Portfolio

Joonhaeng Lee

192 combinations of three different Korean letters on the surface of a cube - Joonhaeng Lee written in Korean.

Joonhaeng Lee

Email: joonhaenglee.korea@gmail.com
Skype: joonhaenglee.korea
Cell: +82 10 3783 9002

STUDY INTEREST

Mass Customization and Fabrication of Architectural Objects
Web-Based Generative Design
Folding Geometries Using Standardized Flat Materials

EDUCATION

2009 - 2016 Yonsei University, Seoul, Korea
Bachelor of Engineering, Architecture Design

2014 National University of Singapore, Singapore
School of Architecture, Exchange Student

ACHIEVEMENTS

2017 Google HackFair 2017 Seoul Exhibition
Toilet control system for high-rise building using motion sensors with raspberry

2016 5th UAUS Pavilion Competition, Grand Prize from City Mayor
10 square meter size pavilion made of reused coffee ground brick

2015 S Design Award
A selected project of Human Environment & Design department's design festival
Interactive bulletin board using Twitter API

2009 - 2012 National Science & Technology Scholarship
A full-ride scholarship as student excellence in mathematics and science

SKILLS

Rhino&grasshopper, Catia(Digital Project), Revit&Dynamo, AutoCAD
Processing, Python scripting
Laser cutting, CNC milling

LANGUAGES

Proficient English, Basic Spanish, Native Korean

PROFESSIONAL WORK EXPERIENCE

Syntegrate - BIM Consultant

Duties included computational design, façade panelization, automated 2D fabrication drawing generation, fabrication data extraction, custom script development

Projects

2018 Paradise City Spa & Entertainment, Incheon, Korea - *built*
Parametric modeling and fabrication data creation for aluminum panels and sub-structures;

Posco 50th Anniversary Sculpture, Seoul, Korea
Parametric modeling of sculpture, panel shape and size optimization

Louis Vuitton Flagship Store, Seoul, Korea - *on construction*
Parametric modeling of façade; BIM coordination model integration

2017 Pankyo Alpha Dome City, Pankyo Korea - *built*
Generative modeling of 3,000 unique panels and sub-structure, automatic fabrication drawing creation of panels and sub structure

Toyota Corolla Center, Osaka, Japan - *built*
Parametric modeling of curved roof and main structure; roof thickness optimization; clash detection and coordination

The Mount Fuji Heritage Center, Shizuoka, Japan - *built*
Fabrication model creation and CNC cutting path generation of 7000 unique planks of façade

2016 EF Study, Research project
Development of facade prototyping grasshopper tool for architects

Gwangmyeong Cave Master-plan, Gwangmyeong, Korea - *on construction*
Integration of cave scan data and building data; parametric modeling of 100m long time capsule exhibition hall on scan data

Abstract

Ever since I was a kid, I have loved making things by hand- clay toys, Lego, plastic model cars and anything that I could get my hands on. I have also always wondered about the limitations of architecture, for instance how and what was the largest object that could be created, which had led me on a lifelong discovery and study of architecture design. In university, I had the opportunity to work on projects which involved the creation of two human-scale pavilions, which allowed me to gain a deeper appreciation and understanding of the intricate relationship between making and computation.

Those projects inspired me to start my professional career as a BIM consultant, where I have had the opportunity to engage in meaningful projects and relationships with my clients. The happiest moments of my career was watching my clients interact and tinker with my generative designs, which in turn led them to dream up new designs and possibilities that they never imagined possible. For this reason, I am looking to further pursue my education so as to achieve my ambition of designing and publishing generative designs for a greater number of people.

This portfolio has two chapters. Each of the sections begins with a pavilion fabrication project, which was my starting point in utilizing computational methods for fabrication and customization. All of the showcased projects in this portfolio are also related to how I applied computational methods for the fabrication of atypical facade parts and for the customization of design objects.

Contents

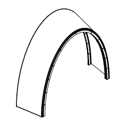
Chapter 1

Fabrication of Building Assembly Parts

When a designer begins drawing on a CAD program - whether it is a line, curve or surface, the visual representations of those objects are inherently mathematical and can be represented by numbers and formulas.

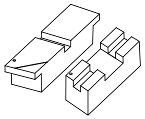
Applying computational methods was exciting for me as I realized that I could extract these quantitative data from geometry and utilize them for overcoming real-world constraints in fabrication - I could now automate repetitive data generation processes to save time, nest geometries to minimize material loss and optimize design data to fulfill the size constraints of fabrication machines.

Given that the shapes of the assembly parts were entirely different in each of the facade projects that I had encountered, I learnt and self-taught myself programming skills to deal with each of these different situations.



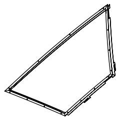
01 Precast Concrete Pavilion (2015)

From sketch to portable precast pavilion.



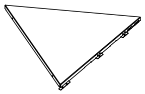
02 Fabrication of 6,969 Unique Planks (2017)

Automation of fabrication data creation process



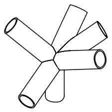
03 Curvy Facade Made of Flat Aluminum Sheets (2017)

Fabrication of facade panels for Alphadome-City



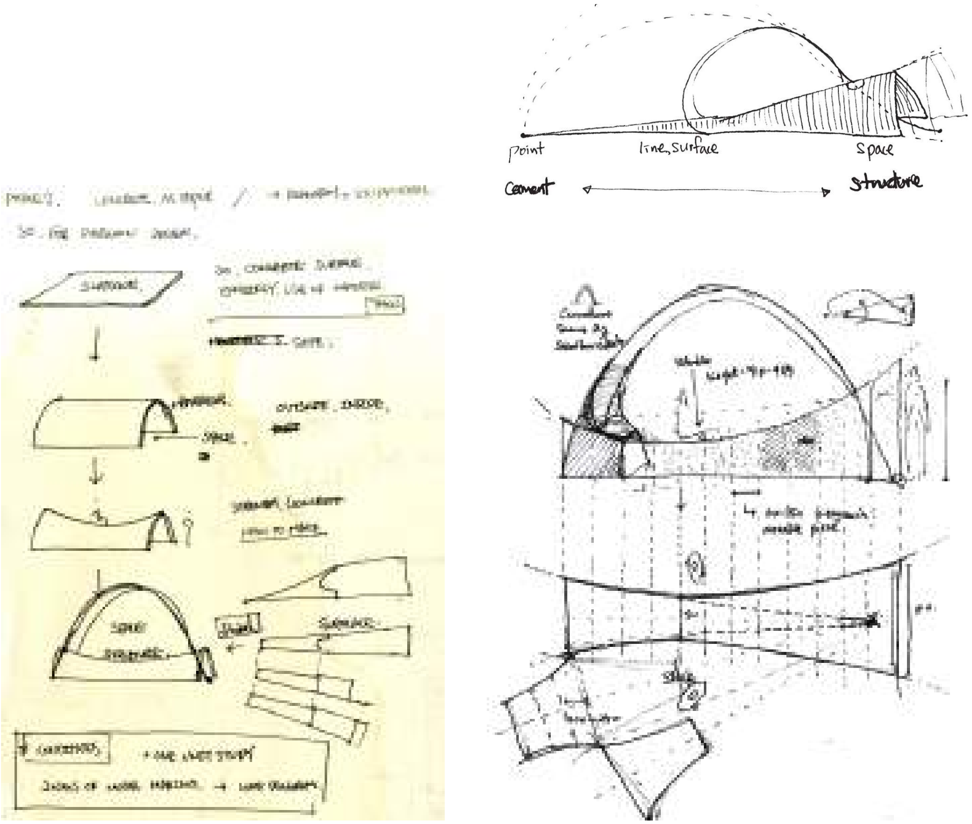
04 From Design Surface to Triangular Panels (2018)

Design and fabrication of Paradise City Chroma interior roof



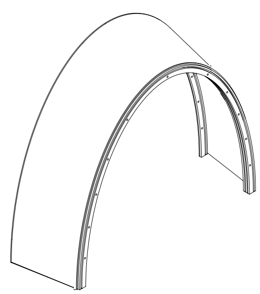
05 Frank Gehry’s Louis Vuitton Flagship Store (2018)

Design development, fabrication and BIM consulting



|| Design Concept Sketches - point, line and space

<u>01</u> Precast Concrete Pavilion



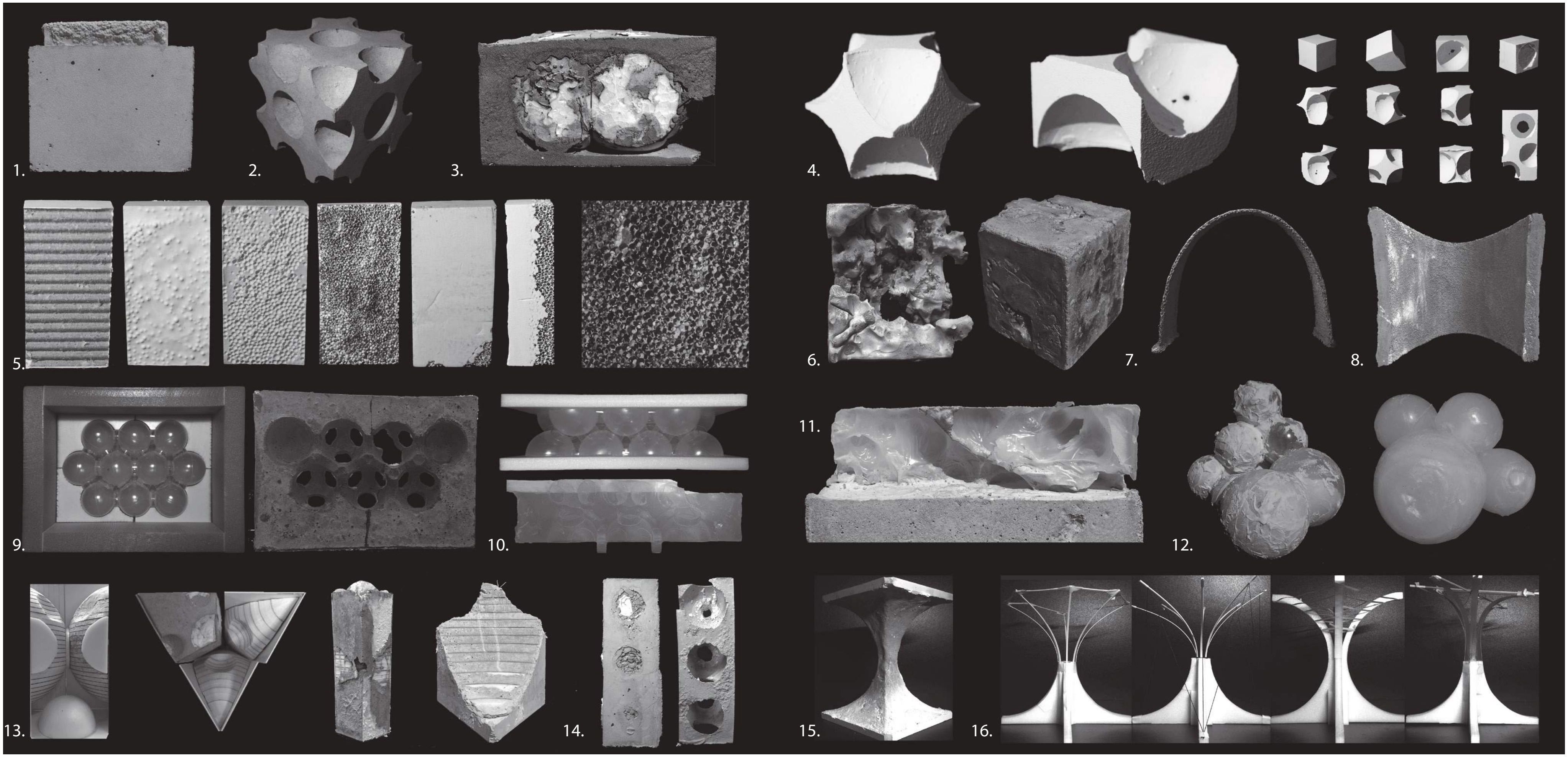
Academic Project, Bachelor Thesis Project
2015.3 - 2015.6, 3 months

Building: 10 m^2 size pavilion

Workscope :
1) Design and fabrication of concrete pavilion.
2) Design assembly and pre-fabrication method for 5-day exhibition.

Background :
After I finished seven architectural design studios focusing on what makes a design good or bad, I was intrigued by the process of how design sketches get translated into actual buildings. Hence, in order to fully experience a design - building process, I constructed a human scale pavilion for my bachelor thesis project.

|| Portable precast concrete design and assembly as shown in the exhibition hall. 2015.6.



|| Concrete pre-fabrication study models to create curved space.

Material / Mold / Remark

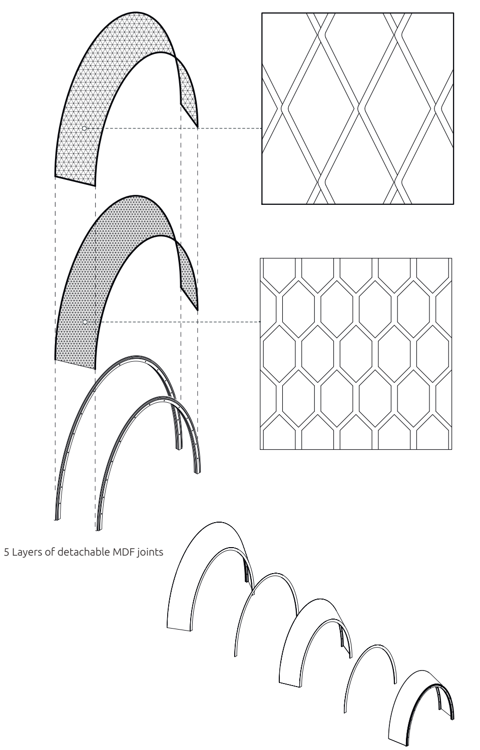
1. Concrete / Sealed empty box inserted in a mold / Make object light
2. Concrete / Sphere form attached on rectangular mold / Create texture
3. Concrete / Sphere form inserted in a mold / Create circular hollow section
4. Gypsum / Mold with sphere geometry / Create circular pattern
5. Gypsum / Mold with various textures on surface / Create texture on gypsum
6. Concrete, paraffin / Rectangular mold / Extract paraffin to make a random texture
7. Cement and sand / Single layered mesh /Ferro-cement, plaster cement on mesh mold
8. Cement and sand / Double layered mesh /Ferro-cement, plaster cement on mesh mold
9. Concrete / A custom mold made of ball pool balls / Create a transparent pattern
10. Paraffin / Mold made of sealed ball pool balls / Create a transparent pattern
11. Concrete, paraffin / MDF plates / Create random texture
12. Gypsum / plastic balls / Create a circular form using standardized material
13. Concrete / Modularize mold made of form / Create three modularized concrete block
14. Concrete / MDF mold with sphere form / Create hollow column
15. Gypsum / Form / Create elastic form
16. Gypsum / Form and wood sticks / Combine gypsum with other materials
17. Concrete / MDF, two layers of Mesh / Selected method to fabricate curvy pavilion

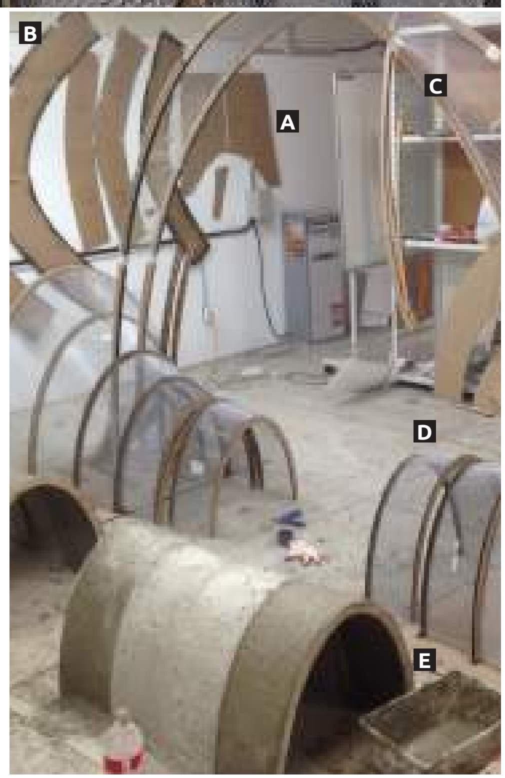
Studying material characteristics of concrete :

To explore the material characteristic of concrete and the mold design method needed to express atypical shapes, I created various study models by hand. I used cement with sand as well as gypsum and paraffin to make models efficiently. I also concentrated on forming a mold with different materials to study texture creation, modulation and assembly method during the studies.

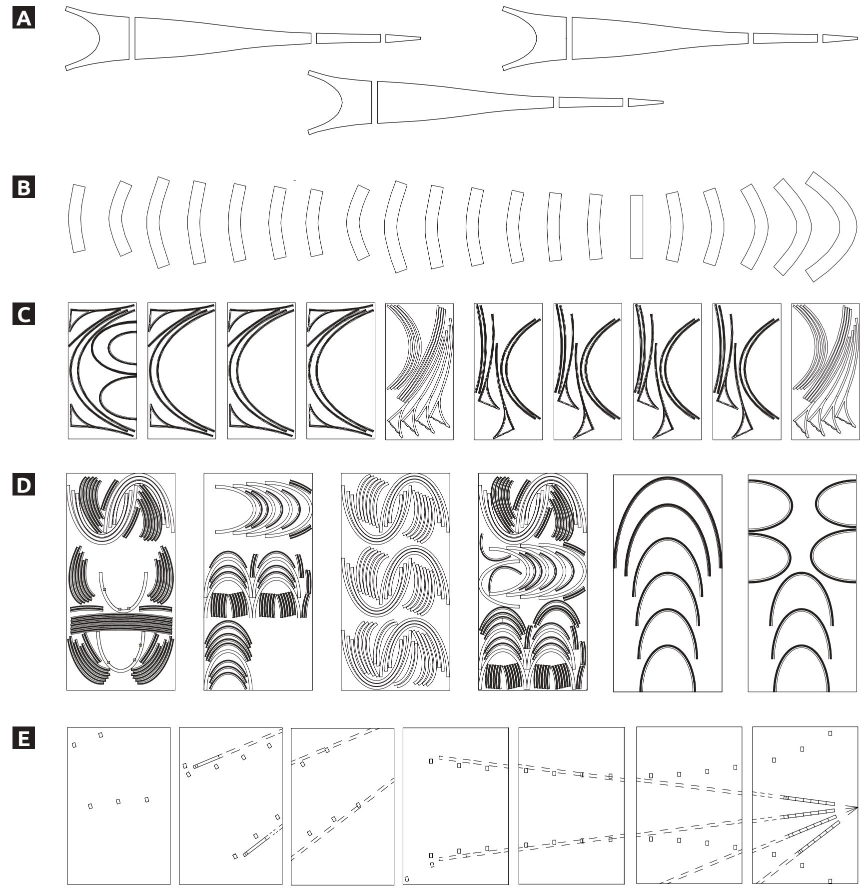
I spent three whole weeks studying the concrete and fabrication method. Eventually, I found the fabrication method named 'Ferro-cement' which Pier Luigi Nervi used for his buildings a while ago. I borrow the concept of this plastering method on the study model no.7,8,17 for the creation of curvy geometry. Afterward, these study models were developed to make an arch shape mold made of two layers of mesh and five layers of MDF panels for the fabrication of the precast concrete block for the pavilion.

|| 17. Mold made of MDF plates and mesh to create light and curvy pre-fabricated concrete block.





|| Pre-assembled mold with 2 layers of mesh and 5 layers of MDF, 2015.5.



|| All 2D geometries to laser cut for fabrication and construction.

