# National College of Ireland

BSHCSD4

Academic Year 2023/2024

Mark Cummins

x20400634

x20400645@student.ncirl.ie

# Epoch Explorer

# Technical Report

# Contents

# Table of Figures

# Executive Summary

This technical report aims to describe the steps taken to develop the application 'Epoch Explorer,' which is an interactive learning tool that seeks to immerse users in environments that are recreations of towns from the past and allow them to learn about the history of different era's. This report provides an insight into the technology used to develop the application; Unreal Engine 5, Blender, Polycam and a variety of other technologies. It provides the reader with an insight into the architecture of the application and the various mechanics of it. An in-depth description of both the functional and non-functional requirements of the application is also provided.

The report will explain in detail the steps taken to create the functionality of the Epoch Explorer application such as the creation of the environment, the ability for a user to learn about objects in the world, to view objects in an in-depth manner, navigate the environment with a map and save and load their progress in their exploratory journey. Each of these facets of the application will be explained with accompanying screenshots of the blueprint code used to develop them and the design process involved.

# 1.0 Introduction

## 1.1. Background

I undertook this project as I have an avid interest in both history and video games. While my project is not necessarily a video game and is more-so aimed at being an interactive learning tool, it is still an interactive media experience which I feel falls under the wider umbrella of video games. I have always had passion for history and would like to share that passion with others and hopefully get them interested in history through a more interactive and engaging medium than traditional learning methods. A video game is a unique way to harness the innate human desire to explore while coupling it with learning about different time periods.

Through this project, I hope to give users a deeper understanding of just how much the towns they inhabit grow and change over time and what our towns may have looked like to our ancestors. I feel that traditional teaching methods for history can be unengaging and tend to cause a disconnect between us in the present day and the people of the past. I hope that with this project and enabling a user to see the town of Dunboyne as our ancestors did, it may change a user's perspective on history and give them a deeper appreciation for it.

## 1.2. Aims

This application aims to achieve an immersive and educational interactive environment for a user to explore and learn about the town of Dunboyne circa 1909. The project aims to have an intuitive user interface that allows users to experience the historical recreation without much prior knowledge of video games. The control scheme and menus aim to be simple, yet aesthetically pleasing and easy to interact with.

The in-engine environment aims to be as historically authentic as possible, with great attention to detail taken in terms of recreating the town as closely as is possible with the current academic and archival resources available.

It aims to allow a user to learn about the various sights of the town and gain a deeper understanding into the lives of our ancestors who lived around the turn of the 20$^{th}$ century by allowing them to see the town in a similar state that our ancestors would have experienced it.

The project aims to be as widely accessible as possible, therefore it will be optimized to run on the widest variety of hardware possible through the ability to tweak settings, optimisation, and testing.

## 1.3. Technology

A variety of technologies were utilised in the creation of this project, the main ones being Unreal Engine 5, Blender 4.0, Polycam, Unreal Mapbox Bridge and Geohive.

Unreal Engine 5 is a game engine that allows a user to create games, simulations, and other interactive experiences [1]. It serves as the game engine for the application and was used to develop the environment and functionality of the application. It allowed me to populate the project by importing assets from Blender and Polycam, gave me the tools required to develop the applications logic through its C++ 'blueprints' system.

Blender is a 3-D modelling tool that was invaluable in creation the of assets required for the project [2]. It allowed me to tweak the assets created using photogrammetry with Polycam. It will also allow me to create entire assets myself, creating their models them and texturing them so they can be imported into Unreal Engine.

Polycam is a 3-D modelling tool that allows a user to convert photographs of real-world objects into 3-D models [3]. It proved invaluable in converting the buildings captured using drone photogrammetry into usable 3-D models.

Unreal Mapbox Bridge is a plugin developed by Daniel Elebash that allows a user to create an Unreal Engine landscape from satellite height-map data [4]. This plugin was crucial when it came to being able to create a historically authentic and to-scale recreation of the town of Dunboyne.

Finally, Geohive is a tool that allows users to access historical ordnance survey maps online [5]. This tool was a great resource when it came to referencing exactly how the town was laid out in 1909.

## 1.4. Structure

This document contains four major headings; the Introduction, the System, the Conclusion, and Further Development and Research. The System section contains information pertaining to the functional, data, usability, and environmental requirements of the application. It contains an overview of the design and architecture of the application and goes into detail regarding the implementation of the various features of the application. Finally, it contains sections dedicated to highlighting the graphical user interfaces of the application and the testing conducted on the application.

# 2.0 System

## 2.1. Requirements

### 2.1.1. Functional Requirements

#### 2.1.1.1. Use Case Diagram


*Figure 1 Epoch Explorer Use Case Diagram*

#### 2.1.1.2. Requirement 1: Start Application

| Use Case Name | Start Application. | Use Case ID | U1 | Priority | High |
|---|---|---|---|---|---|
| Scope | Allows a user to enter the applications environment. | | | | |
| Description | A user can spawn into the environment by clicking on the 'start' button in the main menu. | | | | |
| Pre-condition | The application is running. | | | | |
| Activation | The 'Start' button is clicked. | | | | |
| Post-condition | The user is spawned into the application's environment. | | | | |
| Main flow | 1. Application is running on the user's system.<br>2. System displays the main menu to the user.<br>3. User clicks the 'Start' button.<br>4. System loads the 'Dunboyne' level.<br>5. User is spawned into the level.<br>6. Application pauses and displays the systems control scheme to the user.<br>7. User presses 'x' to close control scheme menu.<br>8. Application begins. | | | | |
| Alternate flow | | | | | |
| Exceptional flow | **E1: The system crashes upon start.**<br>1. User clicks 'Start' button.<br>2. Due to an error, the application fails to start.<br>3. Application closes to desktop and Unreal Engine 5 displays a prompt explaining the error that occurred. | | | | |

#### 2.1.1.3. Requirement 2: Change Settings

| Use Case Name | Change Settings. | Use Case ID | U2 | Priority | High |
|---|---|---|---|---|---|
| Scope | Allows a user to change a variety of the applications settings. | | | | |
| Description | A user can change the settings of the application through the 'settings' button on the main menu. | | | | |
| Pre-condition | The application is running. | | | | |
| Activation | The Settings button is clicked. | | | | |
| Post-condition | The application applies the users desired settings and returns to the main menu. | | | | |
| Main flow | 1. Application is running.<br>2. System displays main menu.<br>3. User clicks the 'Settings' button in the main menu.<br>4. System displays the Settings menu.<br>5. User selects their desired resolution size.<br>6. User selects their desired shadow quality.<br>7. User selects their desired shader quality.<br>8. User selects their desired texture quality.<br>9. User selects their desired window mode.<br>10. User clicks 'apply.'<br>11. The System applies the settings the user has selected.<br>12. The System returns the user to main menu. | | | | |
| Alternate flow | **A1: User changes no settings.**<br>1. The user does not wish to change any settings.<br>2. The user clicks 'back.'<br>3. The system returns the user to the main menu. | | | | |
| Exceptional flow | | | | | |

#### 2.1.1.4. Requirement 3: Exit Application

| Use Case Name | Quit Application. | Use Case ID | U3 | Priority | High |
|---|---|---|---|---|---|
| Scope | Allows a user to exit the application. | | | | |
| Description | A user can quit the application and return to their desktop. | | | | |
| Pre-condition | The application is running. | | | | |
| Activation | The 'Quit' button on the main menu is clicked. | | | | |
| Post-condition | The application is closed, and the user is sent back to their desktop. | | | | |
| Main flow | 1. The application is running.<br>2. The system displays the main menu.<br>3. The user clicks the 'Quit' button in the main menu.<br>4. The application ends all running processes.<br>5. The user is returned to their desktop. | | | | |
| Alternate flow | 1. The user ends the task through other means such as hitting Alt+F4 or through the Windows Task Manager.<br>2. The program stops.<br>3. The user is returned to their desktop. | | | | |
| Exceptional flow | | | | | |

#### 2.1.1.5. Requirement 4: Move player character

| Use Case Name | Move player character. | Use Case ID | U4 | Priority | High |
|---|---|---|---|---|---|
| Scope | Allows a user to move their character in the environment. | | | | |
| Description | A user can navigate the virtual environment by controlling a character using their control device. | | | | |
| Pre-condition | A user has been spawned into the environment. | | | | |
| Activation | The user enters an input on their control device. | | | | |
| Post-condition | The avatar reacts to the users input in the virtual environment. | | | | |
| Main flow | 1. The user has pressed 'start' and is spawned into the level.<br>2. The user enters an input via their keyboard or mouse.<br>3. The game engine recognises the input and translates said input into movement by changing the avatars position in the level.<br>4. The player avatar has been moved. | | | | |
| Alternate flow | | | | | |
| Exceptional flow | **E1: Input device not recognised.**<br>1. No input device is connected or detected.<br>2. The system displays an error message informing the user that an input device cannot be detected. | | | | |

#### 2.1.1.6. Requirement 5: Display object prompt

| Use Case Name | Display object prompt. | Use Case ID | U5 | Priority | High |
|---|---|---|---|---|---|
| Scope | Allows a user to see an interaction prompt when near an interactable object. | | | | |
| Description | When a user approaches an interactable object, the user can see a prompt tied to that object informing them that they can interact with it. | | | | |
| Pre-condition | The user has been spawned into the environment. | | | | |
| Activation | The user enters within a close proximity of an object of interest. | | | | |
| Post-condition | A prompt is displayed informing a user that they can interact with the object. | | | | |
| Main flow | 1. A user walks toward an object of interest.<br>2. When a user enters within a close proximity to an object, a prompt is displayed on the object informing a user they can interact with it.<br>3. When the user hits the interaction button, the objects information menu will be displayed to the user. | | | | |
| Alternate flow | **A1: User enters object's radius but does not interact with it.**<br>1. A user walks toward an object of interest.<br>2. When a user enters within a close proximity to the object, a prompt is displayed on it.<br>3. The user walks away from the object.<br>4. The prompt disappears. | | | | |
| Exceptional flow | | | | | |

#### 2.1.1.7. Requirement 6: Interact with objects information menu

| Use Case Name | Interact with objects information menu. | Use Case ID | U6 | Priority | High |
|---|---|---|---|---|---|
| Scope | Allows a user to interact with an object's information menu. | | | | |
| Description | A user can interact with an objects information menu to learn about the object and its history. The user can play an audio file that is a transcription of the objects information and view a historical photograph of the object. | | | | |
| Pre-condition | A user has pressed the interact button on the objects prompt. | | | | |
| Activation | The objects information menu is displayed on the user's screen. | | | | |
| Post-condition | The user can interact with the various elements in the object's information menu. | | | | |
| Main flow | 1. A user has pressed the interact button on an objects prompt.<br>2. The objects information menu is displayed on the user's screen.<br>3. A historical photograph of the object is displayed in the menu.<br>4. The user can read a text field containing information about the object.<br>5. The user can click on a button that plays an audio transcription that details the history of the object.<br>6. The user can click close, and the menu is removed from their screen. | | | | |
| Alternate flow | A1: User stops audio transcription early.<br>1. The user clicks on the button the plays an audio transcription of the object's history.<br>2. The user clicks close during the audio playback.<br>3. Audio playback is stopped, and the menu is removed from their screen. | | | | |
| Exceptional flow | | | | | |

#### 2.1.1.8. Requirement 7: Display map

| Use Case Name | Display map. | Use Case ID | U7 | Priority | Medium |
|---|---|---|---|---|---|
| Scope | Allows a user to open a map of the environment. | | | | |
| Description | A user can press the 'M' button on their keyboard to open a map of the environment to better aid them in navigation. The user can place custom waypoints to aid their traversal of the environment. | | | | |
| Pre-condition | A user has spawned into the environment. | | | | |
| Activation | The user presses the button assigned to open the map. | | | | |
| Post-condition | The map is displayed on a user's screen. | | | | |
| Main flow | 1. A user has spawned into the environment.<br>2. A minimap is displayed on their screen.<br>3. They press the 'M' button on their keyboard to open their main map.<br>4. The main map is displayed on the user's screen.<br>5. The user can place a custom waypoint on the main map.<br>6. The waypoint will be displayed on their main map and minimap to aid in their navigation of the environment. | | | | |
| Alternate flow | | | | | |
| Exceptional flow | | | | | |

#### 2.1.1.9. Requirement 8: Display Pause menu.

| Use Case Name | Display Pause Menu. | Use Case ID | U8 | Priority | Medium |
|---|---|---|---|---|---|
| Scope | Allows a user to pause the application. | | | | |
| Description | A user can press a button on their keyboard that will pause the application and display a pause menu on their screen. | | | | |
| Pre-condition | A user has spawned into the environment. | | | | |
| Activation | The user presses the 'P' button to pause the application. | | | | |
| Post-condition | The application is paused, and the pause menu is displayed on the user's screen. | | | | |
| Main flow | 1. A user has spawned into the environment.<br>2. The user presses the 'P' button to pause the application.<br>3. The application stops and displays the pause menu on the user's screen.<br>4. The user can save their progress by clicking the 'Save' button.<br>5. The user can click 'Return to main menu' to go back to the application's main menu.<br>6. The user can exit the application by clicking 'Quit Application'.<br>7. The user can resume the application by clicking 'Resume'. | | | | |
| Alternate flow | **A1: User loads a save file.**<br>1. User presses the 'P' button to pause the application.<br>2. Application stops and displays the pause menu on the user's screen.<br>3. The system loads the save file and starts the application from their save point. | | | | |
| Exceptional flow | | | | | |

#### 2.1.1.10. Requirement 9: Prevent user from traversing out of bounds

| Use Case Name | Prevent user from traversing out of bounds. | Use Case ID | U9 | Priority | Low |
|---|---|---|---|---|---|
| Scope | Stops a user from navigating outside of the intended play area. | | | | |
| Description | A user will receive a warning when they navigate outside of the playable area of the environment. When this occurs, they will receive a warning on their screen. If the warning is not heeded, they will be placed back at their original spawn point. | | | | |
| Pre-condition | A user has spawned into the environment. | | | | |
| Activation | The user navigates outside of the playable area of the map. | | | | |
| Post-condition | The user is respawned at their original spawn point. | | | | |
| Main flow | 1. A user has spawned into the environment.<br>2. The user navigates into an unfinished or out-of-bounds area of the map.<br>3. The system displays a warning message on the user's screen and a countdown begins.<br>4. The user does not heed the warning message.<br>5. At the end of the countdown, the user is respawned at their initial spawn point. | | | | |
| Alternate flow | **A1: User returns to playable area before countdown finishes.**<br>1. The user has navigated into an unfinished area.<br>2. The system displays a warning message on the user's screen.<br>3. The user heeds the warning message and returns to the playable area of the map.<br>4. The warning message disappears from the user's screen. | | | | |
| Exceptional flow | | | | | |

### 2.1.2. Data Requirements

The data requirements of the Epoch Explorer application are as follows:

**Mesh Data**: Unreal Engine stores data pertaining to the mesh's of the environment. This includes the landscape data and object data.

**Texture data**: Unreal Engine stores all the texture data used within the application. These textures are mapped to their corresponding meshes to ensure a visually pleasing environment.

**Save data**: Unreal Engine stores the save files of user. These save files log the game state at the time of saving and upon being loaded into memory, the engine recreates the game state present in the save file.

**Image data**: The images of objects that appear in their information menus are stored in Unreal Engine and are retrieved upon runtime in the information menu.

**Audio data**: The audio transcription present in the object information menus are stored in Unreal Engine as .wav files and are retrieved upon runtime when the user clicks the 'play audio' button.

**Text data**: The text data associated with objects is stored in Unreal Engine and is retrieved upon runtime.

**Settings data**: The application settings that a user sets are stored in a 'GameUserSettings.ini' file and are applied on runtime.

**UI assets**: The various UI assets are stored in Unreal Engine as texture files. These assets are applied to widgets and these widgets are retrieved on runtime.

### 2.1.3. Environmental Requirements

**Operating system**: Epoch Explorer is compatible with Windows operating systems.

**System requirements**: Due to the fidelity of the assets and the size of the environment, the user should ideally have a system that meets the minimum requirements of the application. These requirements will be better determined when the application is developed further and more assets are added - but the application was developed using a GTX 1070Ti, 16GB of RAM, and an Intel Core i5-8600K. Ideally, a user's PC should be of a similar specification to my own system to ensure a comparable experience, but the ability to change resolution and graphics settings means that a user with less powerful hardware will still be able to experience the application, albeit at a decreased level of fidelity.

**Screen resolution**: A user should ideally have a screen resolution of between 1920x1080 or 2560x1440 to get intended level of fidelity for the application. However, all user interface elements are responsive and will adapt to any screen size. The application was developed on a system with a 2560x1440 monitor, but a user can still experience the application with resolutions higher or lower than this by changing their resolution in the settings menu.

**Inputs**: A user should have a mouse and keyboard to interact with the application.

**Storage**: Due to the nature of the application being a video game, the file size is quite large because of the number of textures and other assets contained within it. The user will need at least 66GB of storage to download and run the application.

### 2.1.4. Usability Requirements

The Epoch Explorer application was developed with the user experience at the core of its design ethos. To achieve this, it aims to adhere to Jakob Nielsen's '10 Usability Heuristics for User Design' [6]. When the application is started, the user is presented with a minimalistic and easy to read main menu that conveys key functions of the application. The user can start the application, change the applications settings, or exit the application. These options are all conveyed through high-contrast, minimalist and easy to read user interface widgets, embodying principle #8, 'Aesthetic and Minimalist Design'. The design of the menus in the application are consistent throughout, even echoing the colour scheme and design of the Epoch Explorer logo. This showcases principle #4, 'Consistency and Standards'. The ability of a user to customise the applications settings is in keeping with principle #7, 'Flexibility and Efficiency of Use'.

The widget elements are responsive and react to a user's inputs. For example, when a user hovers their mouse over an element and clicks it, the elements change colour to give the user feedback that their input is being recognised. This embodies principle #1, 'Visibility of System Status'.

Once the user has started the application, a screen is displayed that lists the controls for the character. This is to ensure that even users unfamiliar with interactive experiences or common control schemes can understand what input is needed to control the character. This highlights principle #10, 'Help and Documentation'.

When navigating the environment and approaching an object, any object that is interactable will display a prompt with the corresponding interaction button displayed. This is to ensure that the user knows exactly what objects are interactable and which ones are not. The fact the widget appears on screen means the user does not need to break up their experience by consulting documentation to confirm the control scheme needed to interact with an object. This embodies principle #6, 'Recognition rather than Recall'.

When an objects information menu is displayed, a user will be able to interact with the menu by playing an audio transcription of this text. This is to ensure accessibility for those that might need or prefer an audio transcription of the information about the object. The text-to-speech voice chosen has an Irish accent, since the applications setting is rural Ireland.

Should a user leave the playable area of the application, they will be informed of this fact by easy to understand, red text. This follows principle #9, 'Help Users Recognize, Diagnose and Recover from Errors'.

Finally, in terms of performance, the application should run at a continuous 60 frames per second with little stuttering or performance drops.

## 2.2. Design & Architecture

**Application design**: The application was designed using Unreal Engine for all the components. The logic behind each component is handled by 'blueprint' code which is a GUI interface for implementing C++ code. There are various classes of objects in the project such as widgets, levels, assets etc. Each of these can have underlying logic applied to them with blueprints. The blueprints system follows the concept of Object-Oriented Programming [7].

The key features of the application are the menu systems - including saving and loading data, the map systems, the 'out of bounds' system, and the object information menu system.

For the save system, a file will be generated by unreal engine that keeps a log of the game state at the time of saving. This file will be retrieved upon loading - resuming the game from that point.

The map system will provide a top-down view of the environments terrain and will be ever-present on the user's screen in the form of a minimap. A larger version of the map that is only shown when an input is pressed is also available. When the map button is pressed, the map widget will be displayed by Unreal Engine and a user will be able to place custom waypoints on this map. When a user clicks on their map, Unreal Engine will take the co-ordinates of the user's mouse click and translate it to the corresponding environment co-ordinates. A waypoint actor will be spawned at these environment co-ordinates. This waypoint actor's location will then be displayed on the main map and minimap in the form of a widget component to act as a navigation aid for the user to travel towards.

Finally, the object interaction system is a major design aspect of the application. It involves showing a prompt on an object when a user is within a certain distance. When the interact button is pressed, Unreal Engine will display a widget on the user's screen. This widget is the objects information menu. This menu will be interactable and will allow a user to play an audio file which is a transcription of the text present in the widget. This audio file will be stored in Unreal Engine and will be retrieved when the 'play audio' button is pressed.

**Level design**: The level design is a crucial aspect of the project that ensures that the application is as immersive and historically authentic as possible. Because the application is a recreation of a real place as it existed in the past, careful work has been put into ensuring it is as accurate a recreation as possible. Accuracy was achieved by referencing ordnance survey maps to get an exact layout of the town from the period. The map was generated using topographic data from satellite captures using 'Unreal Mapbox Bridge.' Placeholder 'whiteboxed' [8] buildings were then added to the level to represent the buildings that were present during the period captured by the ordnance survey.

One building was scanned using drone photogrammetry to ensure its preservation and accuracy. This process involved flying a drone around the object in question in an orbit and taking pictures every few seconds to ensure all angles of the object were captured. These images were then input into Polycam - which generated a 3D model of the object. This 3D model was then inputted into blender and the unwanted vertices were deleted. This edited 3D model was then exported back into Unreal Engine and placed in the applications environment.

Historical photography was consulted throughout the project to ensure an accurate recreation of the other aspects of the environment such as road surfaces, grass, foliage, and other features of the town. These historical images and historical information used in the application were sourced from the 'Old Dunboyne Society' [9], 'The Historical Picture Archive' [10], the National Built Heritage Service [11] and Meath County Council [12].

## 2.3. Implementation

This section of the report will discuss the implementation of the main features of the Epoch Explorer application and a thorough explanation of the blueprint code used to realise these features. Each feature is housed in its own section, and certain features fall under the category of a subsection of a main feature.

### 2.3.1. Main Menu

The main menu is its own level called `MainMenuMap`. This is what the user will load into when the application is running. The actual level itself is empty, but when the application runs, the main menu widget is retrieved and shown on the user's screen. Widgets are a form of blueprint in unreal that allow for the handling of user interface elements. The main menu widget was created and then populated with a variety of assets created using the program 'paint.net' [13] and then imported into Unreal Engine. This process of asset creation and importation was used to create all the widgets for this project.

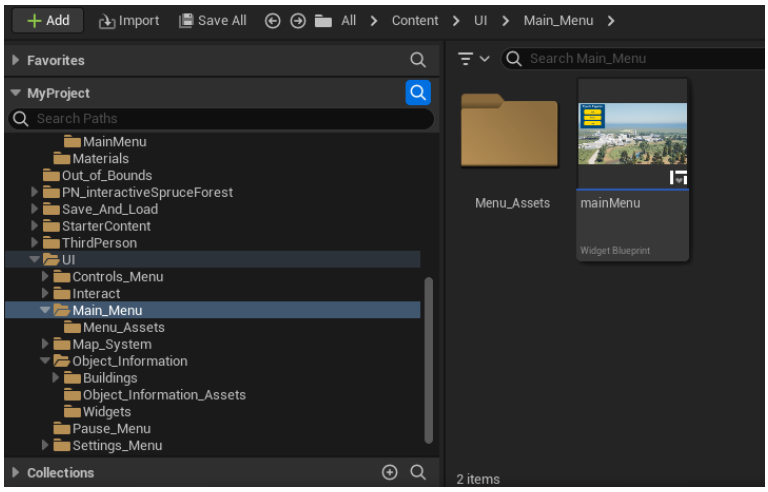


*Figure 2 Main Menu Widget Blueprint*

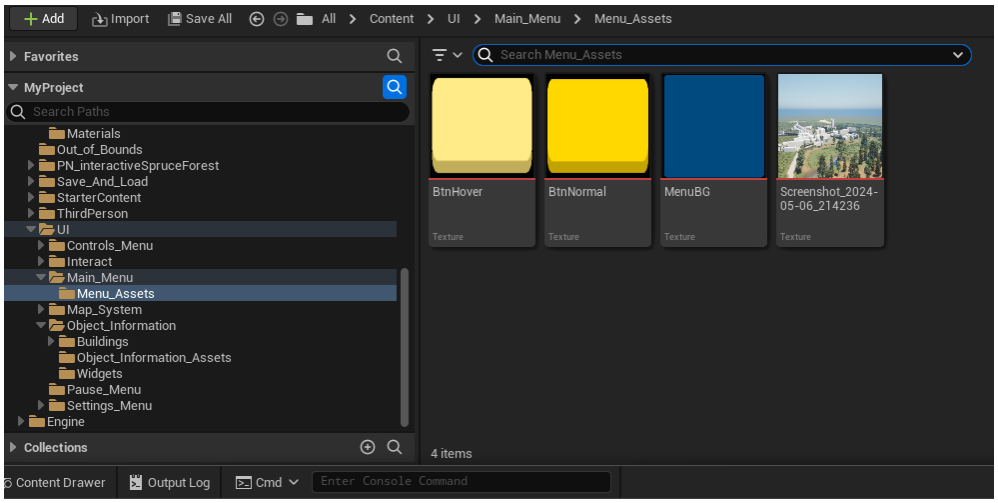


*Figure 3 Main Menu Widget Assets*

At this point, the settings menu is also retrieved and the `OnPlay` event is run, which retrieves the user's current graphics settings.

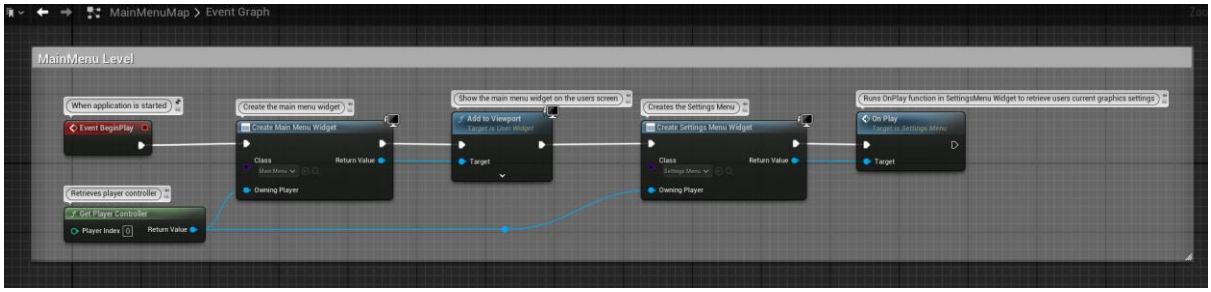


*Figure 4 MainMenuMap Blueprint*

The `mainMenu` widget blueprint itself has the underlying logic that allows a user to spawn into the environment, change their settings and quit the game. When the `mainMenu` is loaded, the users mouse cursor will be shown on screen and their input mode will be set to `UI only`, which allows them to interact with user interface elements. When a user clicks the 'Start' button, the application will open the 'Dunboyne' Level. The players mouse cursor, used to navigate the main menu, will be removed from their screen and their input mode will be set to `Game` mode, ensuring that they can move their character when spawned into the level. When the user clicks the 'Settings' button, the `mainMenu` widget will be removed from their screen. The `settingsMenu` widget will then be created and shown on their screen.

