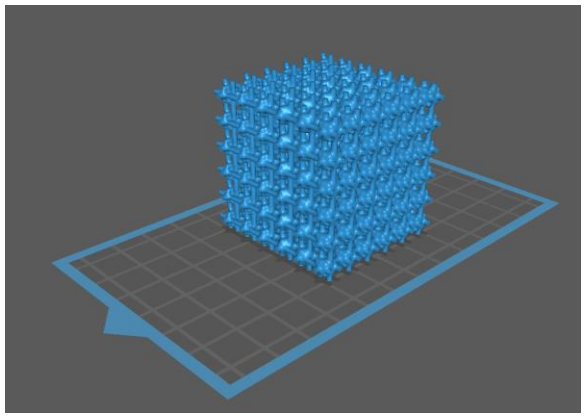


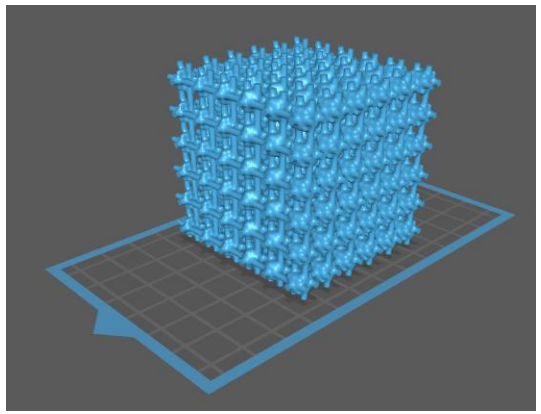
All sliced models require relatively simple and reduced supports

