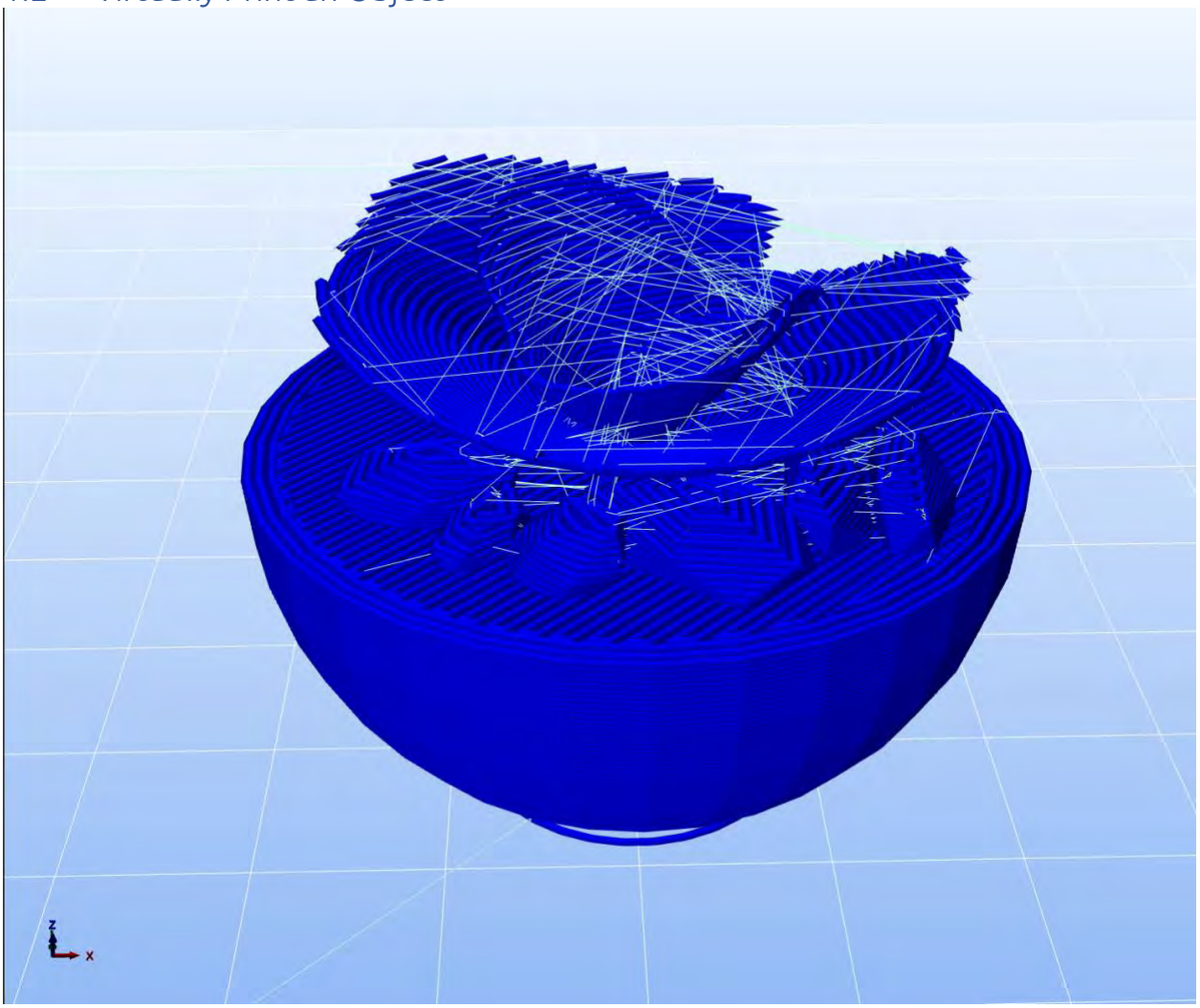
From Design to Machine Instructions

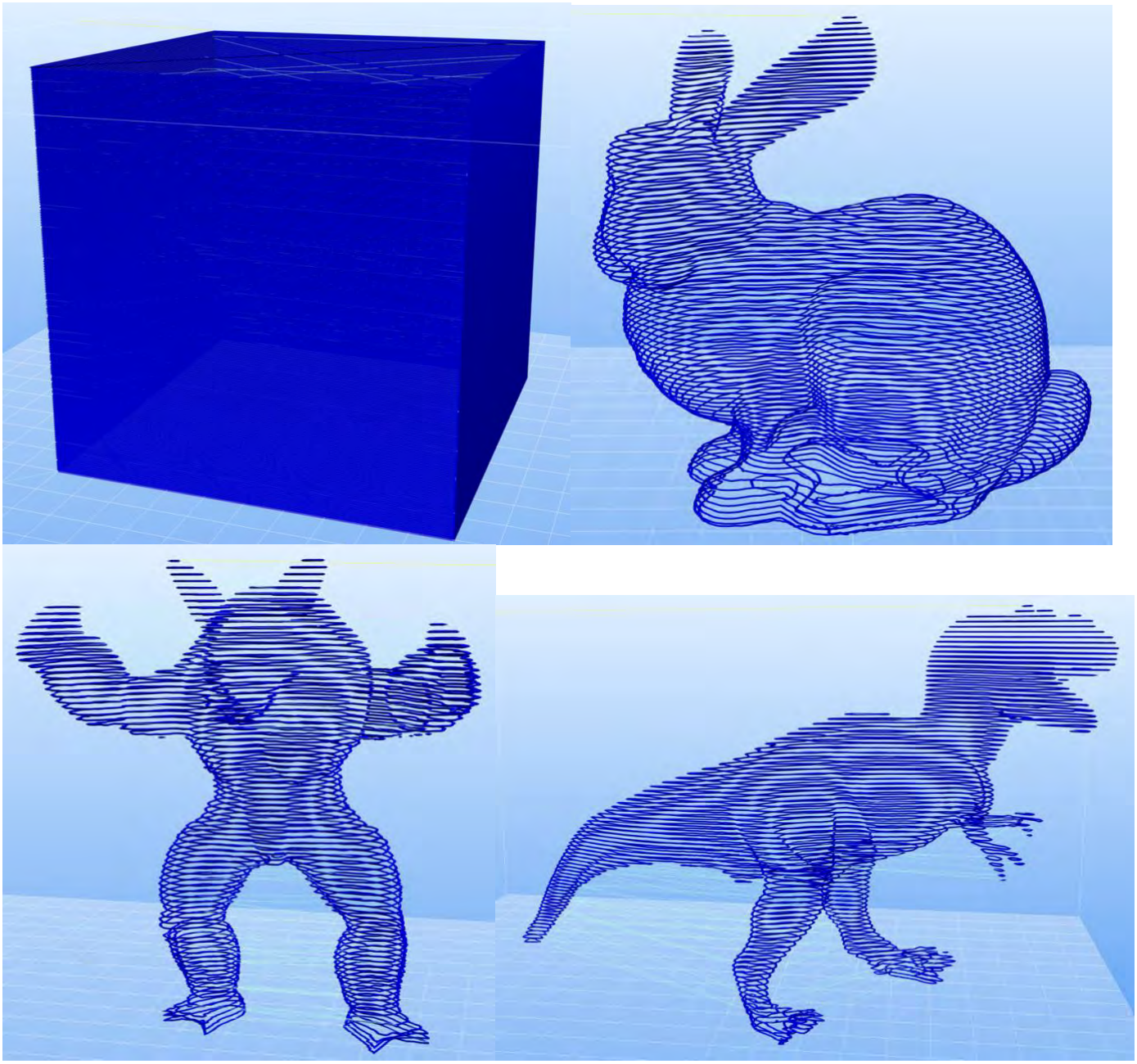
1. Printing with a Virtual Printer

1.2 Virtually Print an Object



2. Mesh Slicing

2.1 Slicer results



Edge case handling: edge orientation, duplicate edges, and the first and last layers were the sources of edge cases. To get rid of edge orientation (which caused situations where a contour would trace back on itself) I sorted the start and end vertices by their distance squared to the origin. This standardization allowed me to find duplicate edges and remove them to help with processing and redundant edges. I also then added an extra case for the first and last slices where I forced triangles to be built to prevent bad ordering of edges to create slices that couldn't be created.

