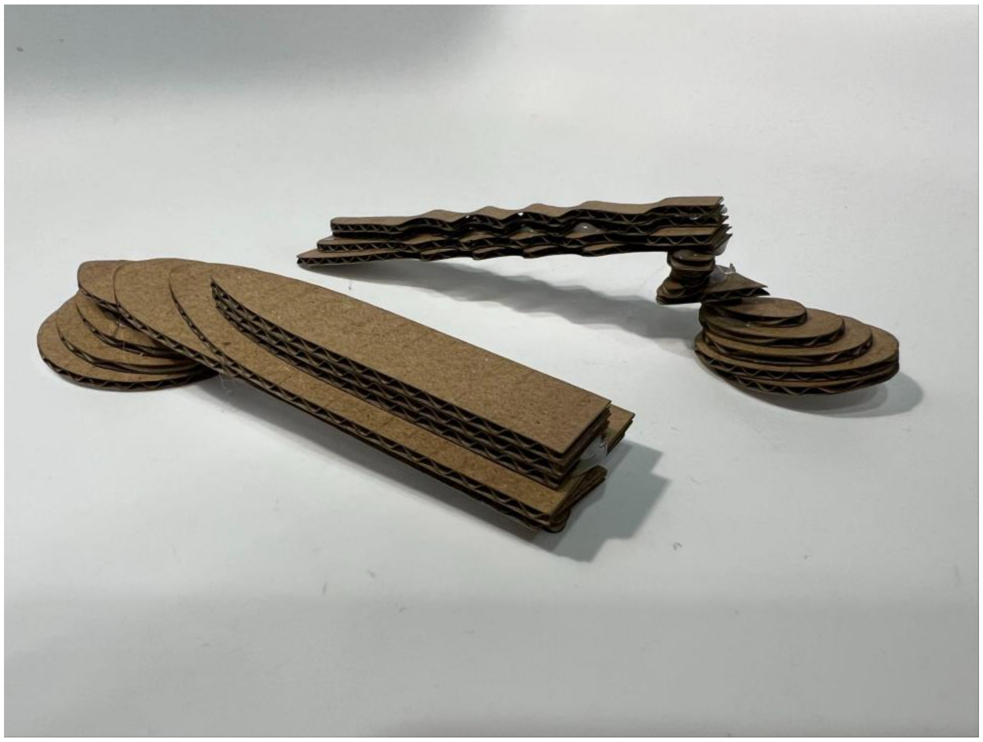
L2 — Parametric Slicer   L2.gh

Using Grasshopper, a built-in Rhino plug-in, the second large assignment focused on using parameterized, computationally generated 3D objects, in conjunction with a layer slicer, to create laser-cut, stacked objects. This project allowed us to play around with the Grasshopper software, understanding its capabilities and potential. Further, it revealed a different method of generating sliced objects using geometry and planes to slice out contours for the laser cutter.



