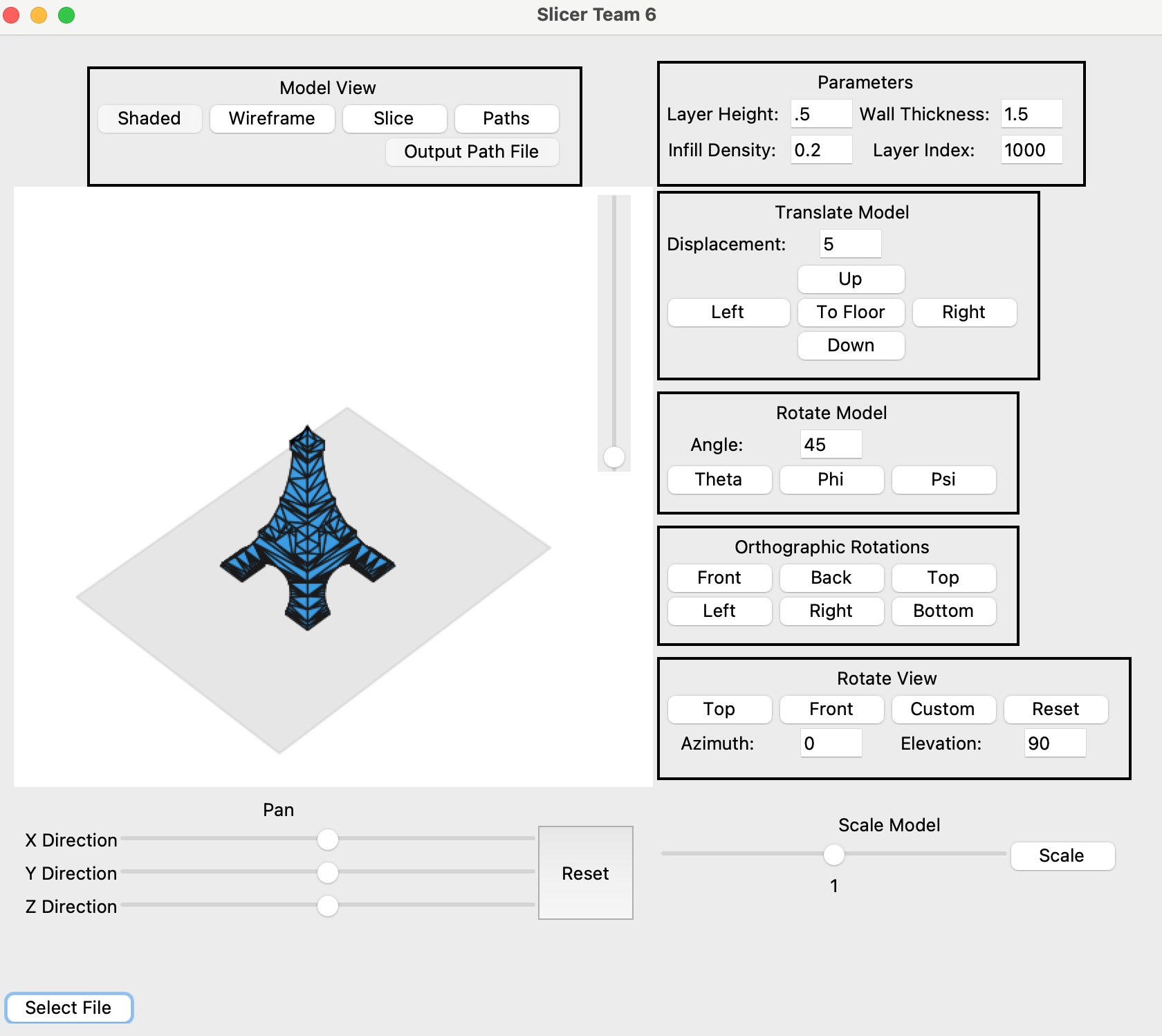
Execution:

The cleanest way to access our code is download from our git repository. You can clone our repo at this link: <https://github.com/jmorris335/DesAuto_Project/>

Unfortunately we were unable to submit a single executable (though everything should in place to do so). In addition to the provided modules, you will need to download the package dependencies (which are all standard, opensource packages like numpy, matplotlib, etc.). You can do this quickly by running <pip install -r requirements.txt> into the terminal after navigating to the root directory. The project requires Python 3+ and tkinter 8.5+.

