1. Purpose

This document outlines the design information of the Cast Engine project. The software was developed during the Fall of 2021 in order to have an api agnostic way to render computer graphics *scenes*[1].

2. Scope

2.1 High-level overview

The scope of the Cast Engine is to have a UI that allows the user to create scenes in an intuitive way. They should also be able to do this creation in an api agnostic manner while maintaining consistency between api's.

2.2 Requirements Overview

- Using the UI
    - Insert different types of objects into the scene
    - Drag and drop different types of lights
    - Configure lighting and rendering settings
- The Underlying Engine
    - Abstracted use of:
        - ShaderPrograms
        - Renderers
        - Windows

2.3 Assumptions & Limitations

The engine will be limited to only Windows machines. We assume a functioning Windows 10 build & a working MinGW 64-bit compiler. The engine makes heavy use of the mentioned C++17 features, so it will not compile earlier versions.

3. Solution Design Overview

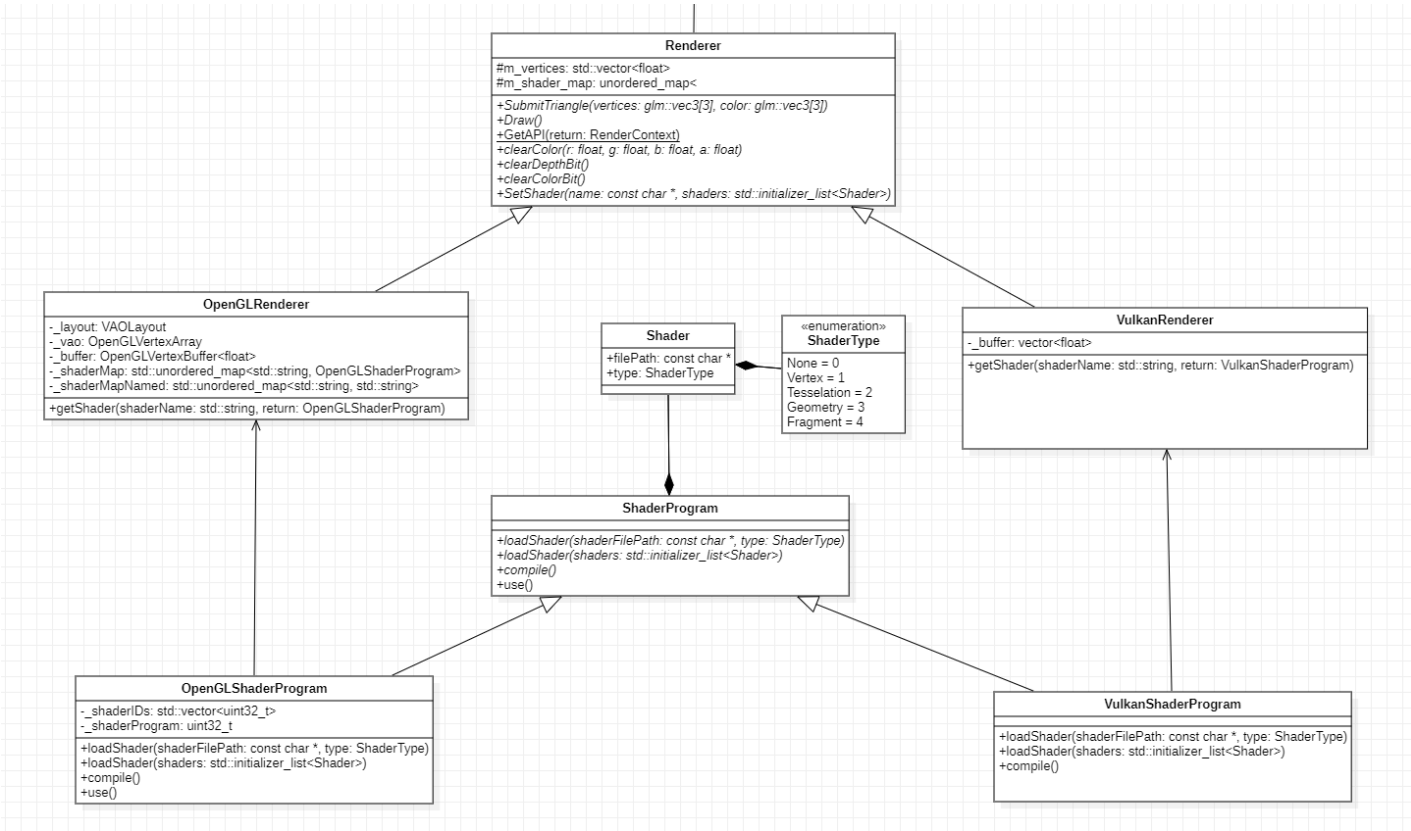
3.1 Software

The compiler of choice is limited to MinGW 64-bit, and uses C++17 & C compiled with g++. The build system uses CMAKE and Makefiles with help from bat files for scripting most of the building. The project also uses python in the build process to locate new files and allow for debugging and testing to go faster during development, however, this will not be shipped in release.