RESIN SUPPORT PLACEMENT

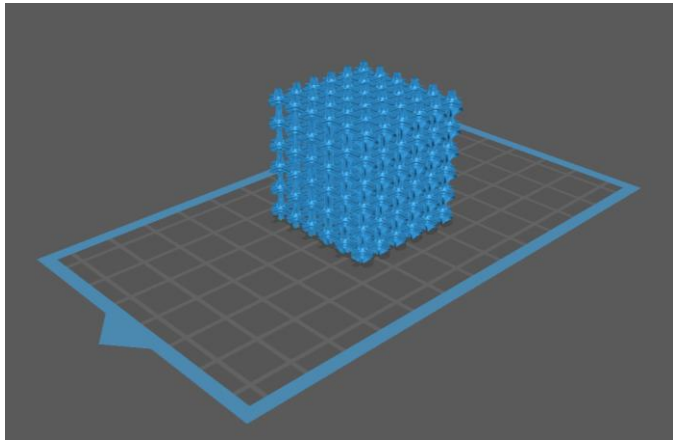
1

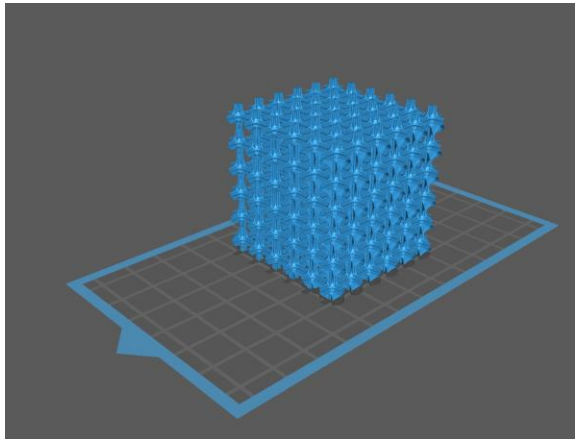


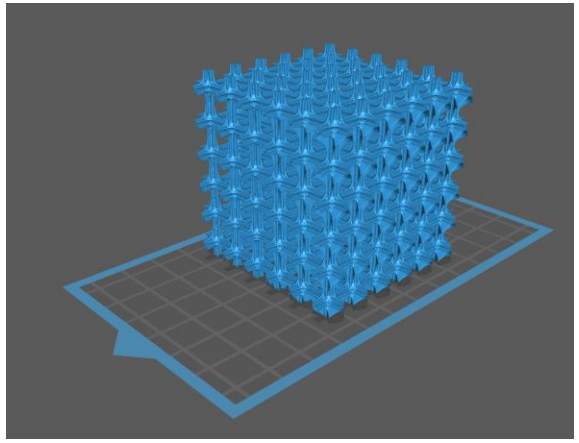




2



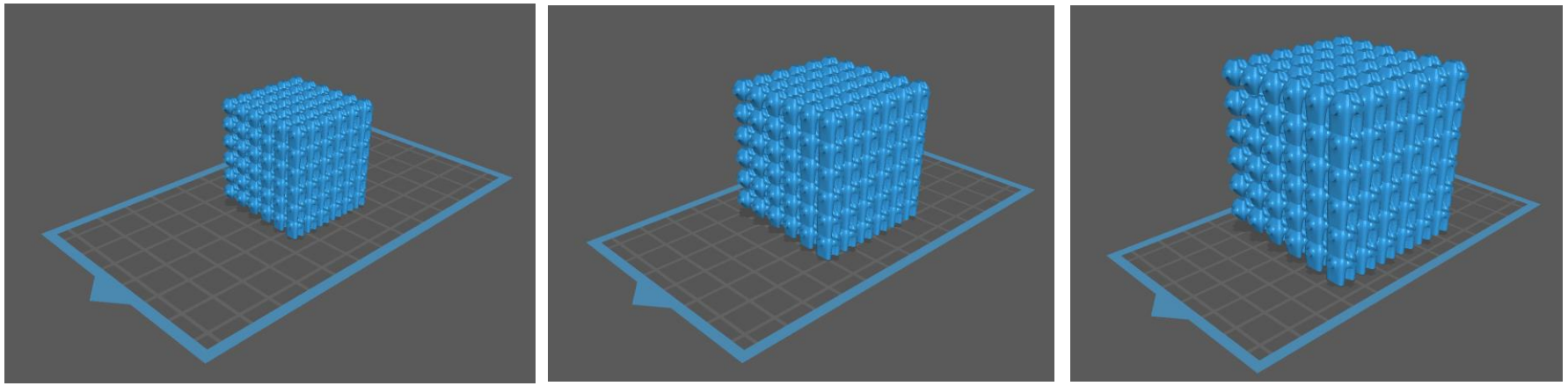




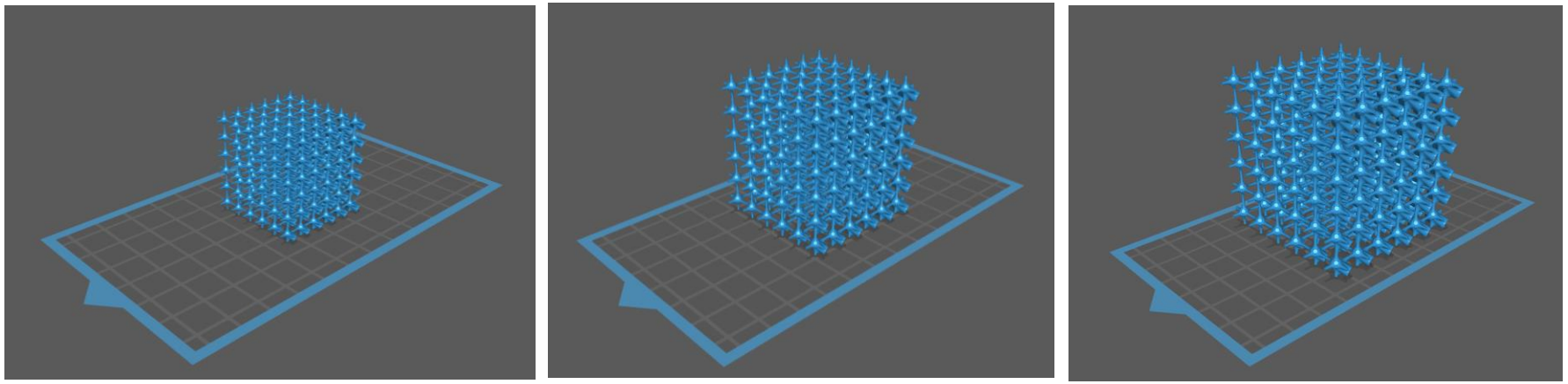
Ranges in scale

PRINT CATALOGUE

3



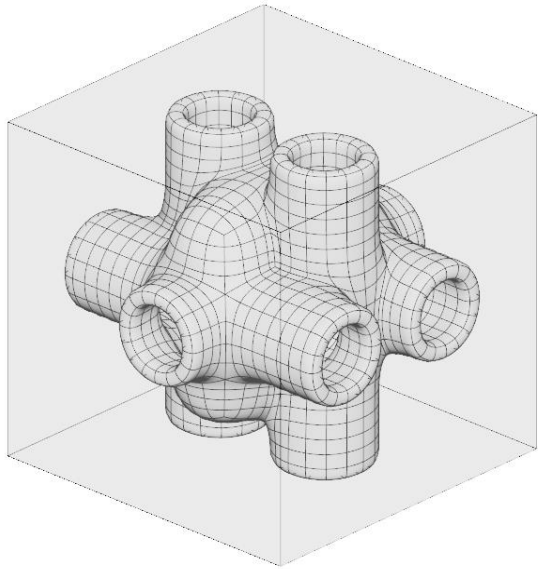
4

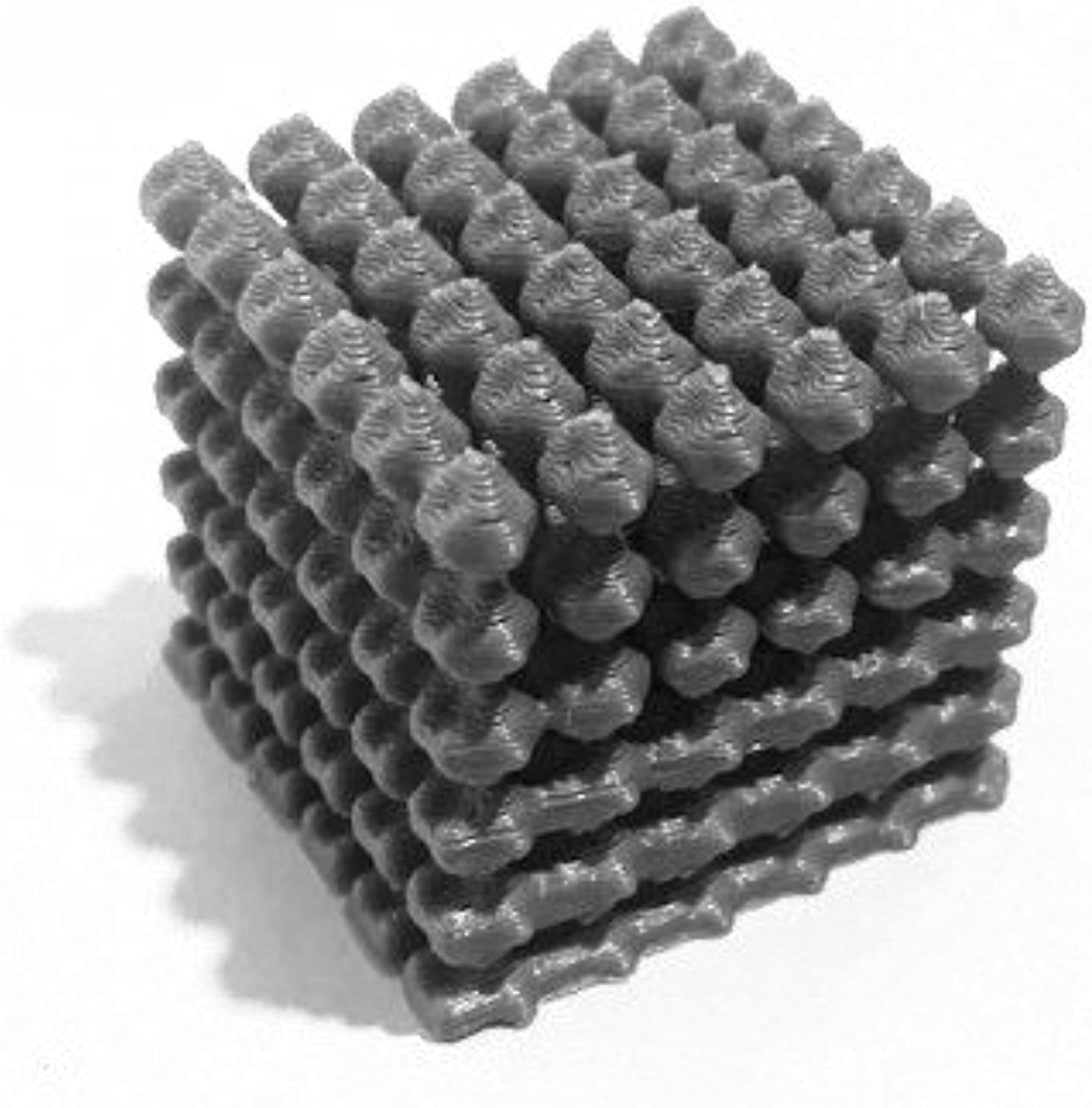


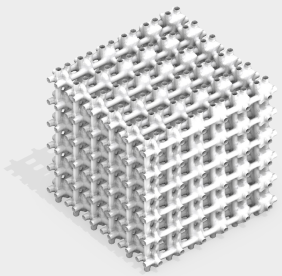
