**IMAT3904 Report Template**

Please write in the boxes below. Expand the boxes as you need to, however this report should not exceed 2 pages (not including test case data)

| Name: | Chaouki Mabrouki | P Number: | 2655779 |
|---|---|---|---|
| Github Username: | Chaouki Mabrouki | Github Repo URI: | https://github.com/IMAT3904/cw23-ChaoukiMab |

**Please summarise the functionality of your game engine (bullet points are fine):**

- **Rendering System**:
- A basic 3D rendering system is implemented.
- 3D data such as cubes and pyramids can be submitted for rendering.
- Basic shaders are used for rendering with vertex and fragment programs.
- Textures are loaded and applied to 3D objects.
- Simple lighting is applied in the fragment shader.

- **User Input**:
- Keyboard input is used for moving objects in the scene (e.g., position adjustment with WASD keys).
- Mouse events are captured and logged.
- Event handling for various user interactions including key presses, mouse movements, and window events.

- **Event System**:
- A comprehensive event handling system to manage different types of events (e.g., window resize, key press, mouse events).
- Event dispatchers and handlers are used to handle and respond to events efficiently.

- **Logging System**:
- A logging system is implemented using spdlog.
- Logs various events and errors to the console.

- **Window Management**:
- Window creation and management using GLFW.
- Handling window properties such as title, size, and Fullscreen mode.

**What testing have you performed and what testing strategy was used? (test cases on next page please)**

The game engine was tested using the Google Test framework, focusing primarily on the event handling system. Unit tests were written to verify the correct functioning of event callbacks for various events, including window close, focus, lost focus, resize, and moved events, as well as key press, key release, key typed, mouse button press, mouse button release, mouse move, and mouse scroll events. The testing strategy aimed to ensure that event callbacks were correctly triggered and handled, and that the event properties were valid and properly passed to the handlers. However, due to the scope constraints, the application of testing was limited beyond event handling. There was no extensive integration or system testing performed, and the testing was mainly focused on individual components of the event system.

**Did you change any code as a result of use a profiler or a GPU debugger, if so how?**

Profiling and debugging efforts included the use of NVIDIA's Nsight GPU debugging tool. This tool was instrumental in identifying and fixing shader compilation and linking issues, as well as optimizing rendering performance. During the debugging process, Nsight helped pinpoint inefficiencies in the shader

programs and provided insights into GPU performance bottlenecks. As a result of using Nsight, several shader-related issues were resolved, leading to more stable and efficient rendering. The use of this tool significantly improved the quality and performance of the rendering pipeline, ensuring that the shaders were correctly compiled and linked, and that the rendering process was optimized for better performance.

**How have you approached your time management for this piece of work?**

The project was approached with a not a well-structured time management strategy because of personal health issues I lost a lot of time however, I was determined and push through the phases to not fail the module on my resit, phases involved implementing core functionalities like rendering, input handling, and event management. A task list was maintained to track progress and prioritize features based on the grading rubric and what I'm allowed to get as maximum in my resit. Challenges included balancing the implementation of features with writing tests and ensuring that the minimum requirements for each relevant grading criterion were met within the given timeframe. Despite these challenges, the project progressed and adjustments to the plan were made as needed.

**What have your learned from whilst building your game engine what would you do differently next time?**

Building the game engine provided valuable insights into various aspects of game development, particularly in using GLFW for window management and OpenGL for rendering. The process of integrating and using Google Test for unit testing in a C++ project was also a significant learning experience. One of the key takeaways was the importance of starting comprehensive testing early in the development process. Future improvements could focus on implementing more advanced rendering techniques and optimizing the rendering pipeline for better performance. Enhancing the logging system to include file logging would improve debugging and traceability. Additionally, improving the documentation and providing a detailed README with usage instructions and controls would make the project more accessible and easier to use.

Appendix:

**TEST CASE DATA**