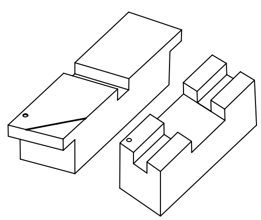
Design, Fabrication, and Construction :

Completing all the elements of the pavilion in three months was very challenging - I had to choose appropriate building materials to use, while managing the expenses, construction time, and the logistics for the installation.

The experience that i gained through the of the fabrication and construction of pavilion further sparked my interest in the fabrication and construction projects of atypical three-dimensional buildings.

|| Interior wall of Mount.Fuji Heritage Center, 2017.8.

02 Fabrication of 6,969 Unique Planks

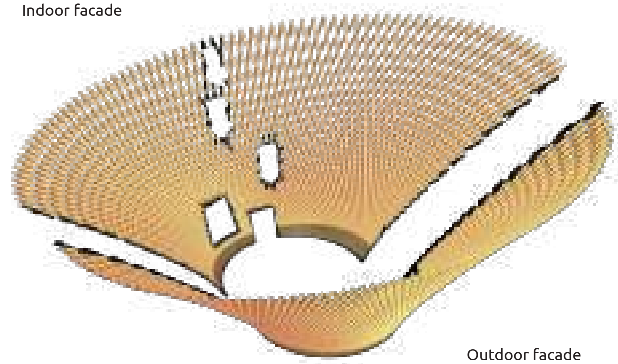


Professional Project , Syntegrate
2016.12 - 2017.3 , 4 months

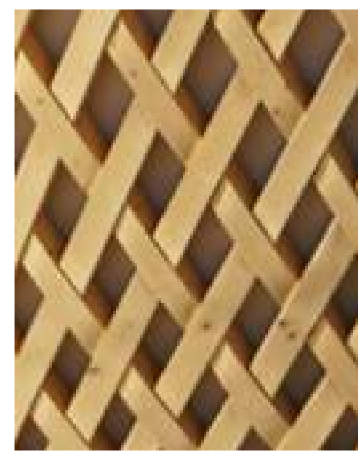
Building: Mount. Fuji Heritage Center, Japan
Architect : Shigeru Ban Architects
Client : Shelter, Timber fabricator

Workscope:
1) Create 6,969 facade planks layout into 140mm*140mm*4M(3M) size timber.
2) Create cutting zone of each unique planks to assign pathway to machine.

Client Remarks and Requirements :
"Joonhaeng, it is impossible to create Building Transfer Language (BTL) paths for all of the 7,000 pieces of planks in the allotted construction schedule. It will take a year to create all that data. Moreover, we need to order timber next week to meet the construction schedule. Do you have any ideas how i can estimate the number of timbers for fabrication and create BTL paths automatically?"

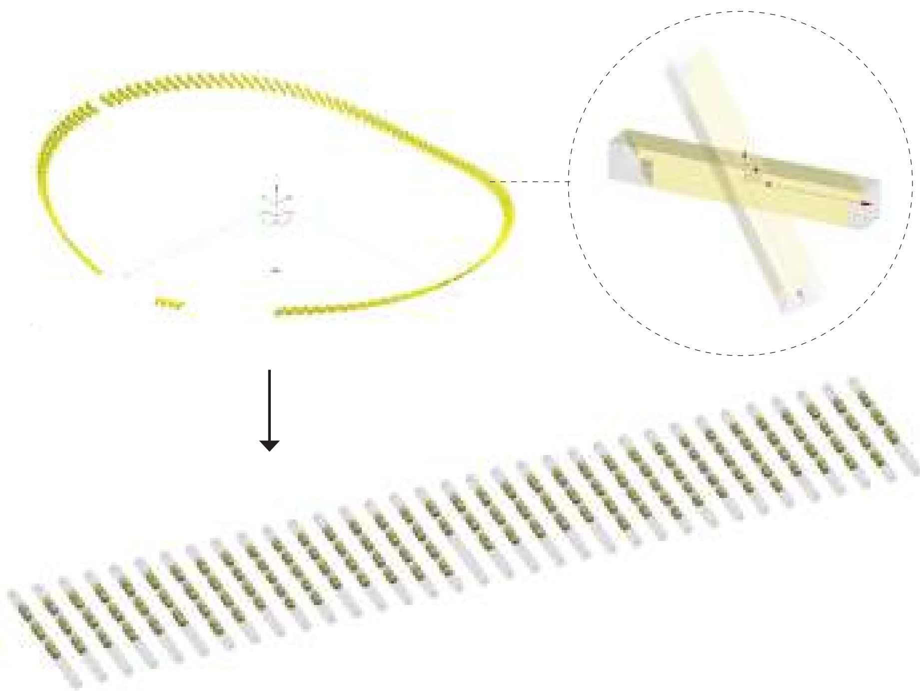


*The initial design model is provide by Shelther.
*The type of planks are classified by Joonhaeng Lee.

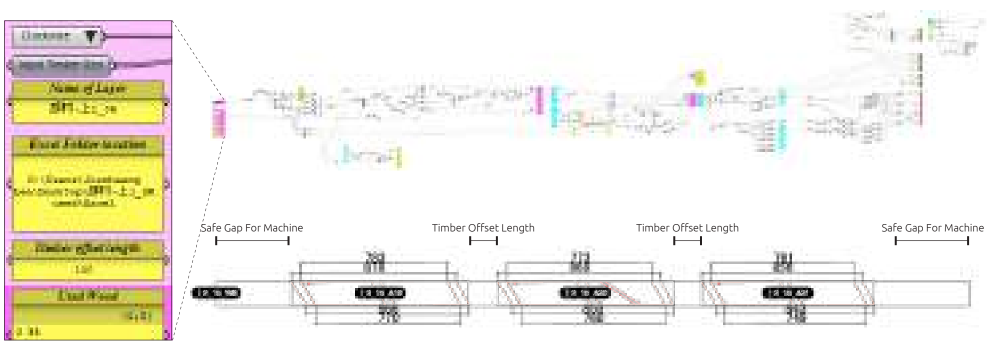


|| Master model of five types planks. Precise cut applied to create woven pattern design intent.

Relocation of Design Model Planks Through Automation Script



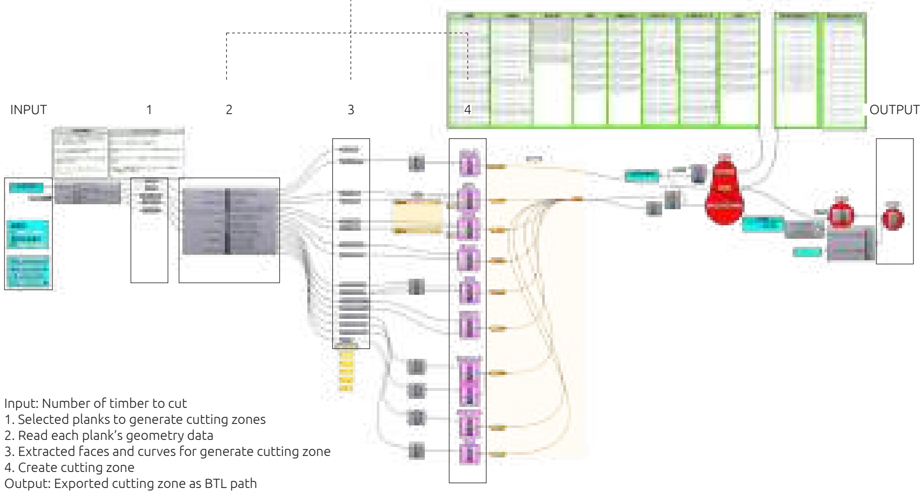
|| Input: A row of planks (7th row of T2 type planks image).
|| Output: Relocated planks in 4 or 3 meter-long timber.



|| Output of geometry contains annotation data for inspection and installation reference.

|| 2,511 T2 type planks relocated in 1,318 4 or 3 meter-long timbers (35% of whole facade)

Automation - A Algorithm to Cut More than 1,000 Planks



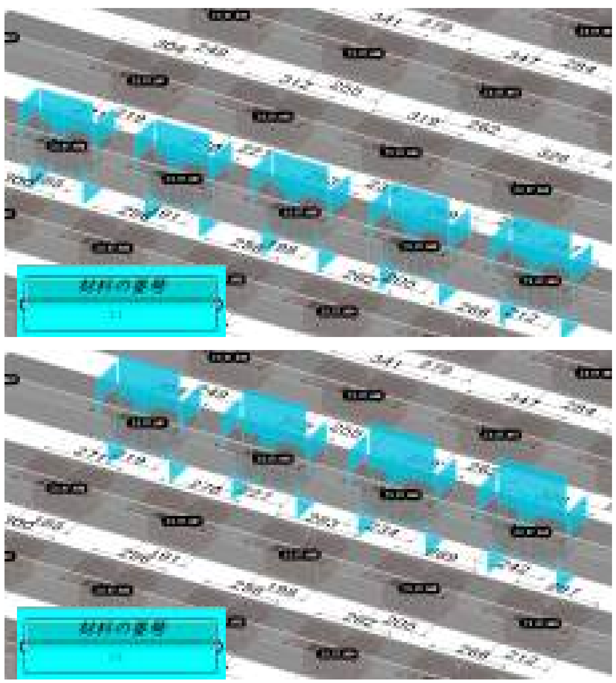
Input: Number of timber to cut
1. Selected planks to generate cutting zones
2. Read each plank's geometry data
3. Extracted faces and curves for generate cutting zone
4. Create cutting zone
Output: Exported cutting zone as BTL path

|| Plank Reader - Classifies faces and edges of planks in input timber and create cutting zones.

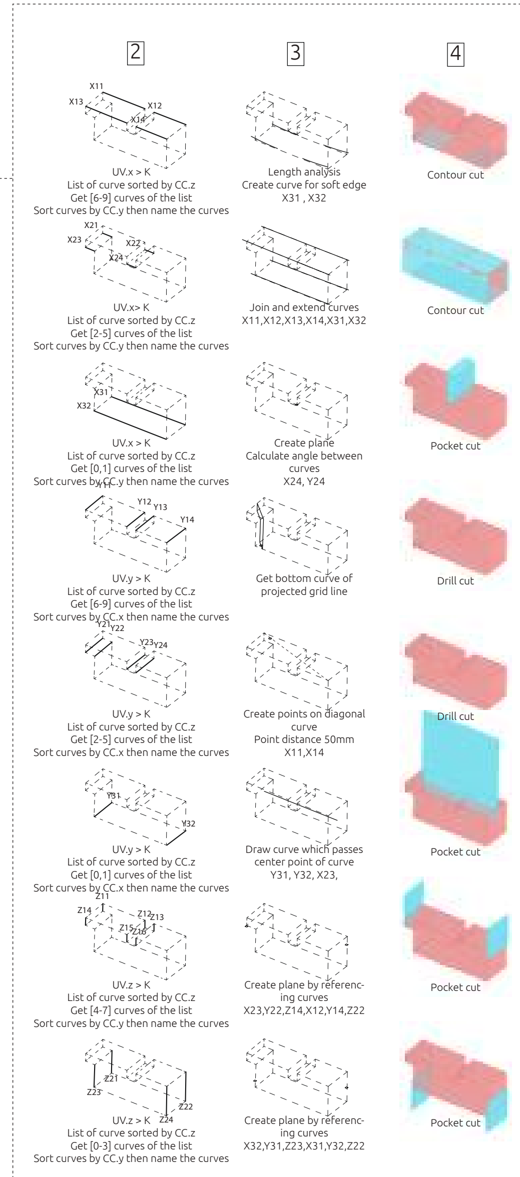
Client Shelter use Woodpecker plugin to create Building Transformation Language (BTL) paths for executing their custom cutting machine. The plugin converts input point, curve, and plane data to cutting zones.

In this project, it was impossible to create the input data manually because the shapes of each plank are entirely different. It took approximately 15 minutes to make and assign input geometries for creating BTL path. Moreover, during the manufacturing process, a cutting method was changed several times to fulfill the additional requirements from the architect and owner.

In this situation, I developed the code that could read the geometrical data of planks and creates BTL paths automatically. I delivered he method for fabricating all 4,778 individual planks in only 4 algorithms.



|| Only typing number of timber to create BTL path.



|| Geometry classification by sorting XYZ values

|| BTL path extracted from the code

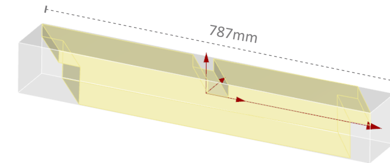
|| Shelter Factory, Yamagata, Japan, 2017.4

Nesting Algorithm Development

- How can I optimize the combinations of planks to minimize material loss?

Planks Length List [16] =

[1025,1031,1038,1045,1052,1059,1066,1073,1079,1084,1088,1091,1091,1090,1087,1083,1076,1067,1057,1045,1031,1015,998,980,961,940,919,897,875,852,830,808,787,766,747,729,713,656,654,654,656,661,667,676,687,699,713,729,746,765,785,807,829,851,873,895,915,934,950,964,976,986,994,1001,1007,1013]



Planks Length List[16][32] = 787

Algorithm 1 OUTPUT

“Good, but too much space wastage”

1) Place planks in 4 meter-long timber one by one.
2) If there are no space to place planks, put it in the next timber.
3) If a filled timber can be replace by 3m, replace it.

Timber Placement	Index	Timber Length	Waste
	0,1	[1025, 1031]	994
	2,3	[1038, 1045]	967
	4,5	[1052, 1059]	939
	6,7	[1066, 1073]	911
	8,9	[1079, 1084]	887
	10,11	[1088, 1091]	871
	12,13	[1091, 1090]	869
	14,15	[1087, 1083]	880
	16,17	[1076, 1067]	907
	18,19	[1057, 1045]	948
	20,21	[1031, 1015]	1004
	22,23	[998, 980]	1072
	24,25,26	[961, 940, 919]	80
	27,28,29	[897, 875, 852]	276
	30,31,32	[830, 808, 787]	475
	33,34,35	[766, 747, 729]	658
	36,37,38,39	[713, 656, 654, 654]	73
	40,41,42,43	[656, 661, 667, 676]	90
	44,45,46	[687, 699, 713]	801
	47,48,49	[729, 746, 765]	660
	50,51,52	[785, 807, 829]	479
	53,54,55	[851, 873, 895]	281
	56,57,58	[915, 934, 950]	101
	59,60	[964, 976]	1110
	61,62	[986, 994]	1070
	63,64	[1001, 1007]	1042
	65	[1013]	2187

Algorithm 2 OUTPUT

“Improved efficiency, but still room for improvement”

1) Place planks in 4 meter-long timber one by one.
2) If there is no spaces left to place planks, find a plank in the list which is shorter than the left area. If there is, place the plank.
3) Repeat process...

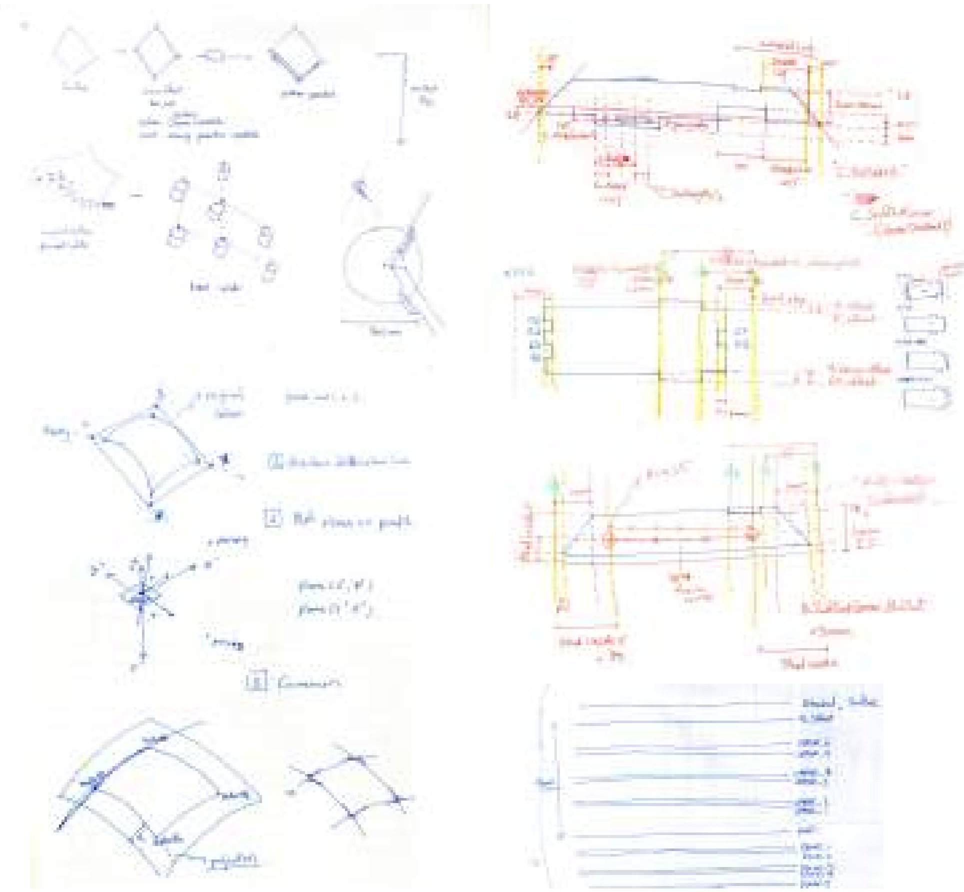
Timber Placement	Index	Timber Length	Waste
	0,1,30	[1025, 1031, 830]	14
	2,3,31	[1038, 1045, 808]	9
	4,5,32	[1052, 1059, 787]	2
	6,7,34	[1066, 1073, 747]	14
	8,9,35	[1079, 1084, 729]	8
	10,11,36	[1088, 1091, 713]	8
	12,13,46	[1091, 1090, 713]	6
	14,15,47	[1087, 1083, 729]	1
	16,17,48	[1076, 1067, 746]	11
	18,19,50	[1057, 1045, 785]	13
	20,21,29	[1031, 1015, 852]	2
	22,23,26	[998, 980, 919]	3
	24,25,27	[961, 940, 897]	102
	28,33,37	[875, 766, 656]	603
	38,39,40,41	[654, 654, 656, 661]	125
	42,43,44,45	[667, 676, 687, 699]	21
	49,51,52	[765, 807, 829]	499
	53,54,55	[851, 873, 895]	281
	56,57,58	[915, 934, 950]	101
	59,60	[964, 976]	1110
	61,62	[986, 994]	1070
	63,64	[1001, 1007]	1042
	65	[1013]	2187

Algorithm 3 OUTPUT

“Perfectly optimized”

1) Create all possible combination of planks.
2) Find a plank combination which creates the shortest waste.
3) Place that combination and remove combinations which contains already placed planks.
4) Repeat process...

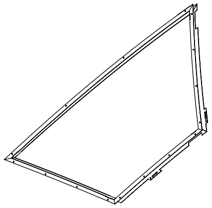
Timber Placement	Index	Timber Length	Waste
	11,8,35	[[1091, 1079, 729]	1
	12,63,51	[1091, 1001, 807]	1
	13,23,52	[1090, 980, 829]	1
	10,3,33	[1088, 1045, 766]	1
	14,15,47	[1087, 1083, 729]	1
	9,64,31	[1084, 1007, 808]	1
	16,2,50	[1076, 1038, 785]	1
	17,19,32	[1067, 1045, 787]	1
	4,58,27	[1052, 950, 897]	1
	0,25,57	[1025, 940, 934]	1
	22,61,56	[998, 986, 915]	1
	49,43,38,39	[765, 676, 654, 654]	1
	36,46,42,37	[713, 713, 667, 656]	1
	5,59,28	[1059, 964, 875]	2
	1,21,29	[1031, 1015, 852]	2
	20,62,54	[1031, 994, 873]	2
	6,60,53	[1066, 976, 851]	7
	65,24,26	[1013, 961, 919]	7
	7,18,34	[1073, 1057, 747]	23
	45,44,41,40	[699, 687, 661, 656]	47
	55,30,48	[1045, 980, 896]	129



|| Study of generative stiffener creation logic and setting out parameter.

|| Alpha Dome City South Facade, 2018.5.