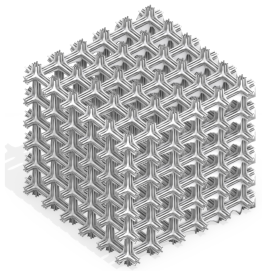
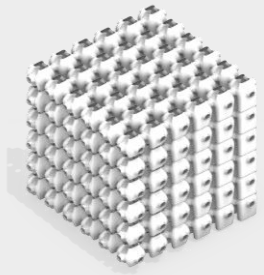
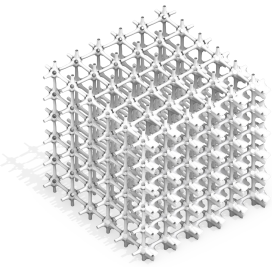
All sliced models require relatively simple and reduced supports

PRINT CATALOGUE

PLA PRINT TEST

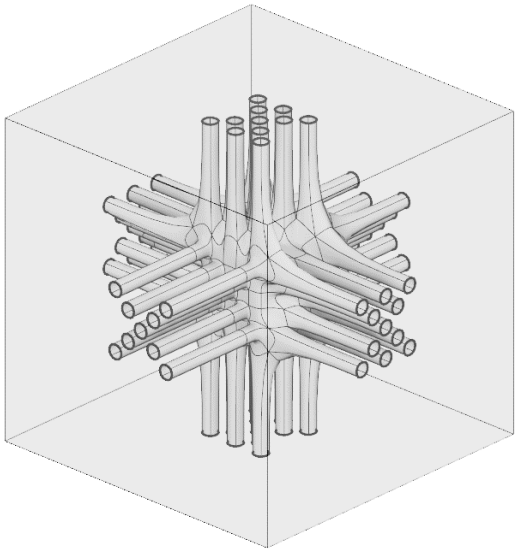




Geometry #	Geometry	Cubic Sizing (mm)	Volume (ml)	Weight (g)	Time (min)
1		50	30.98	34.1	222
		40	15.86	17.4	180
		30	6.69	7.4	136
2		50	12.35	13.6	222
		40	6.32	7.0	180