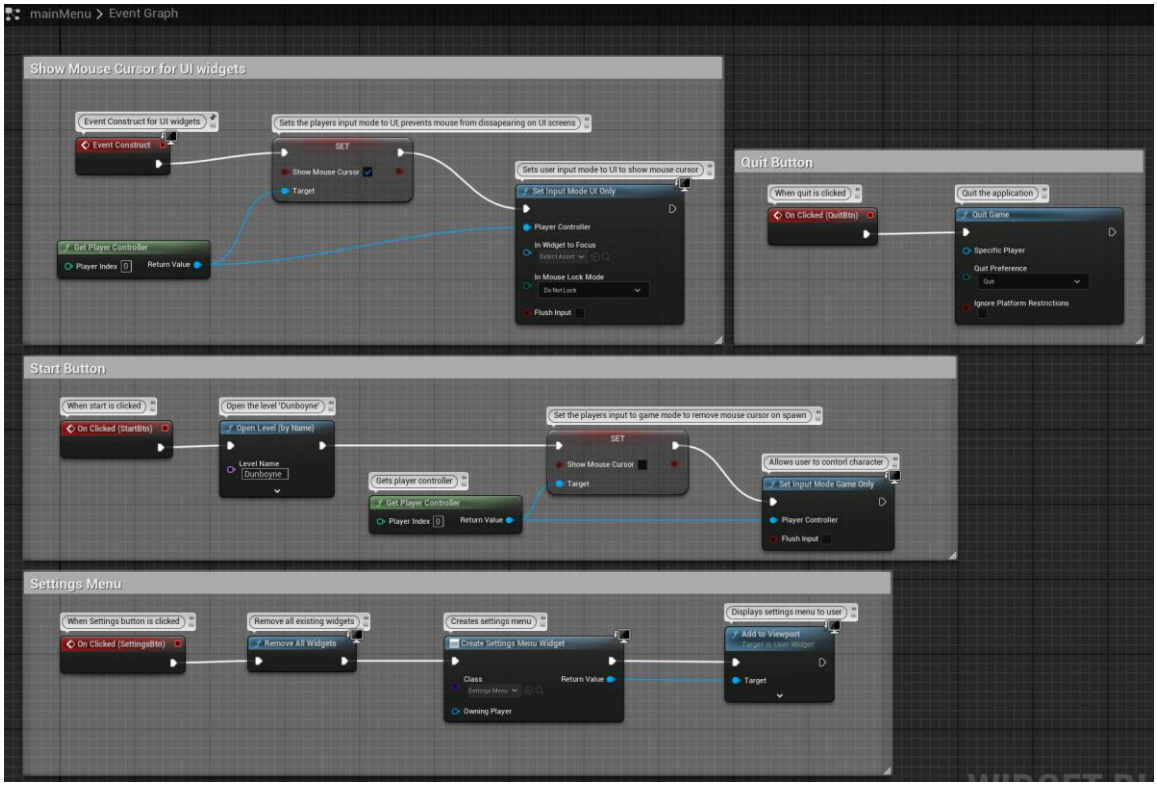


*Figure 5 mainMenu Widget Blueprint*

### 2.3.2. Display Controls

When the application starts, the controls will be displayed to the player. This is handled by the code in the `BP_FirstPersonCharacter` blueprint, which pauses the application when the Dunboyne level is loaded – and then displays the `Controls` widget. When a user presses the X key on their keyboard, the application is un-paused, and the `Controls` widget is removed from their screen.

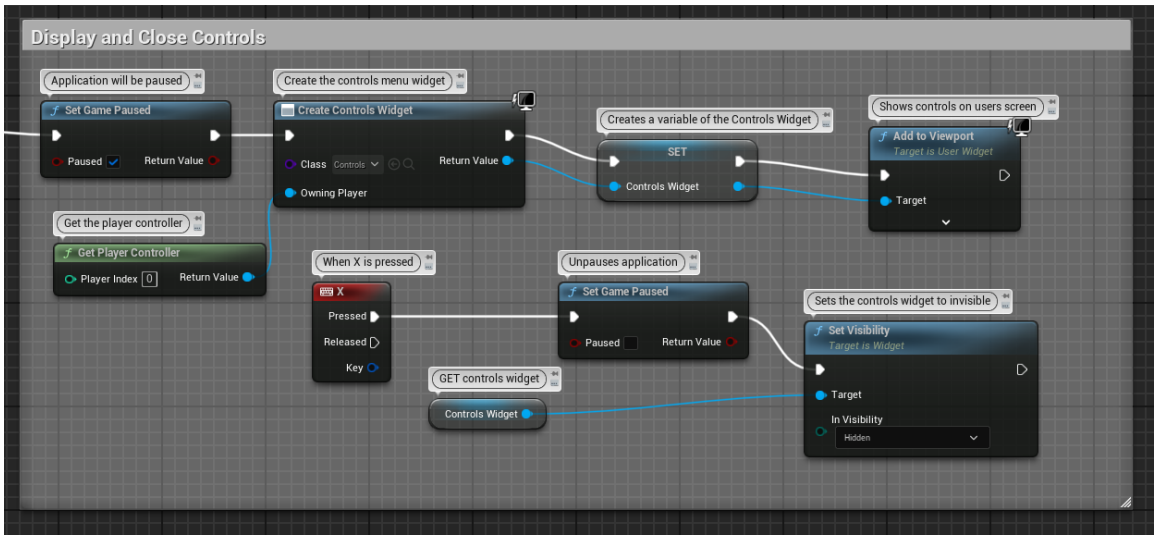


*Figure 6 Controls Widget Blueprint*

### 2.3.3. Pause Menu

If a user wants to pause the application to either take a break, save their progress, load a save file, return to the main menu, or quit the application - the pause menu allows them to do so. The pause menu is handled by code housed inside the `BP_FirstPersonCharacter` blueprint. When the 'P' button is pressed by the user on their keyboard, the engine will create the `pauseMenu` widget and show it on the user's screen. The users' controls will also be switched to `UI Only` mode, so they can use their mouse to click on the `pauseMenu` buttons. Finally, the application is paused.

An input action for pausing was created under actions.

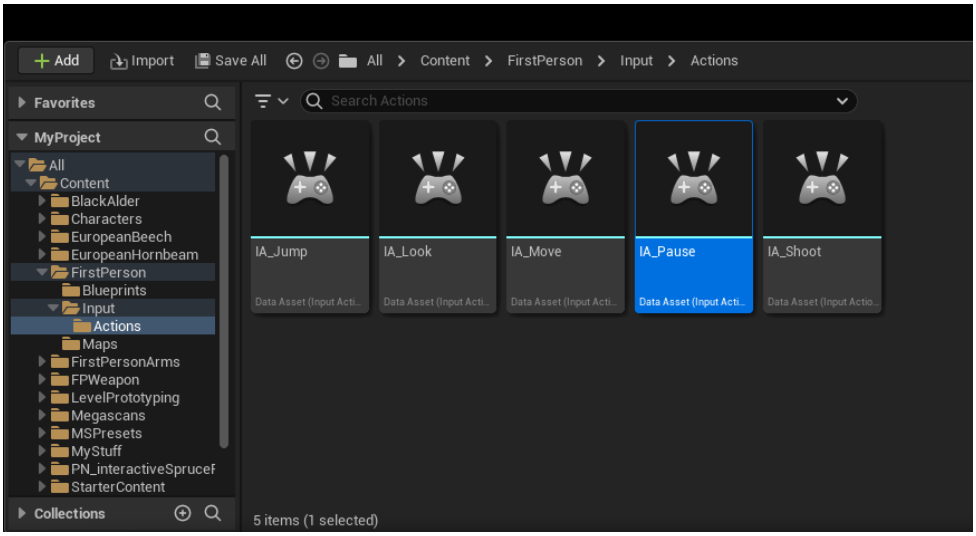


*Figure 7 Pause Input Action*

This input action, `IA_Pause` was then set to trigger when the ‘P’ button on the user’s keyboard is pressed.

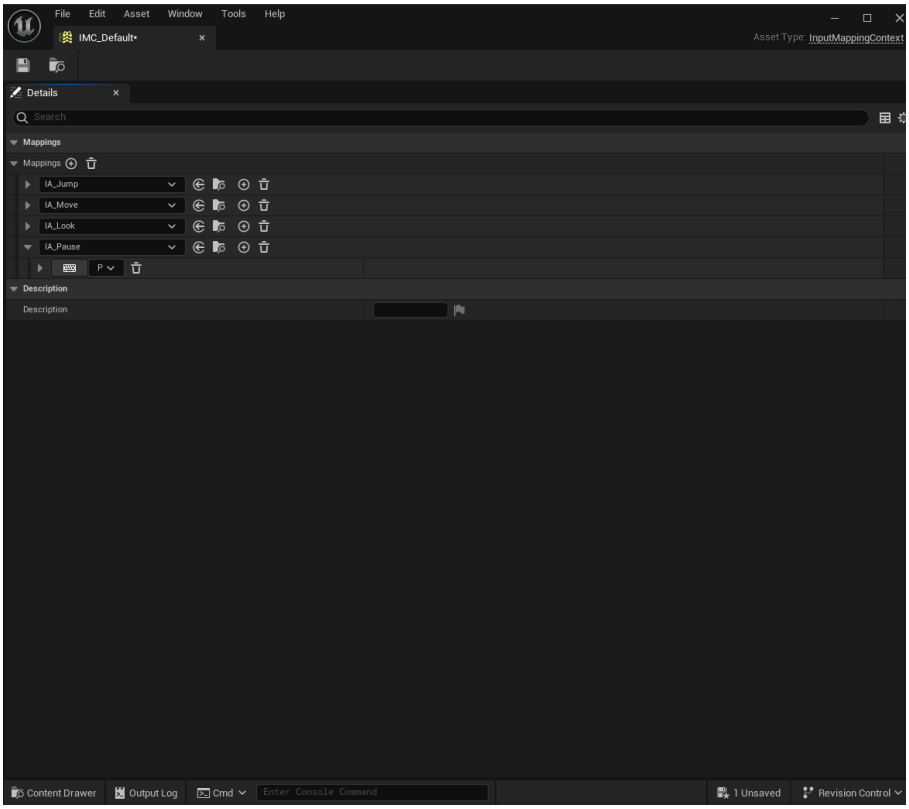


*Figure 8 IA_Pause Key Assignment*

When the ‘P’ button is pressed, the application is paused, and the `pauseMenu` widget will be displayed on the user's screen.

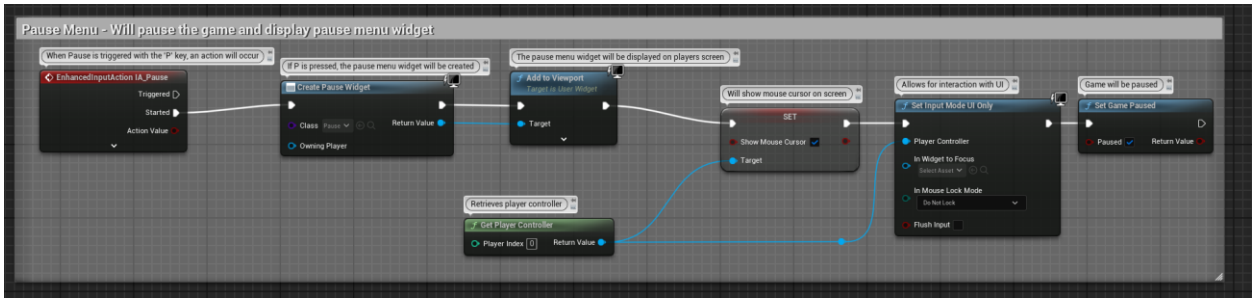


*Figure 9 pauseMenu Blueprint*

### 2.3.3.1. Resume Application

When a user has opened the pause menu, they can choose to resume the application by clicking the 'Resume' button. When this button is clicked, the game state is set to un-paused. The mouse cursor is removed from the user's screen and their game mode is set to `Game` mode. Finally, the `pauseMenu` widget is removed from their screen.

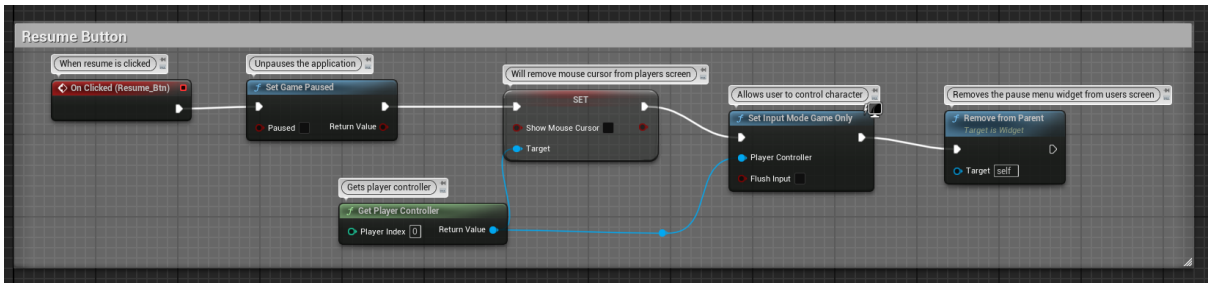


*Figure 10 Resume Button code in pauseMenu widget Blueprint*

### 2.3.3.2. Save/Load Application

To implement the ability for a user to save their game, two blueprint classes were created; `EEGameInstance` of type `GameInstance` and `EESave` of type `SaveGame`. These blueprints allow Unreal Engine to store information about the game instance and the users save data.

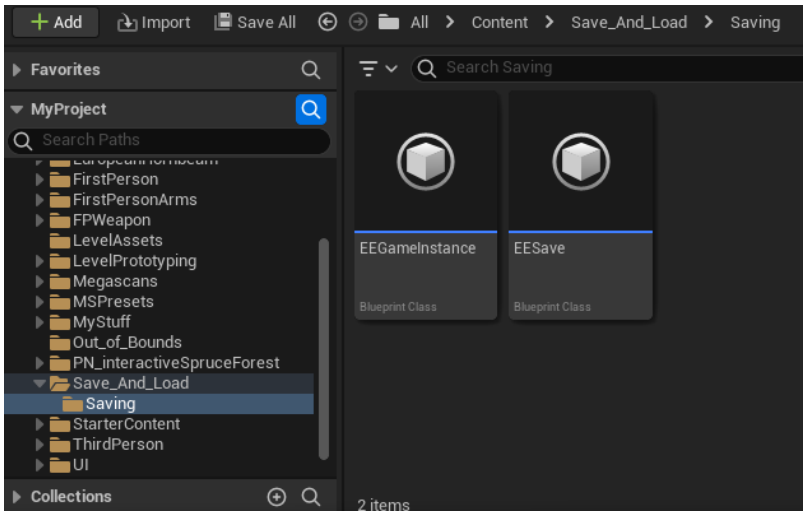


*Figure 11 EEGameInstance and EESave Blueprint Classes.*

In the `EESave` class, a variable called `PlayerTransform` was created. In Unreal Engine, a 'Transform' variable refers to the co-ordinates of an 'actor' in the environment. This variable will save the location of the player character and allow the user to load the application and resume from the location they last saved their progress at.

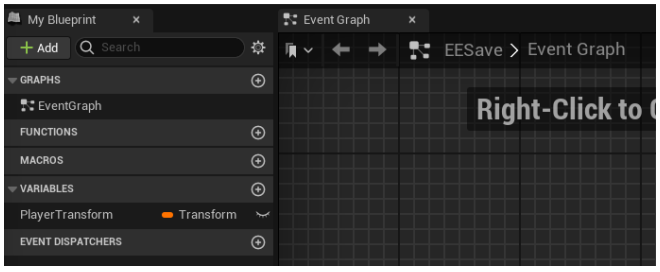


*Figure 12 PlayerTransform Variable in EESave Blueprint*

In the `pauseMenu` widget blueprint, the application will retrieve the users current game instance, cast it to the `EEGameInstance` class and set it to a variable called `EEGameInstanceReference`.

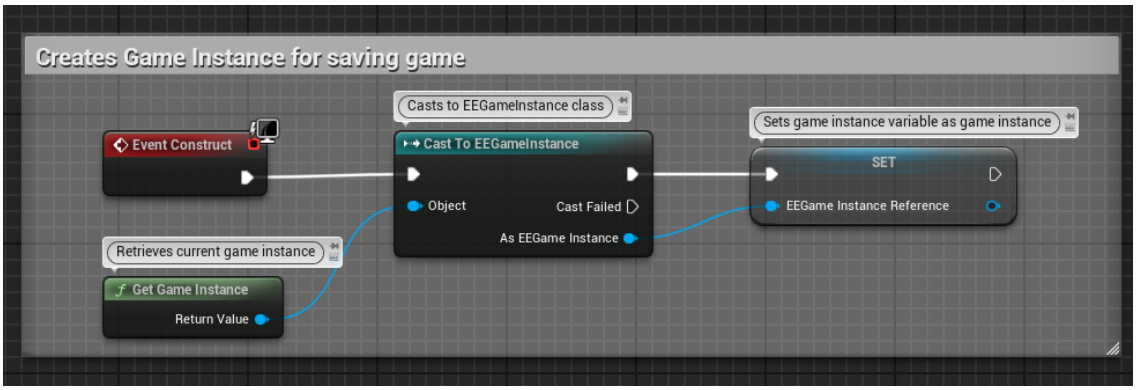


*Figure 13 Setting EEGameInstanceReference Variable*

In the `EEGameInstance` blueprint, a series of variables were created that handle the references to the classes required for the save game functionality. A variable called `Player` is an object reference to the player character. `Player` is used to cast to the `GameInstance` class, which allows it to access variables from the player character, in this case, their transform/location.

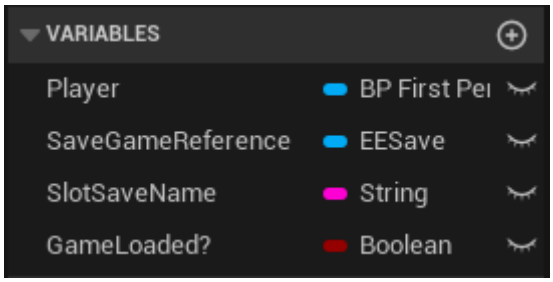


*Figure 14 Variables in EEGameInstance Blueprint*

Next, a series of functions were created that handle the logic for saving and loading the application.


*Figure 15 Functions in EEGameInstance*

The `SaveGame` function handles the logic behind saving the player location to the save game slot. It creates a save game object using the `EESave` blueprint. This is then set as a variable called EE`SaveGameReference`. The `PlayerTransform` variable inside `EESave` is then set to the transform value of the player character. Finally, this transform information is stored inside the `EESaveGameReference` variable, and saved as a save slot called `SlotSaveName` which will be used to load the save game information.

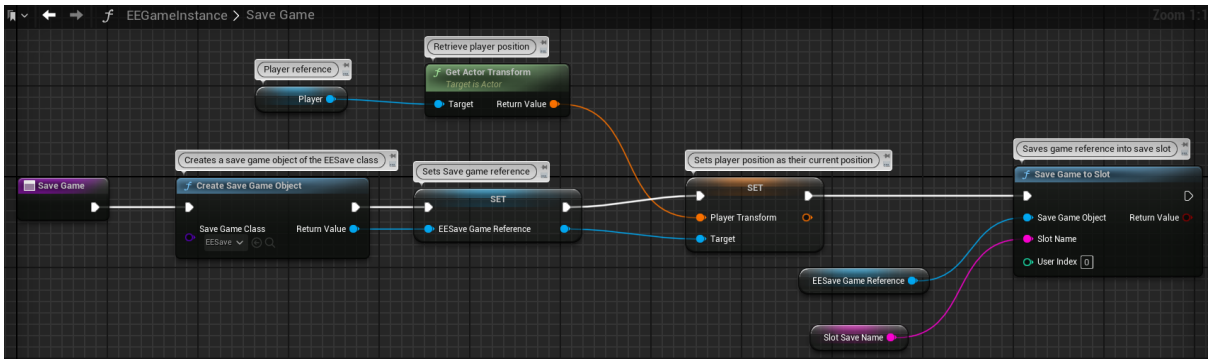


*Figure 16 SaveGame Function Blueprint*

The `LoadSaveGame` function handles the logic behind loading the information from the `SaveGameSlot` variable. It loads the slot, then casts to the `EESave` class, setting the `SaveGameReference` variable with the information held inside the `SlotSaveName` variable.

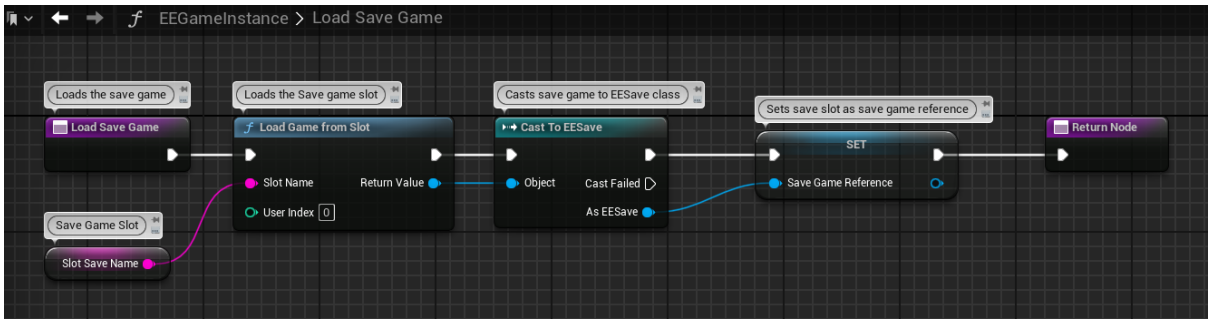


*Figure 17 LoadSaveGame Function Blueprint*

The `LoadApplication` function handles the logic to extract the actor transform information from the `SaveGameReference` variable, which allows the player character to be spawned in at the transform co-ordinates stored using the `SaveGame` function.

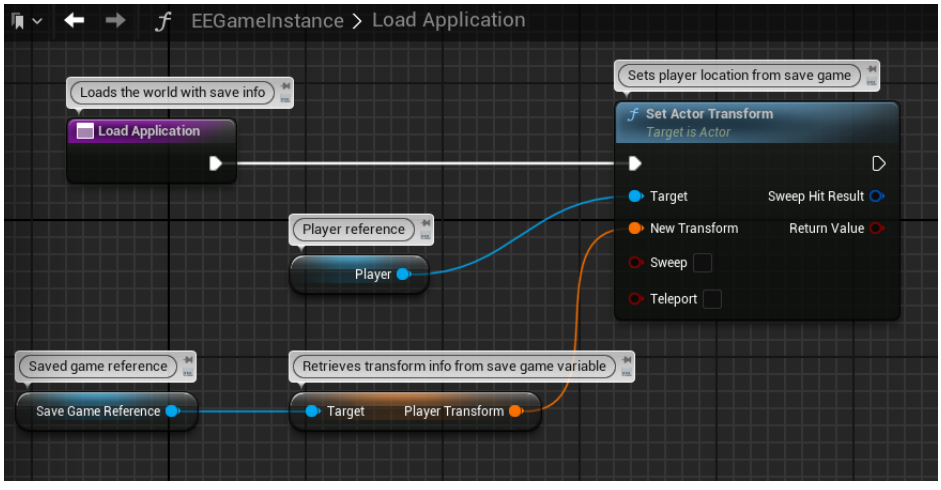


*Figure 18 LoadApplication Function Blueprint*

With the saving and loading handled, in the `EEGameInstance` blueprint, an event called `LoadGameFromMenu` was created that will handle the loading of a save game. When this event is called, a Boolean called `Game Loaded?` will be set to true to indicate that the loading process has begun. Next, the `LoadSaveGame` function is run that will retrieve the `EESaveGameReference` from the `SlotSaveName` slot. Once this occurs, the application will open the Dunboyne level, remove the mouse from the user's screen and then set their input mode to `Game`.

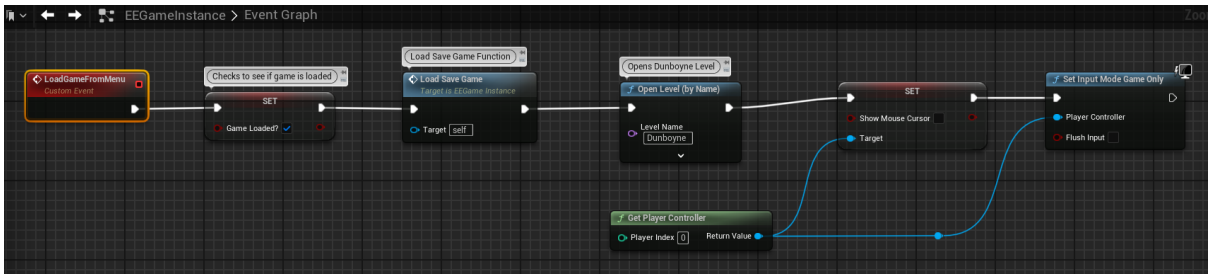


*Figure 19 EEGameInstance Blueprint*

With all the required functions present, the loading functionality is handled in the `BP_FirstPersonCharacter` blueprint. When the player spawns into the world, the `Player` variable inside `EEGameInstanceReference` will be set as the current player using a reference to `Self`. The application will check to see if the `Game Loaded?` Boolean is set to true – which indicates that the loading process had begun. If this is false, nothing will happen, and the game will start from the normal starting point. If this Boolean is true, the application will run the `LoadSaveGame` and `LoadApplication` functions and set the `GameLoaded` Boolean to false. This will spawn the player character at the transform co-ordinates held in their save file, starting the level from that position.

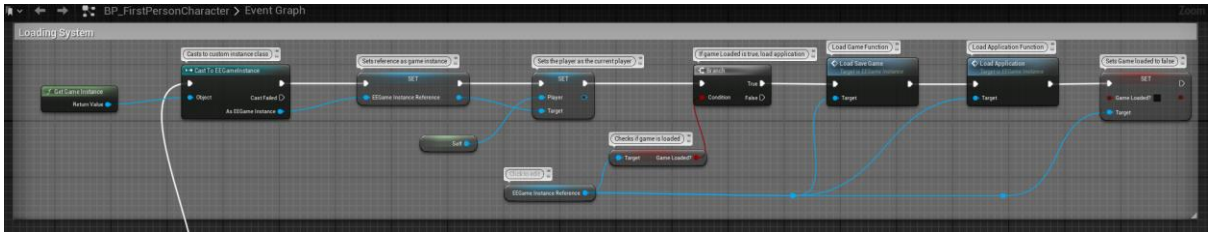


*Figure 20 Loading System in BP_FirstPersonCharacter Blueprint*

The user can save their game by using the 'Save' button in the `pauseMenu` widget. When the user clicks the 'Save' button, a `SaveSlotName` will be saved in the `EEGameInstanceReference` variable using the `SaveGame` function. The mouse cursor will then be removed from the user's screen and their input mode is set to `Game` mode. Finally, the `pauseMenu` is removed from the user's screen and the application is un-paused.

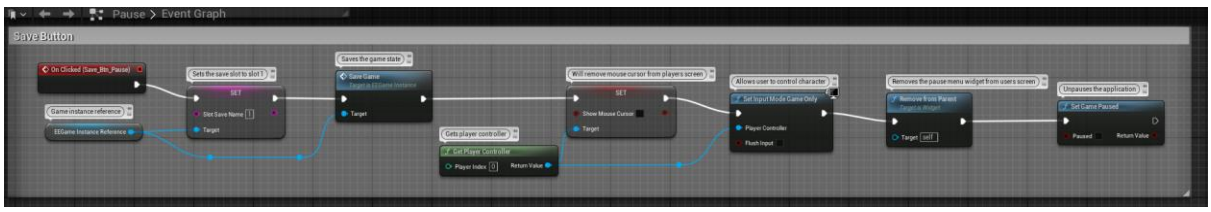


*Figure 21 Save Button System in pauseMenu Widget Blueprint*

The user can load their game by using the ‘Load’ button in the `pauseMenu` widget. When the user clicks the ‘Load’ button, the `SlotSaveName` variable will be retrieved from the `EEGameInstanceReference` variable using the `LoadGameFromMenu` function, which will spawn the player at the transform co-ordinates saved when they clicked ‘Save’.

