Engine Report

The task for this assignment was to create a game engine with several features and then making a simple game to demonstrate those features. A game engine in short is “the software framework to build and create video games” (Interesting Engineering, 2016) and are essential in making games quickly and easily as most of the foundation is ready for you. Some engines are built from previously existing games, Unreal, or built to be generic game engines, Unity. Building a good game engine takes time and a lot of effort so when studios, often AAA, the make their own will use it for several games with it before even considering changing.

At the start of the project Git was used to keep a version control to make sure that if anything where to go horrendously wrong that a pervious would still exist to work from. Also, by using Git the project could be worked on from anywhere and since I used CMake to build project it was very portable and easy to pull and push to git as the file size was never too large. Throughout the project it was especially important that I keep my code as consistent as possible to make sure that my code is easy for others to understand and use. Some of the things I tried to keep consistent was with variables, they first letter in the name isn’t capitalized, functions, every word in the name is capitalized, and parameters, there are _ before the word, all following a naming convention..

The engine architecture is structured around the Entity-Component pattern, as you can see in *Figure 1* this pattern includes one “System” or “Core” that holds a reference to every “Entity”, often a generic class, in the scene and calls them in the game loop. With “Entity” in turn keeping a list of “Components”, a feature or behaviour, which have been attached to it. Components can be added are removed before or during runtime, so this architecture is also extremely flexible.

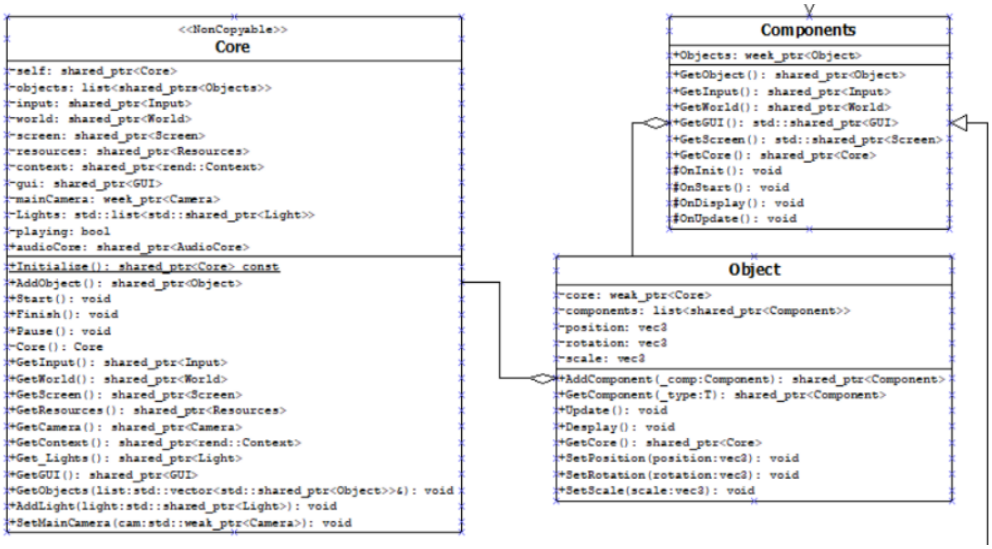


Figure 1

There are several reasons for choosing this pattern over an Inheritance pattern, one of the main reasons being that an architecture based around Inheritance can easily often create large inheritance hierarchies that can be difficult to understand and/or extend whereas with an Entity-Component pattern this issue is avoided all together. Also, components can be added or removed from an Entity easily making this pattern extremely flexible. This type of architecture is extremely popular in modern game engines because of this, Unity being a main example as it is one of the most popular game engines in the market for many indie companies and hobbyists with it claiming that "50% of mobile games are powered by Unity".

