****

**Faculty of Science and Technology**

**2022/2023**

**Level 6**

**Game Engine Programming**

**Game Engine Architecture – Code and Report**

**1. Overview**

In this project, I seek to produce a game engine and to produce a simple game within the game engine I produce. I plan to use C++ in combination with CMake to create a consistent, standardised codebase for the game engine. I plan to implement the following list of features into my game engine:

* An Entity Component system to define base structure definitions for certain objects and their types.
* A mesh renderer to draw to player and models.
* A collision system to allow the player to interact with other.
* An environmental audio system to play sounds when interacting with objects or to provide information to the player.
* An input management system to control the player and camera.
* A resource management system to load the game’s assets.
* A shadow rendering system

The game I plan to implement within my game engine is fairly simple. I plan to have a camera following a player in the third person, where the player must collect pickups and avoid obstacles within a time limit for a high-score arcade-style game.

**2. Research**

Even though I had a base understanding of an Entity Component system, I decided to do some further research into the topic to try and deepen my understanding of the concept.

Paraphrasing from Austin Morlan and Krishna Rungta; an Entity is an ID of sorts relating to a specific object within a simulated space. The Entity itself has no data or behaviour but rather relies on Components to mange this.

The Component itself is a struct of data that represents a singular behaviour to be performed by its Entity. A Component should be designed to be reusable on multiple Entities, and therefor should be flexible in definition.

Finally, the System is the logic that operates on the components stored in Entities. The system iterates upon a list of stored local entities and acts upon the list of components stored within each individual Entity.

**3. Program design**

For my game engine, as planned I used an Entity Component system to define the base structure definitions for Objects and their Components in my Program. I used a Core structure to define the System managing the Entities and their components. I also used an Entity structure to define an Entity within the game, which aggregates from Core to allow the Core to hold references to its Entities (see Figure 1).

Graphical user interface, application

Description automatically generated

Figure 1: UML Diagram representing the definition of the Core structure and the Entity Structure.

Graphical user interface, application, table, Excel

Description automatically generatedFurther to the relation between Core and Entity, I have a base Component structure which aggregates from Entity. This allows Entity to hold references to its Components. To define more specialised and functional Components, I created structures that inherited from the base Component (see Figure 2), adding all I thought were necessary to achieve the features I envisioned for my game engine.

Figure 2: UML Diagram representing the definition of Component and a non-exhaustive list of inherited Components.

**Graphical user interface, application, table, Excel

Description automatically generated**For the game itself within my game engine, I used a simple approach. I created some extra, more specialised components as I did not believe they would be suited for the engine for a more general usage case. These components included a component defining a Player, a controller for the Camera, a component defining a pickup, a component defining an Obstacle, a component managing the background Music which represented the time-limit for collecting pickups and a component managing the announcement which explains the rules of the game and how to play (See Figure 3).

Figure 3: UML Diagram representing the definition of Component and a non-exhaustive list of inherited Components specific to the game created within the engine.

**4. Analysis and Conclusion**

I believe the Game Engine I have produced is of good quality and is user-friendly, allowing a third party game to be made with ease. My Entity Component System works as intended and implements most of the originally intended features. My code is consistently written, and easily readable. My code is also written to be very safe, with the use of smart pointers and exception handling. In my opinion the game created within my Game Engine demonstrates the basic capability of the Engine well, and incorporates it’s features to a high standard.

Despite these advantages, there are some areas within the project I feel could be improved. Even with the consistency within the code, high-quality structure of the program and it’s ease of readability, I believe I could have commented the code better. As is, I believe the comments are of a good standard as set by Doxygen, however they are not as extensive as they could have been and some sections of code are missing comments entirely.

I also would have liked to implement the shadow rendering system, as described by my original specification. However, I wanted to focus on the functionality and quality of the rest of the program before this, as shadows are more of a quality-of-life feature than an essential one.

Moving forward, I would make an effort to ensure the full program is consistently and thoroughly commented. I would also look to go through more thorough research to allow a deeper understanding of the topic, to allow a higher efficiency in my work to be able to produce a higher-quality end product with more features.

**References**

* Austin Morlan, 2019. *A Simple Entity Component System (ECS) [C++]* [Online]. Available from: <https://austinmorlan.com/posts/entity_component_system/> (Accessed 12th December 2022)
* Guru99 - Krishna Rungta, 2022. *Entity Component System* [Online]. Available from: <https://www.guru99.com/entity-component-system.html> (Accessed 12th December 2022)

**Assets Used**

* <https://sketchfab.com/3d-models/minecraft-grass-block-84938a8f3f8d4a0aa64aaa9c4e4d27d3> - Model and texture for Cube Obstacle
* <https://www.myinstants.com/en/instant/bing-chilling-clean-john-xina-57735/> - Pickup sound
* <https://sketchfab.com/3d-models/lowpoly-milk-91492741c1fe4b5b80c89d8202664cd5#download> – Model and texture for Milk carton pickup
* <https://www.youtube.com/watch?v=hjGZLnja1o8> – Background Music