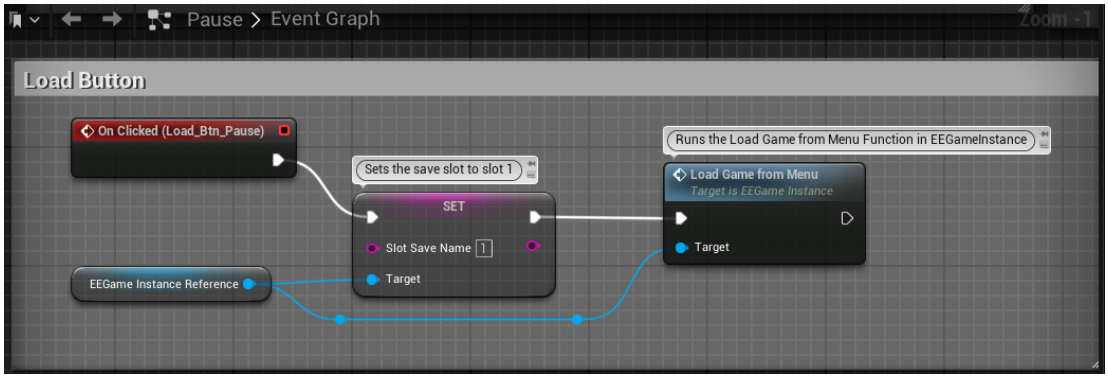


*Figure 22 Load Button System in pauseMenu Widget Blueprint*

### 2.3.3.3. Return to Main Menu/Quit Application

Through the `pauseMenu` widget blueprint, the user can return to the Main Menu or quit the application entirely. When the ‘Return to Main Menu’ button is clicked by the user, the `MainMenuMap` level will be opened, returning the user to the main menu. When the user clicks the ‘Quit Application’ button, the application will be closed.

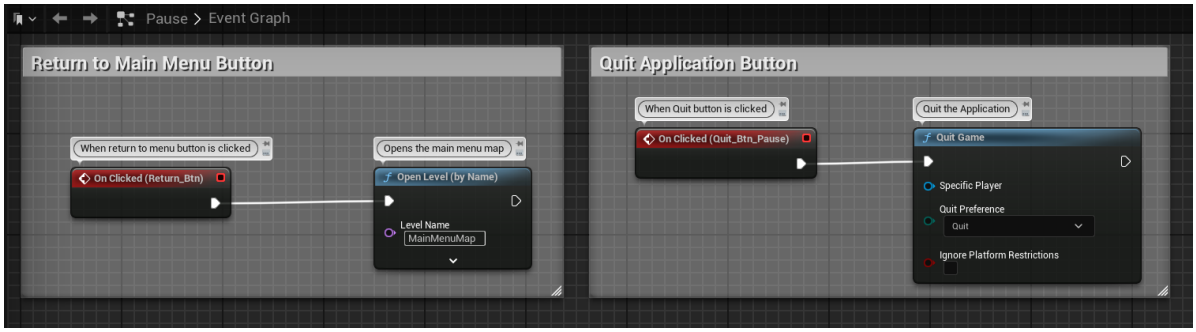


*Figure 23 Return to Main Menu and Quit Application button systems in pauseMenu widget blueprint*

### 2.3.4. Display Object Prompt

When a player is in the environment and walks close to an interactable object, the object will display an interaction prompt due to an `OnComponentBeginOverlap` event making the Interaction widget visible. When the player walks away from the object, the interaction prompt will disappear due to an `OnComponentEndOverlap` event making the interaction prompt invisible.

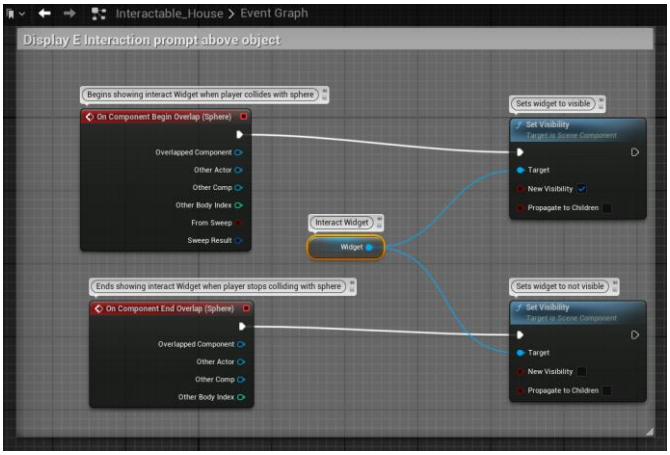


*Figure 24 Display Interaction Prompt blueprint*

These events work by virtue of having a collision sphere placed around the object. When a user collides with the sphere, the interaction prompt is made visible to the user. When the user stops colliding with the sphere by walking out of its radius, the interaction prompt disappears.

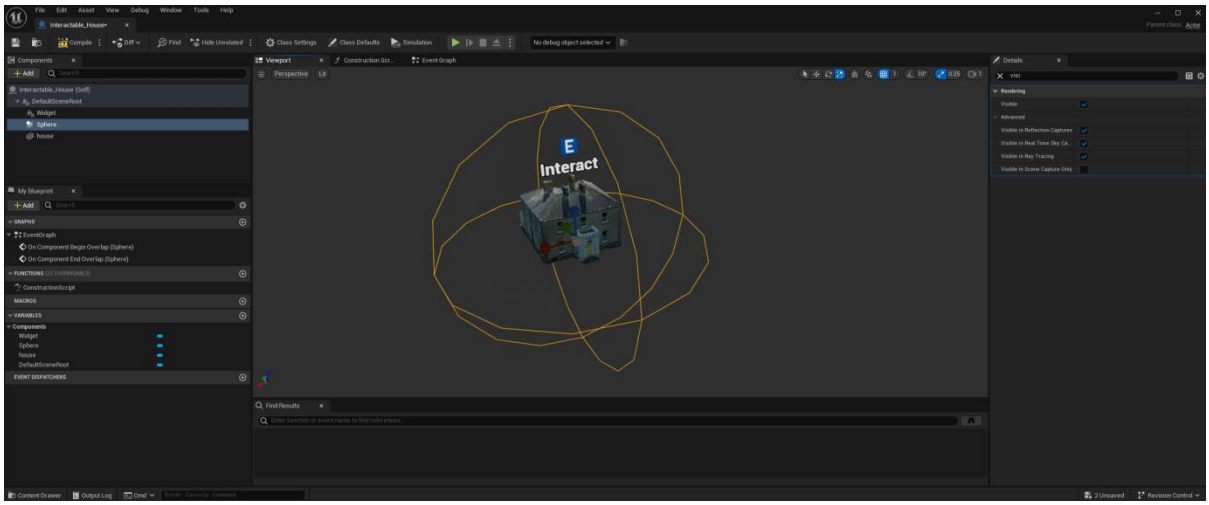


*Figure 25 Collision Sphere Placed around object to trigger Events*

### 2.3.5. Object Information System

The object information system recognises when a player has entered a collision sphere of an object and has pressed the interaction button. When this occurs, the object information widget for the relevant object is displayed. A blueprint interface called `InteractionInterface` was created that handles the interaction between the user and an object. Any interactable object has the `InteractionInterface` assigned to it. An input action called `IA_Interact` was also created that maps the 'E' button on the user's keyboard to the interaction functionality.

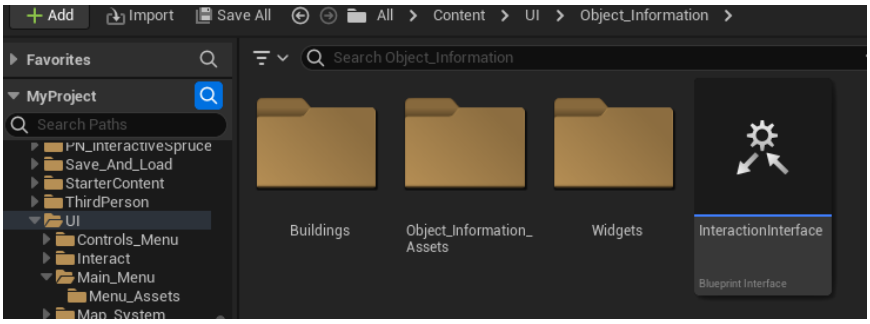


*Figure 26 Interaction Interface*

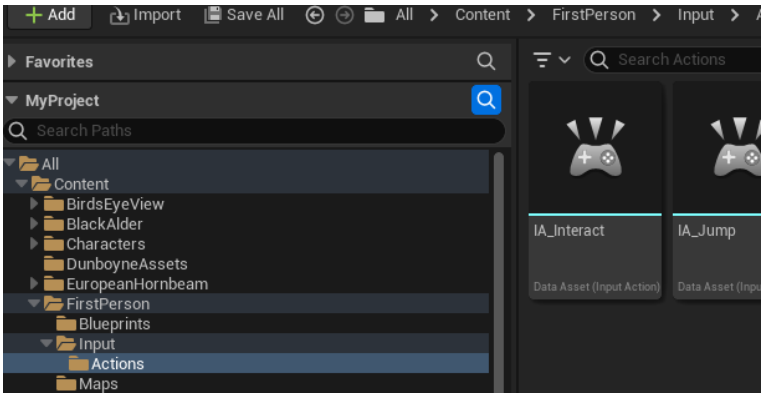


*Figure 27 Input Action IA_Interact*

In the `BP_FirstPersonCharacter` blueprint, when the input action `IA_Interact` is triggered by the user pressing 'E' on their keyboard, the application creates an array of any actors that are overlapping with the player character. The application will execute a for-each loop and will check to see if the actor in the array implements the `InteractionInterface` interface. If this is the case, then the `Interaction` event will be triggered, and the loop will break. If this is not the case, the loop will run again.

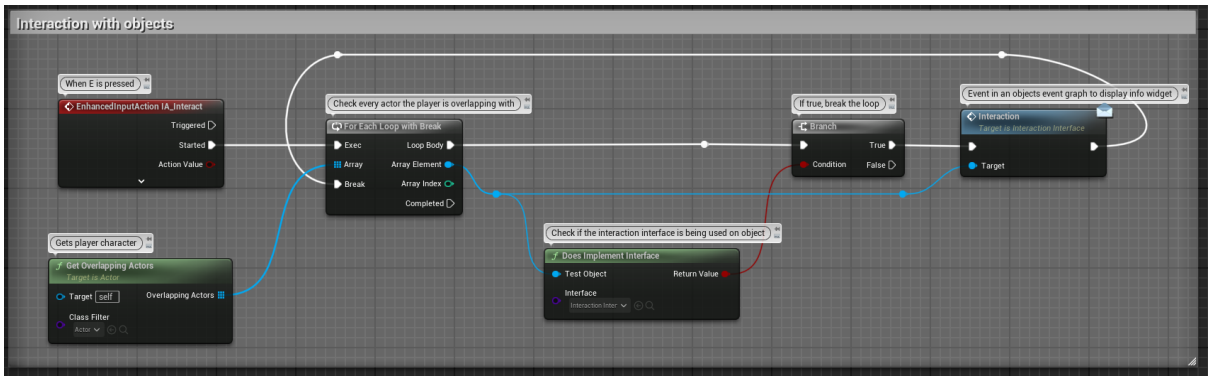


*Figure 28 Detection of Interaction with Objects Blueprint*

When the `Interaction` event in an objects blueprint has been triggered, the application will create the widget for the corresponding object and display it on the user's screen. In this case, the Parochial House object. The mouse cursor will be shown on the user's screen and their input mode will be set to `UI` so that they can interact with the information widget. The game state is also set to paused.

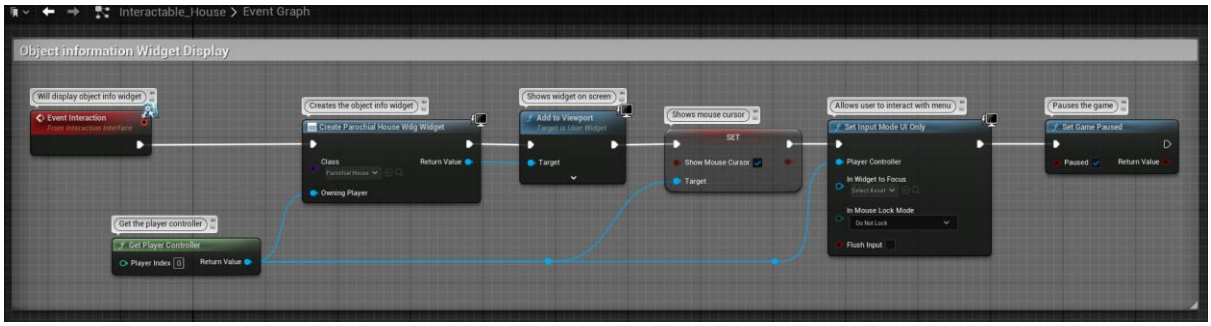


*Figure 29 Interaction Event in interactable object's Blueprint*

Each object has its own unique information widget that displays historical information about the object.

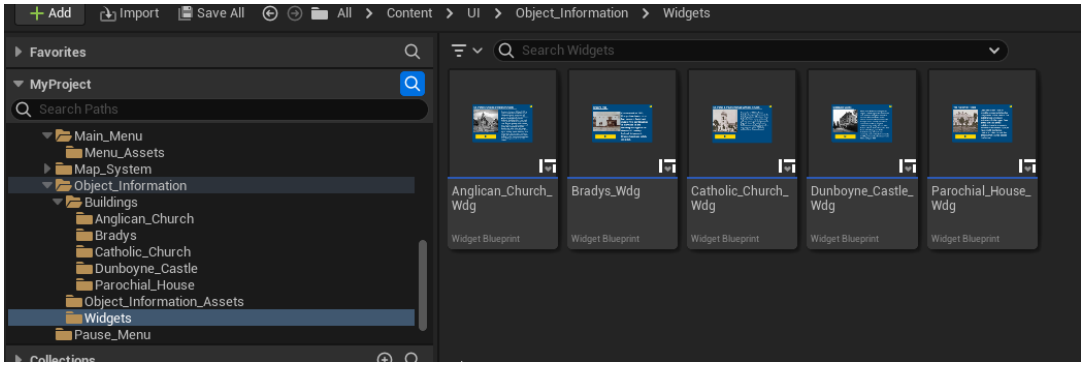


*Figure 30 Object information Widgets*

Each widget has also an audio transcription of the object's description in the form of a .wav audio file.

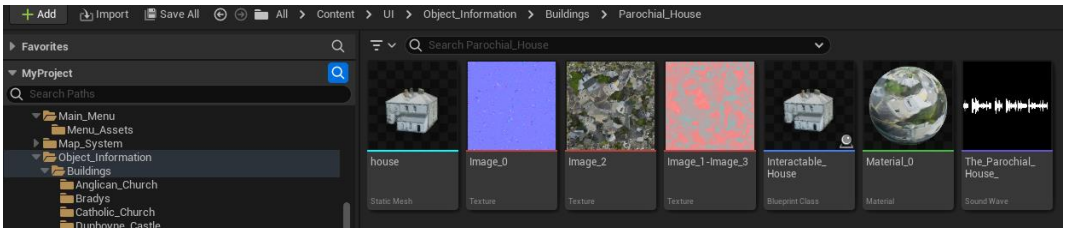


*Figure 31 Audio transcription file for the Parochial House Object*

These .wav files were created using the website ‘Narakeet’ [14], which converts text into text-to-speech .wav files.

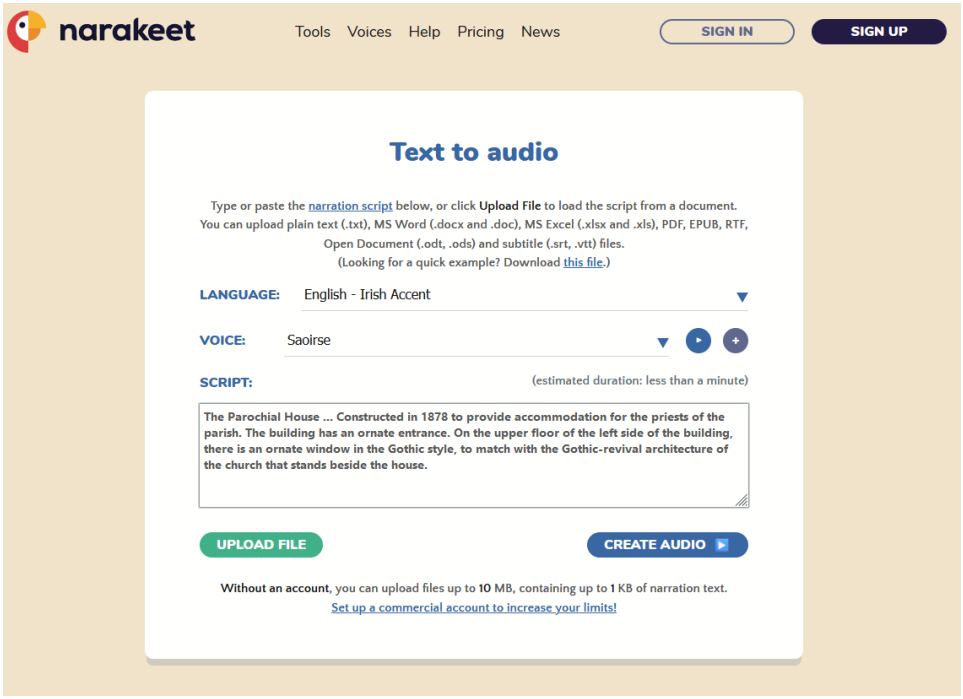


*Figure 32 Narakeet in use for the audio transcription of objects*

When a user presses the ‘Play Audio’ button on an information widget, the application will play the .wav file for the object by spawning a sound using a `SpawnSound2D` node. This sound is set as a variable called `Transcription`, which is required to stop the sound during playback should a user wish to do so. A Boolean `PlayTranscription`, is set to true to indicate that a sound is playing - and a delay with a length equivalent to the length of the sound file is set. This is done to prevent the user continuously pressing the ‘Play Audio’ button and restarting the sound file each time. Once the period of the delay is over, the `PlayTranscription` Boolean will be set to false, and the `DoOnce` node will be reset so another sound can be played.

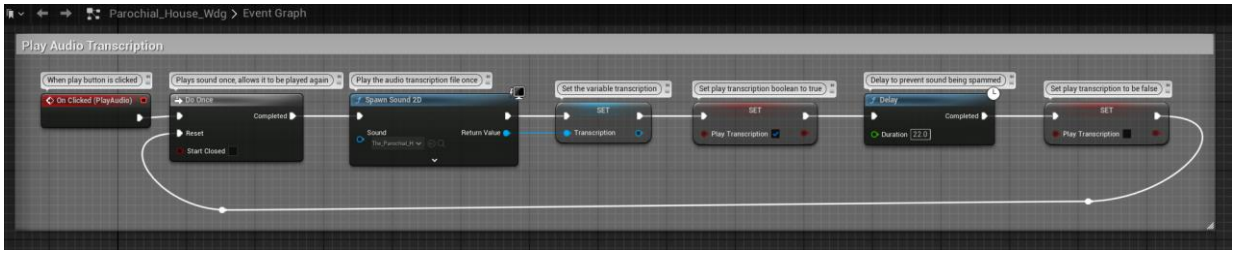


*Figure 33 Play audio transcription Blueprint*

When a user wants to close the object information widget on their screen, they can click the 'X' button on the top right of the widget. When this button is clicked, the application will un-pause the game and remove the mouse cursor from the user's screen and set their input mode to `Game`. Then, a branch will trigger. The application will check if the `PlayTranscription` Boolean is true which indicates that an audio file is currently playing. If this is the case, the `Transcription` variable will be retrieved and stopped – stopping the playback of the audio transcription .wav file. The object information menu will then be removed from the user's screen. If the `PlayTranscription` Boolean is false, the information widget will be removed from the player screen.

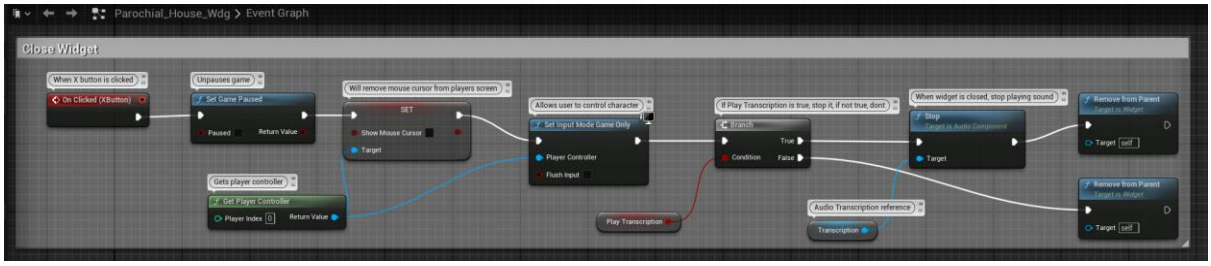


*Figure 34 Close Information widget and Stop Audio Playback Bluprint*

### 2.3.6. Out of Bounds System

The out of bounds system uses a similar collision detection functionality to what I implemented to show the interaction prompt on an object. An actor called `Boundary` was created that consists of a box. Four of these `Boundary` boxes surround the playable area of the environment and are invisible to the user during runtime. A widget called `BoundaryMessage` was also created to display a message on the user's screen, warning them that they have reached the boundary of the level.

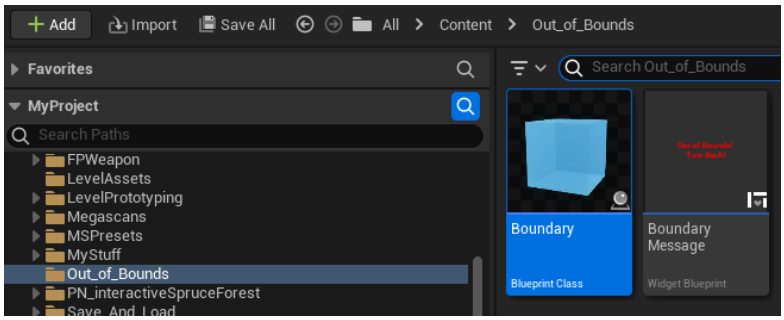


*Figure 35 Boundary Actor and BoundaryMessage Widget*

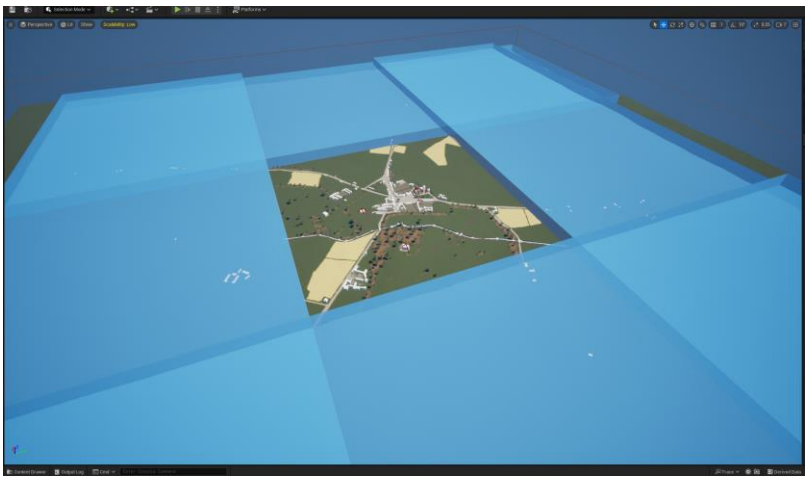

*Figure 36 Boundary Objects Surrounding Playable Area*

When a user traverses to the edge of the playable area, their player character will overlap with the `Boundary` object. The application will recognise when this overlap occurs using a `OnComponentBeginOverlap` event. When the player is colliding with the `Boundary` object, the Boolean `BeyondBoundary` will be set to true. The `BoundaryMessage` widget is created, set as a variable called `BoundaryWidget`, and added to the user's screen. This widget informs the user that they are beyond the level boundary and should turn back. When a user collides with the boundary, a delay of 10 seconds will start, giving the user time to turn around and return to the playable area. After 10 seconds, the application will check to see if `BeyondBoundary` Boolean is true. If this is the case, the `DestroyActor` function will be run, destroying the player character.

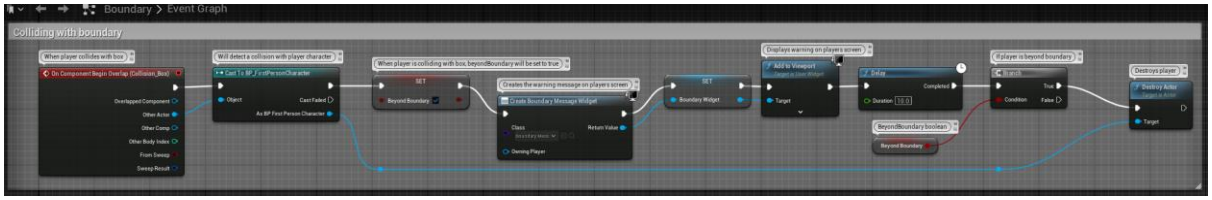


*Figure 37 Colliding with Boundary Blueprint*

If a user has stopped colliding with the `Boundary` object, the `BeyondBoundary` Boolean will be set to false. The `BoundaryWidget` will then be removed from the user's screen and the variable will be set to `null`.

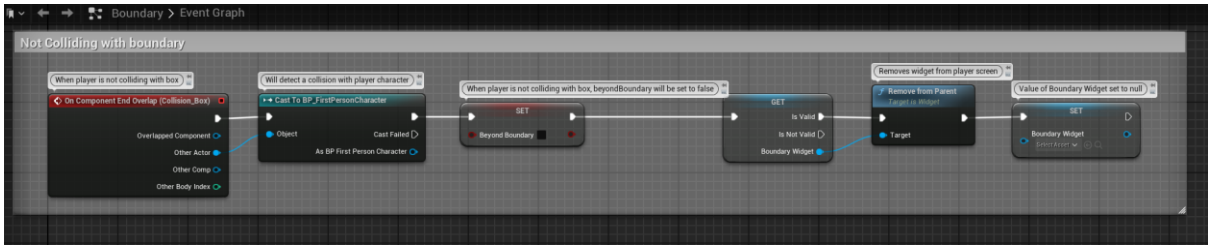


*Figure 38 Not Colliding with Boundary Blueprint*

To handle the destruction of the user's character when they have run out of time in the boundary, a 'respawn' system was implemented. The respawn system functions by firstly setting a variable in the `BP_FirstPersonGameMode` game mode class. This variable called `SpawnTransform` sets the transform co-ordinates of the initial spawn point of the player character when the level is first loaded.