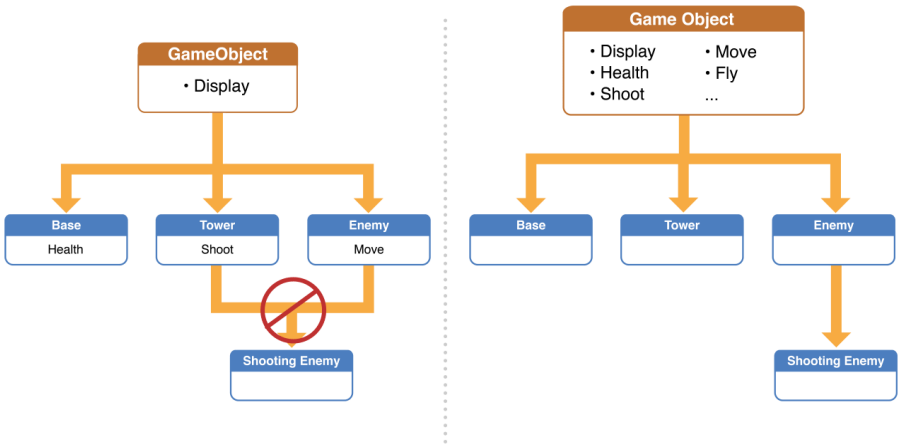


Figure 2 - (Apple, 2018)

Safety is an important part of any engine, the last thing a user wishes to see is their engine crashing on them and not have an explanation on what they may have done wrong. To keep my engine safe for the user by using smart pointers and having exceptions anywhere the code may fail.

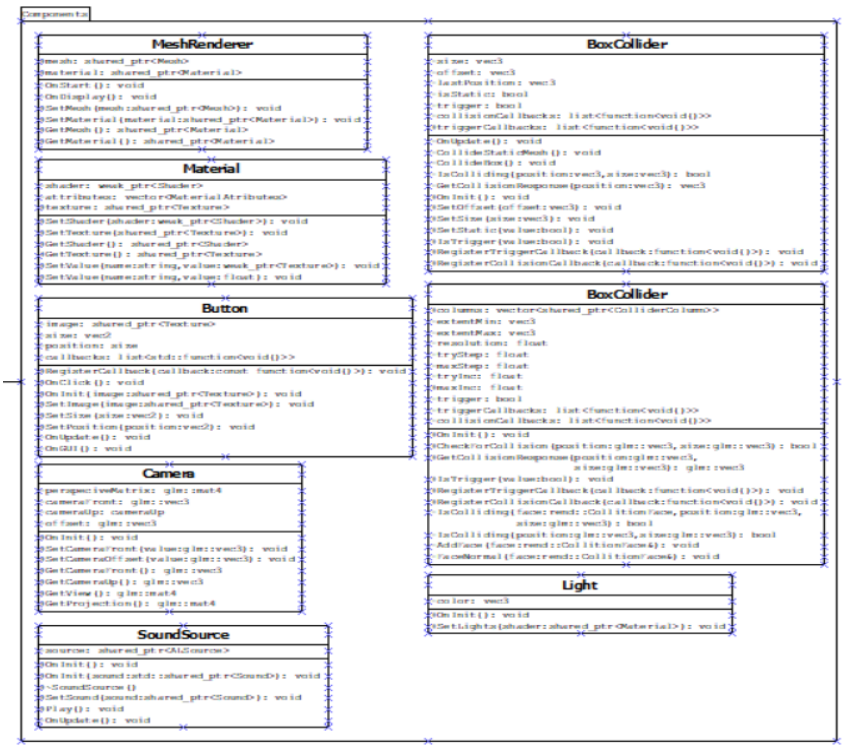
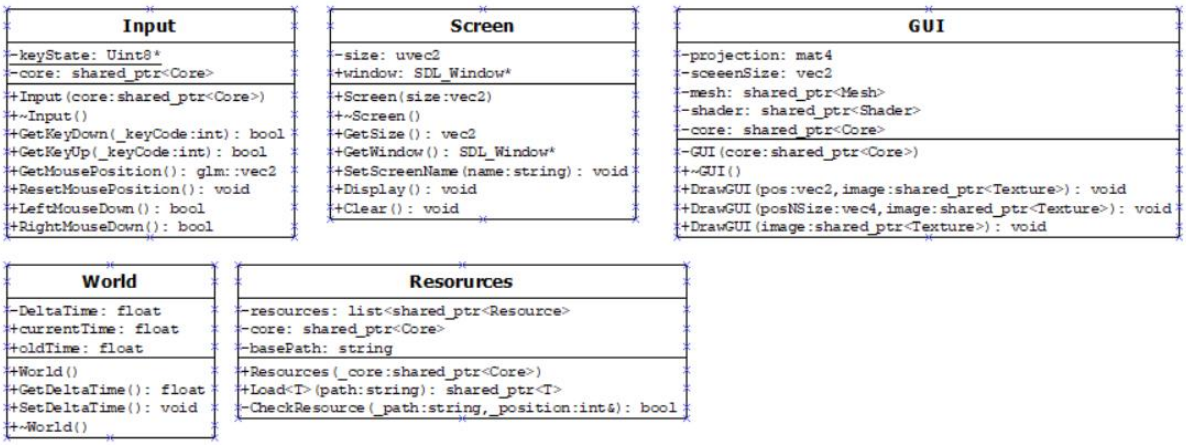