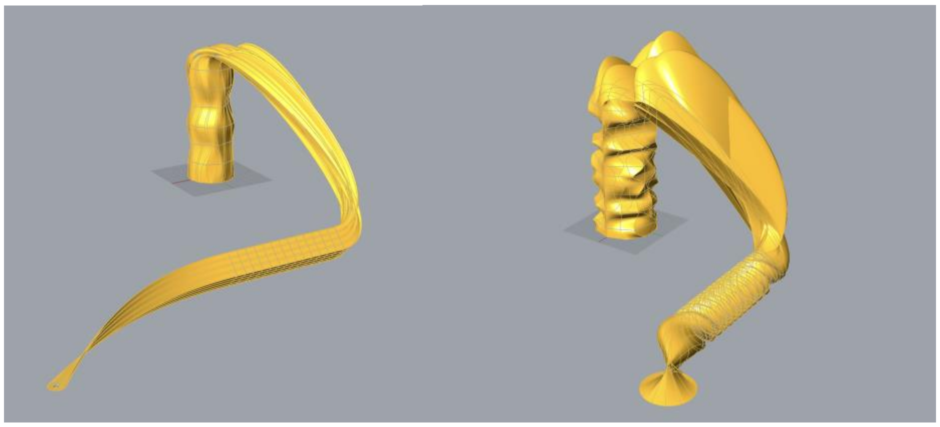
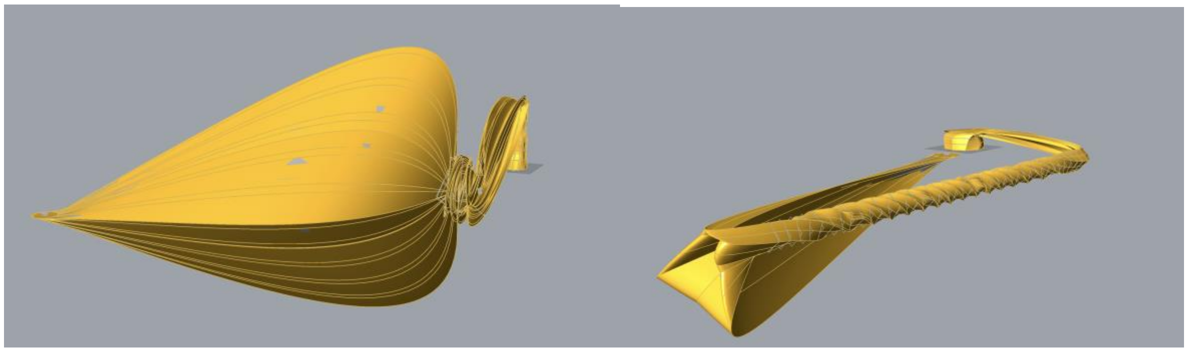
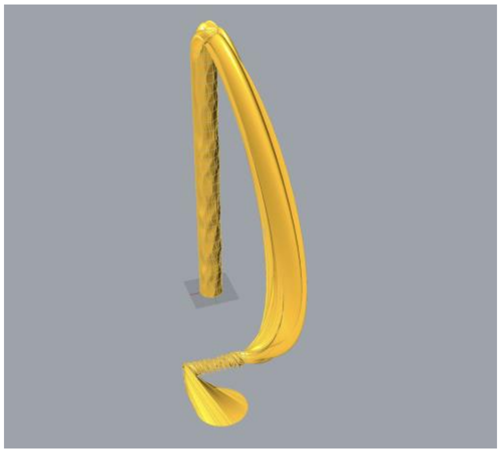
Materials Used

- Cardboard (~2mm thick)

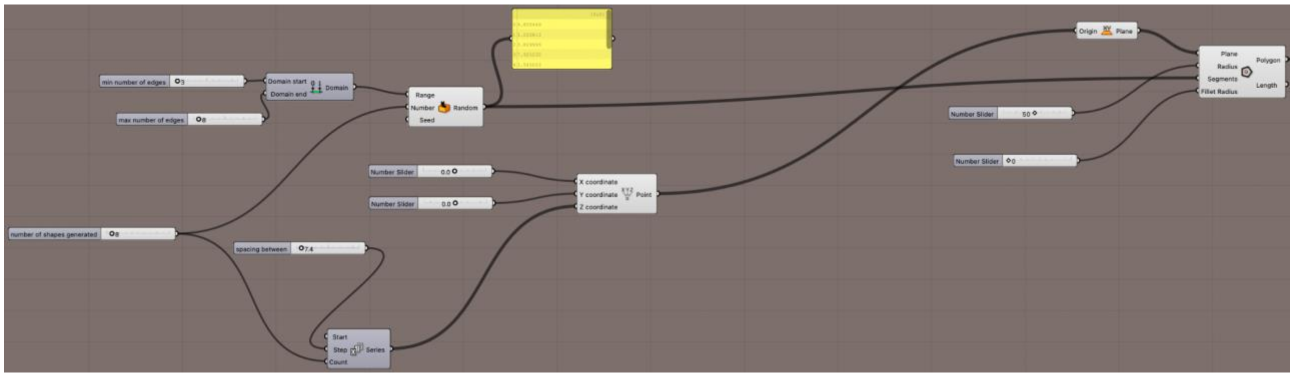
Math Used

- Geometry (areas, volumes, centroids...)
- Cartesian coordinates (points, lines...)

