This document comprises the design for the second prototype of my project. With a fundamental aspect of the main product completed, my next task is to incorporate the next fundamental of the product – player interaction and movement. This consists of implementing a first-person perspective character to control and allowing it to spawn within the procedural generation. This prototype will also consist of a few tweaks and modifications on features from the first prototype.

**User Interface Design – Visual Tweaks**  
With the addition of the player and player movement & spawning, having the prompt to provide information on how to control the player to users will be helpful to them. This will most probably be positioned on the right, away from the other elements of the main menu. Part of the menu I wanted to incorporate is key-triggers for certain actions. Including keyboard commands to start and quit for example will bring about greater accessibility to the users and overall provide a better experience using the prototype.  
The user interface for the map generation will be restarted from scratch as the purpose for it has changed with this new prototype. In prototype one, it was just a means of exiting that scene, but with a first-person camera implemented, that canvas will now hold a crosshair – this crosshair will act as the central point of view for the player when playing. Exiting the scene as such will now be changed to simply the escape key.

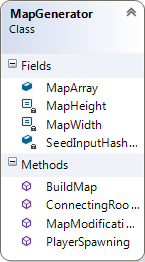
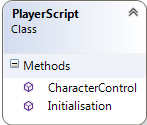
**Identifying Classes – How will it be Built?**

Class Additions

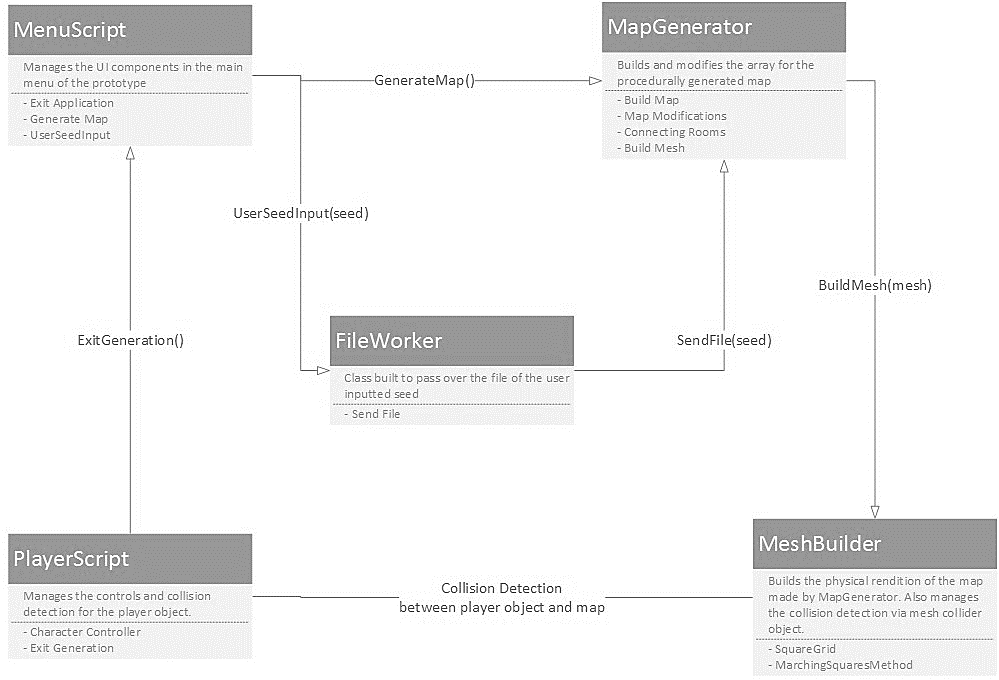
* **Player Script:** This new class will be added into this prototype as the controlling mechanics for the player object. This will be embedded into a player object within the Unity environment. This will also control the camera too as the player controls the object in a first-person perspective.
  + Method – Initialisation: This is one of the methods that is available through the Unity engine, anything written into this method is perform the moment the program is started. Here will be processes such as cursor invisible resetting any values.
  + Method – Character Control: This method will contain the bulk of the class, including the use of the character controller within Unity to provide movement to the player. This will also include the camera control – having the camera connected to the input of the mouse.

Class Changes/Tweaks

* **Map Generator:** The Map Generator script will have a new method implemented to be done once the map has been completed.
  + Method – Player Spawning: This method manages the spawning of the player into the procedurally generated map. The method will search through the map array to find a point where the map is a floor and therefore safe to spawn and well then spawn the player in that location.
* **Camera:** The Camera script is no longer required. When it was initially implemented, it was meant as a method to gain a visual on the map since it had no player object to roam around it; with the addition of the player object and its own camera for a first-person perspective, this class and its associated camera object are not needed and will therefore be removed all together.

New Class Diagrams

Class Communication  
To show class communication, I have created a communications diagram to show the interactions between the scripts. With the additional class and having classes triggering multiple others, it was easier to present it in this way. This doesn’t show how the scenes in Unity interact.



**Pseudo Code Algorithms**