		30	2.67	2.9	136
3		50	52.26	57.5	225
		40	26.77	29.4	181
		30	11.29	12.4	137
4		50	11.38	12.5	222
		40	5.83	6.4	179
		30	2.46	2.7	136

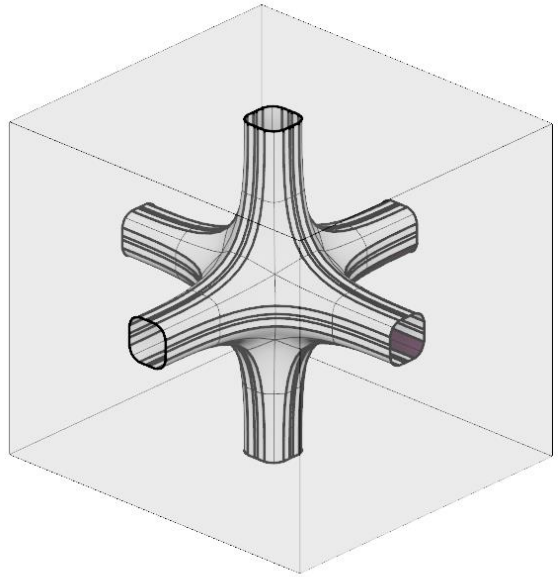
LESSONS LEARNED/ DISCOVERIES

Geometry with a large number of vertices was more difficult to manipulate with code. Simple and clean geometries gave the best and most understandable results



466 Vertices

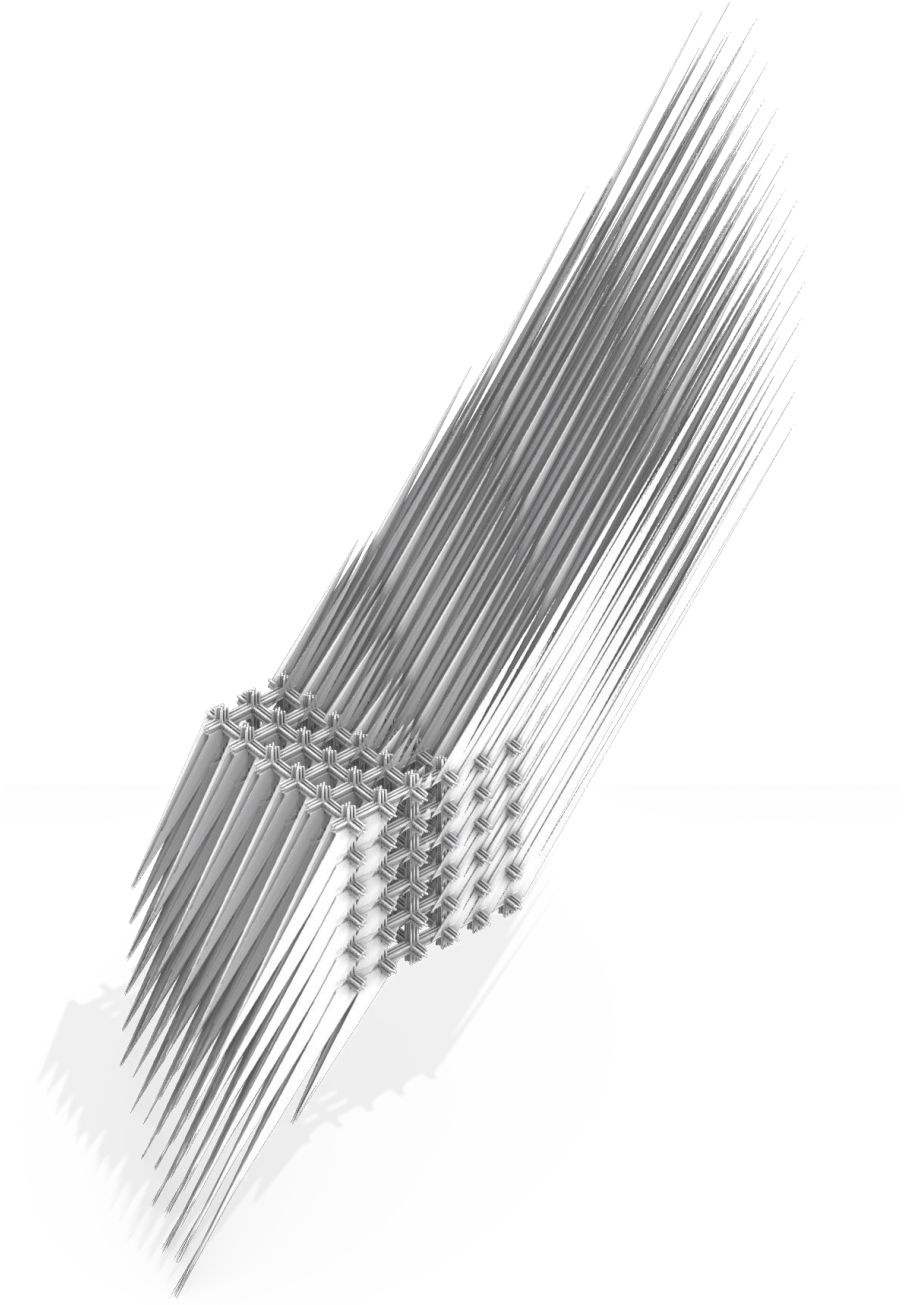
Vs.