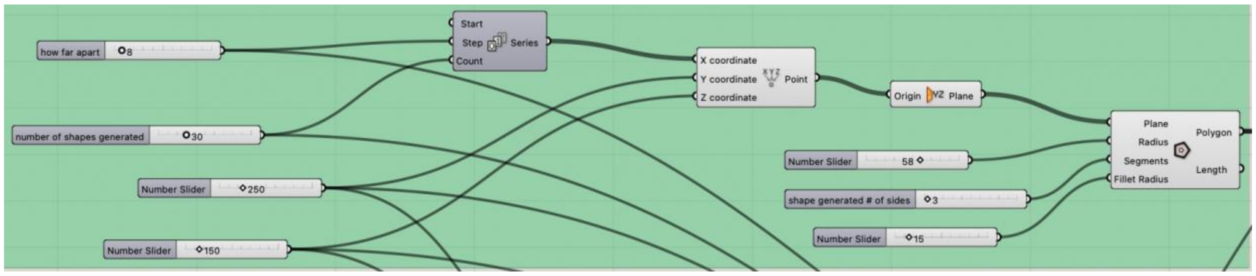
Task #1

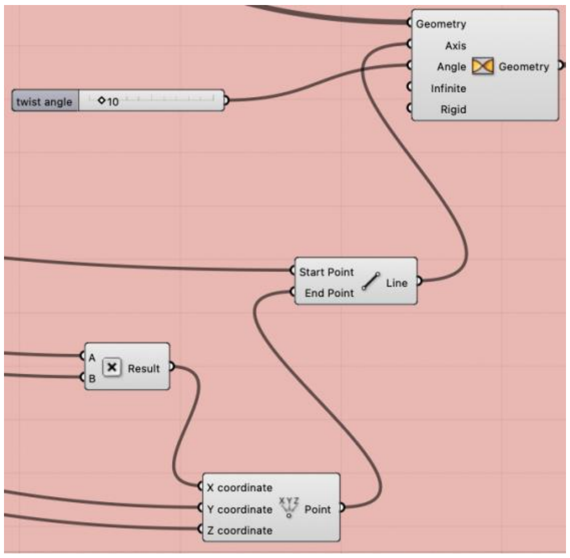
The first task of this assignment centered on generating 3D objects based on a minimum of 10 parameters. In doing so, we would then generate 5 different iterations by altering the parameters of our initial model. For mine, I began with a column generator in which each step generated a different shape based on my parameters, then tying it into a twistable shape stack, culminating in a circle with the ability to move around the XY-plane.



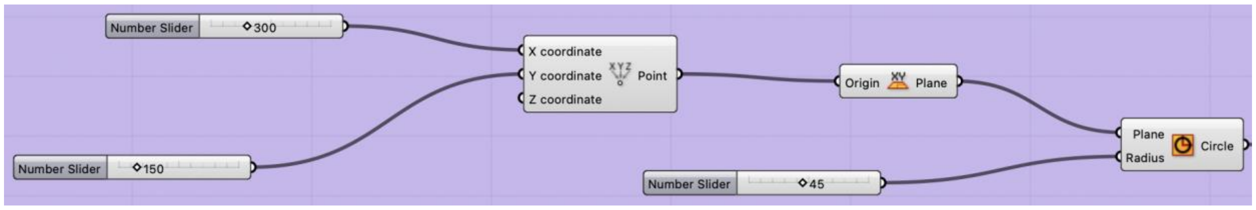
My first set of parameters came through my random, regular shape, column generator. In setting the domain for the minimum and maximum edges allowed for each shape, we are provided our first two parameters. Next, for the third parameter you can set the amount of shapes generated, and in the same breadth for the fourth, you can set how far they are spaced from one another. These four, in conjunction with the x-coordinate and y-coordinate locations, provide our fifth and sixth parameters while allowing us to generate the majority of the column—its location, size, number of layers. The final step to generate the column is to actually generate each shape, providing our seventh and eighth parameters in the shape radius and the filet radius of them.