<u>03</u> Curvy Facade Made of Flat Aluminum Sheets

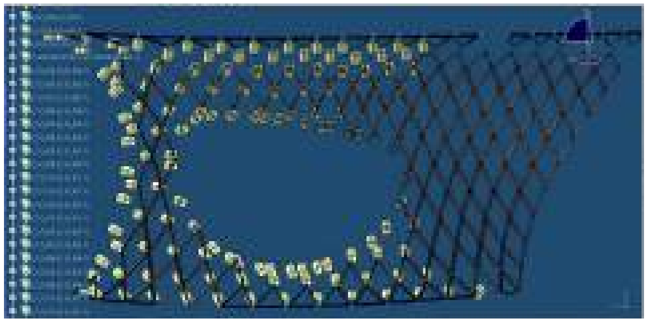


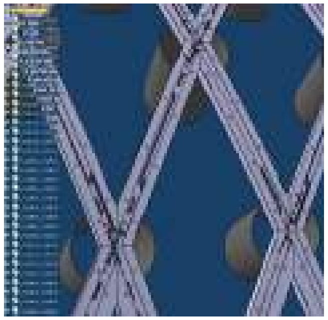
Professional Project , Syntegrate
2017.12, 1 month

Building : Alpha Dome City, Pankyo, Korea
Architect : Junglim Architects
Client : Steellife, Aluminum Facade Fabricator and Installer
Supervisor :
Seonwoo Kim, Director, Syntegrate,
Honghyun Kim, Senior BIM Consultant, Syntegarte
Daeyeol Ju, Fabricator, Steellife

Workscope:

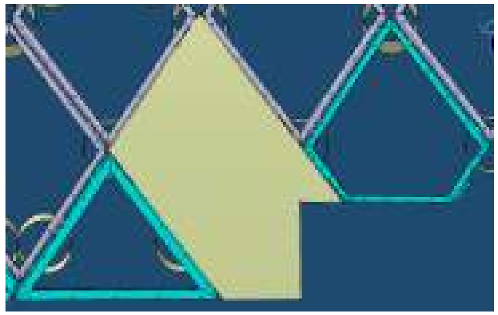
1) Application of rectangular panel Catia Template on regular panels on South facade.
2) Design Catia Power Copy for stiffener geometry creation of non-rectangular shape panels.
3) Create fabrication drawings of the panels.

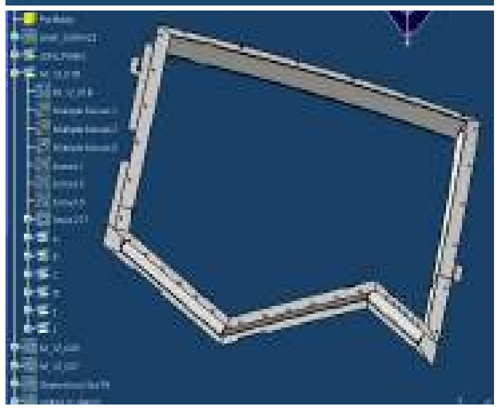




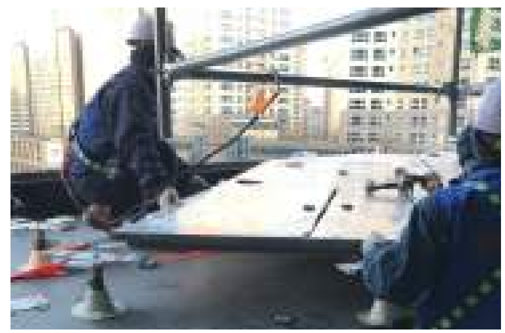
|| BIM model for fabrication - Application and adjustment sttifner and the joint hole.

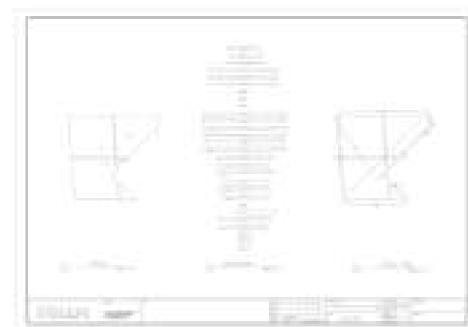
*The initial design surface is provided by Steellife.



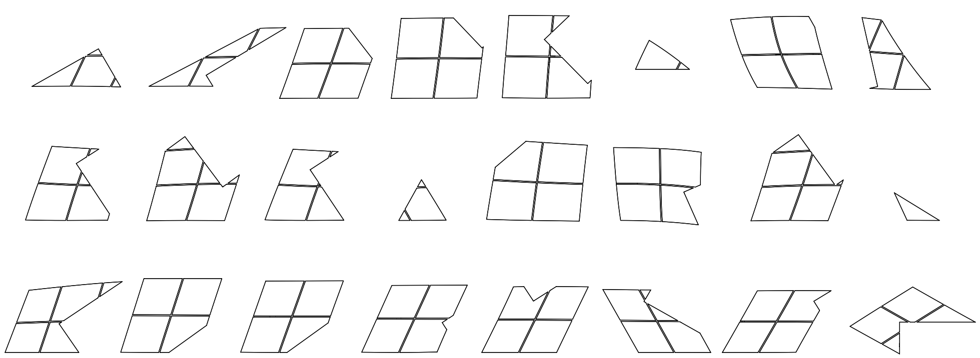


|| Generative stiffeners and three curve inputs





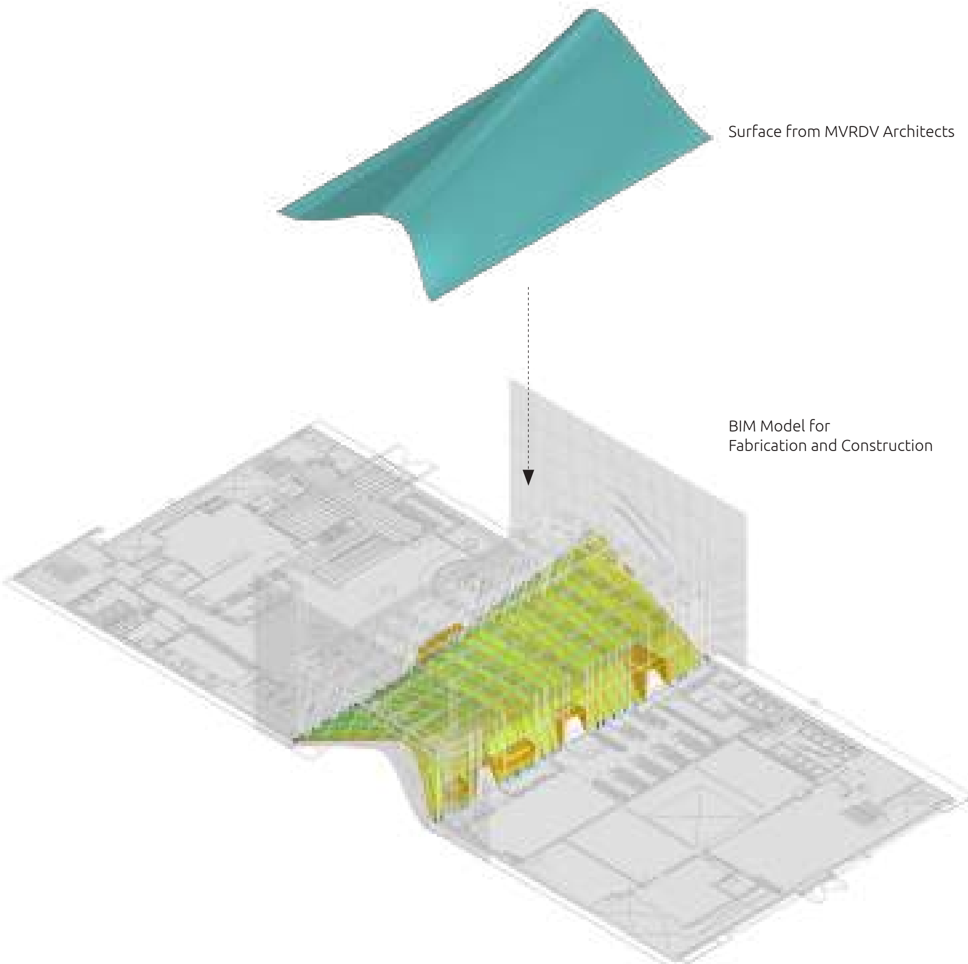
|| Factory visit to check manufactured stiffeners, Installation of panels on site. 2017.12.



|| Create 3D geometry and 2D laser cut curves of Irregular trimmed panel beside structure columns

Is it possible to convert any 3D geometry into 2D planar geometry for fabrication using a laser cutter?

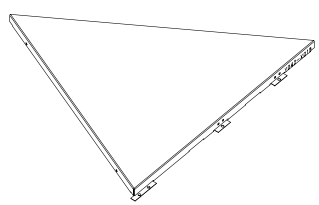
After finishing this project, I realize that the fabricator made all of the facade assembly parts with laser cut pieces of flat 3mm thick aluminum sheets. He knows by his 20 years of experience that flat sheets and laser cut manufacture are enough for making panels of this facade design. I thought that the conversion of 3D to 2D is a fascinating topic for making curvy objects. What if product designers could upload the model of NURBS surface on the web page and download a corresponding planar figure which allows them to create the prototype of their geometry easily? Which math or computer science subjects should one learn to convert 2D to 3D, and 3D to 2D freely?



04 From Design Surface to Triangular Panel

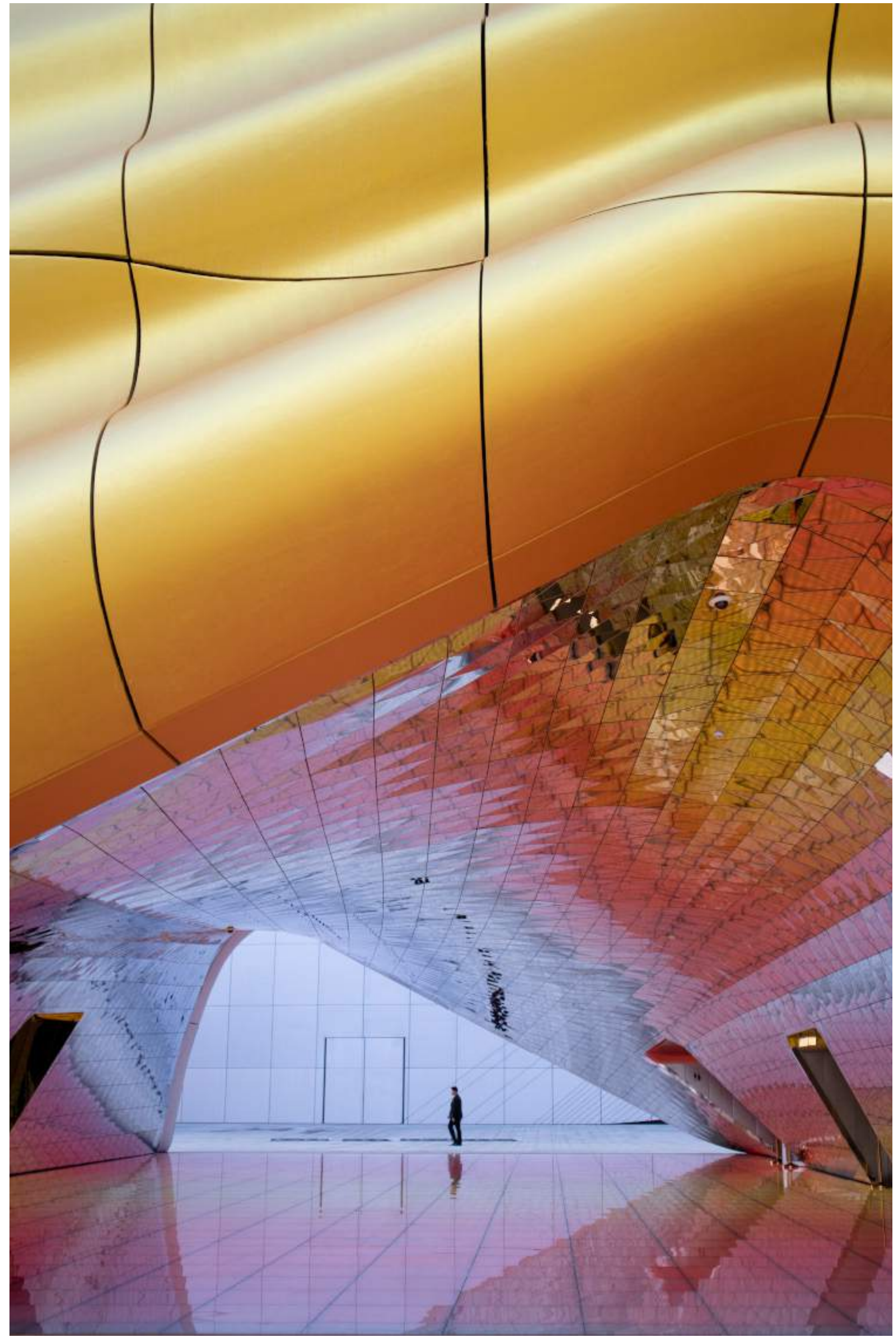
Professional Project , Syntegrate
2018.2 - 2018.4, 3 months

Building : Paradise City Hotel Chroma, Incheon, Korea
Architect : MVRDV
Client : Steellife, Aluminum fabricator and installer
Supervisor : Kwangchun Park, CEO of Steellife
Coworker : Heejin Chae, BIM Consultant, Syntegrate

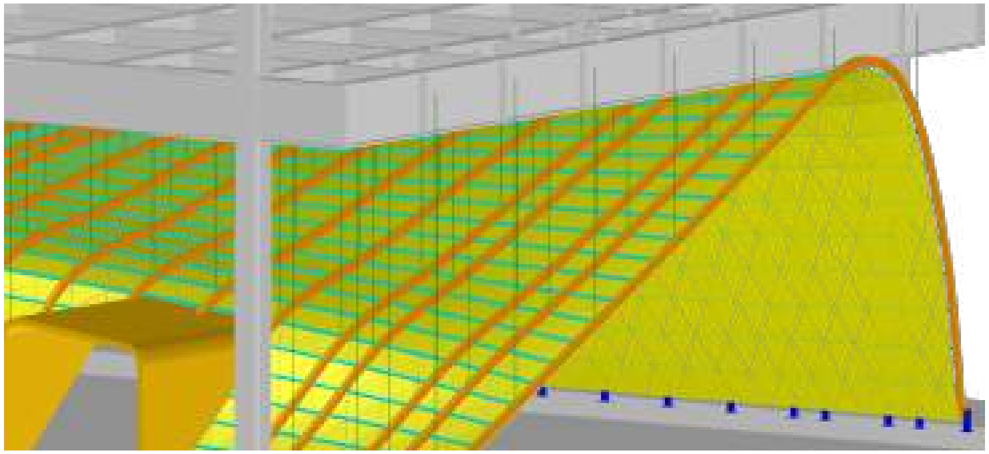


Workscope:

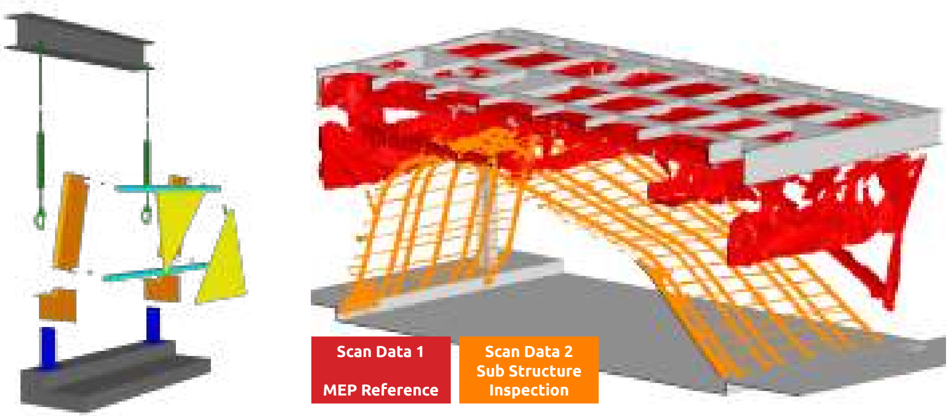
1) Design development of the interior roof of the building.
2) Flat triangular panelization of design surface and sorting of panel data.
3) Fabrication data creation of substructure.



|| Paradise City Hotel Chroma Interior Roof, 2018.11



|| Creation of BIM model for fabrication. All geometries have names, lengths, and areas for utilization.

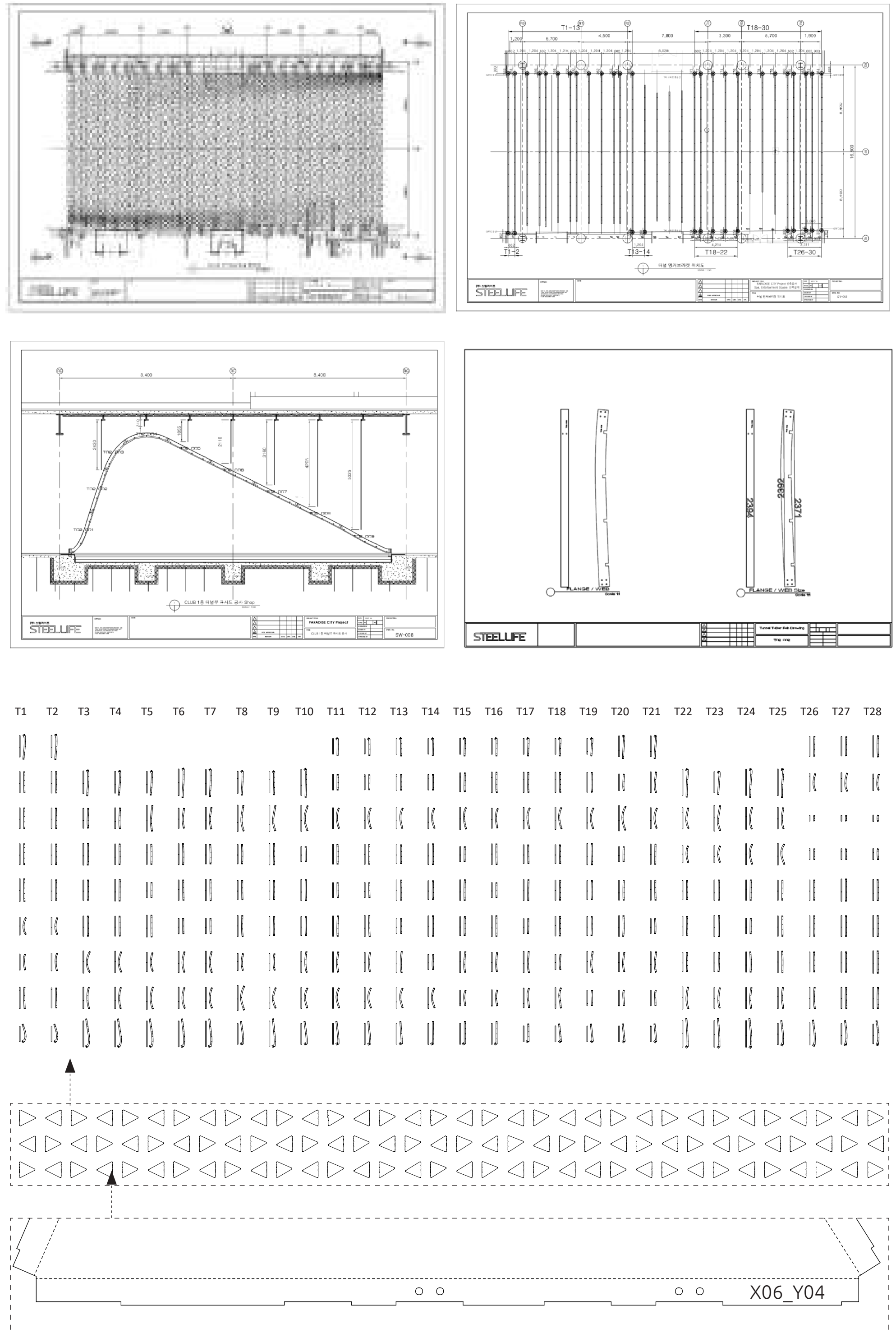


|| Substructure concept model. Integration and utilization of scan data for design development.