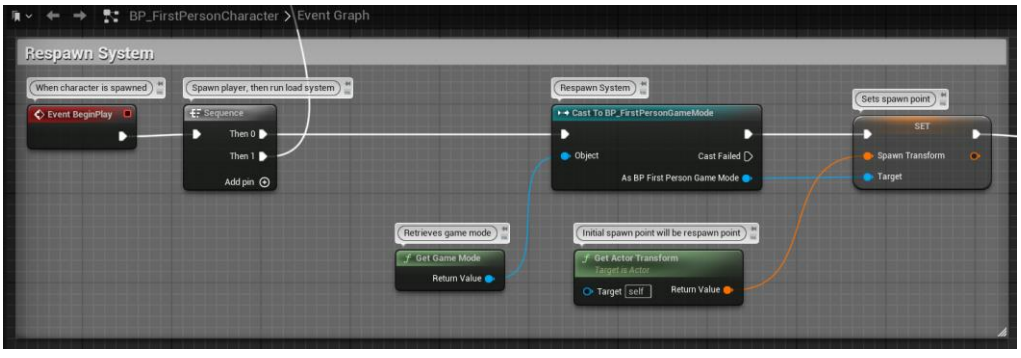


*Figure 39 Setting of SpawnTransform Variable*

With the `SpawnTransform` variable set, in the `BP_FirstPersonGameMode` game mode class, a custom event called `Respawn` was created. The application will retrieve a reference to the player character and will spawn an actor of the same class as the player character at the location set by the `SpawnTransform` variable. Once the new player actor is spawned, the player will possess the new actor. Respawning the player.

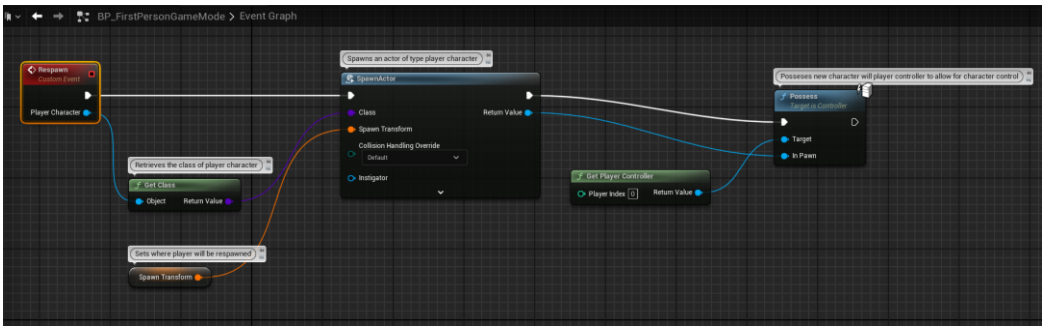


*Figure 40 Respawn Event Blueprint*

The `Respawn` event is used in a custom `Destroy` event that overrides the default `Destroy` event present in Unreal Engine. The new `Destroy` event casts to the game mode `BP_FirstPersonGameMode` and retrieves the current player's game mode. This `Destroy` event then calls the `Respawn` event from the Game Mode class, which passes in a reference to the current player character. This `Destroy` is used in the 'Collliding with Boundary' system in the `Boundary` blueprint to destroy the player character and respawn them if they are out of bounds for more than 10 seconds.

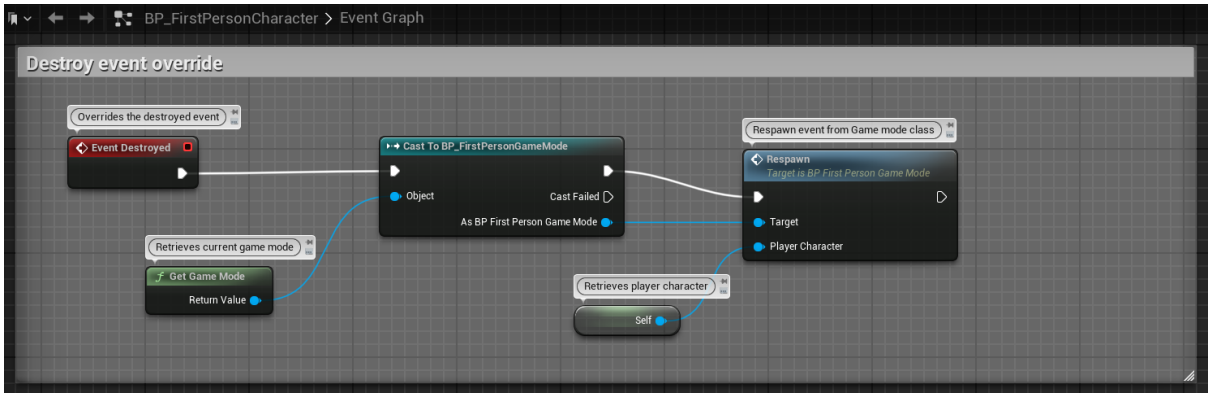


*Figure 41 Destroy Event Override Blueprint*

### 2.3.7. Minimap and Main Map system

For the map system, there is a minimap that is always present on the user's screen with a series of hard-coded waypoints that represent the major locations in the town. This aids them in navigating the environment. Should the user want to travel a larger distance or get a better overview of the layout of the town, they can press the 'M' key on their keyboard to open a main map. The user can click anywhere on the map with their mouse cursor to add a custom waypoint, which will serve as a navigation aid for them to follow. These custom waypoints are reflected in both the minimap and main map, just as the hard-coded waypoints are. To create the minimap, I first needed an image to serve as the map. I went onto the ordnance survey viewer Geohive and took a screenshot of the town of Dunboyne from 1909. This image encompasses the playable area of the Epoch Explorer application.


*Figure 42 Dunboyne Map Asset*

#### 2.3.7.1. Minimap system

Once I imported this map of Dunboyne into Unreal Engine, I created a material instance for the minimap called `Mini_Map_Material`. I also created a simple circle image with a chevron in the middle to represent the player character. This was imported into Unreal Engine and applied as an opacity mask over the Dunboyne map texture. With the textures imported, I needed to adjust the size of the map texture to fit within the confines of the circle and have an appropriate level of zoom. To do this, I needed to adjust the UV values of the map texture. UV values in Unreal Engine refer to a set of parameters that handle the offset, rotation, and scale of a texture in the engine. The Unreal Engine documentation in conjunction with a tutorial by 'Valsogard Enterprise' [15] proved invaluable in implementing this feature. I set the U and V scale parameters to 0.16 which I felt was a good level of zoom for the map texture.

Next, I needed a way to move the map texture to simulate the map moving in the minimap when the player moves in the environment. For this, I needed to retrieve the location of the player and tie the X and Y coordinates of the player to the U and V values of the map texture. To achieve this, I created a material parameter collection. Inside this parameter collection, I created a vector parameter called `PlayerCoOrds` that tracks the players X and Y coordinates in the environment. I also added a scalar parameter called `PlayerRotation` that tracks the players Z axis in the environment. The setting of these values is handled in the `PlayerHUD` blueprint.

With the tracking parameters, the map would move – but not at a speed that properly reflected the movement speed of the player. To ensure the map moved at the correct speed, the UV values of the texture needed to be adjusted. I retrieved the `PlayerCoOrds` parameter and broke this value into two components, R (Representing the player's X axis value) and G (Representing the players Y axis value).

When the player moves forward in the X axis in the environment, the map texture needed to be dragged downwards to represent the player moving “North”. So as the X value of the player increases - the V value of the map texture needed to be subtracted to “drag” the map texture downwards at the correct speed. The value of 52000 was settled on after rigorous testing. I would spawn into the environment and choose a fixed point of reference such as a building. I would walk to the southern-most point of the building and begin a timer. With the timer running, I would then walk to the northern-most point of the building and stop the timer when I reached this point. With this timer value, I would then tweak the value subtracted from the map textures V parameter until the scrolling of the map matched up with the player movement exactly. With the value set, when I would reach the northern-most point of a building with my player character, the chevron on the minimap would be at the northern-most point of the building too, meaning that the players movement speed and the map scrolling matched up correctly. The same process was repeated for the U value of the map, but this time I would traverse from the western-most point of a building to the eastern-most point with a timer running.

The UV parameters in Unreal Engine are constrained between a value of 0-1. Because you can only have a value in between 0-1, the 52000 that the X axis value is subtracted from is then divided by 52000 to convert it to a value between 0-1. The same is also done for the Y axis value.

Finally, an offset is added to ensure that the centre of the minimap is placed where the player character is in the environment. To figure out this value, I adjusted the offset values until the centre of the map reflected the spawn point of the character, which is the x: 0 and y: 0 point of the environment. Values of 0.4756 and 0.4 achieved the result of centring the minimap on the players location.

These steps handled the movement of the map texture up-down and left-right, but I needed to also rotate the map texture to reflect the players rotation in the environment. To do this, I added a `CustomRotator` node and set the rotation centre to be 0.5 and 0.5 to allow the map texture to rotate around the chevron in the middle of the minimap that represents the player location. I then plugged in the `playerRotation` parameter, which handles the tracking of the player characters rotation value. This value is set in the `PlayerHUD` class.

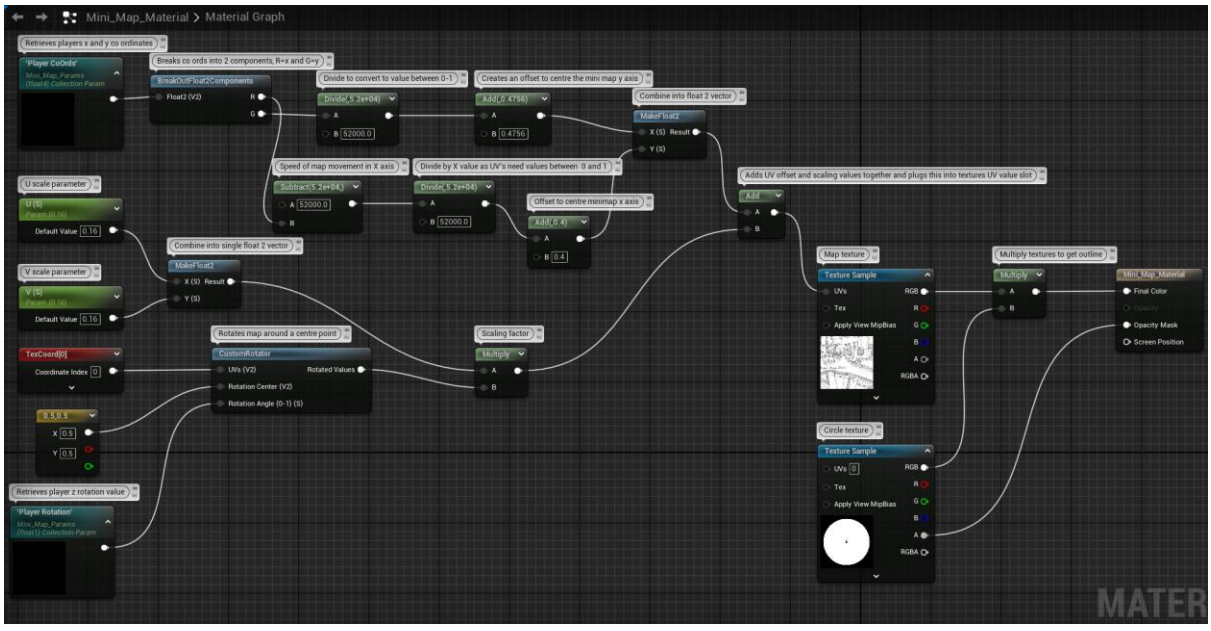


*Figure 43 Mini_Map_Material Blueprint*

The `PlayerHUD` class handles the creation of the minimap widget and the setting of the values of the `PlayerCoOrds` and `PlayerRotation` parameter values. An event tick is used to update these values every frame in the engine. The player character is retrieved, and their location is returned using a `GetActorLocation` node. This location is then plugged into a `SetVectorParameterValue` node, and the value is set as the `PlayerCoOrds` parameter. The player rotation is then retrieved using a `GetActorRotation` node. The z axis is retrieved, which is the players rotation value. In the game environment, the player rotation is a value between -180 and 180. The `CustomRotator` node can only take in a value between 0-1. Therefore, the player rotation value needed to be converted. A `SelectFloat` node was used to handle this conversion. If a rotation value is less than 0, the A value will be selected. The A value is the rotation plus 360. If the value is above 0, B will be selected. B is the raw value. These values are then divided by 360 to convert them between 0-1. The final value is then plugged into the `SetScalarParameterValue` node and assigned to the `PlayerRotation` parameter.

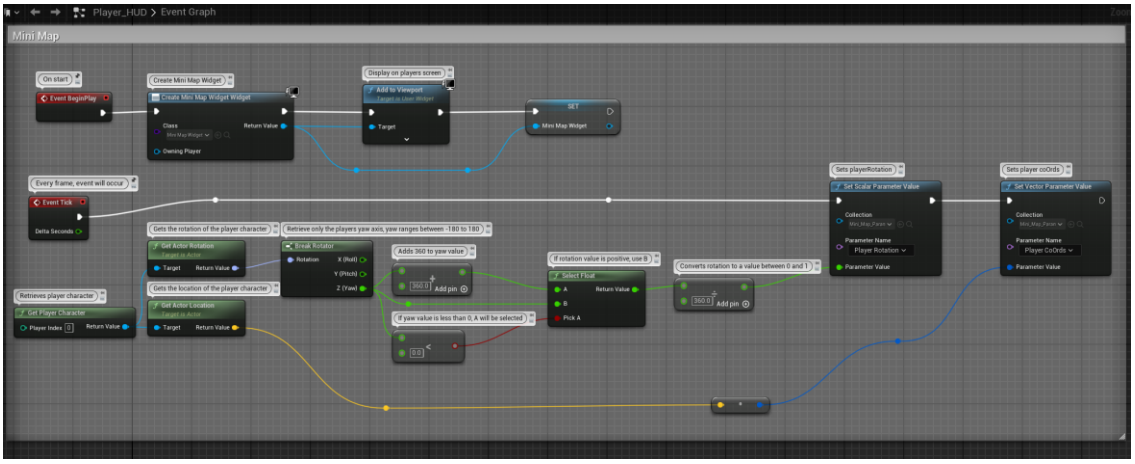


*Figure 44 Mini Map Logic in PlayerHUD Blueprint*

With the movement of the minimap to scale with the players movement set up, I then went about adding waypoints to the minimap. The first step in this process was to create an actor blueprint that would serve to retrieve the world position of a waypoint, and an icon widget to show icons on the minimap widget.

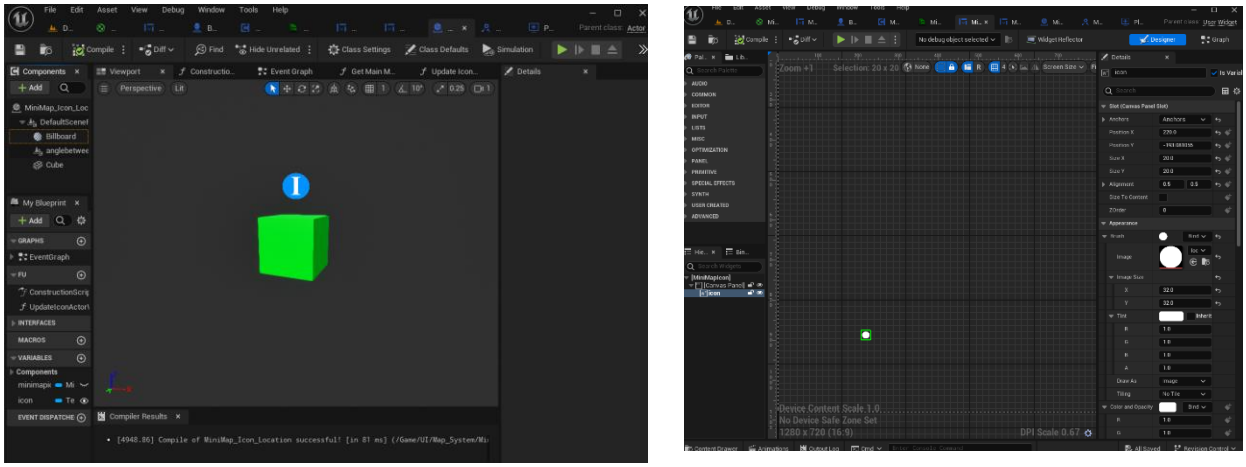

*Figure 45 MiniMapIconLocation Actor and MiniMapIcon Widget*

Once these assets were created, I then added blueprints to the MiniMapIconLocation actor. When the application is started, the MiniMapIcon widget will be created. Because there are multiple different icons for each different actor, the icons for each are updated using the UpdateMiniMapIconWidget event, which is housed in the MiniMapIcon widget class. Once the MiniMapIcon widget is added to the user's screen, a scene component called AngleBetween2Vectors is attached to the player character. This scene component is used to get the game engine to run the calculations for the Angle Between 2 Vectors formula [16]. This formula is used to update the position of the icons on the minimap relative to the player character's location. Once this scene component is attached to the player, the UpdateIconActorWorldLocationUsingAB2V function is run. This function is run every frame using an event tick to continuously update the relative location between the player character and a MiniMapIconLocation actor.

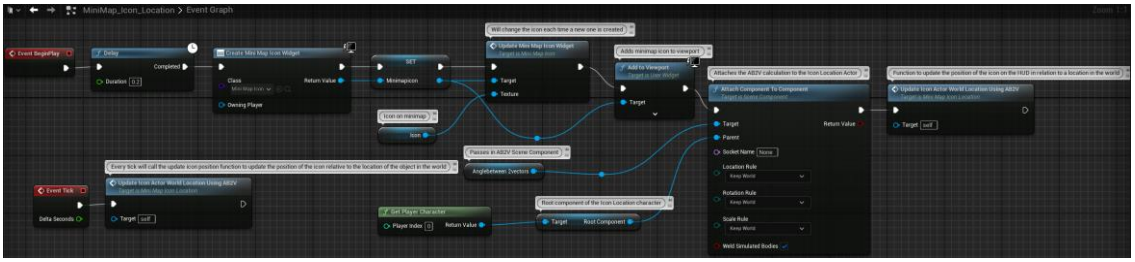

*Figure 46 MiniMapIconLocation Actor Blueprint*

The UpdateIconActorWorldLocationUsingAB2V retrieves the location of the actor using a GetActorLocation node. The location of the MiniMapIconLocation actor is passed into a SetWorldLocation node along with the scene component variable, AngleBetween2Vectors. The MiniMapIcon variable set in the MiniMapIconLocation actor event graph is then retrieved and is passed into the UpdateMiniMapIconWidgetPosition event, which is housed in the MiniMapIcon widget blueprint. The relative location between the MiniMapIconLocation actor and the player character is retrieved using the GetRelativeLocation node. This does the Angle Between 2 Vectors calculation and the result is passed into the UpdateMiniMapIconWidgetPosition event.

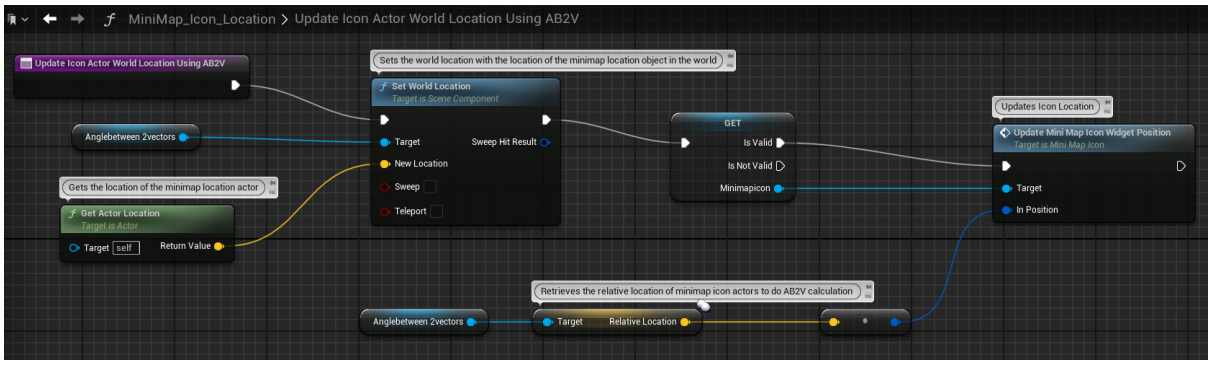


*Figure 47 UpdateIconActorWorldLocationUsingAB2V Function Blueprint*

With all the required functions set up, I could now add these waypoint icons to the minimap. This was handled by the UpdateMiniMapIconWidgetPosition event in the for the MiniMapIcon widget blueprint. The UpdateMiniMapIconWidgetPosition event uses the relative location between the player character and a MiniMapIconLocation actor in the form of the InPosition input. This input is applied through a Clamp node with a value of 4000, which sets the extent to which the icons can move on the minimap widget. Without a Clamp in place, the icons would leave the bounds of the minimap widget as the player character moves around the environment. The InPosition input is then broken into a vector 2D so that the X and Y values of the MiniMapIcon widgets can be adjusted independently. This is done to ensure that the positions of the MiniMapIcon widgets on the minimap

are accurate relative to the location of the corresponding `MiniMapIconLocation` actor in the environment. To achieve this, a difference ratio is applied to the `MiniMapIcon` widgets. Once this ratio is applied, the `MiniMapIcon` widget positions are set using a `SetPosition` node and are shown on the minimap. How the difference ratio for these widgets was calculated is detailed in the next paragraph.

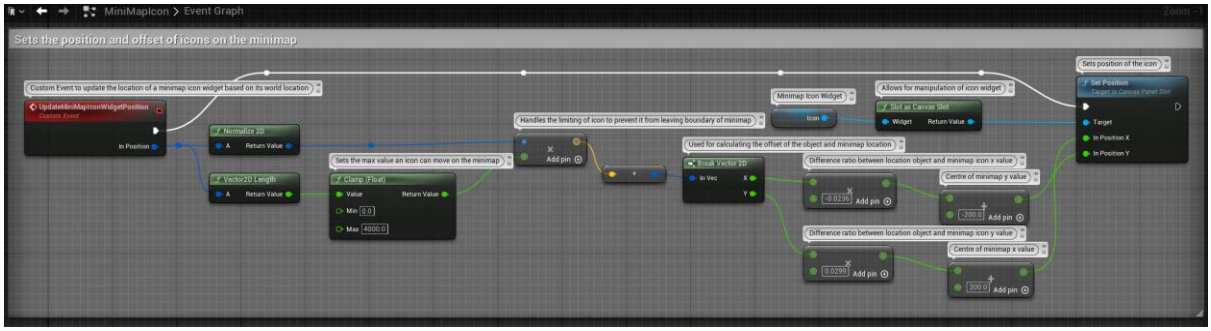


*Figure 48 UpdateMiniMapIconWidgetPosition Event Blueprint*

To determine the difference ratio between a `MiniMapIconLocation` actors' location in the environment and the location on the minimap of its corresponding `MiniMapIcon` widget, I needed to use a reference point. As mentioned previously, the centre of the minimap corresponds to the spawn location of the player character, which is set at an environment position of x: 0 and y: 0. I then placed a `MiniMapIconLocation` actor inside the centre of Parochial House object. Its environment coordinates were x: 2158.153346 y: 2200.530248. These are the offset values I needed.

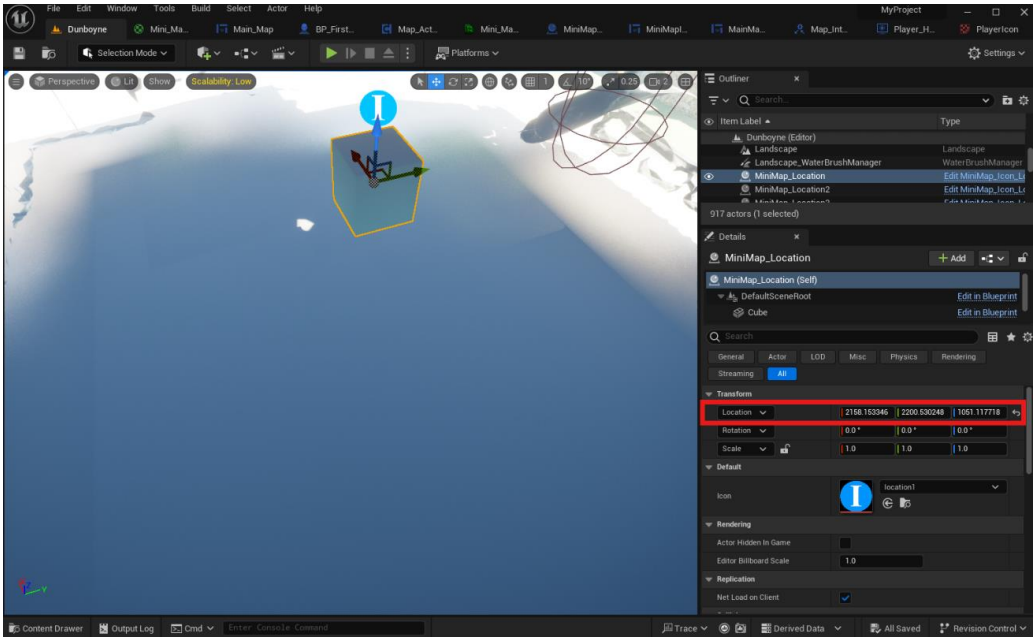


*Figure 49 MiniMapLocation Actor Inside Parochial House Object*

With these environment coordinates to hand, I then went to the `Mini_Map_Widget` class and moved an icon widget to the position of the `MiniMapIconLocation` reference actor inside the Parochial House. The coordinates on the `Mini_Map_Widget` of this icon were x: 266.0 and y: -251.081055. The centre chevron of the `Mini_Map_Widget` is x: 200.0 and y: -200.

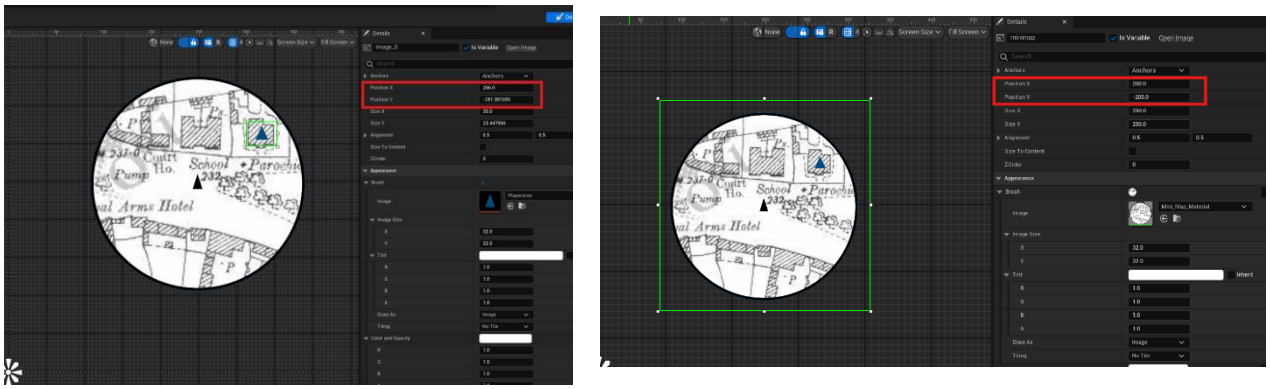

*Figure 50 Values of widget location on the Mini_Map_Widget and the Values of the Centre of the Minimap*

Using these values, I could calculate the correct ratio to set for the `MiniMapIcon` widgets. The distance between the x value of the centre of the minimap and the icon on the minimap was 66 (266 – 200 = 66). The distance between the y value of the centre of the minimap and the icon was -51.081055 (-251.081055 - -200 = -51.081055).

I then divided this difference with the offset values retrieved from the test in *Figure 49*, so 66 / 2200.530248 = 0.0299, and -51.081055 / 2158.153346 = -0.0236. These values were then used to properly set the ratio between the `MiniMapIcon` widgets on the minimap and their corresponding `MiniMapIconLocation` actors in the environment.