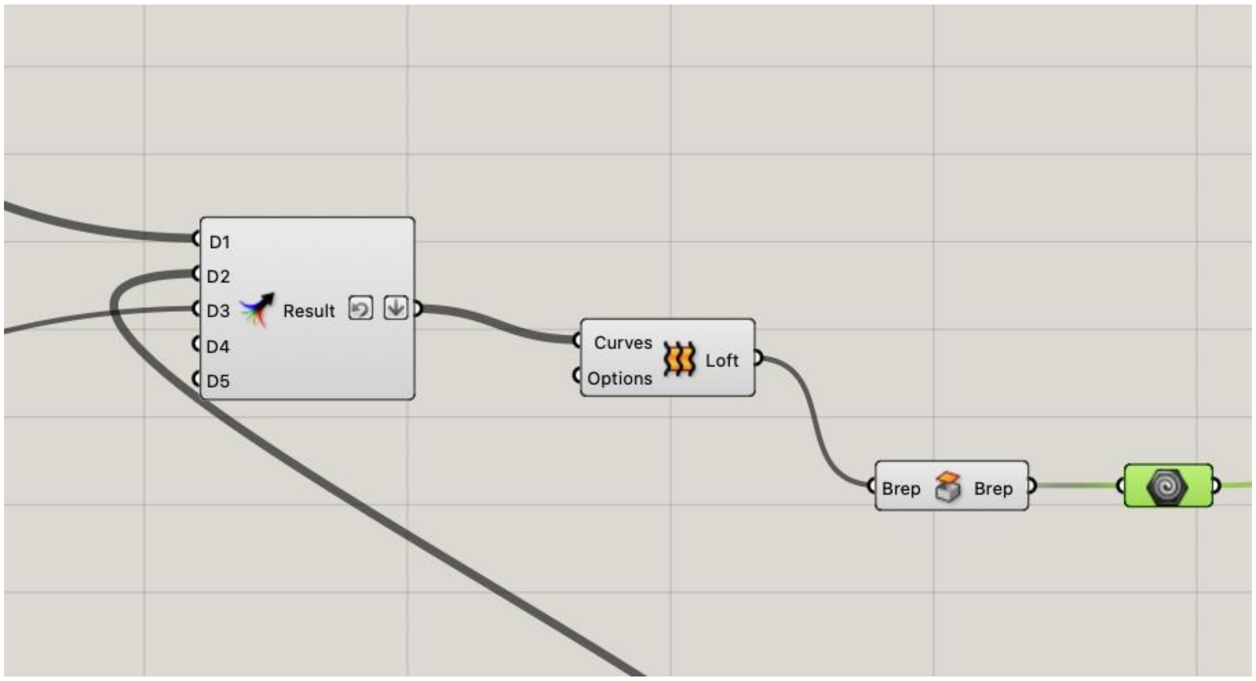
For the next few parameters, I decided to add a stack of one singular shape which I could eventually use for a twist. Here, we find our next 7 parameters, most of which perform very similar functions to the shapes of the columns, just without the variability of edge count.