31 Vertices

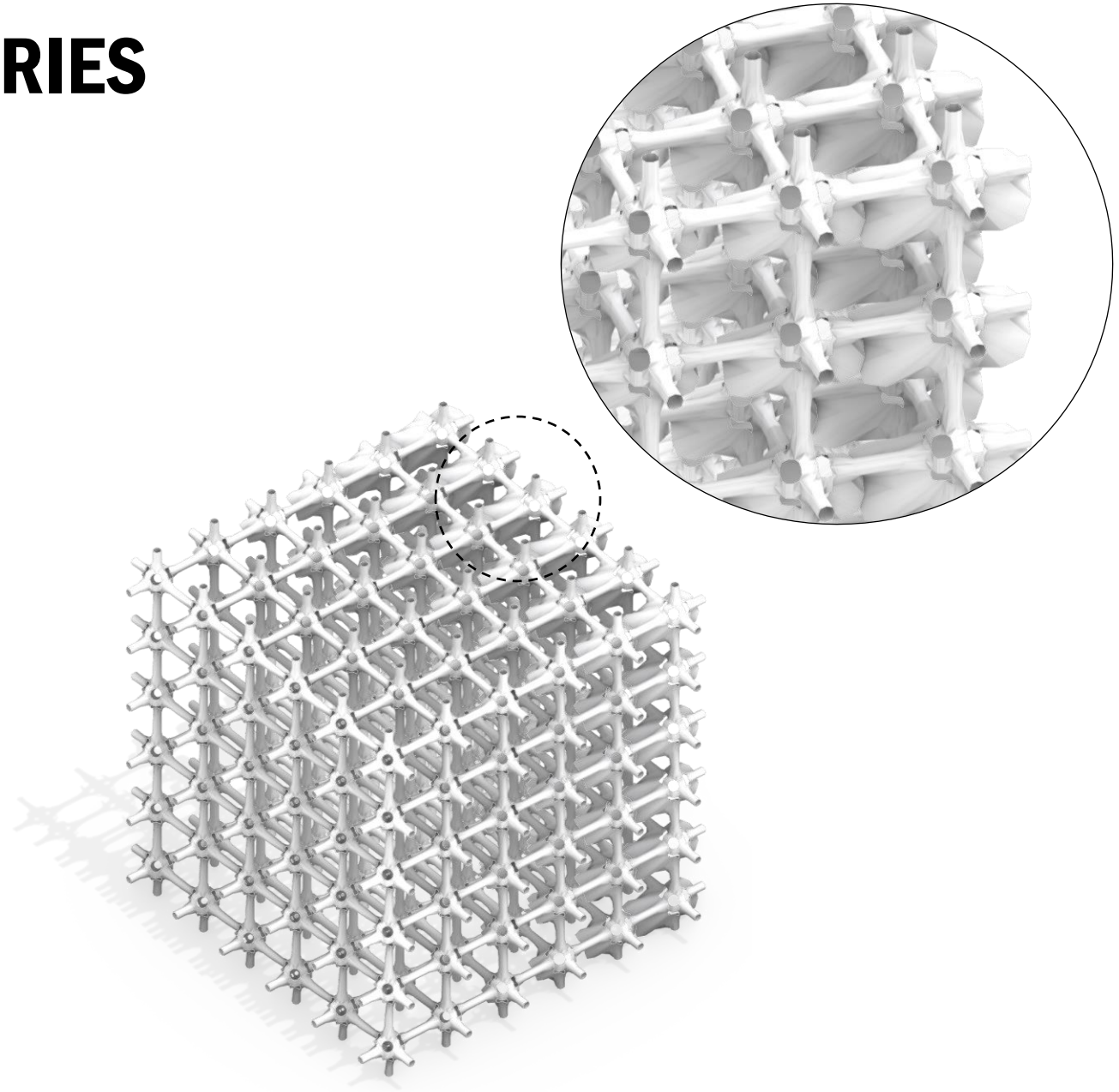
LESSONS LEARNED/ DISCOVERIES

Minor tweaks prior to bringing the component into Maya, such as moving it to the origin, making it the correct size, and testing how it overlapped were essential to a successful code



