Blocked: An STL to block structure converter for faster prototyping and increased interchangeability of 3D printed parts

24-783 Spring 2023 Project Proposal

Unoptimized - Abraham George, Amirpouya Hemmasian, Charlotte Avra, Connor Geshan, Yayati Jadhav

Motivation

Our proposed program is motivated by an engineering problem that we are facing in the Mechanical and Artificial Intelligence Lab (MAIL), where we are working on additive manufacturing using a drone. For the project, we are planning to have the drone assemble structures out of pre-manufactured (through 3D printing) building blocks. However, in order to do this, we need to have a way of taking the goal structure (in the form of an STL file) and figuring out how best to build it out of a set of predefined blocks.

To achieve this goal, the proposed program seeks to convert an STL file into instructions for building it using blocks. The program plays a similar role to a 3D printer's slicer program, which converts a 3D model into instructions for the printer to follow. This program is not limited to the specific application of drone additive manufacturing but can also be useful for building any structure out of smaller units.

Proposed Solution

Our proposed program will input an STL provided by the user and convert it to a block structure. At its simplest, the block structure will include 1x1 bricks as shown in Figure 1.

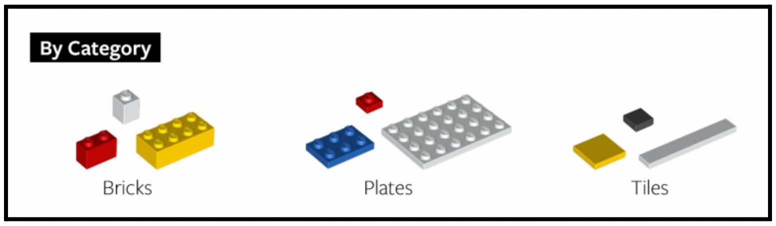

Figure 1: Possible brick types and dimensions

For a more complex implementation, the graphical user interface (GUI) will also allow the user to select what other block types and dimensions they wish to include in the final model (e.g. 2x4 bricks, 1x3 plates, etc.) as shown in Figure 2.

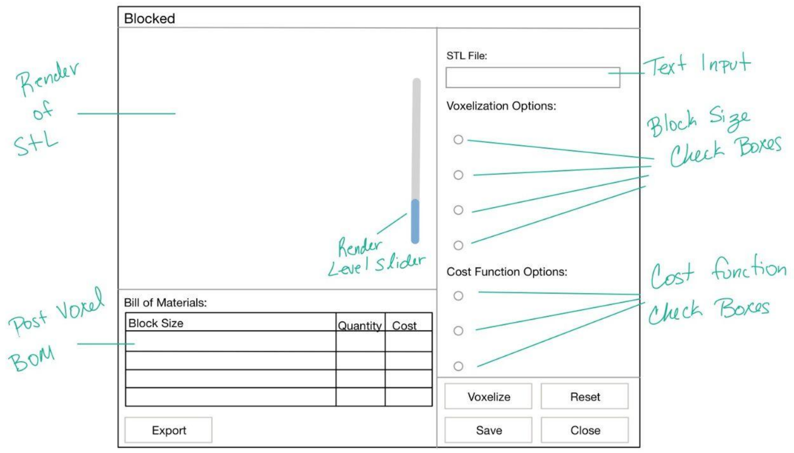

Figure 2: Sketch of graphical user interface

Once the block structure is complete, the user will have an option to run a structural stability analysis. This provides the user with a rating of how stable the object is in the input orientation. The program will also output a bill of materials (BOM) including each type of block and the quantity required. The BOM will also include the amount of filament required to print each. Lastly, the program will provide the user with layer wise steps of how the model is built from the base up.

Careful planning is essential to ensure stability and efficiency during the assembly of complex objects. In this project, we propose the use of an algorithm based on the Testuz et al. method to optimize the layout of voxels for achieving maximum stability during the assembly process. The proposed algorithm comprises three main steps. First, the STL file is converted to point clouds. Second, the point cloud is voxelized to a block of 1x1. Finally, the layout is determined based on an algorithm for assembly with the objective of maximizing stability, such as the one presented by Testuz et al.

Justification

This project will include combining FsSimpleWindow with a standard GUI library, rendering of a STL file, generation of multiple cost functions to optimize block selection and assembly/stackability, selection of block type to minimize overall fabrication cost, recompilation of voxelized parts into a single structure, and stability analysis post voxelization. Because of the aforementioned aspects, this project will be of meaningful difficulty not only for the course, but also for a group size of 5.

Combining FsSimpleWindow with a standard GUI library would involve integrating the functionalities of FsSimpleWindow with the widgets and tools of a pre-existing GUI library, such as Qt or wxWidgets. This would enable the creation of more complex and versatile user interfaces for applications built with FsSimpleWindow.

Rendering and voxelizing an STL file involves converting a surface mesh into a set of voxels or 3D pixels. This can be done using a variety of algorithms, such as marching cubes or octree-based methods. Voxelization is useful for our 3D printing and assembly use case.

Generating a cost function to optimize block selection and assembly/stackability will require a clear understanding of the design requirements and constraints. Some cost functions to consider might include material cost, production time, and ease of assembly. An optimization algorithm, such as a genetic algorithm or simulated annealing, could be used to search for the optimal block selection and arrangement.

Recompiling voxelized parts into a single structure would involve merging the individual voxel grids into a unified structure, which can be done using various techniques such as boolean operations, interpolation, or surface extraction.

Stability analysis post-voxelization would involve evaluating the structural integrity and stability of the final voxelized structure. This can be done by applying gravity to the rendering simulation ,or other forces. Additionally, ensuring each block has support from below using a 2-D grid layout or checking supports of each overhang.

4. References

[1] Brickify (https://brickify.it/)
[2] Testuz, R. P., Schwartzburg, Y., & Pauly, M. (2013). *Automatic generation of constructable brick sculptures* (No. CONF, pp. 81-84). Researchgate.net