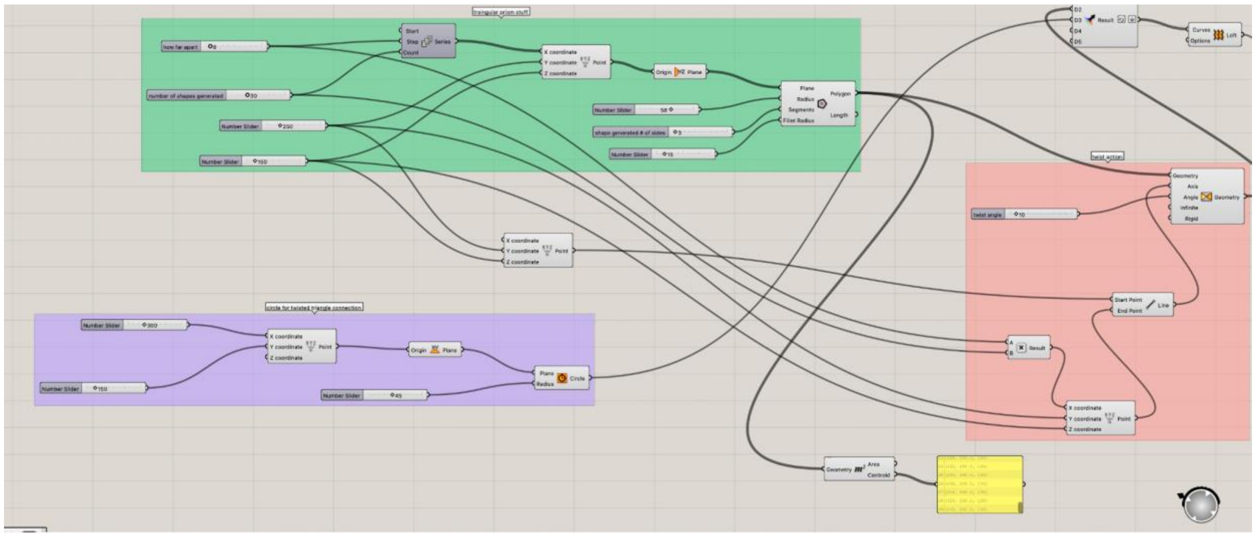
After generating the shape stack, I could then move into generating the twist function for it. Because it was a uniform shape, there was no need for a bounding box, instead requiring the centroid of the top and bottom layer to draw an axis between, allowing for the twist action to occur based on the angle. This introduced another parameter in the angle itself.



Finally, I decided to add a simple circle on the XY-plane which could move around to allow for funky, unpredictable, lofting as the data from each of my three sets was collected. This introduced my final three parameters, coming from the x and y-coordinates, as well as from the radius slider of the circle itself.

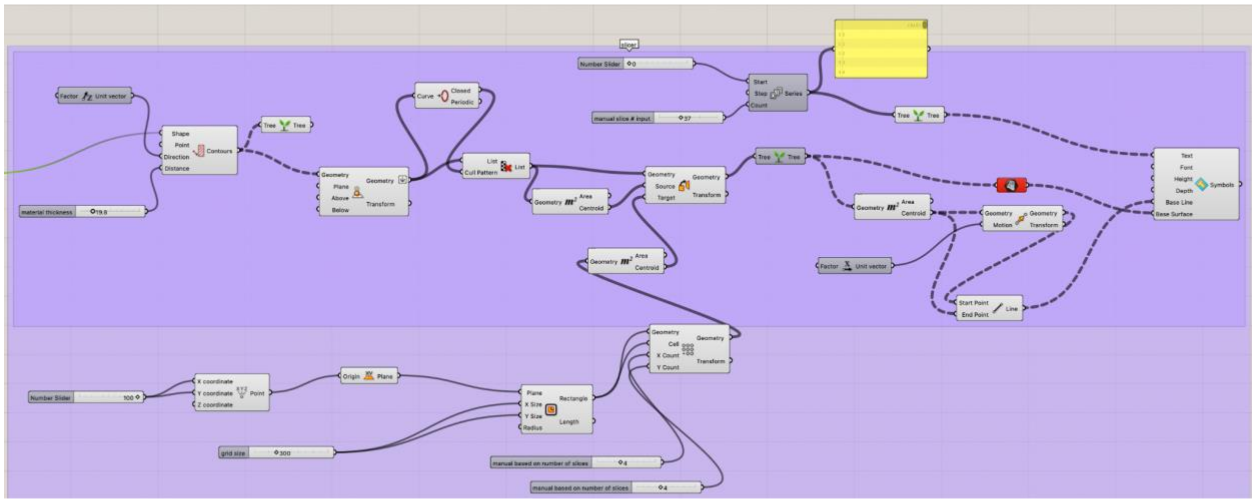


Finally, in order to combine each of my three shape generations, I fed each data stream into a "result as tree" in order to loft them together as one object, capping the holes, and creating the final geometry to be sliced.