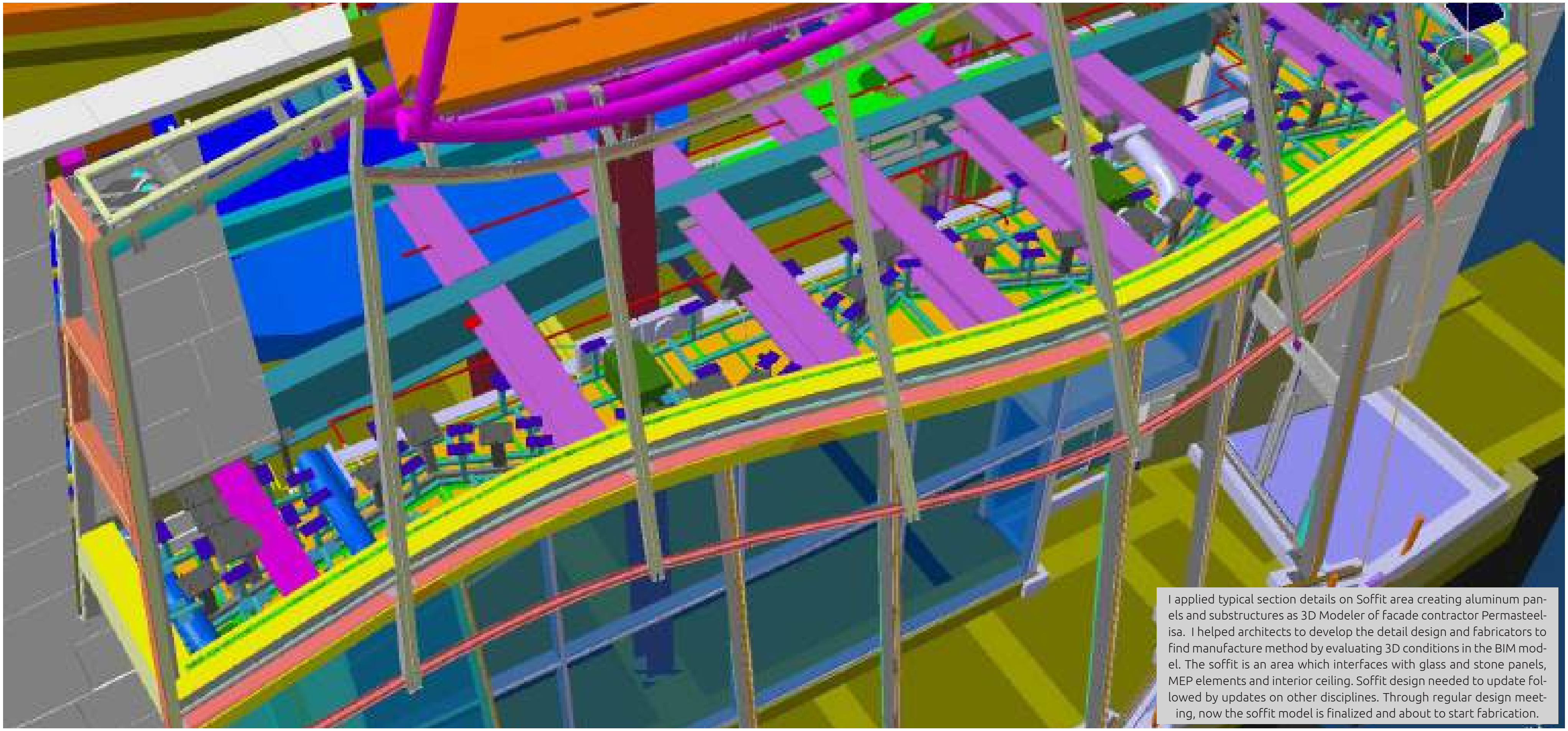
Client Remarks and Requirement :

"The architect gave us only a single lofted surface and asked us to create every single data for design and fabrication. We need to create a BIM Model for design development and construction. Let's optimize the size of the triangular panel to meet raw aluminum sheet size to fabricate eight panels on each aluminum sheets. Then, let's convert a smooth surface into segmented surfaces to minimize the depth of substructure system."

|| Paradise City Hotel Chroma Interior Roof, 2018.11



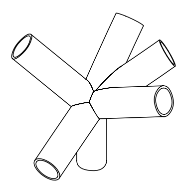
|| Construction drawings and fabrication drawings for laser cut.



I applied typical section details on Soffit area creating aluminum panels and substructures as 3D Modeler of facade contractor Permasteelisa. I helped architects to develop the detail design and fabricators to find manufacture method by evaluating 3D conditions in the BIM model. The soffit is an area which interfaces with glass and stone panels, MEP elements and interior ceiling. Soffit design needed to update followed by updates on other disciplines. Through regular design meeting, now the soffit model is finalized and about to start fabrication.

|| Soffit BIM model Integrated with structure, MEP and facade model. All 3D geometries of Soffit panels and substructures are created by joonhaeng Lee under guidance of supervisor Gianpaolo Mancuso.

<u>05</u> Frank Gehry's Louis Vuitton Flagship Store

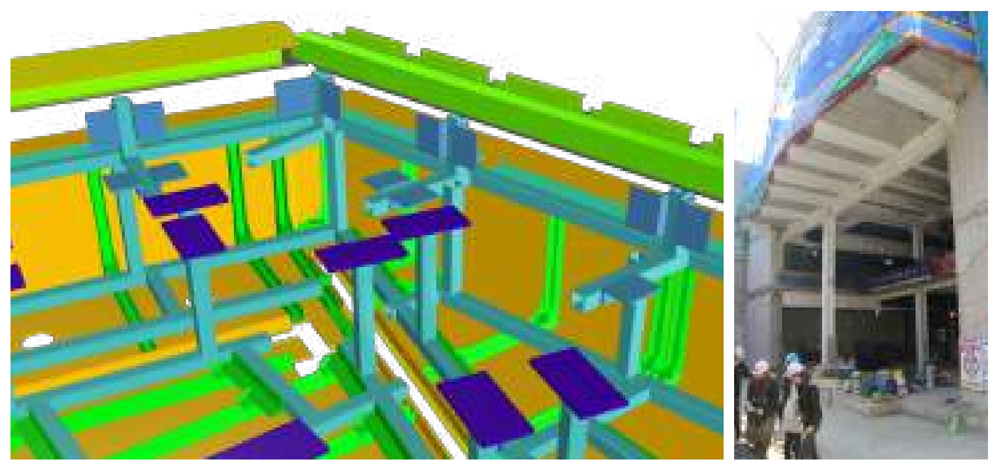


Professional Project , Syntegrate
2018.1- present
Building : Louis Vuitton Flagship Store, Seoul, Korea
Architect : Gehry Partners
Client : Permasteelisa Hongkong

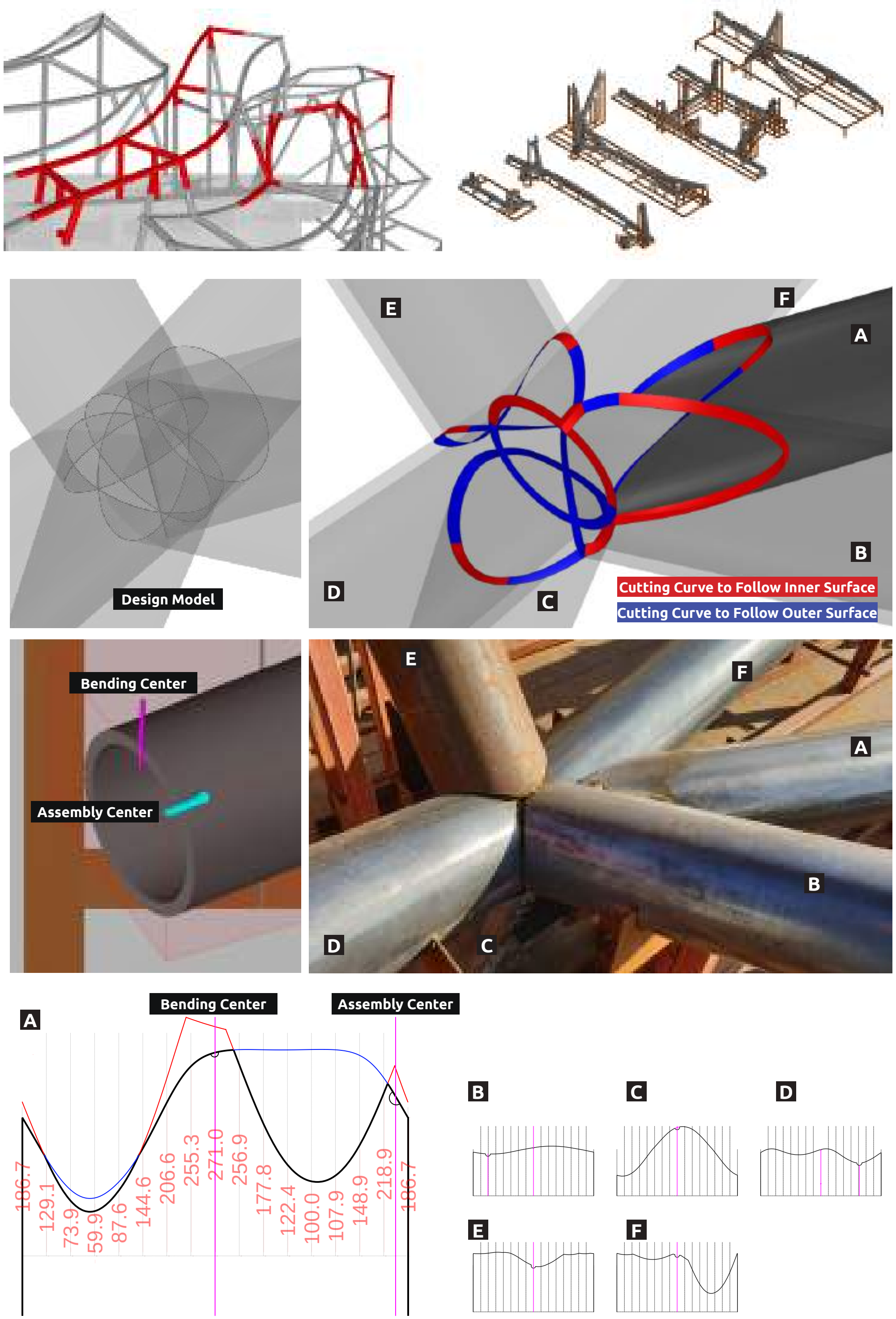
Supervisor :
Gianpaolo Mancuso , Architect, Permasteelisa
Ilya Bourim, Architect, Syntegrate
Denny Kim, Fabricator, Glainno

Coworker :
Bryan Jan Tolentino (2D drawing), Permasteelisa

Workscope :
1) Design development of soffit to minimize a gray area without a clash with other disciplines.
2) Create Circular Hollow Section (CHS) structure fabrication data.
3) BIM coordination on site.



|| Soffit BIM Model. Fourteen panels and substructures are shaped differently. Twenty materials used for substructures which have own names and RGB code in the BIM model. The exact size of plates, the number of aluminum angles, and the length of hollow sections were calculated accurately.



|| Fabrication data creation process of CHS. CHS assembly on jig before wielding.

|| Glainno factory in Jisan and Construction site in Seoul. 2018.12.

Fabrication of Circular Hollow Section (CHS) :

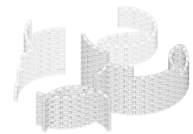
165.2mm diameter with 12mm thickness steel bent CHS hold hundreds of glass panels. The structure also follows the curve of Frank Gehry trademark design. The curvature and length of CHS are all different. Since both ends of CHS meets with an end of other CHS, creating precise cutting curves of each end sections was essential for fabrication. Since workers could cut CHS from the outer surface of CHS using an oxygen cutter, both sides of surfaces, the outer and inner surface of CHS, were calculated to avoid clashes of the inner surface between two CHS.
The building has 40 assembly parts, and each of them contains 5 - 15 CHS. Joonhaeng Lee managed to create fabrication data including a bent center point, curvature radius, edge cutting curves, and assembly jig geometry data of 5 assembly parts under guidance of fabricator Denny Kim, Glainno.

Chapter 2

Mass Customization of Design

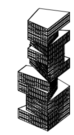
Whenever I create a parametric design algorithm, I would eagerly share the code with my friends and fellow architects and designers. I have always been interested in the co-creation that takes place between the designer and the client/user where they work together to finalize the shapes and the deliverables. With a big smile, my clients would often sit next to me and ask me to alter the input parameters to generate the various interesting design options, that are made easily accessible to them through computation.

What if everyone, from a young kid to an older person, could now create one-of-a-kind furnitures, houses or other architectural objects by simply clicking, dragging and uploading input images on the web? How can I create code that can generate a customized form with an optimized assembly and fabrication method for anyone? How can I elevate and enrich the user experience of playing with geometry to make it fun and intuitive?



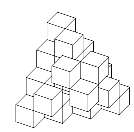
06 Brick Pavilion Maker(2016)

Make a coffee ground brick pavilion in 1,800 USD.



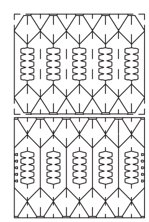
07 Make Your Building in 3 Minute! (2016)

Building Prototype Tool.



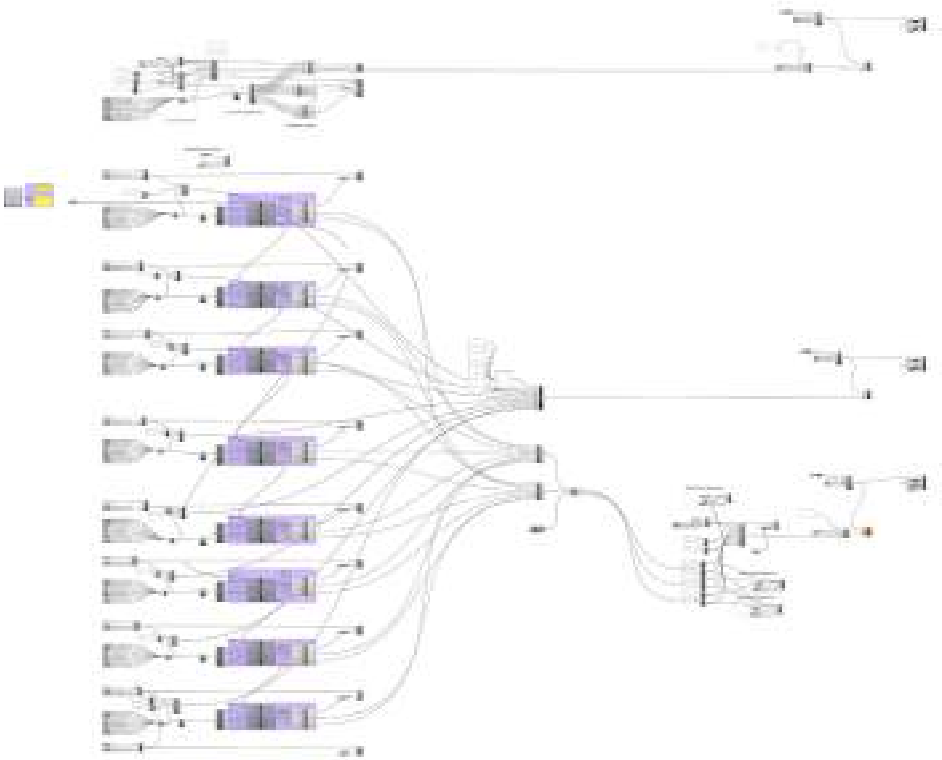
08 Perfect Random Box (2017)

Find your house!

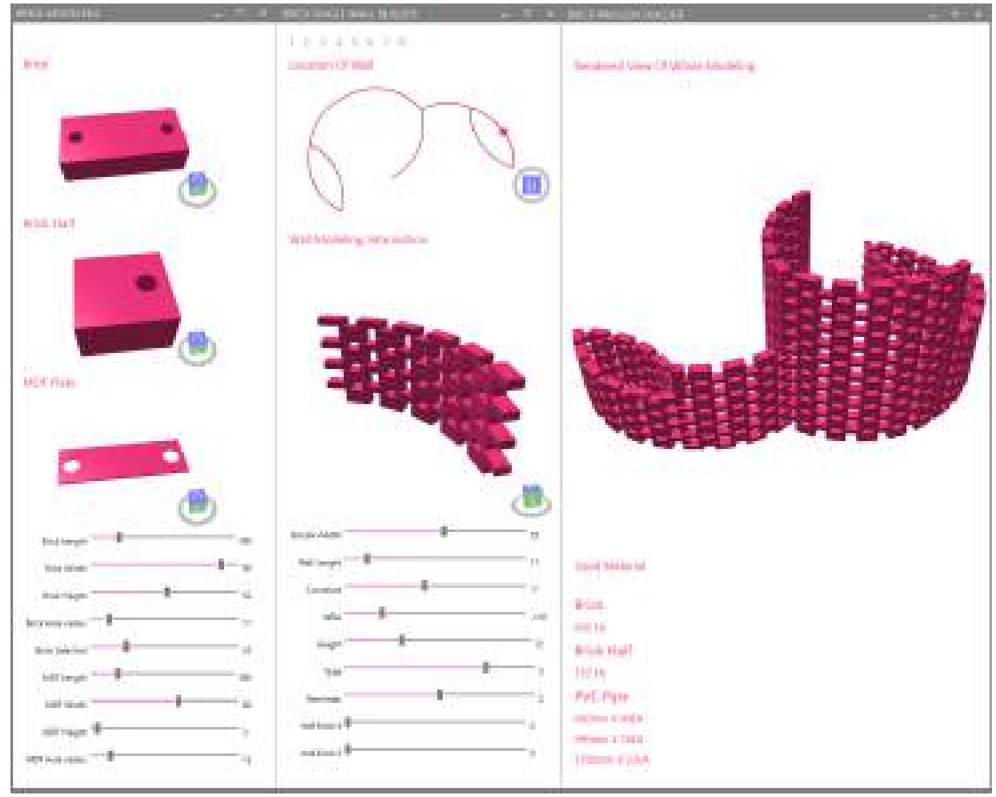


09 New Year Folding Sculpture (2018)

Interlocking paper folding Sculptures designed by users.



|| My first grasshopper project.
The code was designed to be used by ten teammates. User interface window was developed using the Human plugin for easy utilization from our team.



