*I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any other award.*

*Signature:* Ruari Joshua McGhee

## Ruari McGhee – S1432402

## Computer Games (Software Development)

## <HTTPS://GITHUB.COM/RUARI026/WGE-2019-COURSEWORK>

Working with Game Engines Coursework: Report

Contents

[Ruari McGhee – S1432402 0](file:///C:\Users\ruari\Documents\GitHub\WGE-2019-Coursework\WGE%20Coursework\Assets\Report\WGE%20Report.docx#_Toc7139751)

[Computer Games (Software Development) 0](file:///C:\Users\ruari\Documents\GitHub\WGE-2019-Coursework\WGE%20Coursework\Assets\Report\WGE%20Report.docx#_Toc7139752)

[HTTPS://GITHUB.COM/RUARI026/WGE-2019-COURSEWORK 0](file:///C:\Users\ruari\Documents\GitHub\WGE-2019-Coursework\WGE%20Coursework\Assets\Report\WGE%20Report.docx#_Toc7139753)

[Scene 1 2](#_Toc7139754)

[Events 2](#_Toc7139755)

[Communication Between Scripts 2](#_Toc7139756)

[Software Design Patterns 2](#_Toc7139757)

[Scene 2 3](#_Toc7139758)

[Part I 3](#_Toc7139759)

[Camera Controller 3](#_Toc7139760)

[Player Controller 3](#_Toc7139761)

[Part II 3](#_Toc7139762)

[Dialogue File Structure 3](#_Toc7139763)

[Dialogue Editor Tutorial 3](#_Toc7139764)

# Scene 1

*- Give an overview of the scripts used to implement this scene. This should include explanation for important methods and variables used. You should make special note of how events are used in the scene and the communication between scripts. Describe what software design patterns have been used, and how these are implemented including the script or scripts involved. You should also discuss any sorting algorithms used and their operation.*

## Events

## Communication Between Scripts

## Software Design Patterns

# Scene 2

## Part I

### Camera Controller

Inspiration for the camera controller taken from Fez and Super Mario World

The camera movement is split into two movement forms for each axis:

* When the character hits a platform the camera lerps to the player’s Y position
* The player has a small zone in which they can move freely before the camera follows their X position

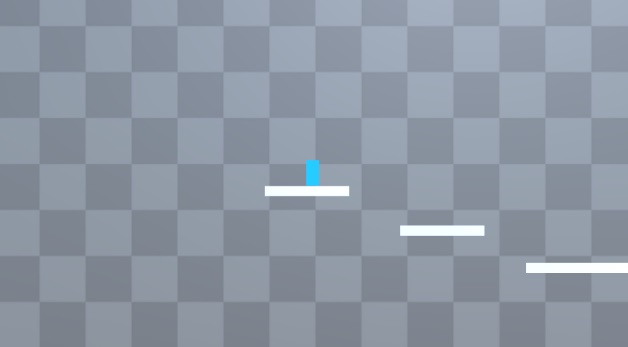
During normal play the camera is zoomed out to show a large amount of the environment, when the player enters dialogue with the NPC the camera zooms in to show both actors clearly

Figure 2 - Camera Dialogue View

Figure 1 - Camera Regular View

### Player Controller

The Player Controller follows the state design pattern

When developing the camera controller this made it easy to assign different events based on what state the player controller was in:

* When the player was walking the camera controller would get the player’s y position and move to that position
* When the player was jumping the camera controller would build up its camera shake timer

This also allowed other objects to remove control from the player without having to disable the scripts from the gameobject itself since the player controller had a state called MovementState.DISABLED built into it. This technique was used by the dialogue system when the player entered dialogue with the NPC.

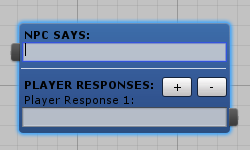
## Part II

### Dialogue File Structure

### Dialogue Editor Tutorial

The Dialogue Editor uses a node-based system, it has been deliberately designed like unity’s own animator controller editor

##### Nodes



##### Other Elements

# References

Creating Editor Windows in Unity - <http://gram.gs/gramlog/creating-editor-windows-in-unity/>

Creating a Node Based Editor in Unity - <http://gram.gs/gramlog/creating-node-based-editor-unity/>

The Theory and Practice of Cameras in Side-Scrollers - <http://www.gamasutra.com/blogs/ItayKeren/20150511/243083/Scroll_Back_The_Theory_and_Practice_of_Cameras_in_SideScrollers.php>