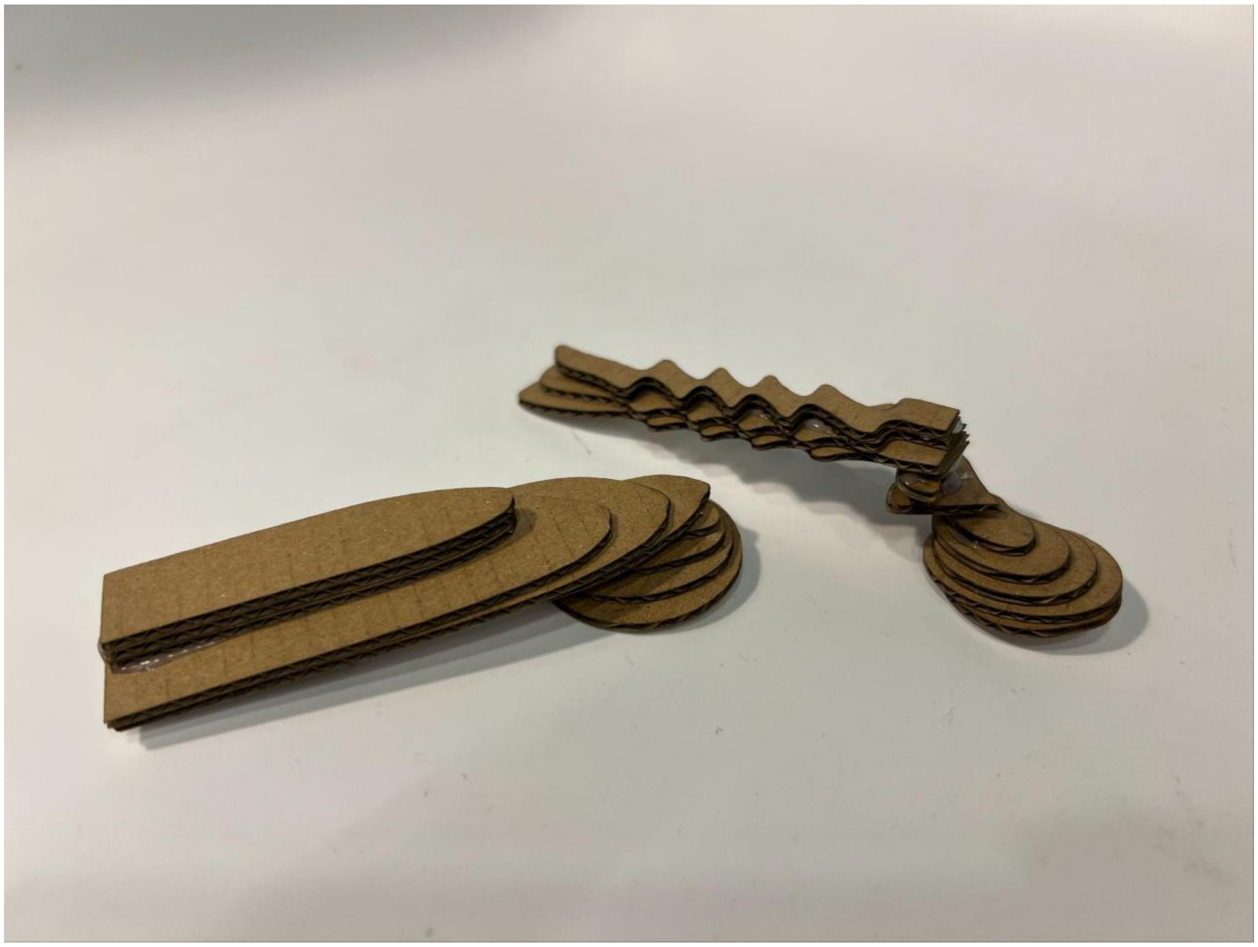
Pictured above is the twist action in full view, as well as the circle generation joining the result tree.