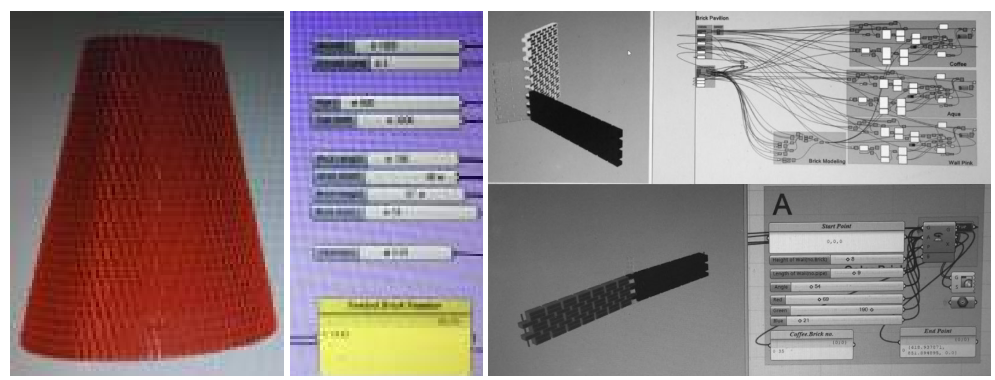
|| User Interface of Brick Pavilion Maker
The size of two types of brick can be modified by pulling number slide bars or inserting integers. It updates the shape of the overall pavilion. The script counts the number of material used for the design. When the number of bricks exceeded the amount our expense allowed, we removed a row of bricks from a selected wall.

08 Brick Pavilion Maker - Creating a pavilion in 1,800 USD

Academic project, Group work
2016,4 - 2018.6 3 Months

Building: 5th Union of Architecture University in Seoul (UAUS) Pavilion
Architect : Yonsei University UAUS Team
Pavilion Design Competition, Grand awards, and my first grasshopper project

Workscope:
1) Design and develop a pavilion made of coffee ground.
2) Create a brick pavilion prototyping tool and estimate cost of the design.
3) Fabricate bricks and construct building on site.



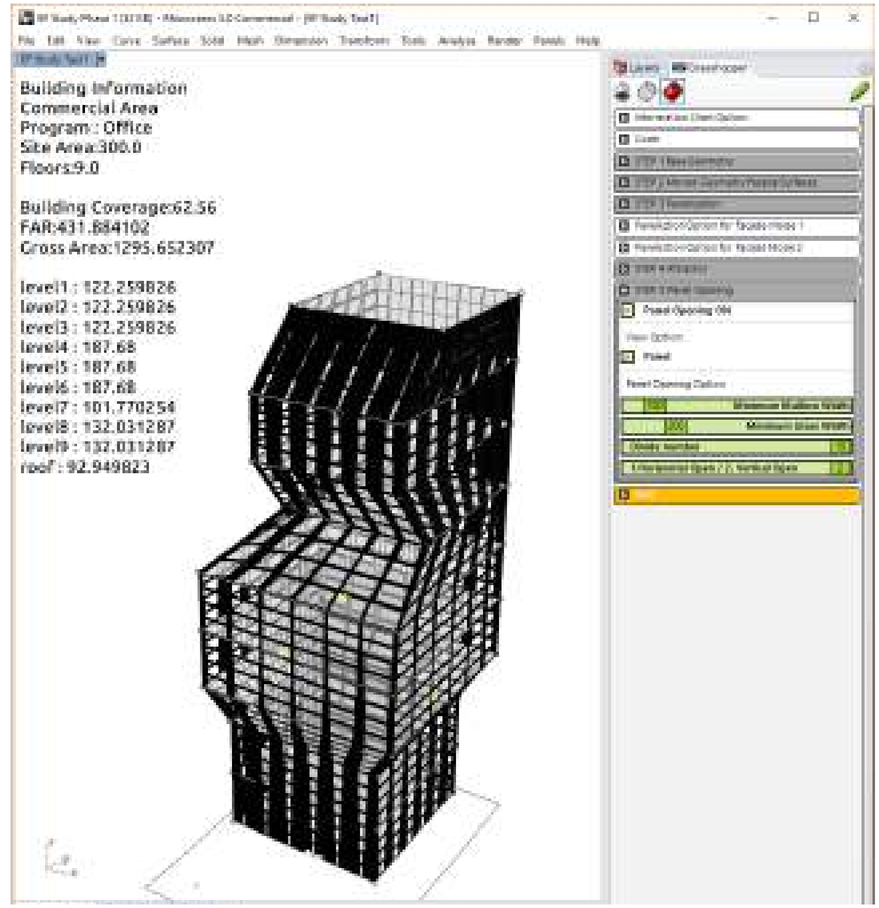
|| Early versions of the Brick Pavilion Maker.

Fabrication and Construction of Coffee Ground Pavilion :

The main theme of 5th UAUS pavilion exhibition was “Replay of Architecture”. Our team proposed bricks made of coffee ground what usually treated like trash. We tested the strength of bricks contains a different ratio of coffee ground to produce bricks can perform enough strength. During the project, Coffeecube, an Eco-friendly startup company in Korea, gladly supported us and provide mixtures of coffee ground and environment-friendly chemicals to make coffee ground bricks.

At the moment, our concern was regarding the number of people and number of times needed to visit Starbucks in order for us to procure a sufficient amount of coffee grounds. Also, how much cement and aggregates did we need to order. During the whole process of design, fabrication, and construction, Brick Pavilion Maker provided us with numerical values which our team could utilize.

|| What if facade shape prototyping is as easy as making a intent noodles?



|| User Interface of EF Study
Remote controller with six steps and four properties setup tabs.

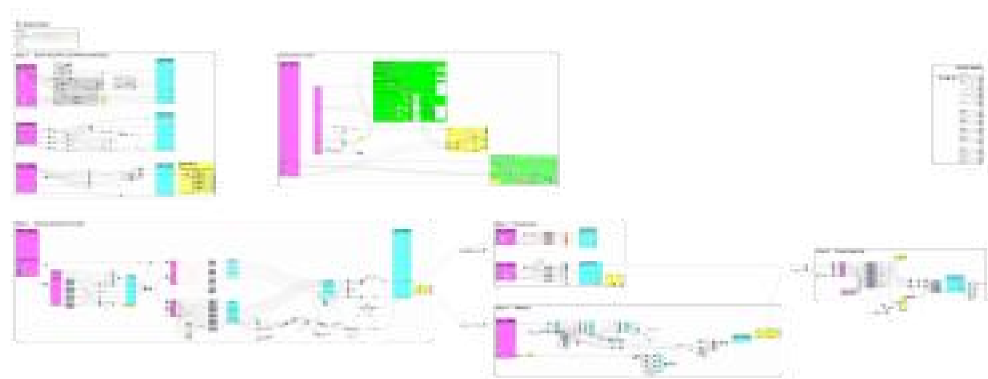
<u>07</u> Make Your Building In 3 Minutes!

Professional Project , Syntegrate
2016.10, 2 weeks

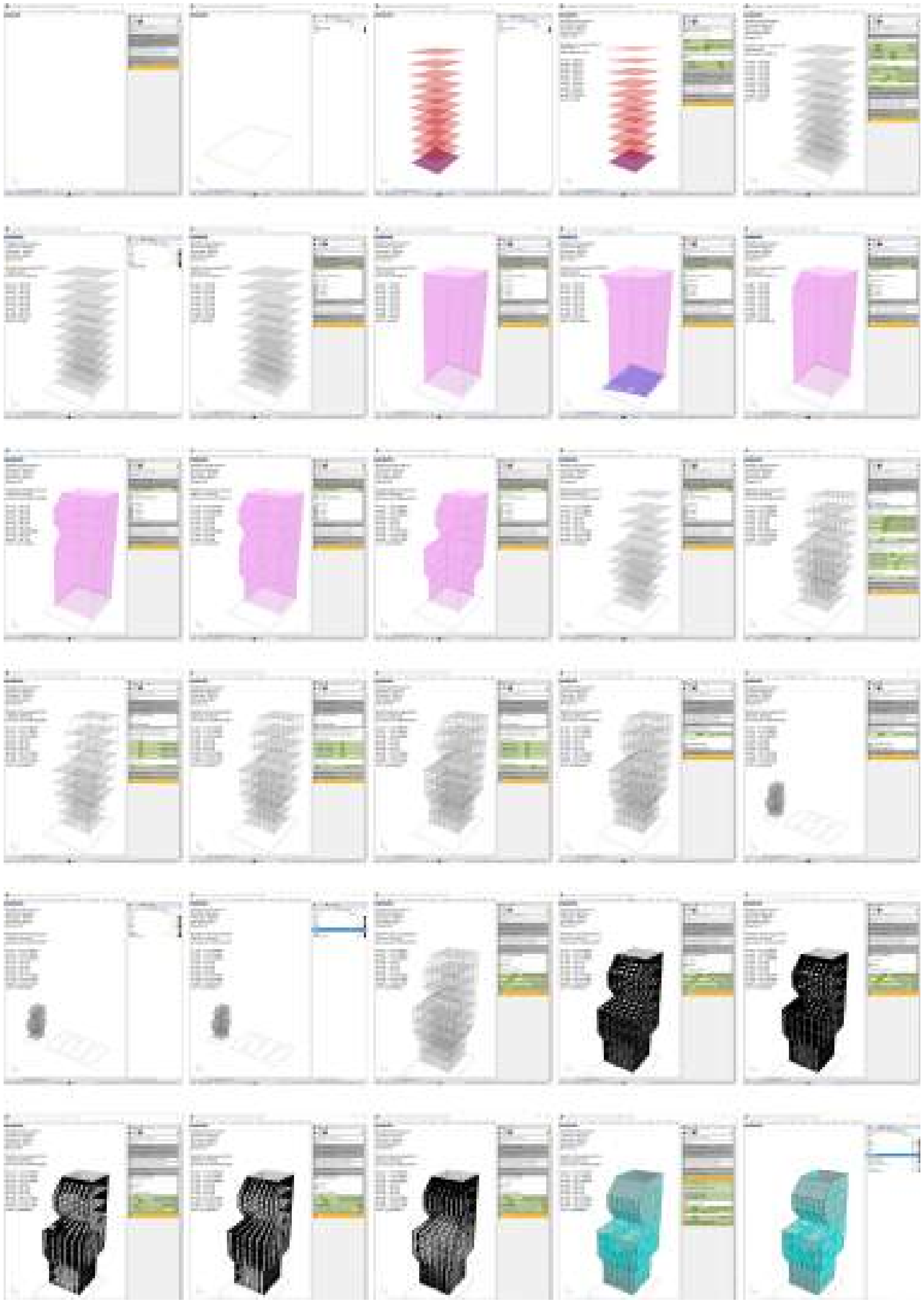
Program : Elevation and Facade(EF) Study
Architect : Gansam Architects
Client : Gansam Architects
Supervisor : Sanghyun Son, Architect, Gansam

Workscope:
1) Create a facade prototyping tool for Rhino users to use in the schematic design stage.
2) Include tacking function of the floor area ratio and building coverage ratio.
3) Create geometry which can be directly fabricated using a 3D printer.

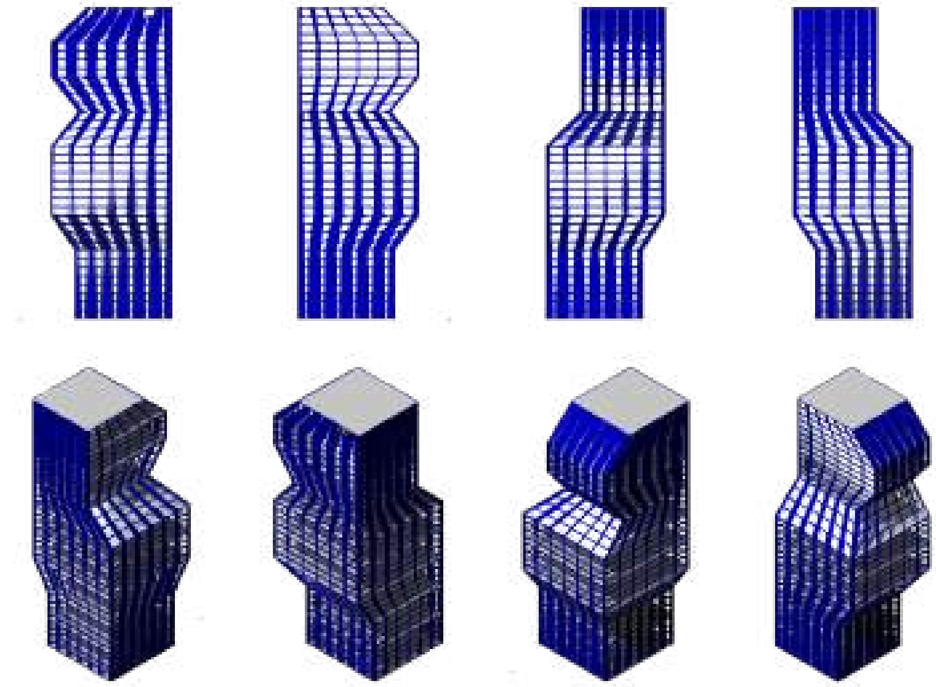
Background:
The business development team of Gansam wanted to make design automation process on design development phase due to maximizing work efficiency. The “EF study phase 1” is a grasshopper code developed in this situation. Using this code, architects can make a building’s form and facade in 3-5 minute. The script considered to be used efficiently for architects do not have grasshopper background.



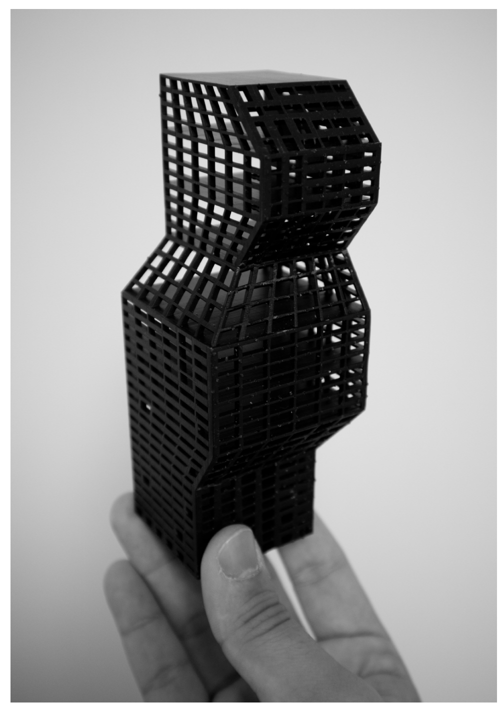
|| Grasshopper script image
6 Steps to make facade geometry. Input and Output of each steps were colored as magenta and cyan, and user display related functions are colored as green and white.



|| Building creation sequence - It took five to ten minutes for the first user to create their building



|| Four facade elevations created from EF study.
The program allows users to create a building form by pulling edge points of each floor surface.

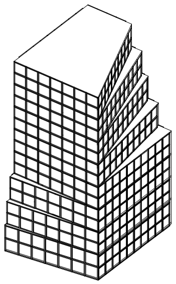


|| EF study allow user to create 3D print-able geometry

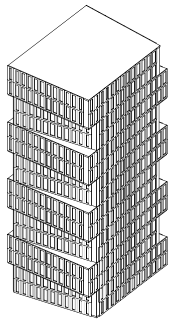
Interactive Program and Triggers :

What I cared most on this program was creating triggers to make program reacts to the users' action. When a user submits initial FAR and the number of floors, the program generates the floor surfaces and edge points geometries. The program keeps reading those data by searching specific ID. When the user pulls the points, in other words when the points' XYZ coordination is changed, It became a trigger to update the mass of the building.

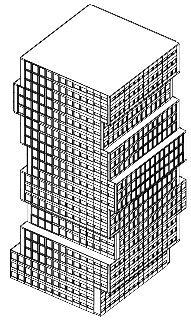
After two weeks development of code, I present the EF Study to the owner of the company, 60-year-old architect. He felt uncomfortable to the fact that the program designs the building by pulling points rather than drawing sketches. The presentation day leaves me a homework to deeply think about the relationship between generative design and traditional way of architecture design.



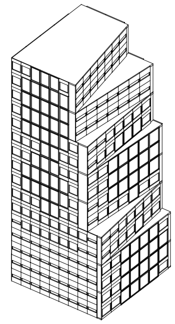
Floors:10
Coverage:69.961058
FAR:537.992716
Gross Area:1613.978149



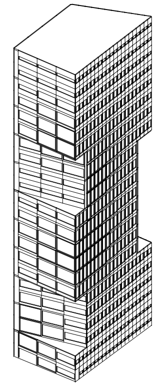
Floors:8
Coverage:69.4
FAR:563.2
Gross Area:1689.6



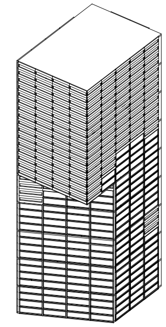
Floors:8
Coverage:69.4
FAR:563.2
Gross Area:1689.6



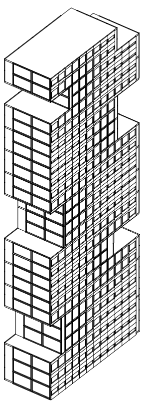
Floors:10
Coverage:69.961058
FAR:537.992716
Gross Area:1613.978149



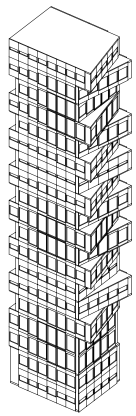
Floors:11
Coverage:51.733333
FAR:517.82352
Gross Area:1553.470559



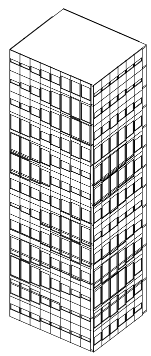
Floors:9
Coverage:57.566667
FAR:456.076176
Gross Area:1368.228527



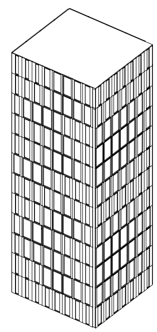
Floors:13
Coverage:45.746667
FAR:530.524555
Gross Area:1591.573666



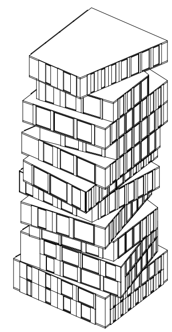
Floors:17
Coverage:35.923323
FAR:553.038696
Gross Area:1659.116089



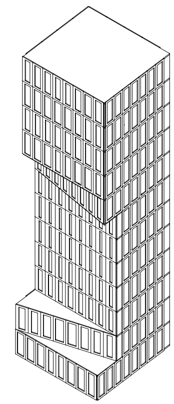
Floors:12
Coverage:49.28
FAR:591.36
Gross Area:1774.08



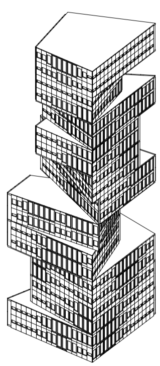
Floors:11
Coverage:53.24
FAR:585.64
Gross Area:1756.92



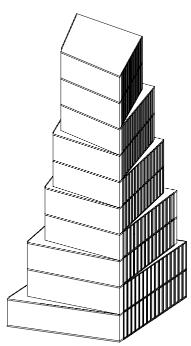
Floors:11
Coverage:69.94215
FAR:564.746862
Gross Area:1694.240585



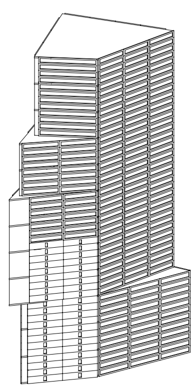
Floors:12
Coverage:63.93333
FAR:641.72531
Gross Area:1925.175929



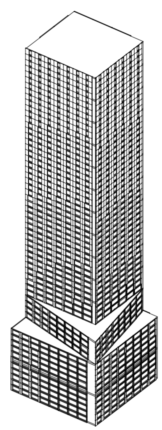
Floors:11
Coverage:60.318674
FAR:382.393228
Gross Area:1147.179684



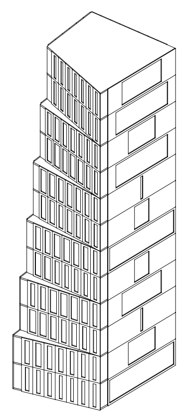
Floors:10
Coverage:65.071206
FAR:477.836226
Gross Area:1433.508677