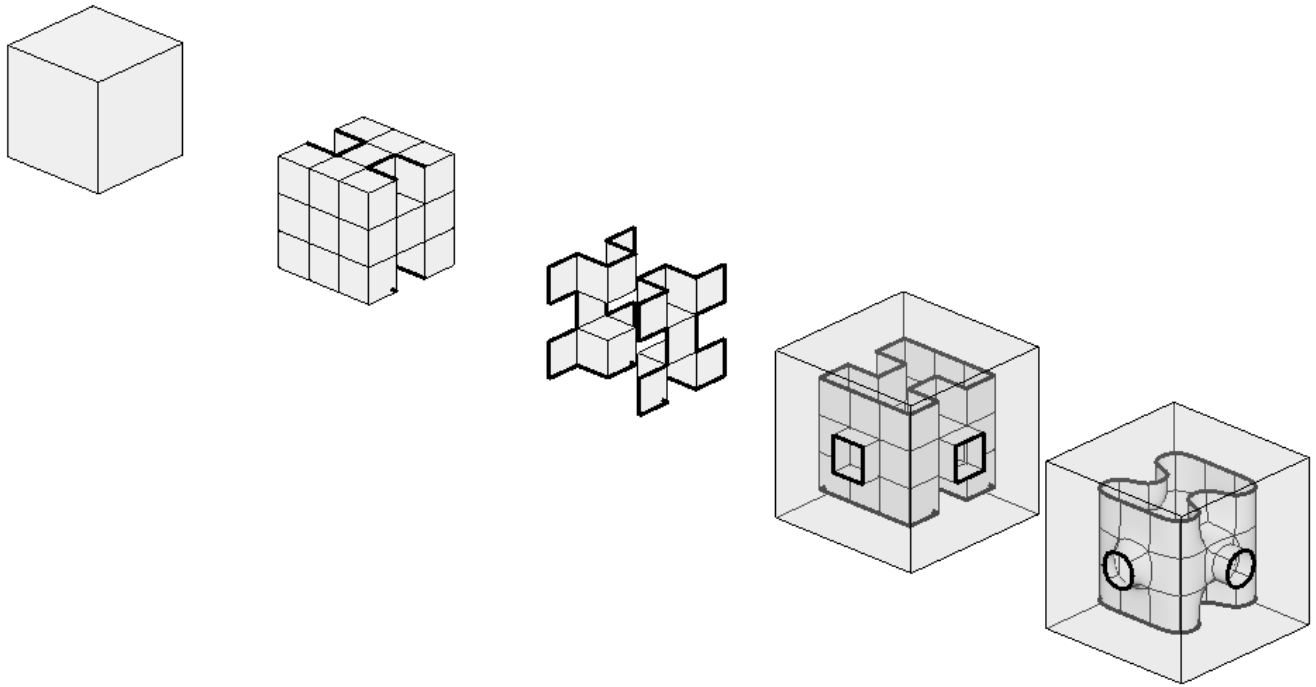
LESSONS LEARNED/ DISCOVERIES

Scaling up the component in the code sometimes created faces that were too big and overlapped with the adjacent component



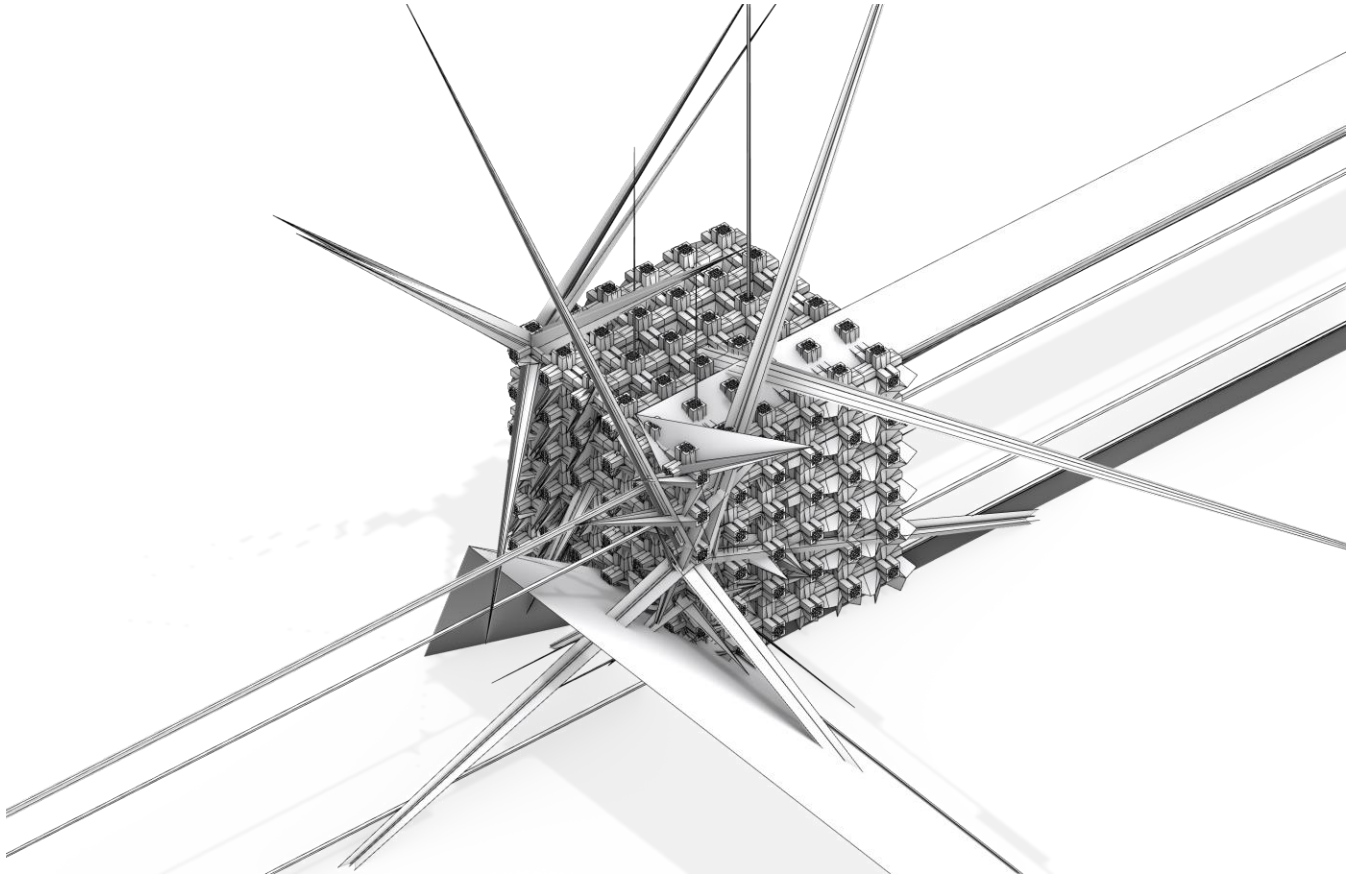
LESSONS LEARNED/ DISCOVERIES

When creating the component, the most successful iterations we're the ones that started with a cube, subdivisions were added, and faces or edges were moved or deleted to obtain the desired shape