Figure 3

As shown in figure 3, there are several components include in the engine when it was submitted. Each Component has a specific function that changes drastically with from each one, ranging from simple ones like Lights, that are mainly used to pass information to the shader on any lights in the scene and doesn't even need to use the OnUpdate() function, to more complex ones like the Box Collider that requires several variables, like the objects position and a list from "Core" of all the objects in the scene with a Box Collider, and functions to work properly.

The main reason I choose to make these Components was the fact that we did not have a lot of time to create this engine. I choose features that are I consider requirements for a game to function, the ability to load and render an mesh and change how said mesh looks, and only when those where done did I try and make a start some other features. Though you may notice I do not have a Transform component, the reason for this is that since every entity would need one anyways and there would probably be a lot of Components that would use it I decided to cut out the middle man and have the transform functions directly on the object. And since the Component class is meant to be a parent class to build from anyone using the Engine can create their own Components that they can then add to an Entity in the scene. This gives the user more freedom to make any type of game they wish.

Now not everything in a game engine can be a Component. There are several classes that Core controls that are not Entities or Components but are instead used to get for things like drawing GUI to the screen or control APIs or load resources. My core class creates a reference for each one of these classes when it is initialised and then if a component needs one, for example they need load in a texture, they can go up the stack from their reference of the entity they are assigned to core, since all entities have a reference to core when they are created. Now other game engines, like Unity, use static class instead that can just be called without going up that stack but that but I choose not to use this as static functions is often not as flexible and is only able to run a single game instance which is difficult would make things difficult if I choose to make streaming server or multiple instances.



Overall, I am pleased with how my game engine ended up as it fulfils its main purpose of making it easier to make a game. I was able to make my small demo with a range of features much quicker than if I built from the ground up. Some features like the Mesh Renderer took some time to implement but made it very quick to simply call when needed, same for the Box Collider.

Though there are some area's that I did wish to spend more time on. For example, there are 2 classes in the engine that are not complete, Fonts and Material Attribute, that I choose to leave in the engine to show some components I wish to implement in the future. With Fonts I did try to have it working before the submission time, but it proved more complex than I expected ad was

ultimately cut to work on other things. Also in the future I plan to do some more complex programming for my shaders even though I was able to program and load in a shader and even had one that would have the objects interact with a single light in the scene I was only able to get it working with only one light.

In the end my engine has a solid base that I can and will build upon in the future for other projects. I have learned a great deal about game architecture and why some things are done in some ways, like with having components instead of inheritance, and also my understanding of other engines have improved, with me now understanding why some engines are built the way they are.

References

Apple (2018). Problems Evolving An Inheritance-Based Design. [image] Available at: https://developer.apple.com/library/archive/documentation/General/Conceptual/GameplayKit_Guide/EntityComponent.html [Accessed 5 Jan. 2020].

Gregory, J. (2015). Game Engine Architecture, Second Edition, 2nd Edition. 2nd ed. eBook: CRC Press.

Interesting Engineering (2016). How Do Game Engines Work?. [online] Interestingengineering.com. Available at: https://interestingengineering.com/how-game-engines-work [Accessed 6 Jan. 2020].

Unity. (2020). Unity Public Relations Fact Page. [online] Available at: https://unity3d.com/public-relations [Accessed 11 Jan. 2020].

Wilson, K. (2002). Game Object Structure: Inheritance vs. Aggregation. [online] Gamearchitect.net. Available at: http://gamearchitect.net/Articles/GameObjects1.html [Accessed 5 Jan. 2020].