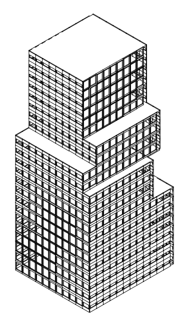
Floors:12
Coverage:61.181936
FAR:598.789086
Gross Area:1796.367258



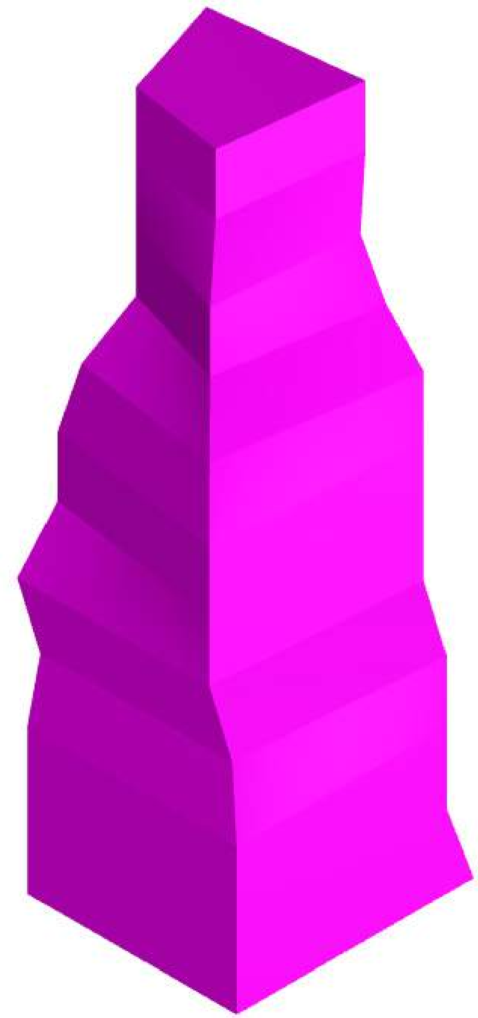
Floors:14
Coverage:58.283494
FAR:591.676091
Gross Area:1775.028274



Floors:12
Coverage:62.491403
FAR:581.58357
Gross Area:2044.750711

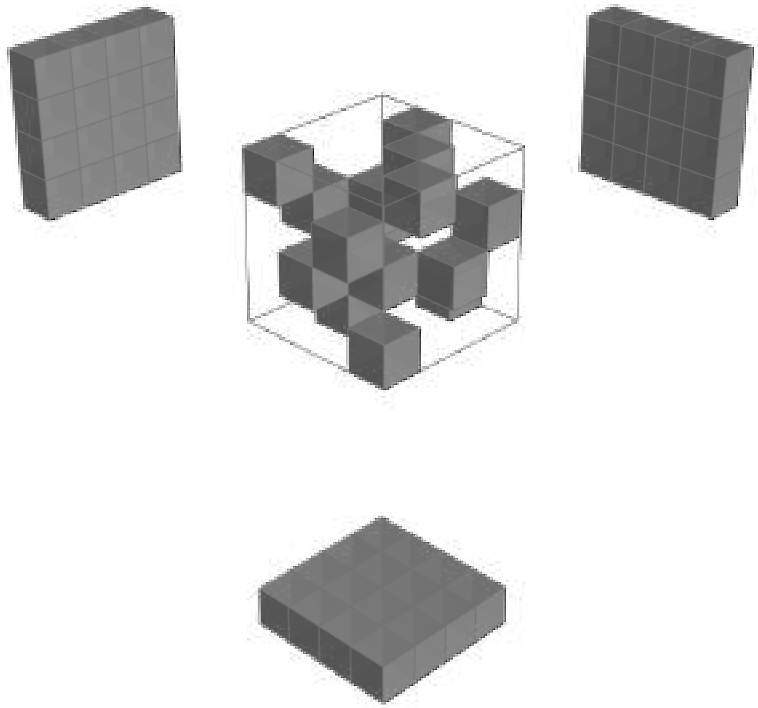


Floors:10
Building Coverage:68.0
FAR:549.08
Gross Area:1714.24



Building prototypes created by users under same site condition :

Initial Input Conditions
Site : 15 meters x 20 meters
Floor Area Ratio Limit: 600 %
Building Coverage Limit: 70%



When a cube divided into identical N^3 smaller cubes and combination of N^2 selected cubes among them make a square when it projected to any edge direction, let's call this geometry as Perfect Random Box. The Image represents Perfect Random Box when N = 4.

<u>08</u> Perfect Random Box - find your house!

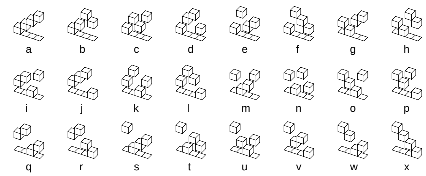
Personal Project
2017.1-

Workscope:
1) Create various geometry under a particular rule.
2) Find all geometry options when n of Perfect Random Box is 4.
3) Create an algorithm that computes Perfect Random Box when n is larger than 4.

Background:

When we look at a tree from a distance, we see the green circular boundary of leaves and straight main trunk. However, when we look at the tree in beneath the leaves, there are vast empty spaces between leaves and branches. The spaced is used as a playground of birds, living area of bugs. The Perfect Random Box is a metaphor of a tree. From a specific direction, it looks like a perfect square. However, various open spaces between divide cube give us to make this geometry as an option of the housing design.

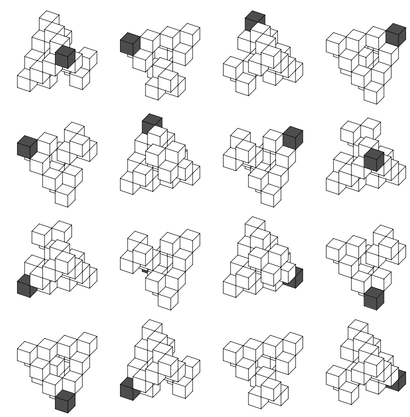
24 Options on each layers



Compatibility of 2 layers

Name	Numbers				Possible Combination Elements										All Combinations			
a	1	2	3	4	a	h	j	k	n	q	r	s	w	x	a	h	q	x
b	1	2	4	3	b	g	i	l	m	q	r	t	w	x	a	h	r	w
c	1	3	2	4	c	h	k	l	n	p	q	t	u	v	a	j	q	s
d	1	3	4	2	d	g	k	l	m	o	r	s	u	v	a	k	n	x
e	1	4	2	3	e	g	i	j	n	o	p	t	v	w	b	g	q	x
f	1	4	3	2	f	h	i	j	m	o	p	s	u	x	b	g	r	w
g	2	1	3	4	g	b	d	e	p	q	r	u	w	x	b	i	r	t
h	2	1	4	3	h	a	c	f	o	q	r	v	w	x	b	l	m	w
i	2	3	1	4	i	b	e	f	n	p	r	s	t	v	c	h	q	v
j	2	3	4	1	j	a	e	f	m	o	q	s	t	u	c	k	n	v
k	2	4	1	3	k	a	c	d	m	n	p	t	v	x	c	k	p	t
l	2	4	3	1	l	b	c	d	m	n	o	s	u	w	c	l	n	u
m	3	1	2	4	m	b	d	f	j	k	l	u	v	w	d	g	r	u
n	3	1	4	2	n	a	c	e	i	k	l	u	v	x	d	k	m	v
o	3	2	1	4	o	d	e	f	h	j	l	s	t	x	d	l	m	u
p	3	2	4	1	p	c	e	f	g	i	k	s	t	w	d	l	o	s
q	3	4	1	2	q	a	b	c	g	h	j	s	v	x	e	g	p	w
r	3	4	2	1	r	a	b	d	g	h	i	t	u	w	e	i	n	v
s	4	1	2	3	s	a	d	f	i	j	l	o	p	q	e	i	p	t
t	4	1	3	2	t	b	c	e	i	j	k	o	p	r	e	j	o	t
u	4	2	1	3	u	c	d	f	g	j	l	m	n	r	f	h	o	x
v	4	2	3	1	v	c	d	e	h	i	k	m	n	q	f	i	p	s
w	4	3	1	2	w	a	b	e	g	h	l	m	p	r	f	j	m	u
x	4	3	2	1	x	a	b	f	g	h	k	n	o	q	f	j	o	s