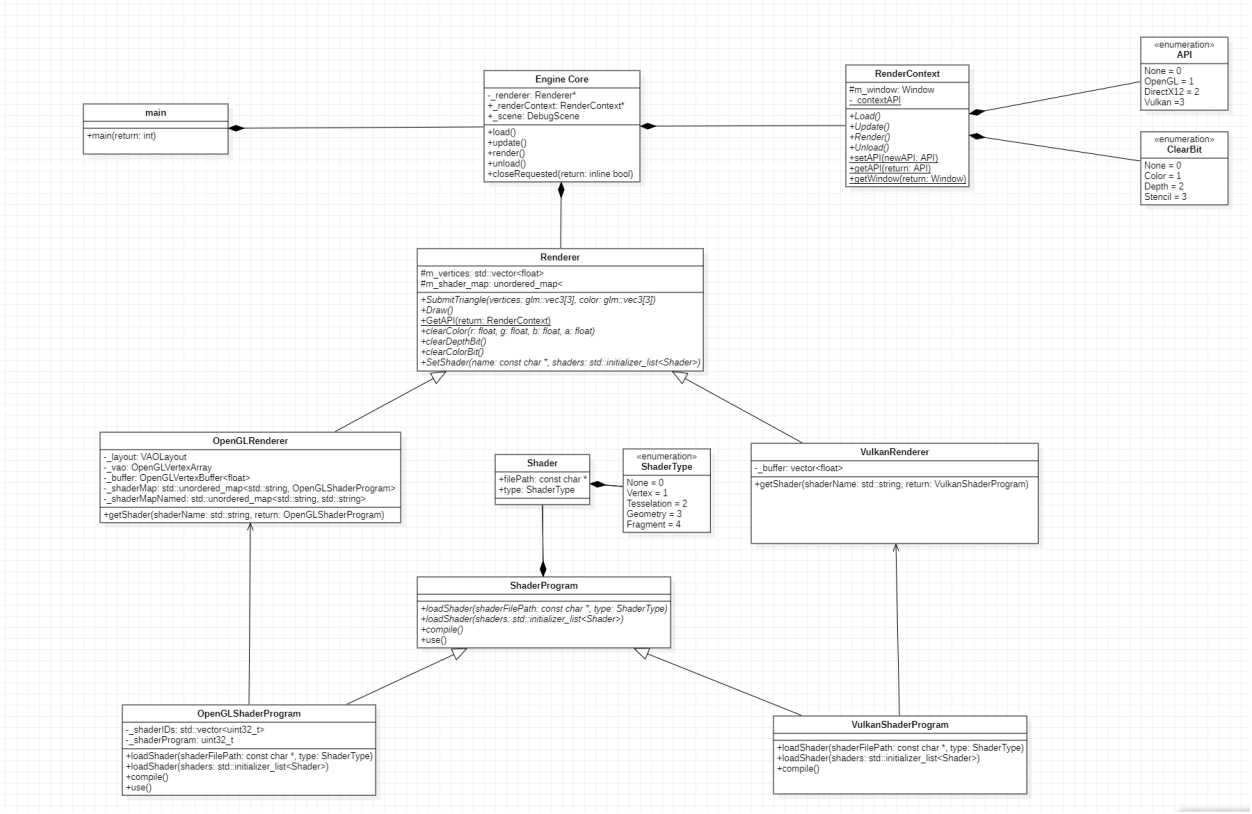
4. System Architecture

4.1 Development View

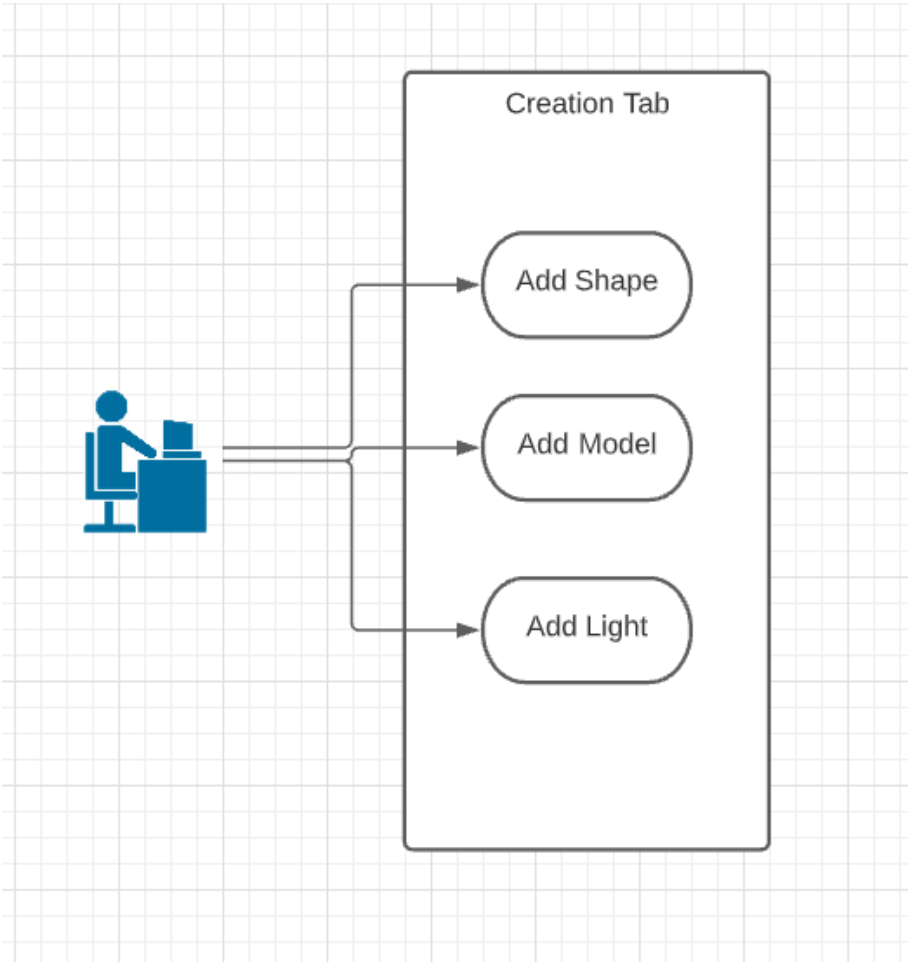
4.1.1 Render API

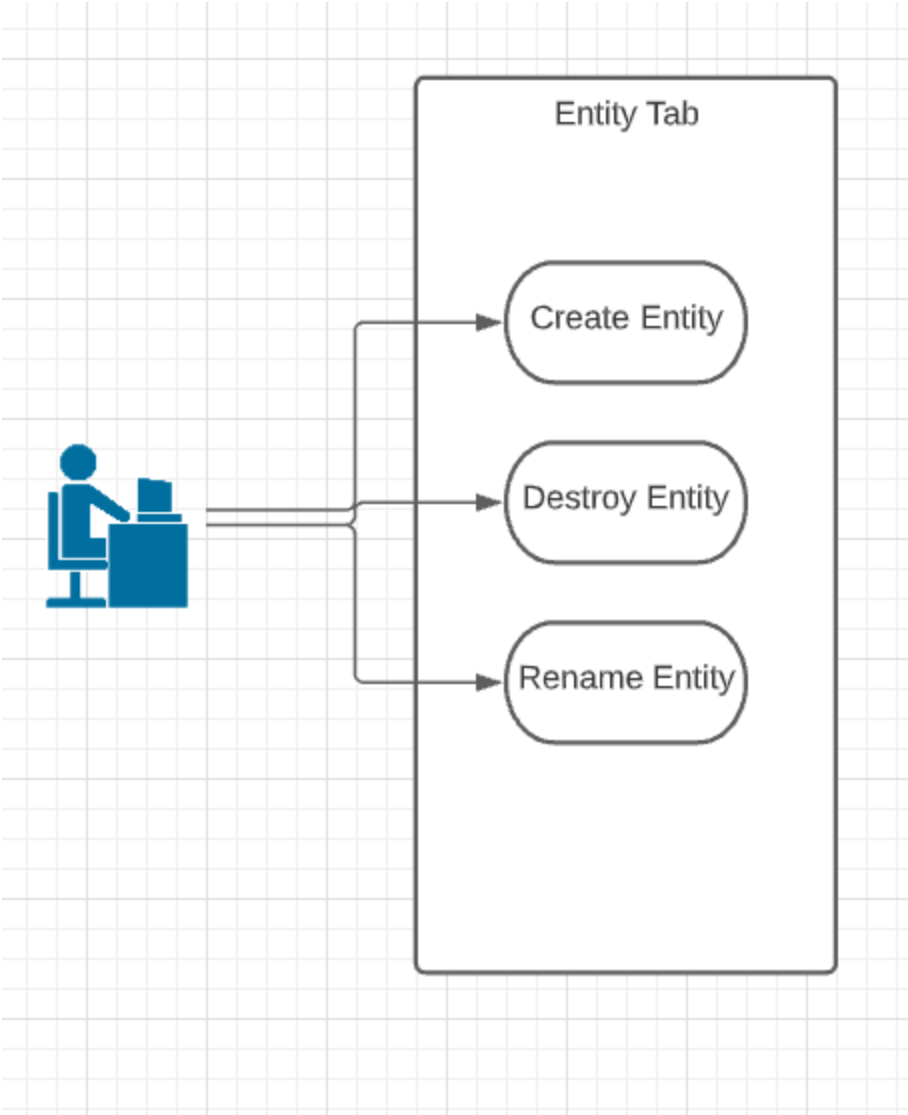


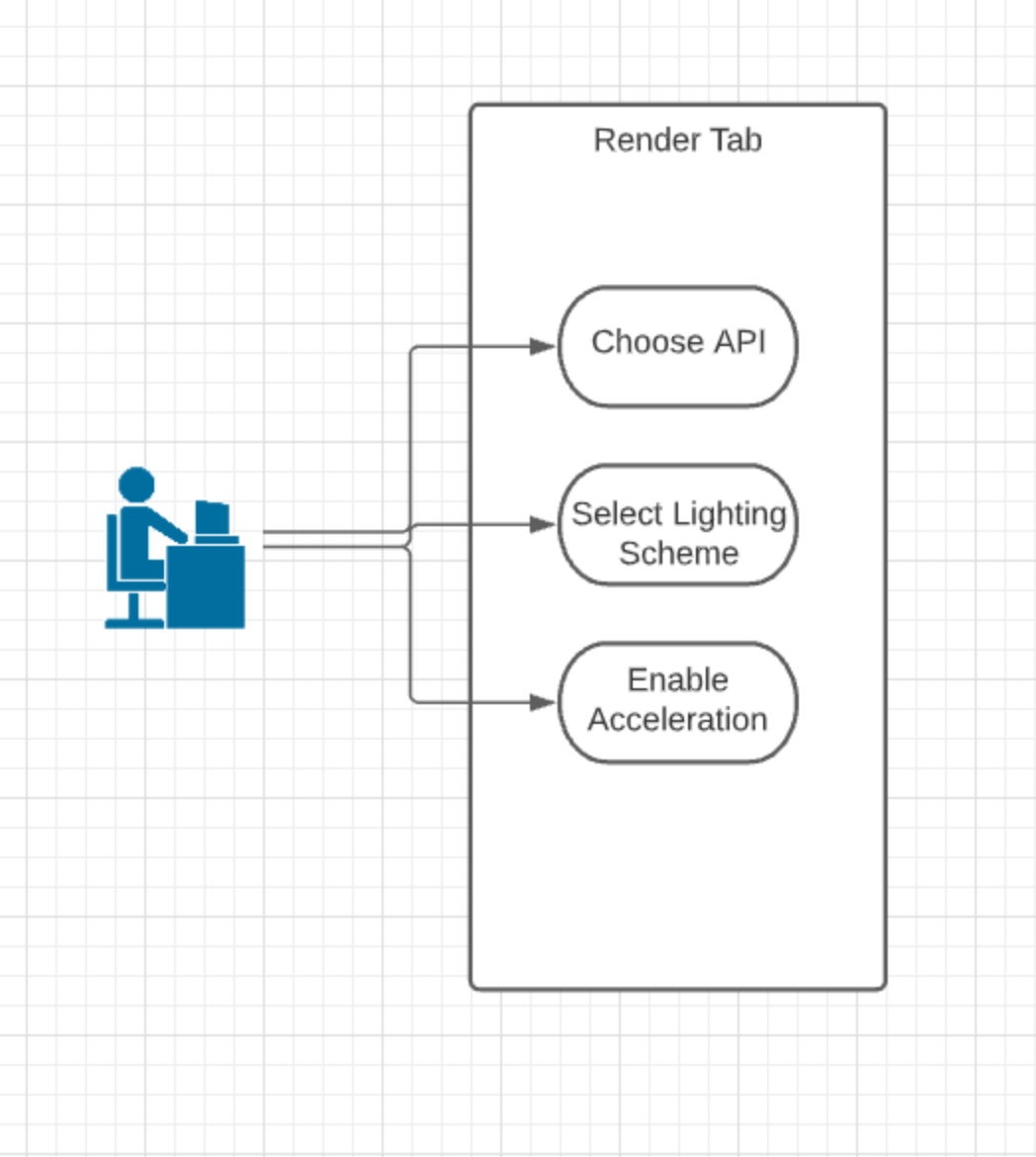
4.1.2 Engine API



4.2 Use Case Diagrams




Entity Tab
Create Entity
Destroy Entity
Rename Entity


Render Tab
Choose API
Select Lighting Scheme
Enable Acceleration

5. Configuration Specification

5.1 Compiler Installation

MinGW installation by itself doesn't suffice. We need 64-bit instructions to allow for acceleration and better optimization(especially in regards to memory fitting.) MinGW 64-bit is obscure in terms of obtaining, so a MinGW 64-bit installation link will be included in the project. We also need to set our path environment variables to include the binaries of this MinGW 64-bit package.

5.2 Build Configuration

The project build system is very intricate. It requires CMAKE Version 3.8 or above. It also requires python version 3.8.10 and up. We also require that your MinGW version of "mingw32-make" be renamed to "make" so that our scripts can call make. This requires a renaming of that file in the binary of MinGW.

6. Roles and Responsibilities

Development Team: Luke Roche

Product Owner: Luke Roche

Scrum Master: Luke Roche

Everything: Luke Roche

7. Glossary

Term	Description
Scene	A scene is a collection of meshes (models, cubes, prisms) in a 3D world that includes things such as lighting.
Lighting Scheme	The algorithm used for lighting a scene
Build System	A system for building the project. This is the system that makes sure a runnable exe is produced from the project.
MinGW	A native windows port of the GNU compiler collection (GCC)
CMAKE	A build tool that facilitates the build system/