Task #2

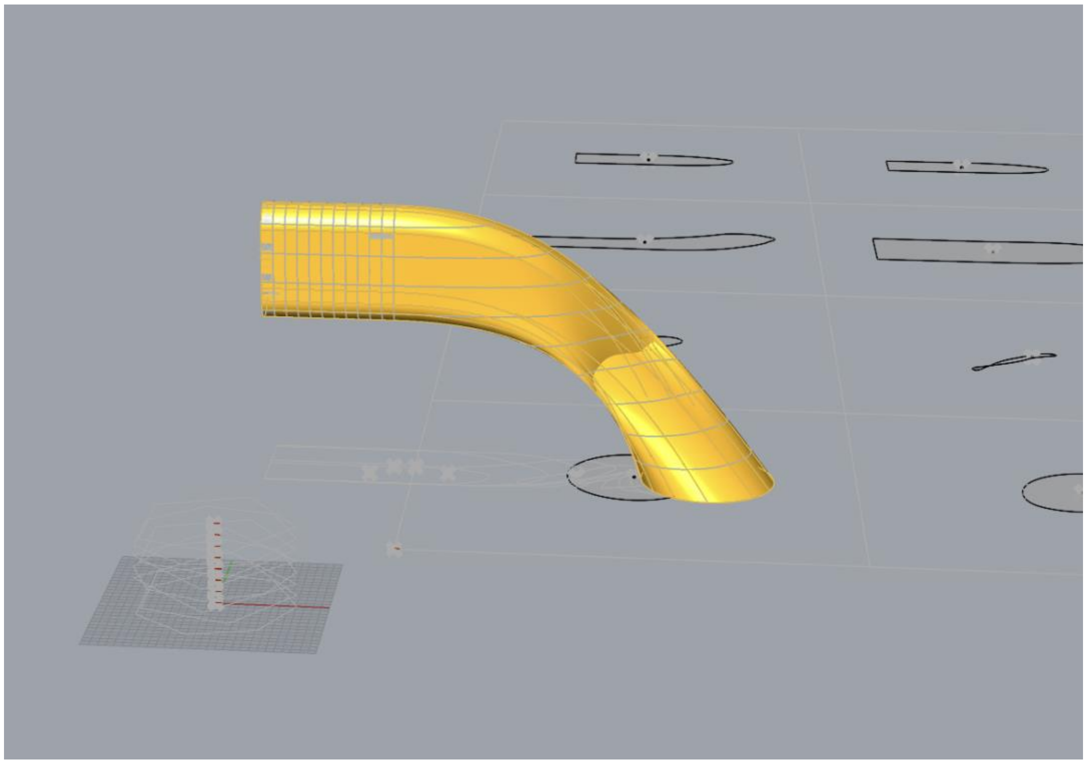


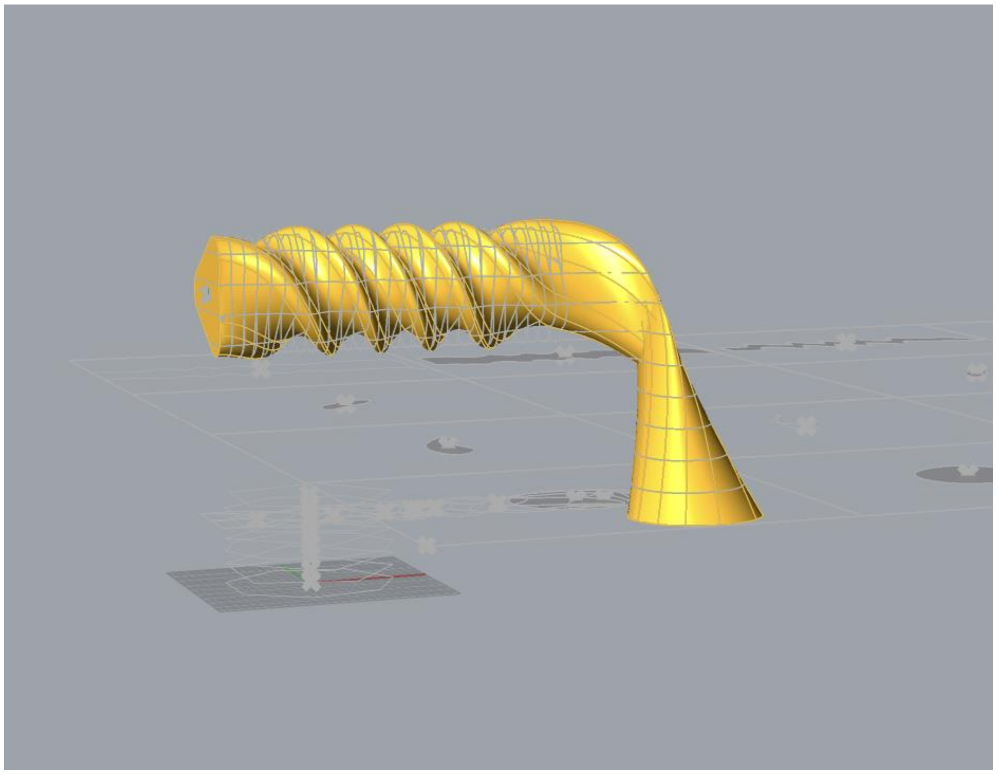
Task two of the second large project asked us to create a z-axis slicer to generate contours of our shape based on a set material height. Each slice was to be laid out on the XY-plane to allow for the file to translate into something laser-cuttable. I began by generating contours based on my 3D object turned geometry. Using the z-axis to slice and a user input of layer thickness, a certain number of contours is generated. This number is input later for the number of cells used to place down each slice, as well as to provide the correct number of text-generated-numbers for each. Before this, however, the contours are moved "to plane" where they are then sorted in a cull pattern using closed curves as the boolean true/false value, allowing us to ignore unclosed, unhelpful curves generated. From here, we can move to the lower level of my flow chart where the cell grid is generated to lay out each piece. I began by generating a rectangle, and then a cell system within it, which has parameters adjustable to number of slices, size of slices, and x-y-coordinate positioning for the layout itself. Using the centroid of each slice and of each cell, we can place each cut in the center of its respective cell. Similarly, using the centroid, a graft tree for each surface to be separated, and that initial number of slices, we can use "text on surface" to place a number on each cut for ease of fabrication after the pieces are laser-cut. I would also like to note how series functions introduced ridiculous amounts of variables (thousands) which slowed my Rhino down greatly for an unknown reason, forcing me to manually input the number of contour layers generated each time.