### 2.3.7.2. Main Map System

With a minimap system in place, I wanted to allow the user to get a better view of the entire environment map with a larger, main map they could view on their screen by pressing the 'M' key. Using this main map, they could then place a custom waypoint on the map that they could travel to. To achieve this, I first created a new widget called `MainMap` and set the image in this widget as the ordnance survey map of Dunboyne from *Figure 42*. Next, I created an interface called `MapInterface` that contains the function `MapToggle`, which consists of a Boolean. This Boolean is used in the event `MapToggle` in the `PlayerHUD` class, which checks to see if the `MapToggle` Boolean has been set to true by the user by pressing 'M' on their keyboard. If `MapToggle` is set to false, the map should be removed from the user's screen. Then, a for-each loop will remove every one of the `MiniMapIcon` widgets from an array of icons. Once this is done, the users game mode will be set to `Game` mode and the mouse cursor will be removed from their screen. If `MapToggle` is true, the `MainMap` widget will be created and added to the user's screen. Then, the `UpdateMainMapIcons` collapsed graph will be run.

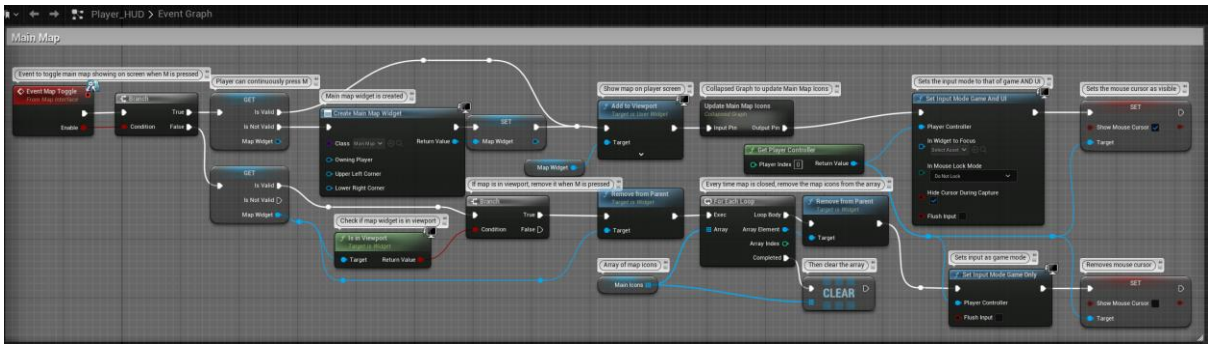

*Figure 51 MapToggle event in PlayerHUD Blueprint*

The UpdateMainMapIcons collapsed graph will retrieve every actor that has the MainMapIconInterface implemented. These actors are the player character and all the MiniMapIconLocation actors. A for-each loop is run that will create a MainMapIcon widget for every actor with the MainMapIconInterface interface. It will then run the function GetMainMapIconData that retrieves the Location, Icon widget and Rotation for each actor. This function is used in both the MiniMapIconLocation actors and the player character actors. Once the GetMainMapIconData function is run, the output of this function is input into the CreateMainMapIcon event which is housed in the MainMapIcon widget class. The result of this event is then added to the players screen and the icons for each actor are added to an array of icons.

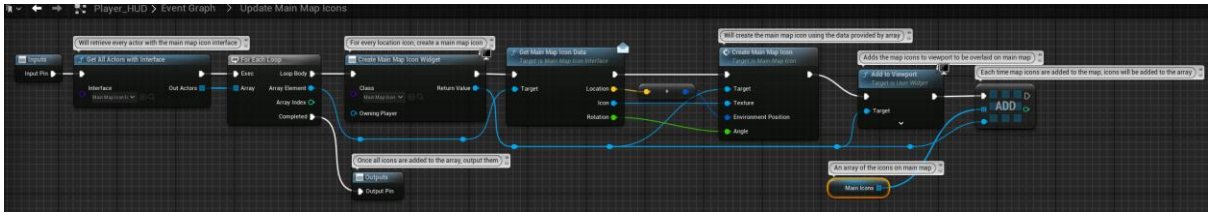

*Figure 52 Update Main Map Icons Collapsed Graph Blueprint*

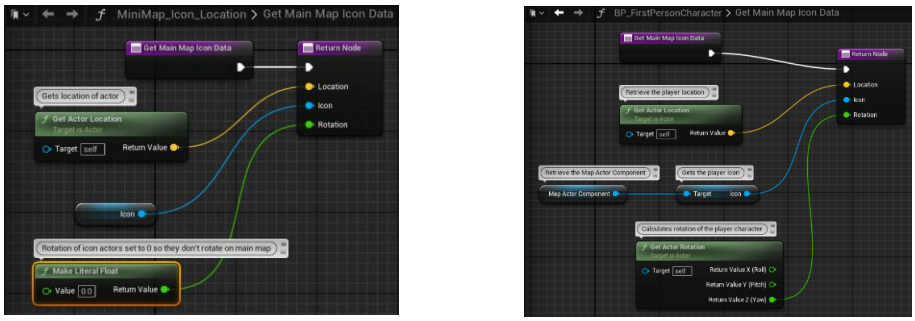

*Figure 53 Get MainMapIconData functions in MiniMapIconLocation and BP_FirstPersonCharacter Blueprints*

The CreateMainMapIcon event in the MainMapIcon widget class works by setting the texture, environment position and angle parameters of an icon widget on the MainMap widget. The position of the icons are run through a ratio and offset first to ensure that the position of the MainMapIcon widgets on the MainMap align with the position of their corresponding MiniMapIconLocation actors in the environment. This ratio and offset was calculated in the same manner as the ratio and offset for the minimap icons discussed in *Figure 50*. The values of the ratio and offset on the MainMap are different to the values used on the Mini_Map_Widget due to the fact that the MainMap is larger.

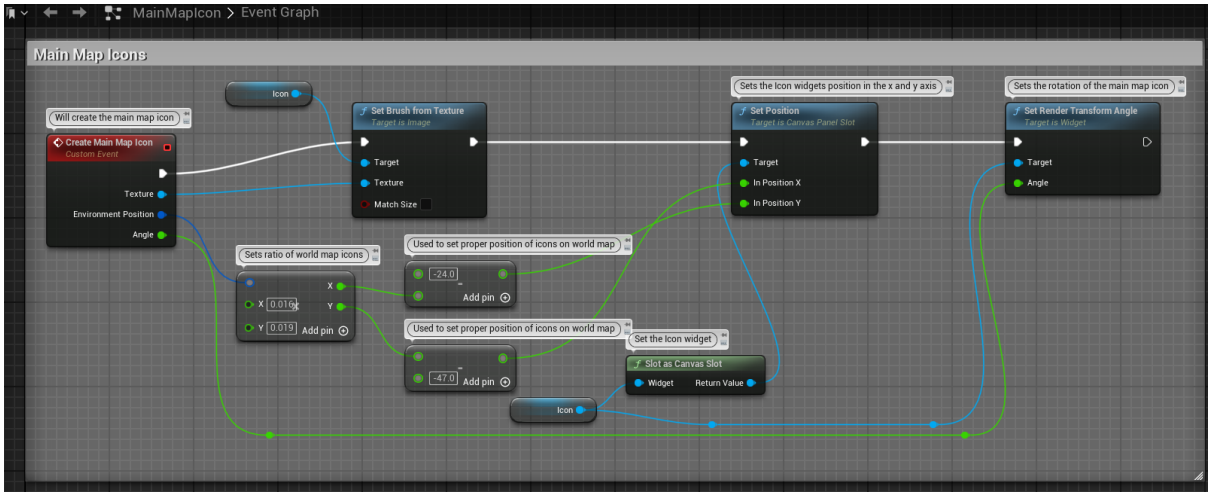


*Figure 54 CreateMainMapIcon Event in MainMapIcon Blueprint*

With the icons for the static `MiniMapIconLocation` location objects functioning, I added the ability for a user to add their own custom waypoints to the map. I achieved this by creating a child class of the `MiniMapIconLocation` actor – called `CustomMiniMapIconLocation`. I then added two functions to the `MainMap` widget class. The first function `OnMouseClick` occurs when a user clicks anywhere on the `MainMap` widget. This will run the `MousePositionWorld` function that converts the coordinates of the spot that the user clicked on the `MainMap` widget to the corresponding environment coordinates, which is then used to spawn a `MiniMapLocation` actor. This function works by retrieving the x and y coordinates of the players mouse position on the screen with a `GetMousePositionScaledByDPI` node. The function then retrieves the users screen size and scale, so that the calculation will work for any screen size. The mouse position and viewport size and scale are subtracted from one another and then converted to a 2D vector. The x and y co-ordinates are then multiplied by 100 to properly match the environment co-ordinates. This result is then output back out to the `OnMouseClick` function.

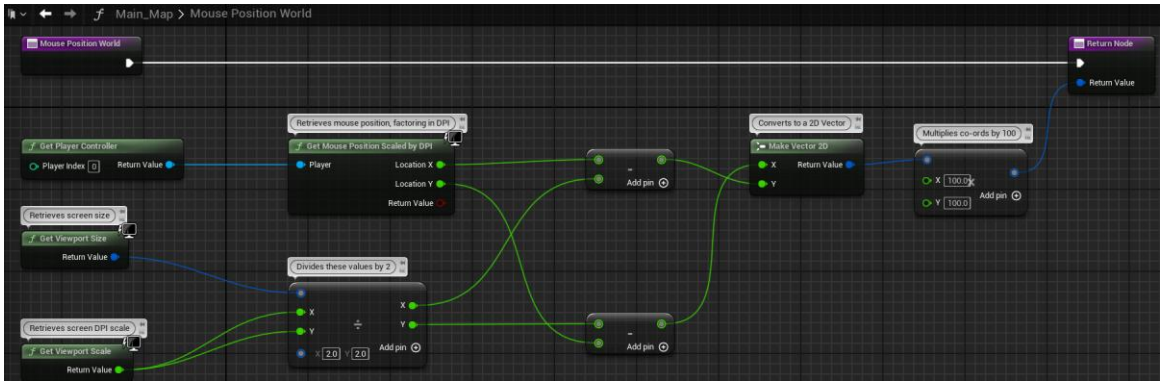


*Figure 55 MousePositionWorld Function in MainMap Blueprint*

Back in the `OnMouseClick` function, all actors of the `CustomMiniMapLocation` class are retrieved. The value output by the `MousePositionWorld` function is converted from a 2D vector to a vector, which is used for environment location coordinates. This vector is then converted into a 'transform' that is plugged into a `SpawnActor` node. This will spawn a `CustomMiniMapLocation` actor at the coordinates the user clicked on the `MainMap` widget. Once this has happens, the `MapToggle` Boolean will be set to true, allowing for the `MapToggle` event in the PlayerHUD class to trigger, adding a custom icon to the map.

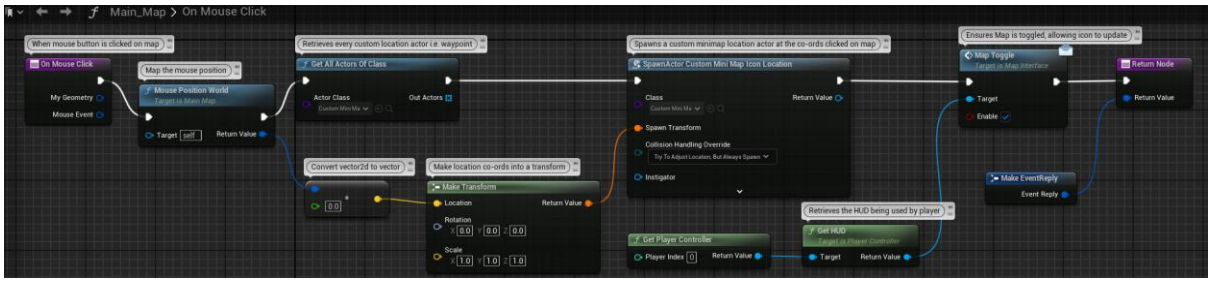


*Figure 56 OnMouseClick Function in MainMap Widget Blueprint*

### 2.3.8. Creating the level

To authentically recreate the Dunboyne of 1909, I needed to have an accurate reference point to work from. I wanted to ensure that the applications map was to-scale, and that the layout of the town was as accurate as possible. To do this, I used the tool 'Unreal Mapbox Bridge' to capture a section of Dunboyne on a map. This tool creates a 16-bit heightmap image of the selected area. I set the Unreal Engine map size as 2017x2017, and I drew a box around the area I wanted to capture that was 3km x 3km. I input the co-ordinates: latitude 53.41952 and longitude -6.47622. This would be the centre of the map. Mapbox Bridge exported the selected section of the map as 16-bit heightmap data.

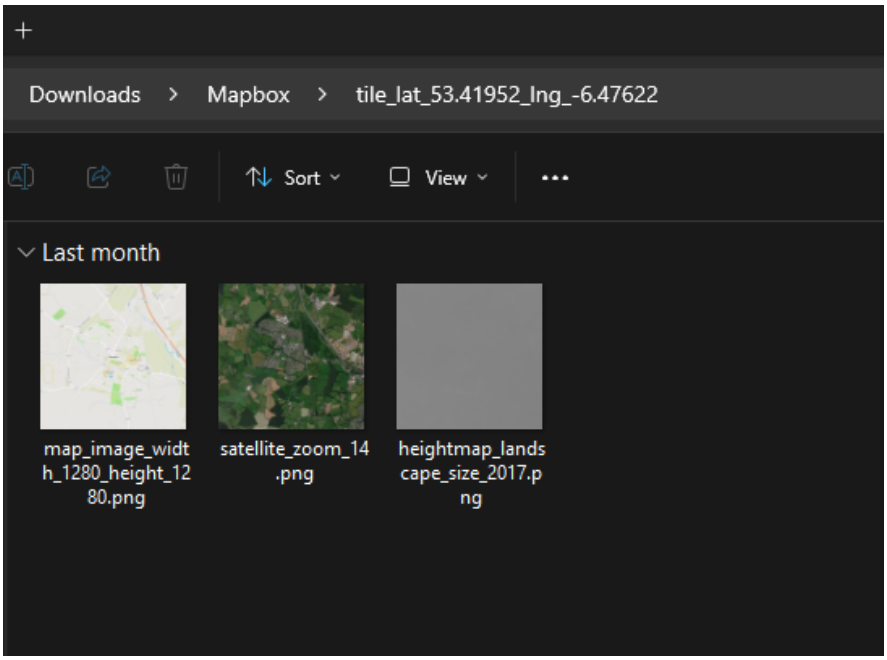


*Figure 57 Files that Unreal Mapbox Bridge created*

I then imported this heightmap file into a new level in Unreal Engine - which created the landscape.

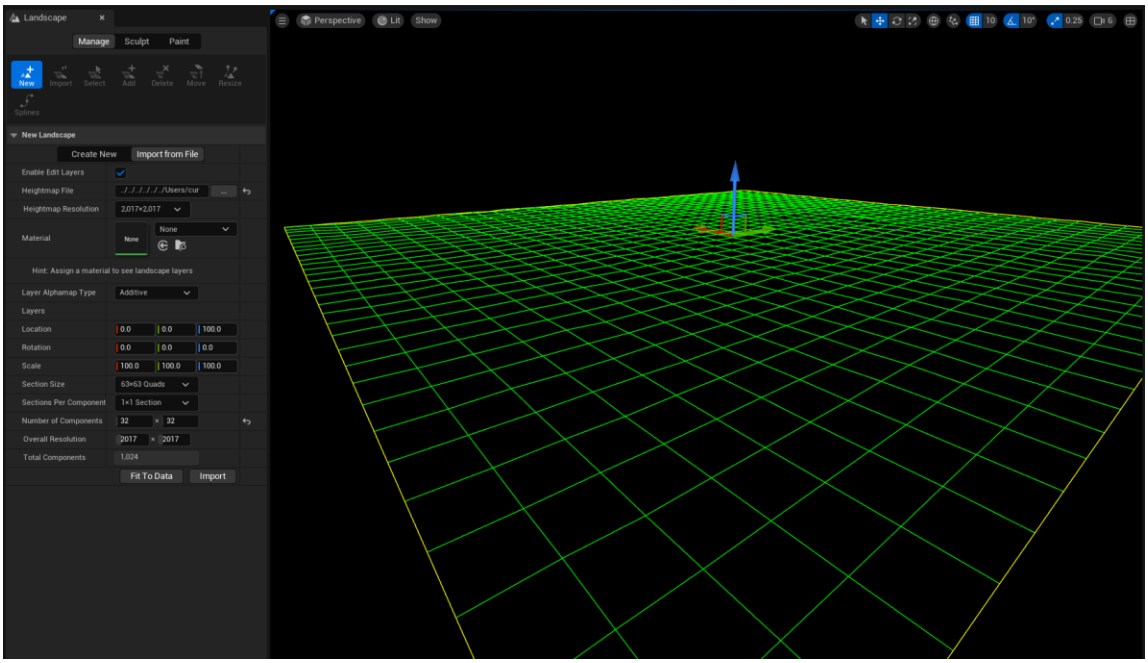


*Figure 58 Importing the Unreal MapBox Bridge heightmap to create an Unreal Engine landscape*

Once I created the landscape, I added a directional light to represent the sun and a sky atmosphere to act as a skybox. Exponential Height Fog was also added to simulate the haze that is naturally present on the horizon in the real world.

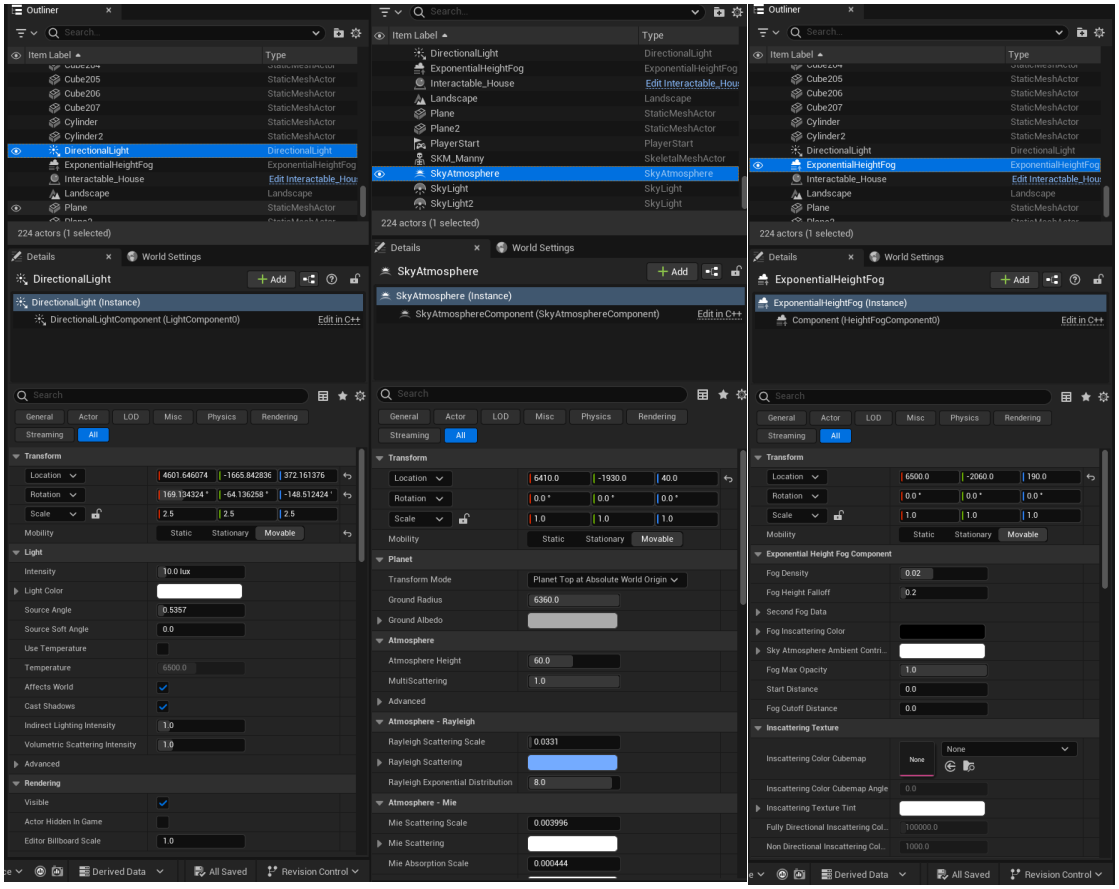

*Figure 59 Directional light, Sky Atmosphere and Exponential Height Fog added to project*

After lighting the environment, I needed to be able to accurately place the buildings in the town and ensure that everything was properly to scale. Unreal Mapbox Bridge also exported a .png of the satellite data in addition to the heightmap data, which I could use as a reference. On the Geohive ordnance survey site, I input the coordinates as: latitude 53.41952 and longitude -6.47622. These were the same as I had input in Unreal Mapbox Bridge.


*Figure 60 Geohive 1909 ordnance survey of Dunboyne*

Now that I had the Unreal Mapbox Bridge image and the Geohive image, I used image editing software to overlay the two images on top of each other, ensuring that they were both aligned and to-scale.

*Figure 61 Unreal Mapbox Bridge and Geohive maps combined*

I then imported this combined image as a texture sample into Unreal Engine and overlaid it onto the landscape that the heightmap had created.

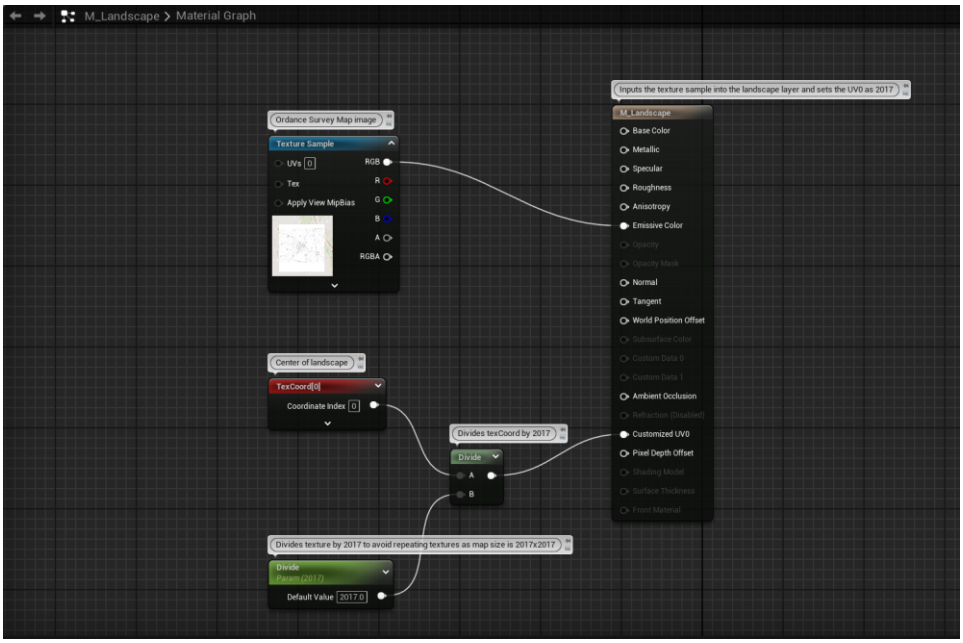


*Figure 62 Material created to texture landscape code snippet*

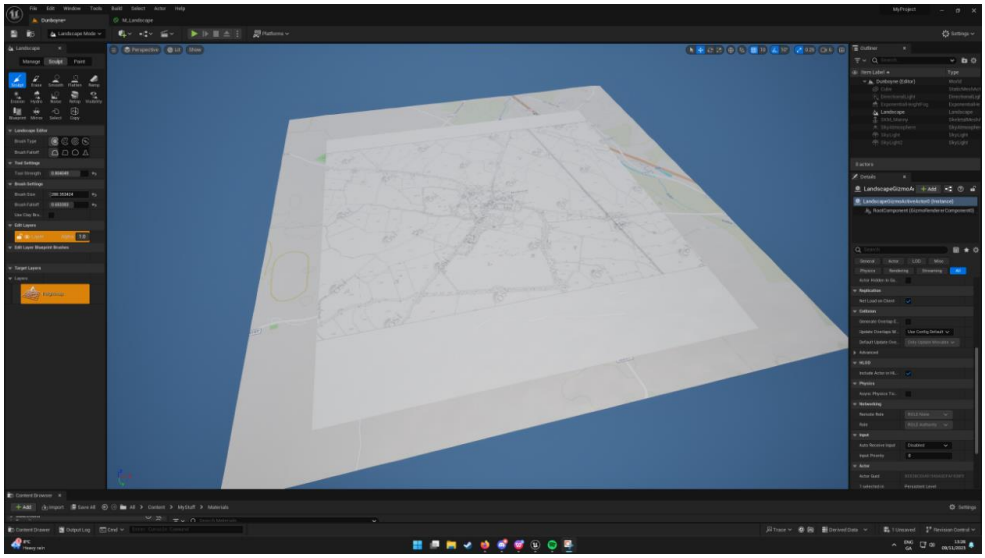

*Figure 63 Texture overlaid onto the landscape*

Once I had this landscape completed, I could use it as a reference to begin creating placeholder buildings to get an idea of the layout of Dunboyne from 1909. This process of 'whiteboxing' was done by placing down cube objects and re-sizing them to fit the outlines of the buildings that were present during the 1909 ordnance survey. In future development, I can then replace these placeholders with finished assets. To showcase how these art assets will be created, I completed a scan of the Parochial House building, as will be highlighted in the section 2.3.9.  going over the use of photogrammetry in the application.

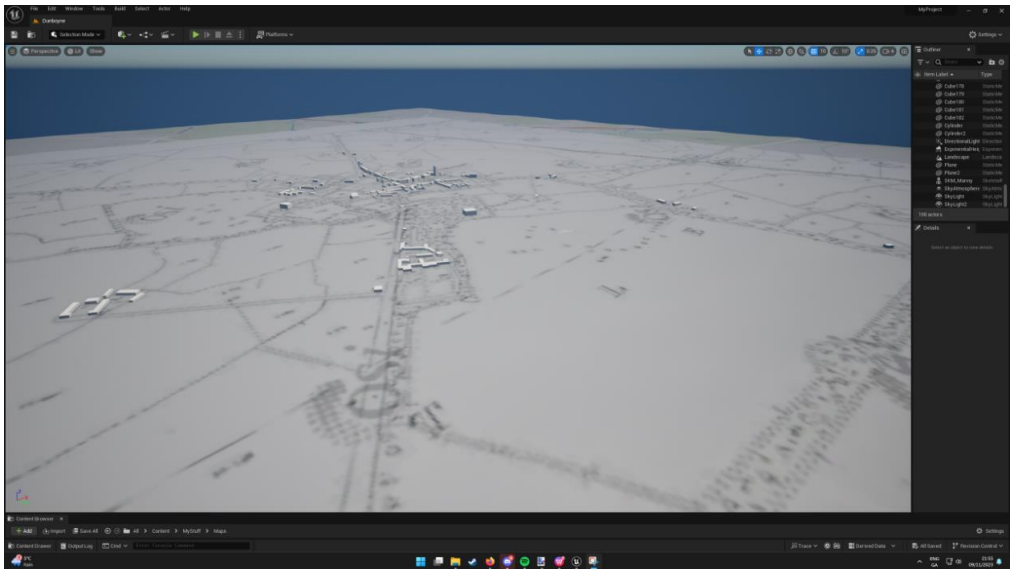

*Figure 64 Placeholder buildings placed on the landscape*

With the buildings of the map ‘whiteboxed’, I then moved onto placing the foliage assets of the project. These foliage assets were downloaded from Unreal Engine’s ‘Megascan’ [17] asset library, a repository of free-to-use 3D and 2D assets. I added all the tree assets I wanted to use into the project and enabled ‘Nanite’ [18] on them. Nanite functions by dynamically changing the number of polygons used in a 3D asset based on the distance the player character is from that object. This ensures that high-polygon assets can be used with minimal drops in performance. Because I would be placing so many foliage assets, it made sense to utilise Nanite to improve the performance of the application.

