GEP Report

Introduction

This report goes over the creation of my own game engine, Unbelievable Engine 6. It will cover the key features of my game engine, including resource loading, a basic collision system, an audio system, and an input handling system. My game engine demo game has an interactable block which plays an audio and disappears when touched.

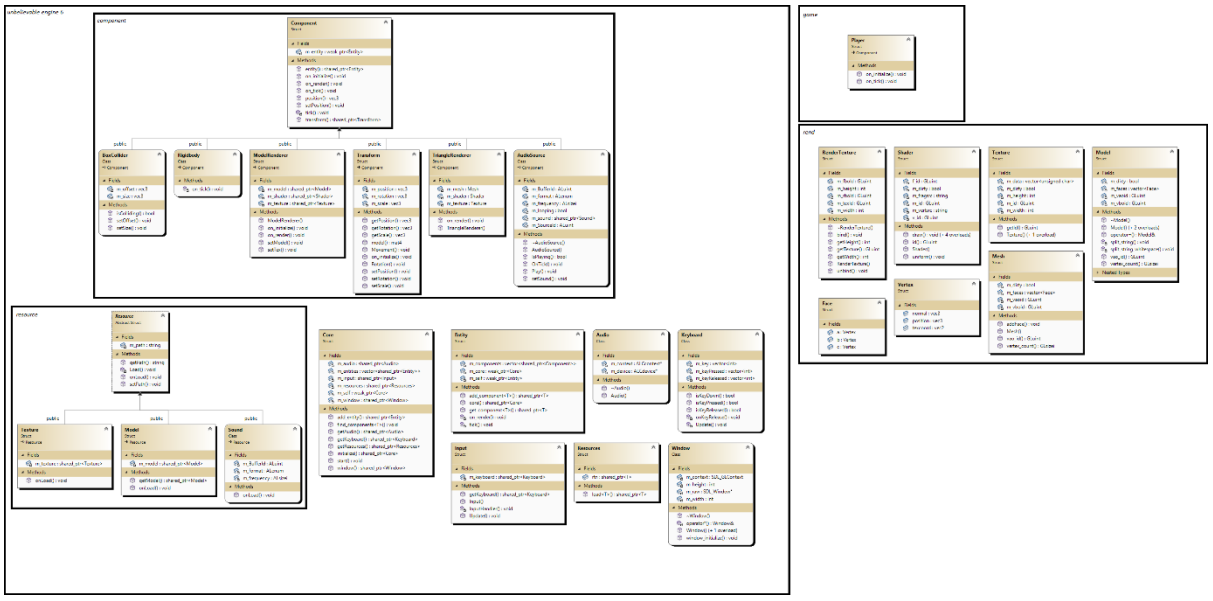
Research

The research I undertook for this project was on different Entity Component Systems, with the most notable example being Unity's ECS. The ECS gives me a number of advantages, as it allows me to give an entity multiple different components that define its behaviour, such as its position, scale and input. It also allowed me to create an efficient and maintainable system, which I could scale for my game to be much larger with many entities.

My ECS allows me to reuse components such as transform and my colliders.

Architecture and Design

My engine was designed to be intuitive to a user of different engines, such as Unity or Unreal Engine. An example of this would be the collision system, which you can just add to a new entity simply, and it is plug and play for the most part. The diagrams provided allows me to clearly communicate the structure and flow of my game engine, showing how different systems in my code interact and how they're organised.



My UML diagrams display my Unbelievable Engine 6, my game and my rend, which shows how modularised my engine is. In my Unbelievable Engine 6 UML, it has multiple smaller groups inside it for Component and its inheritors and Resource and its inheritors.

Development

During development, I took a number of precautionary methods to ensure code standards were kept high. An example of this is my version control, using meaningful commit messages to maintain clean and understandable past versions, which are accessible to me that allows me to go back to them and fix any bugs.

Another example of this is consistent performance testing, in which I tested all major changes (audio, collision etc), that allowed me to make sure that my new changes did critically break my code or degrade the performance of my engine and game.

Analysis and Conclusion

My engine was broken into modular subsystems - like my mesh renderer, collision, audio etc – and that allowed me to create a codebase that is easy for me or a new user to manage and extend, while making it easy to debug any specific problems. This also allowed me to add new features and fix issues without disrupting the rest of the engine.

My codes resource management system also helped me to optimise asset loading, which allowed me to manage resources like textures, meshes and sounds without overloading the user's memory or CPU. It also allowed me to avoid unnecessary loading and unloading, which improves game performance.

In future iterations, I would add cross-platform compatibility, which would allow me to create games on other platforms and allow functionality with controllers and platform specific games that would not be ideal for computers.

I could also add complex features such as networking systems which allow users to play with each other over the internet, allowing VoIP communication too.

References

Freesound.org. (2017). *Freesound - Left Grass/Grassy Footstep 3 by Ali_6868*. [online] Available at: https://freesound.org/people/Ali_6868/sounds/384859/ [Accessed 13/1/2025].

Technologies, U. (2025). *ECS for Unity*. [online] unity.com. Available at: https://unity.com/ecs. [Accessed 13/1/2025]

Terra, J. (2022). *Entity Component System: An Introductory Guide | Simplilearn*. [online] Simplilearn.com. Available at: https://www.simplilearn.com/entity-component-system-introductory-guide-article. [Accessed 14/1/2025]