Report for IMAT3904 – Game Architecture – P16195518

**Introduction**

In this assignment the objectives were to: create and implement a fully functioning game engine that includes a way to import assets, scenes and models, either via Maya or another modelling software. Include design patterns such as components and interfaces that could be used to aid in the creation of a game from this engine. Use JSON, XML or Text files to aid in the management of the game data, scenes and game flow. Finally create a basic demonstration of what the game engine can do when used.

**Text Files & Level Creation.**

Within the Game.h and Game.cpp files the user is able to do many things including add or remove objects in the scene, translate and rotate these objects where they want them to be, change the colour of the background and set up correctly would be able to change the colour of any objects added to the screen. To create simple cubes we first create a cube game object in game.h, then the cube is moved using the transform component, given a colour and drawn to the screen using the drawcube function.

To create simple levels, the loadLevel class at the bottom of the screen reads and loads text files line by line in order to display on the screen what the level is trying to achieve. In the text file example in the engine already it is loading a simple cube level by using a for loop to read the position, orientation and scale of each cube in the file. This would allow for levels to be created and input commands to be set in place to easily move from one level/text file to another. To create a working game. This way is a lot more efficient than separately inputting each object into the level and trying to scale, translate and position it correctly in the window.

To switch levels there is two files called SceneStateComp.h and SceneStateComp.cpp which, if implemented correctly, would allow for a simple input command to be created to allow for the user to switch levels in the game as the press of a button. This would allow for as many levels as the user wanted to include with as many buttons to change the levels as needed.

**Input Handling.**

Within InputHandler.h there is the starting point to add input commands that the user can change on the fly in order for the player to be able to interact with the game. This could mean being able to move objects using key presses, mouse movements etc. These commands can be implemented separately to different objects within the game as to add more complexity to games. To be able to implement new input commands, classes would have to be created for each command and referenced within an execute class to determine exactly what the command is going to do and to what object on the screen.

**JSON Files.**

Implementing JSON files would be similar to text files except the code would read the scale, position and orientation of the models, using obj files, within the JSON file and creates it as a game object rather than simple shapes line by line. These game objects get created onto the screen and pushed back as either environmental objects or player objects. JSON files are a lot more appealing to the user for game creation and this is something that should be implemented. To import a model from Maya for example the user would have to open their model, execute the Python code as a JSON file and add the JSON file into the correct section of the code to allow for the model to be added to the game.

**Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Before** | **After** | **Works Correctly?** |
| **Adding new cubes to the game world** |  |  | **YES** |
| **Changing the position of the cubes** |  |  | **YES** |
| **Change the background colour** |  |  | **YES** |