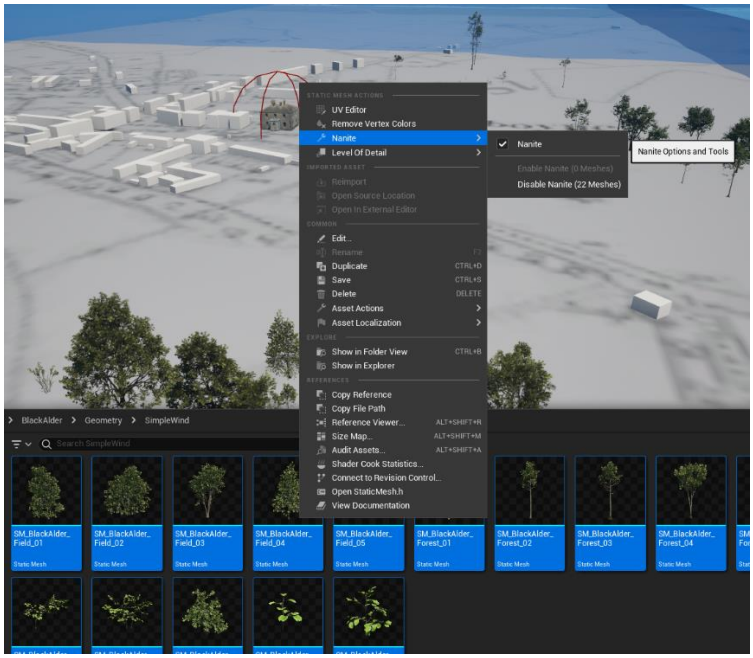


*Figure 65 Enabling Nanite on Foliage Assets*

After this, I then placed the tree assets in the environment, with careful attention paid to ensure that they were in period-accurate locations and were the correct size based on their lifespan. This was achieved be constantly referencing back to the ordnance survey map from 1909 and historical photography from the period the project is set.

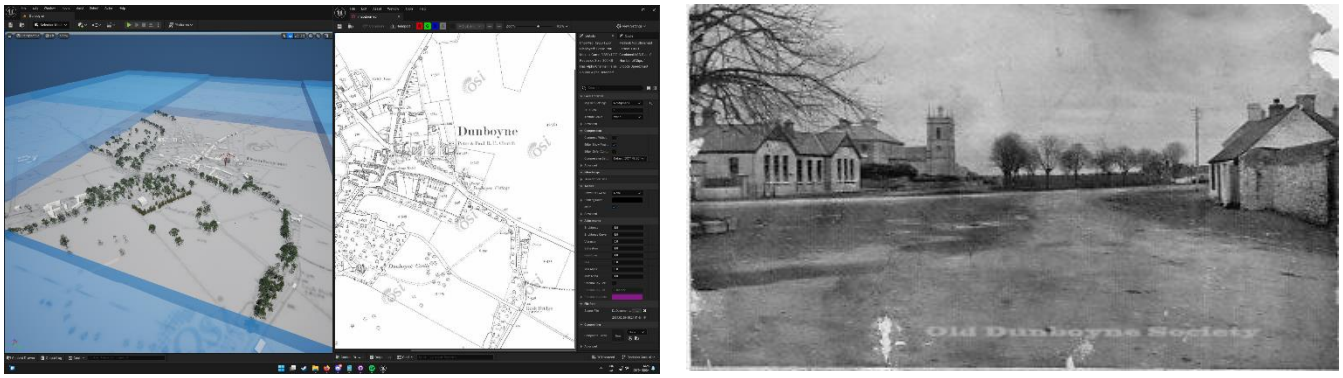


*Figure 66 Placement of trees with Map and Historical photograph being referenced*

After placing the trees in the environment, I then placed walls, fences, and hedgerows – paying close attention to ensure that they matched the historical photography available. When these were placed, I then began the process of texturing the environment to ensure that the recreation looked more immersive and realistic. By using multiple Megascan texture assets that comprise of captures of real-world textures such as grass, mud, and leaves, I "painted" textures onto the environment, using the ordnance survey map as a guideline to ensure an authentic representation of the town.

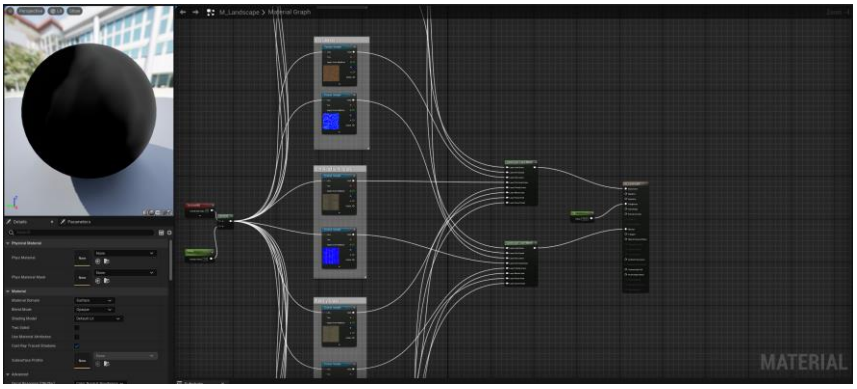


*Figure 67 Combining the various Megascan assets into a paintable texture*

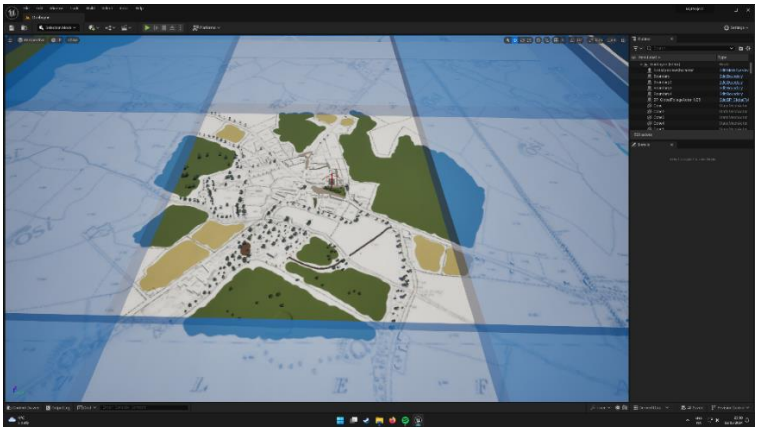

*Figure 68 "Painting" of the landscape with Megascan textures in progress*

After the textures were placed, I then added the final foliage assets in the form of grass, wheat, and undergrowth in the forested areas of the environment. These served to add an extra level of fidelity and immersion to the visual aspects of the application. I also added water to the river Boyne that runs through the town using the 'Water' plugin in Unreal Engine.

*Figure 69 Screenshot of Dunboyne environment as of the project's completion*

### 2.3.9. Photogrammetry

A key aspect of my project is the use of photogrammetry to create assets for the environment. This technique is used to increase the overall fidelity of the application as well as to act as a form of historical preservation. Buildings that have changed little since 1909 are ideal candidates for the technique and one of the key buildings I identified was the Parochial House that was present in historical photographs of the period.

I contacted the local parish priest Fr. O'Connor and explained my project, requesting permission to capture the building using drone photography - to which he agreed. My father is a licensed drone pilot with the Irish Aviation Authority, so he controlled the drone while I described what needed to be done for the photogrammetric process. The drone was flown in an orbit around the Parochial House, with pictures being taken every few seconds. The drone was then lowered in altitude and the process was repeated until a full 360-degree capture of the building was completed.

When these photographs were captured, I uploaded them to my PC and did a quick quality control of the images, only keeping ones that were usable by virtue of not having any sections of the building cut off.

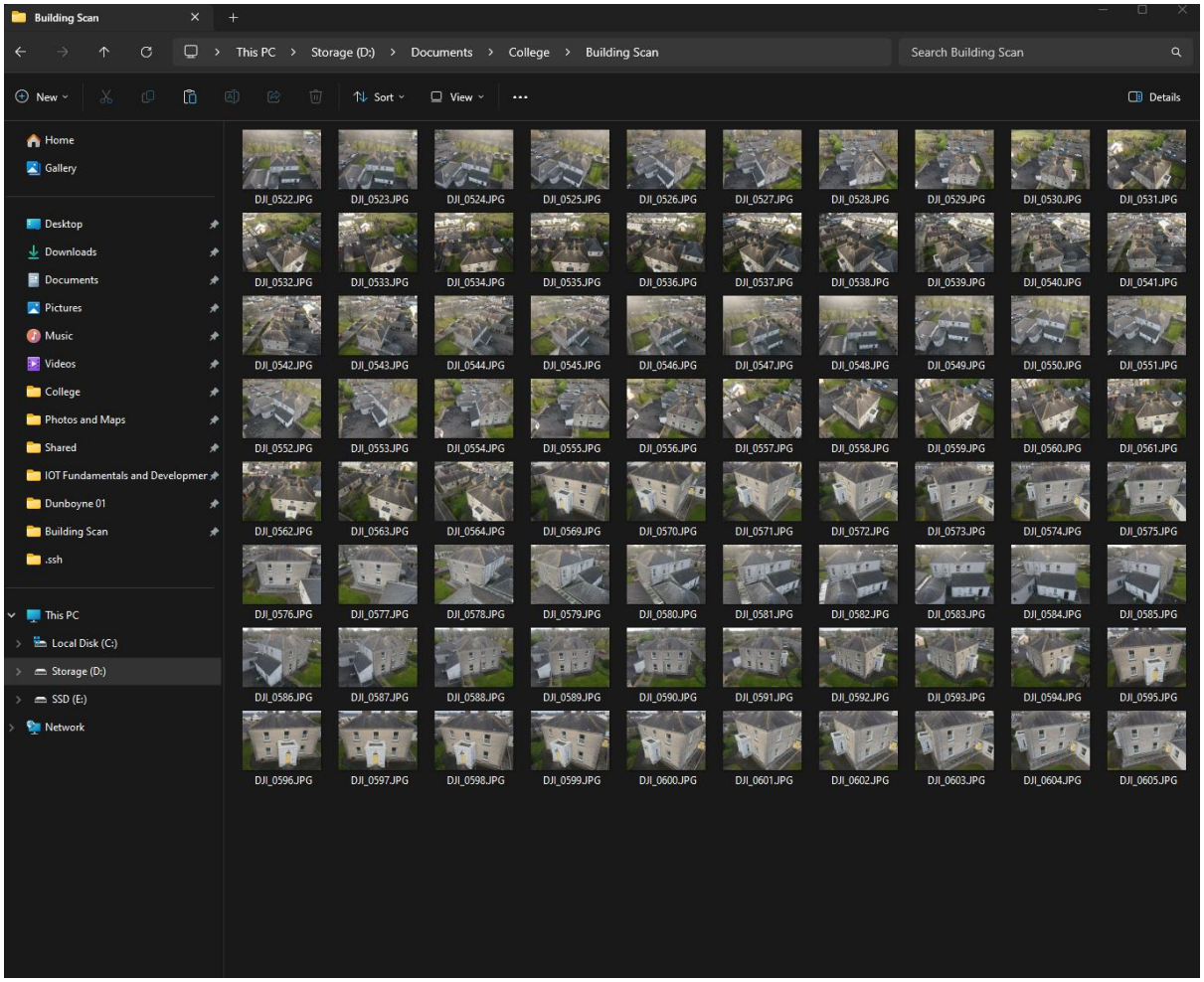


*Figure 70 Images of the Parochial House captured using drone*

I then uploaded these images from the drone to Polycam and set the details setting to ‘Full,’ as I wanted to create the highest quality 3-D model possible.

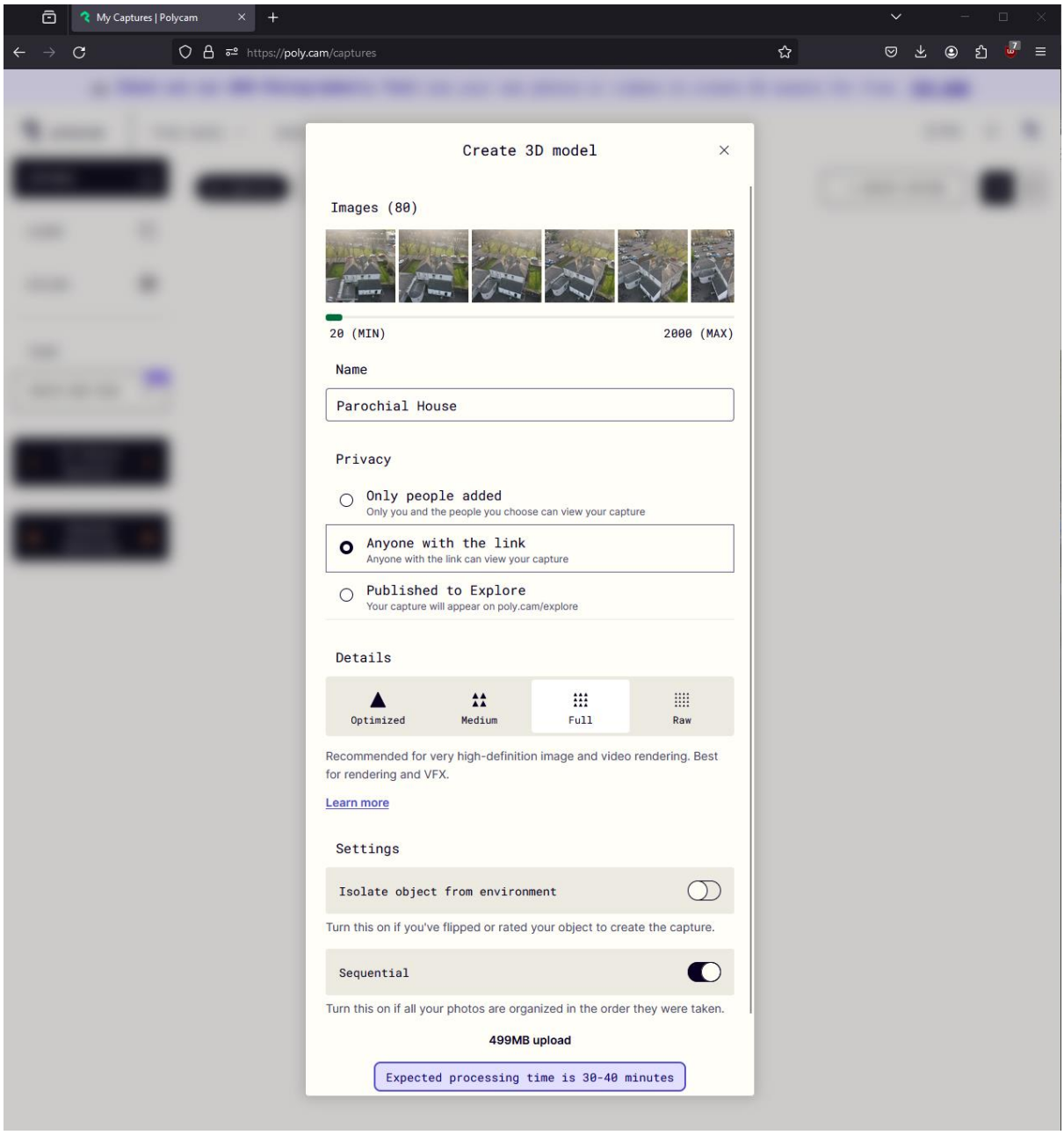


*Figure 71 Inputting images of the Parochial House into Polycam*

After around an hour of processing, Polycam finished rendering the 3-D model - and this was the result:

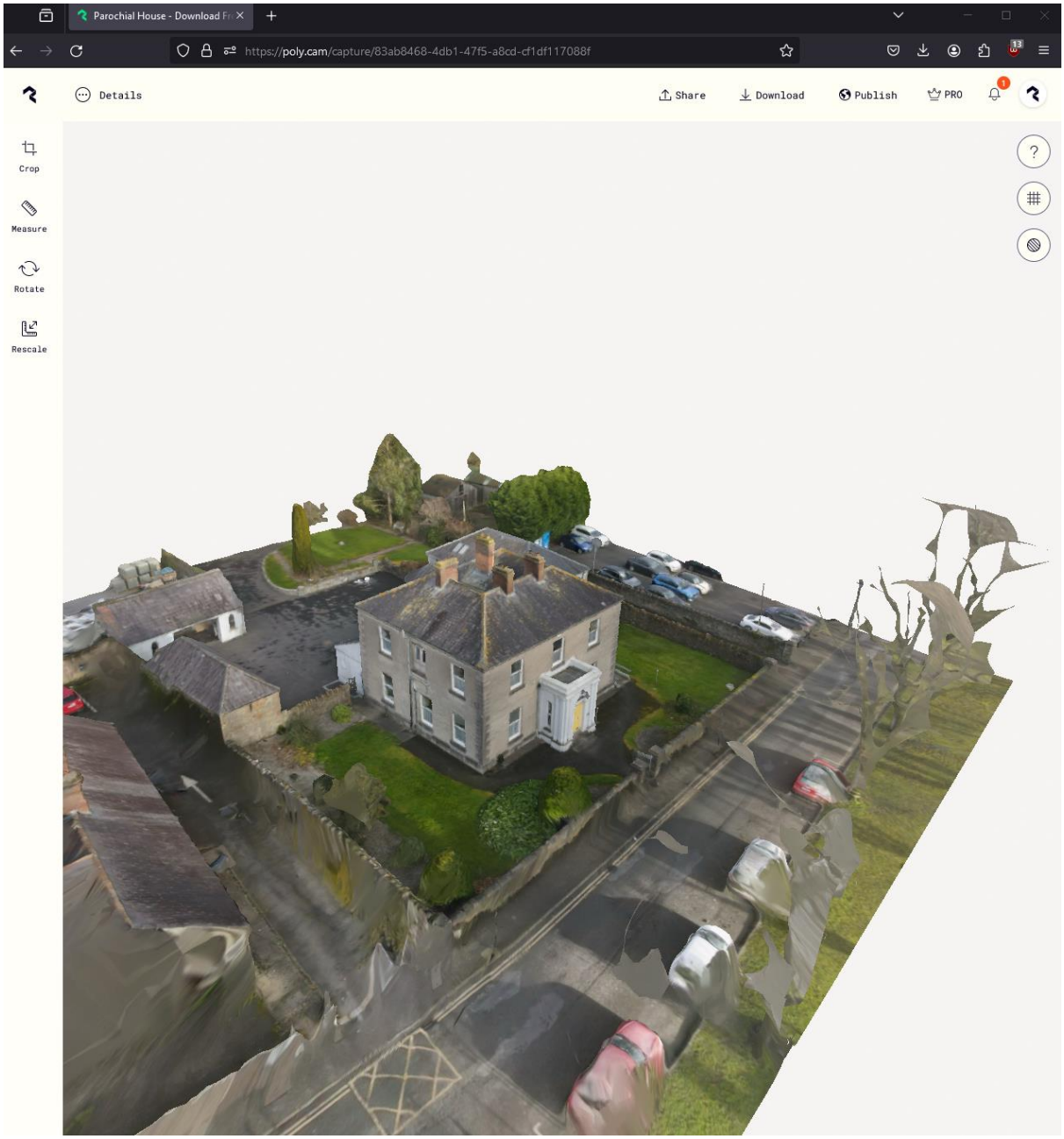


*Figure 72 3-D model of the Parochial House generated in Polycam*

As you can see in *Figure 72*, every detail that was captured in the photographs was added to the model, but the areas that appeared most frequently in the images such as the house itself, are of much higher fidelity than the surrounding areas.

I exported the 3-D model that Polycam created as a .gltf/glb file and imported this file into Blender.

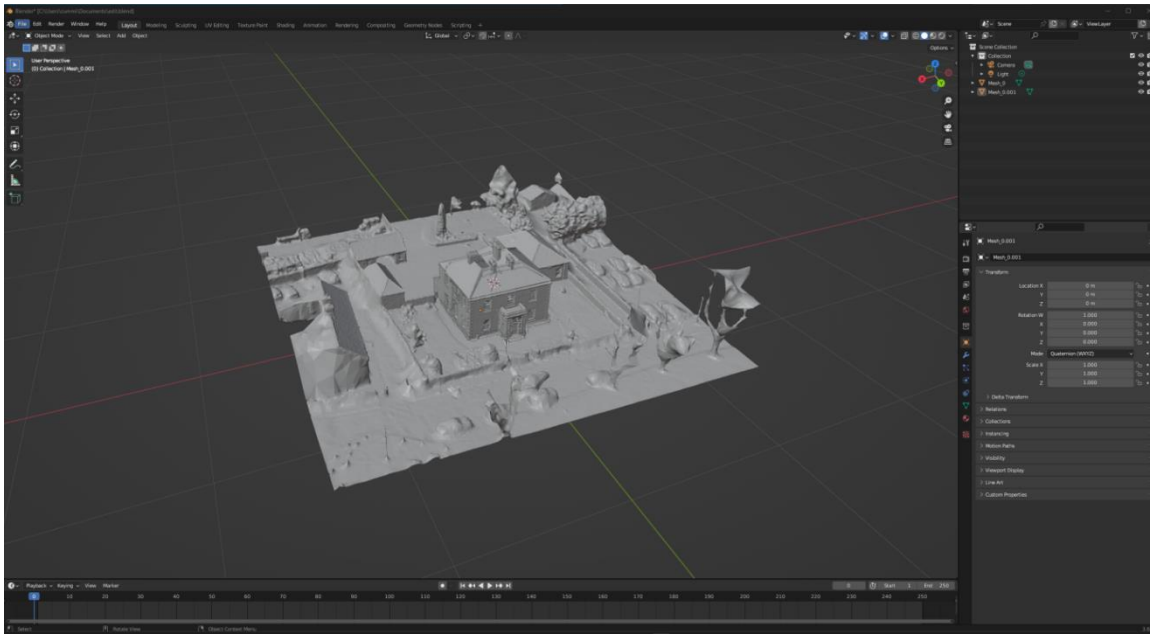

*Figure 73 Polycam model imported into Blender*

Because I only want the building as it existed in 1909, I needed to edit the 3-D model and remove unwanted vertices such as the surrounding road, trees, walls, and the other buildings that have been added onto the property as extensions since 1909.

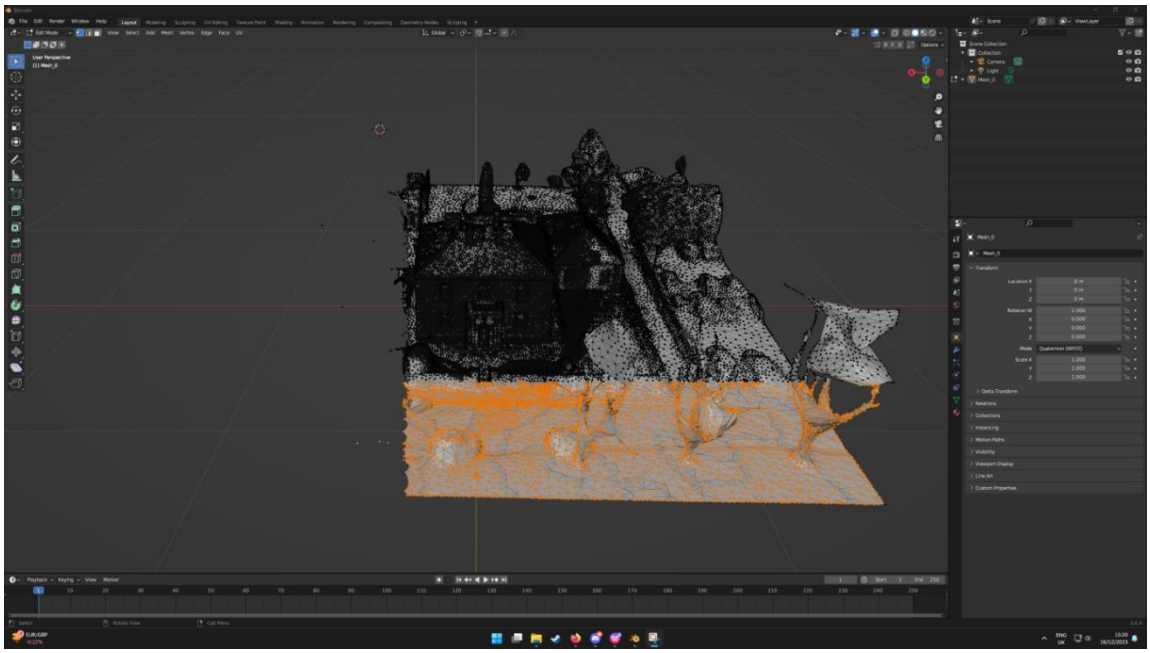

*Figure 74 Removing unwanted vertices from the model in Blender*

Once this edit was finished, I had the final 3-d model of the Parochial House.

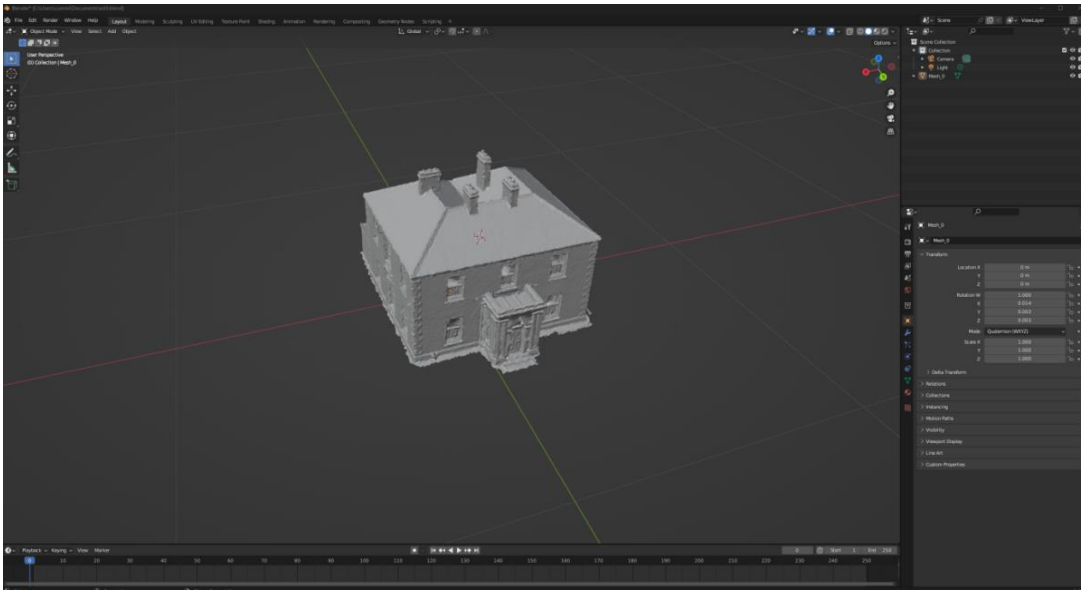

*Figure 75 Final 3-D model in Blender after editing*

This 3-D model was then exported as a .gltf/glb file and was imported into Unreal Engine 5 as an asset.