Identical shapes when rotation is allowed

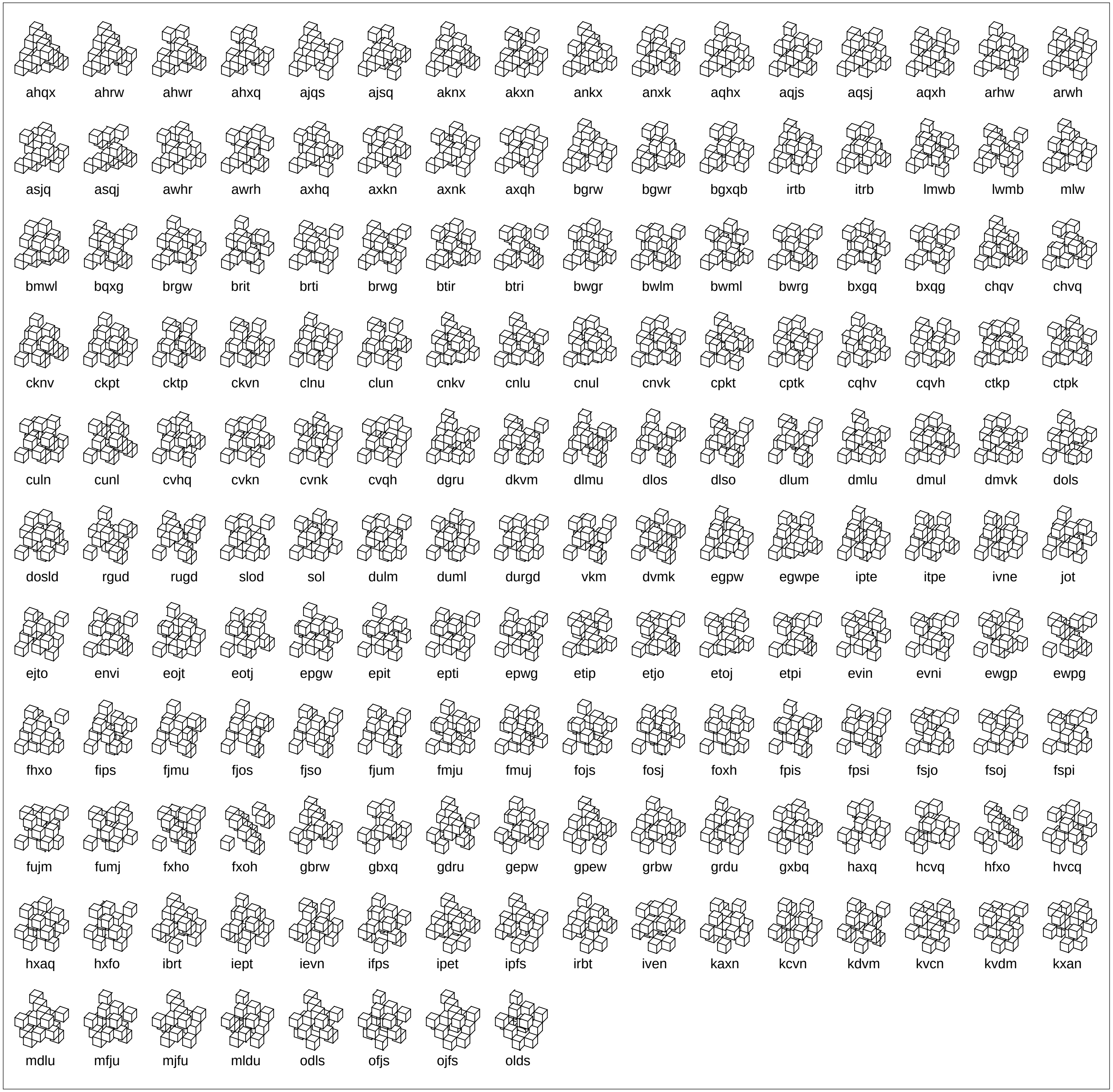


Python script which generate all combinations

Perfect Random Box All Combinations When n = 4

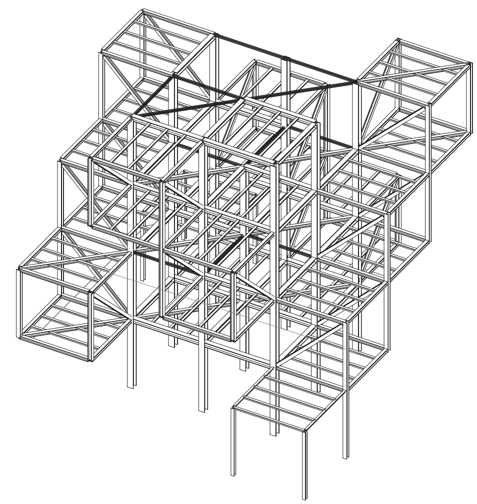
((0, 1, 2, 3), (1, 0, 3, 2), (2, 3, 0, 1), (3, 2, 1, 0)), ((0, 1, 2, 3), (1, 0, 3, 2), (2, 3, 1, 0), (3, 2, 0, 1)),
((0, 1, 2, 3), (1, 0, 3, 2), (3, 2, 0, 1), (2, 3, 1, 0)), ((0, 1, 2, 3), (1, 0, 3, 2), (3, 2, 1, 0), (2, 3, 0, 1)),
((0, 1, 2, 3), (1, 2, 3, 0), (2, 3, 0, 1), (3, 0, 1, 2)), ((0, 1, 2, 3), (1, 2, 3, 0), (3, 0, 1, 2), (2, 3, 0, 1)),
((0, 1, 2, 3), (1, 3, 0, 2), (2, 0, 3, 1), (3, 2, 1, 0)), ((0, 1, 2, 3), (1, 3, 0, 2), (3, 2, 1, 0), (2, 0, 3, 1)),
((0, 1, 2, 3), (2, 0, 3, 1), (1, 3, 0, 2), (3, 2, 1, 0)), ((0, 1, 2, 3), (2, 0, 3, 1), (3, 2, 1, 0), (1, 3, 0, 2)),
((0, 1, 2, 3), (2, 3, 0, 1), (1, 0, 3, 2), (3, 2, 1, 0)), ((0, 1, 2, 3), (2, 3, 0, 1), (1, 2, 3, 0), (3, 0, 1, 2)),
((0, 1, 2, 3), (2, 3, 0, 1), (3, 0, 1, 2), (1, 2, 3, 0)), ((0, 1, 2, 3), (2, 3, 0, 1), (3, 2, 1, 0), (1, 0, 3, 2)),
((0, 1, 2, 3), (2, 3, 1, 0), (1, 0, 3, 2), (3, 2, 0, 1)), ((0, 1, 2, 3), (2, 3, 1, 0), (3, 2, 0, 1), (1, 0, 3, 2)),
((0, 1, 2, 3), (3, 0, 1, 2), (1, 2, 3, 0), (2, 3, 0, 1)), ((0, 1, 2, 3), (3, 0, 1, 2), (2, 3, 0, 1), (1, 2, 3, 0)),
((0, 1, 2, 3), (3, 2, 0, 1), (1, 0, 3, 2), (2, 3, 1, 0)), ((0, 1, 2, 3), (3, 2, 0, 1), (2, 3, 1, 0), (1, 0, 3, 2)),
((0, 1, 2, 3), (3, 2, 1, 0), (1, 0, 3, 2), (2, 3, 0, 1)), ((0, 1, 2, 3), (3, 2, 1, 0), (1, 3, 0, 2), (2, 0, 3, 1)),
((0, 1, 2, 3), (3, 2, 1, 0), (2, 0, 3, 1), (1, 3, 0, 2)), ((0, 1, 2, 3), (3, 2, 1, 0), (2, 3, 0, 1), (1, 0, 3, 2)),
((0, 1, 3, 2), (1, 0, 2, 3), (2, 3, 1, 0), (3, 2, 0, 1)), ((0, 1, 3, 2), (1, 0, 2, 3), (3, 2, 0, 1), (2, 3, 1, 0)),
((0, 1, 3, 2), (1, 0, 2, 3), (3, 2, 1, 0), (2, 3, 0, 1)), ((0, 1, 3, 2), (1, 2, 0, 3), (2, 3, 1, 0), (3, 0, 2, 1)),
((0, 1, 3, 2), (1, 2, 0, 3), (3, 0, 2, 1), (2, 3, 1, 0)), ((0, 1, 3, 2), (1, 3, 2, 0), (2, 0, 1, 3), (3, 2, 0, 1)),
((0, 1, 3, 2), (1, 3, 2, 0), (3, 2, 0, 1), (2, 0, 1, 3)), ((0, 1, 3, 2), (2, 0, 1, 3), (1, 3, 2, 0), (3, 2, 0, 1)),
((0, 1, 3, 2), (2, 0, 1, 3), (3, 2, 0, 1), (1, 3, 2, 0)), ((0, 1, 3, 2), (2, 3, 0, 1), (3, 2, 1, 0), (1, 0, 2, 3)),
((0, 1, 3, 2), (2, 3, 1, 0), (1, 0, 2, 3), (3, 2, 0, 1)), ((0, 1, 3, 2), (2, 3, 1, 0), (1, 2, 0, 3), (3, 0, 2, 1)),
((0, 1, 3, 2), (2, 3, 1, 0), (3, 0, 2, 1), (1, 2, 0, 3)), ((0, 1, 3, 2), (2, 3, 1, 0), (3, 2, 0, 1), (1, 0, 2, 3)),
((0, 1, 3, 2), (3, 0, 2, 1), (1, 2, 0, 3), (2, 3, 1, 0)), ((0, 1, 3, 2), (3, 0, 2, 1), (2, 3, 1, 0), (1, 2, 0, 3)),
((0, 1, 3, 2), (3, 2, 0, 1), (1, 0, 2, 3), (2, 3, 1, 0)), ((0, 1, 3, 2), (3, 2, 0, 1), (1, 3, 2, 0), (2, 0, 1, 3)),
((0, 1, 3, 2), (3, 2, 0, 1), (2, 0, 1, 3), (1, 3, 2, 0)), ((0, 1, 3, 2), (3, 2, 0, 1), (2, 3, 1, 0), (1, 0, 2, 3)),
((0, 1, 3, 2), (3, 2, 1, 0), (1, 0, 2, 3), (2, 3, 0, 1)), ((0, 1, 3, 2), (3, 2, 1, 0), (2, 3, 0, 1), (1, 0, 2, 3)),
((0, 2, 1, 3), (1, 0, 3, 2), (2, 3, 0, 1), (3, 1, 2, 0)), ((0, 2, 1, 3), (1, 0, 3, 2), (3, 1, 2, 0), (2, 3, 0, 1)),
((0, 2, 1, 3), (1, 3, 0, 2), (2, 0, 3, 1), (3, 1, 2, 0)), ((0, 2, 1, 3), (1, 3, 0, 2), (2, 1, 3, 0), (3, 0, 2, 1)),
((0, 2, 1, 3), (1, 3, 0, 2), (3, 0, 2, 1), (2, 1, 3, 0)), ((0, 2, 1, 3), (1, 3, 0, 2), (3, 1, 2, 0), (2, 0, 3, 1)),
((0, 2, 1, 3), (1, 3, 2, 0), (2, 0, 3, 1), (3, 1, 0, 2)), ((0, 2, 1, 3), (1, 3, 2, 0), (3, 1, 0, 2), (2, 0, 3, 1)),
((0, 2, 1, 3), (2, 0, 3, 1), (1, 3, 0, 2), (3, 1, 2, 0)), ((0, 2, 1, 3), (2, 0, 3, 1), (1, 3, 2, 0), (3, 1, 0, 2)),
((0, 2, 1, 3), (2, 0, 3, 1), (3, 1, 0, 2), (1, 3, 2, 0)), ((0, 2, 1, 3), (2, 0, 3, 1), (3, 1, 2, 0), (1, 3, 0, 2)),
((0, 2, 1, 3), (2, 1, 3, 0), (1, 3, 0, 2), (3, 0, 2, 1)), ((0, 2, 1, 3), (2, 1, 3, 0), (3, 0, 2, 1), (1, 3, 0, 2)),
((0, 2, 1, 3), (2, 3, 0, 1), (1, 0, 3, 2), (3, 1, 2, 0)), ((0, 2, 1, 3), (2, 3, 0, 1), (3, 1, 2, 0), (1, 0, 3, 2)),
((0, 2, 1, 3), (3, 0, 2, 1), (1, 3, 0, 2), (2, 1, 3, 0)), ((0, 2, 1, 3), (3, 0, 2, 1), (2, 1, 3, 0), (1, 3, 0, 2)),
((0, 2, 1, 3), (3, 1, 0, 2), (1, 3, 2, 0), (2, 0, 3, 1)), ((0, 2, 1, 3), (3, 1, 0, 2), (2, 0, 3, 1), (1, 3, 2, 0)),
((0, 2, 1, 3), (3, 1, 2, 0), (1, 0, 3, 2), (2, 3, 0, 1)), ((0, 2, 1, 3), (3, 1, 2, 0), (1, 3, 0, 2), (2, 0, 3, 1)),
((0, 2, 1, 3), (3, 1, 2, 0), (2, 0, 3, 1), (1, 3, 0, 2)), ((0, 2, 1, 3), (3, 1, 2, 0), (2, 3, 0, 1), (1, 0, 3, 2)),
((0, 2, 3, 1), (1, 0, 2, 3), (2, 3, 1, 0), (3, 1, 0, 2)), ((0, 2, 3, 1), (1, 3, 0, 2), (3, 1, 2, 0), (2, 0, 1, 3)),
((0, 2, 3, 1), (1, 3, 2, 0), (2, 0, 1, 3), (3, 1, 0, 2)), ((0, 2, 3, 1), (1, 3, 2, 0), (2, 1, 0, 3), (3, 0, 1, 2)),
((0, 2, 3, 1), (1, 3, 2, 0), (3, 0, 1, 2), (2, 1, 0, 3)), ((0, 2, 3, 1), (1, 3, 2, 0), (3, 1, 0, 2), (2, 0, 1, 3)),
((0, 2, 3, 1), (2, 0, 1, 3), (1, 3, 2, 0), (3, 1, 0, 2)), ((0, 2, 3, 1), (2, 0, 1, 3), (3, 1, 0, 2), (1, 3, 2, 0)),
((0, 2, 3, 1), (2, 0, 1, 3), (3, 1, 2, 0), (1, 3, 0, 2)), ((0, 2, 3, 1), (2, 1, 0, 3), (1, 3, 2, 0), (3, 0, 1, 2)),
((0, 2, 3, 1), (2, 1, 0, 3), (3, 0, 1, 2), (1, 3, 2, 0)), ((0, 2, 3, 1), (2, 3, 1, 0), (1, 0, 2, 3), (3, 1, 0, 2)),
((0, 2, 3, 1), (2, 3, 1, 0), (3, 1, 0, 2), (1, 0, 2, 3)), ((0, 2, 3, 1), (3, 0, 1, 2), (1, 3, 2, 0), (2, 1, 0, 3)),
((0, 2, 3, 1), (3, 0, 1, 2), (2, 1, 0, 3), (1, 3, 2, 0)), ((0, 2, 3, 1), (3, 1, 0, 2), (1, 3, 2, 0), (2, 0, 1, 3)),
((0, 2, 3, 1), (3, 1, 0, 2), (2, 0, 1, 3), (1, 3, 2, 0)), ((0, 2, 3, 1), (3, 1, 0, 2), (2, 3, 1, 0), (1, 0, 2, 3)),
((0, 2, 3, 1), (3, 1, 2, 0), (1, 3, 0, 2), (2, 0, 1, 3)), ((0, 2, 3, 1), (3, 1, 2, 0), (2, 0, 1, 3), (1, 3, 0, 2)),
((0, 3, 1, 2), (1, 0, 2, 3), (2, 1, 3, 0), (3, 2, 0, 1)), ((0, 3, 1, 2), (1, 0, 2, 3), (3, 2, 0, 1), (2, 1, 3, 0)),
((0, 3, 1, 2), (1, 2, 0, 3), (2, 1, 3, 0), (3, 0, 2, 1)), ((0, 3, 1, 2), (1, 2, 0, 3), (3, 0, 2, 1), (2, 1, 3, 0)),
((0, 3, 1, 2), (1, 2, 0, 3), (3, 1, 2, 0), (2, 0, 3, 1)), ((0, 3, 1, 2), (1, 2, 3, 0), (2, 1, 0, 3), (3, 0, 2, 1)),
((0, 3, 1, 2), (1, 2, 3, 0), (3, 0, 2, 1), (2, 1, 0, 3)), ((0, 3, 1, 2), (2, 0, 3, 1), (3, 1, 2, 0), (1, 2, 0, 3)),
((0, 3, 1, 2), (2, 1, 0, 3), (1, 2, 3, 0), (3, 0, 2, 1)), ((0, 3, 1, 2), (2, 1, 0, 3), (3, 0, 2, 1), (1, 2, 3, 0)),
((0, 3, 1, 2), (2, 1, 3, 0), (1, 0, 2, 3), (3, 2, 0, 1)), ((0, 3, 1, 2), (2, 1, 3, 0), (1, 2, 0, 3), (3, 0, 2, 1)),
((0, 3, 1, 2), (2, 1, 3, 0), (3, 0, 2, 1), (1, 2, 0, 3)), ((0, 3, 1, 2), (2, 1, 3, 0), (3, 2, 0, 1), (1, 0, 2, 3)),
((0, 3, 1, 2), (3, 0, 2, 1), (1, 2, 0, 3), (2, 1, 3, 0)), ((0, 3, 1, 2), (3, 0, 2, 1), (1, 2, 3, 0), (2, 1, 0, 3)),
((0, 3, 1, 2), (3, 0, 2, 1), (2, 1, 0, 3), (1, 2, 3, 0)), ((0, 3, 1, 2), (3, 0, 2, 1), (2, 1, 3, 0), (1, 2, 0, 3)),
((0, 3, 1, 2), (3, 1, 2, 0), (1, 2, 0, 3), (2, 0, 3, 1)), ((0, 3, 1, 2), (3, 1, 2, 0), (2, 0, 3, 1), (1, 2, 0, 3)),
((0, 3, 1, 2), (3, 2, 0, 1), (1, 0, 2, 3), (2, 1, 3, 0)), ((0, 3, 1, 2), (3, 2, 0, 1), (2, 1, 3, 0), (1, 0, 2, 3)),
((0, 3, 2, 1), (1, 0, 3, 2), (3, 2, 1, 0), (2, 1, 0, 3)), ((0, 3, 2, 1), (1, 2, 0, 3), (2, 1, 3, 0), (3, 0, 1, 2)),
((0, 3, 2, 1), (1, 2, 3, 0), (2, 0, 1, 3), (3, 1, 0, 2)), ((0, 3, 2, 1), (1, 2, 3, 0), (2, 1, 0, 3), (3, 0, 1, 2)),
((0, 3, 2, 1), (1, 2, 3, 0), (3, 0, 1, 2), (2, 1, 0, 3)), ((0, 3, 2, 1), (1, 2, 3, 0), (3, 1, 0, 2), (2, 0, 1, 3)),
((0, 3, 2, 1), (2, 0, 1, 3), (1, 2, 3, 0), (3, 1, 0, 2)), ((0, 3, 2, 1), (2, 0, 1, 3), (3, 1, 0, 2), (1, 2, 3, 0)),
((0, 3, 2, 1), (2, 1, 0, 3), (1, 2, 3, 0), (3, 0, 1, 2)), ((0, 3, 2, 1), (2, 1, 0, 3), (3, 0, 1, 2), (1, 2, 3, 0)),
((0, 3, 2, 1), (2, 1, 0, 3), (3, 2, 1, 0), (1, 0, 3, 2)), ((0, 3, 2, 1), (2, 1, 3, 0), (1, 2, 0, 3), (3, 0, 1, 2)),
((0, 3, 2, 1), (2, 1, 3, 0), (3, 0, 1, 2), (1, 2, 0, 3)), ((0, 3, 2, 1), (3, 0, 1, 2), (1, 2, 3, 0), (2, 1, 0, 3)),
((0, 3, 2, 1), (3, 0, 1, 2), (2, 1, 0, 3), (1, 2, 3, 0)), ((0, 3, 2, 1), (3, 0, 1, 2), (2, 1, 3, 0), (1, 2, 0, 3)),
((0, 3, 2, 1), (3, 1, 0, 2), (1, 2, 3, 0), (2, 0, 1, 3)), ((0, 3, 2, 1), (3, 1, 0, 2), (2, 0, 1, 3), (1, 2, 3, 0)),
((0, 3, 2, 1), (3, 2, 1, 0), (1, 0, 3, 2), (2, 1, 0, 3)), ((0, 3, 2, 1), (3, 2, 1, 0), (2, 1, 0, 3), (1, 0, 3, 2)),
((1, 0, 2, 3), (0, 1, 3, 2), (2, 3, 1, 0), (3, 2, 0, 1)), ((1, 0, 2, 3), (0, 1, 3, 2), (3, 2, 1, 0), (2, 3, 0, 1)),
((1, 0, 2, 3), (0, 2, 3, 1), (2, 3, 1, 0), (3, 1, 0, 2)), ((1, 0, 2, 3), (0, 3, 1, 2), (2, 1, 3, 0), (3, 2, 0, 1)),
((1, 0, 2, 3), (2, 1, 3, 0), (0, 3, 1, 2), (3, 2, 0, 1)), ((1, 0, 2, 3), (2, 3, 1, 0), (0, 1, 3, 2), (3, 2, 0, 1)),
((1, 0, 2, 3), (2, 3, 1, 0), (0, 2, 3, 1), (3, 1, 0, 2)), ((1, 0, 2, 3), (3, 2, 1, 0), (0, 1, 3, 2), (2, 3, 0, 1)),
((1, 0, 3, 2), (0, 1, 2, 3), (3, 2, 1, 0), (2, 3, 0, 1)), ((1, 0, 3, 2), (0, 2, 1, 3), (3, 1, 2, 0), (2, 3, 0, 1)),
((1, 0, 3, 2), (0, 3, 2, 1), (3, 2, 1, 0), (2, 1, 0, 3)), ((1, 0, 3, 2), (3, 1, 2, 0), (0, 2, 1, 3), (2, 3, 0, 1)),
((1, 0, 3, 2), (3, 2, 1, 0), (0, 1, 2, 3), (2, 3, 0, 1)), ((1, 0, 3, 2), (3, 2, 1, 0), (0, 3, 2, 1), (2, 1, 0, 3)),
((1, 2, 0, 3), (0, 1, 3, 2), (2, 3, 1, 0), (3, 0, 2, 1)), ((1, 2, 0, 3), (0, 3, 1, 2), (2, 1, 3, 0), (3, 0, 2, 1)),
((1, 2, 0, 3), (0, 3, 1, 2), (3, 1, 2, 0), (2, 0, 3, 1)), ((1, 2, 0, 3), (0, 3, 2, 1), (2, 1, 3, 0), (3, 0, 1, 2)),
((1, 2, 0, 3), (2, 1, 3, 0), (0, 3, 1, 2), (3, 0, 2, 1)), ((1, 2, 0, 3), (2, 1, 3, 0), (0, 3, 2, 1), (3, 0, 1, 2)),
((1, 2, 0, 3), (2, 3, 1, 0), (0, 1, 3, 2), (3, 0, 2, 1)), ((1, 2, 0, 3), (3, 1, 2, 0), (0, 3, 1, 2), (2, 0, 3, 1)),
((1, 3, 0, 2), (0, 1, 2, 3), (3, 2, 1, 0), (2, 0, 3, 1)), ((1, 3, 0, 2), (0, 2, 1, 3), (3, 1, 2, 0), (2, 0, 3, 1)),
((1, 3, 0, 2), (0, 2, 3, 1), (3, 1, 2, 0), (2, 0, 1, 3)), ((1, 3, 0, 2), (3, 1, 2, 0), (0, 2, 1, 3), (2, 0, 3, 1)),
((1, 3, 0, 2), (3, 1, 2, 0), (0, 2, 3, 1), (2, 0, 1, 3)), ((1, 3, 0, 2), (3, 2, 1, 0), (0, 1, 2, 3), (2, 0, 3, 1)),
((2, 0, 1, 3), (0, 2, 3, 1), (1, 3, 2, 0), (3, 1, 0, 2)), ((2, 0, 1, 3), (0, 3, 2, 1), (1, 2, 3, 0), (3, 1, 0, 2)),
((2, 0, 1, 3), (1, 2, 3, 0), (0, 3, 2, 1), (3, 1, 0, 2)), ((2, 0, 1, 3), (1, 3, 2, 0), (0, 2, 3, 1), (3, 1, 0, 2)),
((2, 1, 0, 3), (0, 2, 3, 1), (1, 3, 2, 0), (3, 0, 1, 2)), ((2, 1, 0, 3), (0, 3, 2, 1), (1, 2, 3, 0), (3, 0, 1, 2)),
((2, 1, 0, 3), (1, 2, 3, 0), (0, 3, 2, 1), (3, 0, 1, 2)), ((2, 1, 0, 3), (1, 3, 2, 0), (0, 2, 3, 1), (3, 0, 1, 2))

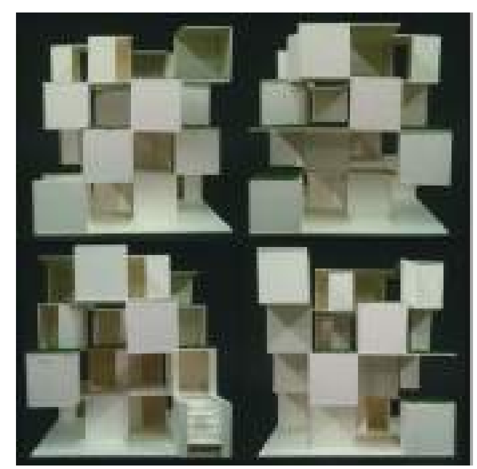
|| Process of creating all Perfect Random Boxes when n is 4.



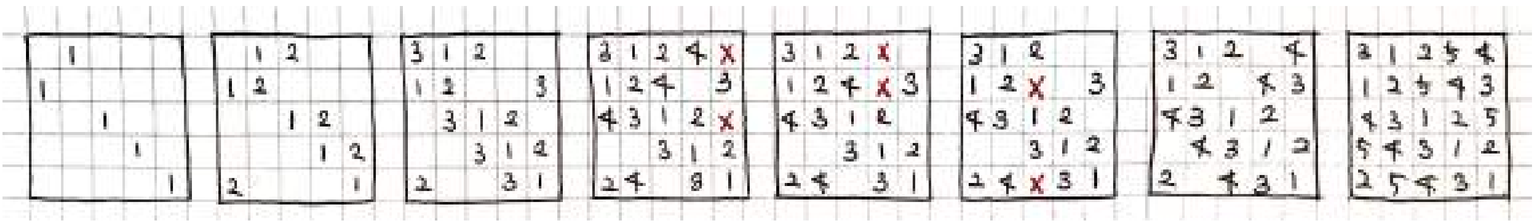
|| 168 Unique Geometry of Perfect Random Box When n is 4 :

Can client select one of these geometries for his housing which is perfectly fit with his lifestyle? For example, the selection of geometries that allows parking 12-meter long truck, growing 6-meter height bamboo in the garden, and providing 24 hours shaded outdoor space for a sweltering summer.

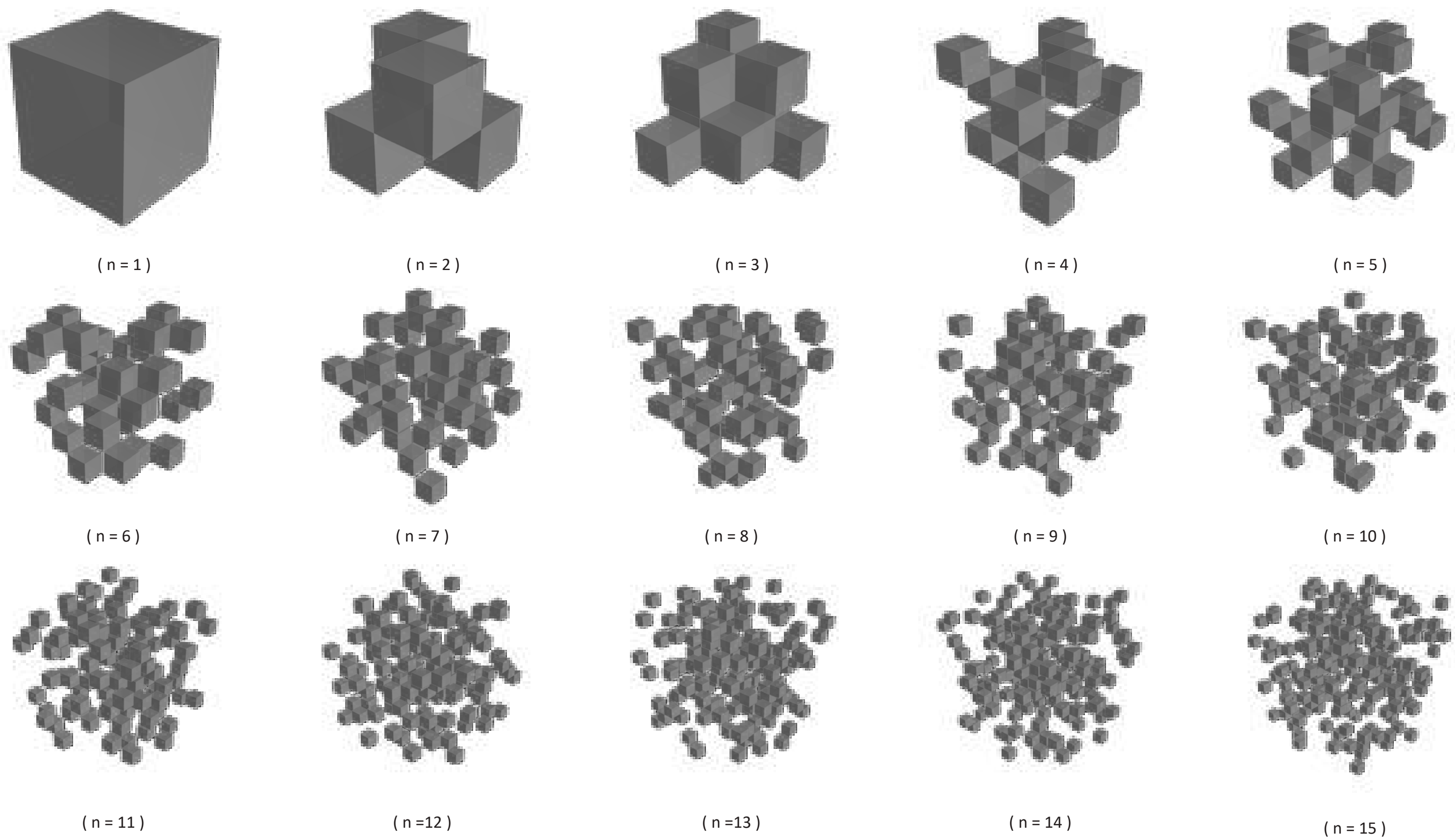




|| Study models of Perfect Random Box house



|| Manual creation of Perfect Random Box with hand calculation when n is 5.
5 by 5 chart indicates the location of boxes seen from the top view and the numbers in each cell indicate the level of the boxes.
As more boxes are filled with numbers, there are cases that numbers cannot be inserted more as marked as red X in 4th, 5th, and 6th image of the diagram. In those cases, I remove the numbers from cells and insert the number in other places. Based on this logic, I create 40 lines of Python code that does the same computation as I do but much faster.

