**Game Engine Programming Assignment Report**

Introduction

This report will cover the implementation of this project, in which the aim was to design and create a simple and functioning game engine, along with a tech demo to demonstrate some of these features. The project began with gathering information and inspiration from existing game engines (e.g. Unity), and taking their pros and cons into consideration before deciding on how they could be improved. Before starting the implementation of the engine, there had to be a clear design structure that permitted any expansion once a simple structure was created. The engine needed to begin with a basic Component Entity System (CES), that worked alongside a ‘core’ that was used to control the engine, and withhold the standard game loop. The CES consisted of a system that held game objects and components that can interact with one another. This gave the ability to inherit from the basic component class to create necessary components for the game objects, which allowed for control over certain features when used outside of the engine library and within a game created using the engine. Extra required features, such as a class to hold the screen’s data, were also specified to allow the core to pass this information to the components. With the endless possible expansion of the engine, time limited the implementation of certain features that could have been included, but a functioning engine and tech demo was still able to be developed to the initial specification.

Research and Development

During the development of the engine, research was required when designing certain features and implementing them. Unity was used as reference point when designing a lot of the game engine’s features; an example of this is the idea of using game objects which ‘are the fundamental objects’ that are used by the engine and can ‘represent characters, props and scenery’ (Unity 2018). This also gave aid when designing the functionality of the basic component. Unity’s components have the ability to communicate with one another, allowing for advanced functionality and efficiency. They can also be easily interacted with by the user on the game development side, so all of these abilities were deemed necessary to include in the engine design. Issues arose during the evolution of some the features; one example of this was the creation of the game loop and calculating time. In order to fix the framerate of the game, the game loop must be controlled by time and to only be repeated after a particular amount of time had passed. The amount of time passed between each frame must therefore be calculated (deltaTime). This is calculated by taking the number of ticks (milliseconds) that have passed (which can be gathered using SDL’s getTicks() function), and subtracting the number of ticks that had passed by the previous frame. This gives you the difference in time between the last and current frame, which you can then use to delay the program at the end of the loop and fix the framerate of the game. When first implemented, the deltaTime was only calculated when required, rather than during every frame. Once it was calculated during every iteration of the game loop, objects’ movements were much smoother.

High Level Description – 1200 Words

Analysis - 250 Words

References

https://docs.unity3d.com/Manual/class-GameObject.html

https://docs.unity3d.com/ScriptReference/GameObject.html

https://docs.unity3d.com/ScriptReference/Component.html