*Figure 76 Parochial House model placed in the applications environment*

## 2.4. Graphical User Interface (GUI)

All GUI's are currently placeholders and their colour-scheme and overall design are subject to change as development progresses.

### 2.4.1. Main Menu

In the main menu level of the application, the image below is the widget that appears when a user starts the application. It contains an image of the environment and the interactable menu itself. A user can start the application by clicking the 'start' button. They can access the settings for the application by clicking the 'settings' button, and finally they can quit the application by clicking the 'quit' button.


*Figure 77 The main menu and settings submenu of the application*

### 2.4.2. Controls Menu displayed upon Start

When a user clicks the 'start' button on the main menu, the application loads the Dunboyne level. When this loading has been completed and the user's character is spawned into the world - the application will pause and display a widget that explains the control scheme for the application.

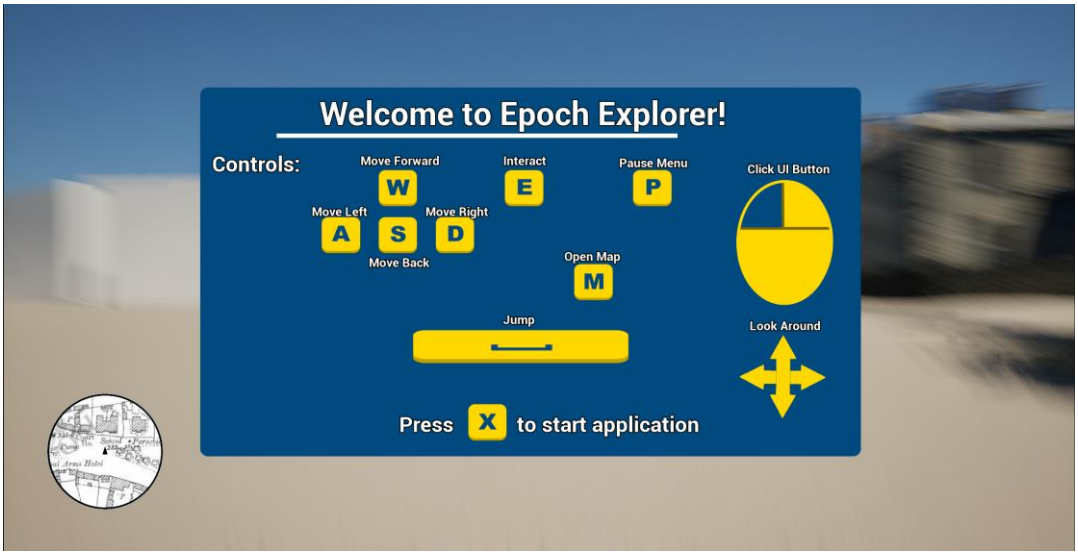


*Figure 78 Control scheme that appears on user's screen upon startup*

### 2.4.3. Pause Menu

When the user is interacting with the application, if they press the ‘P’ button, the application will pause and display a widget that shows the user the pause menu on their screen. From this pause menu the user can click ‘Resume’ to un-pause the application and remove the widget from their screen. They can click ‘Save’ and ‘Load’ to save their progress or load a save file. They can click ‘Return to main menu’ to return to the main menu, and finally they can click ‘Quit Application’ to quit the application entirely.


*Figure 79 Pause menu as it appears in the application*

### 2.4.4. Minimap

When the user spawns, the minimap is visible on the bottom-left corner of their screen. This map updates in real-time with the player movement and the icons that represent locations of interest also update and move around as the player navigates the environment.

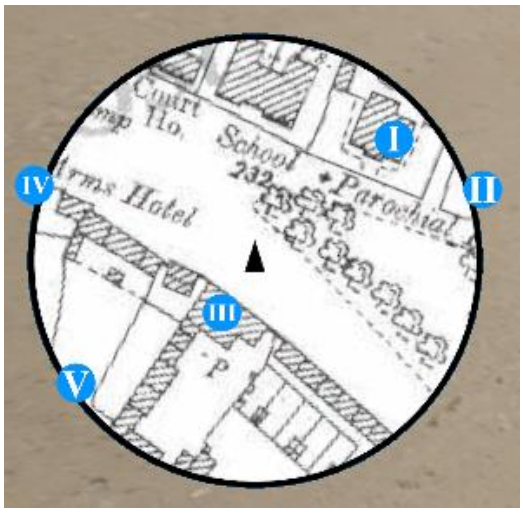


*Figure 80 Minimap with locations of interest*

### 2.4.5. Main Map

When the user presses the ‘M’ key, the main map will appear on their screen. This shows the entire area of the explorable environment and features locations of interest. By clicking their mouse cursor anywhere on the map, the user can place a custom waypoint that they can use as a navigational aid.


*Figure 81 Main map with custom waypoint and locations of interest*

## 2.5 Testing and Evaluation

### 2.5.1 Functional/Unit Tests

Throughout the development process, I conducted a series of tests for the Epoch Explorer application. I performed unit tests for the 'Out of Bounds' feature and the 'Object Interaction' feature. To conduct the tests, I created a folder in my project called 'Testing' and copied the 'Dunboyne' level into this folder. I deleted all the assets that were not required for testing to improve the load times of the tests, such as buildings and foliage and renamed this level to `FTEST_FunctionalTests`. Inside this level, I then created a new game mode called `TestingGameMode` and set this as the default game mode for the level.

With the testing level created, I then began to design unit tests for each of the features I was testing. I will first explain the test for the 'Out of Bounds' system. I created a new test blueprint called `BoundaryTest`. I spawned a `BP_FirstPersonCharacter` into the level which will represent the player character. In the event graph for the test, I added a `AddMovementInput` node which will move the character in a given direction. I input a reference to the `BP_FirstPersonCharacter` and set the `AddMovementInput` as `GetActorForwardVector`. This setup will move the character forward every frame.

I then added an `OnActorBeginBverlap` node for the `Boundary` object. This will check if the character is overlapping or touching the boundary, which will display the out of bounds message. If this is the case, the test will finish successfully.

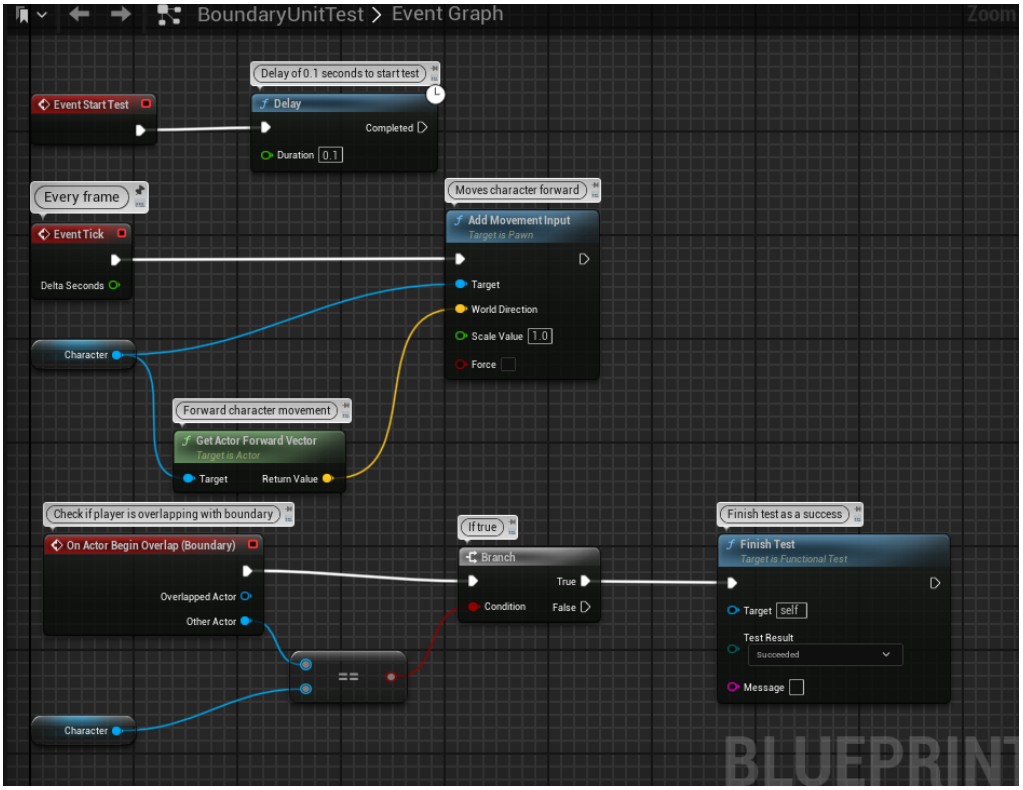


*Figure 82 Boundary System Unit Test Blueprint*

Next, I added the test for the 'Object Interaction' system. This tests to ensure that the 'Interaction' prompt appears on the user's screen when they reach the collision sphere of an object of interest. For this test, I also set the character to move forward using the same nodes. This time however, an object with interaction functionality is placed into the testing level rather than a boundary object. The application will check to ensure that when the character overlaps with the object's collision sphere, the interaction widget appears. If this is the case, the test will finish successfully.

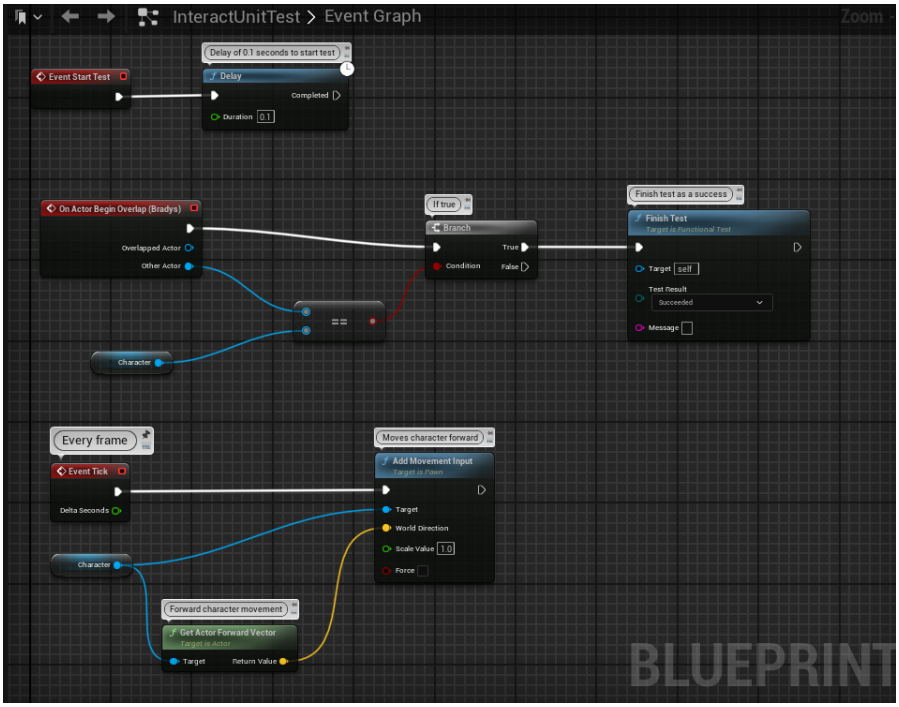


*Figure 83 Interaction Unit Test Blueprint*

The tests are run through the 'automation' section of Unreal Engine's 'Session Frontend' tool. From the results of the unit tests, you can see that both tests passed successfully.

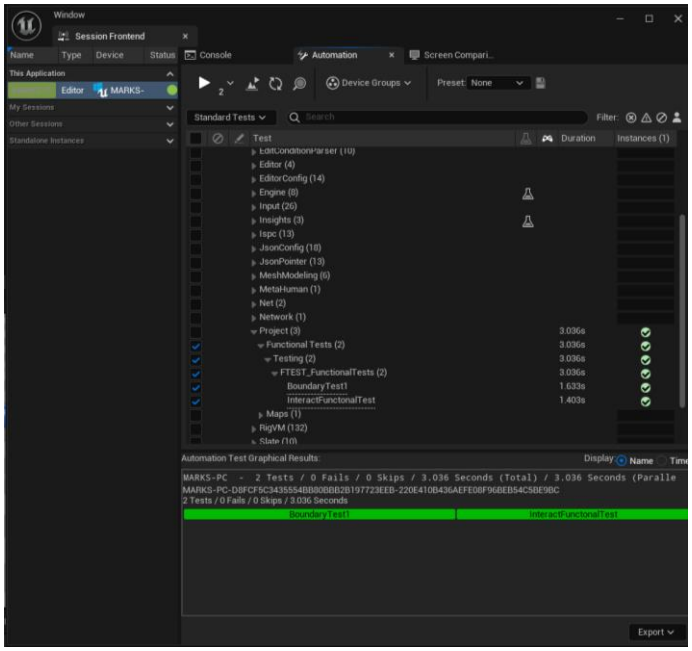


*Figure 84 Unit tests Passing*

In addition to these tests I created, I also ran a suite of tests that the Unreal Engine provides to ensure that my project meets the minimum standard of quality for an Unreal application. These tests ensured that the project does not have any errors or bugs in the code before it is 'packaged' and made available to users. As you can see, all these tests passed successfully.

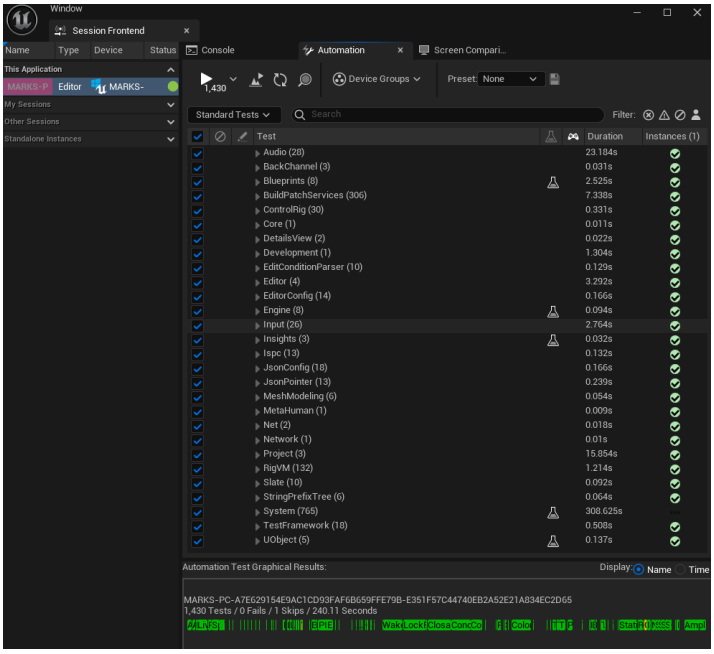


*Figure 85 Suite of Unreal Engine tests Passing*

### 2.5.2 Manual Tests

In conjunction with the unit tests, I also conducted a series of manual tests of the application throughout its development. A key aspect of my application is ensuring that it is easy to use and runs on a wide variety of hardware while maintaining a minimum level of performance. To achieve this, I monitored the performance of the application by typing the `StatStreaming` command in the Unreal Engine command line interface. This provides a live view of a variety of stats pertaining to the performance of the application such as the current frames per second and memory usage.

By using this to test my environment, I found out that all the textures I was using was exceeding the 'Streaming Pool' budget. The streaming pool is a memory repository of all the textures in the level and has a capacity of 1GB. If the application is running while the streaming pool is out of budget, this can result in the stuttering or freezing of the application as textures are added and removed from the pool.

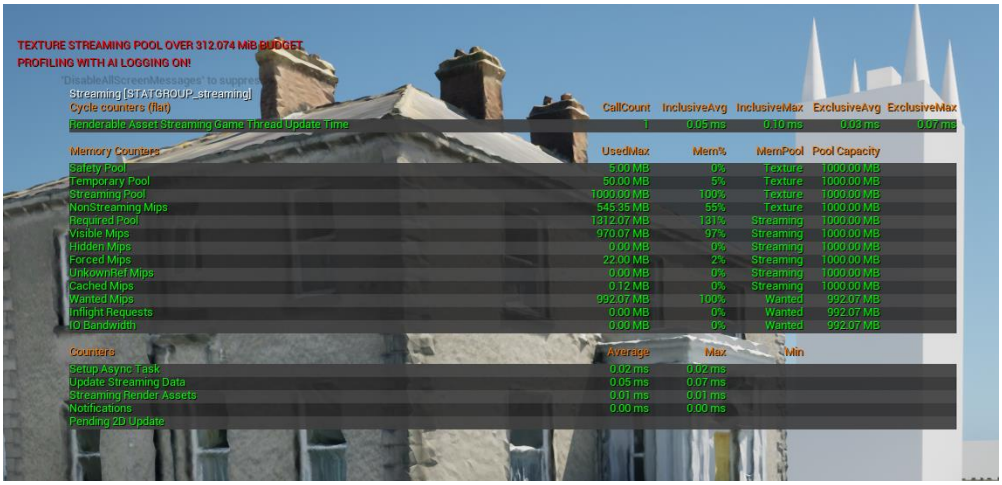


*Figure 86 StatStreaming showing memory statistics*

In addition to the streaming pool problem, my tests highlighted an issue with the fidelity of the foliage assets I had added. When I was not close to the assets, my frames per second were close to 60, which is the ideal target for my application. However, when near a tree, the FPS would drop to below 30, which was undesirable as it would impact a user's experience of the application negatively.


*Figure 87 FPS counter when far away and close to a tree (enlarged for readability)*

To solve this problem encountered by my tests, I decided to reduce the size of the textures of a variety of assets in the environment. As standard, Megascan assets come with a resolution of 4096x4096. This is a very high level of fidelity; however, it requires the latest high-end graphics cards to utilise in an environment without negatively impacting performance. In addition, this is such a high level of fidelity for assets that a user is unlikely to examine up-close, so using valuable streaming memory to store such large textures didn't make sense. Because of this, I changed the size of the textures to 2048x2048 instead. This is still a high-resolution texture, but as you can see it cut the size of a single texture from 16MB to 4MB. As well as the savings in memory, the fps also increased drastically.

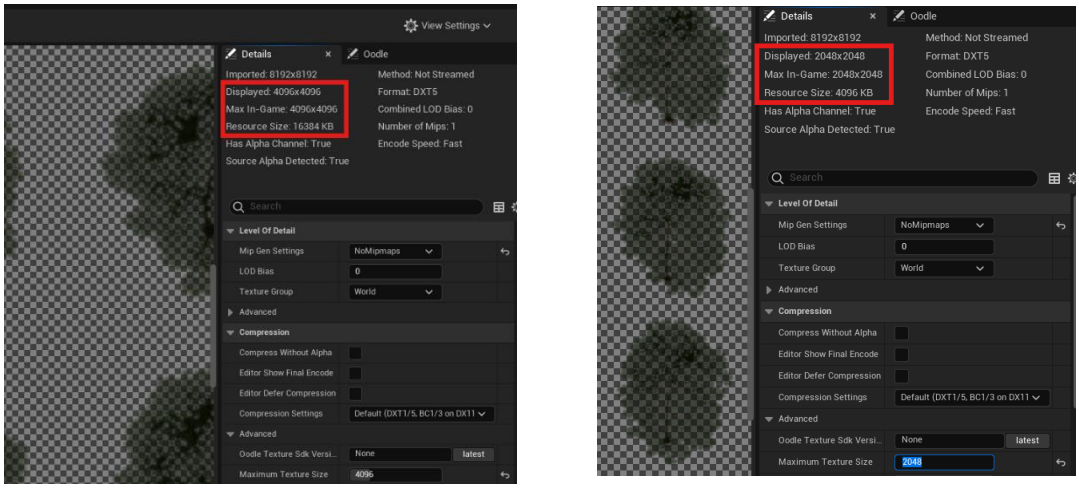


*Figure 88 Changing texture size to improve performance*


*Figure 89 Improved FPS while close to tree after changing texture size*

# 3.0 Conclusions

The development of the Epoch Explorer application has been driven by my passion for history, my enjoyment of video games and my desire to provide others with a more engaging and immersive educational experience. Through using the interactive capabilities that Unreal Engine offers and my historical research, the application aims to bridge the gap between the past and present by allowing users to explore a historically authentic recreation of the town of Dunboyne circa 1909.

The aim of the project was to create an engaging and interactive educational experience that fosters a user's connection with history and gives an insight into how towns develop and grow over time. Through a commitment to provide a high level of attention to detail and authenticity, users can immerse themselves in the past and experience how the town of Dunboyne may have appeared to our ancestors.

At the core of the design ethos of Epoch Explorer was accessibility. The application was developed with the goal of being intuitive and user-friendly, ensuring that even those unfamiliar with video games could use the application quickly and efficiently. This was achieved through the minimalistic and high contrast design of the main menu, controls menu and the object information menu. All these design decisions were inspired by the core principles of Jakob Nielsen's 10 Usability Heuristics for User Interface Design.

By leveraging the power of Unreal Engine, the features of the application such as the map systems, information menus, and boundary systems were constructed via blueprint code. Unreal Engine's Megascan asset library and Nanite system proved invaluable in creating a visually pleasing, performant, and immersive recreation of Dunboyne.

While the Epoch Explorer application currently focuses on a single town, the systems implemented such as the map system, object information menus and photogrammetric scanning can be applied to any setting. This has the potential to be applied to a wide range of time periods and locations and has great potential for commercial viability. My hope is that Epoch Explorer sparks the curiosity and interest in history of its users and gives them a deeper appreciation for the heritage that surrounds us all in our day-to-day lives.

While I am pleased with how the application has developed, I recognise that there are still some limitations with the project in its current state - primarily the fact that many of the buildings in the application are 'whiteboxed'. However, this was a conscious decision based on the timeframe for the project. Most of the time on the project was spent developing the applications systems, user interfaces and environment. With more time and resources, the other buildings of the town can be recreated authentically, further improving the authenticity of the project. I look forward to implementing this in the future.

# 4.0 Further Development and Research

Given additional time and resources, there are a variety of aspects of the project that could be improved upon. The primary concern would be to populate the town with high-quality building assets that are recreated through references to historical photography or photogrammetric scanning. When this is achieved, the application will provide an even more immersive experience to its users.

To further improve the authenticity and immersion of the application, non-playable characters (NPC's) could be added to the applications environment. These NPCs would help to make the town more immersive by having characters go about their routines in the town, making it feel more alive. This could be achieved by using Unreal Engine's 'Metahuman' system that allows for photorealistic depictions of humans in an environment. This would require the creation of many art assets from scratch such as historically accurate clothes to ensure that the authenticity of the project is maintained.

Another aspect that could be introduced would be to then improve the user's ability to fully appreciate these high-quality recreations and scans from all angles in the environment – rather than just from a human's perspective as is available currently. This would be achieved by implementing a 'birds eye view' system. When a user presses the 'birds eye view' button in an object's information menu, they could be transferred into a new view around the object. This could be done using Unreal's camera system. The user would be anchored to the object they are interacting with, and they will be able to move around it via inputs that are mapped out using the blueprint system. All these features would be modular and will be able to be applied to any object in the world, ensuring that a user could view any object of interest from any angle they wish.

Once the recreation of Dunboyne is fully completed, I can then move onto recreating another historical town. The is such a wealth of different towns and settings that I could explore, so choosing the next one will be an interesting endeavour.

Mid-way through the development of the application in March 2024, I came across a project by the National Built Hertiage Service in collaboration with Meath County Council called 'Wonder Wander' [9] which had just been released. This is a historical walking tour that showcases noteworthy local buildings and provides information behind them. This proved to be a very useful resource when sourcing historical information about the buildings in my own project. At the time of writing, I have contacted the National Built Heritage Service querying their interest in attending my showcase of the Epoch Explorer application and I am awaiting their response. My hope is that if they are interested in the application, it may provide an avenue for it to be used in museums or other educational settings.

# 5.0 References


**[1]** Epic Games (2023) *Unreal Engine 5*. Available at: https://www.unrealengine.com/en-US/unreal-engine-5 (Accessed 13 December 2024)

**[2]** Blender (2023) *Blender 4.0*. Available at: https://www.blender.org (Accessed 13 December 2023)

**[3]** Polycam (2023) *Polycam*. Available at: https://poly.cam (Accessed 14 December 2023)

**[4]** JustGeekTechs (2023) *Unreal Mapbox Bridge*. Available at: https://terrain.justgeektechs.com/#/ (Accessed 15 December 2023)

**[5]** Geohive (2023) *Geohive map viewer*. Available at: https://webapps.geohive.ie/mapviewer/index.html (Accessed 16 December 2023)

**[6]** Nielsen Norman Group (1994) *10 Usability Heuristics for User Interface Design*. Available at: https://www.nngroup.com/articles/ten-usability-heuristics/ (Accessed 05 April 2024)

**[7]** Epic Games (2024) *Introduction to Blueprints*. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/introduction-to-blueprints-visual-scripting-in-unreal-engine?application_version=5.3 (Accessed 05 April 2024)

**[8]** Game Developer (2016) *White Boxing Your Game*. Available at: https://www.gamedeveloper.com/design/white-boxing-your-game (Accessed 05 April 2024)

**[9]** Facebook (2024) *Old Dunboyne Society*. Available at: https://www.facebook.com/profile.php?id=100070131743046 (Accessed 07 April 2024)

**[10]** Historical Picture Archive (2024) *Dunboyne*. Available at: https://www.historicalpicturearchive.com/picture-categories/dunboyne/ (Accessed 07 April 2024)

**[11]** National Built Heritage Service (2024) *Wonder Wander Dunboyne*. Available at: https://www.buildingsofireland.ie/app/uploads/2024/03/Wonder-Wander-Dunboyne.pdf (Accessed 07 April 2024)

**[12]** Meath County Council (2009*) Dunboyne Architectural Conservation Area Statement of Character*. Available at: https://www.meath.ie/system/files/media/file-uploads/2024-03/Dunboyne%20Architectural%20Conservation%20Area%20Character%20Statement.pdf (Accessed 07 April 2024)

**[13]** Paint.net (2024) *Paint.net Download*. Available at: https://www.getpaint.net/download.html (Accessed 09 April 2024)

**[14]** Narakeet (2024) *Narakeet*. Available at: https://www.narakeet.com/ (Accessed 27 April 2024)

**[15]** Valsogard Enterprise (2021) *Unreal Engine 5 | How to control texture UVs in Unreal Engine*. Available at: https://www.youtube.com/watch?v=iBSWo4tqbNA (Accessed 27 April 2024)

**[16]** Geeksforgeeks (2024) *Angle between Two Vectors Formula*. Available at: https://www.geeksforgeeks.org/angle-between-two-vectors-formula/ (Accessed April 29 2024)

**[17]** Unreal Engine (2024) *Megascans*. Available at: https://www.unrealengine.com/marketplace/en-US/content-cat/assets/megascans?count=20&sortBy=effectiveDate&sortDir=DESC&start=0 (Accessed 03 May 2024)

**[18]** Unreal Engine (2024) *Understanding Nanite – Unreal Engine 5's new virtualized geometry system*. Available at: https://www.unrealengine.com/en-US/blog/understanding-nanite---unreal-engine-5-s-new-virtualized-geometry-system (Accessed 04 May 2024)

## 6.0 Poster

SOFTWARE DEVELOPMENT

National College of Ireland

# Epoch Explorer

Mark Cummins, BSHCSD4

### Overview

An interactive learning tool that immerses users in virtual recreations of towns from the past. Users can explore a historically authentic recreation of the town of Dunboyne circa 1909 and learn about its history through exploration.