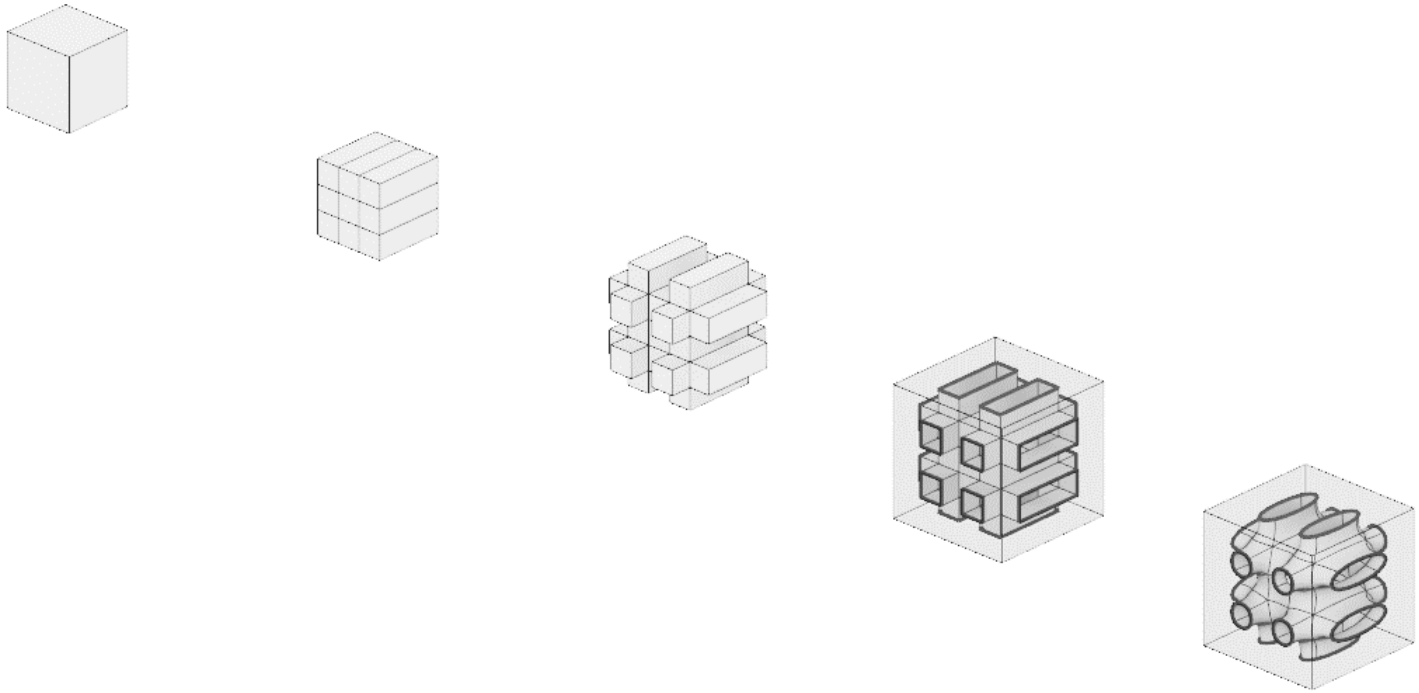
LESSONS LEARNED/ DISCOVERIES

Adding in a thickness into the component in Maya caused the code to become heavy and visual studio would continuously crash. The solution was to either add the solidified to the entire cube after it was obtained from the code or so use a mesh mixer



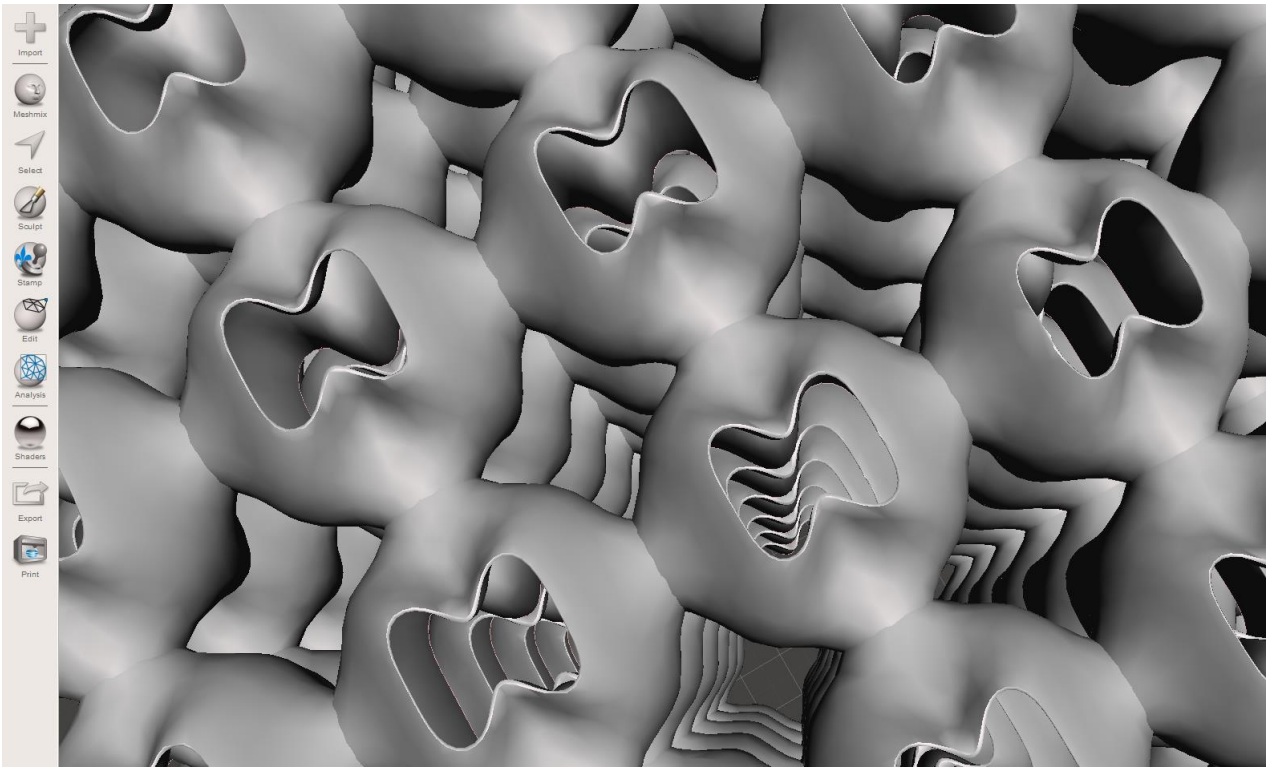
LESSONS LEARNED/ DISCOVERIES

Working procedurally and developing a method of keeping track of steps and parameters when creating each component in Maya aided in modifying shapes to better suit the code