Task #3



For the third task, the goal was to fabricate two of our five generated objects. For mine, I decided to generate a sixth and seventh without the initial shape generator. With the twisted shape and the circle, I still met the 10 parameter minimum requirement, but allowed for easier computation and generation within my Rhino file, as well as providing myself the opportunity to reasonably fabricate something without an unnecessary mess of unclosed and/or improper curves. Overall, generating the slices to be cut was straightforward after getting the slicer working, however, I found my text-on-surface numbers to be breaking my laser cutting files. And so, I did delete them in Rhino before laser-cutting so my 3dm files would properly output. Nonetheless, the numbers do appear as they should in Grasshopper and Rhino. Below you will find each generated shape and its fabricated result:





Problems/Conclusion

In conclusion, I have grown to quite enjoy the capabilities housed within Grasshopper. Still, this project did come with its fair share of trouble. Most of my trouble was technical, and saw Rhino lagging to ad-infinitum after declaring ridiculous numbers of variables, or when it would generate ungodly amounts of polysurface curves for the number text when cutting. All in all though, after working through the technical difficulties, the process was more or less smooth. The slicer threw me through a few loops, mostly in figuring out the closed curve cull pattern, as well as in the grid layout and number attachment. Although once it was functional, it was functional. I wish I would have simplified my designs for the slicer, but I did genuinely enjoy the crazy generation power granted through my multitude of parameters and inputs.