### Motivation

As someone with a passion for both video games and history, I wanted to share these passions with others. Epoch Explorer is designed to be used in an educational setting and is accessible to both young and old, gamer and non-gamer. By harnessing the innate desire for exploration, Epoch Explorer fosters a users learning through an immersive, interactive setting – giving them an insight into the past.

### How it Works

Epoch Explorer utilizes the power of Unreal Engine 5 and its 'Blueprint' coding system to recreate the town of Dunboyne. Drone photogrammetry is utilized in conjunction with Polycam and Blender to create realistic 3-D models of buildings. GeoHive and Unreal Mapbox Bridge are used to exactly recreate the landscape of Dunboyne in Unreal Engine 5.

### Technologies

polycam

GeoHive

blender

UNREAL ENGINE

Unreal Mapbox Bridge

GitHub

# 7.0 Appendices

## 7.1 Project Proposal

### 7.1.1 Objectives

This project sets out to create an interactive historical recreation of the town of Dunboyne circa the early 1900's. The aim of this project is to enable a user to navigate a historically accurate recreation of the town that is recreated using a combination of Blender and Unreal Engine 5.

The environment will be created using Unreal Engine 5 and will be based upon historical photos, maps, and surveys. The buildings and other objects will be created using Blender - with an emphasis on the technique of photogrammetry to produce more immersive and lifelike textures on objects.

The project will allow a user to learn about the history of the town through interactive user interface elements that will display information about the buildings and the other sights of the town. The primary objective of the project is to immerse the user in an authentic recreation of the town so that they gain a greater insight into how much the town has changed over the last 100 years. The project also aims to aid in the historical preservation of the town by ensuring that the recreation is as accurate, authentic, and informative as possible.

### 7.1.2 Background

I chose to undertake this project as I have a keen interest in history. I find learning about the lives of our ancestors and how much the world has changed from the present to be fascinating. When the requirement to pick a final year project arose - I had many different ideas, but none of them were focused on an area that I was passionate about. I reasoned that if I am to spend the next few months working on a project - I would prefer it to be in an area that I have a genuine interest in. Therefore, a historical recreation developed using a variety of software tools would be an interesting and challenging project to undertake.

In addition, there is much that I do not know about the history of my town - even though I have lived here for over 18 years. I felt that in the process of developing an accurate recreation of the town, I would need to gather a wealth of knowledge with regards to my town's history. The knowledge required to create this project would in-turn give me a greater insight into the history of my town and make me more appreciative as to how it has developed in the past century.

In terms of why I chose to create an interactive 3D environment - I have an avid interest in video games and their development. I find that as a medium, video games are uniquely suited when it comes to immersing a user in a historical environment. With film or photography, you are merely a passive observer. With video games that take place in historical settings, you can be an active participant and freely navigate a recreation of a variety of locations.

There are many video games that recreate historical settings such as the 'Assassins Creed' franchise and the designers of these games put meticulous effort into referencing the existing historical documentation to ensure that an authentic recreation is achieved. There is a version of the Assassins Creed games called 'Discovery Tours' that offer tours of the virtual worlds the developers created **[1]**. These tours are used in the field of education to try and engage students with history. I took inspiration for this project from these tours. In terms of both immersion and historical preservation, the medium of video games have a unique value. Outside of the realm of video games, there are also a variety of historical recreations made using 3D modelling technology. An example I came across is by 'Maxim Chichka' - who recreated the historic Schnoor neighbourhood in the town of Bremen, Germany using Unreal Engine 5 **[2]**. I took inspiration from this and ideally, I would hope that my project ends up at a similar state of fidelity and immersion.

What differentiates my project from others I have seen will be the use of photogrammetry and the interactive UI elements. I intend for my project to be a virtual museum exhibit where a user can walk up to a building, interact with a user interface prompt, and then learn about the building and its history. The examples I have found online using Unreal Engine 5 simply recreate an area - but do not implement any systems that allow a user to navigate it through a virtual avatar, such as Maxim Chichka's recreation. Those recreations I have found

such as the Assassins Creed Discovery Tours are not 1:1 recreations and take a variety of artistic liberties in their recreations of the worlds they have designed. For example - while they include key landmarks, these Tour worlds are scaled-down versions of locations. They also do not utilise the technology of photogrammetry to scan actual buildings into their games. My project aims to be a 1:1 recreation of the town of Dunboyne - scanning in actual buildings that were present during the period.

I intend to meet the objectives of the project by using Blender to accurately recreate the towns buildings that have changed drastically or are no longer there, referencing historical photography and maps to ensure authenticity. For buildings that remain unchanged from the period the project focuses on - I intend to use photogrammetry to scan the actual buildings, with some tweaking to undo minor changes such as new windows, weathering and extensions that have occurred over the years.

### 7.1.3 State of the Art

Since the beginning of the field of 3D modelling, there have been examples of those who have used the medium to recreate historical settings. However, my project will differ from others that already exist as it has never been done for my specific town and my project will be utilising the latest in 3D rendering technology. I would consider Unreal Engine 5 to be an emerging technology as it was only released 1 year ago and upon release it was heralded for its ability to create photo-realistic graphics using it's 'Nanite' system for geometry and it's 'Lumen' technology for lighting **[3]**. The use of photogrammetry, Nanite, Lumen and other state-of-the-art methods to create a historical recreation of Dunboyne has never been done before and I feel that it will be both a challenging and rewarding undertaking.

As mentioned in the previous section - during my research I have found examples of people recreating historic streets in Unreal Engine 5, but these recreations are just environments with no ability for the user to navigate the environment using a virtual avatar or interact with specific buildings and learn about them. My project will be unique in the sense that it will allow a user to walk up to a building, interact with it and learn about the history of it through UI elements - which will provide both an educational and immersive experience.

### 7.1.4 Technical Approach

My first step in the development of this project will be to primarily gain an understanding of the tools I will use to develop the project. I plan to complete a series of tutorials in both Blender and Unreal Engine 5 to gain a basic level of competency in these tools as I have never used them in the past.

In terms of the requirements and milestones of the project, there are several that I have identified at this stage. Firstly - I plan to source the required historical information, be they photographs, maps or surveys. I have already taken it upon myself to contact the local historical society as well as the owner of one of the oldest pubs in the town who said they have a variety of images they could provide me with.

Once I have the required source material, I will then start work on the basic layout of the town. I will create the terrain and layout of the town in a rough format using Unreal Engine 5, using the source photographs, maps, and surveys as a guideline. This step will involve recreating the roads and the general topography of the town's terrain. After the initial prototype of the terrain is done, I will then be able to get to work on creating assets such as the buildings and other objects in the town.

This milestone will involve creating rough approximations of the towns buildings in Blender to get a rough layout of the town.

After the general topography and layout of the towns buildings has been completed - I can then work on the milestone of implementing a user-controlled avatar that will allow navigation of the environment. After this task is completed, I will then be able to also implement the various user interface elements such as the ability for a user to interact with a building and learn about its history.

By this stage in development, I expect to be at the mid-point and have a rough prototype that will highlight the core elements of the project - a rough recreation of the town's terrain, it's buildings and the ability for a user to navigate the town.

Once the midpoint has passed, I then plan to start fleshing things out and increasing the fidelity of the town by adding more realistic ground clutter, increasing the count of objects and in general making the recreation of the town truer to life. As part of this task - I will begin the process of photogrammetry for the town's buildings. When the scans are finished - I will then be able to import them into Unreal Engine 5, completing the project.

### 7.1.5 Technical Details

There are a variety of technical requirements for my project. The two main technologies that will be utilised extensively will be 'Blender' and 'Unreal Engine 5'. Unreal Engine 5 will be used as the projects game engine. It will allow me to create the environmental terrain features, place and edit objects and allow for the creation of the project's mechanics such as player movement through an avatar and the interaction with UI elements. Blender will be utilised to create 3-D models from scratch and to refine the photogrammetric scans into usable models.

The photogrammetric scans of assets will be conducted either via drone or iPhone 12. The captured images will then be input into the program 'Polycam' **[4]** that can convert these images into 3-D models. These 3-D models can then be imported into Blender for cleanup or directly into Unreal Engine 5.

### 7.1.6 Special Resources Required

In terms of special resources, I already own a DSLR capable of taking high quality images, but even my mobile phone would suffice for this role. I will need to take photographs of historical buildings in the town - but doing so solely from ground-level would not suffice as I would be unable to capture all the details of a building.

Because of this fact, I will need to conduct aerial scans of buildings using drone photography to fulfil the level of fidelity I want to achieve. My father is a photographer and is also licensed to fly drones with the Irish Aviation Authority. He has indicated that he is willing to accompany me in taking drone photography of the historical buildings that were present in the town during the early 1900's.

I do not intend to undertake this stage of development until early next year - so in the meantime I will contact my local councillor to ask permission to take these photographs as a measure of courtesy.

In terms of ethics and legality - the process of capturing photographs of buildings using drone photography does not fall under GDPR as there is nothing under GDPR that prohibits taking photographs of a public place. The only caveat to this rule is if images of people are captured and published without their consent **[5][6]**. I will ensure that no images of people will be captured as I plan to conduct the aerial photography in the early morning when no other people are present.

Any images that may capture a person incidentally will not be published under any circumstances for this project. This is because the photographs will be used to create 3-D models of buildings ***only.*** Only images of buildings will be 'published' by being implemented in my 3-D environment - no images of people will be published during this project. GDPR only applies to images of people, not buildings.

Regardless of this fact - I also plan to contact the owner of each building I am planning to photograph beforehand as a measure of courtesy to the property owners.

### 7.1.7 Project Plan

As I have only started learning to use the tools required - I do not have an in-depth understanding as to how long exactly each stage of development may take, but I am giving myself around a month to work on major features and 2 weeks on smaller features. As work on the project progresses and I move closer towards the midpoint, the plan will become clearer but as of the time of writing of this proposal - this is an approximate plan for the project's development:

**October 12th - 31st**: Learn Blender and Unreal Engine 5 - complete tutorials to grasp the basics of these technologies

**November 1st - 15th**: Gather required sources - historical photographs, maps and surveys, local testimony

**November 15th - 30th**: Implement basic terrain layout of the town in Unreal Engine 5. Unreal Engine 5 has intuitive systems in place that make the creation of terrain straightforward. The basic terrain layout of the town should be completed during this stage of development.

**December 1st - 20th**: Populate town with basic buildings and objects in line with historical references - models will be placeholders but will serve to illustrate the general layout of the town for the midpoint. Implement user navigation and user interface elements.

**January 1st - 31st**: Conduct photogrammetric scans of key buildings - buildings that have been unaltered from the early 1900's to present. These will be identified in the historical photographs and maps. If they have not changed much in the past century - they can be scanned and included in the project.

**February 1st - 28th**: Conduct photogrammetric scans of textures that may be used on buildings no longer standing - For buildings that no longer exist (such as the old church) or that have changed significantly in the past century, these scans will be approximations of the original building materials to try and be as authentic as possible. For example, if it can be deduced that the building had a facade of a certain colour and material - a photogrammetric scan of a similar facade will take place.

**March 1st - 31st**: Convert photogrammetric scans into models using Blender and Unreal Engine 5. Once the scans have been captured - they can then be converted into 3D models in Blender and then imported into Unreal Engine 5.

**April 1st - 15th**: Finishing touches on terrain, buildings, objects - environment will be populated with clutter and fine-tuned to be more aesthetically pleasing and lifelike.

**April 15th - May 10th**: Final round of testing of the project. Testing will focus on the areas of the user's navigation, the performance of the project and the debugging of UI elements.

**May 12th**: Final submission.

### 7.1.8 Testing

In terms of testing, since my project is focusing on creating a 3-D environment that a user will navigate - I am not certain that I will be able to conduct system or unit tests. Because of this, my testing will most likely be manual testing to ensure that a user can navigate the environment without getting stuck on geometry or running into invisible obstructions etc.

The only aspect I can foresee being unit tested would be the UI features that display information about the history of a building or object in the town. These elements are controlled by systems within Unreal Engine 5 and could possibly be unit tested.

### 7.1.9 Proposal References

**[1]** Ubisoft (2023) *Discovery Tour by Ubisoft: Teacher Learning Resources*. Available at: https://www.ubisoft.com/en-gb/game/assassins-creed/discovery-tour (Accessed: 13 October 2023)

**[2]** Maxim Chichka (2023) *Schnoor made in Unreal Engine 5. Promenad*. Available at: https://www.youtube.com/watch?v=KrGSlGns9HA (Accessed: 13 October 2023)

**[3]** Unreal Engine (2023) *Nanite Virtualized Geometry in Unreal Engine | Unreal Engine 5.0 Documentation*. Available at: https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/ (Accessed: 15 October 2023)

**[4]** Polycam (2023) *Polycam - LiDAR & 3D Scanner for iPhone & Android*. Available at: https://poly.cam/ (Accessed: 16 October 2023)

**[5]** Data Protection Commission (2022) *Guidance on the Use of Drones*. Available at: https://www.dataprotection.ie/sites/default/files/uploads/2022-

05/Guidance%20on%20the%20use%20of%20drones%20-%20May%202022%20Final.pdf (Accessed: 20 October 2023)

**[6]** Citizens Information (2023) *Owning and operating drones in Ireland*. Available at: https://www.citizensinformation.ie/en/travel-and-recreation/sport-and-leisure/owning-and-operating-a-drone/ (Accessed: 20 October 2023)

## 7.2 Reflective Journals

### October

| **Student Name:** | Mark Cummins | **Course:** | BSHCSD4 |
|---|---|---|---|
| **Student Number:** | x20400634 | **Supervisor:** | Emer Thornbury |

Achievements:

This month I began work on the initial stages of my project. The initial few days of this month were spent brainstorming a variety of different ideas I had for the project. I spent the first part of the month researching the viability of these ideas and trying to weigh the pros and cons of each. After some deliberation and a discussion with Frances Sheridan - I settled on a project that focuses on the history of my town. This is area of study that I am quite passionate about, and I feel like it will be an interesting topic to base a project around. I recorded and submitted my project pitch in the form of a three-minute video, which was accepted. I also completed my project proposal document. This involved creating a general overview of my project - why I chose it, how I plan to complete it and the technical requirements of the project.

Reflections:

This month was crucial for my projects progress as it allowed me to settle on a topic for the project. Both the pitch video and the proposal document were good at challenging my ability to research, plan and then convey my ideas in both a verbal and written format. The deliberation and research I conducted in order to complete the proposal document has succeeded in giving me a tangible roadmap that I can follow to see the project to completion. It helped to flesh out my ideas further and gave me confidence that the project was indeed feasible and could be seen to fruition. The proposal also helped me to identify a variety of challenges that I will need to contend with. The most immediate challenge will be to create the terrain for the 3-D environment in Unreal Engine 5. This will act as the 'baseline' for the entire project and will be crucial to its future development and success.

Next Actions:

In order to address the challenge of creating the terrain for my project, I need to collect as many historical sources as possible - be they photographs, maps or surveys. I plan to do this by using a mix of publicly available resources, such as historical photography and maps that are available online, as well as sourcing privately owned photographs. I have already been in contact with a barman in the local pub 'Brady's' who has informed me that he has a variety of historical photographs from the time period in his possession that he could send onto me. These will surely aid me in overcoming the challenge of constructing an accurate recreation of the town's terrain. By using these historical photographs and maps, I will be able to ensure that the in-engine terrain is as accurate and true to life as possible.

## November

| | | | |
|---|---|---|---|
| **Student Name:** | Mark Cummins | **Course:** | BSHCSD4 |
| **Student Number:** | x20400634 | **Supervisor:** | Emer Thornbury |

Achievements:

This month I obtained many examples of historical photography of Dunboyne from online sources that will prove useful when creating buildings and other features of the town for my recreation. I spoke to the owner of the local pub 'Bradys' who said that unfortunately he could not find the photographs he thought he had in his possession. However, this month I applied for and received a 'readers ticket' with the National Library of Ireland. This ticket allows me to gain access a wealth of historical archives and photographic collections that will be very helpful in terms of obtaining references for my project.

I started work on the prototype for my project. I retrieved satellite heightmap data of the Dunboyne area using the 'Unreal Mapbox Bridge' application. This created a heightmap of a 3x3km area around Dunboyne that included topographical data such as rivers and hills. I then imported this heightmap into Unreal Engine 5, which created the landscape for my project. After doing this, I needed to be able to reference the historical 1909 ordnance survey in order to accurately recreate the Dunboyne of this time period. When Mapbox Bridge generated the heightmap, it also created a satellite map of the selected area. I used the same map co-ordinates for both Mapbox Bridge and the online ordnance survey map viewer, GeoHive. This meant the two maps were centred at the same point of reference.

I then took a screenshot of the ordnance survey map and using image editing software, I overlaid this screenshot onto the Mapbox Bridge satellite map, adjusting the scale so both lined up exactly. This final image was then imported into Unreal Engine 5 and overlaid onto the landscape as a texture. Using this texture as a guide, I was then able to start creating simple placeholder buildings in the exact place they stood in 1909. Every building in the 3x3km map was placed and this will serve as a useful reference aid when placing the finalized assets.

After some deliberation with my supervisor, I settled on including Virtual Reality (VR) to interface with my project. A user will be able to navigate the town while using a VR headset. This will ensure that the project is as immersive as possible. Due to the large size of my map and the time constraints of this project, I separated the map into grid references that I can prioritize. Due to the level of fidelity I aim to achieve, I plan to focus on the town centre for this project - but future development can be done on the surrounding areas.

Reflections:

I succeeded in sourcing a variety of historical information and photography. I was also able to create a prototype of the environment that will serve as a basis for future development. However, there are a few challenges remaining that I need to solve before the midpoint deadline on the 20th of December. I need to implement features such as a main menu. I need to add the ability for a user to walk up to a building and interact with it in order to show a user interface widget that will display information about its history. I still need to scan a building using photogrammetry for the midpoint in order to showcase the photogrammetric aspect of my project. To demonstrate the use of Blender for the midpoint, I need to create a building from scratch and then texture it manually and import it into Unreal Engine 5.

I also need to implement the ability for a user to view the recreation using a VR headset instead of a monitor. Because reading text in VR can be challenging due to the resolution limitations, I also plan to include an audio narration of any user interface widgets that display building information.

Next Actions:

To address the need for an example of photogrammetry in the project before the midpoint, I got in contact with Fr. O'Connor of the Dunboyne and Kilbride parish on November 21st. I explained the project to him and requested to scan the Parochial House building in the town centre using drone photogrammetry, as it has

remained largely unchanged since 1909. He accepted, and I will be able to scan the building between the 8th and 10th of December. Ideally, I would need overcast weather for this scan to ensure that there are no shadows or other undesired artifacts on the final asset. Should the weather not be ideal, I plan to scan in the building anyway to serve as a proof of concept for the midpoint - and then re-scan it before the final project submission in May 2024 when there are better weather conditions present.

To demonstrate the use of Blender for the midpoint, I plan to recreate the old Roman Catholic church that is no longer standing. This will be done manually in Blender using photographic sources as a reference. I will then manually texture this building and import it into Unreal Engine 5. This technique of manually modelling buildings will need to happen for multiple buildings that no longer exist in the town that can't be scanned using photogrammetry, but these will be developed after the midpoint.

From my research into the Unreal Engine 5 documentation, I believe that implementing a main menu, the ability for a user to interface with the project using a VR headset and adding in the ability for a user to interact with a widget to display information about a building will be achievable before the December 20th midpoint deadline.

## December

| **Student Name:** | Mark Cummins | **Course:** | BSHCSD4 |
|---|---|---|---|
| **Student Number:** | x20400634 | **Supervisor:** | Emer Thornbury |

Achievements:

This month I continued implementing a variety of features in my project to have it at a presentable state for the midpoint deadline.

I was able to conduct a photogrammetric scan of a building and import it into my project, which allowed me to showcase the utility of photogrammetry and also allowed me to practice the techniques required to conduct this process.

I completed my midpoint report, presentation and demo this month, which served as a valuable reflection and insight into the future of the project.

After attempting to implement VR into my project, I had to make the decision to not go ahead with this plan. I had converted my project into a VR compatible one in Unreal Engine. I had set up a control scheme and a few prototype Graphical User Interfaces (GUI's). However, performance was a major issue. Because of the scale of my project, the number of assets and the graphical fidelity of these assets - VR was not a viable technology to introduce into the project. There were major performance issues that made the application unplayable - such as stuttering, input lag and crashing. Because of these issues, I decided to instead focus my efforts on having a project that a user could interact with using a mouse and keyboard, or a controller.

Reflections:

This month I made great progress in terms of having a working prototype for my project. I was able to get the GUI for the application mostly implemented, I conducted a photogrammetric scan of a building and imported it into the project - which gave me valuable experience in both photogrammetry and the use of Blender. I was also able to get the 'Object Prompt' system of the project implemented.

There are however many challenges still remaining. I still need to implement the 'Object Information Widget' system that displays an informative menu that gives a user information about a buildings history. I also need to implement the 'Birds Eye View' mode that will allow a user to enter a view that allows them to rotate around an object of interest in order to view it from every angle. A map system for the project and an out-of-bounds system will also need to be implemented.

Aside from these functional requirements, I also need to enrich the projects environment to make it more photo-realistic and immersive. This will require the creation of a large number of assets in Blender - such as buildings and other objects of interest.

In addition, I will need to texture the landscape of the project with realistic textures in order to immerse the user in their surroundings.

Next Actions:

While I have listed all the outstanding challenges that remain in the project, I will need to budget my remaining time on the project wisely. If I spread myself too thin by trying to work on small parts of every feature in the project, I may not get any feature fully implemented.

To ensure this undesirable outcome doesn't come to fruition, I aim to complete the most important and functionally complex aspects of the project first. I have determined that the 'Object Information Widget' system will require the largest amount of time and effort to complete, therefore I plan to largely devote myself to getting this implemented in January.

To achieve this goal, I will need to examine the Unreal Engine documentation to gain a greater understanding of the functions available to me in the 'blueprint' system. The Object Information Widget will require many blueprints to implement, so I will need to study and experiment in Unreal Engine to overcome this challenge.

## January

| | | | |
|---|---|---|---|
| **Student Name:** | Mark Cummins | **Course:** | BSHCSD4 |
| **Student Number:** | x20400634 | **Supervisor:** | Emer Thornbury |

Achievements:

This month I made progress on implementing the 'Object Information Widget' system for my project. I created blueprints on the player character that enables the user to press the 'E' key to interact with an object. By using a for-each loop, the system will check if the player character is within the boundaries of an object that is interactable. If this is the case, the object information widget will be displayed on the user's screen with information about the object.

Reflections:

I succeeded in implementing the core functionality of a major feature in the project this month. I still need to make the widget that appears more user friendly and visually appealing – as currently it is just in a placeholder state. Once I flesh out this feature more by making it more interactable and adding visual flair, it will be in a better state.

In February, I will need to work on the 'Birds eye view' system that will allow the user to enter a mode where they can view an object from any angle. This mode will be accessed from the object information widget that appears on their screen, and I anticipate it will involve much time and effort to see to fruition.

Next Actions:

To address these challenges, I plan to flesh out the object information system more to make it easier to navigate and more visually pleasing. Once this is done, I can then add a button that allows a user to enter the birds eye view mode. Having done this, I will then need to implement the code that handles the birds eye view system. For this, I will need to consult the Unreal Documentation and come up with a way to change the players perspective from their character to a controllable camera that orbits an object. This will involve creating new camera objects, editing the control inputs a player has at their disposal, and implementing a wide array of new blueprints.

## February

| **Student Name:** | Mark Cummins | **Course:** | BSHCSD4 |
|---|---|---|---|
| **Student Number:** | x20400634 | **Supervisor:** | Emer Thornbury |

Achievements:

This month I began implementing the birds eye view system for the project a mini-map system. The birds eye view system proved a challenge as due to the nature of Unreal Engine, having a user switch to a different character is doable, but because the input system for the bird's eye view character will be different to the default first-person character – challenges arose. After troubleshooting these issues, I decided to move the birds eye view system down in priority and worked on the mini-map system instead.

This mini-map allows a player to better navigate the environment and aids them in understanding whereabouts in the town they are at any given time. To implement this system, I needed to create a variety of new assets and blueprints. Firstly, I had to create the actual map assets using image editing software and import them into the game engine. Once this was done, I created blueprint logic that retrieved the users X and Y co-ordinates and then moved the map to-scale to reflect the user's movement on the mini-map.

Reflections:

I succeeded in getting the mini-map system mostly implemented. This involved a lot of tweaking of parameters to ensure that the map was to the right scale and moved in-line with the user's navigation of the environment. The mini-map currently moves in the x and y axis correctly, but I am currently tweaking how it moves in the z axis so that it rotates when the users character rotates.

I will keep the birds eye view system on a low priority for now as it will prove a challenge to implement this system in the way that I initially envisioned. I will work on higher priority features in the meantime.

Next Actions:

To address the challenges, I will continue to experiment with blueprint parameters to ensure that the rotation of the mini-map on the z axis is to-scale and in line with the user's mouse movements. This will require a lot of tweaking and experimentation, but I feel this aspect of the project is almost completed. Once this is done, I will then work on implementing the main map system that allows a user to place waypoints in the world that will allow them to better navigate to a point of interest.

## March

| **Student Name:** | Mark Cummins | **Course:** | BSHCSD4 |
|---|---|---|---|
| **Student Number:** | x20400634 | **Supervisor:** | Emer Thornbury |

Achievements:

This month I continued tweaking the mini map system that I began to implement last month. I was able to tweak the scaling and parameters of the minimap such that it now accurately reflects the players movement in the environment, aiding them in their navigation of the town. I also began work on the main map system, where a user can press a key and display a large, detailed map of the town that will also aid them in navigation. This map will also the user to place waypoints that can guide them towards objects of interest. These waypoints will be reflected in the world and on the minimap.

Reflections:

I succeeded in fine-tuning the mini map system and got a large amount of the work done on the main map system. This involved a lot of tweaking of parameters and experimenting in order to come up with a satisfactory implementation of the map system. I still need to complete the waypoint system, which I am currently working on.

Next Actions:

I will need to complete the waypoint system in order to have a fully realised map system in the application. Once this is implemented fully, I can then move onto implementing the out-of-bounds system and also tweak the user interface of the application, as it is currently in a placeholder state.

## April

| | | | |
|---|---|---|---|
| **Student Name:** | Mark Cummins | **Course:** | BSHCSD4 |
| **Student Number:** | x20400634 | **Supervisor:** | Emer Thornbury |

Achievements:

This month I succeeded in implementing the main map and waypoint system. This allows a user to better navigate the environment by giving them points of reference on their minimap and main map for them to travel to. This feature took some time as I needed to implement a system that would calculate the angle between two vectors in order to allow the waypoint positions on the map to update with relation to the player's position. I also added the functionality of a saving and loading system, and the out of bounds system. These systems will let a player resume the game from their save point and will also prevent a user from leaving the boundary of the playable area.

Reflections:

I succeeded in implementing two major systems, the main map waypoints and the out of bounds system. I also implemented a minor system of saving a player's progress. I still need to fix some bugs with these systems and fine tune them to make them more user friendly. I now need to add more art assets and textures to my environment to achieve a more realistic portrayal of the town.

Next Actions:

I plan to add a variety of art assets to the project such as trees, walls, and other objects into the environment. This will make the environment seem more realistic and true-to-life. At every stage of this process, I will consult my references such as historical ordnance surveys and historical photography to ensure that my recreation of the town is as authentic as possible, in-line with the goals of the project. I will also conduct testing of my project and introduce a series of bug fixes.