**KF6017 Software Architecture for Games**

**W16015479**

**Advantages and Limitations to the Game Engine**

In this module we were tasked with creating a high-quality game engine capable of creating 2D game. I have created this using an Object Manager which allows for easy maintenance of the code relating to game objects such as collision allowing for easy inheritance for Game objects.  By using an object manager in association with different subclasses, I have developed code that will help for easy debugging, providing an easy way to detect errors. The ObjectManager will be also be able to be incorporated into different games in the engine with little adaptions needing to be made. This can be a negative however as this leads to high cohesion and low decoupling as the ObjectManager completes all function relating to ObjectList such as adding, checking collisions and deleting. This could be solved by incorporating a component system in order to lower the amount of cohesion by having separate components to process collision, delete objects etc. Another advantage of my game engine is the use of a Level Manager, using a level manager allows for a more organised architecture to the engine. By using a level manager, it allows for more added functionality for the engine such as by using a messaging system it can send messages between classes in order to carry out certain functions, with certain functions including a lives and score system or a game over game state. Like the ObjectManager this can be adapted to other games in the engine with little adaptions needing to be made. This provides a huge amount of possibilities to the engine for programmers, including in my own 2D Game. One more advantage of my game engine is the creation of the GameObject class. In the game object class, it creates virtual functions that each game object can inherit, allowing for polymorphism to be used effectively by the game objects to inherit the methods need for collision, updating and rendering. One final advantage of my game engine is the use of pointers. By using pointers in relation to classes such as a pointer to Bullet in Spaceship allows for easy access of the Bullet classes properties, and is an increasing good practice when programming including for future use of the engine in game creation. In using pointers it means any class wanted to process an action with another class such as a bullet colliding with an rock, by getting a pointer for Bullet in rock allows for it to be easily accessed and added to the ObjectList. There are however limitations to the game engine I have created.

The limitations it has in relation to creating a 2D arcade game is that by using an ObjectManager to carry out each function for new objects such as checking all collisions means that not only is there high cohesion and low decoupling but by using pointers in the architecture it means that the ObjectManager is responsible for deleting objects, meaning if even the smallest error occurs in the engine it can provide the engine with memory leaks which impacts the quality of the game engine, this could be improved upon by using smart pointers to navigate around the issue of memory leaks.  Another limitation to my game engine is that LevelManager depends highly on the Spaceship and Rock level states. Without these level states the game would cease to function, meaning that the LevelManager could not implemented in another game being created in the engine without sufficient coding changes needing to be made to the engine. Although the LevelManager can still inherit methods from GameObject it can't interact with the objects itself. This means that the Game code becomes decoupled from the LevelManager which could lead to potential problems if the LevelManager is removed from the engine which is a limitation to any programmer using the engine. This also means that objects cannot be initiated without the LevelManager class, this leads to low cohesion in the LevelManager class which can be saw as a drawback to the engine for programmers looking to create an game that initiates objects without wanting to use a LevelManager.

Overall I believe the game engine I have developed an engine capable of creating a high level 2D game, with code that a programmer would be able to implement in their own game created in the engine however there would be some certain limitations that need to be considered. For future Game Engine creation, I would aim to eradicate these limitations.

**References:**

Hoon, M. and Kehoe, M., 2003, October. *‘Enhancing architectural communication with gaming engines.’* In Proceedings of the 2003 Annual Conference of the Association for Computer Aided Design In Architecture, Indianapolis, IN, USA (pp. 349-355).

Ohst, D., Welle, M. and Kelter, U., 2003, September. *‘ Differences between versions of UML diagrams.’* In *ACM SIGSOFT Software Engineering Notes* (Vol. 28, No. 5, pp. 227-236). ACM.