Technical report for Game Engine Programming

Fin Hall

1. Introduction

The main objective of this project is to design and implement a robust 3D game engine, showing a clear understanding of game engine architecture. This engine can be used as a foundation for a simple game that has key features such as, model rendering, simple collision detection, GUI, and audio integration. The engine is developed in C++ and uses industry practices such as Git for version control, CMake for building and Doxygen for standardised documentation.

2. Project Specification

The game engine aims to be versatile, efficient and support the development of a simple 3D game. The engine is designed to provide core functionalities required for developing a modern game. The key specifications of the engine are:

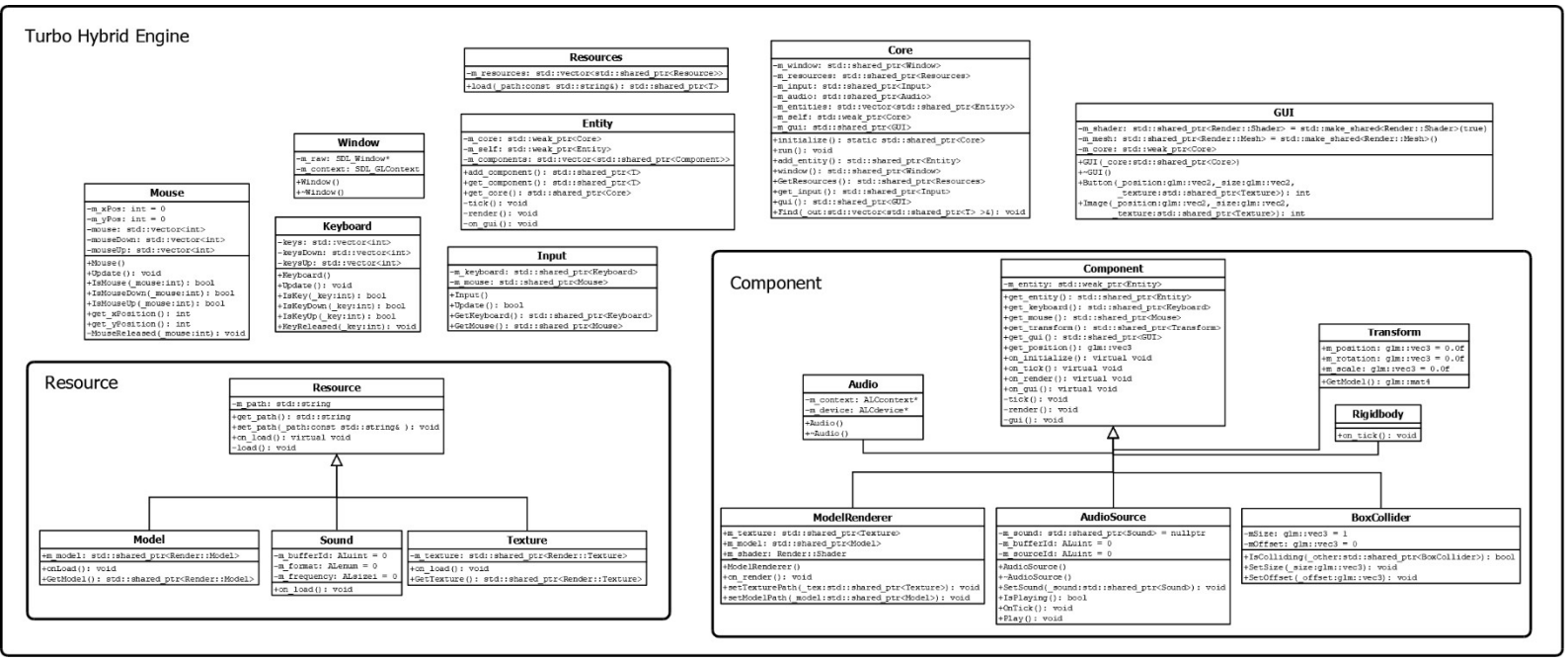
- Model Rendering: capable of rendering the player, obstacles and other games objects, with textures.
- Collision System: simple aabbcc collision detection between multiple objects.
- 2D GUI System: capable of displaying images and adding clickable buttons to the screen.
- Audio System: capable of playing sounds as a result of an action e.g. collisions occur, or buttons are clicked.
- Input Handling: enabling controls from keyboard and mouse including mouse position.
- Resource Management: optimizing the performance and memory usage, by loading assets once.

3. Research and References

The design of my engines component system is similar to unitys MonoBehaviour[i] system. Unitys MonoBehaviour enables modular and flexible development, providing a foundation for understanding how components interact within an entity system. The key similarities are:

- Unitys Start() function is called on the first frame, like my on_initialize() function.
- Unitys Update() function which is called every fame the same as my on_tick() function.
- Unitys OnGUI() function which is called for rendering and handling GUI events is similar to my on_gui() function.

4. Engine Architecture and Design



The architecture of the game engine is centred around an Entity Component System (ECS), a widely recognised system in game development for its modularity and scalability. ECS separates an object's data and behaviour into two distinct components: entities and components. An entity is an object in the game world, while a component is an attribute or behaviour of that entity.[ii]

5. Development Process

The development of the engine followed a structured approach and adhered to industry standards.

Development Process:

- Version control was managed using Git to track changes.
- CMake was utilised for build simplicity and ability to compile across platforms.
- Debugging and testing was performed iteratively using breakpoint and throwing exceptions.

6. Conclusion and Future Work

The development of my game engine successfully showed the application of a modular and scalable architecture using the Entity Component System. The engine integrates essential features including rendering, input handling, and GUI to create a solid platform for game development. Key achievements include:

- component management
- efficient memory handling with smart pointers

- adherence to industry standards like CMake and Git

Future work could focus on enhancing the engine with advanced features such as:

- real-time shadows
- animations
- advanced collision
- ability to support multiple file types

Additionally, expanding the engine's compatibility to support WebAssembly using Emscripten would increase its versatility. These additions would improve the engine, making it better suited for complex game development projects.

---

[i] Unity Technologies (2025). *Unity - Scripting API: MonoBehaviour.OnGUI()*. [online] Unity3d.com. Available at: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/MonoBehaviour.OnGUI.html [Accessed 16 Jan. 2025].

[ii] Journal Of L A T E X Class and Files (2015). Entity Component Systems Usefulness, Literature Review. [online] 14(8). Available at: https://carsonwebster.com/SlugECS/ccwebste_Entity_Component_Systems_Final_Project.pdf [Accessed 16 Jan. 2025].