**A group of white letters and a circle with a black background

Description automatically generated**

**Group Self-Assessment**

AAA(A) Group

Zack Cornfield, Jacob Mondi, Milad Rakhshbahar, Thomas Stiles

**Contents**

[**Testing Details** 3](#_Toc163912275)

[**Suggestions For Improvement** 4](#_Toc163912276)

[**Reuse of your Software** 5](#_Toc163912277)

# **Testing Details**

We did unit testing for our programs to check if the program runs as it was intended. We tested complex parts of the program as we were doing it. This allowed us to have less of a large issue as we continue to develop on our project.

Large parts of the engine such as physics, meshloader, meshrenderer, ECS, and the lua scripts needed to be tested thoroughly to make sure that it was reusable and did not cause issues. We implemented a testing as we go approach where every time we added a large chunk of code we would test it to the best of our abilities to make sure it was made as intended.

We had to do performance testing for meshloader, physics and ecs as they need to be fast and efficient. We already knew that physics would be efficient as it was based off the previous engine’s physics

# **Suggestions For Improvement**

One major improvement for the engine is the utilization of an editor. At a very basic level, an editor would provide an environment to add entities and components. On a more complex level, the editor would provide a scene view, which would show the changes being made in real-time.

# **Reuse of your Software**

All our Libraries and APIs are in a façade (or in other words, they are encapsulated and contained in a single class). Thus, if we wanted to change one of the APIs or Libraries, only one file needs to be changed, instead of filtering through all our code. For example, if we wanted to switch to Vulkan for graphics instead of OpenGL, we would only need to edit code in the Graphic Utility file instead of needing to filter through all the code throughout the engine.

Similarly, our façades can also be pulled from the engine and used in other programs. If we want to use OpenGL in a new engine, we can pull Graphics Utility and use it, since it is not dependent on any other class.

Because the engine utilizes scripting, you will be able to take any of their user-written scripts and use them for a new project in the same engine. For example, if the user wrote a Player Movement LUA script, they could use that same script in a new game (provided it runs on the same game engine).