UnBox3D:
Software Design Document

Version
4.0.0

Group 1: Alexander Ramirez, Vivian Cases, Jacky Lim,
Nicholas Sisneros, Vince Wang

Contents

Revision History

Name	Date	Reasons for Changes	Version
Alexander Ramirez	2025-06-01	First Draft	1.0.0
Vivian Casas	December 11, 2025	Update for snapshot 2	2.0.0
Vivian Casas	December 11, 2025	Update for snapshot 3	3.0.0
Alexander Ramirez	December 11, 2025	Update for snapshot 4	4.0.0

1 Introduction

1.1 Purpose

This document shows the general structure and specific implementation details to provide clarity on the project's direction and targeted features. Version 4 finalizes the document, where we add model exporting and finishing touches to the user interface. This document will ensure developers and stakeholders alike agree upon the direction of the program's development.

1.2 Intended Audience

Our intended audience includes:

- Software Developers
- All stakeholders
- Project Managers
- Quality Assurance Teams/Testers

1.3 System Overview

UnBox3D is a software tool that bridges the gap between digital modeling and physical assembly. UnBox3D allows a user to import a model in .obj format and have it displayed in a special viewport where the user is given the option to simplify the imported model and, once satisfied with their simplifications, export the model to 2D cutouts for fabrication.

2 System Architecture

This section provides a high-level overview of UnBox3D's architectural design.

2.1 System Purpose and Goals

The program seeks to:

- Allow simplifying models

- Automate the unfolding process
- Provide visual feedback during simplification

2.2 General Workflow:

1. **Import Model:** Users import an .obj model into the scene.
2. **Model Parsing:** Assimp is used to parse the imported model's data and create a mesh representation that can be manipulated within the application.
3. **Model Rendering:** The model is rendered in the viewport for the user to begin the simplification process.
4. **Simplification:** The user now has the option to simplify the model by removing objects that are smaller than a certain threshold, or by Blender's decimation modifier where a given object has its geometry simplified while keeping the topology similar to before based on faces' angle constraints.
5. **Export:** The model is passed to Blender where it is unfolded using a Blender extension that automatically unfolds geometry islands and marks the cuts and folds for reassembly.

2.3 Data Flow:

1. When the user imports a model, an Assimp mesh is created
2. The Assimp mesh passes its data to a g3 mesh (DMesh3) so we can modify the data during simplification
3. If simplification is requested, the g3 mesh is altered accordingly:
    - For bounding-box filtering, objects with dimensions below the user-defined threshold are removed from the g3 mesh.
    - For Blender decimation, the relevant data is sent to Blender via its Python API to apply the decimation modifier, and the simplified mesh is retrieved back into the g3 mesh.
4. Every frame, the data is passed from the g3 mesh to an AppMesh (a custom data container created by us)
5. The AppMesh data is passed into buffers for the GPU to access, with the help of OpenTK

3 User Interface

This sections describes the UI and how it is implemented.

3.1 Overview

There are 2 main screens in this program: the splash screen and the viewport.

- **Splash Screen:** The logo is shown as the program loads
- **Viewport:** The viewport features an Import button, a region where the scene is rendered, and an Export button for unfolding.

3.2 Viewport Refreshing

The program is able to refresh the viewport effectively thanks to OpenTK, a wrapper for OpenGL. It takes the data from AppMesh instances and constantly updates the buffers for the GPU to render objects accurately at every frame. After simplification, the viewport reflects the updated model in real-time.

4 Glossary

- UI - User Interface
- Assimp - Open Asset Import Library
- OpenTK - Open Toolkit Library (a C# wrapper for OpenGL)
- Bounding-Box Filtering - A simplification technique that removes objects based on their bounding-box dimensions.
- Blender Decimation Modifier - A tool within Blender that reduces the number of polygons in a mesh while attempting to preserve its overall shape.
- g3 mesh (DMesh3) - A data structure from the g3Sharp library used for 3D mesh representation and manipulation.
- AppMesh - A custom data container created to hold mesh data for rendering.
- GPU - Graphics Processing Unit
- SVG - Scalable Vector Graphics
- Blender Python API - An interface that allows for scripting and automation within Blender using Python.
- OpenGL - Open Graphics Library, a cross-language, cross-platform API for rendering 2D and 3D vector graphics.

5 References

- Software Requirements Specification (SRS): SRS Link
- Original GitHub: Link