Submitted by:

Andrew MacMillan - 100992185, and Cody Webb - 101026276

Part 2: Game Engine

Coding Styles:

Variables –

Variables are to be in camel case and be descriptive.

Example: **private int thisInt = 0;**

Functions –

Functions are to be in Pascal case.

Example: **SomeFunction(){}**

Arguments –

All arguments are to be in lowercase.

Example: **SomeFunction(int argue){}**

Class –

Class names are to be descriptive and be in Pascal case. Follows previously described naming conventions.

Example: **EnemyClass{}**

Commenting –

Detailed and descriptive comments, constructive in nature or informative. Frequently remove unnecessary comments as progress continues.

Source Control –

Descriptive commit messages. If unsure of branch/commit status, contact repo manager.

Project Hierarchy:

All vital classes and components to the game engine should be kept in the source folder, to ensure engine security when exposing the engine to the public.

Minimum PC Requirements:

Minimum:
4GB RAM
Dual Core processor, with at least 1.5GHz per core.
GTX 800 Series or better

Recommended:
8GB RAM
Quad Core Processor i5 Series or better
GTX 900 Series or better

Component Architecture Description:

The Game Engine class will be found in every other class or component in the engine, having a hand in connecting Input with Physics and our Actors. The Actor class however, will have several components that get information from the class itself. Components like Rigidbody, for physics implementation, a Collider, Graphics in the form of UI and Sprites, and Audio devices built into each actor in the scene.

Main Engine Loop Description:

The program created using our engine will initialize first, before starting its processes and moving into the main loop. The main loop consists of accepting input from the user and updating the scene with that knowledge, before rendering what is in the scene at the time (be it a menu or the game itself) and waiting for user input once again. This process will repeat until the 'exit' function is selected which will cancel all the processes and close the application.

Part 3: Questions

1. Currently, our game engine lacks certain key components. Perhaps most importantly, is our game engine's lack of 3D capability. Without this key component, our engine is severely limited to the type game it is able to produce. Additionally, without capable AI templates our engine is rather 'bare-bones' as it were; leaving much to be desired.

2. Our game engine comes equipped to handle a rather basic 2D platformer, but it has potential to be extended upon in order to develop larger, more complex games. In order to do this, we would need to add four major elements unavailable in its current iteration; 3D physics, network connectivity, graphic user interface implementation and artificial intelligence. Any game worth its three dimensional salt requires both an advanced physics engine and artificial intelligence in order to create a world that feels real to the player. Network connectivity is also required to allow for our engine to expand into potential mass player bases and online communities as through other engines. Some form of a graphics user interface would allow for ease of access to

other developers that would be interested in looking to our engine and also allow ourselves to have more visible control over the aspects of our products.

Extending our game to the third dimension would require an overhauled graphical system on both the rendering side, and the art side. We would have to re-optimize our rendering pipeline to enable the engine to handle, and render 3D models. We would then need to add 3D lighting. Additionally we would be adding another component to our physics engine, which would extend upon the 2D physics aspect by adding the third dimension. Adding artificial intelligence into our engine would allow for developers using this engine, including ourselves, to work on projects that involved more robust enemies or obstacles than just blocks or spikes in your path. While network connectivity will allow for games built on this engine to be multiplayer or have online leaderboards giving access to a wider range of genre options.

One potential feature to add to the game engine would be a GUI. Adding a GUI to the engine would open up our product to a much larger potential clientbase as it would create a much more user-friendly environment. In addition to expanding upon our userbase, as GUI would increase productivity as rapidly prototyping products and ideas would be made easier by the fact. If done correctly, this feature could turn into a potential revenue stream as with other popular engines such as Unity, or Unreal we could offer either enterprise grade licensing, or implement royalties if games made with our engine surpass certain revenue milestones.