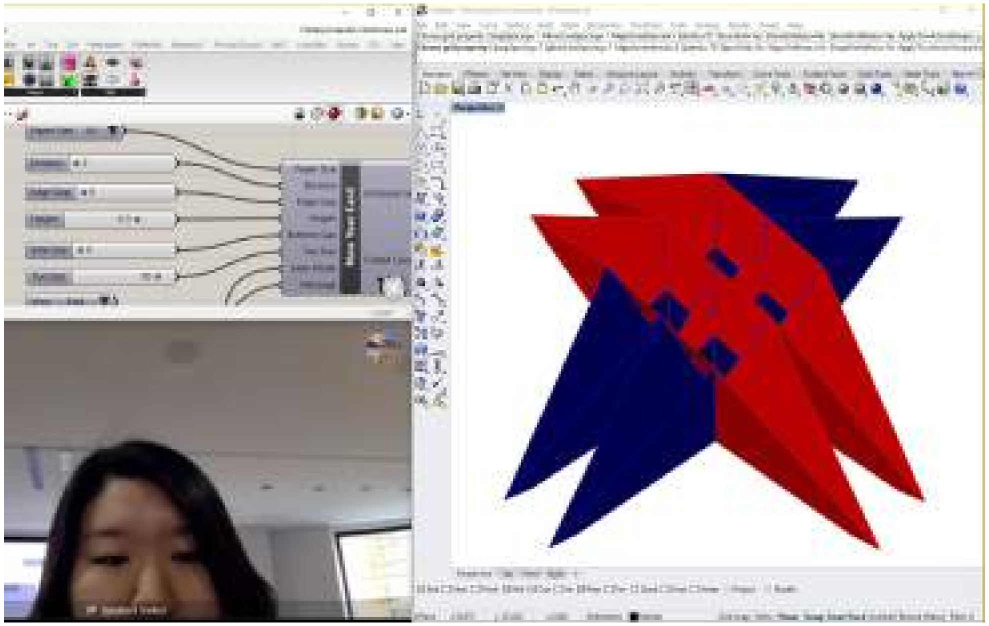


|| Output of Perfect Random Box using Python script when input n is 1 to 15.

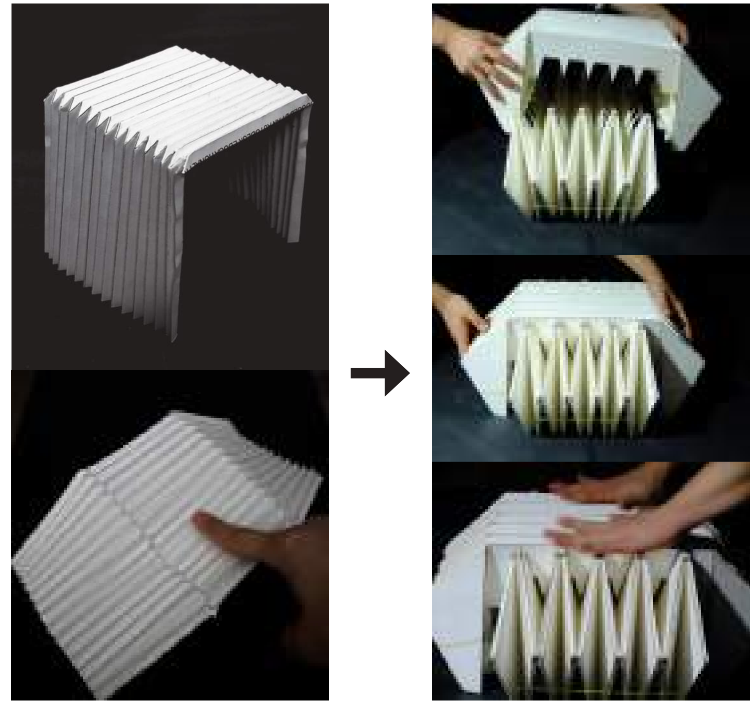
|| Algorithm to create Perfect Random Box of input integer n :

It is easy to compute Perfect Random Box by hand calculation when n is 1,2,3,4, and 5. However, when n becomes 6, hand calculation takes much time to make the geometry. For this reason, I made an algorithm which computes the Perfect Random Box in the same way as I do. It calculates an output less than 100ms in Grasshopper until n becomes 15. However, when n gets to 16, the code starts to break itself because of runtime error.

Similar problems happened while creating a sorting and nesting algorithm in other projects. The limitation of my script leads me to keep my interest in learning computer science subjects to make better code for my design.



|| Skype call screen captured.
Susana is designing her new years folding sculpture using my generative design code.



A single folded paper piece
Vulnerable to pressing force

Interlocking folded paper pieces
Strong enough to sit on

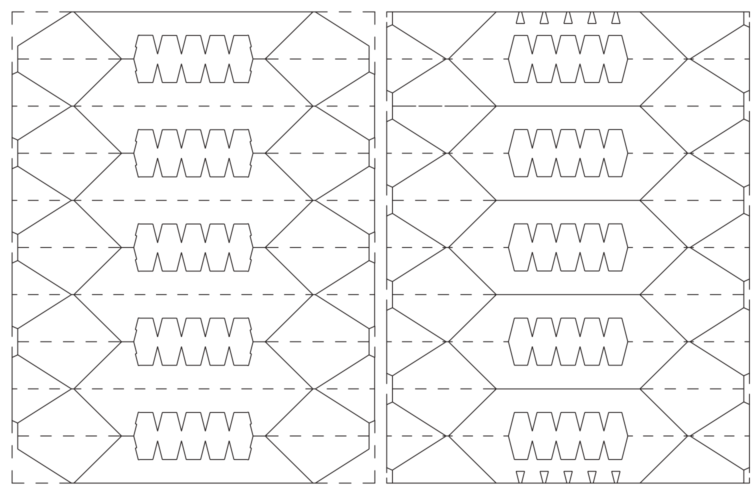
09 New Year Folding Sculpture - Make your geometry!

Personal Project
2018.12 -

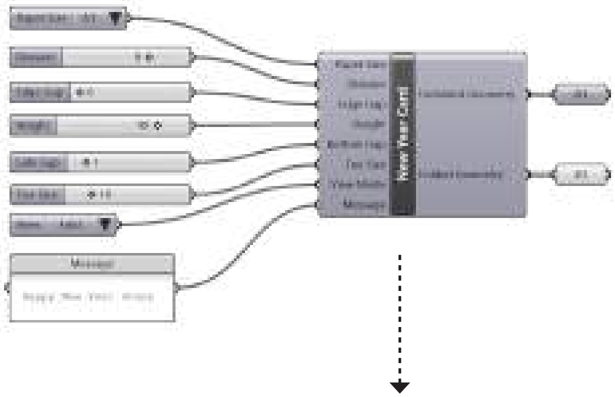
Workscope:
1) Design interlocking mechanism folding paper sculpture
2) Test diversity of design made by general users.

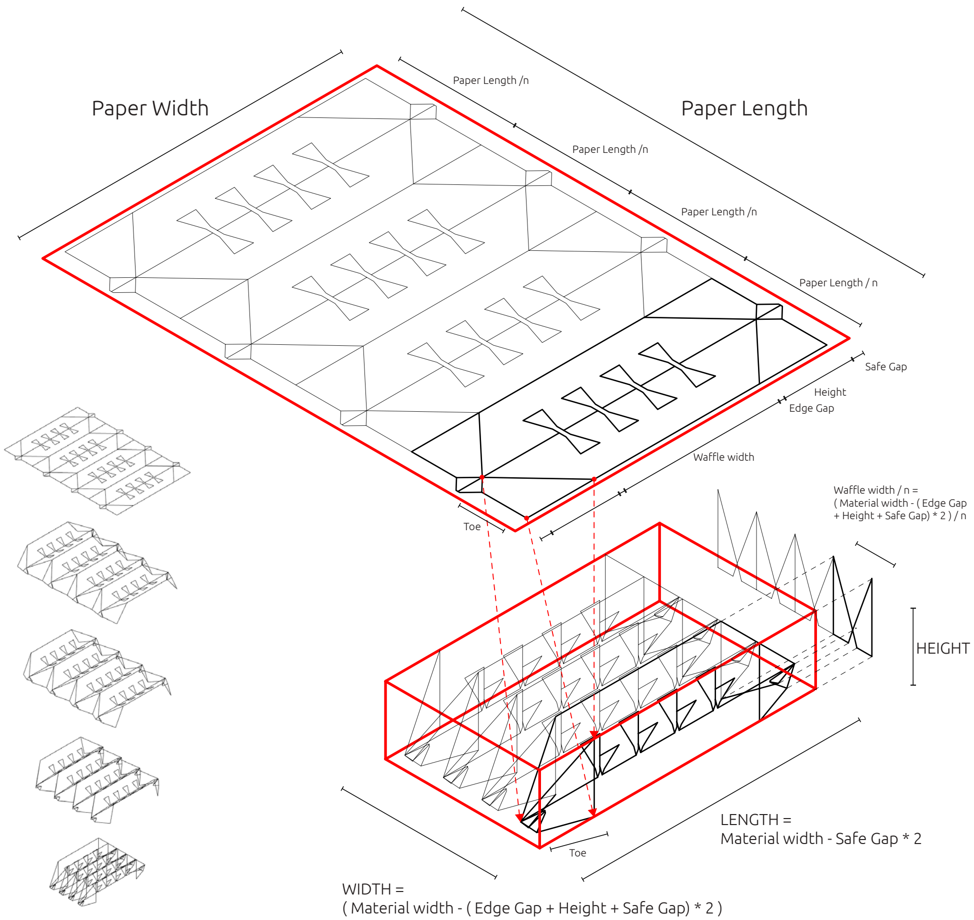
Background :

I used to send a message or an email to my friends in new years day. However, I decided to try something more interesting for my best friends this year, sending them an unfolded paper which will be a one-of-a-kind sculpture after assembled. From early December, I randomly called them by Skype or Facetime and asked them to spend 3 minutes to do something fun.

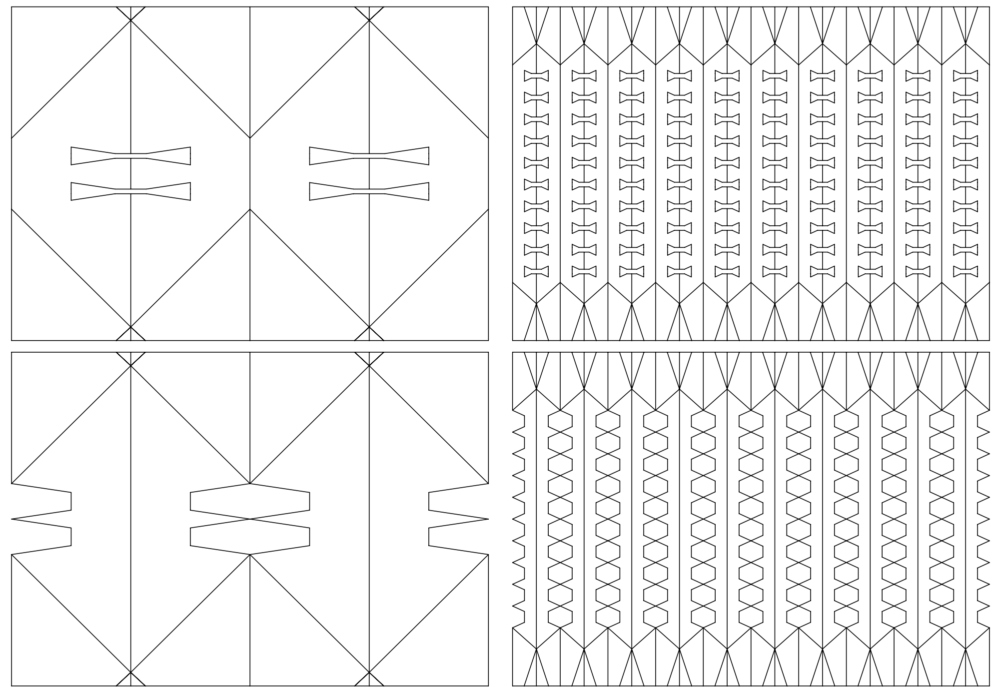


|| Design of interlocking folded paper and unfolded planar geometry for cutting.





|| The Grasshopper script of 5 user inputs. 2D to 3D conversion of paper using trigonometric function.

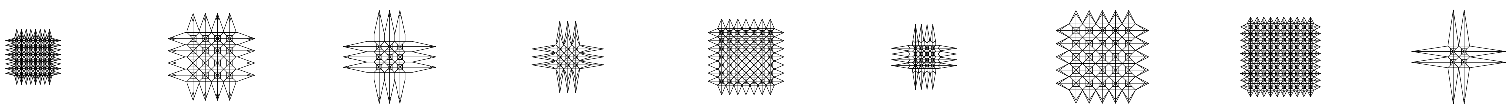


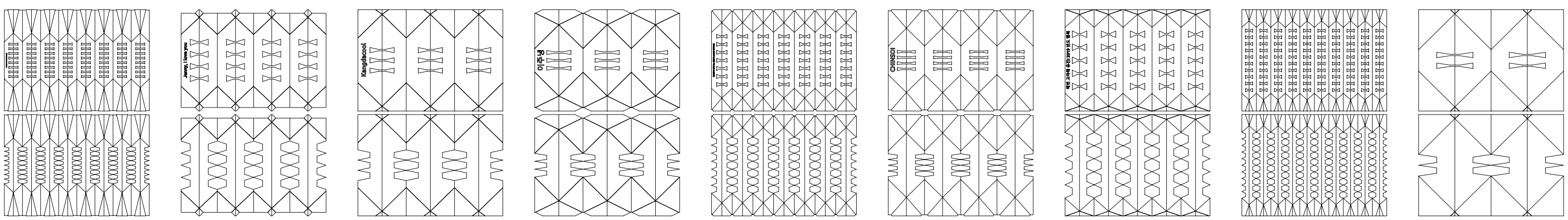
Paper Width = 297 mm
Paper Length = 420 mm
Division = 2
Edge Gap = 0 mm
Height = 113 mm
Safe Gap = 0 mm
Toe Size = 90 mm

Paper Width = 297 mm
Paper Length = 420 mm
Division = 10
Edge Gap = 0 mm
Height = 50 mm
Safe Gap = 0 mm
Toe Size = 10 mm

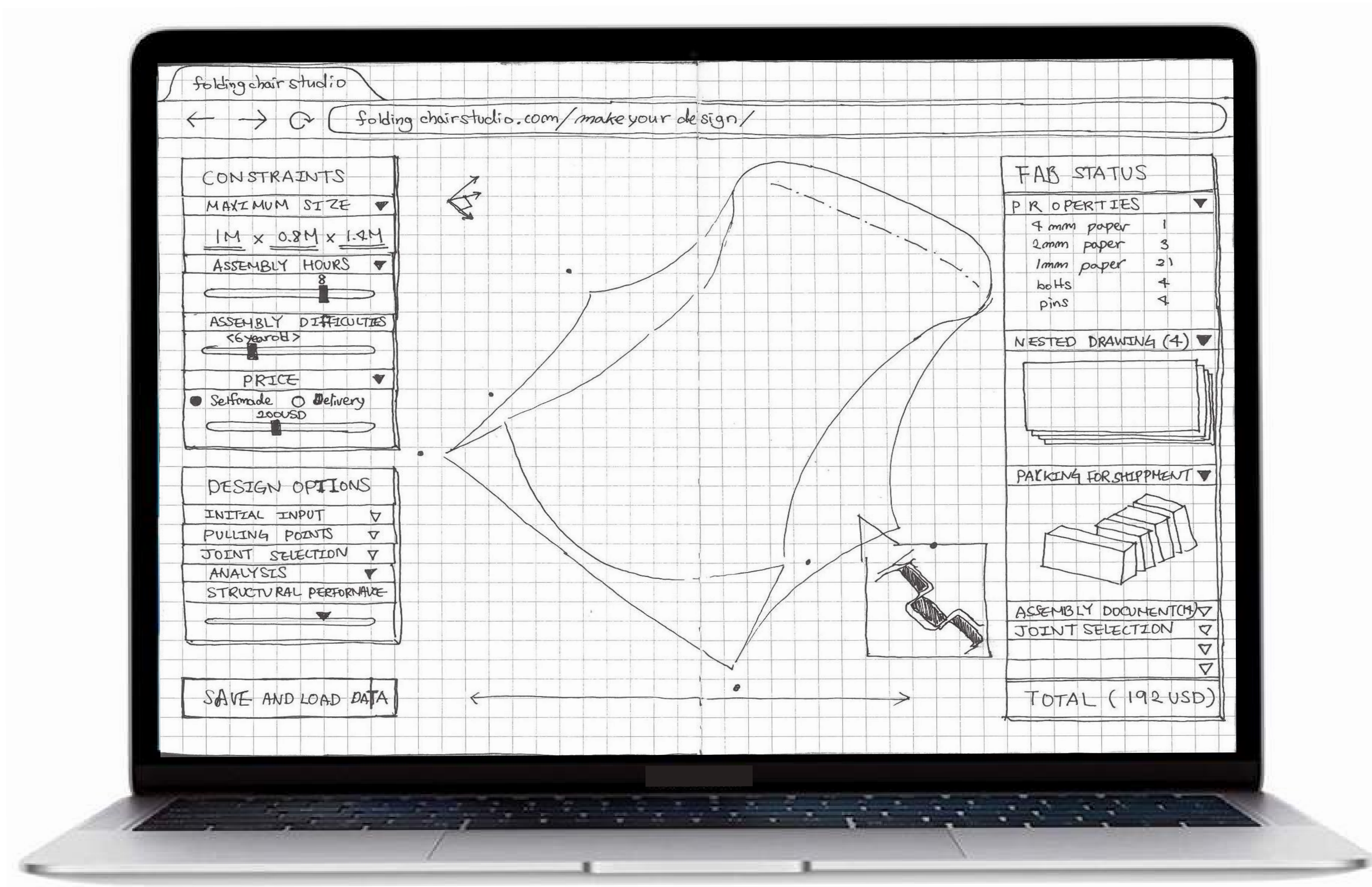
|| Two extreme case geometries created from a single algorithm under same paper size input.

Sangki	Jenny	Kangdacool	Innan	Yeo	Chinso	Yujin	Hwitticus	Susana
Architect	Cultural planner	Doctor	Computer engineer	Architect	Product designer	BIM consultant	Computer engineer	Student
GSD candidate	Adorable fiance	Math nerd	Ambitious guy	Handsome guy	Bibim-bob lover	Animation lover	Free spirit	Sam Smith fan





In my study of architectural design, I have met a lot of great friends in various academic fields who have always supported me in my intellectual pursuits. In December 2018, I asked them to create their folding geometry for a New Year letter (sculpture) for them. We had a quick Skype call and made interlocking folding papers with their numeric input selections. Even under the same design rule and paper size, each output was entirely unique (as seen from the images above). Can the design process of architecture resemble this - where the average customer can simply visit a website, play with geometry and easily create their furniture, facade and even buildings?



The geometries of the cards are created using relatively simple trigonometric functions and made by paper under standalone Rhino Grasshopper programs. I am looking forward to designing more dynamic and fabricate-able geometries on the Internet for everyone. I want to begin my research by designing generative object made of flat sheets - thin aluminum/steel sheets or wood boards. Also, I want to build a foundation for understanding material properties of the material to utilize those for nonlinear fabrication process of my generative design.

After studying at the interdisciplinary environment of architecture, mathematics, and computer science in Master of Design Studies program in Harvard University, I want to be a designer who provides generative design objects on the Internet made of affordable materials for people.