Games Architecture and Concurrency Report

Davide Contini

ID:201940289

# 1 .0 Design

The core of this game engine is composed by two classes which are Game and Scene. Game is one of the three Singletons classes used within the entire project. The use of this design pattern was intentionally used sparely since it can create confusion and makes the code harder to maintain. Since she Game class is also used to as starting point to broadcast messages across the engine the Singleton was best suited, SettingsMananger and ThreadManager also used this design pattern since they also need to be accessible from everywhere across the engine. The other half of the core of this engine is the Scene class. This class has the role of maintaining the reference of all the game objects and systems in the game, it can interact with them, create new ones, destroy or pass messages to them. If desired, it is possible to attach a window to our Game class, this allows the user to either create a renderer or even just pass keyboard and mouse events to the engine. In the current state of the game engine only supports DirectX 11 which is fully implemented by inheriting the Window, Renderer and VBO classes and implementing behaviours specific to that API within those classes, in future, if a renderer which uses a different Graphics API wants to be added, is possible to do so adding platform specific implementation in new child classes.

As mentioned before the scene has the role of storing all the active systems and game objects. An arbitrary number of both can be added to the scene. If no systems are added, then the game will not have any behaviour. Systems are completely independent from each other’s with the exception of physics and collision