2.3 A source of failure

A soup of triangles has no requirement to be part of a manifold mesh which can cause weird slices to occur. One example of non-manifold meshes would be a scenario where two edges intersect across each other as opposed to at a vertex (like the letter "T" but there is no vertex at the intersection). The current implementation on contour generation assumes each edge share common vertices and that's how they link together. A way to detect this error would be to run line-

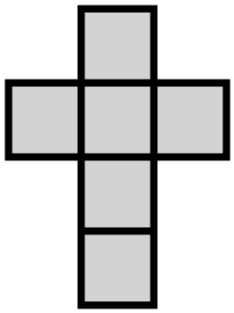
intersection math on each of the edges to detect any edge intersections and check if they occur at a vertex. If they don't, split that edge into two new edges with a common vertex at the intersection point. This will solve this potential issue of given a random triangle soup to slice.

3. Abstractions for Sheet Metal Design

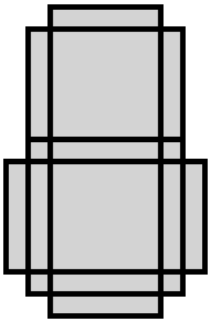
3.1DSL

Tabs of a design will be organized in a tree structure (directed acyclic graph) where the root of the tree is the base tab with no "parent" and all tabs branching from that root piece are connected via bends. A tab internally is defined by an origin point O then two non-parallel 2d vectors A and B. A and B will expand to define the four sides $s_1 = (O \to A), s_2 = \big(A \to (A + B)\big), s_3 = \big((A + B) \to B\big), s_4 = (B \to O)$. These also make up the 4 vertices in the tab. Their orientation is defined by the order they appear in the sides. Child tabs will be initialized by setting a parent, picking one of the four sides s, an offset from the starting vertex of the side d, the angle θ counterclockwise from the vector of s ($0° - 180°$), a length l that is the extent of the tab along the angle θ, and a bend/fold angle α upwards from flat ($0° - 180°$). All these parameters will be used to compute the origin and vectors A and B for the child which will be what gets stored. Child tabs operate off the coordinate system of their parents.

3.3 Cube

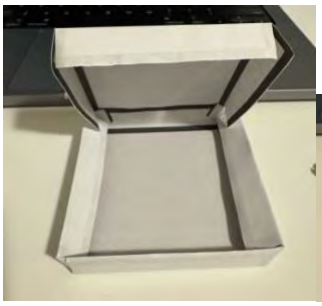


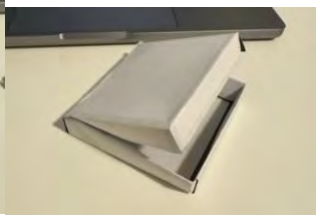
3.4 Pizza Box



Based on this design:

Result:





4. High-Level Discussion about Crafting DSLs for Design and Fabrication

4.1

1. A DSL can help restrict the design space to something that is possible for a carpenter to build based on their tools and their specifications. Providing such a DSL could be useful for that carpenter or their customers when ordering/designing an item because it can ensure that whatever they design will be exactly supported based on the tools they have. Essentially, an impossible order/task would not come from this DSL for carpentry. It should also be relatively readable and help reduce errors in the construction process as the DSL will be specific to the nuances of carpentry.
2. A DSL can help manage a bill of materials as it defines exactly what is needed in the build process. Therefore, a carpenter can use this to define the materials needed to order and how much they cost.

4.2

One thing that this DSL does not verify for are impossible parameters and arrangements. For instance, one could supply parameters that cause a tab to float off in space unconnected to its parent (i.e. when the offset d is greater than the width of the parent/side it is attached to). A simple verifier to check whether $d_c + w_c \leq w_p$ would be all that is needed to verify that the tab c would exist within the bounds of the parent p.

Another example could be that this DSL does not consider the small amount of material used in bending metal. Metal is typically thick enough that bending uses some amount of thickness. Therefore, a design like my pizza box would have problems on some of the folds that fold in on itself as the thickness would cause pieces to be unable to completely fold completely inward on themselves (180°).

AI usage

See previous assignments. TL; DR: used GitHub copilot to assist in code competition (including filling in numpy functions I didn't know exist) but all code intents are my own. I verified all my code.