To run the script, navigate to the src directory and execute \_\_main\_\_.py as a normal python script. This should bring up a very small GUI asking you to select a file. After you select an STL file from the explorer, you should see the following setup shown in the attached figure.

 From here most commands should be self-explanatory.

If you would like to execute a single script, you may be able to run Project Part2 Script.py, though the modules are setup to support the GUI, and not internal callers.

Note that to output the text of the extrusions you’ll need to create the paths by clicking “Paths” and then click “Output Path File”.

Code Management:

The code is organized into a collection of python packages. The GUI files are separate from those interfacing with the STL files. Each basic functionality is expressed in a unique module that are generally called in this order:

1. ReadSTL: Parses an STL file
2. Clipping: Clips the STL to the screen
3. SliceSTL: Slices the STL into slices
4. Extrusion: Creates the infil and extrusion paths
5. Transform: A class for calculating transformation matrices
6. PlotSTL: Performs spatial transformations and plots to a matplotlib plot

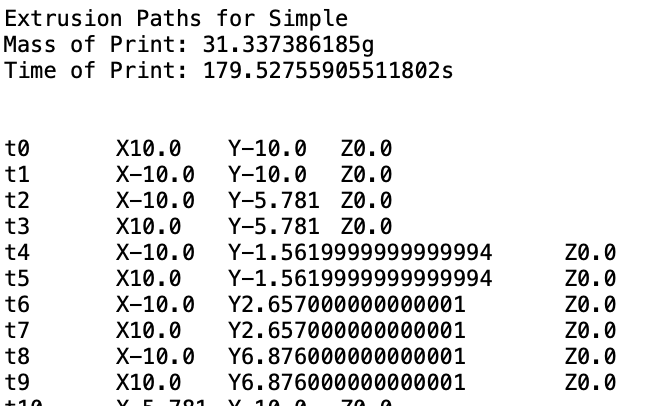
In addition there are modules for:

1. Methods: general methods accessed in multiple modules
2. Offset: Creates an offset of the STL for use in creating walls (not fully functioning)
3. Tests

The GUI files are organized according to the widgets shown on the screen. There organization is self-explanatory.

Extra Features:

Beyond the GUI, the slicer is able to calculate the time and size of the final print (shown in the final output). The slicer also can highlight various slices (for viewing the printing path), which is accessed by changing the layer index in the GUI.



Example text output

Caveats:

Note that we can perform clipping on the STL file, but you will not see our functionality utilized in the GUI (since matplotlib does clipping inherently). But clipping can be seen in the Project Part2 Script.py file. The file also does not do isometric, dimetric, or trimetric projections. In addition, if you want to reset the plot after scaling, you will need to click the “Reset” button twice.

The slicer does not hide hidden lines, though in shaded view it only presents the visible surfaces. (This is due to functionality in matplotlib, not z stacking rendering was implemented by the team).

There are additional caveats when creating paths. The algorithm to do so is very slow and will execute every time the plot updates. Therefore, with large STLs, it is better to test transformations on the shaded or wireframe models. They are possible with the paths, but it will take quite some time to perform. The extrusion paths are plotted to scale, and are the thickness of the layer height. If the paths seem unreasonably big, it is likely due to the layer height being too large. There is currently no method of addressing this. Finally, the STL will not slice properly if it is below the Z plane. Make sure to click “To Floor” before slicing or creating extrusion paths.

Finally, the slicer does not create a full shell of the model, rather creating infill at every slice. The offset feature needed to create shells was almost operational, but creating singularities for about 5% of the walls it was offsetting.