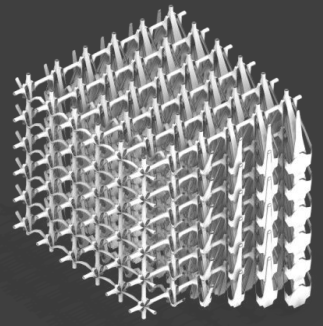
LESSONS LEARNED/ DISCOVERIES

Completed arrays are still not optimized for 3d printing. In our 3d printing experiments, additional (and sometimes extensive) post processing (3D Builder + Meshmixer) was required to translate digital files to the physical realm.

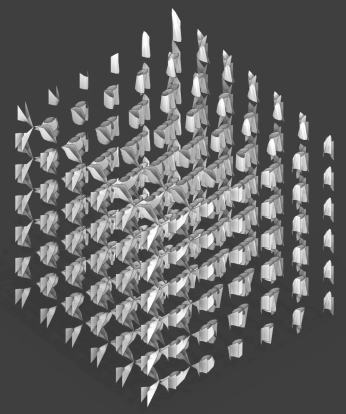


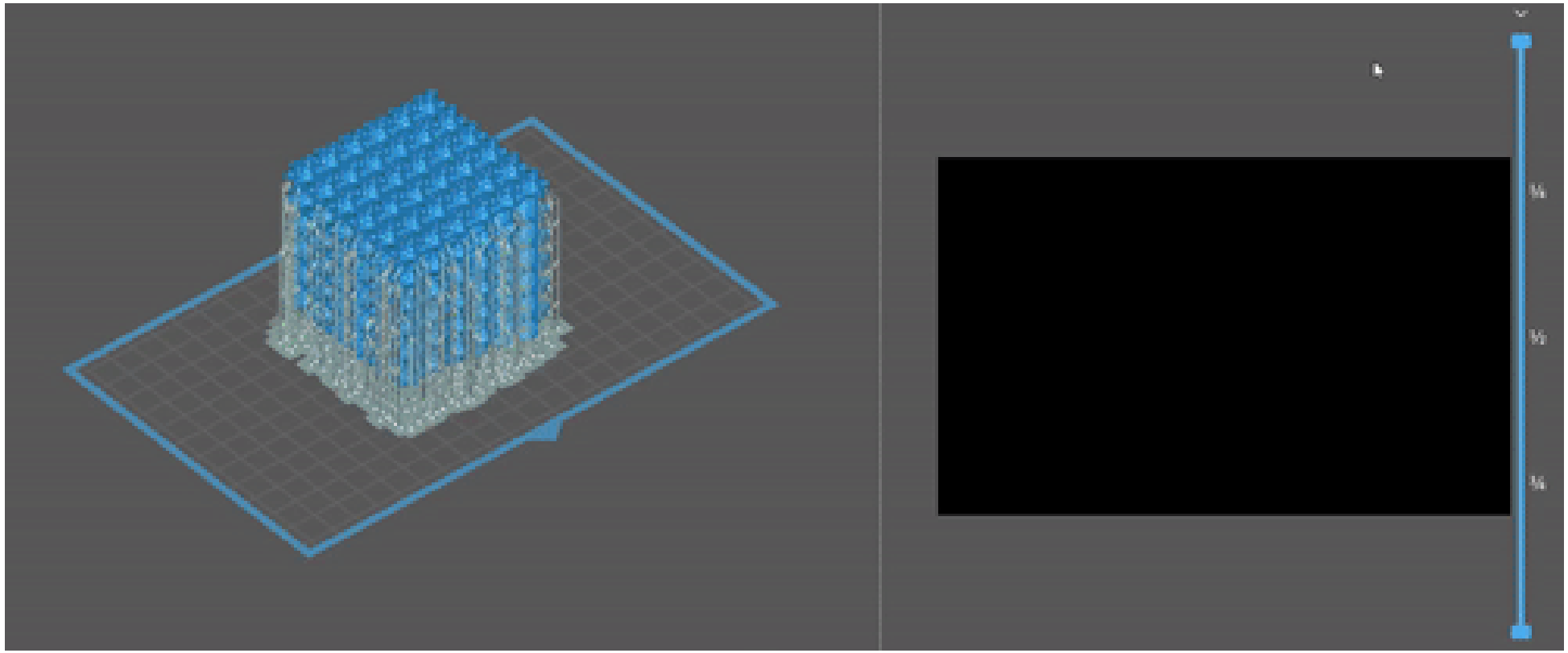
HAPPY ACCIDENTS

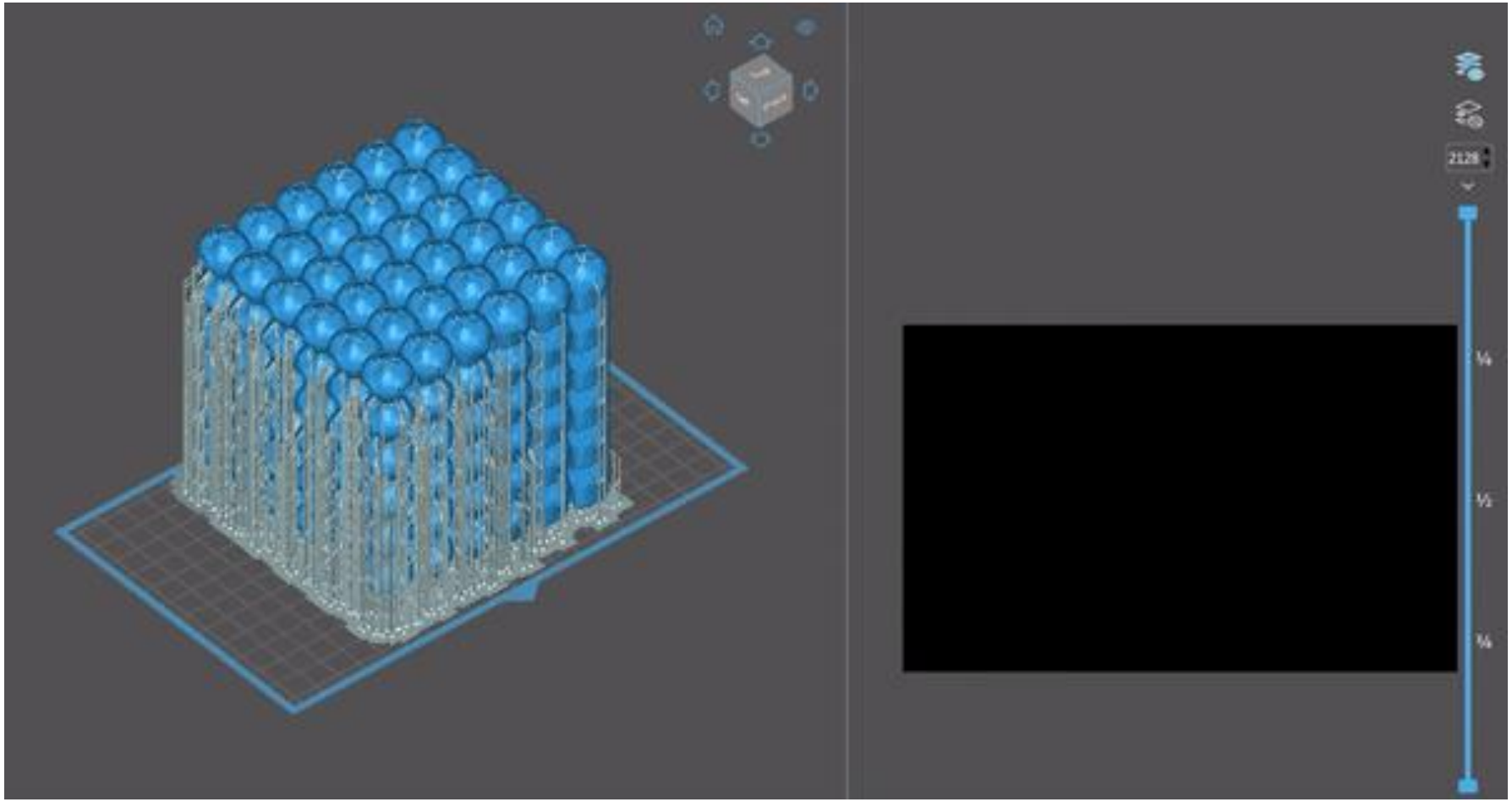
1. Playing with rotation and scale factors within code

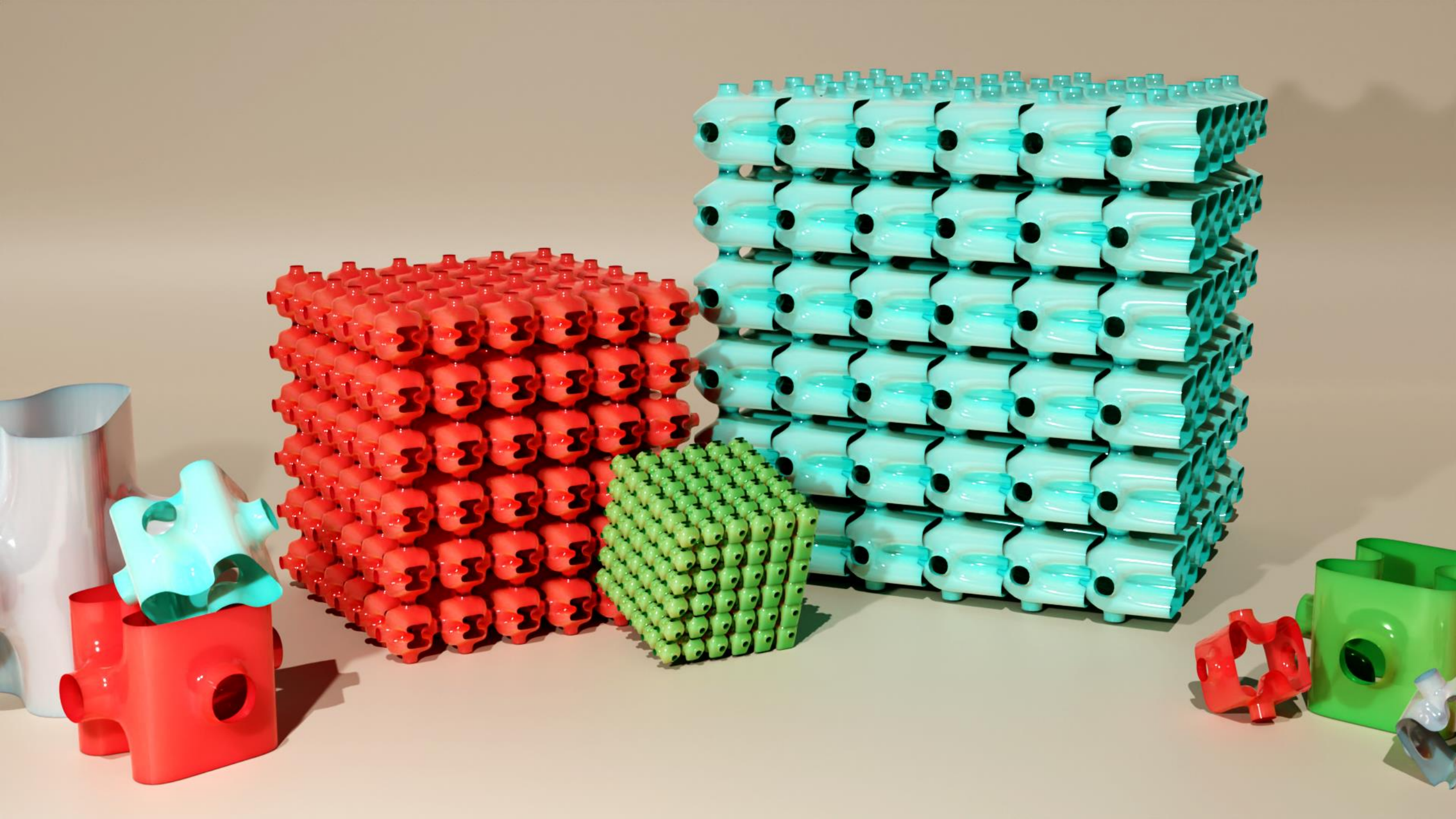


2. Components with minimal surfaces created unconnected geometry






2128
¼
½
¾




THANK YOU !